Password Cracker for Forensic Investigations

# A Digital Forensic Tool for Hash Cracking Using Dictionary Attacks

## Prepared by: Suhaib Rehaman Date: July 26, 2025



# Contents

1. [Introduction](#_bookmark0) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
2. [Literature Review](#_bookmark1) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
3. [Implementation](#_bookmark2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
   1. [System Architecture](#_bookmark3) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
   2. [Code Snippet](#_bookmark4) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
   3. [Dependencies](#_bookmark5) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
4. [Workflow](#_bookmark6) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
5. [Advantages and Disadvantages](#_bookmark7) . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   1. [Advantages](#_bookmark8) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   2. [Disadvantages](#_bookmark9) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
6. [Uses](#_bookmark10) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
7. [Conclusion](#_bookmark12) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
8. [Summary](#_bookmark13) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

# Introduction

In the realm of digital forensics, recovering access to encrypted systems or files is a critical task. Password cracking, a technique used to retrieve passwords from their hashed representations, plays a pivotal role in forensic investigations. This report presents a Password Cracker for Forensic Investigations, a Python-based tool designed to crack MD5 and SHA-1 hashes using dictionary attacks. The tool is tailored for forensic use, emphasizing audit-ability, reliability, and ethical considerations.

The objective is to provide a beginner-friendly yet professional-grade tool that demon strates key forensic concepts while being extensible for advanced applications. This report outlines the tool’s development, implementation, workflow, advantages, disadvantages, practical uses, and conclusions, supported by a literature review of password cracking techniques.

# Literature Review

Password cracking has been extensively studied in cybersecurity and digital forensics. Early works, such as Password Cracking: A Survey (ACM, 2016), highlight the evolution of cracking techniques from brute-force to dictionary attacks. Dictionary attacks, which compare hashed passwords against a precomputed wordlist, are effective for cracking weak passwords due to their speed and reliance on common user behaviors.

Modern tools like John the Ripper and Hashcat leverage optimized algorithms and GPU acceleration to crack hashes eﬀiciently. Research by Kharrazi et al. (2006) emphasizes the vulnerability of MD5 and SHA-1 due to collision attacks, underscoring the need for forensic tools to target these legacy hashes. Recent studies, such as Analysis of Dictionary Attacks (IEEE, 2018), advocate for hybrid attacks combining dictionary and brute-force methods to improve success rates.

This project draws inspiration from open-source tools and academic research, focusing on dictionary attacks for simplicity and forensic applicability. Unlike professional tools, it prioritizes auditability and ease of use, making it ideal for educational and investigative purposes.

# Implementation

The Password Cracker is implemented in Python 3 using the hashlib library for hashing. The tool supports MD5 and SHA-1 hashes and performs dictionary attacks using a user- provided wordlist. Key features include input validation, forensic logging, and detailed reporting.

## System Architecture

* + - Input Validation: Uses regular expressions to verify hash formats (32 characters for MD5, 40 for SHA-1).
    - Dictionary Attack: Computes hashes of wordlist entries and compares them to the target hash.
    - Logging: Records all attempts in a timestamped log file for auditability.
    - Reporting: Generates a forensic report detailing the hash, status, and results.

## Code Snippet

Listing 1: Core Dictionary Attack Function

1 def d i c t i o n a r y\_a tta c k ( target\_hash , hash\_type , wordlist\_path ) :

2 i f not os . path . e x i s t s ( wordlist\_path ) :

3 return {” s ta tu s ” : ” e r r o r ” , ” message ” : f ” Wordlist ␣ f i l e ␣not␣ found : ␣{ wordlist\_path }”}

4 try :

5

6

7

8

9

10

11

12

13

14

15

16

17

with open ( wordlist\_path , ’ r ’ , encoding=’ utf - 8 ’ , e r r o r s=’ i g n o r e ’ ) as f :

f o r l i n e in f :

password = l i n e . s t r i p ( ) i f not password :

continue

computed\_hash = hash\_password ( password , hash\_type )

l o g g i n g . i n f o ( f ” Attempting␣password : ␣{ password }␣( Hash

: ␣{ computed\_hash }) ” )

i f computed\_hash == target\_hash . lower ( ) :

return {” hash ” : target\_hash , ” hash\_type ” : hash\_type ,

” password ” : password , ” s ta tu s ” : ” s u c c e s s ” ,

” message ” : f ” Password␣found : ␣{ password }”

