IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁑ Impact Factor 8.102 ⁑ Vol. 12, Issue 3, March 2023

DOI: 10.17148/IJARCCE.2023.12339

“Implementing Encryption and Decryption of Documents in Java for Enhanced Security”

Unnati Pandya[1], Dr. A. Rengarajan[2]

Masters Student, School of CS&IT, Jain (Deemed-To-Be University), Bengaluru[1]

Associate Professor, School of CS&IT, Jain (Deemed-To-Be University), Bengaluru[2]

Abstract: Crypto system is responsible for encrypting the user’s data and provide a secure mechanism to store it in a virtual drive. This virtual drive will be created by the system for the particular user for the very first while using the system. This system will provide limited storage area where the data can be saved. As we can say that, it’s a cloud storage medium where data can be accessed from any location. This system will also enable you to synchronize your desktop or laptop while using this crypto system.

Keywords: Encryption, Decryption, Documents, Java, Cipher, Text.

I. INTRODUCTION

The internet is becoming a primary source for information. The amount of the information is so overwhelming that users can spend hours and hours browsing the internet. More and more companies do business by sending data in the form of email over the internet. As information becomes an increasingly commodity, and as the communications revolution changes society, so the process of encoding messages known as encryption will play an increasing role to in everyday life. Nowadays our e-mails pass through various computers and this form of communication can be intercepted with ease, so jeopardizing our privacy.

More and more companies do business by sending data in the form of email over the internet. Since we are moving toward a future where the nation will be crisscrossed with high capacity fiber optic data networks linking together all our increasingly ubiquitous personal computers, encryption is the only way to protect our privacy and guarantee the success of the digital marketplace. The art of secret communication, otherwise known as cryptography, will provide the locks and keys of the Information Age. Without paper precautions, personal computer and private networks are exposed to security threats such as eavesdropping, modification, and impersonation making their private and sensitive information totally vulnerable to computer hacker. Therefore, internet security becomes very important to every user. One of the internet security method is data encryption. Encryption is said to be the most effective security method n the internet. It works like a clock that requires a specific key to open it. encryption keep data secure during transmission over internet while it is being t red in a computer. A good encryption system will protect and secure to be good and be able to sustain attack from computer hackers. It has to have to a strong encryption algorithm. Therefor with data security being the main concern at hand, this project will undertake to a good and strong encryption system.

II. LITRETURE REVIEW

Li, J., Huang, (2019) Searchable symmetric encryption (SSE) has been widely applied in the encrypted database for queries in practice. Although SSE is powerful and feature-rich, it is always plagued by information leaks. Some recent attacks point out that forward privacy which disallows leakage from update operations, now becomes a basic requirement for any newly designed SSE schemes. However, the subsequent search operations can still leak a significant amount of information. [1]

Papamanthou, C. B., (2014) Dynamic Searchable Symmetric Encryption (DSSE) enables a client to encrypt his document collection in a way that it is still searchable and efficiently updatable. However, all DSSE constructions that have been presented in the literature so far come with several problems: Either they leak a significant amount of information (eg, hashes of the keywords contained in the updated document) or are inefficient in terms of space or search/update time (eg, linear in the number of documents). In this paper we revisit the DSSE problem. [2]

Cash, D., (2015) Schemes for secure outsourcing of client data with search capability are being increasingly marketed and deployed. In the literature, schemes for accomplishing this efficiently are called Searchable Encryption (SE). They achieve high efficiency with provable security by means of a quantifiable leakage profile. However, the degree to which SE leakage can be exploited by an adversary is not well understood. [3]

DOI: 10.17148/IJARCCE.2023.12339

Cash, D., & Steiner, M. (2014) We design and implement dynamic symmetric searchable encryption schemes that efficiently and privately search server-held encrypted databases with tens of billions of record-keyword pairs. Our basic theoretical construction supports single-keyword searches and offers asymptotically optimal server index size, fully parallel searching, and minimal leakage. Our implementation effort brought to the fore several factors ignored by earlier coarse-grained theoretical performance analyses, including low-level space utilization, I/O [4]

