INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

ESTG

ASSESSMENT
ANALYSIS OF CRYPTOGRAPHIC TOOLS

João Pedro Inês Portela Rosa

ASSESSMENT ANALYSIS OF CRYPTOGRAPHIC TOOLS
João Pedro Inês Portela Rosa

2026

Escola Superior de Tecnologia e Gestão

Mestrado em Cibersegurança
Master in Cybersecurity

Dissertation: Assessment Analysis of Cryptographic Tools

a dissertation report authored by

João Rosa

supervised by

Prof. Luís Barreto

7 January, 2026

ipvc estg

Abstract

This thesis presents an integrated assessment that combines a practical evaluation of cryptographic tools with a comprehensive literature analysis of the field. The literature review traces the evolution of cryptographic research, identifies emerging trends, and highlights the most influential contributions shaping the discipline. Given the accelerating growth of cryptography driven by escalating cybersecurity threats, this dual analysis provides meaningful insight into how the field has developed and where it is heading, reinforcing its essential role in modern security architectures. In parallel, the practical assessment examines how widely used cryptographic tools behave across different configuration scenarios, highlighting their strengths, limitations, and operational effectiveness, with a particular focus on their ability to support secure communication and robust information protection.

Keywords: Literature, Cybersecurity, Cryptographic Tools.

Resumo

Esta tese apresenta uma avaliação integrada que combina uma avaliação prática de ferramentas criptográficas com uma análise bibliográfica abrangente da área. A revisão bibliográfica traça a evolução da pesquisa criptográfica, identifica tendências emergentes e destaca as contribuições mais influentes que moldam a disciplina. Dado o crescimento acelerado da criptografia impulsionado pelas crescentes ameaças à segurança cibernética, esta análise dupla fornece uma visão significativa sobre como o campo se desenvolveu e para onde está a caminhar, reforçando o seu papel essencial nas arquiteturas de segurança modernas. Paralelamente, a avaliação prática examina como as ferramentas criptográficas amplamente utilizadas se comportam em diferentes cenários de configuração, destacando os seus pontos fortes, limitações e eficácia operacional, com um foco particular na sua capacidade de apoiar comunicações seguras e proteção robusta de informações.

Palavras-chave: Literatura, Cibersegurança, Ferramentas Criptográficas.

Aknowledgements

I express my sincere appreciation to Professor Luís Barreto for entrusting me with the opportunity to commence this thesis journey and for guiding me through the world of research and academia with expertise. Your patience and willingness to guide me have been simply incredible, wich has been invaluable in shaping my academic pursuits. I would also like to thank and dedicate this thesis to myself for believing in myself and never giving up in the face of the difficulties encountered along the way. Finally, I would like to wholeheartedly dedicate this thesis to my mother, who, from day one, gave me all the support and confidence I needed to never give up and see this thesis through to the end.

Contents

Acronyms

AES Advanced Encryption Standard

AI Artificial Intelligence

CA Certificate Authority

CSR Certificate Signing Request

CVE Common Vulnerabilities and Exposures

ECC Elliptic Curve Cryptography

GnuPG GNU Privacy Guard

GPG GNU Privacy Guard

GUI Graphical User Interface

MCyber Master in Cybersecurity

OpenSSL Open Secure Sockets Layer

PoC Proof of Concept

RSA Rivest–Shamir–Adleman

SHA Secure Hash Algorithm

SLA Service Level Agreement

SSL Secure Sockets Layer

TLS Transport Layer Security

VeraCrypt Open-source Disk Encryption Software

WN Wireless Network

Chapter 1

Introduction

1.1 Context

In today's digital world, cryptographic tools play a crucial role in protecting sensitive information from growing cyber threats. As online systems become more interconnected and attackers more sophisticated, the need for strong, reliable encryption has never been greater. From safeguarding financial transactions to securing healthcare data and personal communications, cryptography forms the "backbone" of modern digital security.

However, with the rapid expansion of cryptographic research, new tools, techniques, and algorithms are constantly being developed. While this growth is promising, it also creates a challenge: it's becoming increasingly difficult to evaluate which tools are most effective, secure, and suitable for specific use cases. The field now includes everything from classic symmetric and asymmetric algorithms to newer technologies like homomorphic encryption, blockchain, and post-quantum cryptography.

This thesis seeks to bring some clarity to this complex and fast-evolving landscape. It offers a structured analysis of cryptographic tools, exploring how they perform in terms of bad configurations and good configurations, security, efficiency, and overall effectiveness. Alongside this technical evaluation, the study also provides a broader overview of cryptographic research itself, using literature methods to map out major trends, identify influential work, and highlight areas where further innovation is needed, especially as we prepare for future challenges like quantum computing.

1.2 Problem Statement and Motivation

As cyberattacks become more advanced and targeted, ensuring the security of digital systems is more critical than ever. Cryptography lies at the heart of digital defense, but keeping up with the pace of research in this area is becoming increasingly difficult. With thousands of papers, tools, and protocols being published or introduced each year, it's hard to know where to focus efforts—what's truly impactful, what's outdated, and where the gaps still lie.

This overwhelming volume of information makes it challenging for both researchers and practitioners to get a clear picture of how the field is evolving. Without structured analysis, we risk missing important breakthroughs or continuing to invest in tools that may no longer be effective.

This study is motivated by the need to better understand the landscape of cryptographic research and tools. By combining technical analysis with a literature perspective, the goal is to highlight which areas are most active, which technologies are gaining traction, and where future efforts should be directed. The research also aims to support global cybersecurity efforts by identifying patterns of collaboration, innovation hotspots, and opportunities for improvement. Ultimately, the motivation is to contribute to the development of more secure, practical, and future-ready cryptographic solutions.

1.3 Objectives

The objectives of this thesis are to conduct an assessment analysis of cryptographic tools. The study aims to identify key research trends, influential publications, and leading authors while mapping the evolution of cryptographic research and emerging areas. Literature analysis will be conducted to quantify the impact of the investigation, classify significant contributions, and provide a structured view of the cryptographic research landscape. It will also evaluate the effectiveness and security of cryptographic tools, providing insights into their strengths and weaknesses. In the long term, the objective is to help guide future research and development by offering meaningful insights into both the technical performance of cryptographic tools and the structure of the research community behind them. With better understanding, we can work toward stronger, more adaptive

cryptographic systems that are ready to face the cybersecurity challenges of tomorrow.

1.4 Document organization

This dissertation is structured into six main chapters, each building upon the previous to provide a clear, logical progression from foundational concepts to practical analysis and final conclusions. The chapters are organized as follows:

1.4.1 Chapter 1 – Introduction

This chapter establishes the context and motivation for the study. It explains the growing relevance of cryptographic tools in modern cybersecurity, presents the problem statement, and outlines the primary objectives of the research. It also introduces the rationale for combining literature analysis with practical assessment to evaluate cryptographic tools under different configuration scenarios.

1.4.2 Chapter 2 – Literature Review

Chapter 2 provides a comprehensive review of academic work related to cryptographic algorithms, evaluation methodologies, and vulnerability assessment. The literature is categorized into thematic areas such as formal analysis, performance benchmarking, lightweight and IoT-oriented cryptography, side-channel attacks, post-quantum security, and cryptographic misuse detection. This review forms the theoretical basis for the practical experiments conducted later in the dissertation and highlights key trends and challenges within the field.

1.4.3 Chapter 3 – Cryptographic Tools

This chapter presents the three tools selected for practical evaluation: OpenSSL, VeraCrypt, and GnuPG. It explains why these tools were chosen, describes their main functionalities, and details the aspects of security that will be tested. The chapter also introduces the methodology used to construct secure and insecure scenarios, preparing the reader for the analysis that follows.

1.4.4 Chapter 4 – Assessment of Cryptographic Tools

Chapter 4 contains the core practical component of the thesis. It provides a step-by-step assessment of each tool under both well-configured and poorly configured environments:

- OpenSSL: Certificates, server configurations, and attacks such as Heartbleed.
- VeraCrypt: Volume creation, mounting, and brute-force attempts using John the Ripper.
- GnuPG: Key generation, encryption/decryption, passphrase strength testing, and offline password-recovery attacks.

Through these experiments, the chapter highlights how configuration choices directly influence the security, resilience, and exploitability of cryptographic systems.

1.4.5 Chapter 5 – Discussion

This chapter analyzes and interprets the results obtained in Chapter 4. It discusses the security implications of misconfiguration, the persistence of certain vulnerabilities, and the lessons learned from the comparative assessment of the three tools. It also explores broader impacts on security policy and outlines technical recommendations to mitigate the identified weaknesses.

1.4.6 Chapter 6 – Conclusion and Future Work

The final chapter summarizes the main findings of the study and reflects on the contributions made by combining literature analysis with practical experimentation. It also identifies limitations and proposes directions for future research, including improvements in automated configuration analysis, enhanced vulnerability-testing frameworks, and further exploration of post-quantum cryptographic tools.

Chapter 2

Literature review

The literature on cryptographic tools, methods, and evaluation frameworks has expanded significantly over the past two decades, reflecting the increasing complexity of digital systems and the growing demand for secure communication mechanisms. Early research between **2003 and 2010** primarily focused on foundational cryptographic properties, protocol security, and algorithmic comparisons. These studies established the conceptual basis for understanding authentication, cipher strength, and secure protocol behavior.

From 2011 onward, the research landscape began to diversify, incorporating lightweight cryptography, embedded-system implementations, and energy–efficiency considerations. This shift corresponded with the widespread adoption of mobile and Internet-of-Things (IoT) devices, which prompted new constraints on cryptographic design and motivated the development of low-power algorithms, optimized hashing methods, and modular exponentiation enhancements.

Between 2015 and 2024, the literature trends indicate a notable intensification of research output, with particular growth in topics such as side-channel resistance, post-quantum cryptography, cryptographic misuse detection through static and dynamic analysis, and secure data-storage infrastructures. The emergence of automated evaluation tools, machine-learning-based vulnerability detection systems, and formal verification languages further illustrates the maturing intersection between software engineering and applied cryptography.

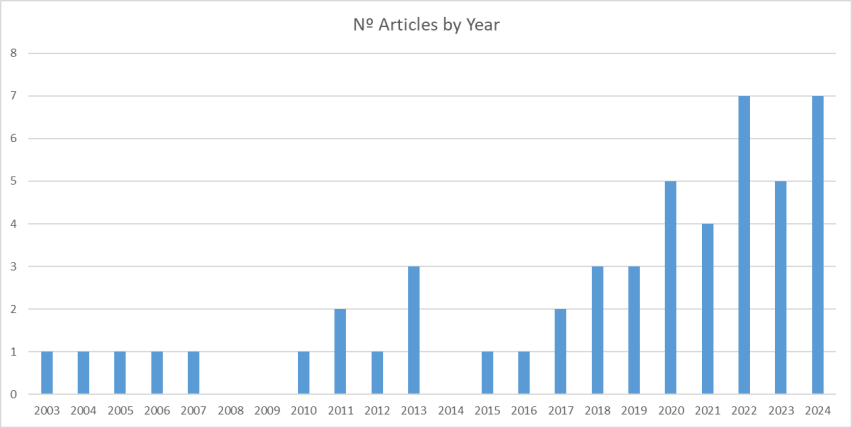

Figure 2.1: Nº Articles by Year

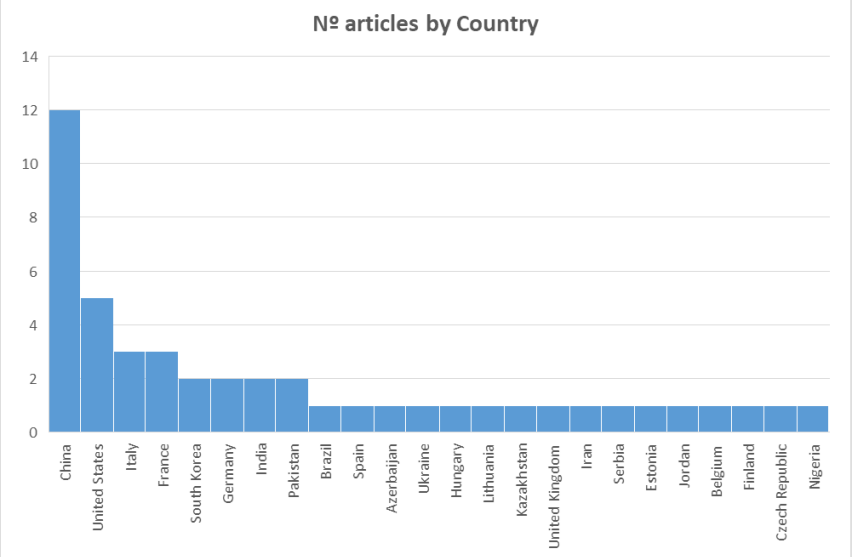

Figure 2.2: Nº Articles by Country

Figures 2.1 and 2.2 summarize the temporal and geographical distribution of the 50 selected articles. The annual publication trend demonstrates a consistent increase, particularly after 2018, highlighting the rapid development of advanced cryptographic analysis techniques. Country-level distribution reveals a strong concentration of contributions from China, followed by the United States and several European and Asian countries, indicating a globally distributed yet uneven research landscape.

Together, these literature patterns contextualize the subsequent thematic analysis and provide the empirical foundation for understanding how cryptographic research has evolved in response to technological, security, and regulatory pressures.

2.1 Formal Analysis and Authentication Protocols (2003–2010)

Research in formal verification and authentication protocols has long aimed to ensure logical soundness and provable security properties within cryptographic systems. This line of work provided the theoretical and methodological foundations for subsequent empirical analyses of real-world cryptographic tools and implementations.

Riccardo Focardi (2003) – "A Comparison of Three Authentication Properties." Research on authentication protocol verification has traditionally emphasized the formal modeling of correctness properties to assess protocol soundness [15]. This work contributed a unified analytical framework for authentication evaluation using process algebra, reinforcing theoretical underpinnings for later empirical assessments of protocol robustness. Such conceptual models provide the methodological basis for analyzing weaknesses and deployment issues in modern cryptographic tools, as explored in the present thesis.

I. Bertolotti (2005) – "Automatic Detection of Attacks on Cryptographic Protocols: A Case Study." Automated verification and detection mechanisms have long been central to protocol security analysis [5]. This study developed a verification system capable of identifying cryptographic attacks, highlighting the role of automation in improving reliability and repeatability of assessments. The emphasis on automatic detection aligns with the current thesis's recommendation for integrating vulnerability scanning and formal analysis in open-source cryptographic environments.

Xiao Meihua (2010) – "On Formal Analysis of Cryptographic Protocols and Supporting Tool." Formal analysis tools have played a critical role in verifying the logical correctness of cryptographic protocols [28]. This study advanced tool-assisted formal verification approaches that enable consistent reasoning about protocol safety properties. Such work informs the theoretical background of this thesis, which investigates the reliability of practical deployments against established formal models.

2.2 Performance Evaluation and Benchmarking of Cryptographic Algorithms (2004–2024)

Research on the performance benchmarking of cryptographic algorithms has evolved from early throughput and energy-efficiency analyses to systematic evaluations considering reproducibility, scalability, and platform heterogeneity. The studies grouped under this theme collectively highlight how algorithmic optimization, measurement accuracy, and reproducible testing have shaped modern cryptographic tool assessment practices.

Bringel Filho (2004) – "PEARL: A Performance Evaluator of Cryptographic Algorithms for Mobile Devices." Early performance evaluations of cryptographic algorithms focused on throughput and computational efficiency under constrained hardware environments [14]. By introducing the PEARL tool, this research established one of the first systematic approaches to benchmarking cryptographic primitives across mobile platforms. This methodological orientation directly parallels the experimental component of this thesis, which assesses performance and configuration behavior in open-source encryption tools.

B. Damjanović (2013) – "Performance Evaluation of AES Algorithm Under Linux Operating System." Empirical studies on algorithmic performance under specific operating systems have become foundational to cryptographic benchmarking [10]. This work evaluated AES[31] execution on Linux platforms, providing measurable insights into efficiency and processing overhead. The methodology reinforces the practical orientation of the present thesis, which also investigates performance variations across open-source cryptographic tools.

Jevgenijus Toldinas (2011) – "Energy Efficiency Comparison with Cipher Strength of AES and Rijndael Cryptographic Algorithms in Mobile Devices." Studies comparing cipher efficiency and energy consumption have provided empirical insights into algorithm selection for mobile systems [53]. By correlating cryptographic strength with energy performance, this work emphasized measurable trade-offs between robustness and resource utilization. The approach parallels the benchmarking methodology adopted in this thesis for assessing cryptographic tool efficiency.

A. Ometov (2021) – "A Comprehensive and Reproducible Comparison of Cryptographic Primitives Execution on Android Devices." Reproducibility in cryptographic benchmarking has become increasingly important for validating experimental results across heterogeneous environments [34]. This work provided a comparative assessment of cryptographic primitives on Android devices, establishing standardized procedures for measuring performance and reliability. The methodological rigor of this study parallels the thesis's experimental approach, which also emphasizes reproducible testing of open-source encryption tools.

Antonio Francesco Gentile (2024) – "A Performance Analysis of Security Protocols for Distributed Measurement Systems Based on Internet of Things With Constrained Hardware and Open Source Infrastructures." Performance analysis in IoT-based distributed systems remains central to understanding the balance between computational efficiency and communication security [17]. This study examined multiple security protocols within open-source infrastructures and constrained hardware settings, identifying performance bottlenecks and scalability limitations. Its conclusions provide empirical grounding for the thesis's investigation of real-world tool performance and deployment efficiency.

2.3 Lightweight and IoT-Oriented Cryptography (2011–2024)

Research on lightweight cryptography and IoT security has increasingly emphasized the trade-off between computational efficiency and adequate protection for constrained environments. These works collectively explore how lightweight algorithms, certificateless schemes, and energy-aware cryptographic designs enable secure communication in resource-limited and distributed systems.

Dae Hyun Yum (2011) – "Lightweight One-Time Signature for Short Messages." Lightweight cryptographic signatures have become essential for short-message and IoT communications [58]. The study introduced a one-time signature scheme optimized for efficiency, anticipating the increasing need for low-cost, secure primitives. This direction relates closely to the literature findings of this thesis, which identify lightweight cryptography as a growing research frontier in tool design and practical deployment.

Moreno Ambrosin (2016) – "On the Feasibility of Attribute-Based Encryption on Internet of Things Devices." The feasibility of advanced cryptographic schemes within constrained hardware environments has been increasingly examined [3]. This study evaluated attribute-based encryption for IoT devices, highlighting implementation challenges and computational limitations. The work complements the literature trend identified in the thesis, wherein lightweight and context-aware encryption methods gain prominence.

Sumit Singh Dhanda (2020) – "Lightweight Cryptography: A Solution to Secure IoT." The proliferation of IoT devices has driven the need for lightweight cryptographic solutions capable of maintaining adequate security with minimal computational overhead [11]. This study outlined the theoretical framework and implementation strategies for lightweight cryptography, framing the paradigm shift toward efficiency-driven security models. The conclusions echo the trends identified in the thesis's bibliometric analysis regarding the dominance of IoT-oriented cryptographic research.

Saddam Hussain (2021) – "A Lightweight and Provable Secure Identity-Based Generalized Proxy Signcryption (IBGPS) Scheme for Industrial Internet of Things (IIoT)." Identity-based cryptography has evolved as a response to the management challenges of traditional public-key infrastructures [20]. This article developed a lightweight signcryption scheme tailored for industrial IoT, emphasizing provable security and low computational demand. The research reflects ongoing efforts, also identified in this thesis, to integrate provable-security concepts into practical, resource-limited environments.

Deng Xiang (2022) – "A Secure and Efficient Certificateless Signature Scheme for Internet of Things." Certificateless cryptography has emerged as a promising solution to mitigate key-management complexity in IoT ecosystems [57]. This study introduced a signature scheme that eliminates dependence on traditional certificate authorities while maintaining computational efficiency. The findings complement the thesis's discussion of lightweight and scalable security mechanisms in distributed and resource-constrained environments.

