Brute-Force Password Attack for Automated PDF Decryption with a Graphical Interface

Dr.K.Venkatesh

Assistant Professor

Department of Computer Science and Engineering

Kalasalingam Academy Of Research and Education, Krishnankoil-626126,

[venkiur@gmail.com](mailto:venkiur@gmail.com)

Ch.Gopi Krishna

Department of Computer Science and engineering

Kalasalingam Academy Of Research and Education, Krishnankoil-626126, Virudhnagar, India

[99220040367@klu.ac.in](mailto:99220040367@klu.ac.in)

Kalluru Bhavesh Shankar

Department of Computer Science and Engineering

Kalasalingam Academy Of Research and Education, Krishnankoil-626126, Virudhnagar, India

[99220040083@klu.ac.in](mailto:99220040083@klu.ac.in)

T.Jani Reddy

Department of Mechanical Engineering

Kalasalingam Academy Of Research and Education, Krishnankoil-626126, Virudhnagar, India

[99220040356@klu.ac.in](mailto:99220040356@klu.ac.in)

S.Imran

Department of Computer Science and engineering

Kalasalingam Academy Of Research and Education, Krishnankoil-626126, Virudhnagar, India

[99220040204@klu.ac.in](mailto:99220040204@klu.ac.in)

Abstract: Sensitive information is frequently shared securely using PDFs (Portable Document Format), which frequently use encryption to prevent unwanted access. But in cases where an encrypted PDF's password is misplaced or forgotten, it becomes crucial to get access to the file again without jeopardizing its integrity. In this project, we offer a solution to this issue by creating an automated tool for decrypting PDFs that recovers the password for the document through a brute-force password attack. Because of the tool's intuitive graphical user interface (GUI), which was created with Python's Tkinter library, even users with little technical experience can use it. With the use of a password list, the system is made to make the process of decrypting encrypted PDF files easier. A brute-force attack is a technique that involves methodically trying every password until the right one is discovered. Users can load an encrypted PDF, supply a text file with a list of possible passwords, and designate an output file where the decrypted PDF is to be saved. This process is automated within the application. The program iterates through the password list, trying each one until either the right password is found or all other options have been tried. More sophisticated decryption methods, like dictionary attacks or hybrid strategies that incorporate aspects of rule-based and brute-force attacks, may be added to this project in the future. Adding multi-threading capabilities to expedite the password recovery process is another possible improvement. To increase its applicability, the tool could also be expanded to support more encryption formats and PDF versions. In conclusion, the brute-force password recovery method for decrypting encrypted PDF files can be easily and practically accomplished with the help of the PDF Decryptor tool. Its graphical user interface and automated password recovery process make it a user-friendly tool for both novice and expert users. This project highlights the value of password security while supplying a useful tool for practical situations by providing a means to unlock locked PDF files.

Keywords: PDF Decryption, Brute-Force Attack, Python Tkinter, PyPPF2 Library, Password Recovery.

# **I.Introduction**

One of the most widely used formats for safely sharing documents online is the Portable Document Format (PDF). Because of their portability and capacity to preserve formatting across multiple devices, Adobe PDFs, which are widely used for a variety of purposes, including official documents, academic papers, contracts, and reports, are developed by the company. PDFs frequently have encryption built in to further strengthen security—only authorized users with the right password can view the content. For sensitive data, like financial records, medical reports, or private agreements, encryption is especially crucial. The same security precaution, though, may not work if the password is misplaced or forgotten. Regaining access to an encrypted PDF in such circumstances can be difficult. In numerous real-world situations, it becomes evident that a dependable method for decrypting password-protected PDFs is required. Professionals who work with encrypted documents, for example, may forget or misplace their passwords from time to time, making important information unavailable. People who receive password-protected PDFs from collaborators or inherit digital documents might not be able to access the content without the password. Although there aren't many options for recovering passwords, a brute-force password attack is a useful technique. This method involves methodically testing different password combinations until the right one is discovered. Brute-force attacks are efficient even though they can take a while, especially if the password is short or relatively simple.