Asharov, G., Segev, G., & Shahaf, I. (2021) A searchable symmetric encryption (SSE) scheme enables a client to store data on an untrusted server while supporting keyword searches in a secure manner. Recent experiments have indicated that the practical relevance of such schemes heavily relies on the tradeoff between their space overhead, locality (the number of non-contiguous memory locations that the server accesses with each query), and read efficiency [5]

Bellare, M.,(2007) We present as-strong-as-possible definitions of privacy, and constructions achieving them, for public-key encryption schemes where the encryption algorithm is deterministic. We obtain as a consequence database encryption method that permit fast (ie sub-linear, and in fact logarithmic, time) search while provably providing privacy that is as strong as possible subject to this fast search constraint. One of our constructs, called RSA-DOAEP, has the added feature of being length preserving [6]

Van Liesdonk, P., (2010) Searchable encryption is a technique that allows a client to store documents on a server in encrypted form. Stored documents can be retrieved selectively while revealing as little information as possible to the server. In the symmetric searchable encryption domain, the storage and the retrieval are performed by the same client. Most conventional searchable encryption schemes suffer from two disadvantages. First, searching the stored documents takes time linear in the size of the database, and/or uses heavy arithmetic operations [7]

Abdalla M,(2005) We identify and fill some gaps with regard to consistency (the extent to which false positives are produced) for public-key encryption with keyword search (PEKS). We define computational and statistical relaxations of the existing notion of perfect consistency, show that the scheme of [7] is computationally consistent, and provide a new scheme that is statistically consistent. We also provide a transform of an anonymous IBE scheme to a secure PEKS scheme that, unlike the previous one, guarantees consistency. [8]

Baek, J.,(2008) The public key encryption with keyword search (PEKS) scheme, proposed by Boneh, Di Crescenzo, Ostrovsky and Persiano, enables one to search for encrypted keywords without compromising the security of the original data. In this paper, we address two important issues of a PEKS scheme,“removing secure channel” and “refreshing keywords”, which have not beenconsidered in Boneh et al.'s paper. We point out the inefficiency of the original PEKS scheme due to the use of the secure channel. [9]

Gao, H.,(2021) Due to capacity limitations, large amounts of data generated by IoT devices are often stored on cloud servers. These data are usually encrypted to prevent the disclosure, which significantly affects the availability of this data. Searchable encryption (SE) allows a party to store his data created by his IoT devices or mobile in encryption on the cloud server to protect his privacy while retaining his ability to search for data. However, the general SE techniques are all pay-then-use. [10]

III. PROBLEM STATEMENT

This system will provide limited storage area where the data can be saved. As we can say that, it’s a cloud storage medium where data can be accessed from any location. This system will also enable you to synchronize your desktop or laptop while using this crypto system.For implementing security mechanism, each user’s should have a valid login id and password and the verification of user’s account will be done through their valid email id. Upon accessing this crypto system, users can simply drag and drop their documents from their system to their virtual hard drive where it will take some time for encrypting the documents and process of encryption and decryption will depend upon the size of document and their type. User friendly interface has been provided, so that users can easily access all the provided features. It uses strong bit encryption mechanism and its virtual hard drive and easily loaded and unloaded at any location as per the user’s choice.

Requirements: Project will have the capability to securely encrypt and decrypt sensitive or confidential information. The solution must also be user-friendly and easy to use.

Security Measures: My system has robust security measures in place to prevent unauthorized access, hacking, or theft of sensitive information. This includes encryption algorithms, password protection, and user authentication.

DOI: 10.17148/IJARCCE.2023.12339

User Authentication: The solution must provide user authentication, such as username and password, to ensure that only authorized users have access to the encrypted data.

Key Management: The solution must have a key management system in place to manage the encryption keys and ensure that they are securely stored and protected.

Access Control: The solution must provide access control features, such as user permissions, to ensure that users can only access the information that they are authorized to access.

User Interaction: The solution must provide a user-friendly interface that makes it easy for users to encrypt and decrypt data.

Technical Support: The solution must provide technical support and training to users to ensure that they can effectively use the solution and understand how to use its security features.