2.4 Side-Channel and Implementation Security (2018–2024)

The study of side-channel attacks and implementation-level vulnerabilities has evolved from manual signal analysis to automated frameworks capable of early-stage leakage detection. Research within this domain focuses on ensuring that cryptographic algorithms remain resilient against physical and software-based exploitation, forming a crucial intersection between theory and implementation security.

Houssem Maghrebi (2018) – "On the Use of Independent Component Analysis to Denoise Side-Channel Measurements." The mitigation of side-channel noise through statistical and signal-processing techniques has become vital to implementation-level security research [26]. This paper applied independent component analysis to enhance side-channel attack detection accuracy. Although distinct from software configuration vulnerabilities, this perspective enriches the broader context of security assessment frameworks referenced in the thesis.

Danilo Šijačić (2020) – "Towards Efficient and Automated Side-Channel Evaluations at Design Time." Automation in security evaluation has been increasingly applied to the early stages of system design [47]. This study presented a framework for preemptive side-channel analysis, reducing the likelihood of vulnerabilities emerging in deployed implementations. The emphasis on early verification and automation resonates with the thesis's advocacy for systematic testing and evaluation of cryptographic tools prior to deployment.

Ju-Hwan Kim (2020) – "SIV: Raise the Correlation of Second-Order Correlation Power Analysis to 1.00." Advancements in side-channel attack techniques have led to increasingly sophisticated analytical models [24]. The study explored methods for improving correlation accuracy in second-order power analysis, emphasizing the need for resilient countermeasures in algorithmic implementations. This focus extends the thesis's examination of operational vulnerabilities beyond software-level misconfigurations to include hardware-based attack vectors.

David Pokorný (2022) – "Equivalent Keys: Side-Channel Countermeasure for Post-Quantum Multivariate Quadratic Signatures." Post-quantum cryptographic research has increasingly addressed the resilience of new schemes against side-channel exploitation [38]. This article presented an equivalent-keys countermeasure to enhance the robustness of multivariate quadratic signatures, contributing to the transition toward secure post-quantum implementations. The study supports the thesis's literature observation that post-quantum cryptography and implementation security are converging research trends.

Yaru Wang (2023) – "A Survey of Side-Channel Leakage Assessment." Comprehensive surveys on side-channel assessment have synthesized existing methodologies to establish benchmarks for evaluating implementation security [56]. This study consolidated findings across multiple attack vectors and mitigation strategies, providing a structured overview of the field. Its systematic approach supports the thesis's theoretical background on evaluation techniques applicable to software-based cryptographic systems.

Tristen Teague (2023) – "Towards Cloud-Based Infrastructure for Post-Quantum Cryptography Side-Channel Attack Analysis." The convergence of cloud computing and post-quantum cryptography has generated new paradigms for scalable security testing [50]. This work proposed a distributed infrastructure for evaluating side-channel resistance of post-quantum algorithms, enabling large-scale automated analysis. The framework's emphasis on scalability and automation reflects core principles echoed in the thesis's recommendations for systematic cryptographic evaluation environments.

2.5 Formal Tools, API Misuse Detection, and Automation (2017–2024)

Research on formal tools and automated analysis for cryptographic security has increasingly focused on detecting implementation flaws, API misuses, and logical inconsistencies within software systems. The integration of automation, data-flow analysis, and machine learning models has significantly improved reproducibility and accuracy in identifying vulnerabilities, forming a key foundation for secure cryptographic tool development.

Alexandre Braga (2017) – "Practical Evaluation of Static Analysis Tools for Cryptography: Benchmarking Method and Case Study." The development of automated tools for detecting cryptographic misconfigurations represents a key evolution in software assurance research [7]. This paper established benchmarking criteria for evaluating static analysis tools applied to cryptographic codebases. The emphasis on automation and reproducibility aligns closely with the thesis's objective of integrating systematic testing into the evaluation of open-source cryptographic environments.

Cong Sun (2021) – "CryptoEval: Evaluating the Risk of Cryptographic Misuses in Android Apps with Data-Flow Analysis." Empirical analyses of cryptographic misuse in software applications have underscored the prevalence of insecure implementation practices [48]. CryptoEval introduced a data-flow-based framework to identify and classify such misuses in Android applications. This tool-oriented methodology aligns closely with the thesis's practical assessment of how open-source encryption systems may embed similar configuration or API misuse risks.

Sazzadur Rahaman (2022) – "From Theory to Code: Identifying Logical Flaws in Cryptographic Implementations in C/C++." The translation of cryptographic theory into code often introduces logical inconsistencies that compromise system integrity [42]. This research systematically identified such flaws in C/C++ implementations, demonstrating how programming errors can weaken cryptographic assurance. The analytical approach aligns with the thesis's practical exploration of misconfigurations and vulnerabilities in open-source cryptographic software.

Sharmin Afrose (2023) – "Evaluation of Static Vulnerability Detection Tools With Java Cryptographic API Benchmarks." Evaluations of static analysis tools for cryptographic codebases have sought to determine their accuracy in detecting misuse and vulnerability patterns [1]. This research benchmarked multiple static analysis tools using Java cryptographic APIs, highlighting discrepancies in detection coverage and false-positive rates. The methodology and findings reinforce the thesis's focus on practical tool evaluation and the reproducibility of cryptographic vulnerability assessments.

Lizhen Wang (2023) – "Intelligent Detection of Cryptographic Misuse in Android Applications Based on Program Slicing and Transformer-Based Classifier." The integration of machine learning into cryptographic misuse detection has emerged as a promising approach for identifying insecure code patterns [55]. This paper applied program slicing and transformer-based classifiers to detect improper use of cryptographic APIs in Android applications. The methodology reflects the thesis's emphasis on intelligent automation and data-driven evaluation for improving the reliability of cryptographic software development.

Miles Frantz (2024) – "Methods and Benchmark for Detecting Cryptographic API Misuses in Python." Empirical studies of cryptographic misuse in software libraries have increasingly emphasized the need for standardized testing frameworks [16]. This work established benchmarks and detection methods for identifying insecure uses of Python cryptographic APIs, offering reproducible metrics for comparative assessment. The findings reinforce the thesis's practical orientation toward evaluating the security posture of open-source cryptographic environments.

2.6 Post-Quantum and Advanced Cryptographic Frameworks (2022–2024)

Post-quantum cryptography has become a major area of focus in contemporary security research, emphasizing resilience against quantum attacks and the formal verification of emerging schemes. Alongside this evolution, advanced cryptographic frameworks have introduced new languages, infrastructures, and evaluation methodologies to ensure both theoretical soundness and implementation security.

David Pokorný (2022) – "Equivalent Keys: Side-Channel Countermeasure for Post-Quantum Multivariate Quadratic Signatures." Post-quantum cryptographic research has increasingly addressed the resilience of new schemes against side-channel exploitation [38]. This article presented an equivalent-keys countermeasure to enhance the robustness of multivariate quadratic signatures, contributing to the transition toward secure post-quantum implementations. The study supports the thesis's literature observation that post-quantum cryptography and implementation security are converging

research trends.

Tristen Teague (2023) – "Towards Cloud-Based Infrastructure for Post-Quantum Cryptography Side-Channel Attack Analysis." The convergence of cloud computing and post-quantum cryptography has generated new paradigms for scalable security testing [50]. This work proposed a distributed infrastructure for evaluating side-channel resistance of post-quantum algorithms, enabling large-scale automated analysis. The framework's emphasis on scalability and automation reflects core principles echoed in the thesis's recommendations for systematic cryptographic evaluation environments.

Michael J. D. Vermeer (2024) – "Evaluating Cryptographic Vulnerabilities Created by Quantum Computing in Industrial Control Systems." The anticipated impact of quantum computing on industrial security frameworks has prompted renewed scrutiny of cryptographic robustness [54]. This research analyzed the vulnerabilities introduced by post-quantum threats in industrial control systems, identifying weak points in current encryption deployments. Such analysis complements the thesis's literature findings concerning the emergence of quantum-resilient cryptographic paradigms in applied contexts.

Chunning Zhou (2024) – "A New Approach of Evaluating the Security Against Differential and Linear Cryptanalysis and Its Applications to Serpent, NOEKEON and ASCON." Efforts to improve the methodological rigor of cryptanalysis continue to shape cipher evaluation practices [60]. This study proposed an enhanced framework for assessing resistance to differential and linear attacks, applying it to established block ciphers. Its contribution to quantitative cryptanalysis provides theoretical depth to the thesis's broader exploration of algorithmic security assessment.

P. Sun (2024) – "EasyBC: A Cryptography-Specific Language for Security Analysis of Block Ciphers Against Differential Cryptanalysis." Formal specification languages tailored for cryptographic evaluation have facilitated more systematic and automated analysis processes [49]. EasyBC introduced a domain-specific language for evaluating block ciphers against differential cryptanalysis, simplifying complex verification workflows. The approach resonates with the thesis's advocacy for tool-assisted formalization and repeatability in cryptographic evaluation methodologies.

2.7 Cryptographic Infrastructure, Storage, and Trust Management (2013–2022)

Research on cryptographic infrastructures and trust management has focused on ensuring secure data storage, integrity, and key management across distributed environments. The works within this category explore privacy frameworks, distributed storage mechanisms, and trust computations that underpin the stability of modern cryptographic ecosystems.

F. Raji (2013) – "CP2: Cryptographic Privacy Protection Framework for Online Social Networks." Research on privacy-preserving frameworks in social networking environments has focused on the integration of cryptographic mechanisms to mitigate information leakage and enhance user confidentiality [43]. This study introduced the CP2 model, which employs layered encryption and access control strategies to protect personal data exchanges. Such approaches illustrate how theoretical cryptographic constructs can be operationalized within complex distributed systems, aligning with the thesis's examination of applied cryptographic tool behavior.

Jean-Guillaume Dumas (2017) – "Dual Protocols for Private Multi-Party Matrix Multiplication and Trust Computations." Advances in multi-party computation have broadened the understanding of collaborative yet privacy-preserving data processing [12]. This work presented dual cryptographic protocols that enable distributed trust evaluations while maintaining confidentiality. The study's theoretical contributions connect with the thesis's exploration of trust and confidentiality principles in practical cryptographic implementations.

F. Honecker (2022) – "Comparison of Distributed Tamper-Proof Storage Methods for Public Key Infrastructures." The reliability of distributed key management systems remains a pivotal element in maintaining trust across cryptographic infrastructures [19]. This paper compared multiple tamper-proof storage methods within PKIs, emphasizing resilience and decentralization. The discussion aligns closely with the thesis's interest in secure key handling and infrastructure-level security practices in open-source cryptographic tools.

2.8 Novel Algorithmic and Computational Optimization Approaches (2007–2024)

Research into algorithmic innovation and computational optimization has aimed to enhance the performance, scalability, and adaptability of cryptographic systems. The following studies examine hybrid approaches, improved encryption mechanisms, and hardware-aware computation strategies that contribute to the continuous evolution of cryptographic efficiency.

N. K. Kasumov (2007) – "A Cryptographic Tool as an Emulator of Loss-Free Data Compression Procedures." The integration of cryptographic and data-compression mechanisms has been explored as a means to enhance efficiency and storage optimization [23]. This work proposed an algorithm functioning simultaneously as a cryptographic and compression emulator, reflecting early efforts to merge data transformation processes. The conceptual overlap supports the thesis's discussion on the interaction between algorithmic design and system-level performance.

I. González (2006) – "Ciphering Algorithms in MicroBlaze-Based Embedded Systems." Research addressing embedded cryptographic systems has consistently sought to balance algorithmic strength with resource efficiency [18]. The implementation of ciphering methods within MicroBlaze architectures demonstrated that optimization in embedded contexts can yield substantial performance gains. This focus on constrained environments mirrors the analysis conducted in this thesis concerning lightweight cryptographic deployments and energy–security trade-offs.

N. Nedjah (2012) – "Massively Parallel Modular Exponentiation Method and Its Implementation in Software and Hardware for High-Performance Cryptographic Systems." Parallel computation techniques have been leveraged to accelerate cryptographic operations in high-performance environments [33]. This research demonstrated significant improvements in modular exponentiation through hardware–software co-design, underscoring the impact of implementation optimization. The methodological focus complements this thesis's examination of software-based cryptographic performance under practical workloads.

Xin Zheng (2019) – "An Efficient and Low-Power Design of the SM3 Hash Algorithm for IoT." The optimization of hash algorithms for low-power environments has gained significance in contemporary IoT security research [59]. This work proposed a hardware-oriented implementation of the SM3 hashing algorithm to balance energy consumption and computational reliability. The results exemplify how performance constraints influence cryptographic design, directly relating to the thesis's discussion on efficiency and algorithmic adaptability in constrained systems.

Mojisola (2022) – "An Improved Random Bit-Stuffing Technique with a Modified RSA Algorithm for Resisting Attacks in Information Security (RBMRSA)." Efforts to improve classic algorithms through parameter and entropy reinforcement continue to shape applied cryptography [29]. The proposed RBMRSA method incorporated random bit-stuffing to enhance resistance against conventional attacks, blending established cryptographic foundations with adaptive countermeasures. The study mirrors the thesis's thematic concern with balancing algorithmic innovation and implementation reliability.

I. Prots'ko (2024) – "Implementing Montgomery Multiplication to Speed-Up the Computation of Modular Exponentiation of Multi-Bit Numbers." Algorithmic optimization techniques such as Montgomery multiplication have been key to improving computational efficiency in cryptographic systems [39]. This work implemented and analyzed accelerated modular exponentiation methods for large-number computations, demonstrating notable performance gains. These findings correspond with the thesis's technical assessment of algorithmic efficiency and computational optimization in cryptographic operations.

Szilárd Pfeiffer (2024) – "D(HE)at: A Practical Denial-of-Service Attack on the Finite Field Diffie–Hellman Key Exchange." Research addressing protocol-level vulnerabilities has underscored the practical risks of misconfigured key exchange mechanisms [37]. This article documented a feasible denial-of-service attack against the Diffie–Hellman key exchange, emphasizing the importance of parameter validation and implementation hardening. The study complements the thesis's applied dimension, which similarly examines security flaws within cryptographic tool deployments and their mitigation strategies.

2.9 Additional Studies in Cryptographic Evaluation, Randomness, and Applied Security (2013–2023)

A number of complementary studies address distinct yet related aspects of cryptographic evaluation, including algorithm recognition, randomness testing, outsourced computation, and applied encryption analysis. These works contribute additional perspectives to the literature landscape of cryptographic research, particularly in the assessment of randomness, implementation robustness, and emerging cryptographic models.

Rui Chang (2013) – "The Research on Cryptographic Algorithms Recognition Technology." Efforts to identify and classify cryptographic algorithms have contributed to advancing forensic and diagnostic capabilities within security systems [8]. This work presented techniques for algorithm recognition, supporting the detection of specific encryption schemes within applications. The approach is conceptually aligned with the present thesis's concern with identifying and categorizing cryptographic components in real-world tools.

Tilo Müller (2015) – "A Systematic Assessment of the Security of Full Disk Encryption." The security of full-disk encryption solutions has been a recurring topic in applied cryptography [30]. Through a systematic evaluation of vulnerabilities and usability trade-offs, the study established critical benchmarks for assessing data-at-rest protection mechanisms. These findings underpin the thesis's analytical component, which explores similar assessment criteria in contemporary encryption software.

Moreno Ambrosin (2016) – "On the Feasibility of Attribute-Based Encryption on Internet of Things Devices." The feasibility of advanced cryptographic schemes within constrained hardware environments has been increasingly examined [3]. This study evaluated attribute-based encryption for IoT devices, highlighting implementation challenges and computational limitations. The work complements the literature trend identified in the thesis, wherein lightweight and context-aware encryption methods gain prominence.

D. Bogdanov (2018) – "Rmind: A Tool for Cryptographically Secure Statistical Analysis." Efforts to combine cryptographic protection with statistical computation have led to the emergence of privacy-preserving analysis tools [6]. The Rmind

framework demonstrated how data confidentiality can be maintained throughout statistical processing, contributing to the discourse on secure computation environments. These principles resonate with the thesis's consideration of data-centric security in cryptographic applications.

Malik Qasaimeh (2018) – "Software Randomness Analysis and Evaluation of Lightweight Ciphers: The Prospective for IoT Security." Investigations into the adequacy of randomness sources and lightweight cipher performance continue to shape IoT security research [41]. This study analyzed randomness quality and cipher efficiency, offering empirical metrics applicable to constrained systems. Such analyses complement the thesis's benchmarking of cryptographic tool performance under varying computational conditions.

Peter McLaren (2019) – "Deriving ChaCha20 Key Streams From Targeted Memory Analysis." Recent investigations into key recovery through memory analysis have highlighted vulnerabilities inherent to cryptographic implementations [27]. This study demonstrated that ChaCha20 key streams can be reconstructed via targeted memory examination, evidencing the persistence of residual data threats. Such findings reinforce the importance of secure memory management practices, a concern mirrored in the thesis's analysis of open-source cryptographic tools and their configuration weaknesses.

Zhidan Li (2019) – "An Efficient ABE Scheme with Verifiable Outsourced Encryption and Decryption." Attribute-based encryption with outsourced processing has become a focus of scalability research in cloud and IoT security [25]. This article proposed a verifiable and efficient variant of ABE that enables secure delegation of computation. Its findings illustrate the evolution of cryptographic architectures toward modularity and verifiability, key themes reflected in the thesis's bibliographic mapping of modern cryptographic tools.

Nitin Pundir (2022) – "Power Side-Channel Leakage Assessment Framework at Register-Transfer Level." The assessment of side-channel vulnerabilities at early design stages represents a proactive approach to cryptographic security [40]. This study proposed a register-transfer-level framework for evaluating leakage risks, aiming to minimize exploitable flaws in hardware implementations. Such preventive evaluation methods complement the thesis's advocacy for early-stage security validation and system-

atic testing of cryptographic systems.

Pengxu Tian (2022) – "EAFS: An Efficient, Accurate, and Forward Secure Searchable Encryption Scheme Supporting Range Search." Searchable encryption mechanisms have advanced to balance data accessibility and confidentiality in secure storage systems [52]. The EAFS scheme enhanced both efficiency and forward security while enabling range queries, illustrating the evolution of practical encryption designs. This conceptual progression resonates with the thesis's literature focus on the intersection of usability and cryptographic strength.

Algazy (2023) – "Evaluation of the Strength and Performance of a New Hashing Algorithm Based on a Block Cipher." The development and empirical testing of novel hash algorithms derived from block cipher principles reflect ongoing efforts to optimize cryptographic primitives [2]. This study introduced and evaluated a new hashing construction, comparing its computational performance and collision resistance with existing standards. Its analytical orientation parallels the thesis's approach to assessing cryptographic tools through systematic performance metrics and implementation validation.

2.10 Summary

Despite significant progress in formal verification, benchmarking and the design of lightweight cryptographic tools, several persistent challenges and research gaps remain. Notably, the literature reveals that side-channel vulnerabilities and API misuse continue to threaten real-world deployments, often due to the slow or uneven integration of automated detection and mitigation tools. While the transition towards quantum-resilient frameworks introduces promising theoretical innovations, practical evaluation and large-scale adoption lag behind advances made in controlled environments. Furthermore, discrepancies in benchmarking results and methodological standards call into question the reproducibility and generalisability of results across platforms and use cases. Ultimately, these gaps highlight the need for further research into scalable, automated assessment procedures and policy frameworks that can promote the wider and more consistent deployment of secure, adaptive cryptographic solutions.

Chapter 3

Cryptographic Tools

This chapter presents the cryptographic tools selected for practical evaluation in this thesis. The analysis focuses on how each tool behaves under secure (well-configured) and insecure (poorly configured) scenarios, allowing the identification of weaknesses, misconfiguration risks, and differences in resilience across implementations. By comparing tools that represent different cryptographic paradigms, protocol libraries, disk-encryption systems, and public-key encryption frameworks, this study aims to provide a comprehensive understanding of the impact of configuration choices on security.

