International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Cipher Break – Digital Evidence Decryption and Management System

Vivek Mahyavanshi
Department of Computer Science
Parul Institute of Engineering and Technology
Vadodara, Gujarat
2203031260126@paruluniversity.ac.in

Anagha
Department of Computer Science
Parul Institute of Engineering and Technology
Vadodara, Gujarat
2203031260008@paruluniversity.ac.in

Aditi Agrawal
Department of Computer Science
Parul Institute of Engineering and Technology
Vadodara, Gujarat
2203031260006@paruluniversity.ac.in

Swathi Kolapudi
Department of Computer Science
Parul Institute of Engineering and Technology
Vadodara, Gujarat
2203031260100@paruluniversity.ac.in

***Abstract*—Handling encrypted digital evidence creates two recurring bottlenecks: accessing password-protected files and, once accessed, documenting and controlling every subsequent action. Many available tools focus mainly on decryption and then leave investigators to record steps manually, which can introduce gaps in traceability. To address this, we developed Cipher Break, a system that combines password recovery with end-to-end evidence activity tracking—from initial upload through report generation. Implemented using Python (Flask) and PostgreSQL, Cipher Break supports both dictionary and brute-force attacks and executes jobs across multiple threads to improve throughput while preserving correctness. Each decryption attempt, verification event, and access action is automatically logged to maintain a continuous chain of custody. Security is strengthened through role-based access control, SMS-based login verification, and SHA-256 integrity checks to detect unauthorized modification. Feedback from forensic practitioners and law-enforcement users informed interface refinements, clearer audit trails, and more granular visibility controls. Overall, Cipher Break provides a practical Digital Evidence Decryption and Management System that is fast when required, strict where it matters, and simple to operate in day-to-day investigative work.**

Index Terms—Digital evidence, forensic decryption, Flask, chain of custody, tamper detection, secure audit logs, investigation workflow, practitioner feedback.

I. INTRODUCTION

In many digital investigations, a major challenge is not only identifying relevant artifacts, but also accessing encrypted or password-protected content and then demonstrating that the evidence remained unchanged after access. Encrypted drives, protected folders, and compressed archives are now common in cybercrime investigations.

Investigators increasingly encounter multiple evidence items secured with different protection mechanisms, which can slow analysis and make it difficult to prove authenticity after decryption. In practice, many tools support either password recovery or evidence storage, but not both in an integrated and auditable way. This limitation motivated the development of the Digital Evidence Decryption and Management System (DEDMS), referred to in this work as Cipher Break. The objective is to combine decryption with post-access evidence handling so that recovery, verification, and documentation are treated as one continuous workflow. Evidence is processed in independent blocks so that one block does not depend on another, enabling parallel execution and reducing the risk of overlap during multi-threaded operations. Decryption proceeds only when a correct key is supplied, avoiding forced actions that could compromise integrity. After access is obtained, the system performs integrity verification and records every access or update event. These audit logs support chain-of-custody requirements that are essential for forensic defensibility. In addition, the platform is designed to accommodate evolving security needs by maintaining and updating key security controls over time. Overall, this paper presents a practical framework that unifies decryption, verification, and evidence management to support faster investigations while remaining

transparent, traceable, and suitable for real-world forensic use.

II. LITERATURE SURVEY

Password cracking - how it actually plays out:

In practical investigations, password recovery typically relies on two established approaches. Brute-force search systematically enumerates candidate combinations until the correct password is found [1], [6]. While effective, it becomes computationally expensive as password length and character complexity increase, and execution time can extend significantly [6]. Dictionary attacks instead prioritize likely passwords using curated wordlists (e.g., common words, leaked password sets, and suspect-specific variants), often producing faster results for real-world passwords [1], [11]. In many workflows, dictionary attacks are attempted first, with brute force used as a fallback when the wordlist does not succeed [1], [6]. Most forensic toolchains support both methods; Cipher Break exposes these options directly so investigators can select an approach without switching tools [3].

Existing tools - powerful but fiddly:

Several mature password-recovery tools are widely used in practice, notably John the Ripper and Hashcat [2], [10]. Although these tools are highly capable, they are primarily designed for command-line operation and require careful configuration for each evidence type. Typical setup tasks include selecting the correct hash mode, choosing and tuning wordlists and rules, and chaining multiple commands to support different file formats [6]. In early trials, a substantial amount of effort was spent scripting repetitive steps and validating configurations rather than running recovery itself. Cipher Break was designed to reduce this operational friction by integrating established cracking engines behind a web-based interface, providing sensible defaults, and centralizing forensic logging so that investigators do not need to manually track configuration details across separate tools [5].