Security Requirements: project must ensure that the data is protected from unauthorized access. This includes the protection of data at rest, in transit, and during processing. The encryption method used must be secure and unbreakable.

IV. PROPOSED METHODOLOGIES

Developing a document encryption/decryption system in Java is a complex process that involves several steps and requires in-depth knowledge of encryption algorithms and Java programming. To develop such a system, it is important to understand the problem you want to solve, conduct a thorough literature review, choose the right encryption algorithm, implement the encryption and decryption processes, develop a user interface, test the system, and write a research paper. Understanding the problem is the first step in developing a document encryption/decryption system. You need to identify why you need such a system, what type of information you want to protect, and who will use the system. This will help you define the requirements and choose the right encryption algorithm.

Conducting a thorough literature review is the second step in developing a document encryption/decryption system. You need to research existing encryption/decryption algorithms and techniques used in the industry. This will help you identify best practices, understand common issues, and avoid reinventing the wheel.

Choosing the right encryption algorithm is the third step in developing a document encryption/decryption system. The choice of algorithm will depend on the level of security you require, the type of data you want to encrypt, and the performance requirements of the system. Commonly used algorithms include AES, RSA, and DES.

Implementing the encryption and decryption processes is the fourth and fifth steps in developing a document encryption/decryption system. The encryption process involves converting the plaintext document into ciphertext using a key or password, while the decryption process involves converting the ciphertext back into the original plaintext document using the same key or password. It is important to ensure that the encryption and decryption processes are secure and cannot be easily broken.

Developing a user interface is the sixth step in developing a document encryption/decryption system. The user interface should allow users to upload a plaintext or ciphertext document, enter a key or password, and perform the encryption or decryption process. The user interface should also provide feedback to the user to indicate if the process was successful or not.Testing the system is the seventh step in developing a document encryption/decryption system. The system should be thoroughly tested to ensure that it is secure, performs as expected, and meets the requirements. Unit testing and integration testing should be performed to identify and fix any issues

