AirVault

by

NC Syed Bilal Abbas

NC Sami Asghar

NC Hafsa Hameed

NC Muhammad Umer Khan

Supervised by

Assoc. Prof. Dr. Shahzaib Tahir

Submitted to the faculty of Department of Computer Software Engineering, Military College of Signals, National University of Sciences and Technology, Islamabad in partial fulfillment for the requirements of B.E Degree in Software Engineering

June 2024

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

“AirVault”

is carried out by

NC Syed Bilal Abbas

NC Sami Asghar

NC Hafsa Hameed

NC Muhammad Umer Khan

Under my supervision and that in my judgment, it is fully ample, in scope and excellence, for the degree of Bachelor of Software Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad

Approved by

Supervisor

Assoc. Prof. Dr. Shahzaib Tahir

Dated: June 24, 2024

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

DEDICATION

This thesis is dedicated to our Families, Teachers, Friends, and to our supervisor for their love, endless support, and encouragement.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues, the group members, who through all adversities worked steadfastly and most of all supervisor, **Assoc. Prof. Dr. Shahzaib Tahir** without your guidance it would never have been possible for us to complete our Final Year Project.

Plagiarism Certificate (Turnitin Report)

This thesis has 4 similarity indexes. Turnitin report endorsed by Supervisor is attached.

Name & Signature of Supervisor:

____

Assoc. Prof. Dr. Shahzaib Tahir
(Dept. of IS)

Name & Signatures of Students:

____

NC Syed Bilal Abbas
00000349794

____

NC Sami Asghar
00000343477

____

NC Hafsa Hameed
00000343678

____

NC Muhammad Umer Khan
00000331534

ABSTRACT

Think about an analog of a crucial database, which is not connected to the World Wide Web at all. This is the world of air-gapped networks which hold importance in organization like government sector, bank, and health care centers. However, the conventional approaches towards managing data in these networks are like sharp keys with their unwieldy mechanism and restricted usage. This is where AirVault comes in as a new generation in the way data is stored and managed.

AirVault is used as a secured storage environment within the air-gapped network. It ensures the documents are protected from any unauthorized access, but at the same time unlike a traditional safe the owners of the documents in AirVault are able to easily find and access the information they need. This is done using a technique referred to as searchable encryption, it is like holding a key where you look only that which is important without having to search the entire shelf like a common key.

The advantages of AirVault transcend convenience alone but also encompass organizational and time-saving efficiency. Traditionally, the data technologies are hidden within these secure networks and do not maximize data available within secure networks that enables informed and effective data decision making and innovative activities. This improved data utilization aligns perfectly with Sustainable Development Goal (SDG) 9: Promoting Industry, Innovation and Infrastructure. Finally, using AirVault assists in improving the management of the infrastructures by providing secure means for data collection and analysis with the given compartmentalization of systems.

This thesis provides a further insight into how AirVault works, how it uploads data while maintaining security, how it stores data, how it retrieves data and how it uses indexes so that users can search for data. In this paper, we assess the feasibility of integrating AirVault into operational systems, as well as the overall effect on data safety and growth in the highly sensitive world of air-gapped networks. We also delineate what further studies can be conducted to more improve secure and accessible data storage in these remote zones.

Table of Content

List of Figures

List of Tables

1. Introduction

Due to enhanced storage of data, there is increased insecurity on matters relating to the digital data and thus the need to enhance security of such data. The most secure type of network is the air-gapped network, where the systems are not connected with other networks such as Internet. Of course, such a measure of decentralization has its drawbacks and hinders data management within these networks. This proves to be most disadvantageous when handling the physical transfer of data or copying the data by oneself due to the security threats as well as slow retrieval of data for analysis.

In this thesis, we examine AirVault, a system that provides secure data management in isolated environments, particularly in air-gapped infrastructure. AirVault seeks to address this gap by serving as the critical link between air-gapped networks, which are regarded as some of the most secure local area networks, and data management functionalities that are normally required in a network.

1.1. Overview

AirVault helps to overcome the most significant issue of data storage on air-gapped networks to ensure maximum efficiency. It expresses at the high-level architecture that the AirVault server is another one located solely within the landing zone for storing and managing data. While end-to-end security enables secure communication and transmission of data, Searchable Encryption lets users search and analyze particular data.

A well-designed website and easy-to-use mobile application can create comfortable and informative environment, and the possibility of using an additional data transfer gateway with enhanced protection that can be turned on only in necessary cases allows performing controlled downloading of necessary data as required. Data is encrypted and always secured by proper channels to enhance secure communication, while the implementation of an extended security management layer ensures that the data remains protected from external tampering. First, this architecture focuses on maintaining the security of the stored data, second, it has a simple but intuitive interface, and third, the possibility of scaling up, making AirVault an attractive solution for organizations that look to leverage the full potential of air-gapped data. The world continues to embrace the storage of digital data and as such, the need for enhanced security for data especially confidential information. Workstation isolation, which may exclude the Internet connection, is considered the most secure form of connectivity. Nonetheless, it is essential to understand that such isolated networks will always present certain difficulties with data management within these networks. Previously, the exchange of data is done physically, for one to copy data to another, he or she has to move to the destination then copy, this leads to a lot of insecurity when it comes to data sharing and also affects the way data is accessed and analyzed.

In this thesis, the author presents AirVault, a technology in specifically managed for securing data management in air-gapped environments with the purpose of overcoming the aforementioned difficulties. The key concept of development of AirVault is in the fact that normal air-gapped networks are extremely secure but the functional data management can be very challenging.

1.2. Motivation: The Challenge of Data Security in Air-Gapped Environments

Securing our information in this modern world especially with aspects like financial movements, ideas, and records to health is a secure ideal. Connected networks which are physically separated from other networks and do not allow any connections with other networks are highly secure for storing sensitive information. However, this isolation presents a significant challenge: coordinating the vast repositories of data that are ensconced within these incorrigible fortresses. The process of handling data in traditional air-gapped networks has the following main drawbacks.

Challenges of Traditional Air-Gap Methods

- **Inefficiency**: The transfer of data through disks or even copying them manually poses serious problems such as being tedious, require a lot of effort and most importantly are very vulnerable to errors.
- **Security Vulnerabilities**: Physical transfers pose various security threats such as loss of content, exposure to unauthorized users or sources, and possible violation during transportation out of the security perimeters known as air gapping.
- **Limited Accessibility**: Lack of proper indexing and search ability to retrieve the actual information from the air gap to disfavor analysis and decision making. This slows down timely reactions to various occurrences and holds back the use of significant data analytics.

Such limitations pose a challenge to organizations that could otherwise capitalize on the benefits of stored data within an air-gapped environment but with an equal level of security maintained and in place.

Impact on Organization Operations

- **Delayed Decision-Making**: Slow data access reduces opportunities and hinders quick and adequate reactions.
- **Reduced Efficiency**:Management of data that is poor proves to be costly, especially in the future when the businesses grow and need to accommodate huge volumes of data.
- **Stifled Innovation**: The problem of challenging access to detailed information exposes limitations in the conduct of profitable and effective business analysis.

The Solution: AirVault

AirVault gives a solution to these issues because it is a data management system, secure and efficient in air-gapped environments, which proved to boost the accessibility, efficiency, and security.

1.3. Proposed Solution: Introducing AirVault - A Secure Data Management System with Searchable Encryption

AirVault is being presented as a novel concept to address the complex issue of data management in isolated networks. The plan for this system is to achieve an extremely high level of security by integrating a searchably encrypted database technology to allow searches of data while it remains encrypted or at least encrypted in manners which cannot be read by anyone not authorized to do so. Key features of AirVault include:Key features of AirVault include:

1. Secure Data Movement

AirVault does not come with the usual risks that are incurred when handling physical transfers or having manual copies of data. As earlier said, it uses encryption protocols that are appropriate for segmented networks to guarantee the security of the data transferred within the air-gap environment.

2. Searchable Encryption

AirVault enhances the possibilities usage of searchable encryption. This means that, users can immediately have a speedy access or retrieval of information within the air gap itself thus reducing time and security risks that are posed by data circulation.

This feature makes the data search unearthing as flexible as possible by keeping the data encrypted as well as guarded. This implies that exceptional details can be queried safely, thus keeping sensitiveness and working firmly towards the particular security norms recommended.

3. Effortless Search and Access

Forget downloading entire datasets!AirVault takes advantage of what is known as 'Searchable Encryption', this is the ability of an item to be search on a database without being humanly readable. There is no need to transfer data around the air gap because the ultimate information of interest for the user is already within the air gap itself, and so the user can find what is relevant and useful within a short span of time at least from the air gap point of view.

4. Intuitive User Experience

AirVault prioritizes user-friendliness. Mobile and web applications with a well-designed interface provide a seamless experience for data upload, search, and management within the air-gapped environment. No complex technical expertise is required.

5. Identity-Based Access Control

AirVault efficiently solves this problem by implementing Identity-Based Access Control to provide the necessary balance between data usability and protection within air-gapped networks. For data access only those who have access rights will be able to access the data, while any restricted data is protected from unauthorized users.

1.4. Working Principle: A High-Level Overview of AirVault's Functionality

AirVault solves the problem of storing protected data safely from unauthorized entities and not being able to access it due to isolation from internet. Here's a high-level overview of its functionality:Here's a high-level overview of its functionality:

Data Flow

1. **Data Upload:** Data is loaded into AirVault securely by authorized users under a localized air-gapped environment, available for use through a web or mobile user interface.
2. **Data in storage:** Upon uploading the information into the site, AirVault then follows standard measures to protect the information by encrypting it using strong encryption

techniques. This encrypted data is then securely stored within the AirVault server, this server is housed completely within the Air gap.

3. **Data in Transit:** All communication between the user applications and the AirVault server occurring within the air gap is made even more secure to utilize application-specific encrypted connections with reliable protocols.
4. **Data Download:** When the file is downloaded, information will be sent in the form of a encrypted data between the server and the client to ensure the user is able to open the obtained document without any outside interference.

Encryption and Search

- **Searchable Encryption:** AirVault utilises a strong technology called the searchable encryption. This would ensure that only those with the permissions to look for specific details are able to search through the encrypted data that is uploaded to the AirVault server.
- **Search Queries:** This is the case when the trapdoor which allow users to submit search queries (trapdoor) can employ the use of web or mobile apps. All these queries are safely executed within the gates of an air-gap environment.
- **Relevant Results:** The actual data itself, while being processed in a search query to provide the user with findings, is never exposed in an unencrypted format. This reduces the scope for data transfer, thereby reducing security concerns related to movement of data within the computer.

Overall, AirVault prioritizes

- **Data Security:** Robust encryption, secure communication channels, and controlled data transfer gateways ensure data integrity and confidentiality at all stages.
- **Search Efficiency:** Searchable encryption empowers users to locate specific information quickly, eliminating the need for full data downloads.
- **User-friendliness:** Intuitive web and mobile interfaces provide a seamless experience for data management within the air-gapped environment.

1.5. Operating Environment

Hardware

1. Client Side

On client side, we don't have any specific hardware requirements, but we will test our solution on the system having the specifications:

- **Processor**: Intel(R) Core(TM) i7-8650U @1.90GHz. 4 Cores, 8 Threads
- **RAM**: 16 GB, 2400MHz DDR4

2. Server Side

On server side, AirVault will use Raspberry Pi 4 Model B having the following specifications:

- **Processor**: Broadcom BCM2711 @1.8GHz. 4 Cores, 4 Threads
- **RAM**: 4 GB, 3200MHz LPDDR4 SDRAM

Software

- **IDE**: Pycharm and VS Code
- **Website**: NextJS
- **Mobile App:** Flutter SDK
- **Backend**: Python
- **Database**: SQLite

1.6. Objectives

The research aims to achieve the following objectives:

1.6.1. Overall Objectives (Project Goals):

- Analyze the limitations of traditional data management methods in air-gapped environments.
- Assess the performance of the AirVault and its viability to close the gap between security and accessibility when implemented in the air gapped environment.
- The purpose of this knowledge check is to evaluate the importance and find out the effects of AirVault on operation efficiency, data secure, and all data using in the air-gapped environment.

- Examine possible markets and segment the key areas of application for AirVault's services, especially claiming that the company addresses the needs of various user groups across important industries and sectors.

1.6.2. Academic Objectives (Research Contributions):

- The proposed solution should therefore help to further the research in the application of secure data management solutions needed for air-gapped networks.
- As a result, this paper aims at conducting an analysis on the structural makeup and operational capabilities of AirVault as well as the potential advantages that organizations, which exist within closed network domains, stand to gain from the tool.
- Assess the workability of searchable encryption and Identity-Based Access Control (IBAC) in achieving secure and efficient access to data cross-air gaps.
- Provide the necessary information for researchers and practitioners who want to apply appropriate strategy for data protection as well as access to them within strict DNN environment.

In this way, this thesis offers a contribution toward both work on secure data management specifically for air-gapped networks and the tracking and examination of the broader field of research into DC NM and SC. The work can act as a roadmap for organisations that are in a process of implementing big data platform and ensure maximum use of data while operating at the same time within secure bubbles.

1.7. Scope

Secure data management and its guaranteed protection in enclosed, or air-gapped, systems, the authorization of the users, and the encryption of data storage, as well as, its search function, and control access. It also encompasses the incorporation of searchable encryption methods allowing for proficient search into the encrypted content.

Data Security and Protection

- Air-gapped systems imply high seclusion from the internet with restricted accessibility and secured storage spaces to protect the data.
- Some of the enhanced practices to be implemented include rigorous means of authenticating the users, storage of sensitive information in encrypted form and limiting the access to the data.

Efficient Data Retrieval with Searchable Encryption

- Incorporate searchable encryption to allow system for efficient data retrieval.

- Enables users to securely search the encrypted data within its own or all the departments without exposing the original data to the search engine, enhancing both data security and accessibility within the constraints of air-gapped environments.

Enhanced Access Control

- Specify and implement strict access control mechanisms to regulate and monitor user permissions using Identity-Based Access Control.
- Reduce the risk of unauthorized access to sensitive information by enforcing robust access control measures.

1.8. Non-functional Requirements

Performance Requirements

- Server must respond within minimal time.
- System must inform the user of a timeout when no response is received.
- Administrators must be warned if the system storage is nearing capacity.

Safety Requirement

- Server should be maintained in secure custody
- System should not allow any unauthorized user to access any functionality of AirVault.
- System should always double-check the kind of access, user has before giving the response.
- Secret Keys shouldn’t be shared with users.

Security Requirements

- Search query should be encrypted, so no one can see the exact query.
- Files should only be decrypted on the AirVault server, so no one is able to decrypt the files on their local machine.

Software Quality Attributes

AirVault should have balance between all three characteristics of the triangle of Searchable encryption which are:

- Security
- Searching
- Performance

Availability

System should be available all time 24/7 to users of the organization.

User Friendly

System should be user friendly and very easy to use, the system should follow these key factors accordingly.

- **Clarity**: The system must indicate what is loading & what all actions do.
- **Informative**: The system must inform the user of any issue.
- **Efficiency**: A user can perform an action in a minimal number of steps. No more than 2 or 3
- **Intuitiveness**: It must be clear to the user how to use the system with minimal instructions/tutorials.
- **Low perceived workload**: The system does not appear complex.

Data Integrity

- User data must not be corrupted.
- User data must not be lost due to system failure.
- User data must not be deleted by new data or modal migration.

Code

- Security of all 3rd party libraries must be verified.
- The code has high integrity.
- Server-side code must be stored outside the web server's root.

Localization

- System must be available in English

Business Rules

- Only authorized users can search keywords from encrypted files.
- Users must store their private key on a secure location in their local machine.

1.9. Deliverables

The final outputs of the AirVault project will include a suite of software applications and comprehensive documentation, ensuring robust implementation, usability, and maintenance. These deliverables are

AirVault Web Application

- A fully functional web-based platform for secure data management.
- Features include user authentication, encrypted file uploads and downloads, searchable encryption, and administrative controls.

AirVault Mobile Application

- A mobile version of the AirVault platform developed using Flutter.
- Provides similar functionalities as the web application, optimized for mobile devices.

Backend Services

- FastAPI-based backend implementing core functionalities and API endpoints.
- Includes integration with the Key Management Service (KMS) for key management.

Software Requirements Specification (SRS)

- A detailed document outlining the requirements for the AirVault system.
- Includes functional, non-functional, and technical requirements.

Software Design Document (SDD)

- A comprehensive document detailing the architectural and functional specifications of AirVault.
- Includes use cases, process flows, design details, and UI design principles.
- These deliverables collectively aim to provide a secure, efficient, and user-friendly solution for managing sensitive data in air-gapped environments, ensuring the successful implementation and long-term sustainability of the AirVault project.

1.10. Alignment with Sustainable Development Goal (SDG)

The AirVault project aligns with **Sustainable Development Goal 9: Industry, Innovation, and Infrastructure**. SDG 9 aims to build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation.

How AirVault aligns with SDG 9:

Resilient Infrastructure

AirVault enhances the resilience of data management systems, particularly in critical sectors like healthcare, finance, and government, by ensuring secure and reliable data storage and retrieval in air-gapped environments.

Promoting Innovation

AirVault is providing a novel solution by implementing searchable encryption in an air-gapped network with multi-user access control which makes data handling and querying easy specially for the sensitive data.

Inclusive and Sustainable Industrialization

AirVault is supporting industries by offering them a comprehensive secure data storing solution that can work across different fields so that the data confidentiality remains there.

Infrastructure Development

AirVault is contributing to build modern, efficient, and secure IT infrastructure. This is very important for the growth of digital world and the protection of sensitive information.

By addressing these key aspects, AirVault not only meets immediate data security needs but also supports broader goals of sustainable industrialization and innovation, in line with SDG 9.

2. Literature Review

This chapter explores existing techniques to manage secure data in air-gapped environments and it also explain limitations of existing tools. It also proposes AirVault, an advanced solution that employs searchable encryption to get beyond these limitations. Additionally, the chapter goes into the idea of searchable encryption and explores different approaches essential to AirVault's functioning.