The goal of this project is to create a Python-based PDF decryption tool that can retrieve passwords from encrypted PDF files using brute-force password attacks. Using PyPDF2 to handle PDF files and Python's Tkinter library for the GUI, the application offers a practical and easy-to-use solution for those who have forgotten or lost their passwords. With its simple design, users can decrypt PDFs by providing a list of potential passwords for the brute-force attack, which is accomplished by the application.

The PyPDF2 library, a well-liked Python package for handling PDF files, is the foundation of this application. The tool can read encrypted PDFs and try to decrypt them with the password provided thanks to PyPDF2. Finding the right password takes time, which is one of the main problems with brute-force attacks. The duration required for the tool to test every possible combination increases with the length and complexity of the password.

This project also tackles possible problems with efficiently managing the PDF decryption process and handling large password lists. The program is made to gracefully handle errors, giving users informative error messages in the event that file problems or bad passwords prevent the decryption from succeeding. By doing this, the application guarantees a seamless user experience even when faced with difficulties.  
To sum up, the PDF Decryptor tool provides a workable way for people and businesses who need to access encrypted PDFs again but can't remember their passwords. This utility fills the gap between accessibility and functionality by combining a user-friendly interface with the strength of brute-force password recovery, giving users a vital tool for recovering encrypted documents.

**II. Literature Survey**

The modern digital environment is always at risk from cyberattacks. Because cybersecurity attacks are becoming more complex, it is more important than ever to take effective security measures. Among the greatest methods to secure more data is to use encryption techniques and algorithms.[1]

delicate information via cyberattacks. This study aims to provide an analysis of the function of cybersecurity defences in order to effectively defend against cybersecurity attacks.

One of the most extensively used document formats in the world is the Portable Document Format, or PDF for short. It supports document encryption to guarantee the privacy of user information. In this work, we examine PDF encryption and demonstrate. Two cutting-edge methods for breaching the privacy of encrypted [2].

One of the best ways to ensure data security and privacy is through encryption. The original content of a data is hidden by encryption techniques so that it can only be retrieved with a key that is used in a process known as decryption. The purpose of encryption is to safeguard or shield data from unwanted access, meaning that it cannot be viewed or altered.

[3].

By concentrating on the encryption of non-datafiles, Vaheedbasha Shaik and Dr. Natarajan K.'s paper, "Flexible and Cost-effective Cryptographic Encryption Algorithm for Securing Unencrypted Database Files at Rest and in Transit," closes a significant vulnerability in database security. While datafile protection is handled by traditional encryption techniques like Transparent Data Encryption (TDE), non-datafiles—which, if left unencrypted, can also present serious security risks—are frequently ignored [4].

The main topic of the paper "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data" is the application of Attribute-Based Encryption (ABE) as a reliable method for overseeing fine-grained access control over encrypted data [5]. This method works particularly well in situations such as cloud storage, where it is necessary to encrypt sensitive data but varying access rights may be granted to users according to predefined attributes.

**III. Methodology**

This methodology describes the steps taken to create an automated tool for decrypting PDF files using a brute-force attack. The main goal is to develop an intuitive user interface for choosing password lists, encrypted PDF files, and output file paths. The approach combines a number of programming languages and tools, with an emphasis on using Tkinter and Python for the graphical user interface and PyPDF2 for PDF operations.

The system is organized around a Tkinter-built graphical user interface (GUI), which enables smooth user interaction with the program. There are three main primary parts to the architecture:  
User Interface (UI): Developed with Tkinter, the UI lets users search for files, enter information, and start the decryption   
The PyPDF2 library, which is in charge of reading, decrypting, and saving PDF files, is used for core functionality.  
Brute-Force Method: Until the right password is discovered, the brute-force algorithm goes through a list of possible passwords and tests each one against the encrypted PDF file.