V. IMPLEMENTATION

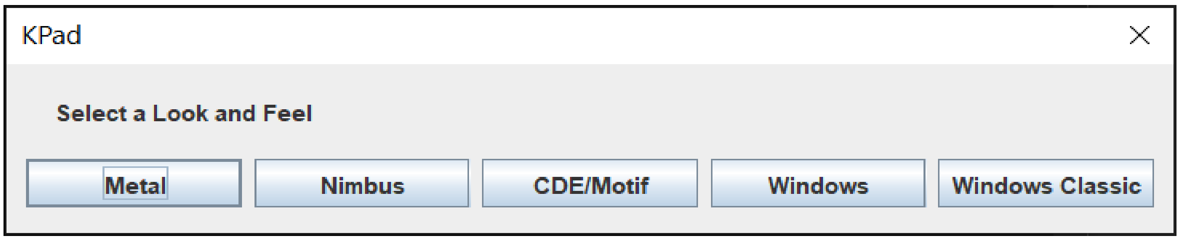


figure 1: Select look for different OS

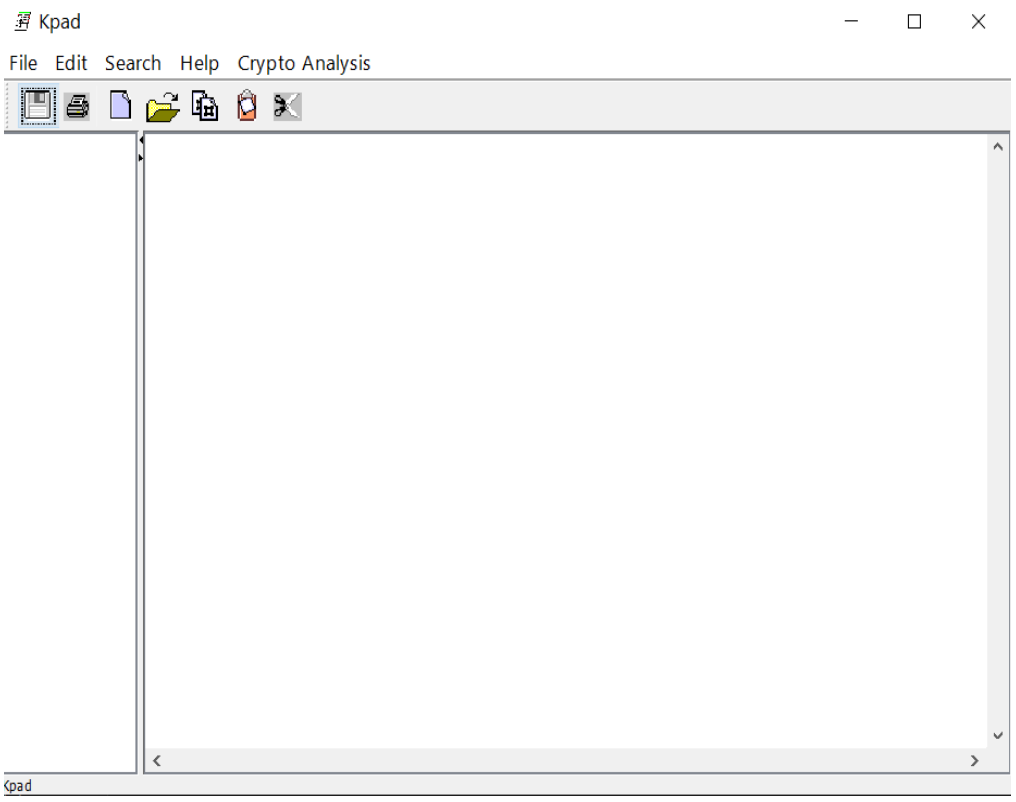
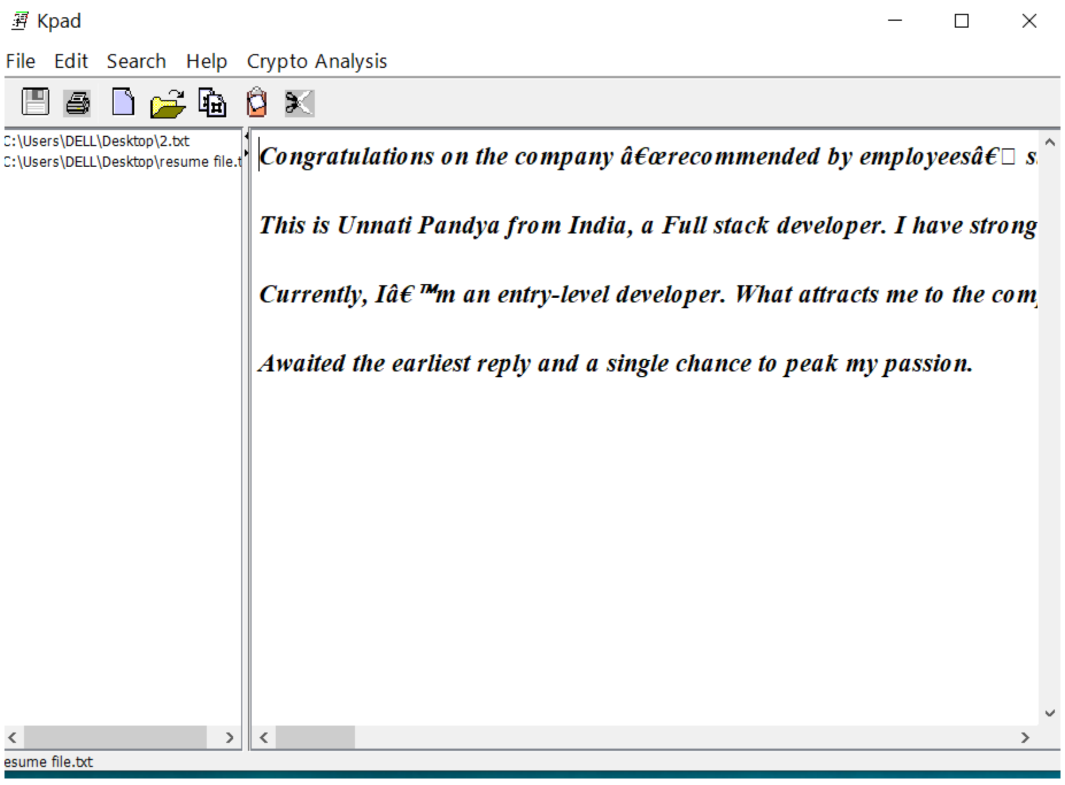