2.1. Existing Secure Data Management Techniques for Air-Gapped Environments

While AirVault emerges as a novel solution, air-gapped environments have traditionally relied on several techniques to manage sensitive data securely. Here's a look at some relevant techniques:

Manual Transfers with Sanitization

- Data is physically transferred out of air gap on removable media (e.g., external hard drives).
- Before transfer, the data undergoes rigorous sanitization procedures to overwrite or erase any sensitive information.
- Once outside the air gap, the data can be accessed and managed using standard tools on a separate network.
- **Security Concerns:** This method is time-consuming, prone to human error, and introduces security vulnerabilities during the physical movement of the media. Lost or stolen drives can compromise data confidentiality.

Multi-Factor Authentication (MFA) for Access Control

- This technique strengthens access control within the air gap by requiring users to provide multiple verification factors (password, token, biometric) to access and manage data.
- MFA adds an extra layer of security, making unauthorized access more difficult.

Data Encryption at Rest and in Transit

- Encryption scrambles data into an unreadable format using encryption algorithms and keys.
- Data at rest (stored within the air gap) can be encrypted to ensure confidentiality even if the storage device is compromised.
- Data in transit (during transfers within the air gap) can also be encrypted to minimize the risk of unauthorized access if intercepted.

Network Segmentation

- It also divides the network into sub-networks in order to effectively contain important information within one or more layers of air gap.
- Each segment operates at a different level of security where access to some data cannot be obtained except through authorised means and in case of a security breach in one segment, it does not affects the other segments.

Role-Based Access Control (RBAC)

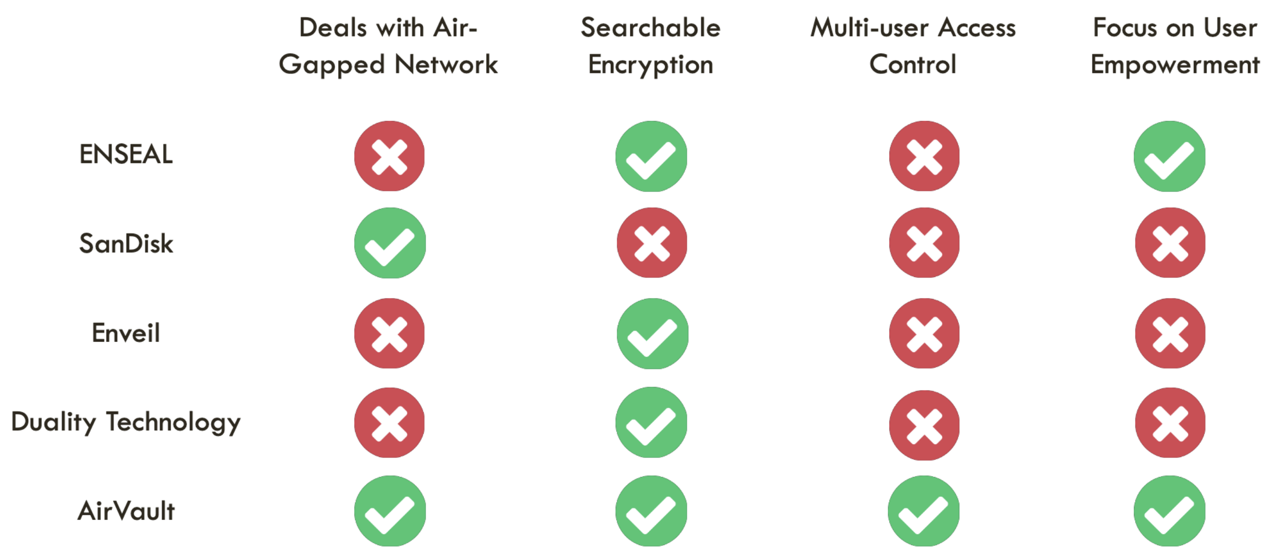
- RBAC provides the user with the authorization to use some data and functions depending with the role of the user allowed in an organization.
- This also minimizes chances of some staff members to access other information they are not supposed to since their access will be limited to only what they need for their operations.

Limitations of Existing Techniques

While these techniques offer some level of security, they fall short of addressing the core challenge of balancing data security with accessibility within air-gapped environments:

- **Manual Transfers:** They are usually slow and time-consuming and render security vulnerable to attacks.
- **MFA, Encryption, Segmentation, RBAC:** These methods have security but are powerless to handle the basic problem of information search and analysis within the air gap. Even in traditional methods, data must be transferred for analysis, which is inconvenient.

Following are some competitors comparison:



	Deals with Air-Gapped Network	Searchable Encryption	Multi-user Access Control	Focus on User Empowerment
ENSEAL	✗	✓	✗	✓
SanDisk	✓	✗	✗	✗
Enveil	✗	✓	✗	✗
Duality Technology	✗	✓	✗	✗
AirVault	✓	✓	✓	✓

Figure 1: Comparison with competitors

AirVault, intends to fill this gap through the incorporation of searchable encryption and secure data management features evolved for air-gapped situations.

2.2. Searchable Encryption: Concepts and Applications

AirVault's core functionality mainly depends on a strong concept of searchable encryption. This section sought to understand the various aspects of searchable encryption and the various possibilities and operations including AirVault within air-gapped networks.

Concepts of Searchable Encryption

Conventional encryption makes data non-indexed, and, thus, invulnerable to different malicious attempts but useless for searching through content. Searchable encryption fills this gap by empowering users with necessary permission to search for certain data without necessarily having to decrypt the entire data base. Here's how it works:

1. **Data Owner:** Encrypt the data under Searchable Encryption Scheme and preserve it into the air gap (AirVault's server).
2. **Search Token Generation:** If a particular user wishes to find out some information, there is what is referred to as search query. From the search feature of AirVault's system, a safe search token is therefore created from the typed term.

3. **Search over Encrypted Data:** It is used to make search on the encrypted data without an actual decryption of the required data. Such data elements are only the ones that are filtered for search requests by the tool.
4. **Relevant Results:** The encrypted data that matches the search query is returned to the user.

2.2.1. Searchable Encryption Schemes

There are many types of searchable encryption scheme that provides different levels of security features depending on the requirements and the constraints. Here's a brief mention of some common schemes:

2.2.1.1. Symmetric Searchable Encryption (SSE) for Air-Gapped Networks

Concept: Data is encrypted with a single key before it is stored inside the air gap. It is also used for decryption of the message at the end of the same key.

Suitability: It is effective and in any situation where the user who is searching for some data is the same person who feeds the system with such data.

Security Considerations: Simple yet effective, SSE might not be as hierarchical with regards to query capabilities. At the same time, certain risks are possible, namely, the threat of disclosing information, depending on the specific scheme.

2.2.1.2. Asymmetric Searchable Encryption (ASE) for Air-Gapped Networks

Concept: The data is encrypted using an asymmetric (public key) encryption scheme prior to being stored within the air gap.

Suitability: It is used whenever the person who is looking for the specific data is a different person from the one who is inputting the data.

Efficiency Considerations: ASE might be less efficient in terms of overall compared to SSE even though it provides for faster searches. A good example of ASE is Public Key Encryption with Keywords Search (PEKS).

2.2.1.3. Homomorphic Encryption (HE) for Air-Gapped Networks

Concept: This type of encryption permits computations to be conducted on the encrypted data which is basically encrypted information also known as cipher text within the air gap. This preserves anonymity of data while some statistical manipulations are possible. Some of the most frequently employed operators include addition and multiplication for processed data.

Performance: HE schemes are usually slower, at least when compared with traditional encryption methods since they involve additional computation. Homomorphic Encryption is of three types namely Partial Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SWE) and Fully Homomorphic Encryption (FHE) and each of them has its own limitations and capabilities therefore the selection of HE should depend on the necessity of business. Fundamentally, HE can be implemented with the help of symmetric key cryptography as well as asymmetric key cryptography.

Public Key Encryption with Keyword Search (PEKS): It allows searching by using keywords as is, without the possibility of the search results containing words that are similar to the keywords used.

Predicate Encryption: The Advanced Search can support more search terms and criteria than a keyword search.

2.2.2. Search Operations Supported for AirVault

Concerning the searchable encryption strategy, AirVault targets secure search on data and could be centered on functionalities that complement efficient data management in the context of air-gapped computing. Here are some potential search operations AirVault support

- **Keyword search**: This means that users can use keyword or phrase to search for any data that is encrypted within the database.
- **File name search:** The files can also be searched according to the name of the files.

By using searchable encryption, AirVault allows the users to search for the required information in the air gap in an effective manner without violating the main prerequisites of data security. This enhances data access and simplifies data analysis procedures in isolated networks to a much higher degree.

2.3. Challenges and Research Gaps in Air-Gapped Data Management with Searchable Encryption

It is easy to see that air-gapped networks pose certain problems concerning data management and searchable encryption. Here's an exploration of these challenges and how AirVault potentially addresses some of the existing research gaps:

Challenges in Air-Gapped Data Management

- **Limited Search Functionality:** The commonly used techniques to search data within air gaps are decryption or sending the copies of the data which neutralizes the air gaps.
- **Data Inaccessibility:** Tasks such as information transfers between various sectors can be slow through manual handling and the search functions of given datasets may also be extremely limited which can significantly delay access to crucial data for analysis or decision making.
- **Security Concerns with Data Movement:** Most data storage media are transferred physically, and this draws the attention to weaknesses and threats, including inadequate security and data flow constriction.

- **Balancing Security and Efficiency:** Most solutions do not inherently provide a balance in creating a secure air gap as well as quickly accessing the information that is stored inside it.

Research Gaps in Searchable Encryption for Air-Gapped Networks

- **Limited Query Complexity:** Certain searchable encryption techniques face problems where the search is more sophisticated than merely looking for keywords. This can prove to be disadvantageous in the sense that it limits data discovery within those air gaps.
- **Information Leakage Risks:** Some of the searchable encryption methods may bring some certain information leakage issue, thus threatening data security.
- **Performance Optimization for Air-Gapped Environments:** Making searchable encryption work in air-gap environments may need to be fine-tuned in ways that might pose certain performance issues when implemented within low end systems.

2.4. How AirVault Addresses Unmet Needs

How AirVault Addresses these Gaps

- **Focus on Efficient Search:** As for the air gap, AirVault uses searchable encryption that helps to form efficient search functions within the air gap. The data owners do not need decrypt the whole set but can provide the concrete information the users are looking for, which increases openness of the information.

- **Reduced Reliance on Data Movement:** AirVault also helps avoid the situation when there will be a need to transfer files between the groups, which can be also a security issue, as well as a performance issue.
- **Security Focus:** AirVault values the security of data because it is providing secure text search solutions that can reduce the leakage of information.

AirVault as a Potential Solution

Therefore, though the research is still ongoing, AirVault's concept of searchable encryption for large-scale distribution appears to be highly workable in dealing with some of the main concerns and deficiencies in air-gapped environments. As a secure data storage service which can support efficient search and retrieval and may possibly replace the existing data migration in these individual closed networks, AirVault provides more effectiveness to data access and security and subsequently the use of data in such enclosed environments.

Additional Research Areas

- Developing efficient practices for the performance-maximizing searchable encryption when implemented within an air-gapped environment characterized by minimal resources.
- Possible additional features beyond keyword search beyond the air gap's capabilities, relying on the further development of machine learning or natural language processing engines.
- Looking at new constructions of searchable encryption schemes which can provide higher security than the existing ones and yet enable efficient searching.

To advance the searchable encryption further for use in protection of air-gapped devices' data, researchers should continue with research in these areas.

3. System Design and Implementation

This chapter discusses the comprehensive structure of the AirVault system, which optimizes for secure data handling in air-gapped environments. Strong safety precautions are given the highest priority in order to protect sensitive information while maintaining essential cryptographic principles of confidentiality, availability, and authenticity.

3.1. System Architecture

AirVault's system architecture is made to function solely in air-gapped cases. It is characterized by various secure operating procedures that help to protect data within an air-gapped space. It has a number of components that are interrelated and serve to meet necessities of the three major characteristics of cryptography, which are; confidentiality, availability and authentication.

Confidentiality

Ensuring the privacy of data is pivotal in AirVault and therefore, any data regardless of the status, as it is being stored, as it is passing from client to server or from server to server, is encrypted. Using advanced cryptography techniques, information is encoded. For data encryption we are using AES-256 and RSA with key size of 2048.

Architecture also includes secure key management practices, where encryption keys are stored in a secure manner on the server. This ensures that even if data is intercepted or accessed by unauthorized individuals, it remains secure as no information can be extracted from that encrypted data.

Identity-Based Access Control (IBAC) for Data Accessibility

Availability within AirVault is implemented through precise access control mechanism. We are using Identity based access control for accessing of data. This method ensures that data is accessible to authorized users at all times within the highly secure and isolated air-gapped environment.

1. Identity Verification

   Users must verify their identity through a secure authentication process before accessing any data. In AirVault we are using JSON Web Token (JWT) for user authentication. As the user requests to access the data, first he has to authenticate himself and then he can only access data if he has permission.

2. Individual Permissions

   Access rights are assigned based on individual user identities. Each user is given permission to access certain folders. A user can be editor or viewer of a folder, if he is a viewer he has only view files/folders and search for keywords within files. But if he is an editor he can create folder, upload files, update files and delete files. Only owner of the folder is allowed to change the access permissions. Only he can grant or remove access.

3. Access Management

   Owner can change access permissions, he can grant and remove access. In addition to that as the user is removed from the system the ownership of all its owned files and folders will be shifted to server admin.

Authentication

Authentication in AirVault is enforced through JSON Web Token (JWT) and our robust access controls. Each user must prove their json token to send request to the server. Server will process the received request from the user if and only if the token proved by the user is valid. This minimizes the risk of unauthorized access and potential data breaches.

Architectural View of AirVault

AirVault adopts a client-server architecture model, wherein the admin or user acts as the client, and the Raspberry Pi serves as the server in alignment with the model. AirVault facilitates users to securely upload files, enabling seamless search functionality across the uploaded data using single search queries. The server will have storage to store encrypted data and execute searches based on user-provided queries, ensuring robust data security and retrieval capabilities.

Our system has following components

1. Clients

   The client side of our system includes both a mobile app and a web app. Our mobile app, AirVault, offers users the convenience of access to documents in air-gapped network. This feature allows users to download documents to their devices, view them without an internet connection, upload new files, and share documents easily. Our web app, on the other hand, is accessible to all web browsers as it is built with web standards.

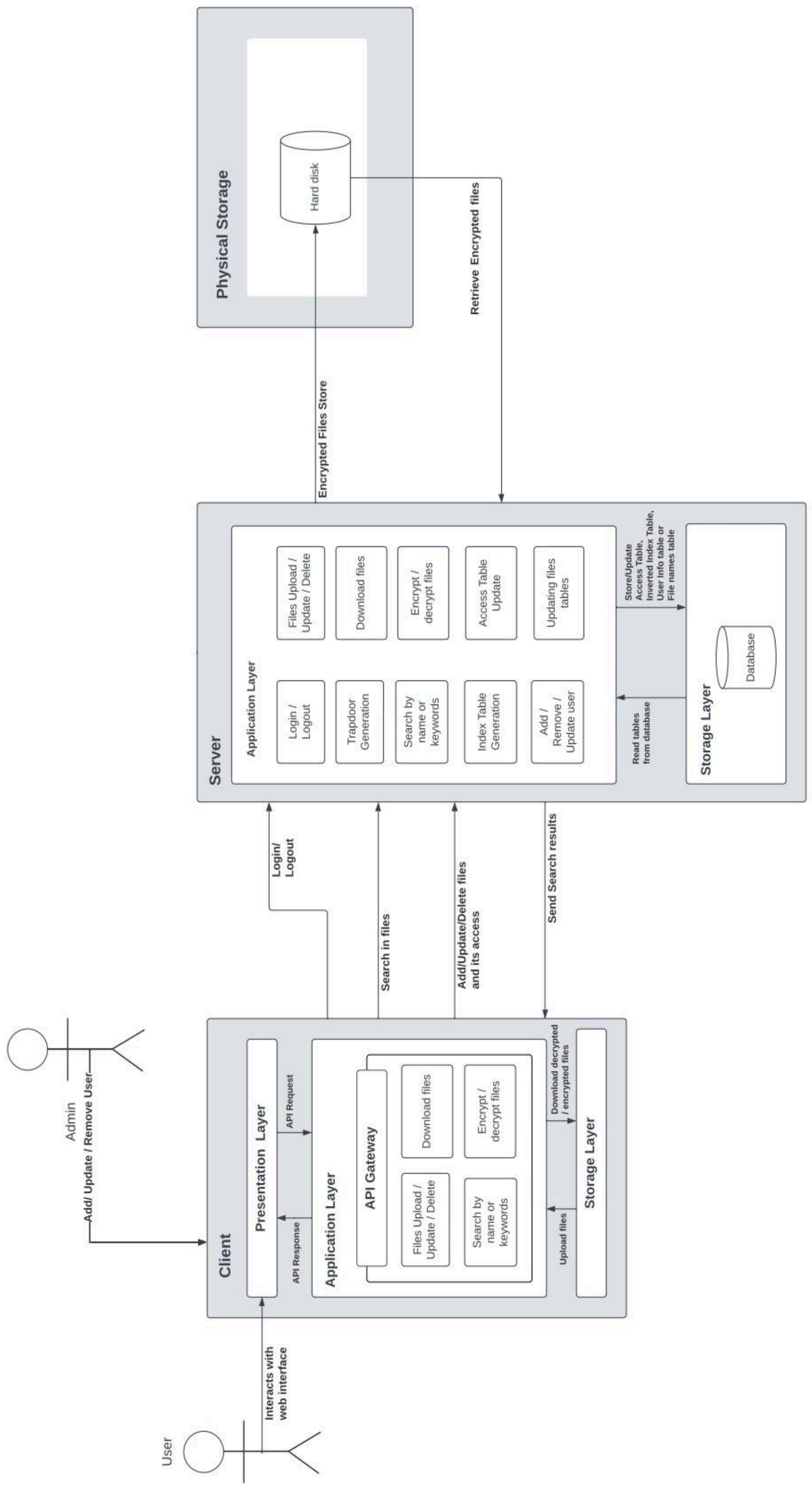


Figure 2: System Architectural Diagram

Each user can have his personal server that would work better and safely from the rest of the servers due to safe user data isolation.