}

return {” hash ” : target\_hash , ” hash\_type ” : hash\_type ,

” s ta tu s ” : ” f a i l e d ” , ” message ” : ” Password␣not␣found ␣ in ␣ w o r d l i s t . ”}

18 except Exception as e :

19 return {” s ta tu s ” : ” e r r o r ” , ” message ” : f ” Error : ␣{ s t r ( e ) }”}

## Dependencies

* + - Python 3.x with standard libraries (hashlib, argparse, logging, re).
    - A wordlist file (e.g., rockyou.txt).

# Workflow

The tool follows a structured forensic workflow to ensure reliability and auditability.

1. Input Collection: The investigator provides the target hash, hash type (MD5 or SHA-1), and wordlist path via command-line arguments.
2. Validation: The tool verifies the hash format and wordlist existence.
3. Dictionary Attack: Hashes each wordlist entry and compares it to the target hash.
4. Logging: Records all attempts, including timestamps and computed hashes.
5. Reporting: Generates a detailed report saved as a text file, summarizing the out- come.
6. Audit Trail: Maintains a log file for forensic documentation.

Example Command:

python password\_cracker.py 5f4dcc3b5aa765d61d8327deb882cf99 md5 rockyou.txt Sample Output:

=== Forensic Password Cracker Report === Hash: 5f4dcc3b5aa765d61d8327deb882cf99 Hash Type: md5

Status: Success

Message: Password found: password Cracked Password: password

=== End of Report ===

# Advantages and Disadvantages

## Advantages

* + - Forensic Auditability: Comprehensive logging ensures all actions are traceable.
    - Simplicity: Uses standard Python libraries, requiring no external dependencies.
    - Ethical Design: Includes warnings for authorized use, aligning with forensic stan- dards.
    - Extensibility: Supports adding new hash types or attack methods (e.g., brute- force).

## Disadvantages

* + - Limited Hash Types: Currently supports only MD5 and SHA-1.
    - Wordlist Dependency: Success depends on the quality and size of the wordlist.
    - Performance: Single-threaded dictionary attacks are slow for large word-lists.
    - No Brute-Force: Lacks brute-force capabilities, limiting its scope.

# Uses

The Password Cracker is designed for forensic investigations with proper authorization. Practical applications include:

* Data Recovery: Recovering passwords for encrypted files or systems in criminal investigations.
* Evidence Analysis: Accessing locked user accounts on seized devices.
* Educational Training: Teaching forensic students about hashing and cracking tech niques.
* Security Auditing: Testing the strength of legacy hashes in organizational systems.

# Conclusion

The Password Cracker for Forensic Investigations is a robust, beginner-friendly tool that demonstrates key forensic concepts through dictionary-based hash cracking. By support ing MD5 and SHA-1 hashes, it addresses common scenarios in digital investigations while maintaining auditability and ethical considerations. The project serves as a foundation for advanced forensic tools, with potential for enhancements like brute-force attacks or additional hash types.

This tool bridges the gap between academic learning and professional practice, offering a practical solution for password recovery in authorized forensic contexts. Future work could integrate advanced cracking methods and optimize performance to handle larger datasets.

# Summary

This report detailed the development of a Python-based Password Cracker for Foren sic Investigations. The tool uses dictionary attacks to crack MD5 and SHA-1 hashes, supported by forensic logging and reporting. Key sections included:

* Introduction: Outlined the tool’s purpose and forensic relevance.
* Literature Review: Reviewed password cracking techniques and tools.
* Implementation: Described the Python code and architecture.
* Workflow: Explained the step-by-step forensic process.
* Advantages and Disadvantages: Evaluated strengths and limitations.
* Uses: Highlighted practical forensic applications.
* Conclusion: Summarized findings and future potential.