Usability and Workflow Gaps:

While prior work often evaluates password recovery in terms of speed and success rate, fewer studies address the workflow complexity that arises when investigators manage multiple protected items concurrently [6]. In operational settings, teams may maintain progress notes in separate spreadsheets or notebooks to track which methods were attempted and when [4]. Such manual tracking can create accountability gaps and increases the likelihood of incomplete documentation. Cipher Break targets this usability gap by combining automation with consistent file-level tracking, clear audit logs, and real-time status updates that reduce reliance on manual record keeping [4], [8].

Evidence Handling, Chain-of-Custody, and Reporting:

In digital forensics, obtaining access to a protected file is only part of the problem; investigators must also demonstrate that the evidence was not altered after access was achieved [4], [7]. Chain-of-custody documentation and accurate timestamping are therefore as critical as the recovered content itself. Without reliable logs, evidence may lose credibility during legal review [4], [7]. During early testing, it was observed that teams often rely on separate software tools or manual sign-offs to confirm authenticity. In Cipher Break, every upload, view, and modification-related event is automatically recorded with user identity, time, and file details to support traceability [4].

Each generated report includes a SHA-256 integrity value, enabling investigators to demonstrate that the decrypted output matches the originally ingested evidence and has not been tampered with after entry [9]. In this way, the system functions as both a password-recovery platform and a structured forensic logbook.

Motivation and Contribution of Cipher Break:

Cipher Break was developed to address a recurring limitation in existing toolchains: the lack of an integrated bridge between decryption activities and digital evidence management [5]. Rather than executing command-line utilities individually, users can upload evidence, select a recovery method (brute force, dictionary), and allow the system to manage execution and documentation [1], [6], [11]. The report generator consolidates timestamps, user actions, hash verifications, and case summaries into downloadable PDF outputs [4], [5]. The overall intent is to make password recovery and evidence tracking accessible to less experienced users while providing the auditability required for professional forensic practice. Ultimately, Cipher Break supports not only recovery of protected data but also defensible evidence handling that helps establish trust in the results [4], [7].

III. METHODOLOGY

Cipher Break was designed with a practical objective: provide a single environment to ingest locked evidence, perform password recovery, and continuously demonstrate that the evidence remained unchanged throughout processing. This section describes the end-to-end workflow—covering authentication, upload, recovery execution, integrity verification, and report generation—using the interfaces and diagrams shown in Figs. 1–7 and Table I.

The description below follows the same sequence reflected in the submitted flowchart and user interfaces (Figs. 1–7) and the comparative summary in Table I.

Overview (Fig. 1):

The workflow is intentionally simple and aligns with common investigative practice: authenticate → upload evidence → select a recovery method → execute cracking jobs → verify integrity → generate reports.

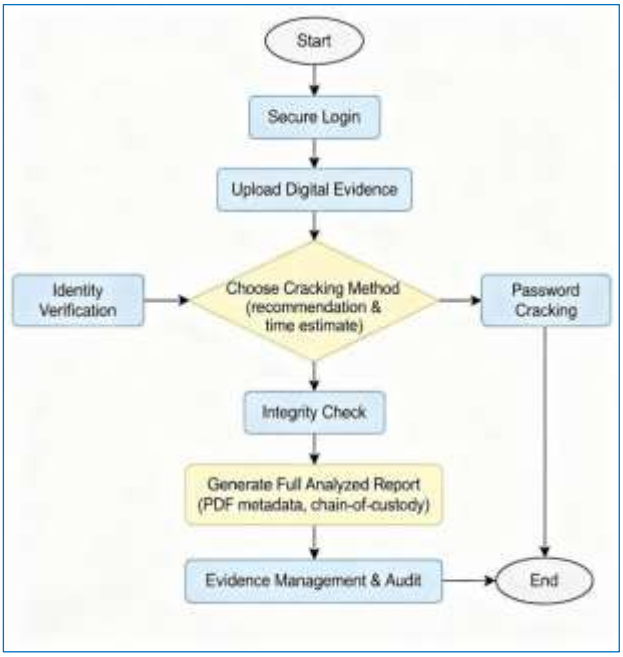

Fig. I. Workflow of Digital Evidence Decryption and Management System

Secure login (Fig. 2):

The login process is designed to be straightforward while enforcing strict access control. Each user authenticates with individual credentials, and the system automatically records the username and timestamp for every sign-in. Evidence access events are logged in the background to preserve traceability without interrupting the workflow. Administrators can enforce password policies and revoke user access when required (Fig. 2).