Figure 2: Home Screen


Kpad
File Edit Search Help Crypto Analysis
C:\Users\DELL\Desktop\2.txt
C:\Users\DELL\Desktop\resume file.t
esume file.txt

Congratulations on the company â€œrecommended by employeesâ€□ s

This is Unnati Pandya from India, a Full stack developer. I have strong

Currently, Iâ€™m an entry-level developer. What attracts me to the com

Awaited the earliest reply and a single chance to peak my passion.

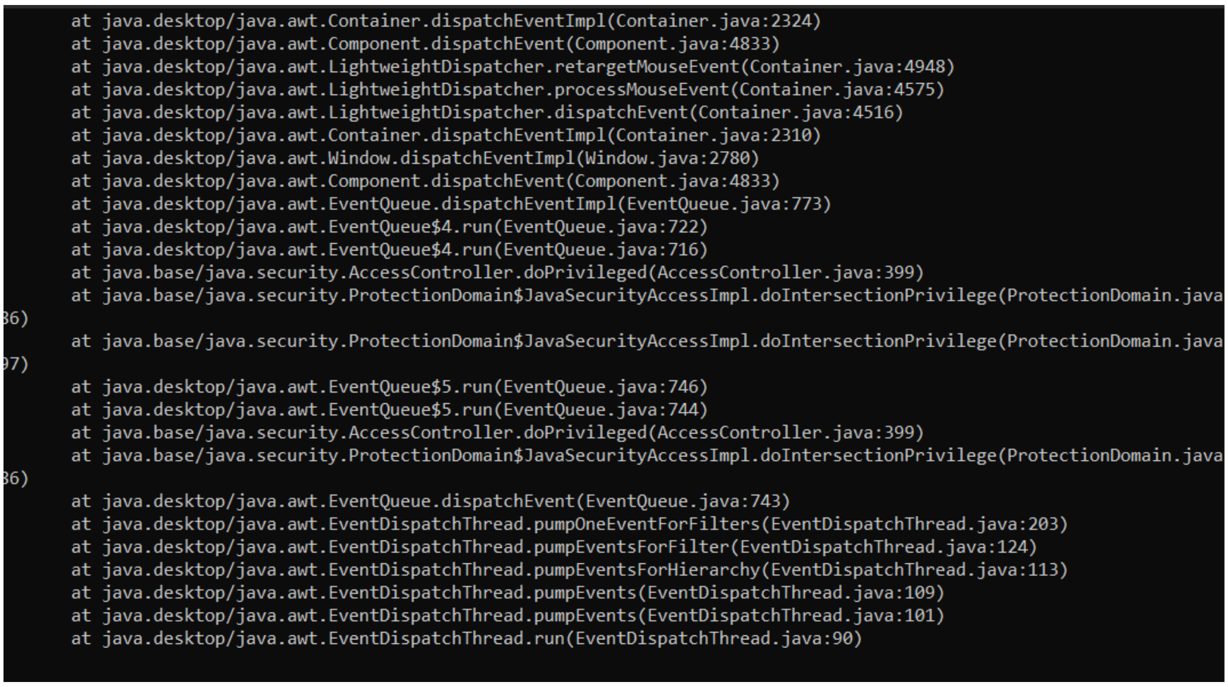
Figure 3: Opened File

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

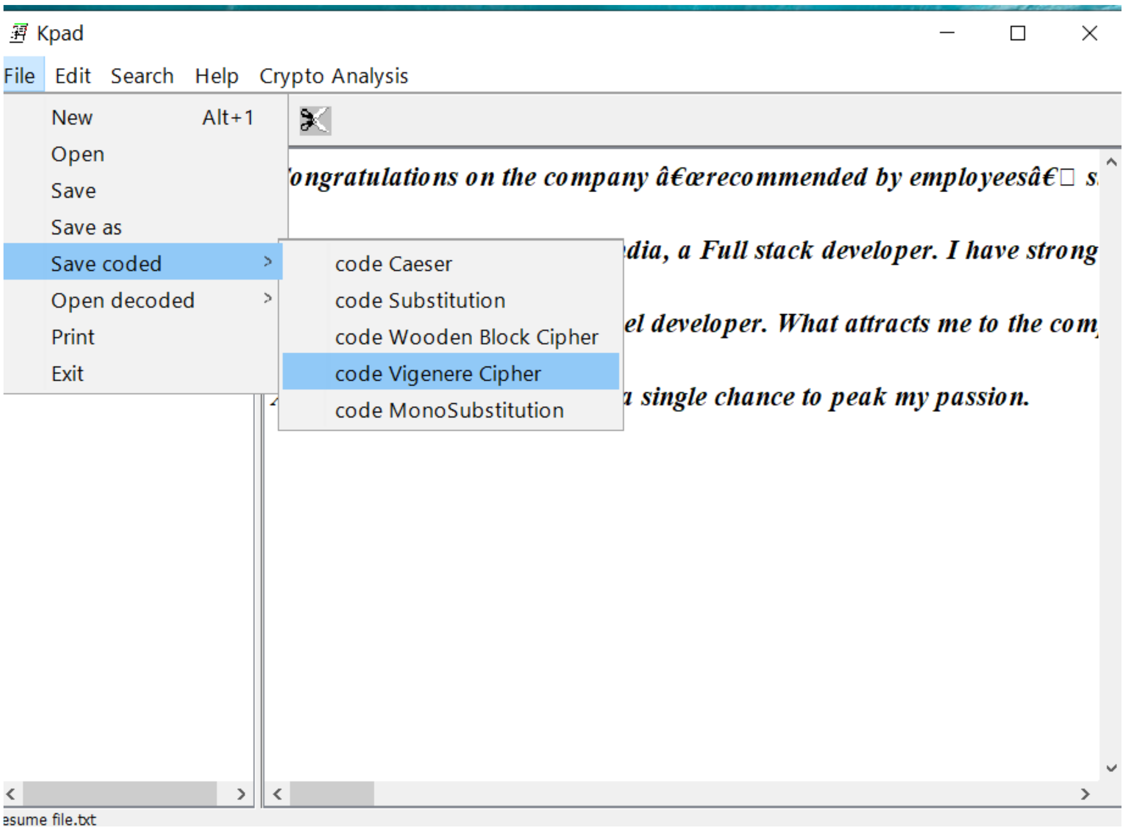
ISO 3297:2007 Certified ⁜ Impact Factor 8.102 ⁜ Vol. 12, Issue 3, March 2023

DOI: 10.17148/IJARCCE.2023.12339



```
        at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2324)
        at java.desktop/java.awt.Component.dispatchEvent(Component.java:4833)
        at java.desktop/java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4948)
        at java.desktop/java.awt.LightweightDispatcher.processMouseEvent(Container.java:4575)
        at java.desktop/java.awt.LightweightDispatcher.dispatchEvent(Container.java:4516)
        at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2310)
        at java.desktop/java.awt.Window.dispatchEventImpl(Window.java:2780)
        at java.desktop/java.awt.Component.dispatchEvent(Component.java:4833)
        at java.desktop/java.awt.EventQueue.dispatchEventImpl(EventQueue.java:773)
        at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:722)
        at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:716)
        at java.base/java.security.AccessController.doPrivileged(AccessController.java:399)
        at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java
86)
        at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java
97)
        at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:746)
        at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:744)
        at java.base/java.security.AccessController.doPrivileged(AccessController.java:399)
        at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java
86)
        at java.desktop/java.awt.EventQueue.dispatchEvent(EventQueue.java:743)
        at java.desktop/java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchThread.java:203)
        at java.desktop/java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread.java:124)
        at java.desktop/java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:113)
        at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:109)
        at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:101)
        at java.desktop/java.awt.EventDispatchThread.run(EventDispatchThread.java:90)
```

Figure 4: JAVA backend screen



'ongratulations on the company â€œrecommended by employeesâ€□ s

dia, a Full stack developer. I have strong

el developer. What attracts me to the com

a single chance to peak my passion.