Python is used to develop the application because of its large library and active community. The subsequent elements are necessary for the development. The implementation process involves several key steps. Numerous buttons, entry fields, and labels make up the user interface. The arrangement is made to intuitively lead the user through the decryption process.

The application's main feature is implemented using a brute-force technique that uses every password in the list in an attempt to decrypt the PDF. The application saves the decrypted PDF to the designated output location after determining the correct password.

Extensive testing is done to make sure the application functions as intended:

Functional Testing: To ensure the application fulfils its intended purpose, each component is tested separately.

Integration Testing: This type of testing involves testing the application as a whole to make sure all the parts work together properly and that the process of choosing files, decrypting them, and saving them is smooth.

Performance Testing: PDFs with different levels of complexity and password lengths are used to test the effectiveness of the brute-force method.

User documentation is provided to assist users in navigating the application.

The above-mentioned methodology describes the methodical approach used to create an automated tool for decrypting PDFs using a brute-force password attack. This application seeks to offer a dependable solution for users requiring access to encrypted PDF files through thoughtful design and implementation, guaranteeing user-friendliness and efficient operation.

This approach can provide a solid basis for the creation and implementation of the PDF decryption tool, guaranteeing transparency throughout the procedure and directing upcoming improvements.

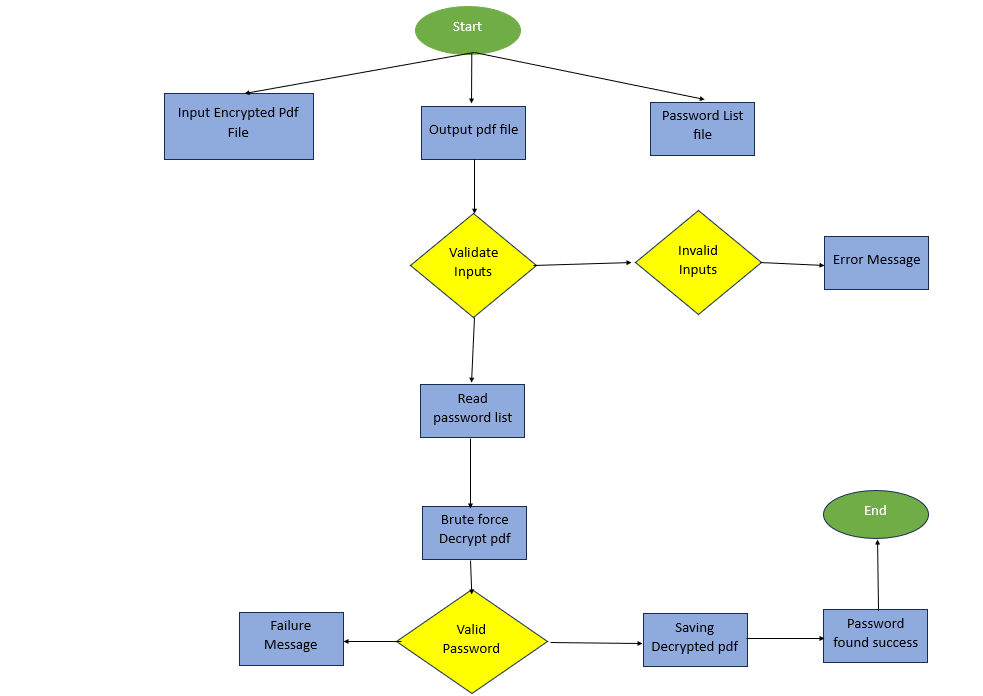
**IV.Block Diagram**

fig 4.1 block diagram

The above figure explains about the working flow of the project which user have to give three paths which are Encrypted file, random passwords file and name of the output pdf file and it will perform brute force attack and if password found mean it will generate same encrypted pdf with out password.