2. Multi-Servers:

   The hardware setup of our system is such that every server is assigned to something as a department. These users have a relationship with one server only, and a user can only perform requests on that specific server only.

   This design make it almost impossible for one user to transact with other departments other than interaction with his or her department in the organization hence establishing certain order within the organization. About users, they can only send requests from their own servers, and the servers, in their turn, can both send and receive requests. This inter-server communication aids in sharing of data at such departmental level.

   To access data on a different server, a user is given permissions. This ensures controlled and secure data access across departments. Thus only registered users can log in to their specific department servers.

3. Storage and Retrieval

   All files are stored in a shared storage system where they are always kept encrypted to maintain security. This encryption ensures that the data is unreadable to anyone who is not authorized user of our system.

   When a file is requested and retrieved, it is transmitted from the server to the client over the network in an encrypted form, this ensures that the data remains secure while transit. Our system can store files up to 5MB. Larger files takes longer time to upload due to larger processing and more keywords generation.

4. Searching within files

   Inverted Index Table is used for faster search within files, keywords are extracted from the uploaded files, they are encrypted and then stored in database with the file in which they occur. This Inverted Index Table speeds up the search process within the files. This feature makes it possible to search for file based on keyword detection this is much more convenient than the process of scanning through content of the each file. Thus, the specified method is essential for our system, as it allows to provide really fast and safe search.

5. Database within server

   AirVault has database within the server which that stores all information related to users and servers. This database also manages permissions for files and folders, ensuring that access controls are properly enforced. In addition to user and permission data, the database stores file metadata, which includes information such as file size, type, and creation date. This metadata is essential for organizing, managing, and retrieving files efficiently.

6. Encryption at each level

   By using hybrid cryptography to protect the files that are stored within AirVault, its system is safe from potential threats. To ensure the security of files as well as keywords, it also employs AES-256 encryption with the ability to enhance the security of its data. Also, AES-256 keys themselves are encrypted by the RSA algorithm in order to bring the perception of higher level of security. Symmetric and asymmetric methods both boost overall security making this method efficient for encrypting data. In addition, any raw code information transmitted over the network is enciphered; for this reason, no information passed through the network at any stage can be intercepted by unauthorized personnel.

3.2. Data Handling and Encryption Process

Data safety is another important area, which is adequately handled by AirVault through encryption and decryption at storage area and transmission. AirVault System ends to end encrypted to protect the files and all other data communicated. End to end encryption is as follows:

Uploading of file

1. Client Side encryption

   As a file is uploaded, it is encrypted using AES-256 algorithm, a great symmetric encryption algorithm which is effective on the client side. AES 256 is then encrypted by the server's public key RSA 2048 and this would provide additional security. The file is also encrypted here and this encrypted file along with the key is sent to the server; this way every detail remains protected until it is transferred.

2. Server side encryption

   The AES-256 key is encrypted with the sender's public RSA key and when the server receives this it uses their own private RSA key to decrypt this. But before it stores the file, the server re-encrypts it with the AES-256 that belongs to the server itself. This double encryption ensures even if some hackers try to penetrate the storage system to seek files, they will be of no help as they cannot comprehend what the files contain. This also ensures the enablement of the server AES-256 key so that it is not taken outside the server.

Retrieval of file

1. Client Side encryption

   During download, the file is retrieved by the server and decrypted with the AES-256 key used to encrypt it. In the second stage it is written with a new one-time-generated AES session key and encrypted again. It is a one-time key and it is encrypted with the help of user's public RSA key for transmission, which means that only the user can decrypt this key. Finally, the encrypted file and key are sent to the client, in the decrypted form as well but all data are encrypted during this process.

2. Server side encryption

   There are two levels of security on the client side and the first one is to use the user's private RSA key to decrypt the one-time AES key and then use the decrypted AES key to decrypt the file. This method also ensures that no unencrypted data is stored on other machines other than the server and hence minimizing on the risks that come with transfer of data. Nevertheless, as long as these data are intercepted, it is impossible to decode without the appropriate keys. Such an approach to using both the symmetric and the asymmetric cryptography allow AirVault to ensure that data is securely stored and, as well as retrieved with requisite protection for it to receive perpetual protection.

3.3. Searchable Encryption Implementation

Searchable encryption in AirVault can be explained in details as follows.

On the client-side

- If the user want to search something just typing a certain keyword in the search bar, the application will encrypt the keyword using the RSA key of the server. It ensures that only the server can decrypt the keyword.
- It then sends this encoded keyword to the server as a search query.

On the server-side

- Upon receiving the request, the server applies its private RSA key to decrypt and restore the original keyword from the given encrypted data.
- The server then decrypts the user's retrieved keyword with the AES 256 version of symmetric key.

- The server compares the encrypted keyword to entries contained in the indexed table that the server holds.
- If a match is found, the server returns the number of files containing the keyword as well as a list of file IDs contained in the index.

Authorization and Response

- This enables the server to check whether the user has the required permission to the identified files.
- When access is allowed, the server collects information on the files effectively permitted in the execution of the program.
- Lastly, it encrypts the authorized file information using the user's public key from RSA encryption. This keeps the response only accessible to the user and ensures it is encrypted.
- The server proceeds to encrypt an array that contains information that identifies the acceptable files within the directory and sends it back to the client side.

Client-side decryption and display

- Upon receiving the following response, the client application decrypt, with the help of the private key, an array of objects which contains information about the files.
- The decrypted data is then displayed on the search page where the user can see the specific information regarding the search results

3.4. Secure Data Transfer within the Air-Gap

The data on AirVault is transferred from one place to another in a very security-conscious manner. Let us move towards understanding the architecture of file downloads that occur when a user wants an encrypted file to be decrypted and then downloaded.

On Client side

- When the user clicks a file to download this file the system comes up with a file identification number. This ID is then encrypted using the server's private RSA key so that the ID can only server can see the file ID.
- The encrypted file ID is then sent to the server as a download request which returns the file to the client.

On Server Side

- The encrypted file ID is decrypted using the user's public RSA key, revealing the original file ID.
- The server locates the requested file. To enhance security, the server encrypts the file with a one-time AES-256 key.
- Additionally, the server encrypts the AES-256 key with the user's public RSA key.
- Finally, both the encrypted file and the encrypted AES-256 key are sent back to the user's machine as a response.

Download File in Machine

- The user's machine first decrypts the AES-256 key using its private RSA key.
- This retrieved key is then used to decrypt the actual file on the user's side.
- Once decrypted, the file is stored locally on the user's machine.

3.5. Access Control Mechanisms

AirVault manages access permissions on a folder-by-folder basis. When you create a new folder, you can assign access rights to specific users. Users inherit access to files within a folder based on their access level for the parent folder. There are two access levels: 'editor' and 'viewer'.

- At this level, only ‘viewer’ access allows the user to download files from the folder.
- The users with access level ‘editor’ have more privileges that include the creation of new files, deletion of files, and downloading of files.

To change permission to a folder, you need to right-click the folder and then select the “Permissions” option. This will open a modal window showing the current users and the role they are assigned to. You can select a particular user and then adjust their permission levels by clicking on the “Update Access” link. After the access level has been changed on the server, the user will be able to work with the folder accordingly to the new level of access granted.

Every time a user tries to access files within a folder, AirVault checks the user’s authorization level for that specific folder. This means that they will be permitted to perform the desired action if their access level includes the authority to do so.

3.6. System Diagram

3.6.1. Use Case Diagram

The use case diagram for the AirVault system is outlined below, highlighting three actors: user admin and server. This diagram provides an overview of the expected behaviour of the actors.

For administrators, the diagram shows essential processes such as adding, updating, or removing users from the system and managing user permissions as well.

Users can manage files by adding, deleting, and updating files based on their access permissions. Moreover, users can conduct searches for files by name or keywords present in the files.

On the server end, the inverted index table, access table and file info table is updated whenever a new file is uploaded. Whenever a user enters a query to search for files, a trapdoor is generated in the backend for the searching purposes. These functions collectively make the AirVault system.

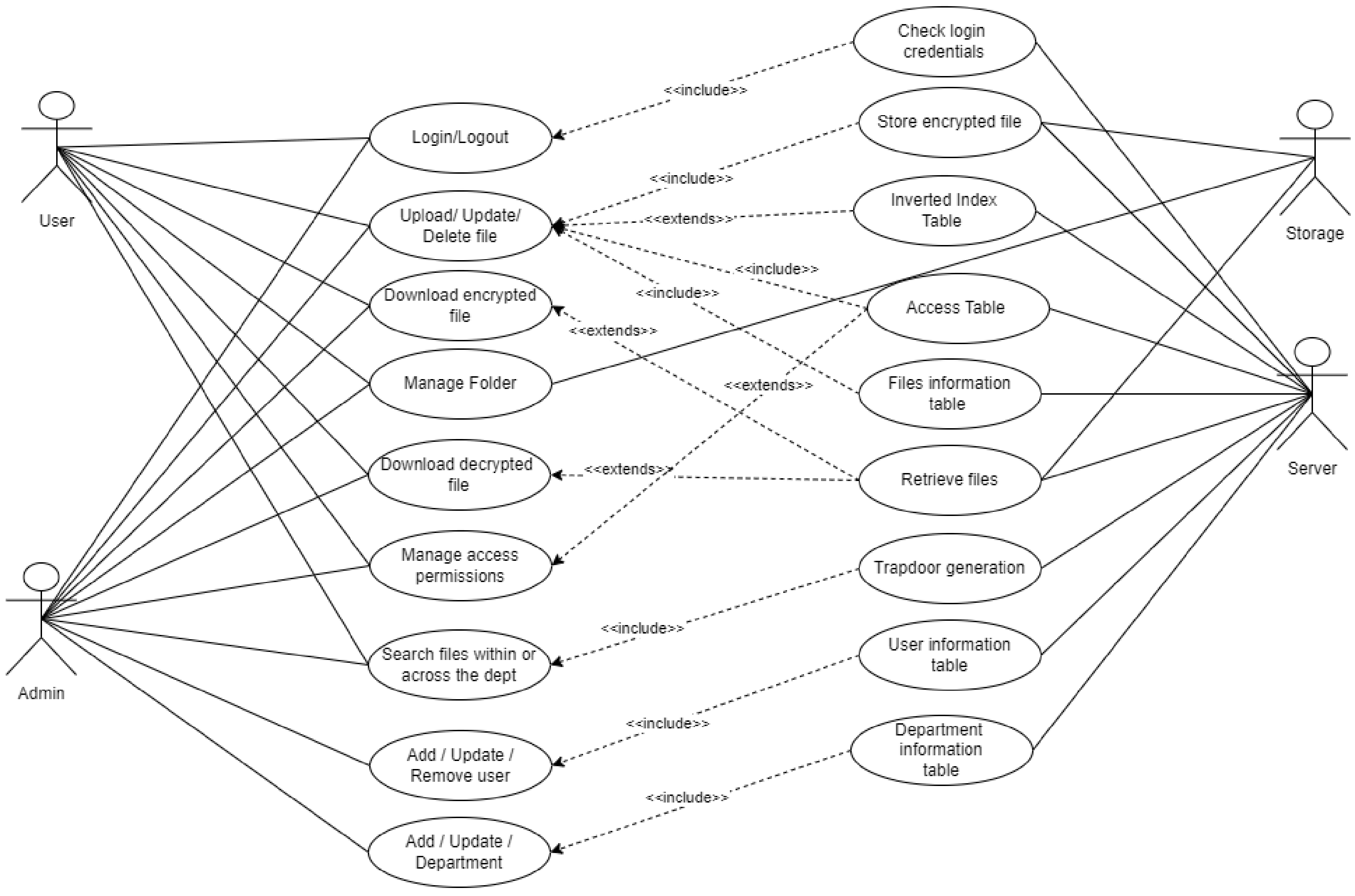


Figure 3: System Use Case Diagram

3.6.2. Activity Diagram

AirVault's activity diagram provides a visual roadmap for a secure and streamlined file management experience. It outlines a user journey, starting with a login that leads to a centralized dashboard post-authentication. Key functionalities highlighted include secure file searches, encrypted file uploads and downloads, user-friendly account management, and administrative functions for authorized users. Aligned with AirVault's commitment, the diagram emphasizes secure file management, streamlined user experience, and enhanced user-friendliness, ensuring the system meets the highest standards of security, efficiency, and user satisfaction.

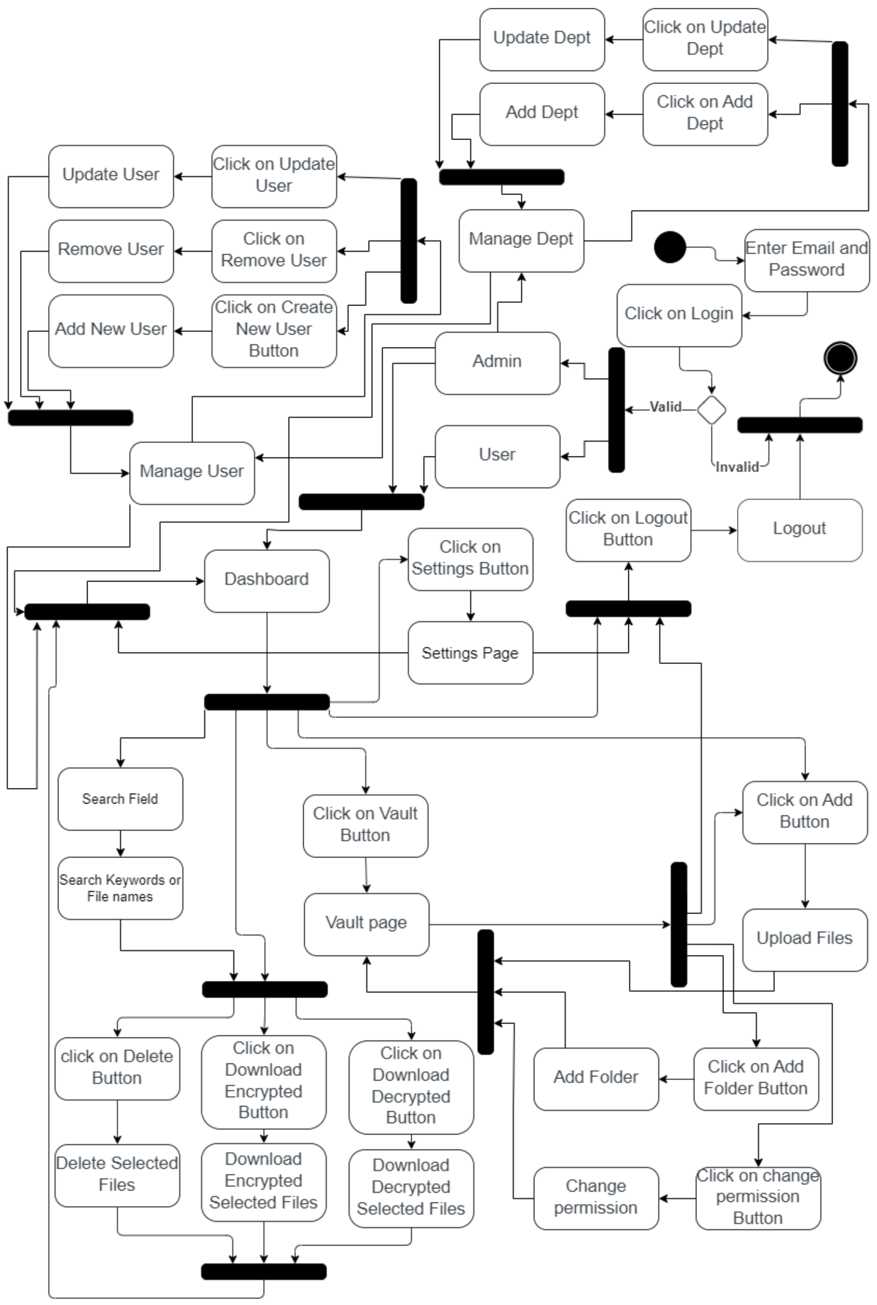


Figure 4: System Activity Diagram

3.6.3. Data Description

AirVault uses an inverted index table for search operations. It is dynamically generated and updated based on content of uploaded files. The database holds comprehensive information, including user and admin details, file metadata, inverted index tables, access tables, and a server information table.

Datasets

As we require extraction of keywords from textual files so Oxford Dictionary is taken as reference (129038 words). For testing purposes, publicly available datasets will be used to determine the accuracy and efficiency of the system.

Data Dictionary

User Table

The User Table stores essential user information, including their unique ID, name, email, password, server details, admin flag, and creation date**.**

Field Name	Type	Required	Description
id	INT	yes	Unique id of added user
name	VARCHAR	yes	Name of added user
email	VARCHAR	yes	Unique email of added user
password	VARCHAR	yes	A strong password for added user
is_admin	BOOL	yes	If the user is admin or not
created_at	DATE	yes	The date user is added in the system
updated_at	DATE	yes	The date user is updated in the system
lock_until	DATE	no	Time till when user is not allowed to log in to the system
public_key	STRING	yes	Used for encrypting of data
access_token	STRING	yes	Used for usder authentication

refresh_token	STRING	yes	Used to refresh access token when it expires

Table 1: User Table

Server Table

The Server Table maintains a record of servers in the system, including their unique IDs, names, and IP addresses.

Field Name	**Type**	**Required**	**Description**
server_id	INT	yes	Unique server id of server
server_name	STRING	yes	Name of server added in the airvault system
server_ip	STRING	yes	IP addresses of all the servers added in the system
has_admin	BOOL	yes	Tells if the server has admin or not
head_dept	BOOL	yes	Check if the server is head dept or not
is_self	BOOL	yes	Tells if the server info is the one running or not
created_at	DATE	yes	The date server is added in the system
updated_at	DATE	yes	The date server is updated in the system

Table 2: Server Table

Inverted Index Table

The Inverted Index Table maps keywords to the documents containing them, enabling efficient keyword-based searches. All keywords are generated using NLTK (natural language processing library) which extracts the keywords, these keywords are then encrypted using AES-256 and are then stored in database.

Field Name	Type	Required	Description
id	INT	yes	Unique id of each index
keyword	VARCHAR	yes	Keyword extracted from the file stored after encryption in database

Table 3: Inverted Index Table

File Information Table

The File Information Table tracks essential file details, including unique IDs, names, creators, creation dates, and last modification time.