The tools chosen for this evaluation were selected based on their relevance, widespread adoption, and representation of distinct families of cryptographic applications.

3.1 OpenSSL

OpenSSL[35] serves as an industry-standard library for implementing secure communication protocols. Open Secure Sockets Layer (OpenSSL) is an open-source cryptographic toolkit widely used for implementing Secure Sockets Layer (SSL)/Transport Layer Security (TLS) protocols and providing essential cryptographic primitives. It supports encryption, decryption, hashing, certificate management, and secure key generation. Due to its crucial role in secure communication, misconfigurations in OpenSSL such as outdated protocols, weak cipher suites, or improper certificate handling can expose systems to significant vulnerabilities. For this reason, OpenSSL is included in the present evaluation to enable a detailed assessment of protocol security under different configuration states. Additionally,

administrators rely on OpenSSL for generating cryptographic keys and CSRs in server infrastructures.

3.2 Veracrypt

VeraCrypt[21] represents a commonly used solution for data-at-rest protection. Open-source Disk Encryption Software (VeraCrypt) is an open-source disk-encryption platform that enables users to create encrypted containers, partitions, or full-disk encryption systems. It supports multiple encryption algorithms, including AES[31], Serpent [4], Twofish[46], and various combinations of these, and provides strong key derivation through PBKDF2[22] iterations. However, weak configurations, such as low iteration counts or short passwords, can significantly reduce the level of protection offered by the tool. For this reason, VeraCrypt is used in this evaluation as a representative example of data-at-rest security solutions and their dependence on user-defined parameters. Cybersecurity professionals use it to safeguard evidence and confidential information during investigations.

3.3 GnuPG

GnuPG[51] provides modern public-key cryptography for secure communication and digital signing. GNU Privacy Guard (GnuPG) is an open-source framework that supports public-key encryption, digital signatures, and key management, integrating seamlessly with email clients, command-line workflows, and certificate-based systems. It implements several widely used encryption standards, including RSA[44], ECC[32], and ElGamal[13]. However, improper use of GnuPG, such as relying on outdated key sizes, insecure hash algorithms, or poor key-storage practices can compromise confidentiality and weaken the overall security of communication. For this reason, GnuPG is included in this analysis to examine the behavior of asymmetric cryptographic systems and to highlight the impact of correct versus incorrect configuration in everyday secure communication.Additionally, administrators use GnuPG to encrypt sensitive files and backups in secure workflows.

This chapter establishes the foundational understanding needed for the practical evaluation presented in Chapter 4. By examining each tool under controlled secure and insecure configurations, the study highlights how configuration quality directly influences cryptographic resilience.

Chapter 4

Assessment of Cryptographic Tools

This chapter presents the practical component of the assessment, where each cryptographic tool is tested under both secure and insecure configurations to evaluate its real-world behaviour. Through a series of controlled experiments, the chapter examines how configuration choices affect the security, resilience, and operational reliability of OpenSSL, VeraCrypt, and GnuPG. The following sections document the setup, execution, and outcomes of these experiments, supported by terminal outputs and practical demonstrations.

The following table will demonstrate what I will evaluate and how I will evaluate the cryptographic tools described in previous chapter:

Tool	**What I Will Evaluate**	**How I Will Evaluate**
OpenSSL	Security and resilience in well-configured and poorly configured scenarios.	Simulate attacks on insecure configurations and exploit known vulnerabilities (Heartbleed, SSLv2/SSLv3 support). Test the tool's response to exploits and analyze differences in behavior between the scenarios.
VeraCrypt	Encryption robustness in both strong (secure) and weak (insecure) configurations.	Create encrypted volumes with both secure and insecure configurations. Simulate brute force attacks or attempt to break encryption using external tools. Compare the effectiveness of the configurations.
GnuPG	Effectiveness in protecting data with strong vs. weak key settings and algorithms.	Test keys and algorithms in secure and insecure configurations. Simulate attacks like brute force, vulnerability analysis on outdated versions, or the use of compromised keys. Evaluate the impact of these configurations.

Table 4.1: Evaluation of cryptographic tools in well-configured and poorly configured scenarios

4.1 OpenSSL

In this section it will be simulated attacks on insecure configurations and exploit known vulnerabilities (Heartbleed, SSLv2/SSLv3 support). Testing the tool's response to exploits and analyze differences in behavior between the scenarios.

4.1.1 Generate the private key for the "Misconfigured scenario"

This subsection introduces the creation of a private key using intentionally weak or outdated parameters, forming the basis of the misconfigured server that will later be exploited.

Figure 4.1: Command to generate the private key (Misconfigured scenario)

What the screenshot shows is `openssl genrsa` command and resulting key file. What is done is creating a server private key using chosen parameters (key size, algorithm).

4.1.2 Create a Certificate Signing Request (CSR) to the Misconfigured scenario

Here, a CSR is generated using the previously created weak key. This step prepares the misconfigured server environment that will later be targeted during testing.

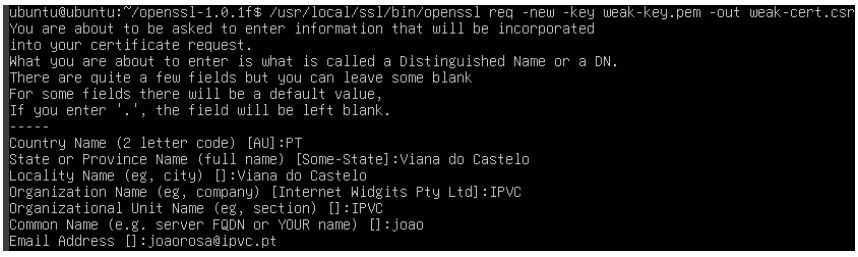

Figure 4.2: Command to Create a Certificate Signing Request (CSR) to the Misconfigured scenario

What the screenshot shows is `openssl req -new` interactive prompts and Certificate Signing Request (CSR) output. What is done is producing a Certificate Signing Request that contains server identity info and the public key.

4.1.3 Generate certificate from CSR to the Misconfigured scenario

This subsection shows how the misconfigured CSR is converted into a certificate, completing the insecure configuration that will be analyzed in the subsequent attack.

Figure 4.3: Command to generate certificate from CSR to the Misconfigured scenario

What the screenshot shows is `openssl x509 -req` or Certificate Authority (CA) signing commands producing a `.crt` file. What is done is self-signing or signing the CSR with a test CA to create a server certificate.

4.1.4 Clone the repository of heartbleed

In this subsection, the Heartbleed proof-of-concept repository is downloaded to prepare the environment used to test the vulnerability on the insecure OpenSSL setup.

Figure 4.4: Command to clone the repository of heartbleed

What the screenshot shows is `git clone` of a Heartbleed[9] proof-of-concept repo. What is done is downloading exploit code/tools that target the Heartbleed bug (Common Vulnerabilities and Exposures (CVE)-era POC).

4.1.5 Attack the insecure server of OpenSSL (Misconfigured scenario) with heartbleed

This part demonstrates how the misconfigured OpenSSL server reacts when targeted by the Heartbleed exploit, revealing the security impact of weak configurations.

Figure 4.5: Commands to start the insecure OpenSSL server and attack with the heartbleed (Misconfigured scenario)

What the screenshot shows is exploit command output (for example: leaked memory, private data). What is done is running the Proof of Concept (PoC) against the test server and capturing the leaked memory slices or secrets.

4.1.6 Generate the private key for the "Well-configured scenario"

This subsection introduces the generation of a secure private key using strong parameters, forming the basis for the hardened OpenSSL environment.

Figure 4.6: Command to generate the private key (Well-configured scenario)

What the screenshot shows is `openssl genrsa` with stronger parameters (larger key, Rivest–Shamir–Adleman (RSA)/Elliptic Curve Cryptography (ECC) selection). What is done is creating a robust key using recommended algorithms and sizes.

4.1.7 Create a Certificate Signing Request (CSR) to the Well-configured scenario

Here, a CSR is generated using secure configurations, ensuring that the resulting server certificate follows best security practices.

Figure 4.7: Command to Create a Certificate Signing Request (CSR) to the Well-configured scenario

What the screenshot shows is `openssl req -new` with secure parameter choices. What is done is producing a CSR that uses the secure key and proper metadata.

4.1.8 Generate certificate from CSR to the Well-configured scenario

This subsection finalizes the well-configured OpenSSL setup by generating a properly signed certificate based on the secure CSR.

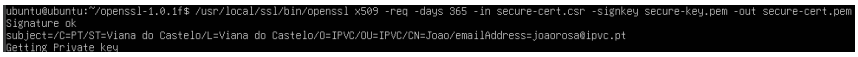

Figure 4.8: Command to generate certificate from CSR to the Well-configured scenario

What the screenshot shows is signing CSR into a certificate using a secure CA process. What is done is issuing a certificate that the secure server will use.

4.1.9 Attack the secure server of OpenSSL (Well-configured scenario) with heartbleed

This final part compares the secure configuration against the Heartbleed exploit, demonstrating how proper OpenSSL hardening prevents or mitigates the attack.

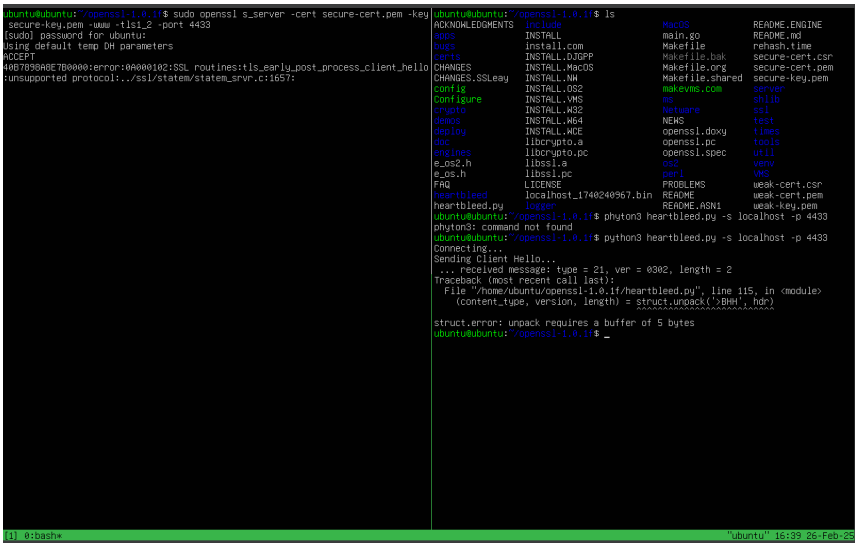

Figure 4.9: Commands to start the secure OpenSSL server and attack with the heartbleed (Well-configured scenario)

What the screenshot shows is the failed exploit attempts or no sensitive data leakage. What is done is re-running the PoC against the hardened server and showing no leakage or reduced impact.

4.2 Veracrypt

This section create encrypted volumes with both secure and insecure configurations and Simulate brute force attacks or attempt to break encryption using external tools, comparing the effectiveness of the configurations.

4.2.1 Add the Veracrypt repository

This subsection introduces the process of adding the official VeraCrypt repository to the system, enabling the installation of the tool through the package manager.

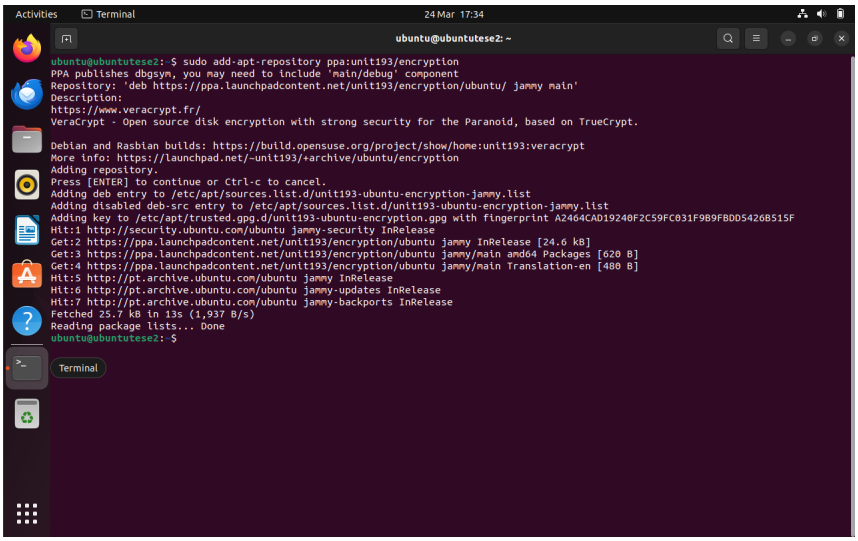

Figure 4.10: Command to add the Veracrypt repository

What the screenshot shows is a command adding a package repository add-apt-repository or showing repository source lines. What is done is configuring the package manager to access the official VeraCrypt packages.

4.2.2 Update and install Veracrypt

Here, the repository is updated and VeraCrypt is installed, preparing the environment for the subsequent creation and analysis of encrypted volumes.

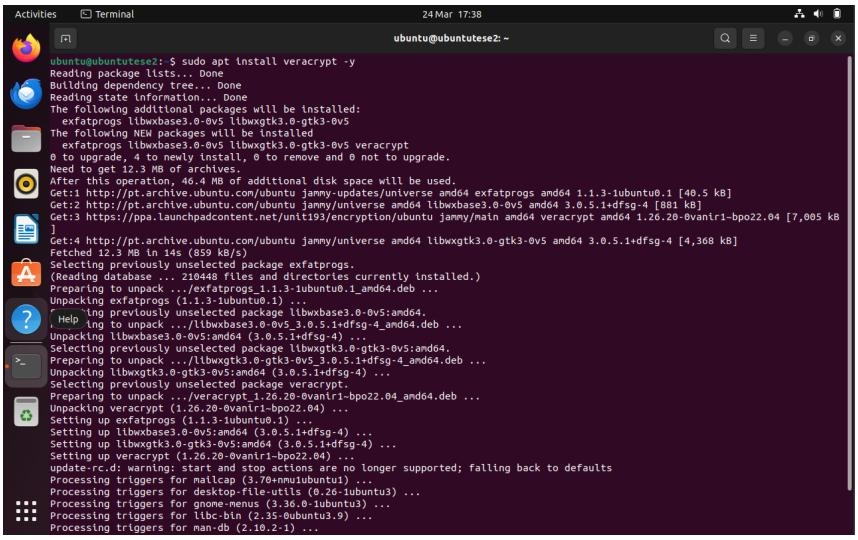

Figure 4.11: Command to install Veracrypt

What the screenshot shows is the `apt update` and `apt install veracrypt` output. What is done is the download and install of the VeraCrypt application and dependencies.

4.2.3 Mounting the disk in a specific folder

This subsection demonstrates how an encrypted volume is mounted into a defined directory, allowing the contents to be accessed after successful authentication.

Figure 4.12: Mounting the disk in a specific folder

What the screenshot shows is `veracrypt --mount <container> /mnt/secure` command. What is done is the unlocking an encrypted container (or partition) by providing a passphrase/keyfile and mapping it to a mount point.

4.2.4 Mounting the Strong Volume

In this part, the strongly configured VeraCrypt volume is mounted using its secure passphrase and keyfiles, establishing the baseline for the secure scenario.

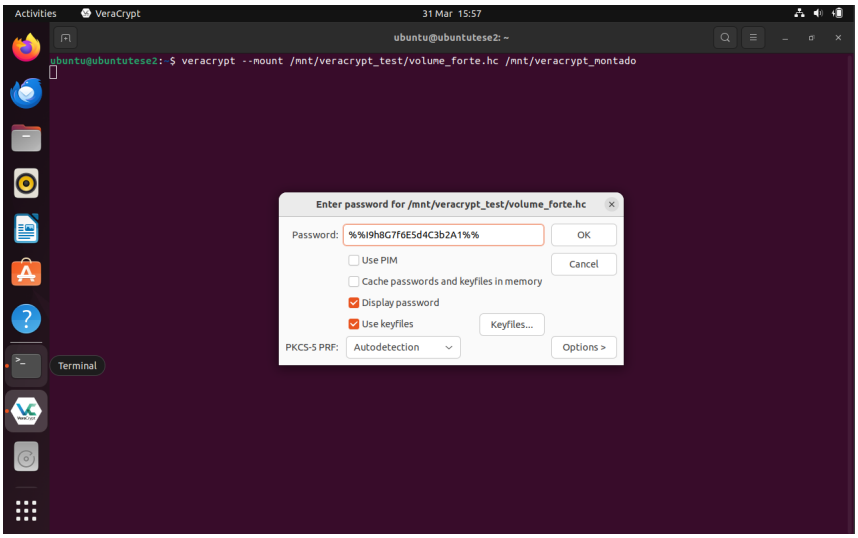

Figure 4.13: Mounting the Strong Volume

What the screenshot shows is the VeraCrypt Graphical User Interface (GUI) (or command-line) prompt showing a successful mount action for the “Strong” volume, usually a selected container file or partition mapped to a drive letter or mount point. What is done is the user supplies the strong-volume passphrase (and any required keyfiles) and issues the mount command.

4.2.5 Mounting the Weak Volume

This subsection shows the mounting of the weakly configured volume, which uses reduced security parameters and will later be subjected to brute-force testing.

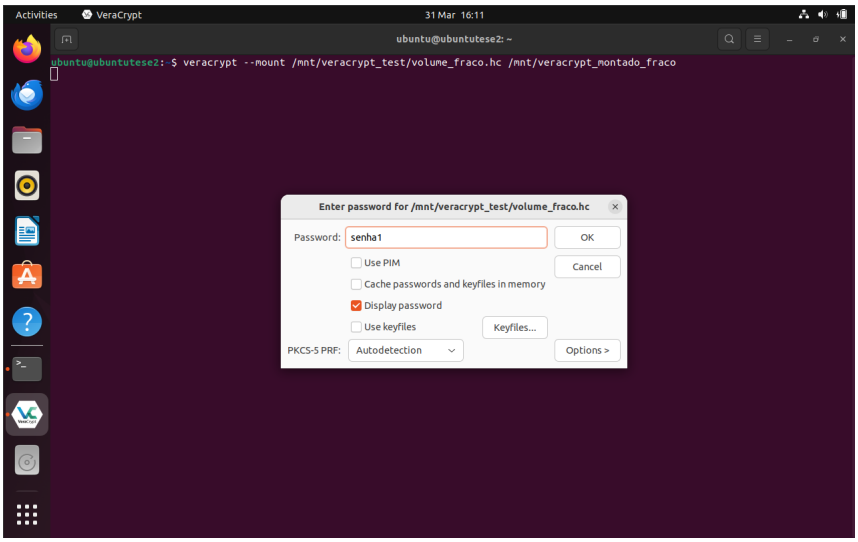

Figure 4.14: Mounting the Weak Volume

What the screenshot shows is the Veracrypt GUI performing a mount of the “Weak” volume, typically showing a successful mount after entering a weak passphrase (or a passphrase chosen to be vulnerable). What is done is the weak container being unlocked using its passphrase, it mounts the filesystem just like the strong volume.

4.2.6 Showing all Volumes mounted

Here, the system displays all currently mounted VeraCrypt volumes, confirming that both the strong and weak scenarios have been successfully loaded.

