CSC-340

Cryptographic Analysis of Data Recovery on Encrypted Systems

Jack Sabol, Brendan Cordwell, and Tanmay Agarwal

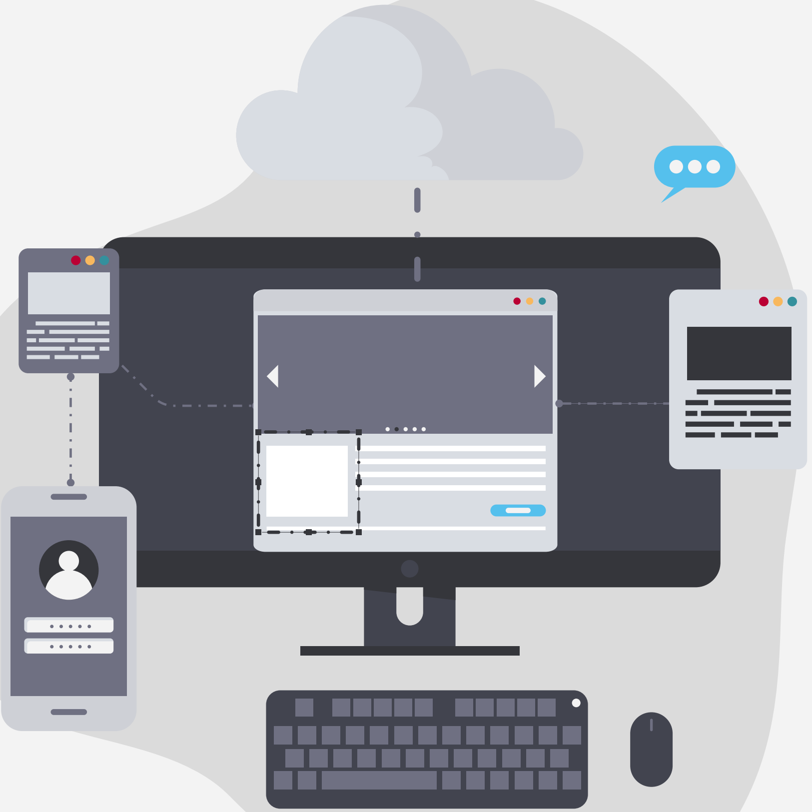


Table of Contents

01 Introduction

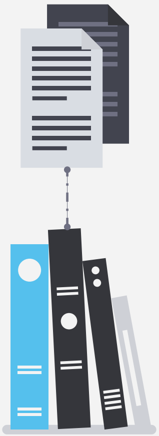
02 Research Questions

03 Methodology

04 Implementation Details

05 Demonstration

06 Results & Conclusion



01 | Introduction

- Encryption protects sensitive data by making it unreadable without the correct key.
- Systems like LUKS (Linux) and BitLocker (Windows) secure storage using strong cryptographic methods.
- Even with encryption, weak passwords, poor configurations, and old algorithms can expose vulnerabilities.

Purpose: This project analyzes how encryption works, where weaknesses appear, and when encrypted data can be recovered.

02 | Research Questions

01

What weaknesses exist in encryption systems like LUKS and BitLocker?

02

How can encrypted data be recovered when these weaknesses are found?

03

What steps can help make encryption stronger and more secure?

03 | Methodology

Experimenting & Testing

- Created virtual machines running Kali Linux and Windows 11
- Encrypted disks with LUKS and BitLocker using both strong and intentionally weak settings
- Used synthetic test data to avoid exposing sensitive information
- Simulated weaknesses: simple passwords, lower iteration counts, outdated configurations

Detection & Recovery Attempts

- Ran password-recovery tools (John the Ripper, Hashcat) using dictionary, hybrid, and brute-force attacks.
- Captured VM snapshots to analyze RAM for key material or decrypted data fragments.
- Used Autopsy to examine decrypted disk images and check for partially recovered files.