**V.Implementation**

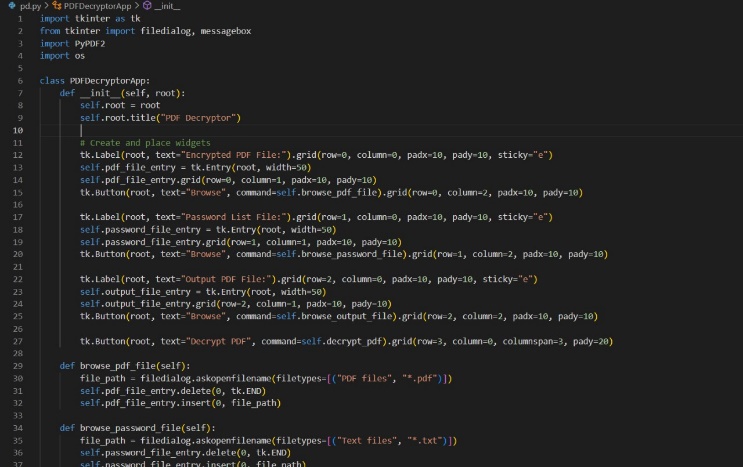


fig 5.1 implemented code

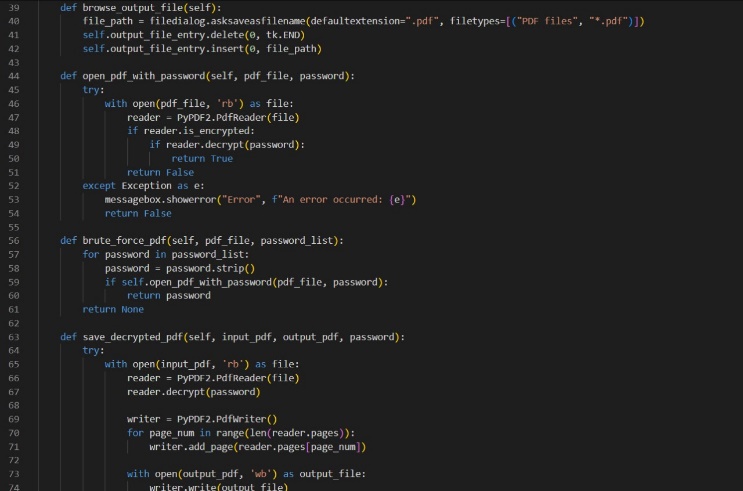


fig 5.2 implemented code

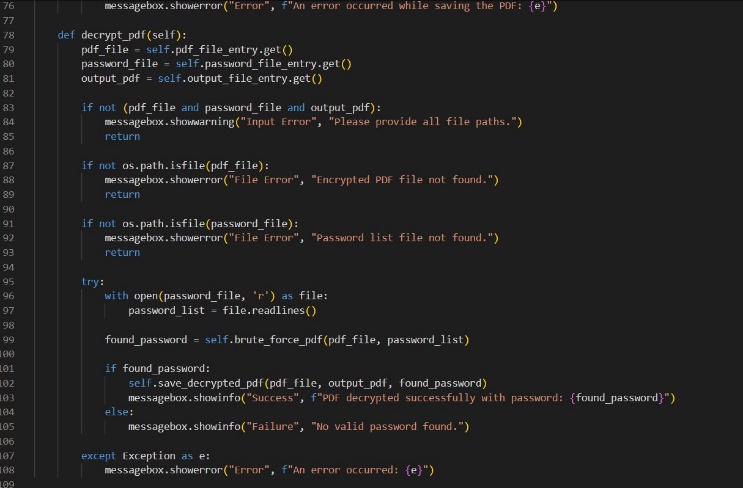


fig 5.4 implemented code

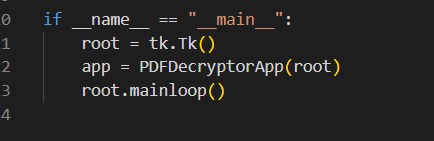
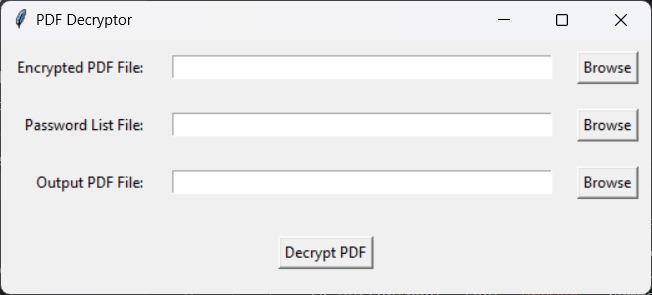
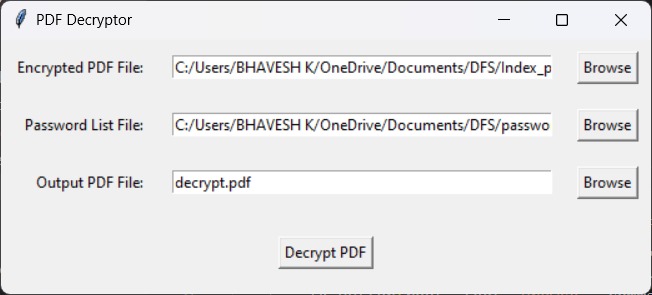


fig 5.5 implemented code

The above figures are implementation of code. which are used to decrypt the pdf by the brute force attack with the graphical user interface.

**VI. Output**





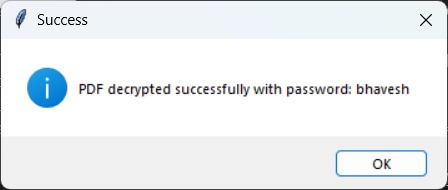


fig 6.1 output

The above images are about the output of the project pdf decryptor. In this user interface we have to give the path for encrypted pdf and name of the output pdf file and some set of random passwords. It will perform brute force and if there any password matches mean it will give new pdf file without password.

**VII.Result**

The project is about the graphical application written in Python that uses a brute-force password strategy to decrypt encrypted PDF files. To do this, a password list file containing possible passwords is used, and the encrypted PDF is compared to the list to determine the correct password. The decoded PDF is located and then saved to an output file that the user designates. The program handles PDF file operations with PyPDF2 and uses the tkinter library for its graphical user interface (GUI). This tool's objective is to provide a quick and easy method for password-protected PDF files to be unlocked. Brute-forcing the password for the encrypted PDF is the application's primary purpose.The brute force pdf() method iterates over the password list, comparing each password to the encrypted PDF file. The function reads