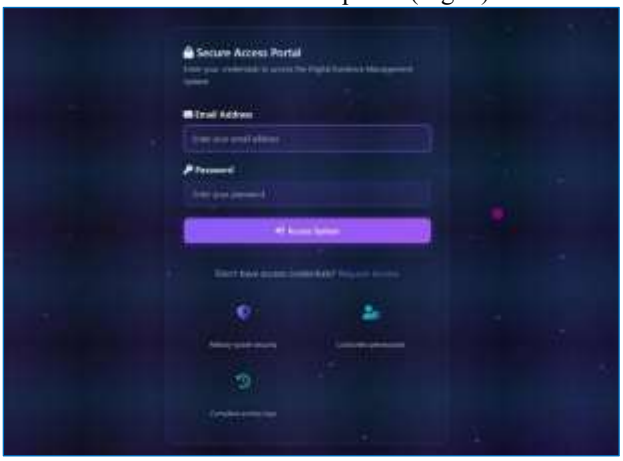

Fig. 2. Login Screen for Secure Access to the Tool

Dashboard overview (Fig. 3):

After authentication, users are directed to the dashboard (Fig. 3), which serves as the central workspace. It provides quick access to evidence upload, active jobs, reports, and user management, along with a summary of the current user and active cases. The layout was kept predictable based on practitioner feedback, prioritizing clear and repeatable steps over unnecessary interface complexity.

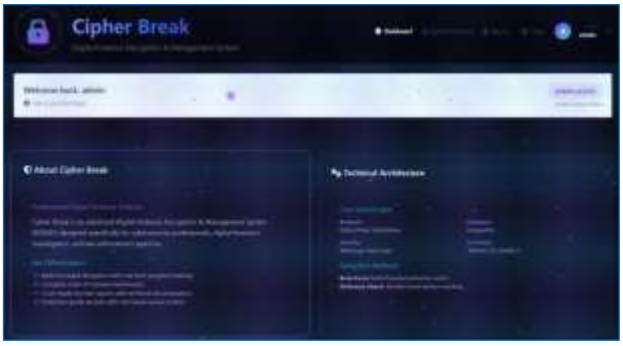

Fig. 3. Dashboard of Cipher Break.

Evidence upload (Fig. 4):

Evidence ingestion is intentionally streamlined: users can drag-and-drop or browse for files, associate the upload with a case identifier, and record brief acquisition notes (e.g., source and seizure context). The system supports common evidence types, including archives and disk images. At upload time, Cipher Break immediately computes and stores a SHA-256 hash for the original file and records the uploader identity and timestamp (Fig. 4). This initial hash serves as the baseline reference for subsequent integrity verification and chain-of-custody documentation.

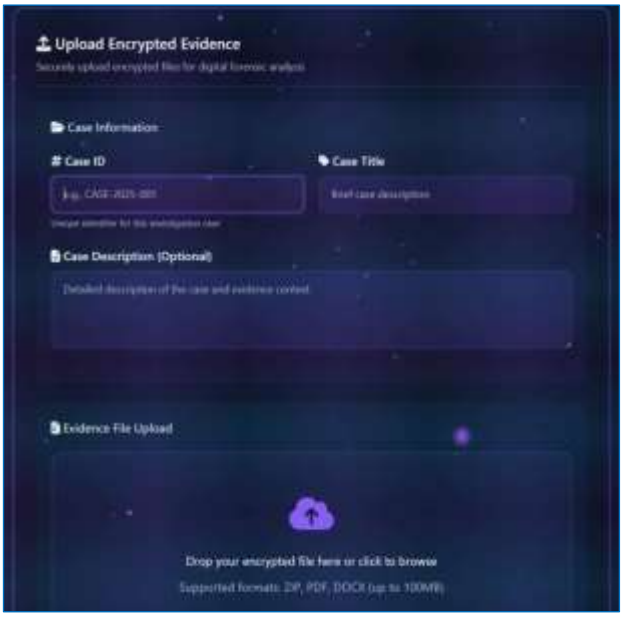

Fig. 4. Upload Interface for Encrypted evidence.