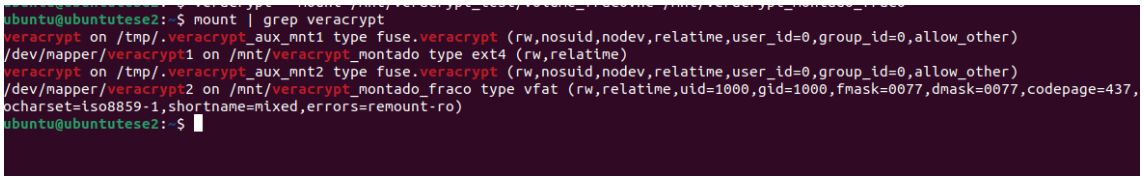

Figure 4.15: Showing all Volumes mounted

What the screenshot shows is the VeraCrypt's "Volumes Mounted" view or a list of currently mounted volumes (drive letters, mount points, volume names, and status). What is done is that VeraCrypt enumerates active mounts and shows metadata (size, mount point, read/write status).

4.2.7 Installing rockyou dictionary

This subsection prepares the attack environment by installing the rockyou password dictionary, which will be used in the brute-force attempts.

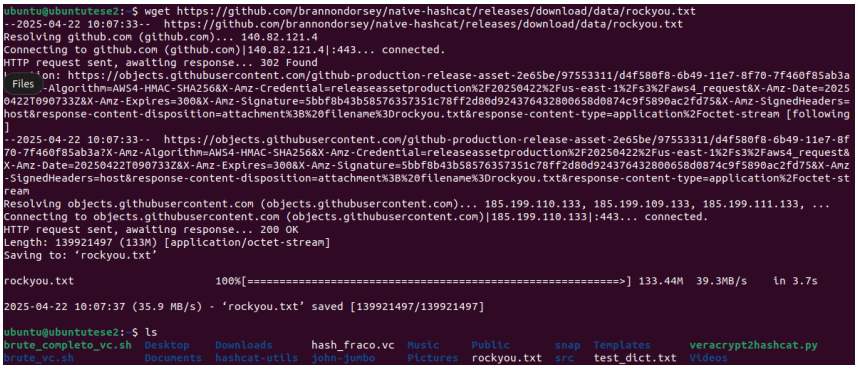

Figure 4.16: Installing rockyou dictionary

What the screenshot shows is the terminal output showing download/installation/extraction of the rockyou[45] wordlist (commonly used for password-cracking). What is done is the rockyou.txt dictionary file is installed or copied into the testing environment so it can be fed to password-cracking tools.

4.2.8 Testing John The Ripper script brutte force attack with rockyou dictionary in weak volume

This part evaluates the weak configuration by running a dictionary attack against the volume's password using John the Ripper and the rockyou wordlist.

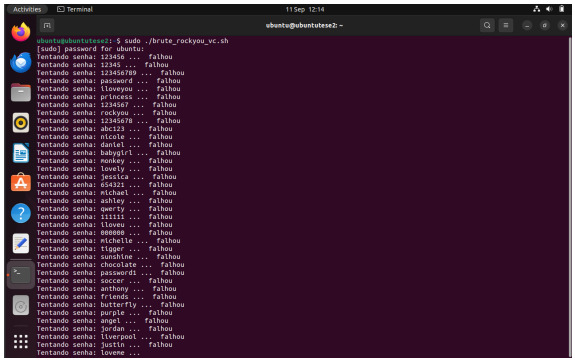

Figure 4.17: Testing John The Ripper script brutte force attack with rockyou dictionary in weak volume

What the screenshot shows is John the Ripper[36] (or a wrapper script) running and attempting to crack a password derived from the weak VeraCrypt volume, with progress/status lines shown. What is done is that the attacker runs John (or a script that extracts the VeraCrypt header/hash) with the rockyou wordlist, John iterates through candidate passwords trying to decrypt the volume header or break the header-derived key.

4.2.9 Script of John the Ripper for weak volume with rockyou dictionary

This subsection presents the script used to automate the extraction of the VeraCrypt hash and execute the brute-force attack with the rockyou list.

Figure 4.18: Script of John the Ripper for weak volume with rockyou dictionary

What the screenshot shows is the attack script used to automate extraction of the VeraCrypt hash and the invocation of John with the rockyou wordlist. What is done is that the script typically runs a header-extraction tool (producing a crackable hash), calls John with parameters (format, wordlist, rules), and records output.

4.2.10 Script of John the Ripper for weak volume with the created dictionary with the correct password to facilitate

Here, an alternative attack script is shown, this time using a small custom dictionary that includes the correct password to test controlled cracking conditions.

```
ubuntu@ubuntutese2:~$ cat brute_completo_vc.sh
VC_CONTAINER="/mnt/veracrypt_test/volume_fraco.hc"
MOUNT_POINT="/mnt/vc_test"
DICT="test_dict.txt"

while IFS= read -r pass; do
  echo -n "Tentando senha: $pass ... "
  if veracrypt --text --non-interactive \
               --mount "$VC_CONTAINER" \
               "$MOUNT_POINT" \
               --password="$pass" \
               --keyfiles="" \
               --pim=0 \
               >/dev/null 2>&1; then
    echo " encontrou!"
    echo "Senha correta: $pass"
    exit 0
  else
    echo " falhou"
  fi
done < "$DICT"

echo "Nenhuma senha do dicionário funcionou."
exit 1
ubuntu@ubuntutese2:~$
```

Figure 4.19: Script of John the Ripper for weak volume with the dictionary created with the correct password to facilitate

What the screenshot shows is an alternate script that uses a custom, small dictionary (which intentionally contains the correct password) to demonstrate a controlled, expedited cracking test. What is done is that the attacker replaces rockyou with a small crafted wordlist and runs John, the correct password is in the list, so John finds it quickly.

4.2.11 Testing John The Ripper script brutte force attack with the created dictionary with the correct password to facilitate in weak volume

This subsection demonstrates how John the Ripper successfully recovers the weak password when using the custom dictionary specifically prepared for this scenario.

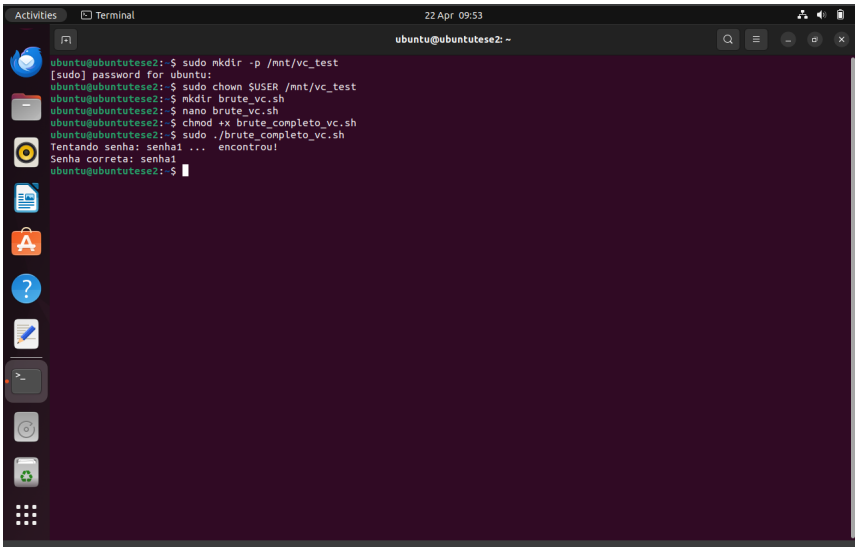

Figure 4.20: Testing John The Ripper script brutte force attack with the created dictionary with the correct password to facilitate in weak volume

What the screenshot shows is John's output showing the correct password found quickly when the small custom dictionary is used. What is done is the cracking process reads the hash, tries each candidate, and reports a successful match.

4.2.12 Testing John The Ripper script brutte force attack with the created dictionary with the correct password to facilitate in strong volume but useless because the attacker don't have the keyfiles

This final part shows that even with a correct password in the dictionary, the strong volume cannot be accessed because the required keyfiles are absent, highlighting the effectiveness of strong configurations.

```
ubuntu@ubuntutese2:~$ cat test_dict22.txt
%%I9h8G7f6E5d4C3b2A1%%
ubuntu@ubuntutese2:~$ nano brute_
brute_completo_vc.sh  brute_rockyou_vc.sh   brute_vc.sh/
ubuntu@ubuntutese2:~$ nano brute_
brute_completo_vc.sh  brute_rockyou_vc.sh   brute_vc.sh/
ubuntu@ubuntutese2:~$ nano brute_completo_vc.sh
ubuntu@ubuntutese2:~$ sudo ./brute_completo_vc.sh
[sudo] password for ubuntu:
Tentando senha: %%I9h8G7f6E5d4C3b2A1%% ...  encontrou!
Senha correta: %%I9h8G7f6E5d4C3b2A1%%
ubuntu@ubuntutese2:~$ cat brute_completo_vc.sh
VC_CONTAINER="/mnt/veracrypt_test/volume_forte.hc"
MOUNT_POINT="/mnt/vc_test"
DICT="test_dict22.txt"

while IFS= read -r pass; do
  echo -n "Tentando senha: $pass ... "
  if veracrypt --text --non-interactive \
               --mount "$VC_CONTAINER" \
               "$MOUNT_POINT" \
               --password="$pass" \
               --keyfiles="" \
               --pim=0 \
               >/dev/null 2>&1; then
    echo " encontrou!"
    echo "Senha correta: $pass"
    exit 0
  else
    echo " falhou"
  fi
done < "$DICT"

echo "Nenhuma senha do dicionário funcionou."
exit 1
```

Figure 4.21: Testing John The Ripper script brutte force attack with the created dictionary with the correct password to facilitate in strong volume but useless because the attacker don't have the keyfiles

What the screenshot shows is John running the strong volume hash, but even if he manages to recover the password, he cannot mount the volume because he does not have the necessary keyfiles in addition to the password to do it. What is done is the same extraction and cracking procedure is run against the strong volume, but without the keyfiles or with a strong passphrase, John cannot derive the correct key.

4.3 GnuPG

In this section will be tested keys and algorithms in secure and insecure configurations. Simulating attacks like brute force, vulnerability analysis on outdated versions, or the use of compromised keys, and evaluating the impact of these configurations.

4.3.1 Creating a secure GPG key (good/strong configuration)

This subsection introduces the generation of a strong GPG key using secure parameters, forming the foundation of the well-configured scenario.

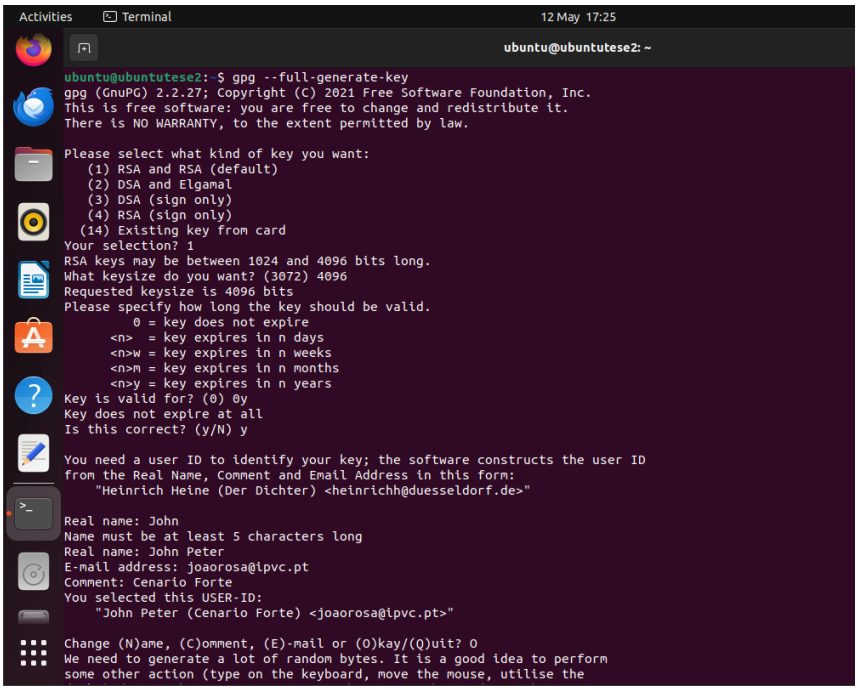

Figure 4.22: Command and secure GPG key configuration (good/strong configuration)

The screenshot shows interactive gpg –full-generate-key prompts requesting key type, size, expiration and user identity (name/email). What is done is that the user selects recommended strong algorithms (e.g., RSA 3072/4096 or ECC options), sets a sensible expiry, and provides identity metadata.

4.3.2 Creating a secure GPG key (good/strong configuration) 2

Here, the secure key-generation process continues, including the setup of a strong passphrase to protect the private key.

Figure 4.23: Secure GPG key configuration (good/strong configuration)

The screenshot shows the passphrase prompt and the final confirmation that the key was generated along with the fingerprint output. What is done is that the user chooses a long, high-entropy passphrase and records the key fingerprint.

4.3.3 Passphrase strong scenario

This subsection highlights the application of a high-entropy passphrase, reinforcing the security of the strong key configuration.

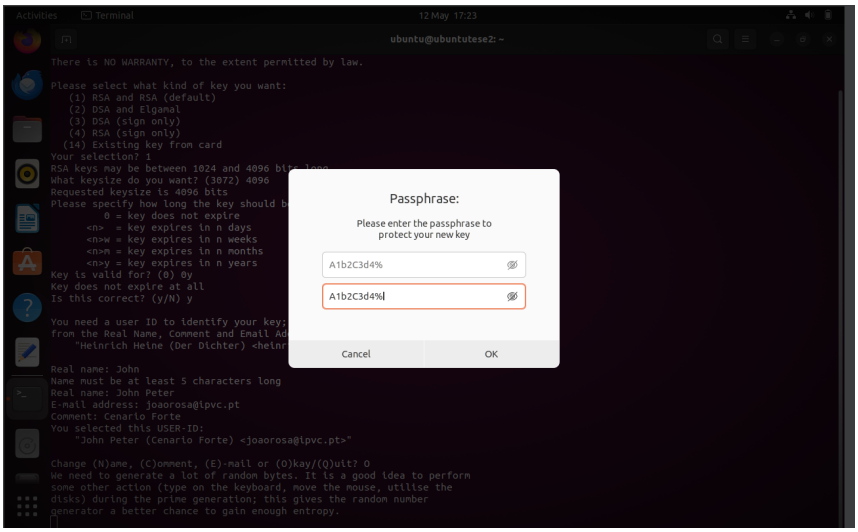

Figure 4.24: Passphrase strong scenario

The screenshot shows a passphrase-entry dialog or terminal prompt showing that a strong passphrase was used. What is done is that a strong passphrase is applied to the key, sometimes illustrated with passphrase entropy or length comments.

4.3.4 Encrypt a file (secure test)

In this part, a file is encrypted using the secure key, demonstrating how confidential data is protected under a strong configuration.

Figure 4.25: Command to encrypt a file (secure test)

The screenshot shows a command like gpg -e -r recipient file.txt and the resulting .gpg file listed. What is done is that the file is encrypted with the recipient's public key so only the corresponding private key can decrypt it.

4.3.5 Decrypt the strong scenario file 1

This subsection shows the initial stage of decrypting a file encrypted under the strong configuration, requiring the user's secure passphrase.

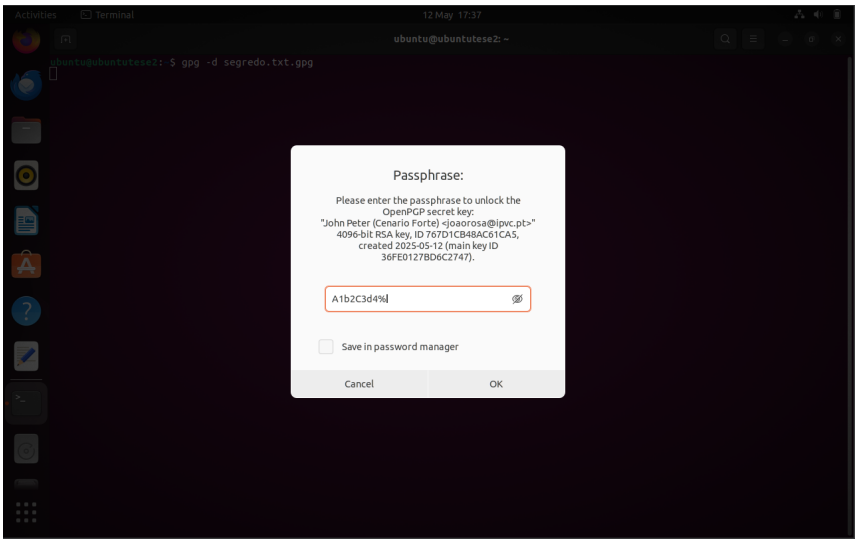

Figure 4.26: Decrypt the strong scenario file

The screenshot shows the command gpg -d file.gpg with the program prompting for the private key passphrase. What is done is that the owner enters their passphrase and GnuPG decrypts and prints or restores the original file.

4.3.6 Decrypt the strong scenario file 2

Here, the decryption process is completed, confirming that only the legitimate key holder can restore access to the protected data.

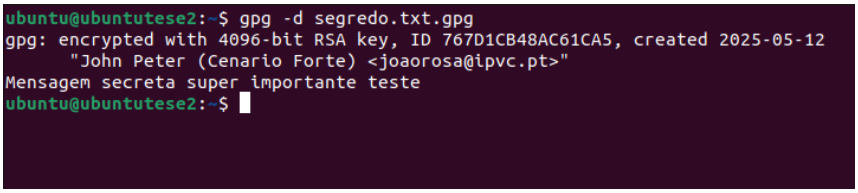

Figure 4.27: Command to decrypt the strong scenario file

The screenshot shows the success output, either the original plaintext or a confirmation of decryption after passphrase entry. What is done is that it confirms the file was correctly decrypted and that the private key is usable.

4.3.7 Export the private key with protection 1

This subsection demonstrates how the private key can be exported while retaining passphrase protection, ensuring it remains encrypted during transport.

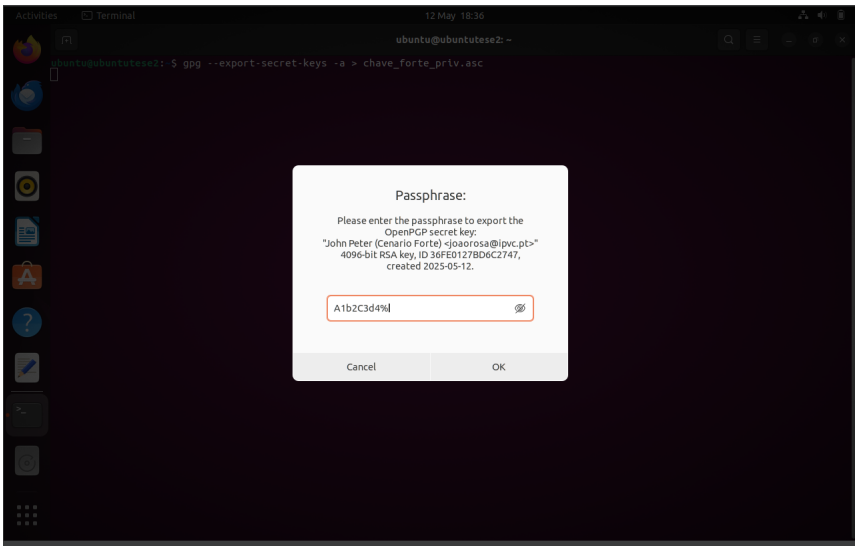

Figure 4.28: Export the private key with protection

The screenshot shows gpg –export-secret-keys -a "User" ¿ secret.asc or gpg –export-secret-keys –armor, followed by a note that the exported key is protected by its passphrase. What is done is that the private key is exported in ASCII-armored format while preserving its passphrase-based symmetric protection.

4.3.8 Export the private key with protection 2

Here, an alternative export method is presented, again preserving the protective passphrase applied to the private key.