Figure 5: Select cipher for encryption

DOI: 10.17148/IJARCCE.2023.12339

VI. RESULT

The document encryption and decryption system in Java offers a user-friendly interface with various features to choose from. Upon selecting the "Look" option, the user can choose from five different interface styles, including the Windows look. The toolbar offers options such as file, edit, and help, while the lower part of the interface provides the user with options to create new files, open files, and copy, paste, or cut text. The encryption process begins by selecting the file that the user wants to encrypt. The system accepts various file formats, including images, text, and png. The user can choose from different encryption methods, depending on their requirements. Once the user has selected the encryption method, the system prompts them to save the encrypted file to their desired location. Once saved, the file is encrypted and can only be accessed using the correct password. To decrypt the file, the user can follow the same steps as the encryption process. Overall, this Java-based system provides a simple and effective way to encrypt and decrypt documents, ensuring the security and confidentiality of sensitive information.

VII. CONCLUSION

Based on the above result, it can be concluded that the document encryption and decryption system developed in Java is a user-friendly and effective solution for securing sensitive information. The system offers various features, including multiple interface styles, file format support, and encryption method options, providing users with the flexibility to choose a configuration that best meets their requirements. The encryption process is straightforward, with clear prompts and instructions guiding the user through each step. The system encrypts files using a strong encryption algorithm, ensuring that only authorized users can access the sensitive data. Overall, this document encryption and decryption system provides a reliable and secure method for protecting confidential information. The encryption and decryption of documents using Java is a reliable and effective way to secure sensitive information. The process involves using strong encryption algorithms to scramble the data, making it unreadable to unauthorized users. Decryption, on the other hand, involves reversing the encryption process to restore the original document. Overall, the document encryption and decryption system developed in Java provides a reliable and secure method for protecting confidential information. Its user-friendly interface, support for different file formats, and strong encryption algorithm make it an essential tool for users who need to safeguard their sensitive data.

REFERENCES

[1] Li, J., Huang, Y., Wei, Y., Lv, S., Liu, Z., Dong, C., & Lou, W. (2019). Searchable symmetric encryption with forward search privacy. IEEE Transactions on Dependable and Secure Computing, 18(1), 460-47.

[2] Papamanthou, C. B., Stefanov, E., & Shi, E. (2014). Practical dynamic searchable encryption with small leakage. In Proc. Netw. Distrib. Syst. Secur. Symp.

[3] Cash, D., Grubbs, P., Perry, J., & Ristenpart, T. (2015, October). Leakage-abuse attacks against searchable encryption. In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (pp. 668-679).

[4] Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M. C., & Steiner, M. (2014). Dynamic searchable encryption in very-large databases: Data structures and implementation. Cryptology ePrint Archive.

[5] Asharov, G., Segev, G., & Shahaf, I. (2021). Tight tradeoffs in searchable symmetric encryption. Journal of Cryptology, 34(2), 1-37.

[6] Bellare, M., Boldyreva, A., & O'Neill, A. (2007, August). Deterministic and efficiently searchable encryption. In Annual International Cryptology Conference (pp. 535-552). Springer, Berlin, Heidelberg.

[7] Van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P., & Jonker, W. (2010). Computationally efficient searchable symmetric encryption. In Workshop on Secure Data Management (pp. 87-100). Springer, Berlin, Heidelberg.

[8] Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., ... & Shi, H. (2005, August). Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In Annual international cryptology conference (pp. 205-222). Springer, Berlin, Heidelberg.

[9] Baek, J., Safavi-Naini, R., & Susilo, W. (2008, June). Public key encryption with keyword search revisited. In International conference on Computational Science and Its Applications (pp. 1249-1259). Springer, Berlin, Heidelberg.

[10] Gao, H., Luo, S., Ma, Z., Yan, X., & Xu, Y. (2021). BFR-SE: a blockchain-based fair and reliable searchable encryption scheme for IoT with fine-grained access control in cloud environment. Wireless Communications and Mobile Computing, 2021.