Choosing an attack method (Fig. 5 & Table I):

After ingestion, Cipher Break assists users in selecting an appropriate recovery strategy. The interface presents a brief, plain-language summary of the file context and provides recommended options (e.g., dictionary versus brute force), along with an estimated time and resource profile (Fig. 5).

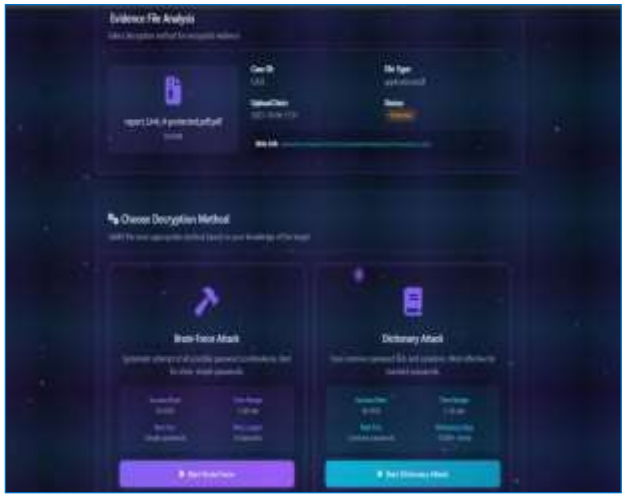

Fig. 5. Interface for Selecting Password Recovery Method.

Typical outcomes observed during testing are summarized below.

Method	Success Rate (%)	Avg Time (min)	CPU Usage
Brute-Force	65-85	15-45	High
Dictionary	80-95	5-20	Medium

TABLE I: METHOD COMPARISON

Model Avg. Decrypt Time (min)	Relative time (%)
Brute Force 25.0	100
Dictionary 10.0	40

TABLE II: PERFORMANCE COMPARISON OF BRUTEFORCE AND DICTIONARY

From the table, it's easy to see that dictionary attacks are faster in most cases, while brute-force ensures nothing is missed when passwords are complex.

Discussion:

Testing revealed a couple of key points about how Cipher Break works in practice:

Dual-method advantage: Combining dictionary and brute-force attacks supports a practical trade-off between speed and coverage. Dictionary attacks often provide faster results for common passwords, while brute force increases completeness when wordlists fail.

Usability: The interface design supports quick navigation and reduces the likelihood of operator error by presenting a clear sequence of actions and minimal configuration overhead.

These numbers are practical estimates that helped investigators pick a strategy quickly: dictionary attacks first for likely passwords, brute force as a fallback. Every recommendation and user decision is recorded so the reasoning is auditable.

Running jobs and tracking progress (Fig. 6):

Recovery jobs execute concurrently across multiple threads. Evidence is partitioned into independent blocks so that progress in one block does not prevent processing of others, improving throughput and reducing contention. The interface displays real-time progress indicators, estimated remaining time, and system load (Fig. 6). For reproducibility, the audit trail records job start and end times, thread identifiers, and the exact command parameters or rule sets applied. When recovery succeeds, the recovered password (stored in an access-controlled area) and decrypted output are linked back to the original evidence entry.

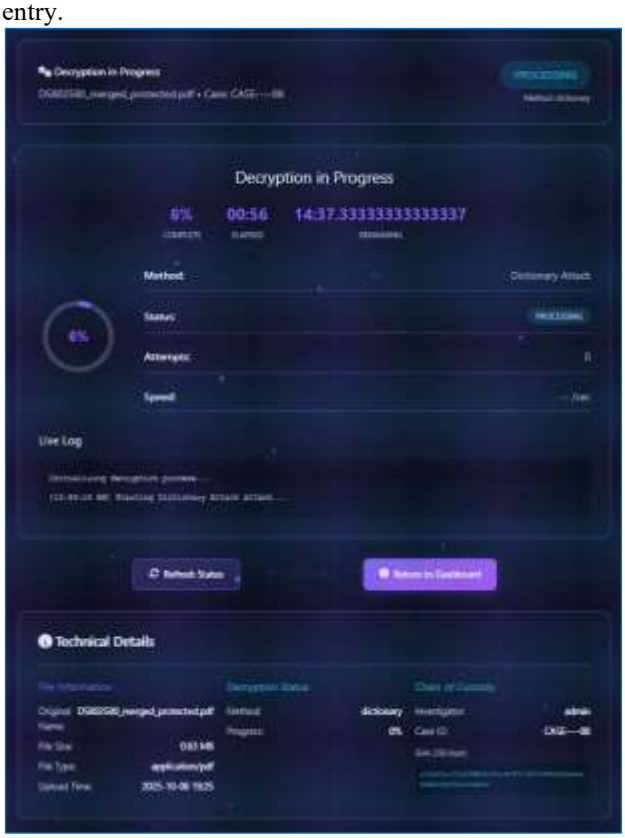

Fig. 6. Page Showing Password Cracking Progress.