every password from the password file, opens it, and checks each one individually. The function open\_pdf\_with\_password () manages to access the password-protected PDF and makes an effort to decrypt it using the current password. The brute force method is halted and the correct password is returned if the password decrypts the PDF.The function returns None and notifies the user that the decryption attempt was unsuccessful if the password is ineffective. PDF Decryption: The PyPDF2 library, notably the PyPDF2, is used to do the actual decryption. To read the encrypted PDF, use the PdfReader() object. Using the supplied password, PdfReader's decrypt () function tries to open the file. The file can then be read and edited if the right password is discovered. This makes it possible for the application to use the PdfWriter() class from the same library to write the decrypted content to a new PDF file.

Once the password is found, the decrypted PDF content needs to be saved to the output file. The save\_decrypted\_pdf () function handles this by: Opening the original encrypted PDF.Decrypting it with the correct password.

Writing the decrypted pages into a new file using the PdfWriter() class. The program adds each page from the reader to the writer and finally saves the file in the location specified by the user.

This program provides a simple, user-friendly way to unlock PDF files that are locked. The usage of brute force means that the right password must be present in the provided password list. The tool provides a useful method of decrypting password-protected PDFs when a list of potential passwords is supplied, despite its dependence. Even users without technical experience can use the process thanks to the graphical interface.