Analysis

- Compared success rates of attacks against different configurations.
- Evaluated LUKS vs. BitLocker resistance to cracking and key extraction.
- Identified which settings and passwords made systems vulnerable.

04 | Implementation Details

System Setup

- **Two VMs:** one Linux (LUKS), one Windows (BitLocker).
- Created:
  - Strong LUKS config (AES-XTS-512, long passphrase, high PBKDF2 iterations).
  - Weak LUKS config (short passphrase, reduced iterations).
  - BitLocker TPM-only mode and password-only mode.
- Added synthetic user files for realistic testing

Tools

- **Hashcat & John the Ripper:** password-cracking attempts
- **RAM snapshot analysis:** looked for decrypted data or key remnants
- **Autopsy:** inspected disk images after recovery attempts

Testing Scenarios

- Weak vs. strong passwords
- Old vs. modern encryption settings
- BitLocker TPM-only vs. password-only
- System states: unlocked, locked, suspended, shutdown

05 | Demonstration

06 | Results & Conclusions

Results

- Weak passwords were cracked quickly
  - Dictionary passwords: cracked within minutes.
  - Short brute-force passwords: cracked within several hours.
  - 12–14+ character passphrases: no measurable progress during testing.
- LUKS performance depended on iteration counts
  - Higher PBKDF2 iterations → much slower guessing rate.
  - Lower iteration counts → fast enough to make brute-force practical.
- BitLocker password-only mode allowed more guesses
  - BitLocker's hashing allowed more guesses per second than LUKS.
  - Weak BitLocker passwords were noticeably easier to crack.
- TPM-only BitLocker blocked password attacks
  - No password-derived material → password attacks impossible.
  - If unlocked, the system gave full access, so physical security mattered.

06 | Results & Conclusions

- Memory analysis
    - Unlocked systems: partial key material or decrypted data could appear.
    - Suspended systems: sometimes showed more data due to RAM preservation.
    - Full shutdown: no key material recovered on either LUKS or BitLocker.
    - Hibernation files were encrypted → no useful remnants found.

- Weak or incorrect configurations caused most failures
    - Low LUKS iterations + weak passwords → successful recovery.
    - Strong passphrases + strong settings → no successful attacks.

Conclusion

- LUKS and BitLocker are secure when properly configured.
- Most vulnerabilities came from weak passwords and poor settings, not the encryption algorithms.
- Strong passphrases, modern configurations, and securing active devices are the best defenses.

References

[1] L. Gasser and I. Aad, “Disk, File and Database Encryption,” in Trends in Data Protection and Encryption Technologies, Springer, 2023, pp. 201–207. doi: 10.1007/978-3-031-33386-6 33.

[2] B. A. Sassani, “Evaluating Encryption Algorithms for Sensitive Data Using HDD, SSHD and SSD Based NAND MLC Flash Memory,” Scientific World Journal, vol. 2020, Article ID 6132312, 2020. doi:10.1155/2020/6132312.

[3] C. Maartmann-Moe, S. E. Thorkildsen, and A. Aarnes, “The Persistence of Memory: Forensic Identification and Extraction of Cryptographic Keys,” Digital Investigation, vol. 6, pp. S132–S140, 2009. doi: 10.1016/j.diin.2009.06.002.

[4] H. Afzal, M. Yousaf, H. Afzal, N. Alharbe, and M. R. Mufti, “Cryptographic Strength Evaluation of Key Schedule Algorithms,” Security and Communication Networks, vol. 2020, Article ID 3189601, 2020. doi: 10.1155/2020/3189601.

[5] S. B. Alkhadhr, “Cryptography and randomization to dispose of data and protect confidentiality,” Cogent Engineering, vol. 4, no. 1, 2017. doi: 10.1080/23311916.2017.1300049.

[6] T. Gross, M. Busch, and T. Muller, “One Key to Rule Them All: Recovering the Master Key from RAM to Break Android’s File-Based Encryption,” Forensic Science International: Digital Investigation, vol. 36, p. 301113, 2021. doi: 10.1016/j.fsidi.2021.301113

THANK YOU!