Saving evidence metadata and access control:

All system actions, including upload, view, decryption attempts, metadata changes, and report generation, are recorded in an immutable audit log. Stored metadata includes file name, size, file type, uploader identity, case identifier, decryption status, timestamps, and (when recovered) the password in a restricted field. Administrative controls determine who can view recovered passwords or export decrypted outputs, ensuring that users access only the information required for their role. These controls simplify auditing by making it possible to trace who performed each action and when.

Integrity verification (SHA-256):

SHA-256 is computed at the time of upload and re-computed after decryption. The system compares the resulting digests and immediately flags any mismatch.

When a mismatch is detected, the job is paused and an alert is recorded so investigators can review the evidence item. Hash values and verification outcomes are included in the final PDF report to support chain-of-custody and integrity claims.

Report generation (PDF):

Once a case or recovery job completes, Cipher Break compiles a structured report for download.

The PDF report includes timestamps, user actions, hash history, cracking logs, and a concise summary of the applied methodology (e.g., dictionary rules, brute-force character set, and thread count). Reports are downloadable and labeled with a system identifier so their provenance can be verified.

User & Admin Management (Fig. 7):

The administration panel (Fig. 7) provides visibility into active users and their recorded activities. Administrators can create or remove accounts and assign roles (e.g., investigator, auditor, administrator) to enforce least-privilege access. Sensitive fields such as recovered passwords are restricted to authorized roles. When a case is ready for submission, it can be locked to prevent accidental modification and to preserve evidentiary consistency.

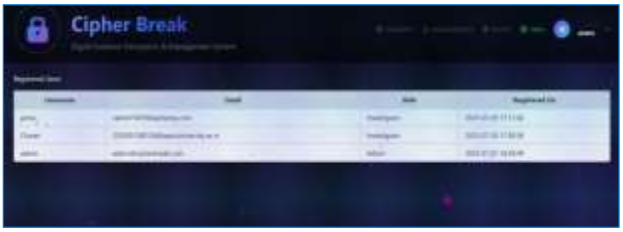

Fig. 7. Interface Showing Registered Users

IV. RESULTS AND DISCUSSION

Cipher Break was evaluated using representative encrypted sample files to assess operational behavior under realistic conditions. During testing, users were able to authenticate, upload encrypted evidence, and select a recovery method manually or accept the system recommendation. After method selection, the platform executed decryption attempts and automatically performed integrity verification throughout processing. Dictionary attacks typically completed in approximately 10 minutes, whereas brute-force attacks required closer to 25 minutes. The recommendation feature selected the more suitable method in about 85% of the tested cases. Generated PDF reports captured key forensic metadata, audit logs, and chain-of-custody details, allowing reviewers to reconstruct what actions occurred and when. Overall, the system reduced manual tracking effort while improving consistency of documentation.

Fig. 8 presents the interface listing processed evidence items, and Fig. 9 shows an example of a generated, downloadable PDF report.

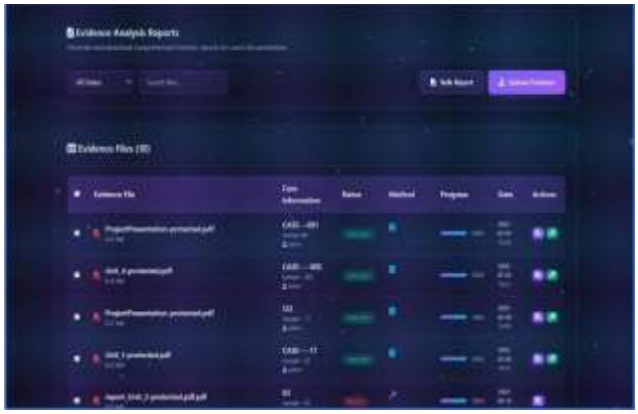

Fig. 8. Interface Showing All processed Evidence File.