Field Name	Type	Required	Description
id	INT	yes	Unique id of each file
document_name	VARCHAR	yes	Keyword extracted from the file
created_at	DATE	yes	The date file is added in the system
updated_at	DATE	yes	The date file is updated in the system
size	INT	yes	The size of added file

Table 4: File Information Table

Folders Table

The Folders Table specifies and maintains the folder structure and keeps the record of permissions as well

Field Name	Type	Required	Description
id	INT	yes	Unique id of folder
folder_name	VARCHAR	yes	Name of created folder
folder_path	STRING	yes	Path of storage where folder is created
created_at	DATE	yes	Date where folder is created in the system
updated_at	DATE	yes	Date when folder is updated in the system
owner_id	INT	yes	Id of owner of folder
editors	TEXT	yes	Ids of all user that are added as editors
viewers	TEXT	yes	Ids of all user that are added as viewers

Table 5: Folder Table

3.6.4. Entity Relation Diagram

AirVault's blueprint for secure file storage is laid out in its ERD. The server acts as the system's heart, managing users, files, and access control. Users interact with files based on their permissions, defined in access lists. Distinct user roles (admins and regular users) have different privileges, and files are encrypted for enhanced security. Search functionality is designed to protect file content, and the structure prioritizes user experience and granular access control.

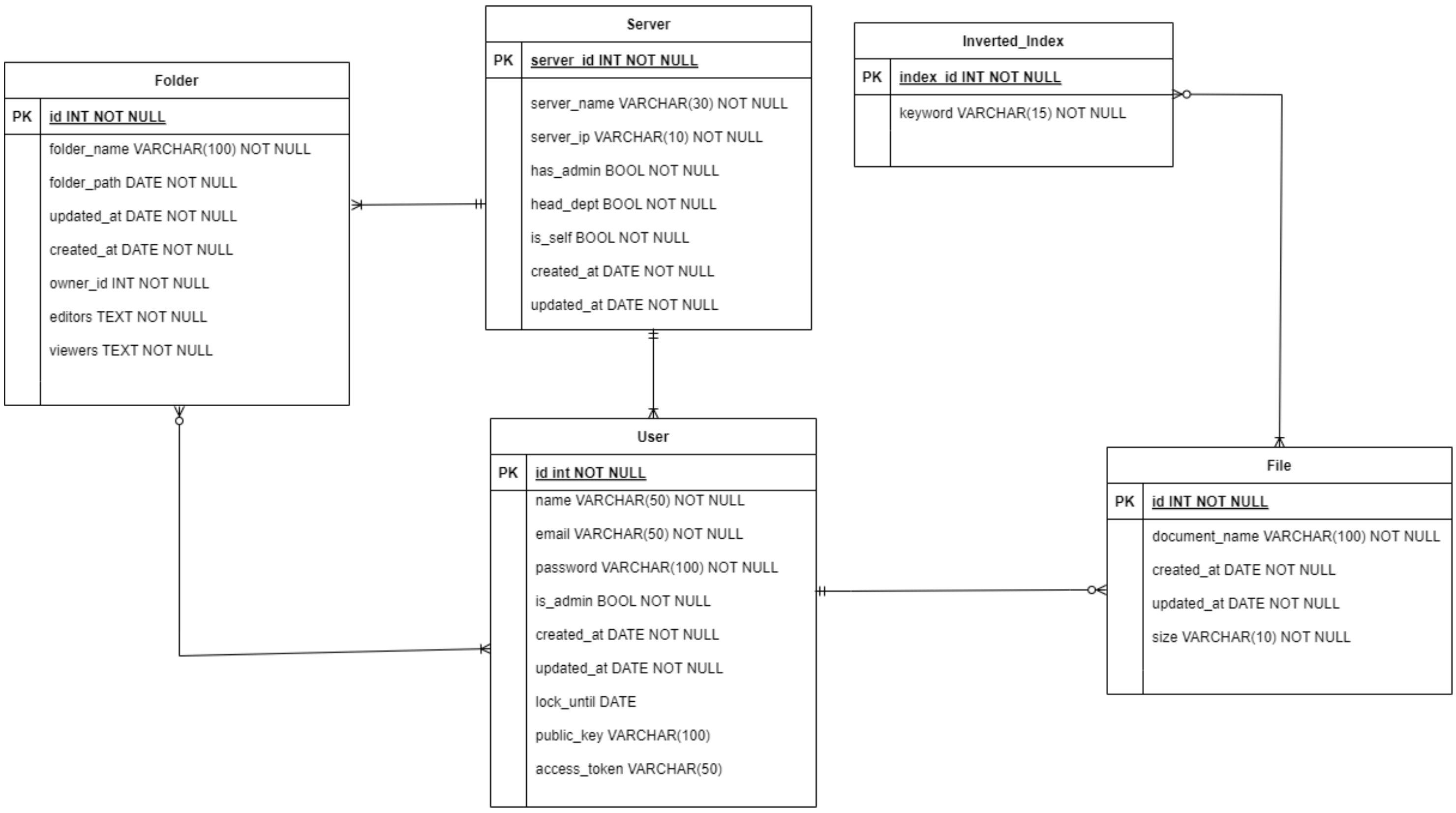


Figure 5: System Erd Diagram

3.6.5. Hardware

The diagram below shows the hardware used in AirVault. We have two Raspberry Pi which are acting as servers connected with the router (which is not connected with the internet), forming an air-gapped network. Also, storage is attached to one of the server to store encrypted files.

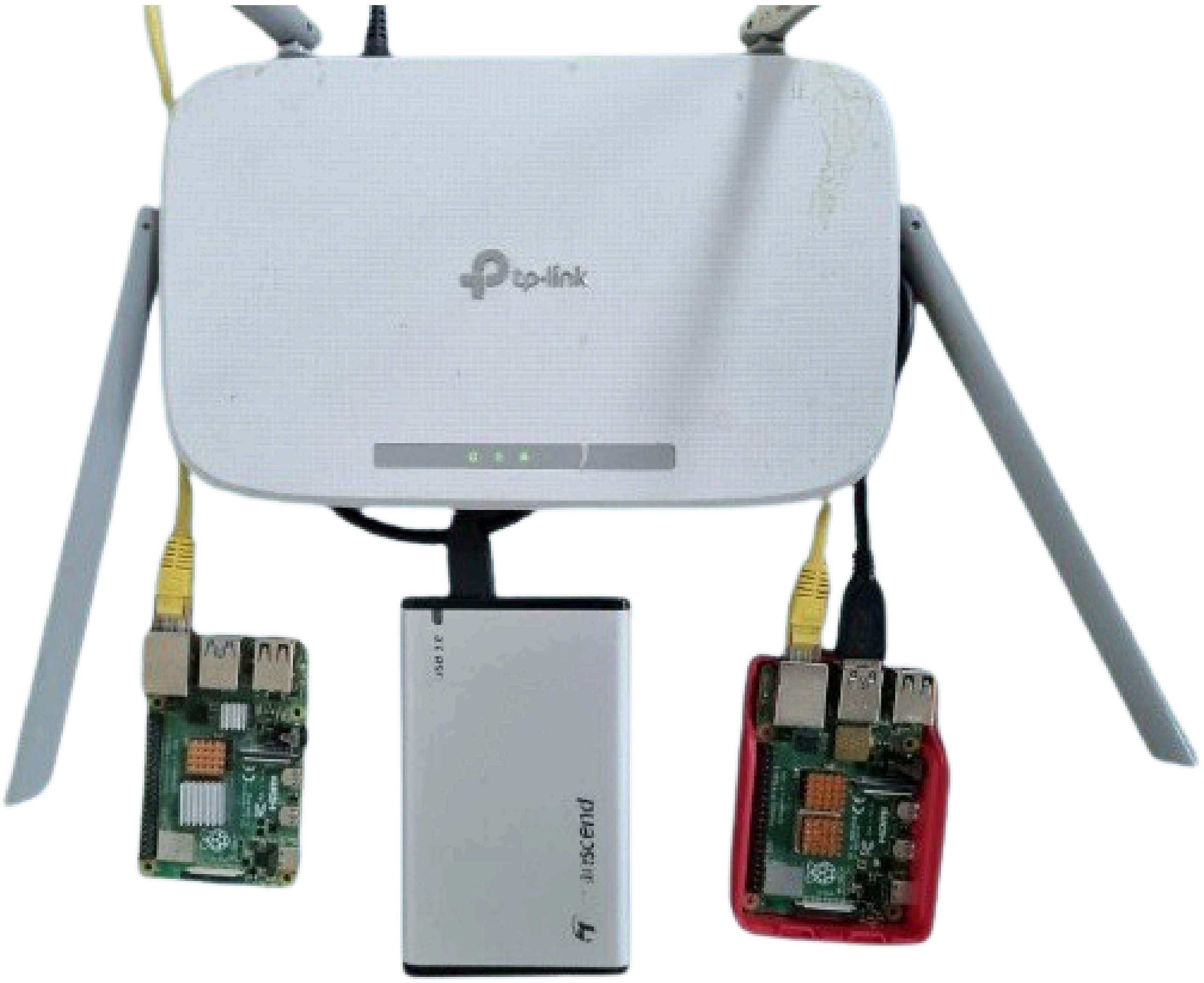


Figure 6: System Hardware

3.7. User Interface

3.7.1. Web Application

3.7.1.1. Login

The users can log in to their account by entering the valid credentials provided to them by the admin. The user will get five chances to enter the correct credentials.

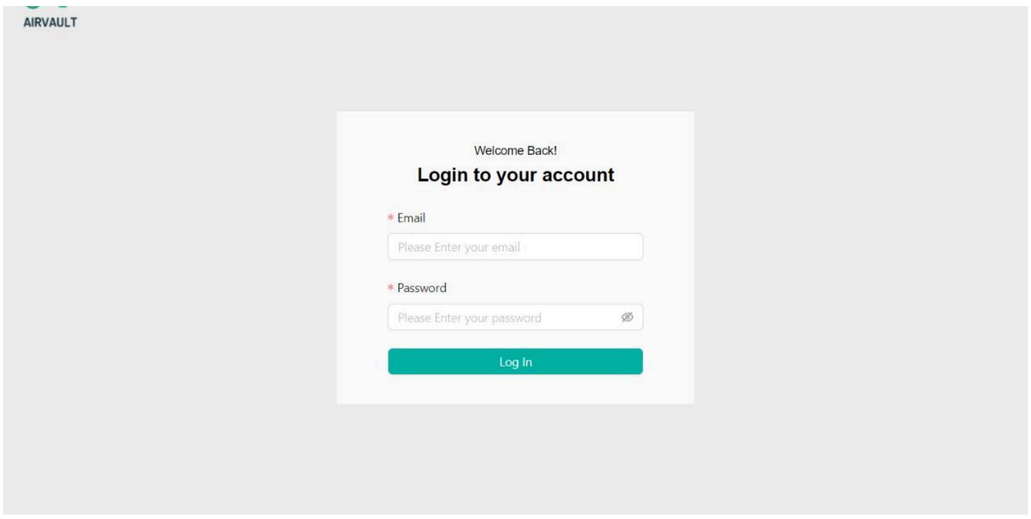
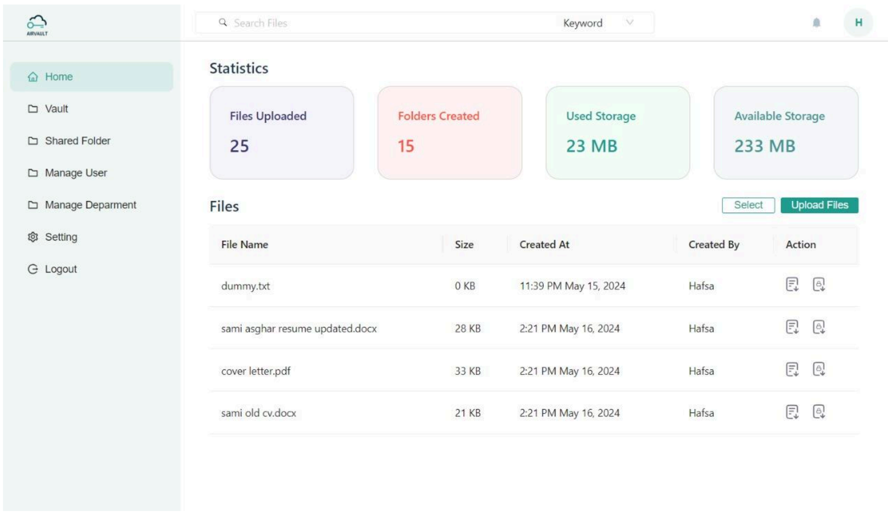


Figure 7: Website Login page

3.7.1.2. Dashboard

Users can view statistics, folders, and all the recently uploaded files. Users can either upload files or select them. Additionally, they have option to navigate to settings and log out from their account.



Statistics

Files Uploaded	Folders Created	Used Storage	Available Storage
25	15	23 MB	233 MB

Files

File Name	Size	Created At	Created By	Action
dummy.txt	0 KB	11:39 PM May 15, 2024	Hafsa	
sami asghar resume updated.docx	28 KB	2:21 PM May 16, 2024	Hafsa	
cover letter.pdf	33 KB	2:21 PM May 16, 2024	Hafsa	
sami old cv.docx	21 KB	2:21 PM May 16, 2024	Hafsa	

Figure 8: Website Dashboard Page

3.7.1.3. Upload File

The User can upload file from the drag and drop in this popup.

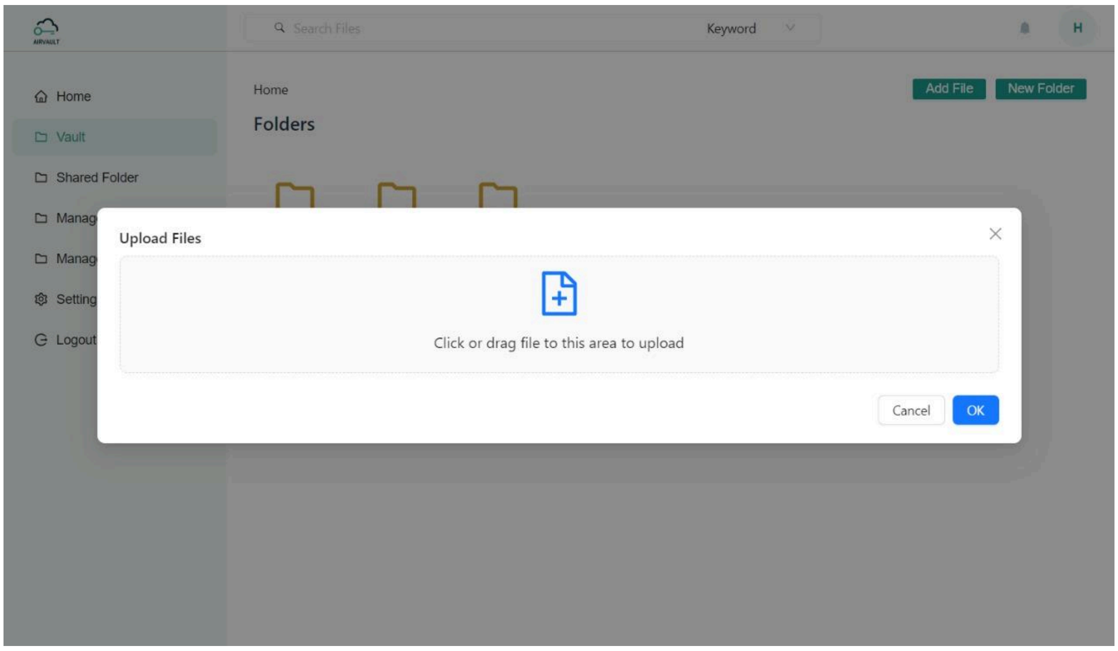


Figure 9: Website Upload Files

3.7.1.4. Vault

The user can create new folders, upload new files and Change permissions in this Vault page.

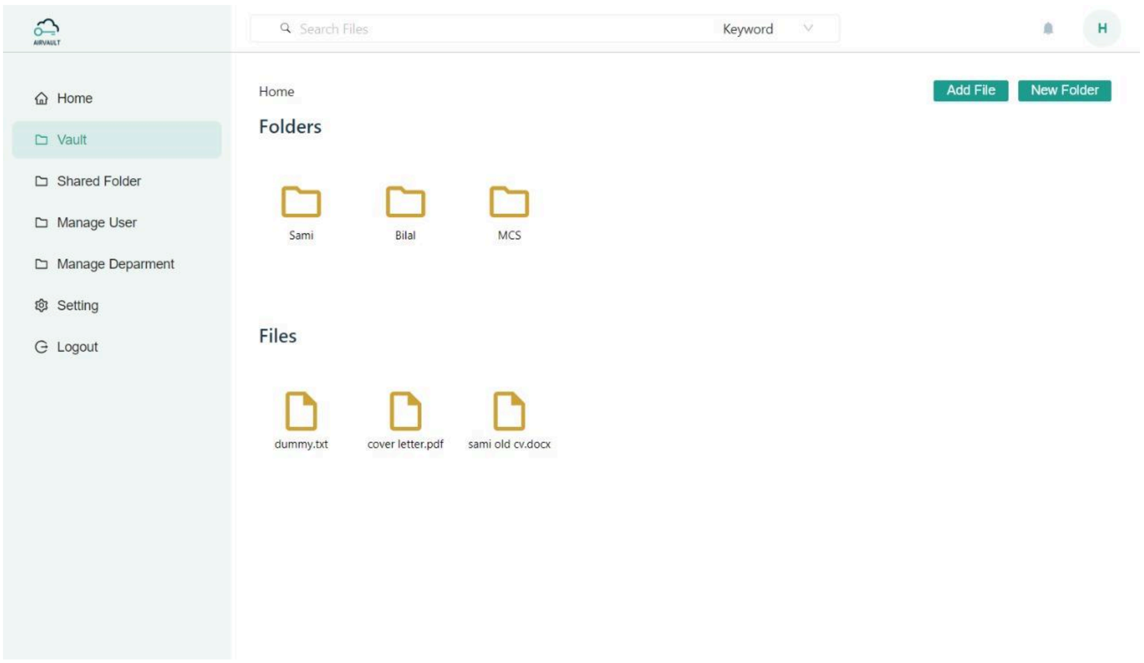


Figure 10: Website Vault Page

3.7.1.5. Search File

The user can search the file from the file name or the keywords in the file.