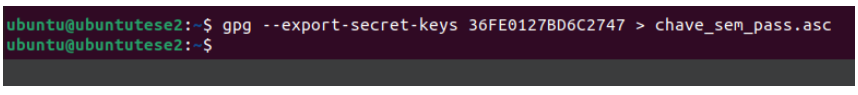

Figure 4.29: Command to export the private key with protection

The screenshot shows an alternative export command, possibly with –export-secret-subkeys or using GnuPG's –symmetric wrapper, and a reminder about secure transport. What is done is that another approach to export is illustrated while keeping the private material encrypted.

4.3.9 Dictionary attack command used to discover passwords in strong scenarios

This subsection introduces the commands used to prepare and launch a dictionary attack against the strong key, simulating an attacker's strategy.

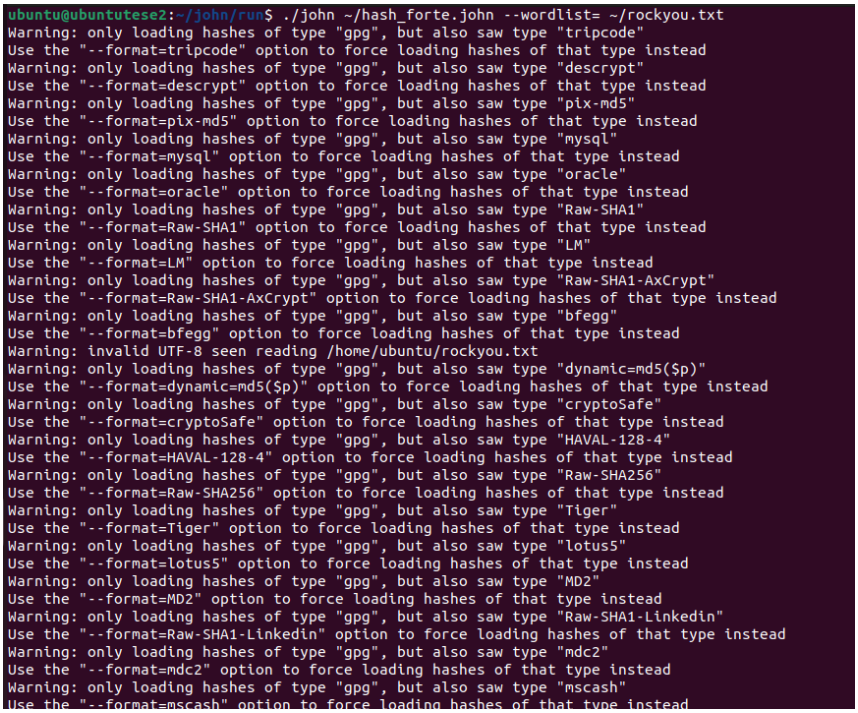

Figure 4.30: Dictionary attack command used to discover passwords in strong scenarios

The screenshot shows the command line used to prepare a password-cracking target and the invocation of John the Ripper with a rockyou.txt wordlist. What is done is that private key files or passphrase verifiers are converted into a format that password-cracking tools can consume, and then the attacker runs a dictionary attack.

4.3.10 Proof of key resistance in the recent strong GnuPG scenario

Here, the results of the attack attempt are shown, demonstrating that the strong passphrase effectively resists dictionary-based cracking.

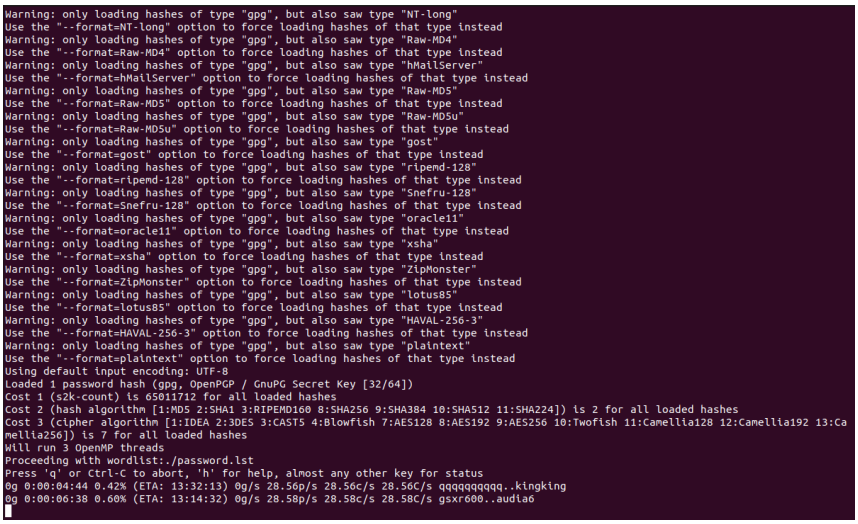

Figure 4.31: Proof of key resistance in the recent strong gnupg scenario

The screenshot shows John's output indicating no matches found or extremely slow progress, or an explicit failure to crack a strong passphrase. What is done is that an attempted dictionary or wordlist attack against the strong key results in no successful recovery.

4.3.11 Compromised Key (Remove passphrase 1)

This subsection shows how an attacker could attempt to remove the passphrase from a compromised key if they manage to obtain access to the unprotected key material.

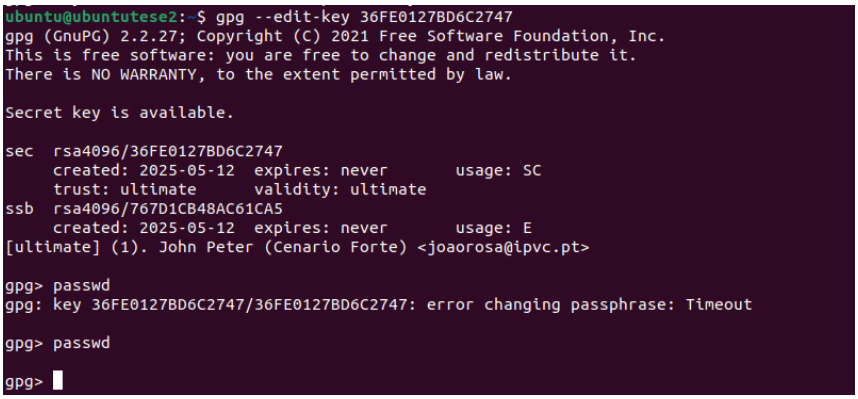

Figure 4.32: Command to Compromised Key (Remove passphrase)

The screenshot shows a command sequence used by an attacker to remove a passphrase, such as gpg –edit-key followed by passwd or exporting and importing unprotected key material, or tools showing passphrase stripping. What is done is that it demonstrates how, if an attacker gains access to an unlocked private key or to the unprotected key file, they can remove passphrase protection.

4.3.12 Compromised Key (Remove passphrase 2)

Here, the attacker continues altering the key configuration, progressing toward stripping its passphrase protection.

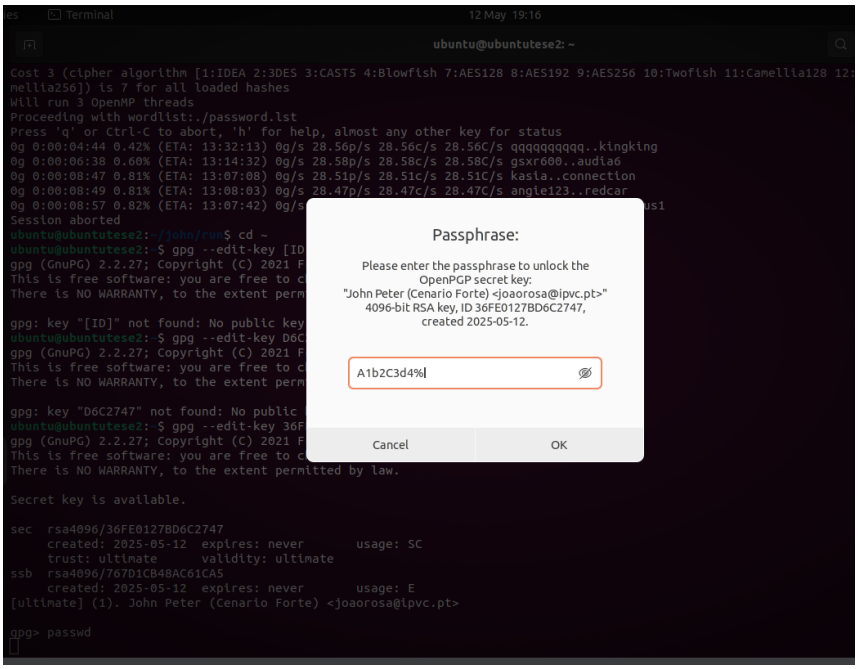

Figure 4.33: Compromised Key (Remove passphrase 2)

The screenshot shows the part where the passphrase is edited, currently showing the original passphrase. What is done is that the attacker finalizes making the key usable without the original passphrase.

4.3.13 Compromised Key (Remove passphrase 3)

This part completes the passphrase-removal sequence, resulting in a private key that can be used without authentication.

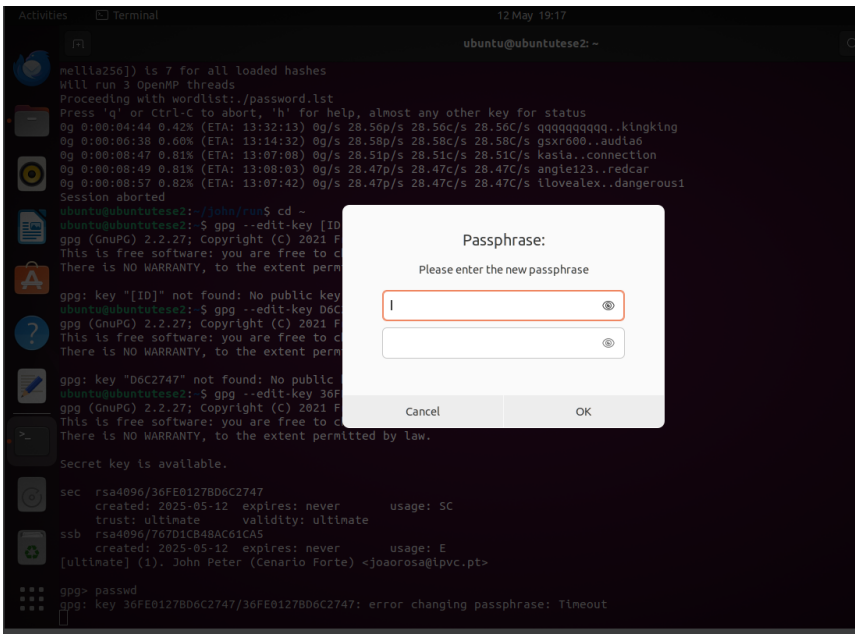

Figure 4.34: Compromised Key (Remove passphrase 3)

The screenshot shows confirmation of the passphrase removal. What is done is that the attacker finalizes making the key usable without the original passphrase.

4.3.14 Compromised Key (Exporting the private key without a passphrase)

This subsection demonstrates the extraction of a private key after its passphrase has been removed, illustrating a complete compromise scenario.

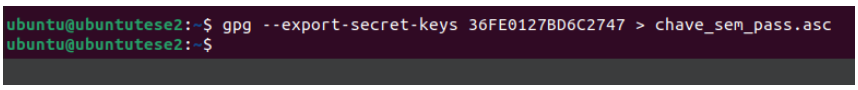

Figure 4.35: Command to Compromised Key (Exporting the private key without a passphrase)

The screenshot shows the command exporting the private key in unencrypted form. What is done is that the attacker creates a portable unprotected private key file usable without a passphrase.

4.3.15 Compromised key (New attacker user)

Here, a new attacker-controlled user environment is created to simulate the import and misuse of the compromised key.

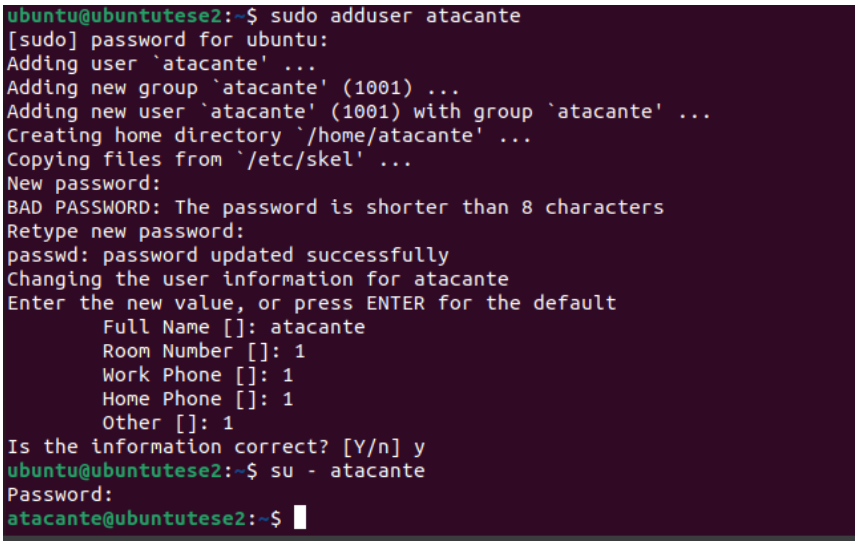

Figure 4.36: Compromised key (New attacker user)

The screenshot shows the creation of a new system account representing the attacker. What is done is that the attacker sets up an environment to receive and use the compromised key.

4.3.16 Compromised Key (Copy files to that user)

This subsection transfers the compromised key files into the attacker's environment, preparing for unauthorized key usage.

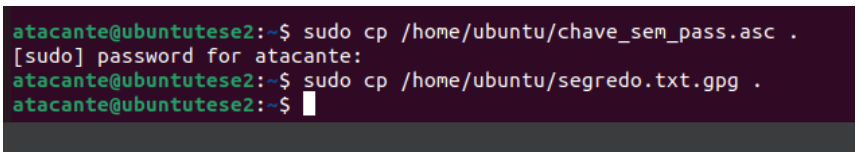

Figure 4.37: Compromised Key (Copy files to that user)

The screenshot shows file copy operations moving the stolen private key into the attacker's account. What is done is that the key file is relocated under attacker control.

4.3.17 Compromised Key (Import compromised key)

In this part, the attacker imports the stolen and unprotected key into their GPG keyring, gaining full control over it.

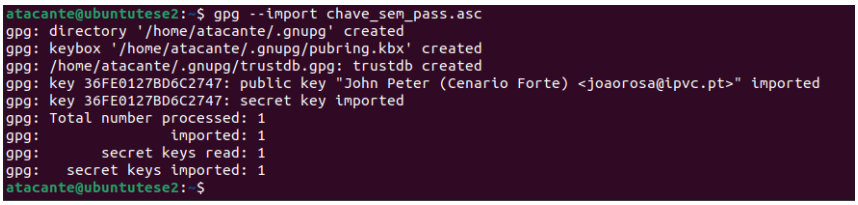

Figure 4.38: Compromised Key (Import compromised key)

The screenshot shows gpg –import secret.asc executed as the attacker and GnuPG listing the imported key. What is done is that the attacker imports the private key into their keyring, making it usable for decryption and signing.

4.3.18 Compromised Key (Effortless Decryption)

Here, the attacker decrypts files without resistance, demonstrating the severe impact of removing or weakening passphrase protection.

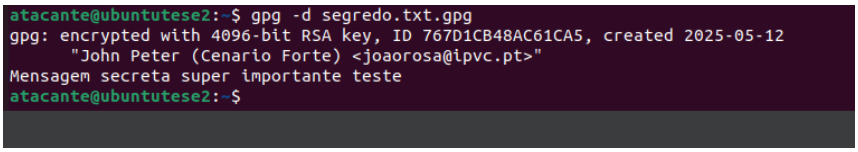

Figure 4.39: Compromised Key (Effortless Decryption)

The screenshot shows the decryption of previously encrypted test files using the imported key with no passphrase prompt or with a trivial passphrase, producing plaintext output. What is done is that the attacker demonstrates full access to the victim's encrypted data.

4.3.19 Creating an insecure scenario command and configuration 1 recent GnuPG

This subsection introduces the first insecure configuration, where weak parameters are used to generate a vulnerable key.

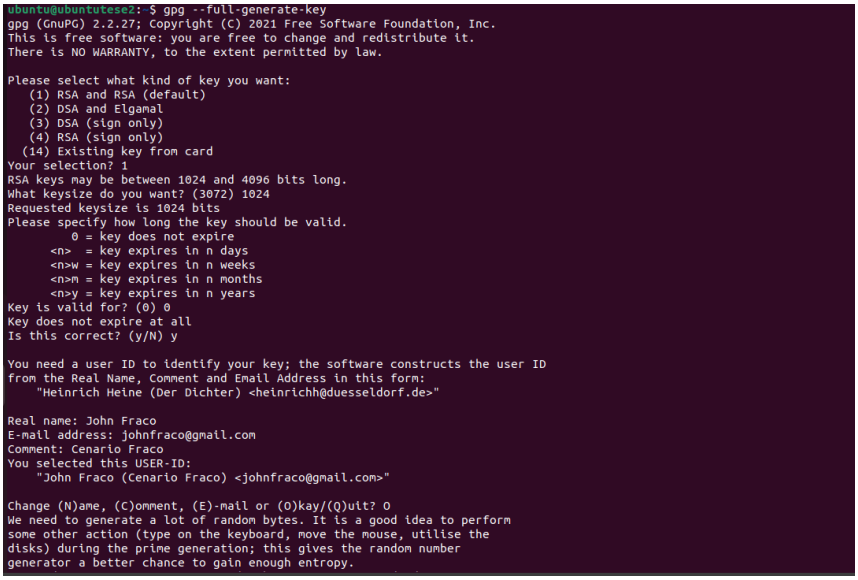

Figure 4.40: Creating an insecure scenario command and configuration 1 recent GnuPG

The screenshot shows the commands used to create a weak GnuPG key, such as a very small RSA key, no expiry, or weak algorithms, or to initialize a key with a short passphrase. What is done is that the user intentionally chooses insecure parameters to create an insecure test key for experiments.

4.3.20 Creating an insecure scenario command and configuration 2 recent GnuPG

Here, the insecure configuration is further expanded, showing additional weaknesses introduced during the key-generation process.

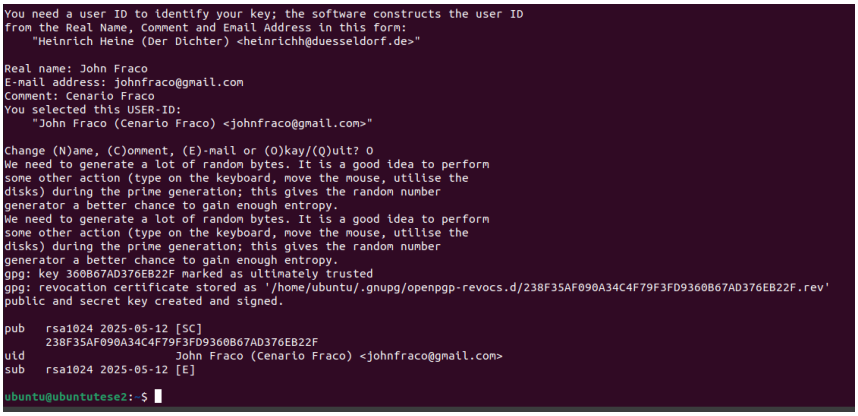

Figure 4.41: Creating an insecure scenario command and configuration 2 recent Gnupg

The screenshot shows further configuration steps for the insecure key. What is done is that the weak-key creation is finalized so it is ready for testing.

4.3.21 Creation of an insecure scenario with a weak password in recent GnuPG

This subsection highlights the use of a weak passphrase, demonstrating how poor user choices significantly reduce the key's security.