**VIII. Conclusion**

The PDF Decryptor graphical interface is an excellent solution developed to solve the difficulty of accessing password-protected PDF files through a brute-force technique. Through the use of a graphical user interface created with the tkinter library, it streamlines a difficult process and makes it simple for non-programmers to engage with the application by choosing files and controlling the decryption process.The integration of PyPDF2, which permits the manipulation of PDF files, is crucial to the program. Three primary inputs are used by the application to help the user: an output path for the decrypted file, a text file with a list of possible passwords, and the encrypted PDF file. When the decryption process starts, the application goes through each password in the list and uses a loop-based brute-force method to try to unlock the PDF. When the right password is discovered, the file is decrypted, saved, and the user is notified of the success and the password that was used. Additionally, the program has a robust error-handling mechanism in place to make sure users are informed when something goes wrong, such missing files or erroneous locations.

It's also critical to emphasize that the application solely depends on the user supplying a text file with potential passwords; it does not employ sophisticated cryptographic techniques to crack encryption. This restricts the use of it to situations where there is a chance that the password can be guessed or recovered, like lost passwords or gaining access to files where a password may be known.In short, the PDF Decryptor application offers an easy and user-friendly solution to the problem of accessing encrypted PDF files. With its user-friendly UI and simple brute-force methods, the application opens up PDF decryption to a large audience. But the strength of the PDF's encryption and the caliber of the password list will determine how successful it is. Despite this drawback, it functions as a handy tool for persons requiring to unlock PDFs when a password is suspected but forgotten.

**IX. References**

1. Encryption Techniques and Algorithms to Combat Cybersecurity Attacks: A Review .

<https://www.researchgate.net/publication/372832281_Encryption_Techniques_and_Algorithms_to_Combat_Cybersecurity_Attacks_A_Review?enrichId=rgreq-ad839262ea677a8b1a521bd2f51738c6-XXX&enrichSource=Y292ZXJQYWdlOzM3MjgzMjI4MTtBUzoxMTQzMTI4MTIzMTM2ODM5OEAxNzExNDA5NDM4ODMx&el=1_x_3&_esc=publicationCoverPdf>

1. Practical Decryption exFiltration: Breaking PDF Encryption <https://www.nds.rub.de/media/nds/veroeffentlichungen/2021/05/05/PDF_Encryption.pdf>
2. A Secure and Fast Approach for Encryption and Decryption of Message Communication. <https://www.researchgate.net/publication/320149845_A_Secure_and_Fast_Approach_for_Encryption_and_Decryption_of_Message_Communication?enrichId=rgreq-d821ce2707df9968d5575f05c6151fc7-XXX&enrichSource=Y292ZXJQYWdlOzMyMDE0OTg0NTtBUzo1NDQ2ODIzOTgxMDU2MDBAMTUwNjg3MzgxMTYxNw%3D%3D&el=1_x_3&_esc=publicationCoverPdf>
3. Flexible and cost-effective cryptographic encryption algorithm for securing unencrypted database files at rest and in transit.
4. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data.
5. A study and performance analysis of rsa algorithm.
6. Password Security in Document Management Systems. (2020).
7. Graphical User Interfaces for Security Applications Ahmed, A., & Tan, C..(2017) .
8. Cybersecurity Threats in Portable Document Format (PDF) Liu, Y., & Wong, C.(2022).

1. Document Security: The Role of Encryption and Password Management Silva, R., & Torres, M. (2018).
2. Python for Cybersecurity: Applications in Document Decryption,Rohers.D and Nguyen.T (2021) .
3. Advanced Cryptographic Techniques for PDF File Security.(2020).
4. Password Recovery Techniques in Document Security Singh.S, Kumar .P (2019).
5. Password Recovery Techniques in Document Security. Wilson R and Green P (2019).
6. Efficient Pdf Encryption and Decryption Techniques (2019).