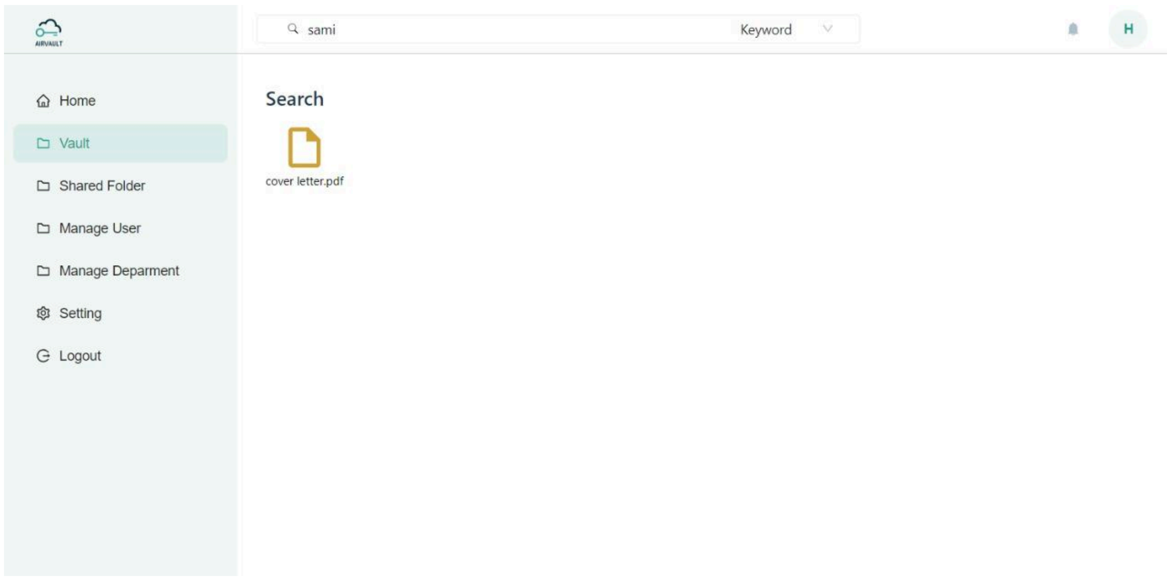
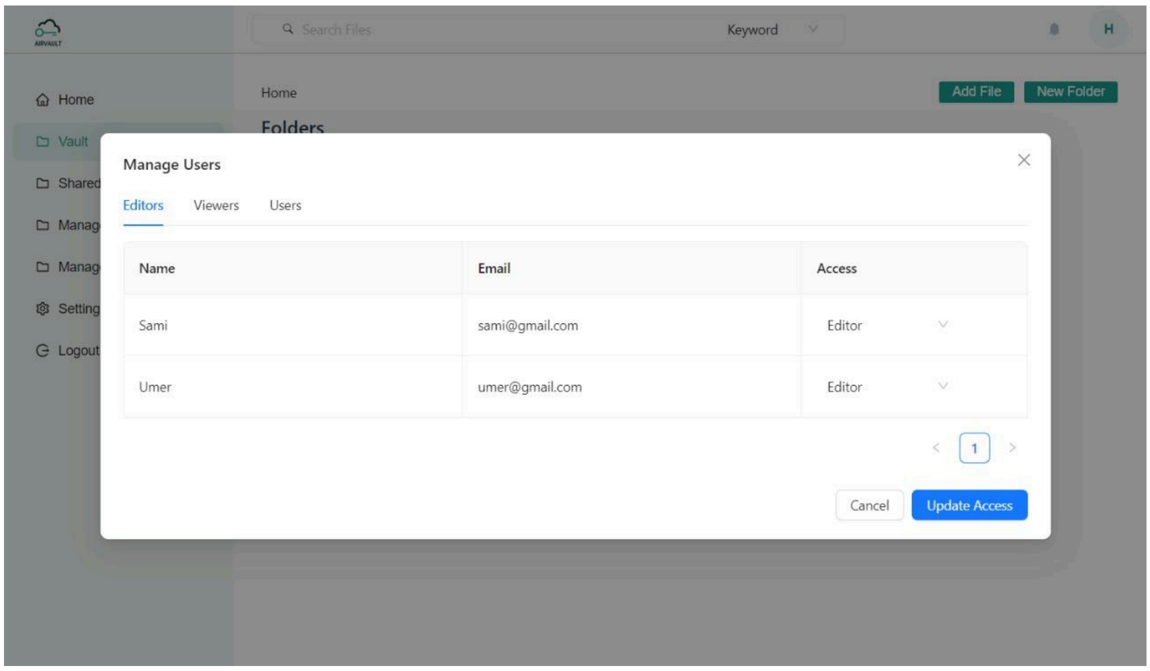


Figure 11: Website Search File

3.7.1.6. Change Permission

Users can change the permission in this popup. Here they can make the user either editor or viewer.

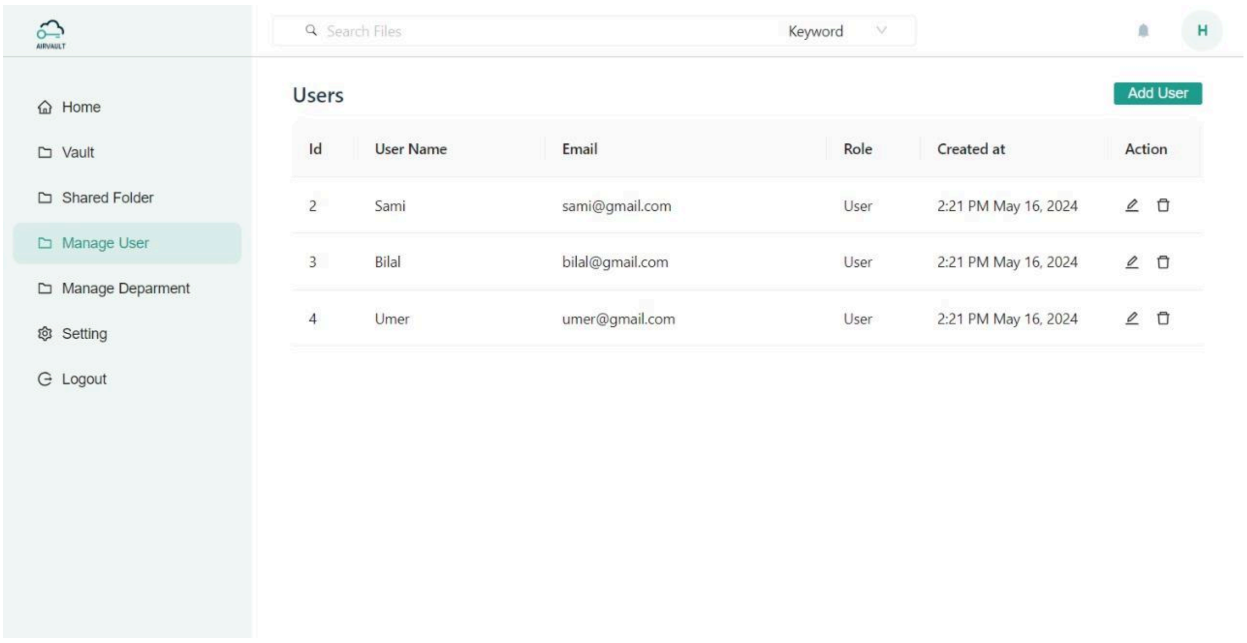


Name	Email	Access
Sami	sami@gmail.com	Editor
Umer	umer@gmail.com	Editor

Figure 12: Website Change File

3.7.1.7. Manage User

The Admin can add new user, edit user or delete user in this manage user page.



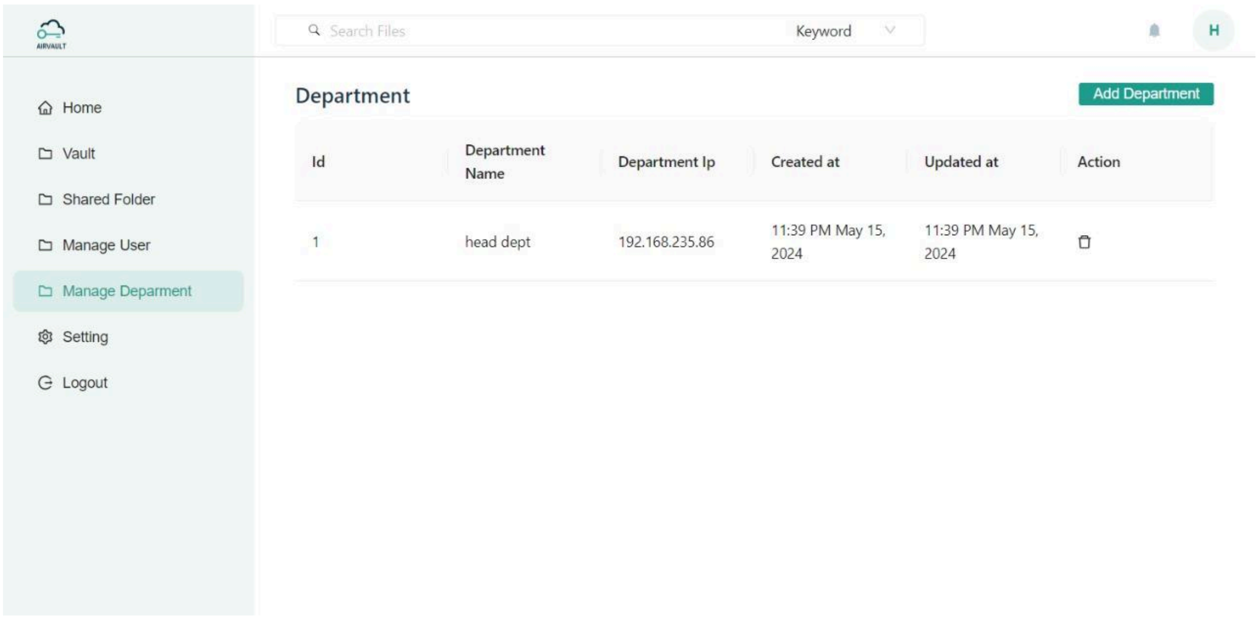
Users

Id	User Name	Email	Role	Created at	Action
2	Sami	sami@gmail.com	User	2:21 PM May 16, 2024	
3	Bilal	bilal@gmail.com	User	2:21 PM May 16, 2024	
4	Umer	umer@gmail.com	User	2:21 PM May 16, 2024	

Figure 13: Website Manage User

3.7.1.8. Manage Department

The Admin can add new Department or edit Department in this manage user page.



Department

Id	Department Name	Department Ip	Created at	Updated at	Action
1	head dept	192.168.235.86	11:39 PM May 15, 2024	11:39 PM May 15, 2024	

Figure 14: Website Manage Department

3.7.2. Mobile Application

3.7.2.1. Login

The users can log in to their account by entering the valid credentials provided to them by the admin. The user will get five chances to enter the correct credentials.

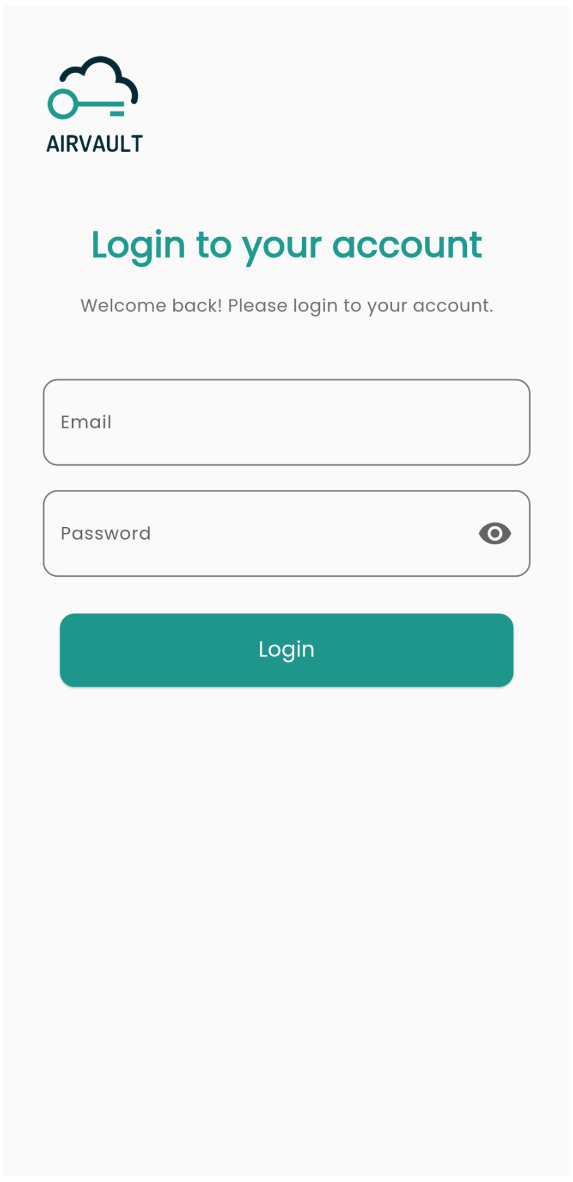
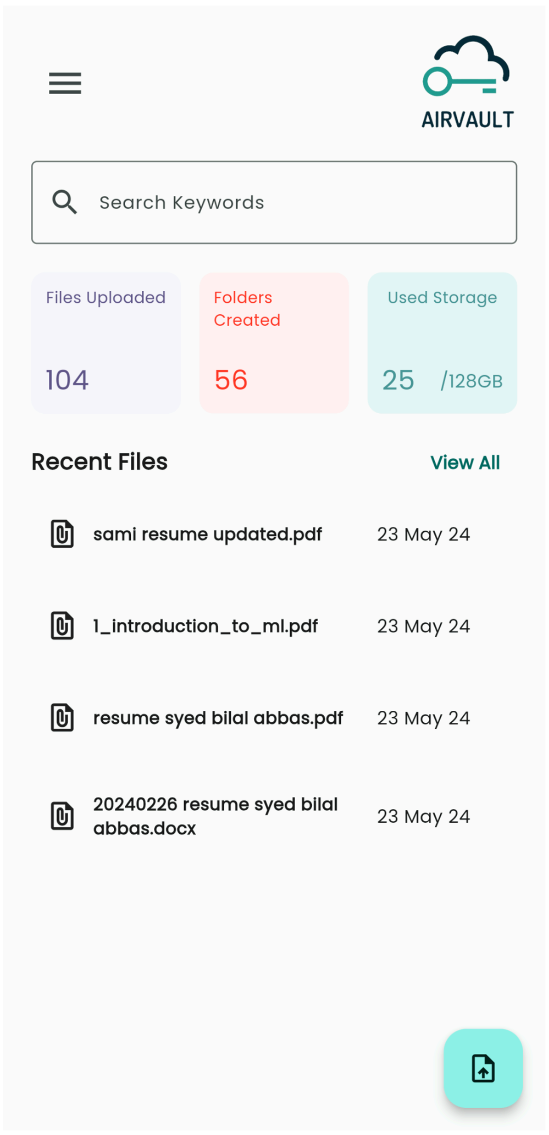


Figure 15: Mobile Login Page

3.7.2.2. Dashboard

The user can view the statistics, folders, and all the recently uploaded files. Users can either upload files or select them. Additionally, they have the option to navigate to settings and log out from their account.



sami resume updated.pdf	23 May 24
1_introduction_to_ml.pdf	23 May 24
resume syed bilal abbas.pdf	23 May 24
20240226 resume syed bilal abbas.docx	23 May 24

Figure 16: Mobile Dashboard Page

3.7.2.3. Upload File

The User can upload file from file button at the bottom.

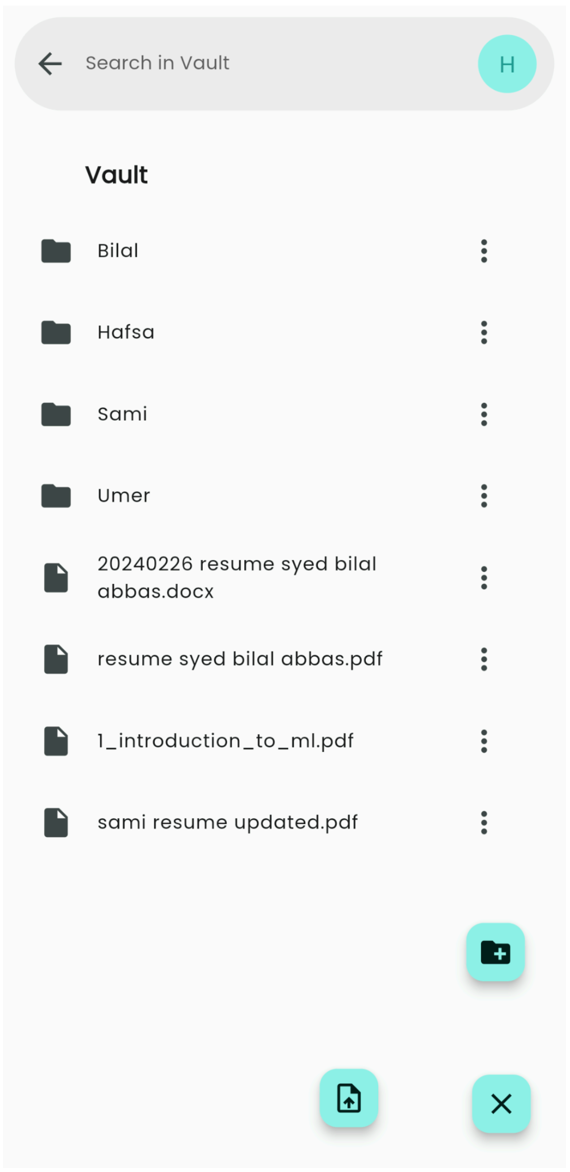


Figure 17: Mobile Upload Button

3.7.2.4. Vault

The user can create new folders, upload new files and Change permissions in this Vault page.