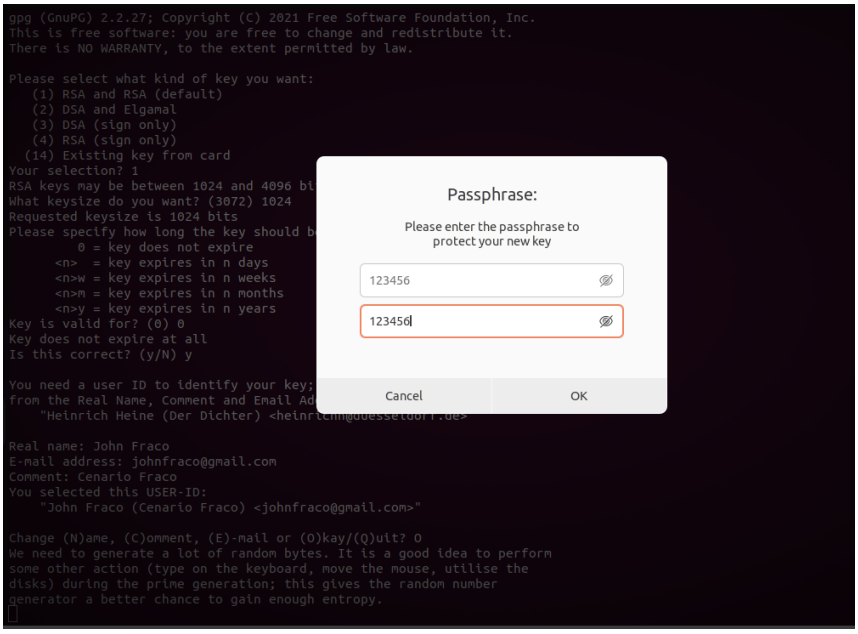

Figure 4.42: Creation of an insecure scenario with a weak password in recent GnuPG

The screenshot shows the passphrase prompt where a short/simple passphrase is entered. What is done is that it assigns an intentionally weak passphrase to the private key.

4.3.22 Recent GnuPG insecure scenario (Encrypting a file (secure test))

Here, a file is encrypted using the weak key, setting up the conditions to show how easily an insecure configuration can be exploited.

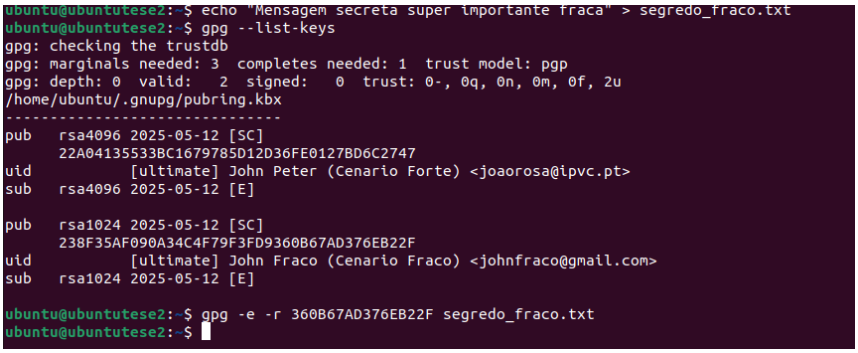

Figure 4.43: Recent GnuPG insecure scenario (Encrypting a file (secure test))

The screenshot shows encryption of a test file using the weak-key recipient, producing ciphertext for the offline cracking experiments. What is done is that it creates a target encrypted file whose decryption depends on a weak passphrase-protected private key.

4.3.23 Recent GnuPG security issue (Decrypting the file)

This subsection demonstrates that the attacker can decrypt the file due to the insufficient protection provided by the weak configuration.

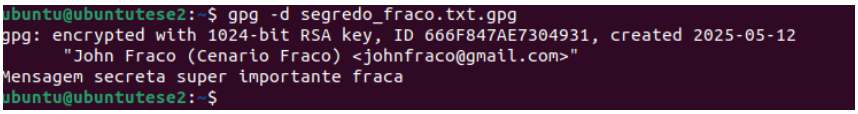

Figure 4.44: Recent GnuPG security issue (Decrypting the file)

The screenshot shows the attempt to decrypt the test ciphertext, possibly succeeding quickly due to the weak passphrase or exported unprotected key. What is done is that demonstrates decryption using recovered credentials or after successful cracking.

4.3.24 Recent GnuPG security issue (Private key export)

Here, the private key is exported with minimal protection, illustrating a critical vulnerability of poorly configured GPG setups.

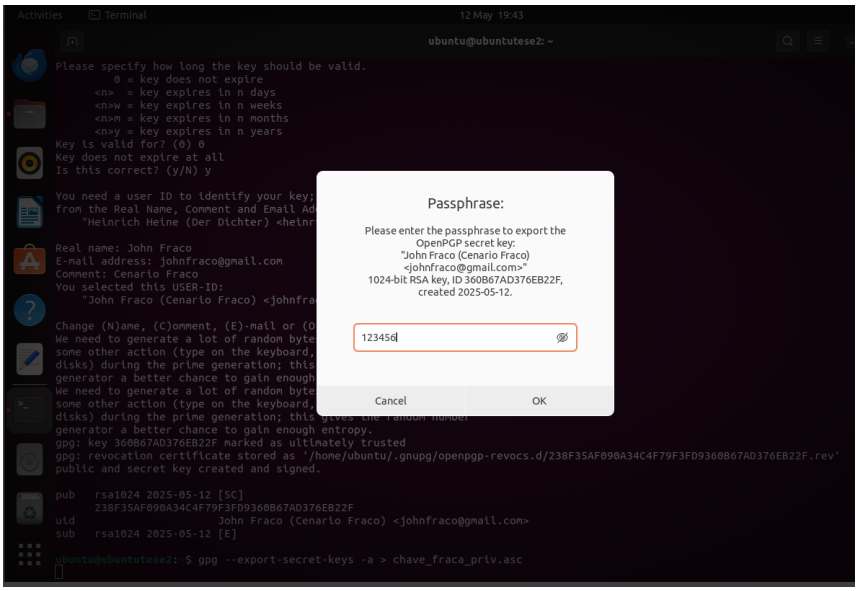

Figure 4.45: Recent GnuPG security issue (Private key export)

The screenshot shows the command `gpg --export-secret-keys -a > chave_fraca_priv.asc` . What is done is the export of the private key.

4.3.25 Recent GnuPG security issue (Extract the hash for JOHN) 1

This subsection shows how an attacker prepares the weak key by extracting its hash, enabling offline password-cracking attacks.

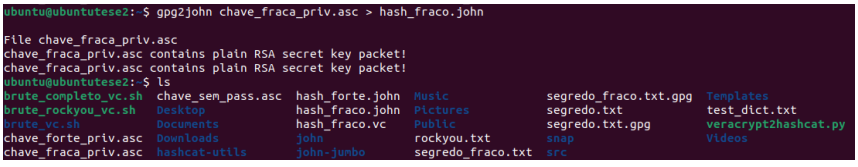

Figure 4.46: Command to recent GnuPG security issue (Extract the hash for JOHN) 1

The screenshot shows the invocation of `gpg2john` or a similar tool on an exported private key to generate a hash file consumable by John the Ripper. What is done is the key file is transformed into a hash representation that preserves the passphrase-verifier information used by GnuPG.

4.3.26 Recent GnuPG security issue (Extract the hash for JOHN) 2

Here, the extraction continues, producing the final hash file to be consumed by John the Ripper during cracking.

Figure 4.47: Command to Recent GnuPG security issue (Extract the hash for JOHN) 2

The screenshot shows the resulting `.john` (or `.hash`) file and short example lines of the format produced. What is done is that shows the exact hash string John will try to crack.

4.3.27 Recent GnuPG insecure scenario (Command used to attack with the dictionary attack with John)

This subsection introduces the dictionary attack launched by the attacker, attempting to recover the weak passphrase.

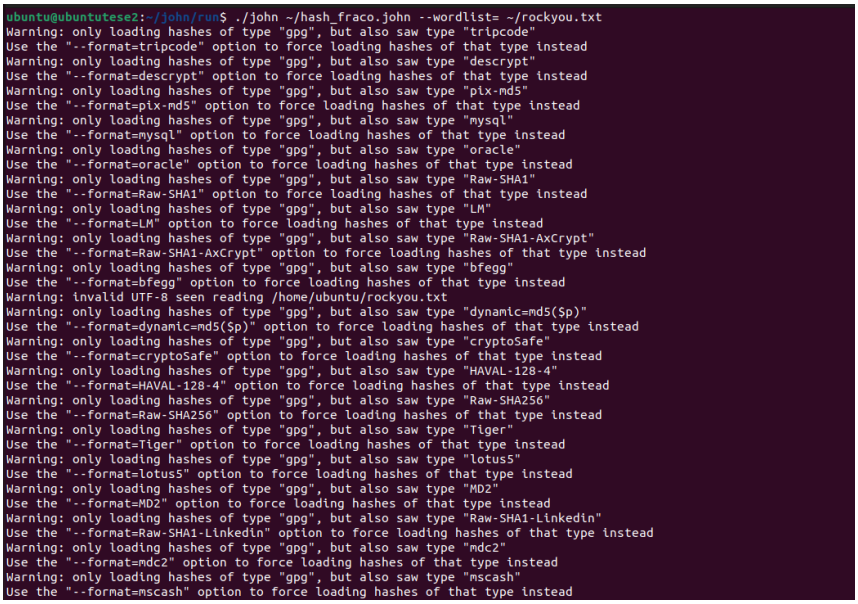

Figure 4.48: Recent GnuPG insecure scenario (Command used to attack with the dictionary attack with John)

The screenshot shows the command invoking John the Ripper with a wordlist and any options/rules used. What is done is runs an offline dictionary attack against the GnuPG-derived hash.

4.3.28 Recent GnuPG insecurity scenario (Final result of dictionary attack with John)

Here, the attack results are presented, showing the successful recovery of the weak passphrase and demonstrating the consequences of insecure configurations.

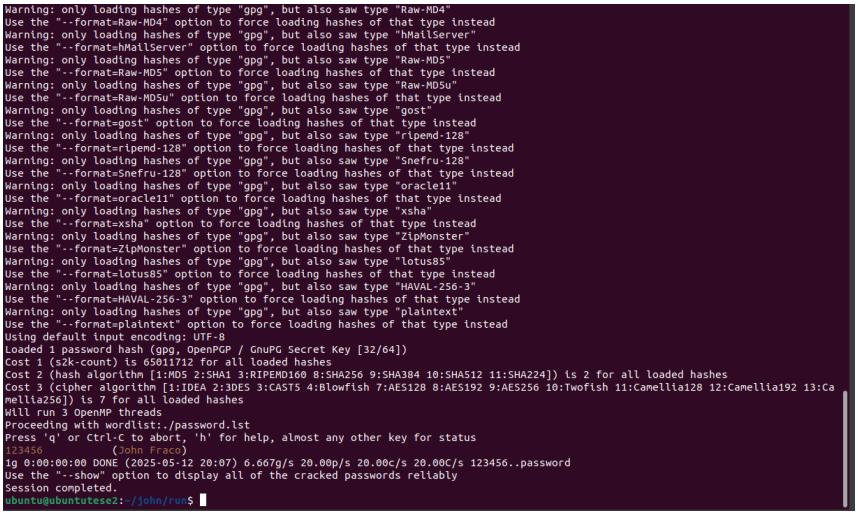

Figure 4.49: Recent GnuPG insecurity scenario (Final result of dictionary attack with John)

The screenshot shows John's output showing a discovered passphrase and possibly a successful decryption using the recovered password. What is done is that the attacker recovers the passphrase and can then decrypt affected files or export/use the key.

Chapter 5

Discussion

This chapter presents a critical reflection on the results obtained, analyzing the impact of the cryptographic choices, tools, and techniques addressed throughout the work. The advantages, limitations, and practical implications of each approach are discussed, relating the methods studied to real-world information security scenarios. Through this analysis, we seek to consolidate the knowledge acquired and identify strengths, weaknesses, and opportunities for improvement in the systems and algorithms evaluated.

5.1 OpenSSL Conclusion

This section summarizes the outcomes of the OpenSSL experiments, comparing how secure and insecure configurations behaved during certificate validation, protocol negotiation, and the Heartbleed attack.

- In the **misconfigured scenario**, the server accepts vulnerable connections (TLS 1.0, SSLv3), and the Heartbleed attack works.
- In the **well-configured scenario**, the server enforces TLS 1.2, which is **not affected** by Heartbleed, so the attack fails.

This shows that even when using a **vulnerable version of OpenSSL (1.0.1f)**, proper configuration can mitigate the attack!

5.2 Veracrypt Conclusion

This section reflects on the results obtained from the strong and weak VeraCrypt volumes, highlighting how password strength, keyfiles, and algorithmic choices influenced the feasibility of brute-force attacks.

- In the **weak volume**, protected with no keyfiles, weak password, Secure Hash Algorithm (SHA)-256, Advanced Encryption Standard (AES), an attacker with a dictionary attack like John the Ripper can figure out the password needed to mount the volume.

- In the **strong volume**, protected with keyfiles, strong password, SHA-512, AES cascade, **without the keyfiles**, the attacker cannot discover the master key needed to mount the volume.

5.3 GnuPG Conclusion

This section reviews the findings from the various GnuPG scenarios, contrasting secure configurations with weak keys, compromised keys, and outdated versions to assess real-world risks.

Scenario	**Protection?**	**Observation**
Strong (RSA 4096 + pass)	✓	Does not break
Weak (RSA 1024 + weak pass)	✗	Breaks quickly
Compromised key	✗	Decrypts effortlessly
Old GPG version (GnuPG 1)	✗	Obsolete algorithms

Table 5.1: Expected Result

5.4 Critical analysis: Why certain vulnerabilities persist

This section provides a broader interpretation of the experimental results, discussing the underlying reasons why misconfigurations and outdated cryptographic practices continue to appear in modern systems.

- **Legacy and compatibility**: Many systems maintain support for legacy protocols and ciphers (example: SSLv3, TLS 1.0) for compatibility with older clients, this prolongs exposure to known vulnerabilities. Testing with OpenSSL showed exactly that: **a vulnerable version can be mitigated by configuration**, but maintaining backward compatibility remains an operational risk.

- **Configuration decisions:** Repeated patterns of insecure configuration occur due to lack of training, time pressures, and complexity of options. Examples: **choosing weak passwords, lack of keyfiles, exporting keys without protection**. See the Veracrypt section for examples.
- **Economics and cost of updating:** Sometimes it is not economically feasible for organizations to update software on all systems, creating windows of exposure.
- **Vulnerable implementations and surface area:** Extensive library + C code (OpenSSL) increases the likelihood of implementation bugs (example: Heartbleed).

5.5 Implications for security policy

This section explores how the assessment results translate into organizational security requirements, outlining the policy measures needed to prevent misconfigurations and ensure safe cryptographic usage.

- **Patch management policy:** Establish Service Level Agreement (SLA) for applying critical updates.
- **Minimum configuration policy:** Define baselines (minimum protocols, allowed ciphers, key expirations).
- **Key and backup management:** Prohibit unaudited exports of private keys, secure storage of backups.
- **Training and periodic auditing:** Mandatory training on password creation, use of keyfiles, and export policies; regular audits (automatic and manual).

5.6 Recommended technical mitigations

This section presents practical and technical countermeasures that organizations can adopt to reduce exposure to cryptographic vulnerabilities and improve overall system resilience.

- Automate the detection of vulnerable versions and insecure configurations (configuration scanners and cipher suite checkers).
- Use of automated auditing tools (example: TLS configuration tests, header checks).
- Integration of Artificial Intelligence (AI)-assisted techniques for vulnerability prioritization and detection of misconfiguration patterns (example: clustering of insecure configurations).
- Establish procedures for key rotation, revocation, and incident response.

Chapter 6

Conclusion and Future Work

This thesis provided a comprehensive examination of cryptographic tools by combining a literature analysis with a practical assessment of three widely used implementations: OpenSSL, VeraCrypt, and GnuPG. The literature review highlighted the evolution of cryptographic research, the increasing emphasis on implementation security, and the persistent gap between theoretical guarantees and real-world usage. The practical experiments confirmed the critical impact of configuration choices, demonstrating that even robust cryptographic primitives can become vulnerable when deployed with weak parameters, outdated protocols, or misconfigured environments. Overall, the findings reinforce that security is not solely determined by the cryptographic algorithms themselves but by the correctness, maintenance, and operational context in which the tools are used. Future research can further expand this study by evaluating a broader set of cryptographic libraries and incorporating additional attack vectors, such as side-channel analysis and automated vulnerability-scanning pipelines. Conducting user-centric studies to understand how developers interact with these tools may also help reduce misconfigurations and improve overall security in practical deployments.

6.1 Limitations

While this research provides valuable insights into the assessment of cryptographic tools and the influence of configuration quality, several limitations should be noted. The experimental scope was restricted to three major tools, which, though representative, may not encompass the full variety of cryptographic implementations and configurations used in real-world systems. Attack scenarios focused on weak configurations and documented vulnerabilities, and new exploits or undiscovered flaws may affect results. Lastly, usability and organizational policy factors were discussed qualitatively, but further empirical research is necessary to fully understand how end-users and administrators interact with cryptographic tools and how that shapes exposure to misconfiguration risks.

References

[1] Sharmin Afrose. "Evaluation of Static Vulnerability Detection Tools With Java Cryptographic API Benchmarks". In: *IEEE Transactions on Software Engineering* (2023). URL: https://ieeexplore.ieee.org/document/9721567.

[2] Algazy. "Evaluation of the Strength and Performance of a New Hashing Algorithm Based on a Block Cipher". In: *International Journal of Electrical and Computer Engineering (IJECE)* (2023). URL: https://www.researchgate.net/publication/371208227_Evaluation_of_the_strength_and_performance_of_a_new_hashing_algorithm_based_on_a_block_cipher.

[3] Moreno Ambrosin. "On the Feasibility of Attribute-Based Encryption on Internet of Things Devices". In: *IEEE Micro* (2016). URL: https://arxiv.org/abs/1611.08098.

[4] Ross Anderson, Eli Biham, and Lars Knudsen. *Serpent Block Cipher*. https://www.cl.cam.ac.uk/~rja14/serpent.html. Accessed on: Nov. 23, 2025. 1998.

[5] I. Bertolotti. "Automatic Detection of Attacks on Cryptographic Protocols: A Case Study". In: *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. 2005. URL: https://link.springer.com/chapter/10.1007/11506881_5.

[6] D. Bogdanov. "Rmind: A Tool for Cryptographically Secure Statistical Analysis". In: *IEEE Transactions on Dependable and Secure Computing* (2018). URL: https://ieeexplore.ieee.org/document/7505613.

[7] Alexandre Braga. "Practical Evaluation of Static Analysis Tools for Cryptography: Benchmarking Method and Case Study". In: *IEEE International Symposium on*

Software Reliability Engineering. 2017. URL: https://ieeexplore.ieee.org/document/8109084.

[8] Rui Chang. "The Research on Cryptographic Algorithms Recognition Technology". In: *International Conference on Intelligent Systems Design and Engineering Applications*. 2013. URL: https://ieeexplore.ieee.org/document/6843544.

[9] Codenomicon. *The Heartbleed Bug*. https://heartbleed.com/. Accessed on: Nov. 23, 2025. 2014.

[10] B. Damjanovic. "Performance Evaluation of AES Algorithm Under Linux Operating System". In: *Proceedings of the Romanian Academy, Series A: Mathematics, Physics, Technical Sciences, Information Science* (2013). URL: https://acad.ro/sectii2002/proceedings/doc2013-2/13-Damjanovic.pdf.

[11] Sumit Singh Dhanda. "Lightweight Cryptography: A Solution to Secure IoT". In: *Wireless Personal Communications* (2020). URL: https://link.springer.com/article/10.1007/s11277-020-07134-3.

[12] Jean-Guillaume Dumas. "Dual Protocols for Private Multi-Party Matrix Multiplication and Trust Computations". In: *Computers & Security* (2017). URL: https://www.scitepress.org/Link.aspx?doi=10.5220/0005957200610072.

[13] Taher ElGamal. *ElGamal Public-Key Cryptosystem*. https://ieeexplore.ieee.org/document/1057074. Accessed on: Nov. 23, 2025. 1985.