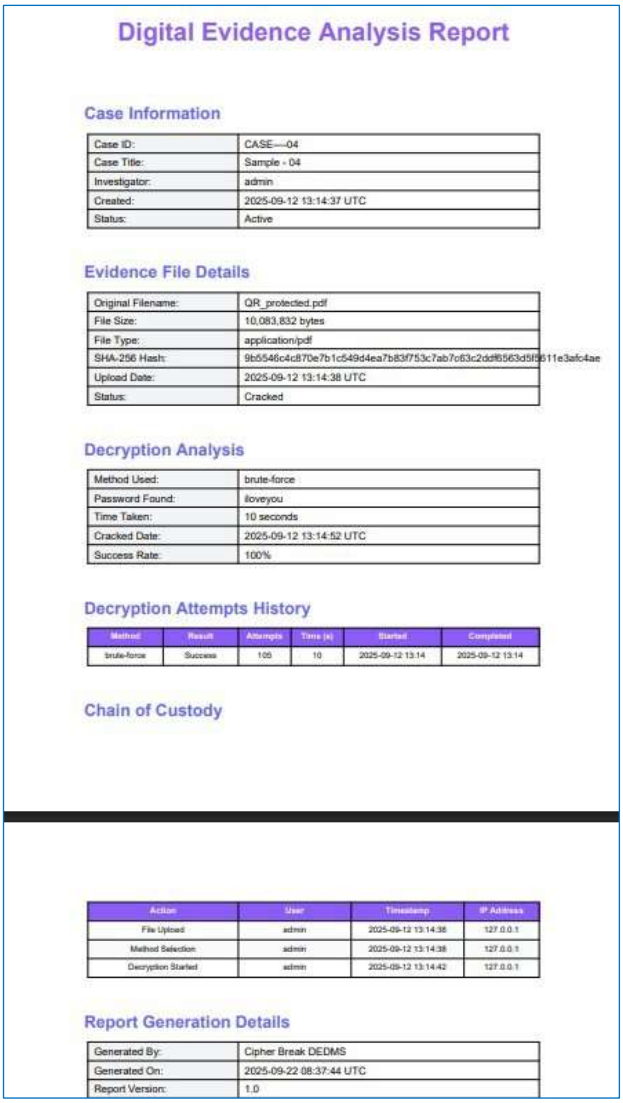

Fig. 9. Page Showing Downloadable PDF report.

Graphical Representation (Fig. I 0):

Fig. 10 provides a visual comparison of observed success rates for the two password recovery approaches. Brute-force recovery achieved approximately 65% to 85% success, depending on password length and complexity.

Dictionary attacks achieved higher success in the evaluated set, typically between 80% and 95%.

This visualization helps investigators quickly identify which strategy is more likely to succeed for a given evidence item and supports informed selection of the most efficient method during casework. Presenting performance trends in a compact form can also reduce trial-and-error and shorten the overall recovery timeline.

Fig. 10. Success rate comparison between brute-force and dictionary attacks.