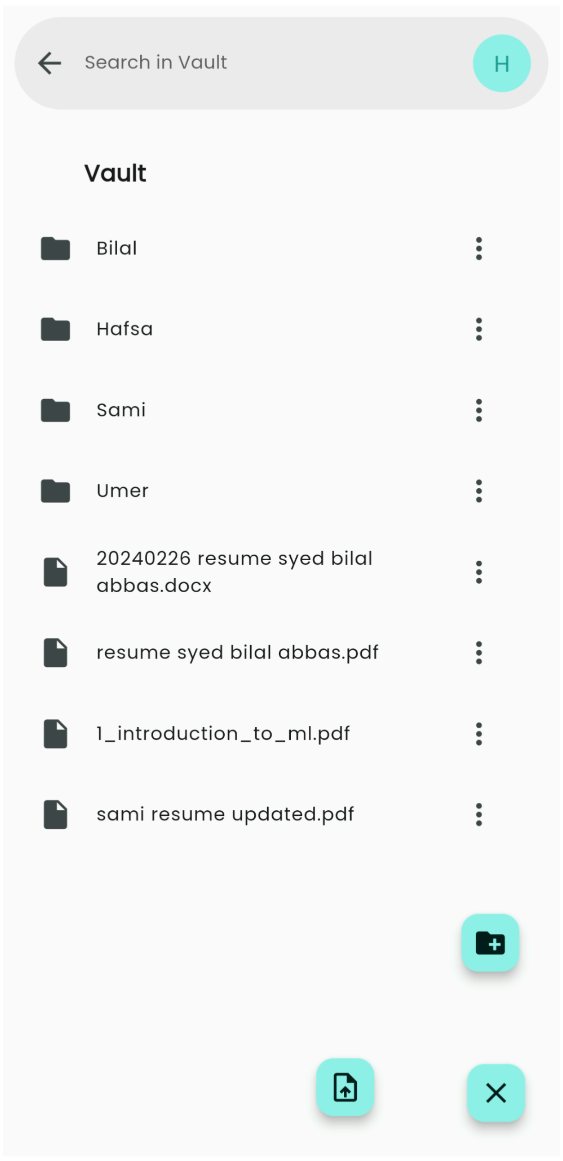


Figure 18: Mobile Vault Page

3.7.2.5. Search File

The user can search the file from the file name or the keywords in the file.

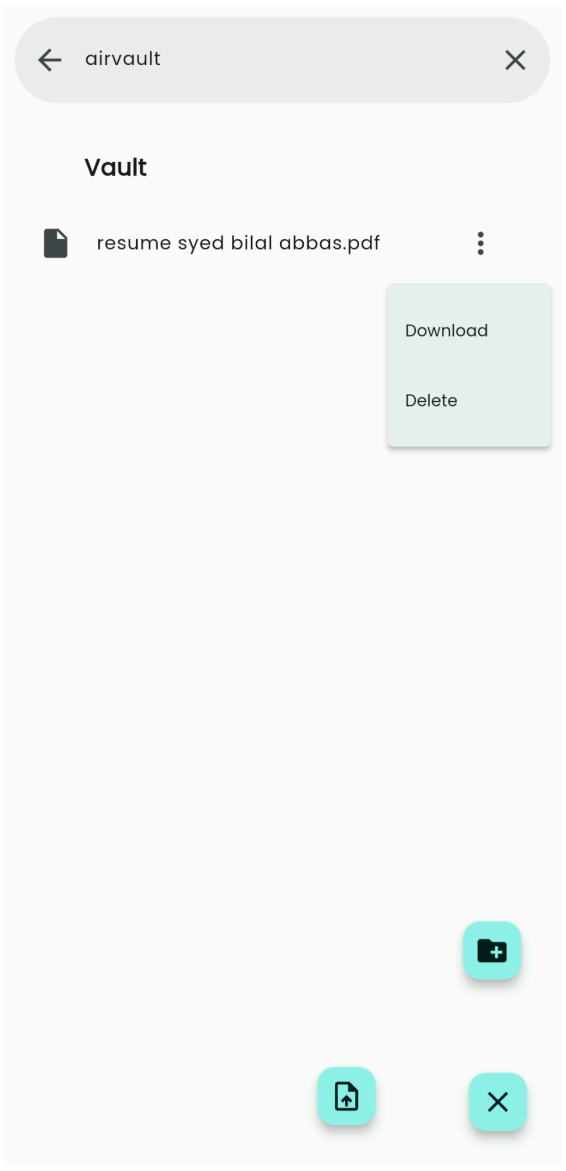


Figure 19: Mobile Search File

3.7.2.6. Change Permission

The user can change the permission in this popup. Here they can make the user either editor or viewer.

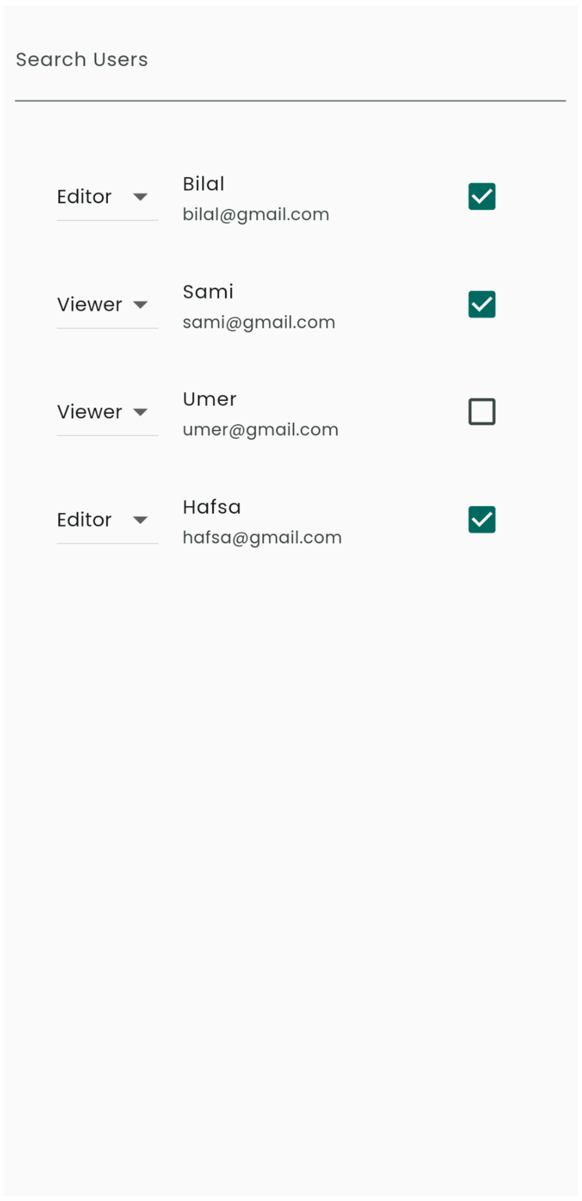
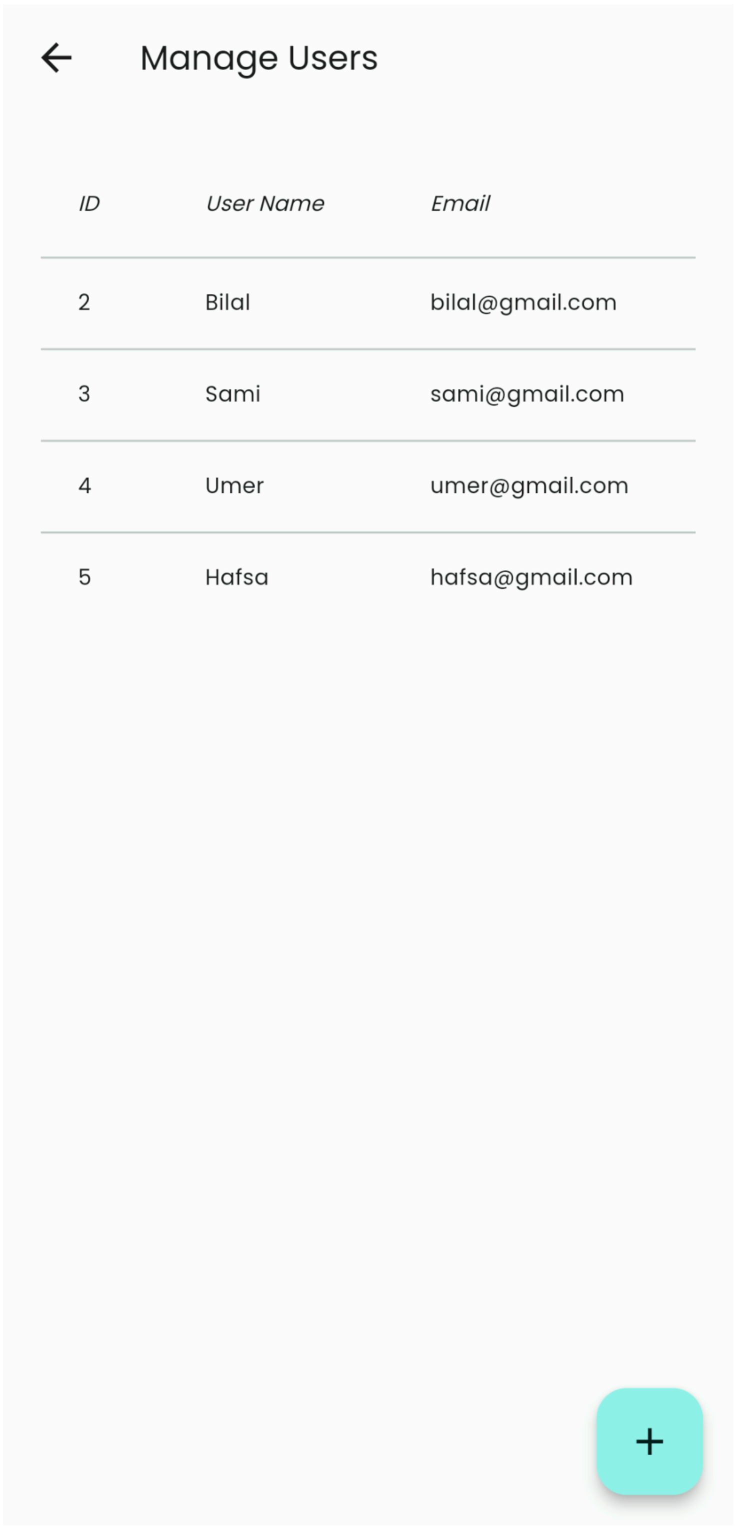


Figure 20: Mobile Change Permission

3.7.2.7. Manage User

The Admin can add new user, edit user or delete user in this manage user page.

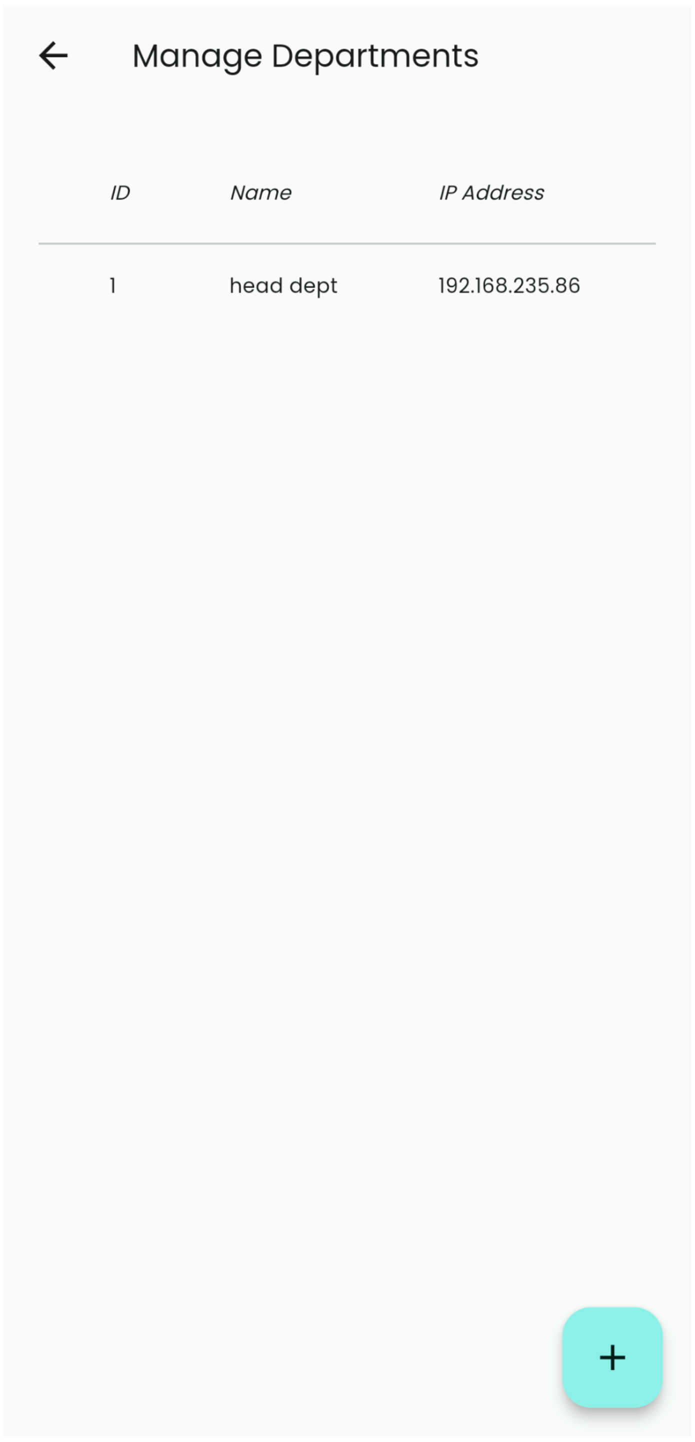


ID	User Name	Email
2	Bilal	bilal@gmail.com
3	Sami	sami@gmail.com
4	Umer	umer@gmail.com
5	Hafsa	hafsa@gmail.com

Figure 21: Mobile Manage User

3.7.2.8. Manage Department

The Admin can add new Department or edit Department in this manage user page.



ID	Name	IP Address
1	head dept	192.168.235.86

Figure 22: Mobile Manage Dashboard

4. System Features

Every feature in this chapter is described in great detail, along with all the required diagrams for greater understanding and acceptance criteria.

4.1. Login

4.1.1. Use Case Diagram

This use case represents how the user interacts with the system while logging in, for the login server will always check the entered credentials if the credentials are correct only then the user will be logged into the system. This process ensures that only users with the correct credentials are granted access to the system.

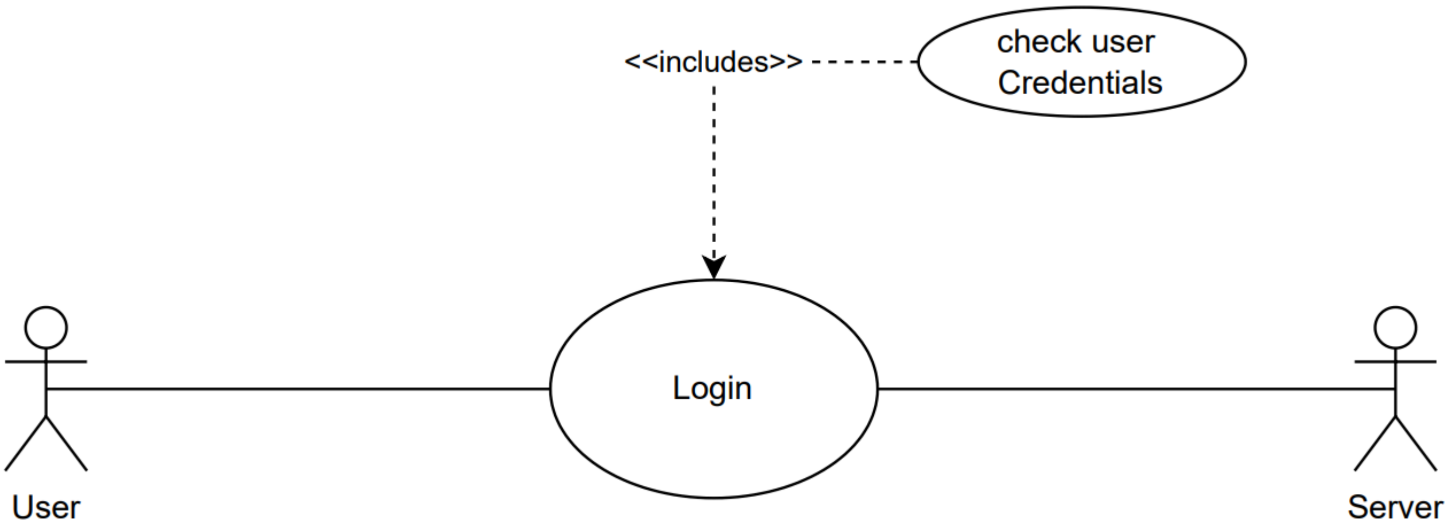


Figure 23: Use Case Diagram Login

This use case table outlines the interactions between the user and the system for the Login functionality.

Actor	User
Alternative Path	System responds with local feedback, If users submit the form with empty fields in Step 2.

Basic Path	1. Users must enter the email and password and submit a login form. 2. System verify user’s credential in database. 3. System redirect to Dashboard, If credentials are correct.
Others	User credentials include email address and password.
Post Condition	User will redirect to Dashboard
Precondition	User must have account on AirVault
Trigger	User visits the Login Page
Use Case Name	Login

Table 6: Use Case Table Login

4.1.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the user logs in to the system. This diagram illustrates each event from mobile app to server.