[14] Bringel Filho. "PEARL: A Performance Evaluator of Cryptographic Algorithms for Mobile Devices". In: *Lecture Notes in Computer Science*. 2004. URL: https://link.springer.com/chapter/10.1007/978-3-540-30178-3_26.

[15] Riccardo Focardi. "A Comparison of Three Authentication Properties". In: *Theoretical Computer Science* (2003). URL: https://www.sciencedirect.com/science/article/pii/S0304397502003651?via%3Dihub.

[16] Miles Frantz. "Methods and Benchmark for Detecting Cryptographic API Misuses in Python". In: *IEEE Transactions on Software Engineering* (2024). URL: https://ieeexplore.ieee.org/document/10474052.

[17] Antonio Francesco Gentile. “A Performance Analysis of Security Protocols for Distributed Measurement Systems Based on Internet of Things with Constrained Hardware and Open Source Infrastructures”. In: *Italian National Conference on Sensors*. 2024. URL: https://www.mdpi.com/1424-8220/24/9/2781.

[18] I. González. “Ciphering Algorithms in MicroBlaze-Based Embedded Systems”. In: *IEE Proceedings - Computers and Digital Techniques* (2006). URL: https://digital-library.theiet.org/doi/10.1049/ip-cdt_20050131.

[19] F. Honecker. “Comparison of Distributed Tamper-Proof Storage Methods for Public Key Infrastructures”. In: *Future Internet* (2022). URL: https://www.mdpi.com/1999-5903/14/11/336.

[20] Saddam Hussain. “A Lightweight and Provable Secure Identity-Based Generalized Proxy Signcryption (IBGPS) Scheme for Industrial Internet of Things (IIoT)”. In: *Journal of Information Security and Applications* (2021). URL: https://www.sciencedirect.com/science/article/pii/S2214212620307900?via%3Dihub.

[21] IDRIX. *VeraCrypt*. https://www.veracrypt.io/. Accessed on: Nov. 23, 2025. 2025.

[22] Burton S. Kaliski and Jim Rusch. *PKCS 5: Password-Based Cryptography Specification Version 2.1*. RFC 8018. Accessed on: Nov. 23, 2025. Internet Engineering Task Force, 2017.

[23] N. K. Kasumov. “A Cryptographic Tool as an Emulator of Loss-Free Data Compression Procedures”. In: *Automatic Control and Computer Sciences* (2007). URL: https://link.springer.com/article/10.3103/S0146411607010087.

[24] Ju-Hwan Kim. “SIV: Raise the Correlation of Second-Order Correlation Power Analysis to 1.00”. In: *Applied Sciences* (2020). URL: https://www.mdpi.com/2076-3417/10/10/3394.

[25] Zhidan Li. “An Efficient ABE Scheme with Verifiable Outsourced Encryption and Decryption”. In: *IEEE Access* (2019). URL: https://ieeexplore.ieee.org/document/8598798.

[26] Houssem Maghrebi. “On the Use of Independent Component Analysis to Denoise Side-Channel Measurements”. In: *Lecture Notes in Artificial Intelligence*. 2018. URL: https://link.springer.com/chapter/10.1007/978-3-319-89641-0_4.

[27] Peter McLaren. “Deriving ChaCha20 Key Streams From Targeted Memory Analysis”. In: *Journal of Information Security and Applications* (2019). URL: https://www.sciencedirect.com/science/article/pii/S2214212619300997?via%3Dihub.

[28] Xiao Meihua. “On Formal Analysis of Cryptographic Protocols and Supporting Tool”. In: *Chinese Journal of Electronics* (2010). URL: https://ieeexplore.ieee.org/document/10151981.

[29] Mojisola. “An Improved Random Bit-Stuffing Technique with a Modified RSA Algorithm for Resisting Attacks in Information Security (RBMRSA)”. In: *Egyptian Informatics Journal* (2022). URL: https://www.sciencedirect.com/science/article/pii/S1110866522000135?via%3Dihub.

[30] Tilo Müller. “A Systematic Assessment of the Security of Full Disk Encryption”. In: *IEEE Transactions on Dependable and Secure Computing* (2015). URL: https://ieeexplore.ieee.org/document/6951337.

[31] National Institute of Standards and Technology. *Advanced Encryption Standard (AES)*. https://csrc.nist.gov/projects/block-cipher-techniques. Accessed on: Nov. 23, 2025. 2025.

[32] National Institute of Standards and Technology. *Elliptic Curve Cryptography (ECC)*. https://csrc.nist.gov/projects/elliptic-curve-cryptography. Accessed on: Nov. 23, 2025. 2025.

[33] N. Nedjah. “Massively Parallel Modular Exponentiation Method and Its Implementation in Software and Hardware for High-Performance Cryptographic Systems”. In: *IET Computers & Digital Techniques* (2012). URL: https://digital-library.theiet.org/doi/10.1049/iet-cdt.2011.0074.

[34] A. Ometov. “A Comprehensive and Reproducible Comparison of Cryptographic Primitives Execution on Android Devices”. In: *IEEE Access* (2021). URL: https://ieeexplore.ieee.org/document/9389752.

[35] OpenSSL Software Foundation. *OpenSSL*. https://www.openssl.org/. Accessed on: Nov. 23, 2025. 2025.

[36] Openwall Project. *John the Ripper Password Cracker*. https://www.openwall.com/john/. Accessed on: Nov. 23, 2025. 2025.

[37] Szilárd Pfeiffer. “D(HE)at: A Practical Denial-of-Service Attack on the Finite Field Diffie–Hellman Key Exchange”. In: *IEEE Access* (2024). URL: https://ieeexplore.ieee.org/document/10374117.

[38] David Pokorný. “Equivalent Keys: Side-Channel Countermeasure for Post-Quantum Multivariate Quadratic Signatures”. In: *Electronics* (2022). URL: https://www.mdpi.com/2079-9292/11/21/3607.

[39] I. Prots’ko. “Implementing Montgomery Multiplication to Speed-Up the Computation of Modular Exponentiation of Multi-Bit Numbers”. In: *Cybernetics and Systems Analysis* (2024). URL: https://link.springer.com/article/10.1007/s10559-024-00720-4.

[40] Nitin Pundir. “Power Side-Channel Leakage Assessment Framework at Register-Transfer Level”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2022). URL: https://ieeexplore.ieee.org/document/9780645.

[41] Malik Qasaimeh. “Software Randomness Analysis and Evaluation of Lightweight Ciphers: The Prospective for IoT Security”. In: *Multimedia Tools and Applications* (2018). URL: https://doi.org/10.1007/s11042-018-5663-8.

[42] Sazzadur Rahaman. “From Theory to Code: Identifying Logical Flaws in Cryptographic Implementations in C/C++”. In: *IEEE Transactions on Dependable and Secure Computing* (2022). URL: https://ieeexplore.ieee.org/document/9524495.

[43] F. Raji. “CP2: Cryptographic Privacy Protection Framework for Online Social Networks”. In: *Computers & Electrical Engineering* (2013). URL: https://www.sciencedirect.com/science/article/pii/S0045790612001681?via%3Dihub.

[44] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. *RSA Public-Key Cryptosystem*. https://people.csail.mit.edu/rivest/pubs.html. Accessed on: Nov. 23, 2025. 1978.

[45] RockYou Data Leak. *RockYou Password List*. https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt. Accessed on: Nov. 23, 2025. 2009.

[46] Bruce Schneier. *Twofish Encryption Algorithm*. https://www.schneier.com/academic/twofish/. Accessed: Nov. 23, 2025. 1998.

[47] Danilo Šijačić. "Towards Efficient and Automated Side-Channel Evaluations at Design Time". In: *Journal of Cryptographic Engineering* (2020). URL: https://link.springer.com/article/10.1007/s13389-020-00233-8.

[48] Cong Sun. "CryptoEval: Evaluating the Risk of Cryptographic Misuses in Android Apps with Data-Flow Analysis". In: *IET Information Security* (2021). URL: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/ise2.12117.

[49] P. Sun. "EasyBC: A Cryptography-Specific Language for Security Analysis of Block Ciphers Against Differential Cryptanalysis". In: *Proceedings of the ACM on Programming Languages*. 2024. URL: https://dl.acm.org/doi/10.1145/3632871.

[50] Tristen Teague. "Towards Cloud-Based Infrastructure for Post-Quantum Cryptography Side-Channel Attack Analysis". In: *IEEE Design Methodologies Conference (DMC)*. 2023. URL: https://ieeexplore.ieee.org/document/10412485.

[51] The GNU Privacy Guard Project. *GnuPG (GNU Privacy Guard)*. https://gnupg.org/. Accessed on: Nov. 23, 2025. 2025.

[52] Pengxu Tian. "EAFS: An Efficient, Accurate, and Forward Secure Searchable Encryption Scheme Supporting Range Search". In: *IEEE Systems Journal* (2022). URL: https://ieeexplore.ieee.org/document/9422409.

[53] Jevgenijus Toldinas. "Energy Efficiency Comparison with Cipher Strength of AES and Rijndael Cryptographic Algorithms in Mobile Devices". In: *Elektronika ir Elektrotechnika* (2011). URL: https://doi.org/10.5755/J01.EEE.108.2.134.

[54] Michael J. D. Vermeer. “Evaluating Cryptographic Vulnerabilities Created by Quantum Computing in Industrial Control Systems”. In: *Journal of Critical Infrastructure Policy* (2024). URL: https://onlinelibrary.wiley.com/doi/10.1002/jci3.12024.

[55] Lizhen Wang. “Intelligent Detection of Cryptographic Misuse in Android Applications Based on Program Slicing and Transformer-Based Classifier”. In: *Electronics* (2023). URL: https://www.mdpi.com/2079-9292/12/11/2460.

[56] Yaru Wang. “A Survey of Side-Channel Leakage Assessment”. In: *Electronics* (2023). URL: https://www.mdpi.com/2079-9292/12/16/3461.

[57] Deng Xiang. “A Secure and Efficient Certificateless Signature Scheme for Internet of Things”. In: *Ad Hoc Networks* (2022). URL: https://www.sciencedirect.com/science/article/pii/S1570870521002018?via%3Dihub.

[58] Dae Hyun Yum. “Lightweight One-Time Signature for Short Messages”. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* (2011). URL: https://www.jstage.jst.go.jp/article/transfun/E94.A/7/E94.A_7_1567/_article.

[59] Xin Zheng. “An Efficient and Low-Power Design of the SM3 Hash Algorithm for IoT”. In: *Electronics* (2019). URL: https://www.mdpi.com/2079-9292/8/9/1033.

[60] Chunning Zhou. “A New Approach of Evaluating the Security Against Differential and Linear Cryptanalysis and Its Applications to Serpent, NOEKEON and ASCON”. In: *The Computer Journal* (2024). URL: https://academic.oup.com/comjnl/article-abstract/67/1/274/6862009?redirectedFrom=fulltext&login=false.

Appendix A

Appendix A – OpenSSL Configurations

This appendix presents the configuration steps to install OpenSSL with screenshots. These were originally detailed in Section 4.1 of the report, now moved here for clarity and conciseness.

A.0.1 Download the vulnerable version of OpenSSL

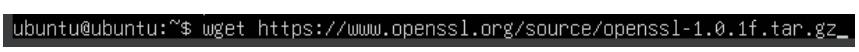

Figure A.1: Command to download the vulnerable version of OpenSSL

A.0.2 Configure the vulnerable version of OpenSSL

Figure A.2: Command to configure the vulnerable version of OpenSSL

A.0.3 Compile the vulnerable version of OpenSSL

Figure A.3: Command to compile the vulnerable version of OpenSSL

A.0.4 Install the vulnerable version of OpenSSL

Figure A.4: Command to install the vulnerable version of OpenSSL

A.0.5 Check if the vulnerable version of OpenSSL it was installed correctly and is active

Figure A.5: Command to check if the vulnerable version of OpenSSL it was installed correctly and is active

Appendix B

Appendix B – VeraCrypt Configurations

This appendix presents the configuration steps and screenshots related to the creation of both strong and weak volumes in VeraCrypt. These were originally detailed in Section 4.2 of the report, now moved here for clarity and conciseness.