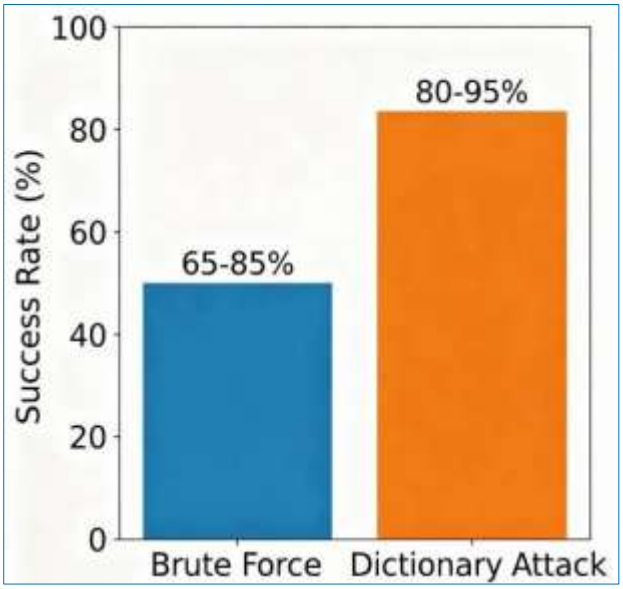

Fig. 10. Success rate comparison

V. CONCLUSION

Cipher Break was developed to reduce the operational gap between password recovery and formal evidence documentation by providing a single system for both decryption and record keeping. Testing across a range of encrypted samples indicates that combining dictionary and brute-force techniques is effective in practice: dictionary attacks often recover common passwords quickly, while brute force provides broader coverage for more complex cases at the cost of additional time. The graphical interface reduces dependence on command-line workflows, which can lower the learning curve for new users. At the same time, the platform maintains detailed audit logs and produces PDF reports containing timestamps and hash values, supporting chain-of-custody requirements when evidence is shared internally or presented for legal review. Future work will focus on extending supported cracking strategies, exploring AI-assisted wordlist suggestions, and enabling optional scaling of storage or compute resources using cloud infrastructure.

REFERENCES

[1] M. S. Alkhateeb, "A Comparative Study on Brute Force and Dictionary Attack for Password Cracking," International Journal of Computer Applications, 2021.
[2] J. Steube, "Hashcat -Advanced Password Recovery," hashcat.net,
[3] 2024.
[4] M. D. Corner and R. Singh, "Improving Efficiency of Password Cracking Tools," IEEE Access, vol. 8, pp. 11234-11241, 2022.
[5] S. K. Saini, "Digital Evidence Management and Chain of Custody,"
[6] Journal of Digital Forensics and Security, 2020.
[7] A. Kulkarni, Digital Evidence Decryption and Management System,
[8] Student Research Project Report, 2025.
[9] M. Conti, V. L. Vigna, and C. C. Zou, "Password Cracking: Techniques, Tools, and Challenges," Journal ofCyber Security, vol. 7,no.2,pp.45-58,2021.
[10] C. Casey, Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet, 4th ed., Academic Press, 2019.
[11] S. Garfinkel, "Digital Forensics Research: The Next 10 Years,"
[12] Digital Investigation, vol. 16, pp. 21-27, 2019.
[13] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed., Wiley, 1996.
[14] J. Steube, "Hashcat - Advanced Password Recovery Tool,"
[15] hashcat.net, 2024.
[16] A. Narayanan and V. Shmatikov, "Fast Dictionary Attacks on Passwords Using GPUs," in Proc. I 3th USENIX Security Symposium, San Diego, 2004, pp. 175-189.
[17] L. P. Patil, A. Bhalavi, R. Dubey, and M. Patidar, "Efficient algorithm for Speech Enhancement using Adaptive filter," International Journal of Electrical, Electronics and Computer Engineering, vol. 3, 2022.
[18] M. Patidar, N. Jain, and A. Parikh, "Classification of normal and myopathy EMG signals using BP neural network," International Journal of Computer Applications, vol. 69, no. 8, 2014.
[19] S. Nagar, M. Patidar, R. Sheikh, S. Singh, A. Kashiv, A. Jain, S. K. Namdeo, et al., "Review and Explore the Transformative Impact of Artificial Intelligence (AI) in Smart Healthcare Systems," in Proceedings of the 2024 International Conference on Advances in Computing Research on Science, 2024.
[20] M. Patidar, P. Bhanodia, K. K. Sethi, S. Rajput, P. S. Patil, C. Tiwari,
[21] A. Gid, et al., \"A Deep Learning Approach to Improving Patient Safety in Healthcare Using Real-Time Face Mask Detection,\" in Proceedings of the 2024 International Conference on Advances in Computing Research on Science, 2024.
[22] S. S. Ahmadpour, D. B. Avval, N. J. Navimipour, H. Rasmi, A. Heidari, S. Kassa, et al., "A New Median Filter Circuit Design Based on Atomic Silicon Quantum-Dot for Digital Image Processing and IoT Applications," IEEE Internet of Things Journal, 2024.
[23] M. Patidar, A. Jain, K. Patidar, S. K. Shukla, A.H. Majeed, N. Gupta, and N. Patidar, \"An ultra-dense and cost-efficient

coplanar RAM cell design in quantum-dot cellular automata technology,\" The Journal of Supercomputing, vol. 80, no. 5, pp. 6989-7027, 2025.

[24] R. Yadav, P. Moghe, M. Patidar, V. Jain, M. Tembhurney, and P. K. Patidar, \"Performance Analysis of Side Lobe Reduction for Smart Antenna Systems Using Genetic Algorithms (GA),\" IEEE Xplore, 2024