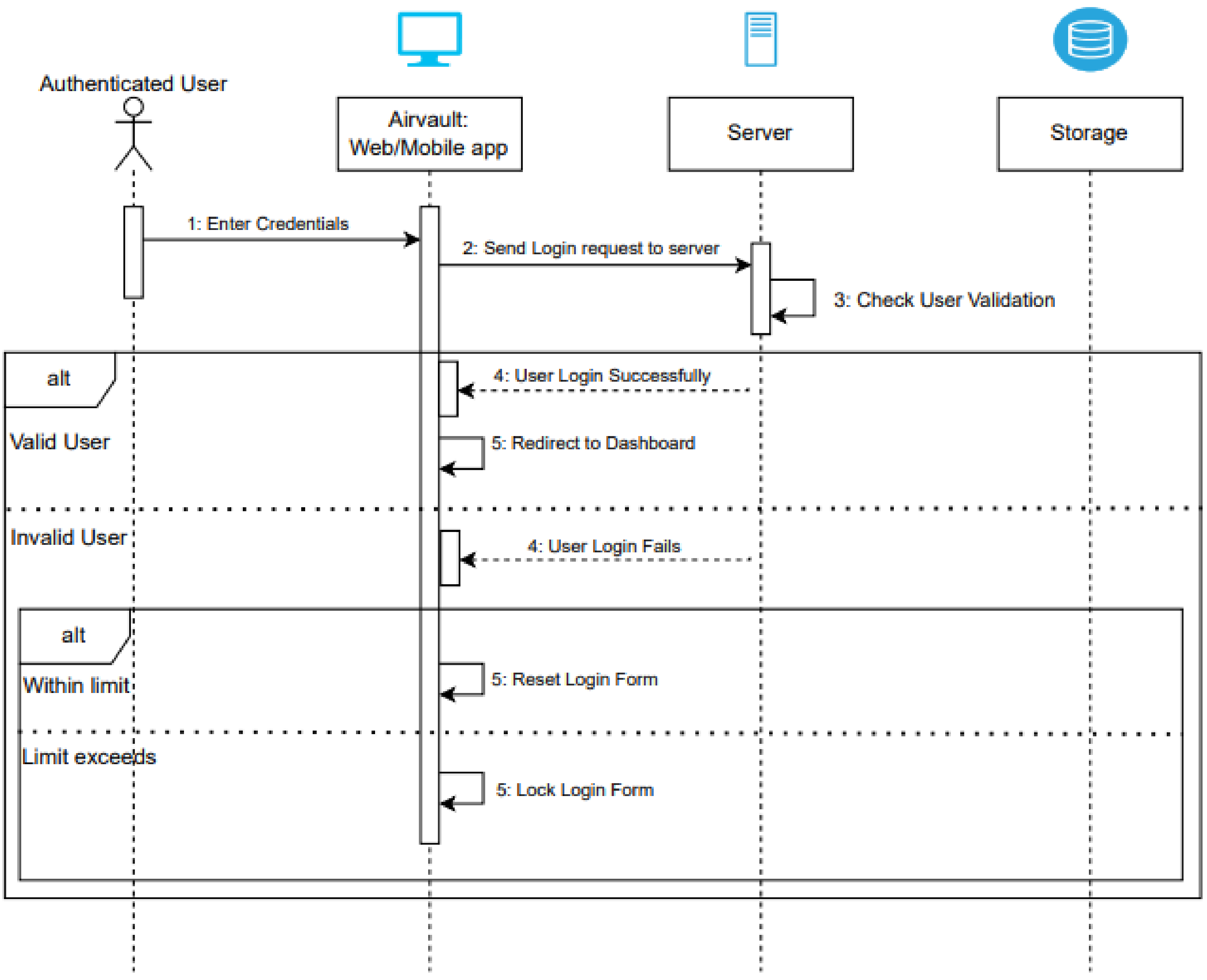


Figure 24: Sequence Diagram Login

4.1.3. Activity Diagram

This activity diagram illustrates the series of activities which will occur at each level of the system as the user logs in.

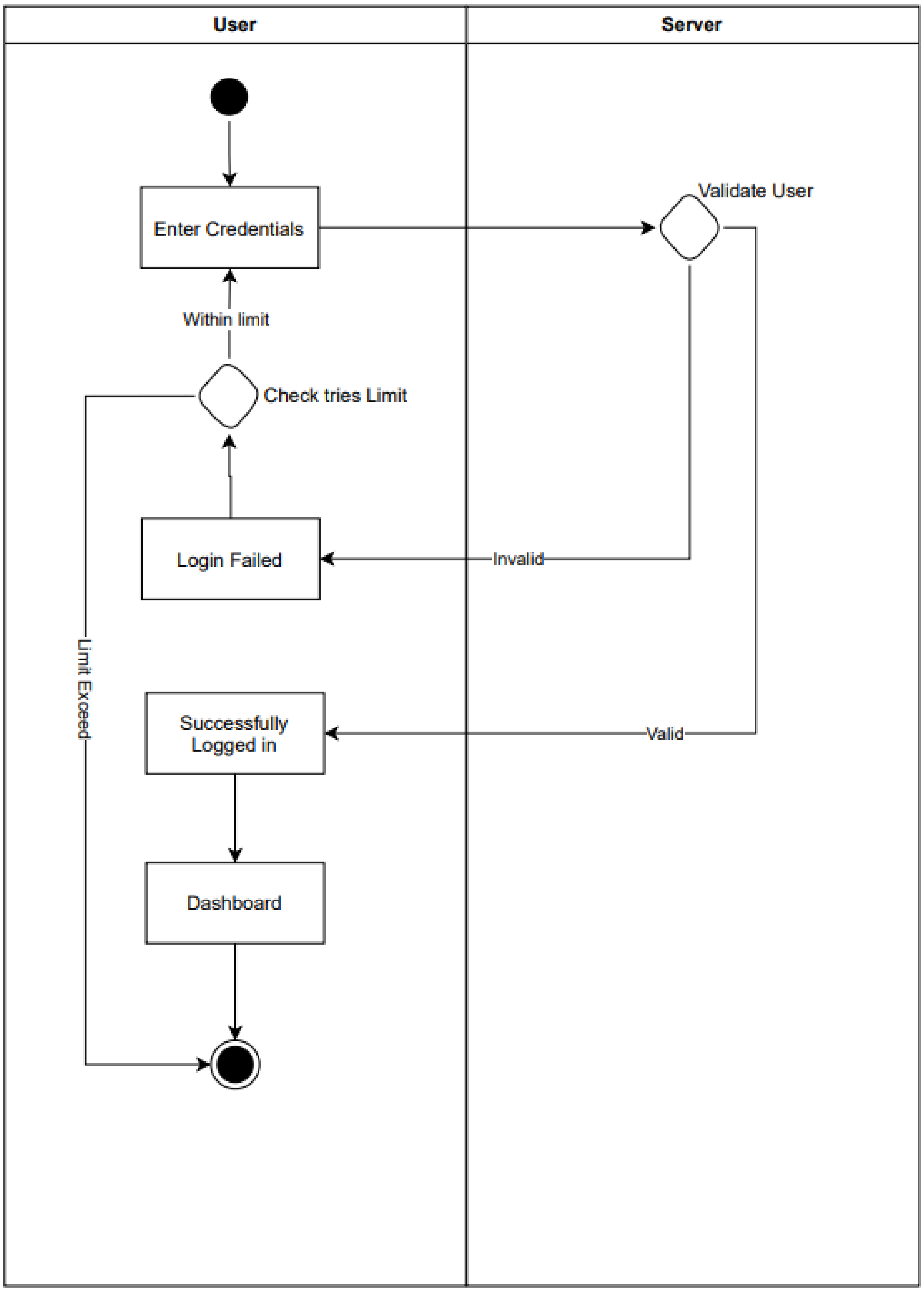


Figure 25: Activity Diagram Login

4.1.4. Block Diagram

This block diagram illustrates the series of events which will occur as the user logs in the system.

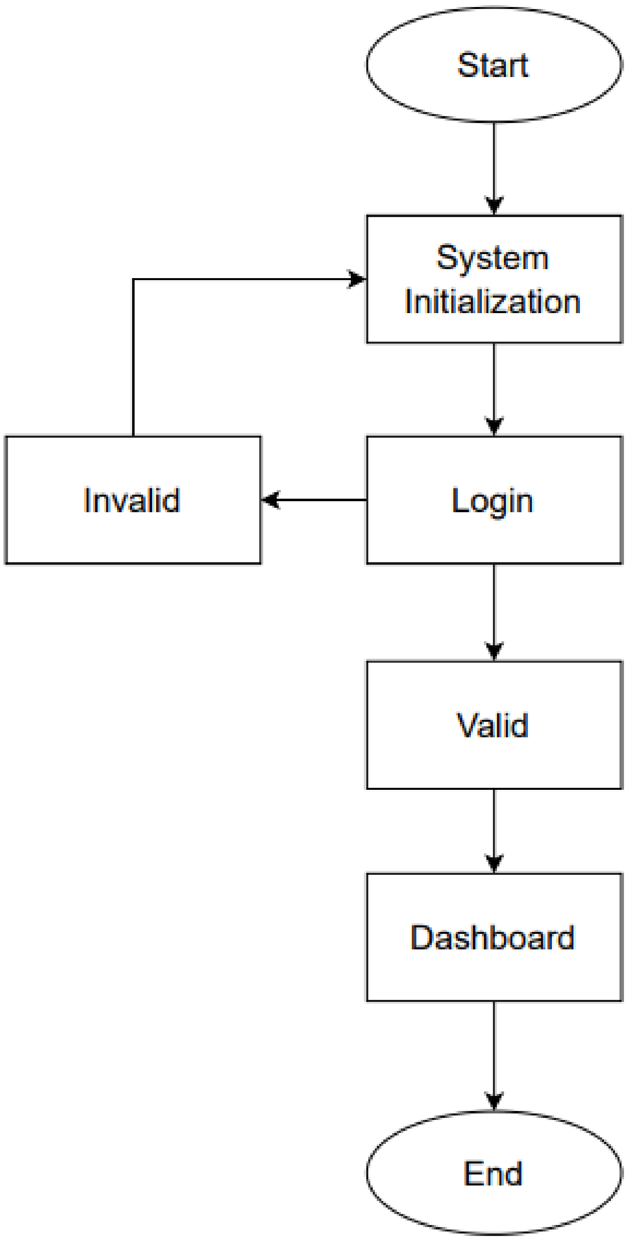


Figure 26: Block Diagram Login

4.2. Upload File

4.2.1. Use Case Diagram

This use case represents how the user interacts with the system while uploading a file, for uploading of file(s) users have to specify permissions for each file. File(s) once uploaded will be sent to the server for further processing.

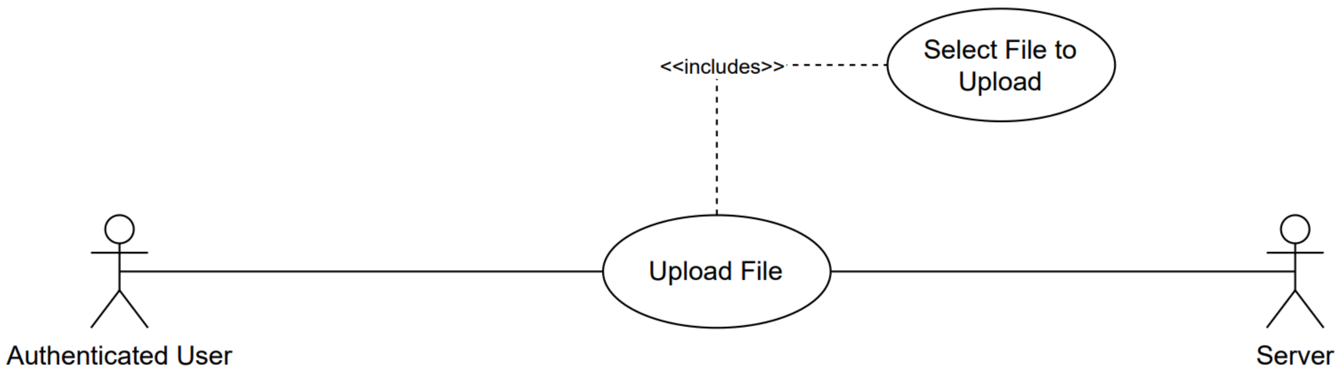


Figure 27: Use Case Diagram Upload File

This use case table outlines interactions between user and system for the Upload File functionality.

Use Case Name	Upload File
Actor	User
Trigger	User clicks on Upload file button
Precondition	User must be logged into their account
Basic Path	1. User selects the file to upload on server 2. System encrypt that file and store it on storage 3. System will respond to the user with a snackbar on file uploads.
Alternative Path	System responds with local feedback, If users selects file which AirVault is not supported
Post Condition	Users can see that file on recent files.

Others	System not store those files, which are not supported by AirVault

Table 7: Use Case Table Upload File

4.2.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the user uploads a file. All the events of file uploading from client side till it is stored in the storage are illustrated in the figure below.

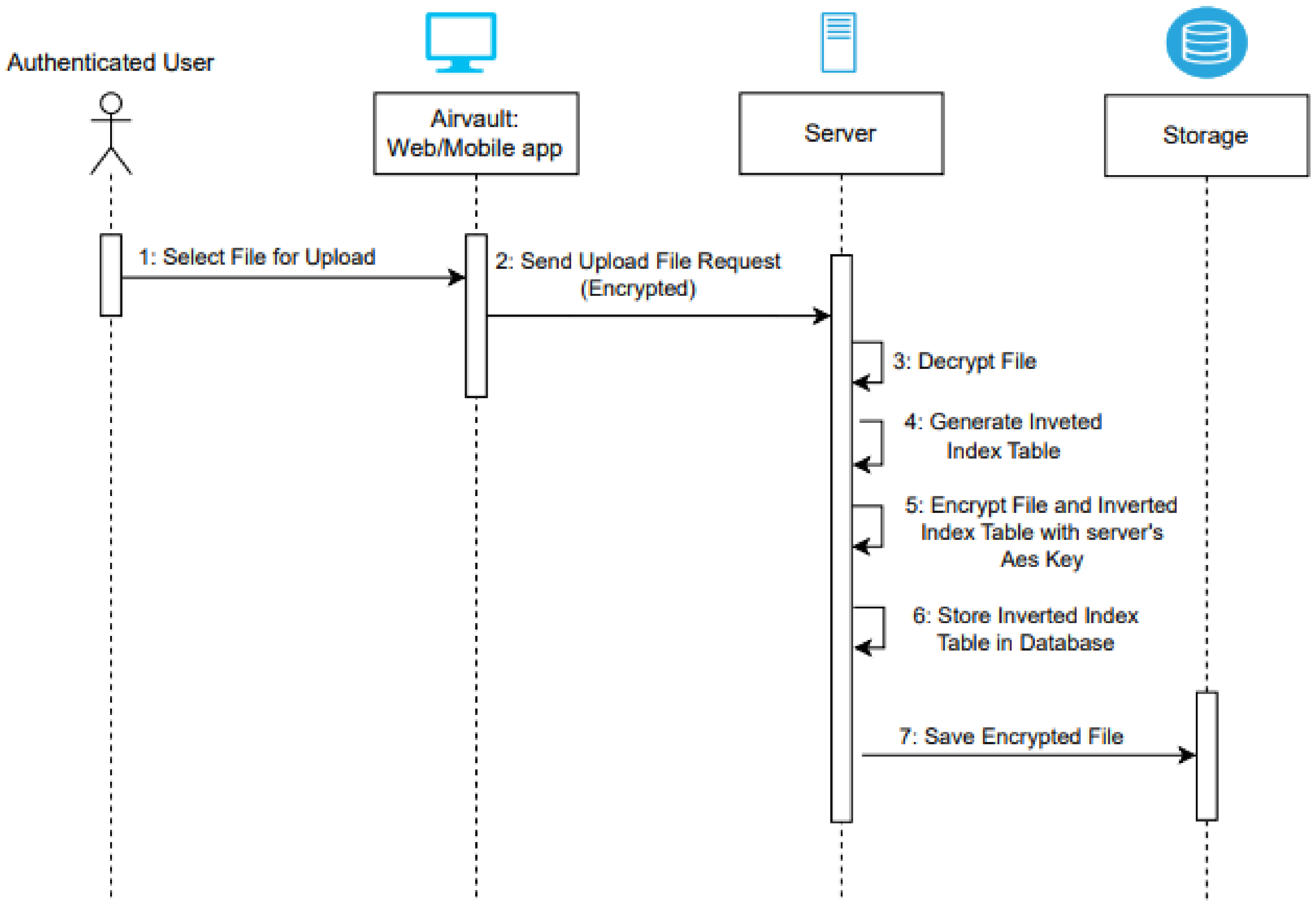

Authenticated User
Airvault: Web/Mobile app
Server
Storage
1: Select File for Upload
2: Send Upload File Request (Encrypted)
3: Decrypt File
4: Generate Inveted Index Table
5: Encrypt File and Inverted Index Table with server's Aes Key
6: Store Inverted Index Table in Database
7: Save Encrypted File

Figure 28: Sequence Diagram Upload File

4.2.3. Activity Diagram

This activity diagram illustrates the series of activities which will occur at each level of the system as the user uploads files. It shows detailed activities of file processing before encrypting and storing it in storage.