B.0.1 Creating the secure Volume (Strong Configuration) Step 1

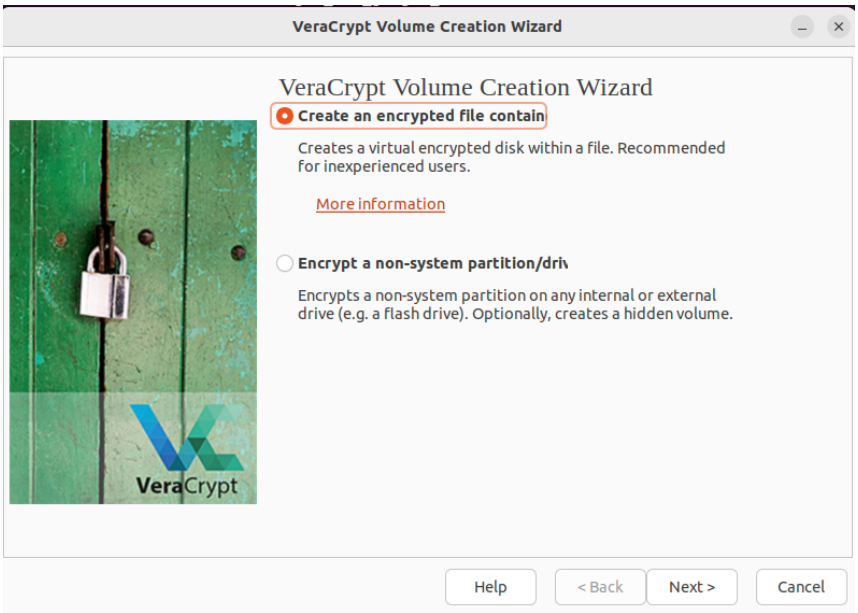

Figure B.1: Veracrypt Volume Creation Wizard

B.0.2 Creating the secure Volume (Strong Configuration) Step 2

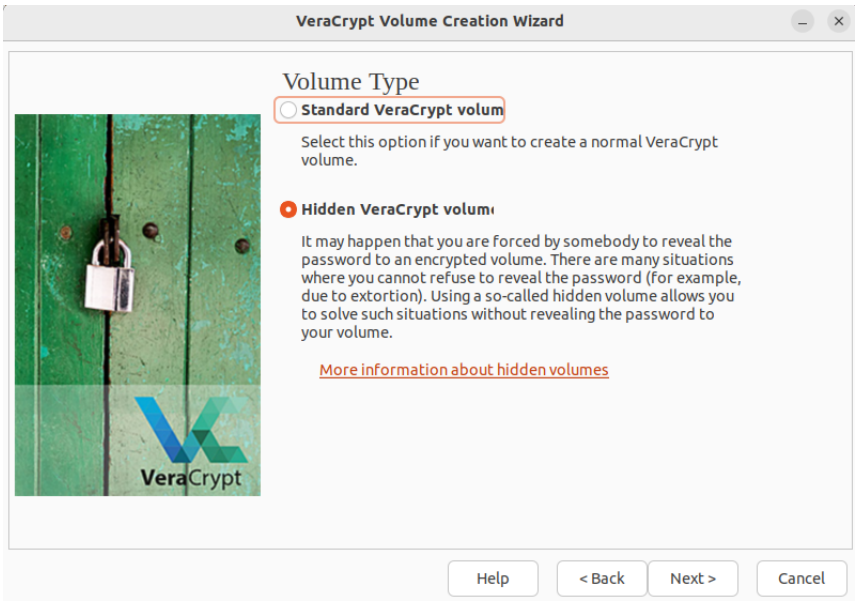

Figure B.2: Volume Type

B.0.3 Creating the secure Volume (Strong Configuration) Step 3

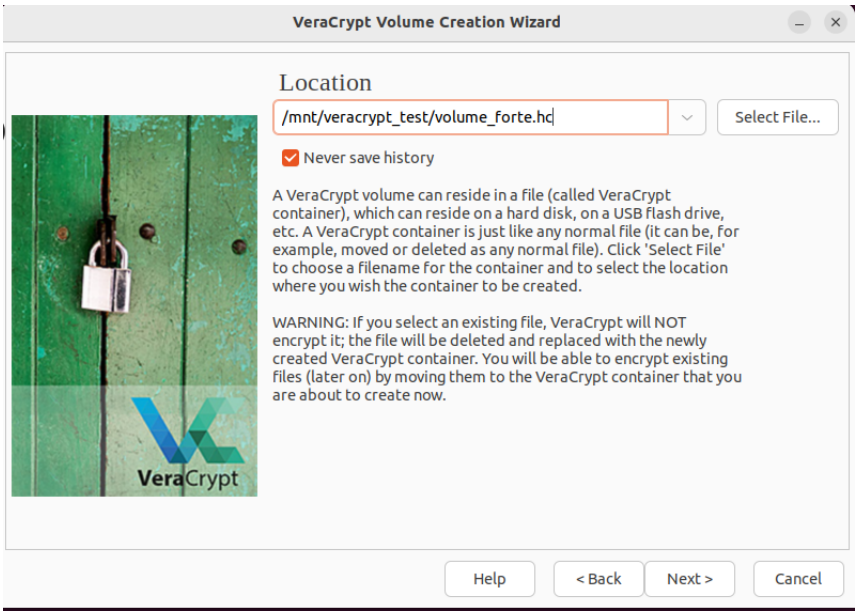

Figure B.3: Location of the file

B.0.4 Creating the secure Volume (Strong Configuration) Step 4

Figure B.4: Hidden Volume Encryption Options

B.0.5 Creating the secure Volume (Strong Configuration) Step 5

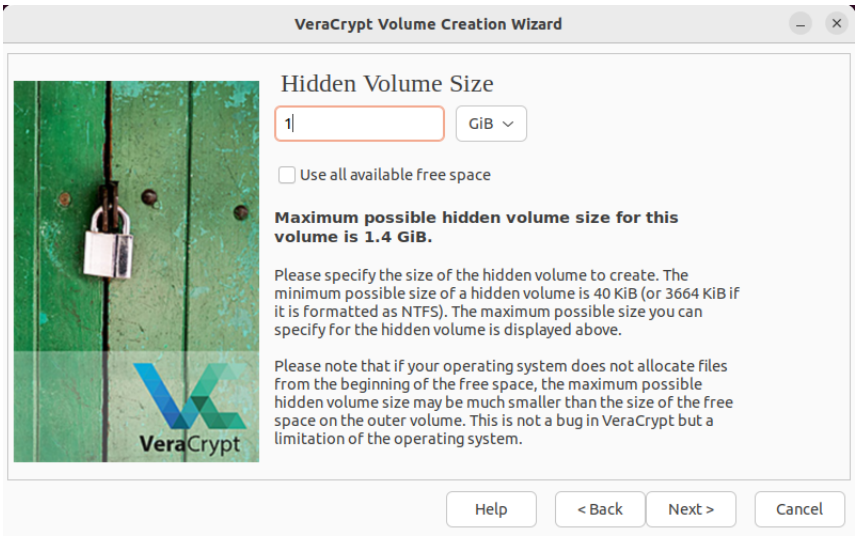

Figure B.5: Hidden Volume Size

B.0.6 Creating the secure Volume (Strong Configuration) Step 6

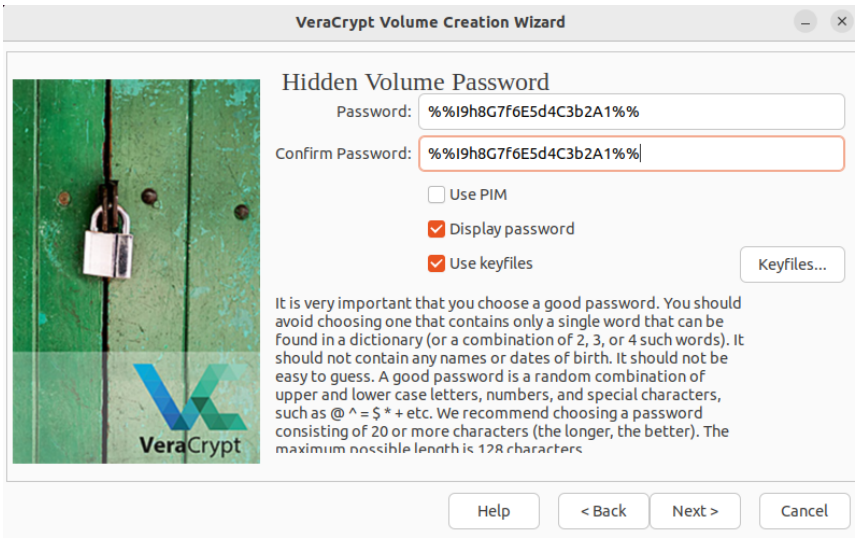

Figure B.6: Hiden Volume Password

B.0.7 Creating the secure Volume (Strong Configuration) Step 7

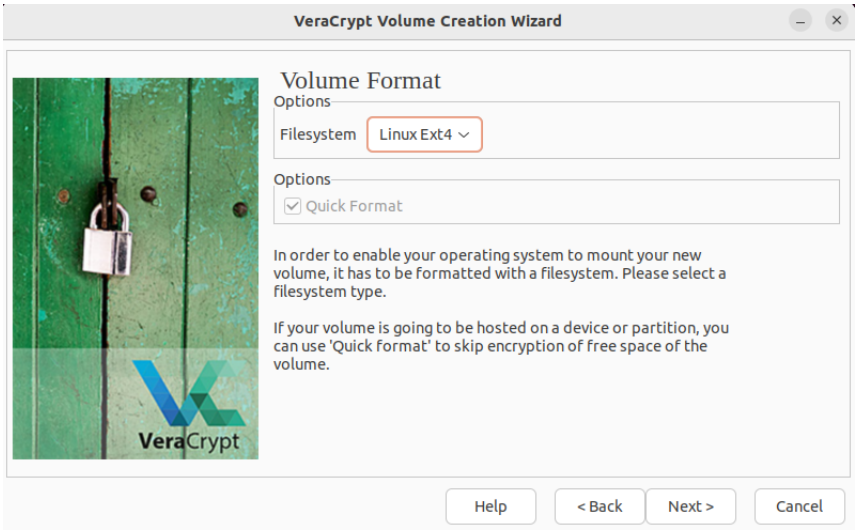

Figure B.7: Volume Format

B.0.8 Creating the secure Volume (Strong Configuration) Step 8

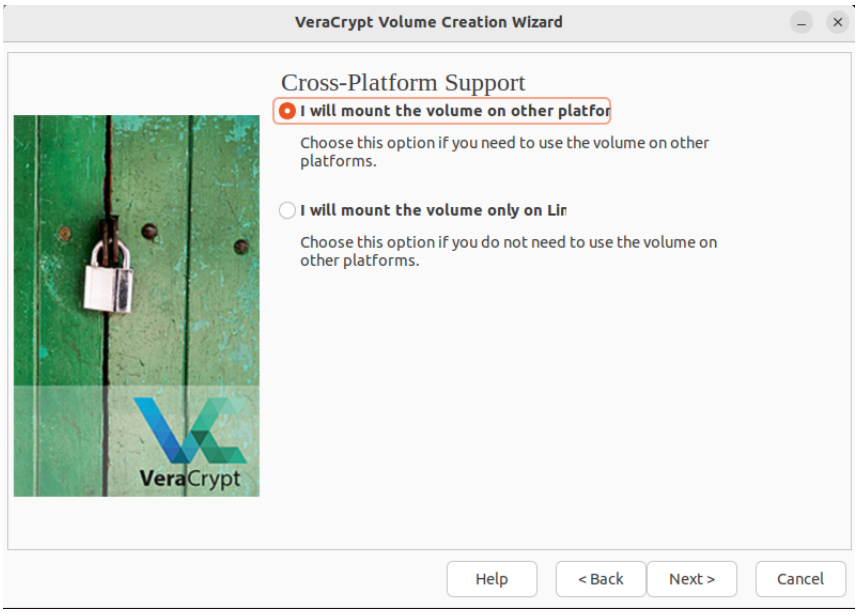

Figure B.8: Cross-Platform Support

B.0.9 Creating the secure Volume (Strong Configuration) Step 9

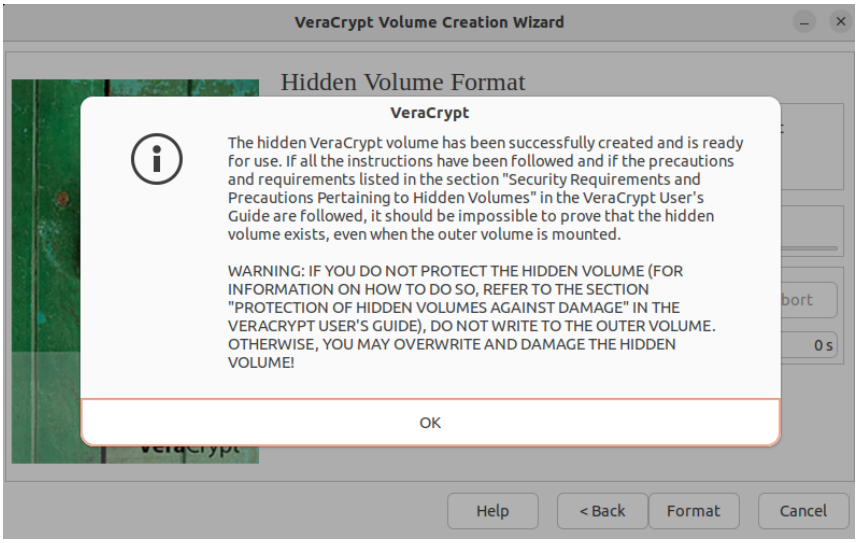

Figure B.9: Hidden Volume Format

B.0.10 Creating the insecure Volume (Weak Configuration) Step 1

Figure B.10: Veracrypt Volume Creation Wizard

B.0.11 Creating the insecure Volume (Weak Configuration) Step 2

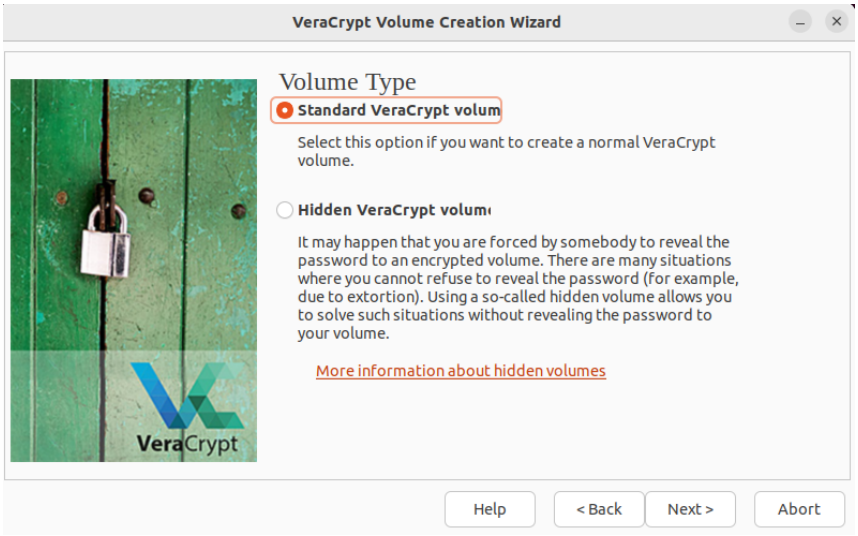

Figure B.11: Volume Type

B.0.12 Creating the insecure Volume (Weak Configuration) Step 3

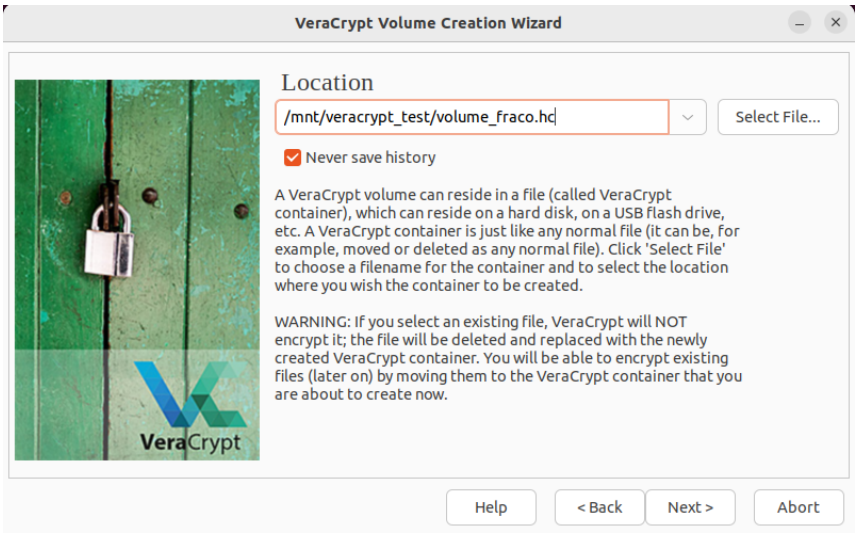

Figure B.12: Location of the File

B.0.13 Creating the insecure Volume (Weak Configuration) Step 4

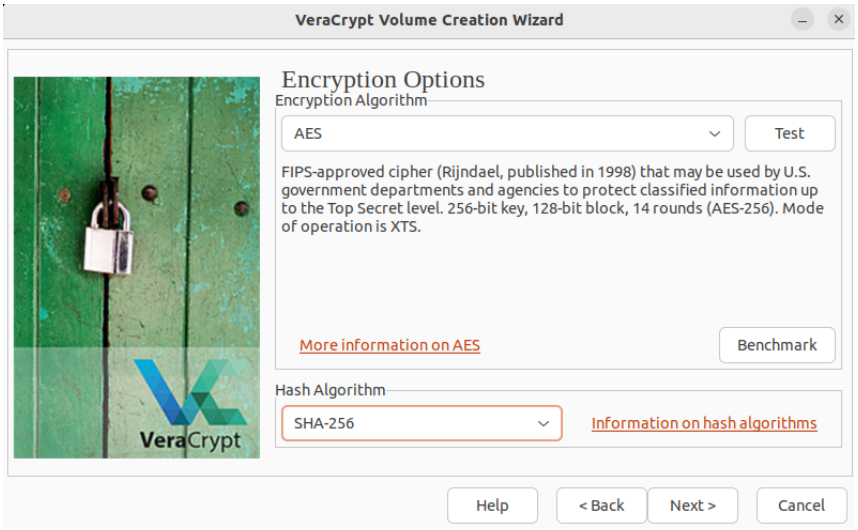

Figure B.13: Volume Encryption Options

B.0.14 Creating the insecure Volume (Weak Configuration) Step 5

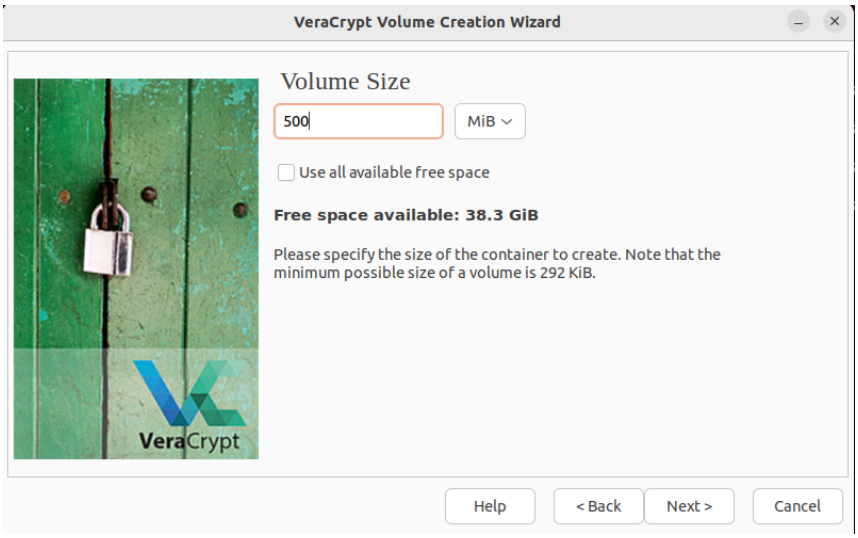

Figure B.14: Volume Size

B.0.15 Creating the insecure Volume (Weak Configuration) Step 6

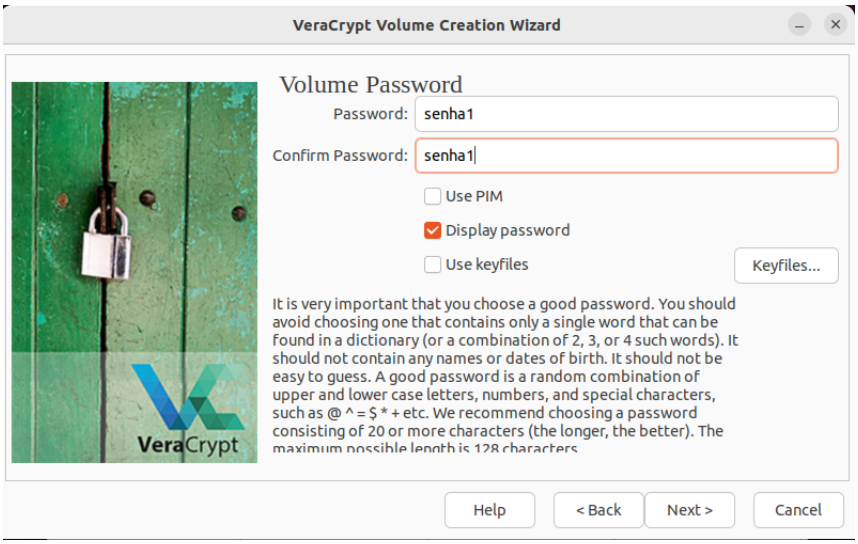

Figure B.15: Volume Password

B.0.16 Creating the insecure Volume (Weak Configuration) Step 7

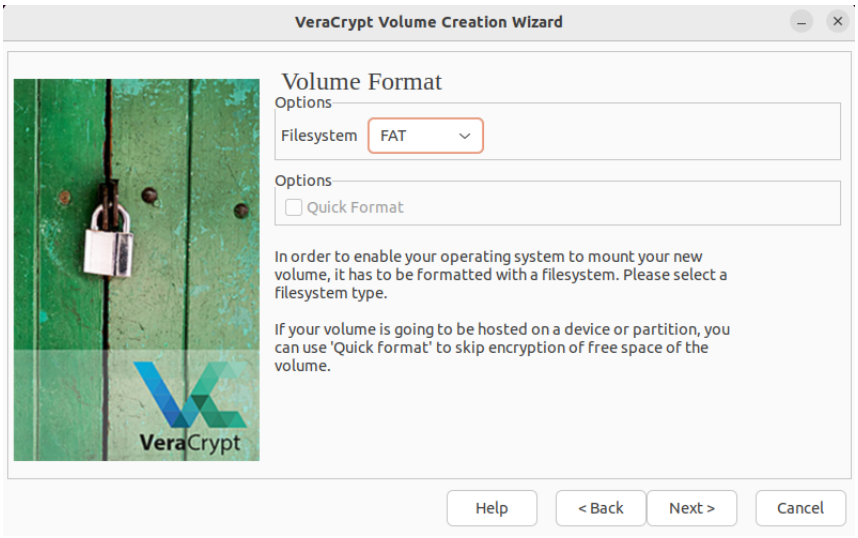

Figure B.16: Volume Format

B.0.17 Creating the insecure Volume (Weak Configuration) Step 8

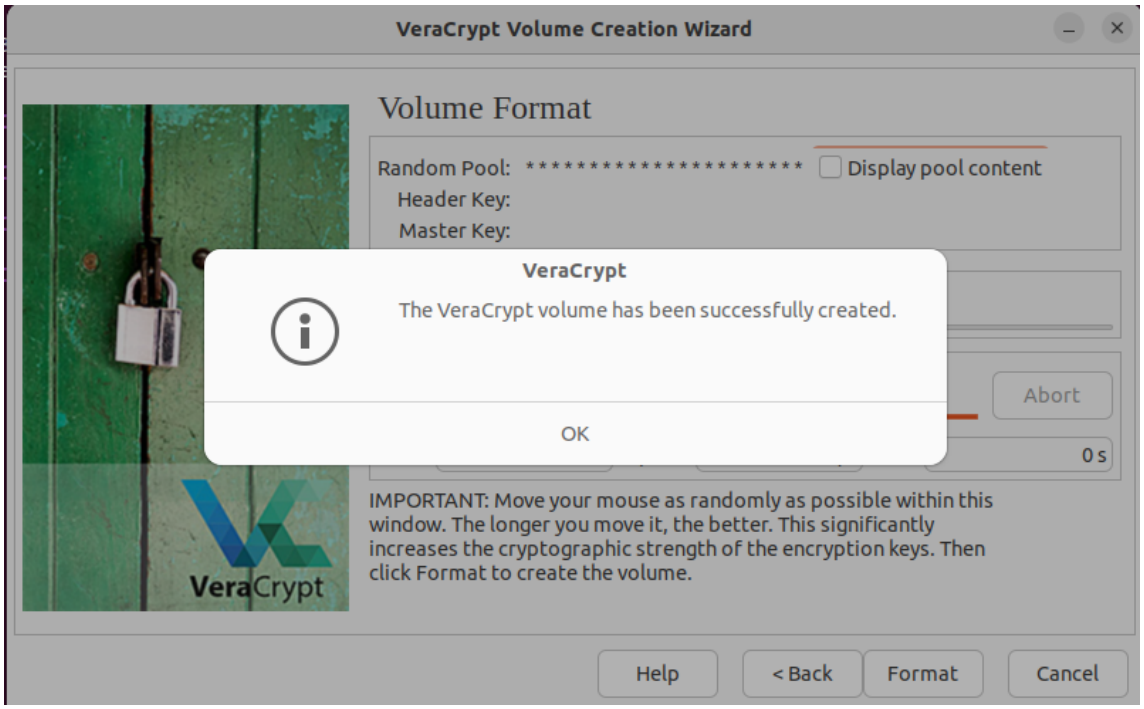

Figure B.17: Volume Creation: Done

Appendix C

Appendix C – GnuPG Configurations

This appendix presents the configuration steps to install GnuPG with screenshots. These were originally detailed in Section 4.3 of the report, now moved here for clarity and conciseness.

C.0.1 Installation of the latest version of GnuPG and verification

```
ubuntu@ubuntutese2:~$ sudo apt install gnupg
[sudo] password for ubuntu:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gnupg is already the newest version (2.2.27-3ubuntu2.3).
gnupg set to manually installed.
The following package was automatically installed and is no longer required:
  john-data
Use 'sudo apt autoremove' to remove it.
0 to upgrade, 0 to newly install, 0 to remove and 2 not to upgrade.
ubuntu@ubuntutese2:~$ ls
brute_completo_vc.sh  Desktop    hashcat-utils  Music     rockyou.txt  Templates            Videos
brute_rockyou_vc.sh   Documents  hash_fraco.vc  Pictures  snap         test_dict.txt
brute_vc.sh           Downloads  john-jumbo     Public    src          veracrypt2hashcat.py
ubuntu@ubuntutese2:~$ gpg --version
gpg (GnuPG) 2.2.27
libgcrypt 1.9.4
Copyright (C) 2021 Free Software Foundation, Inc.
License GNU GPL-3.0-or-later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /home/ubuntu/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cypher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
ubuntu@ubuntutese2:~$
```

Figure C.1: Installation of the latest version of GnuPG and verification