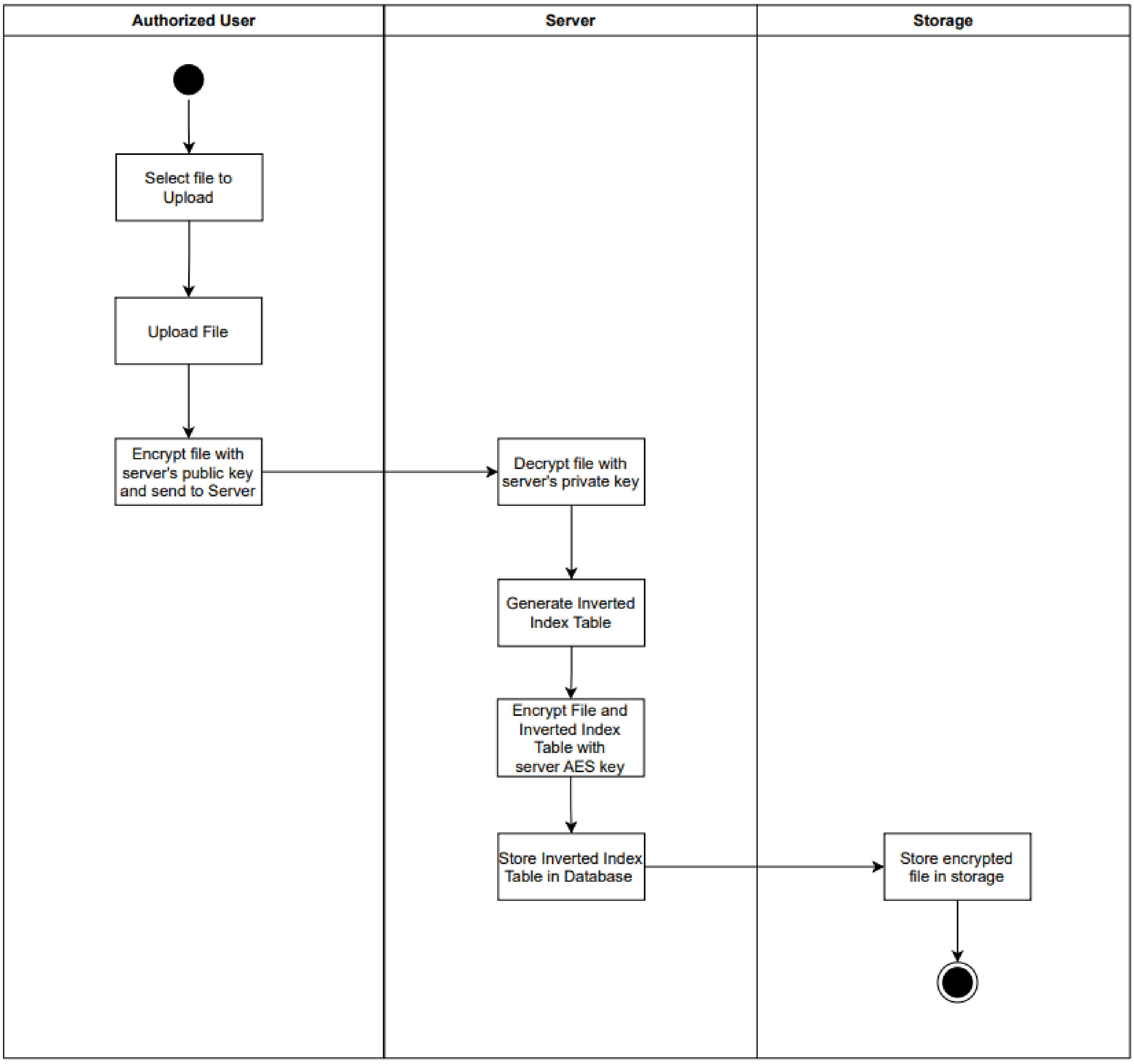

Authorized User
Server
Storage
Select file to Upload
Upload File
Encrypt file with server's public key and send to Server
Decrypt file with server's private key
Generate Inverted Index Table
Encrypt File and Inverted Index Table with server AES key
Store Inverted Index Table in Database
Store encrypted file in storage

Figure 29: Activity Diagram Upload File

4.2.4. Block Diagram

This block diagram illustrates the series of events which will occur as the user uploads file(s).

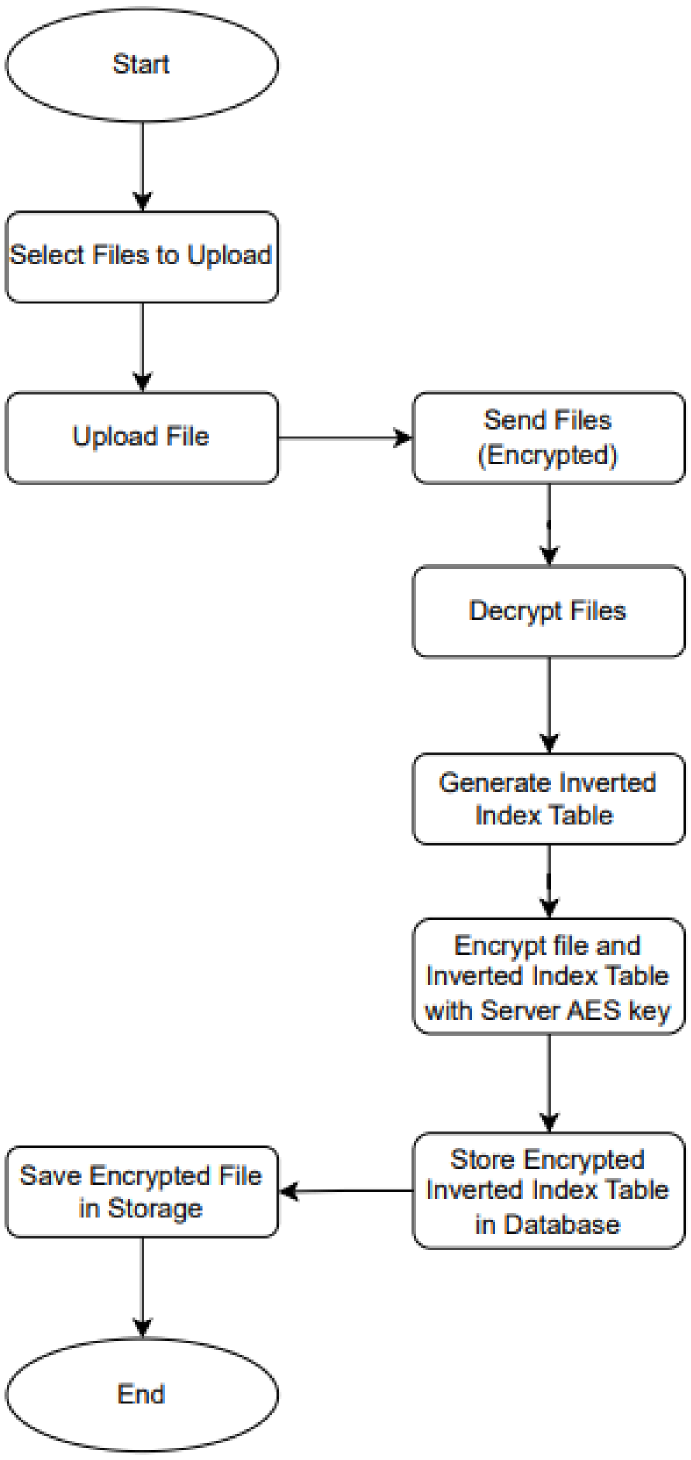

Start
Select Files to Upload
Upload File
Send Files (Encrypted)
Decrypt Files
Generate Inverted Index Table
Encrypt file and Inverted Index Table with Server AES key
Store Encrypted Inverted Index Table in Database
Save Encrypted File in Storage
End

Figure 30: Block Diagram Upload File

4.3. Update File

4.3.1. Use Case Diagram

This use case represents how the user interacts with the system while updating a file, for updating of file(s) users have to specify permissions for each file.

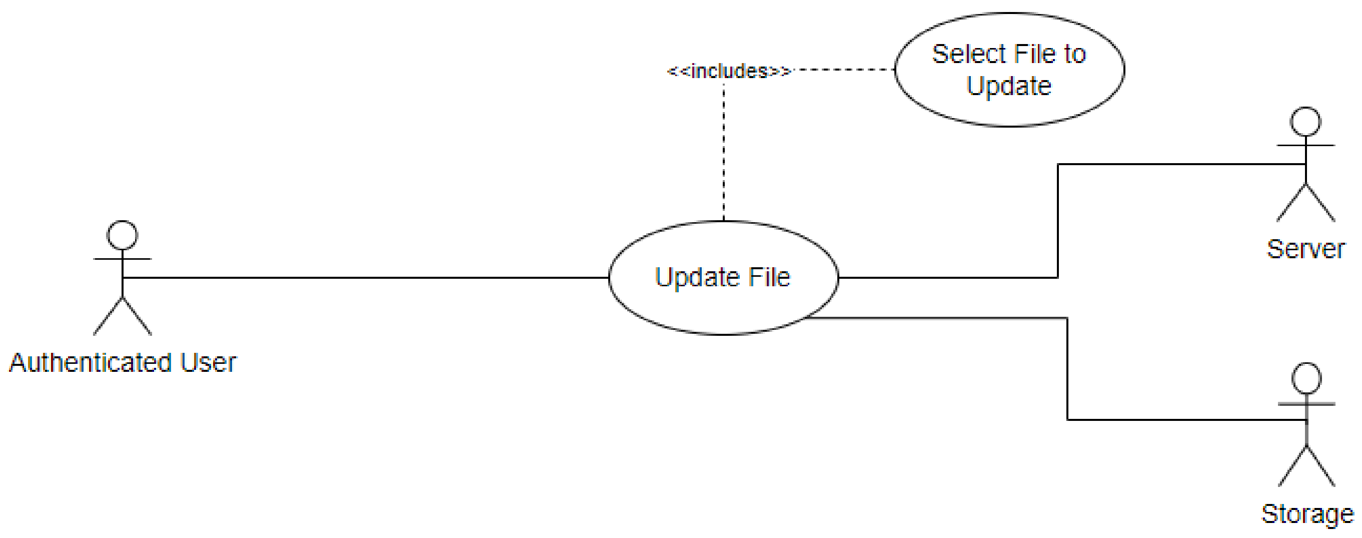


Figure 31: Use Case Diagram Update File

This use case table outlines interactions between the user and the system for the Update File functionality.

Use Case Name	Update File
Actor	User
Trigger	User clicks on Upload file button
Precondition	User must be logged into their account
Basic Path	1. User selects the file to update on server 2. System encrypt that file and update it on storage 3. System respond to user with snackbar on file uploads.
Alternative Path	System responds with local feedback, If users selects file which AirVault is not supported
Post Condition	Users can see that file on vault page.

Others	System not store those files, which are not supported by AirVault

Table 8: Use Case Table Update File

4.3.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the user update a file. All the events of file updating from client side till it is stored in the storage are illustrated in the figure below.

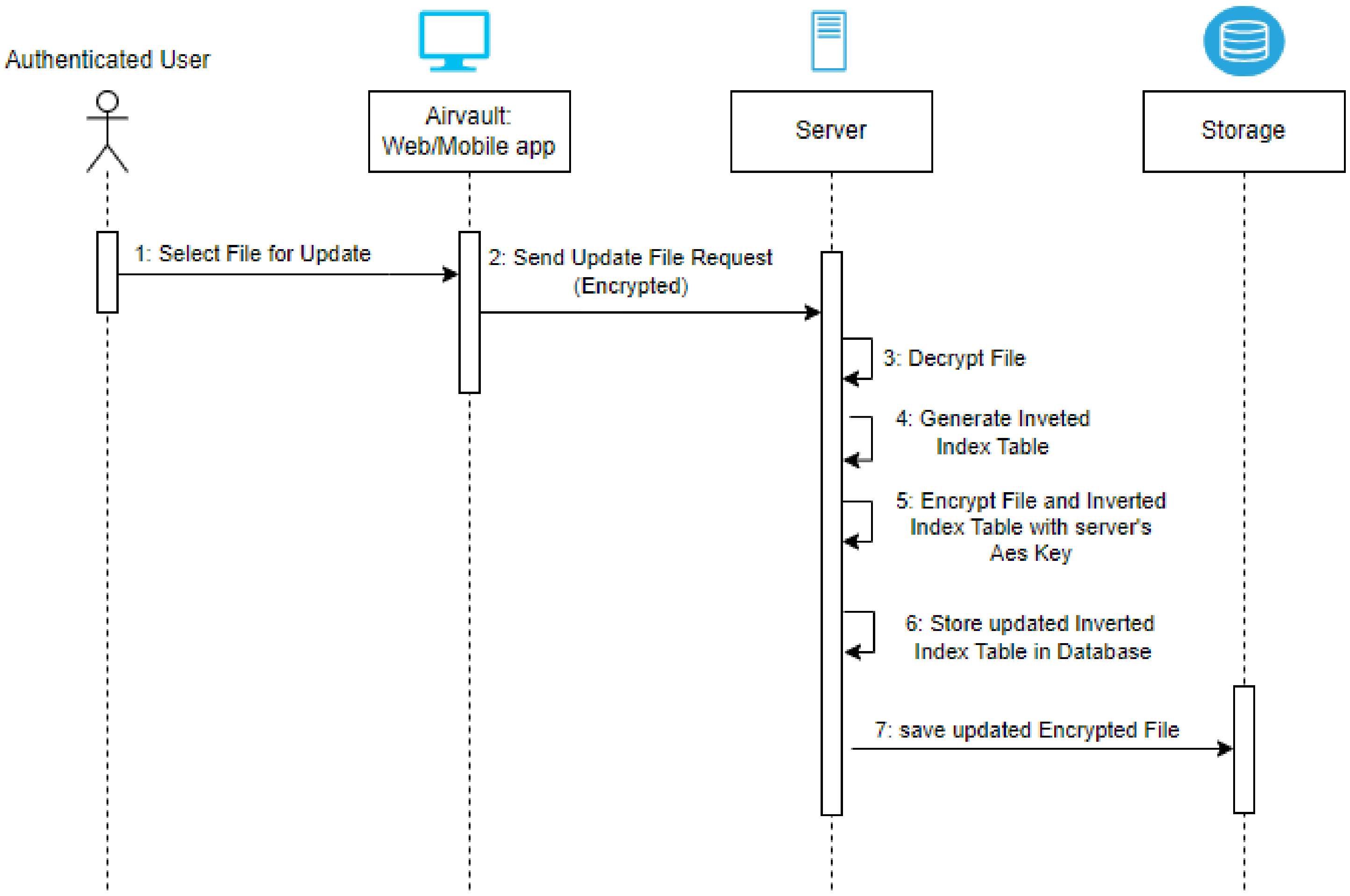

Authenticated User
Airvault: Web/Mobile app
Server
Storage
1: Select File for Update
2: Send Update File Request (Encrypted)
3: Decrypt File
4: Generate Inveted Index Table
5: Encrypt File and Inverted Index Table with server's Aes Key
6: Store updated Inverted Index Table in Database
7: save updated Encrypted File

Figure 32: Sequence Diagram Update File

4.3.3. Activity Diagram

This activity diagram illustrates the series of activities which will occur at each level of the system as the user updates files. It shows detailed activities of file processing before encrypting and storing it in storage.

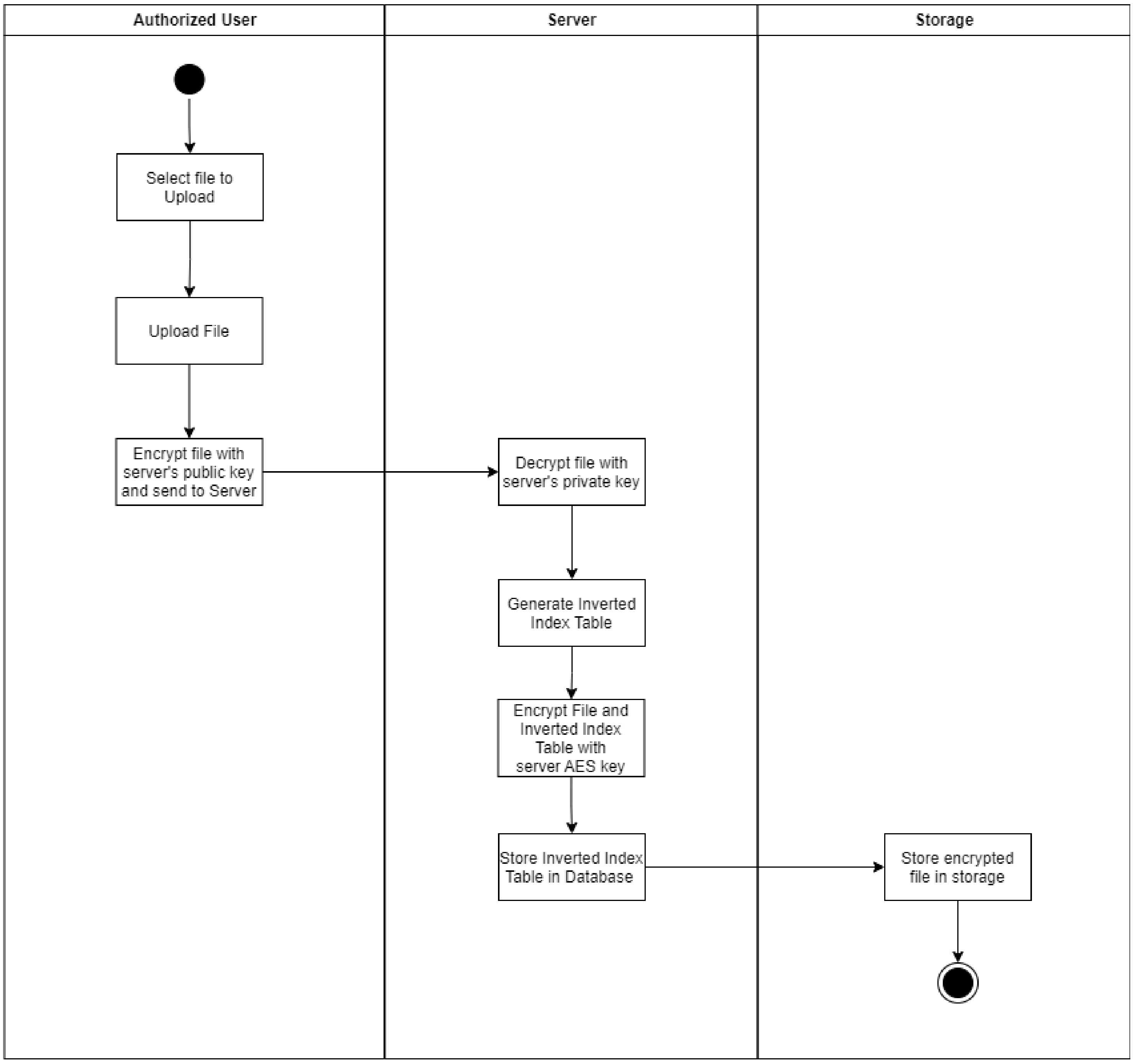

Authorized User
Server
Storage
Select file to Upload
Upload File
Encrypt file with server's public key and send to Server
Decrypt file with server's private key
Generate Inverted Index Table
Encrypt File and Inverted Index Table with server AES key
Store Inverted Index Table in Database
Store encrypted file in storage

Figure 33: Activity Diagram Update File

4.3.4. Block Diagram

This block diagram illustrates the series of events that will occur as the user updates file(s).

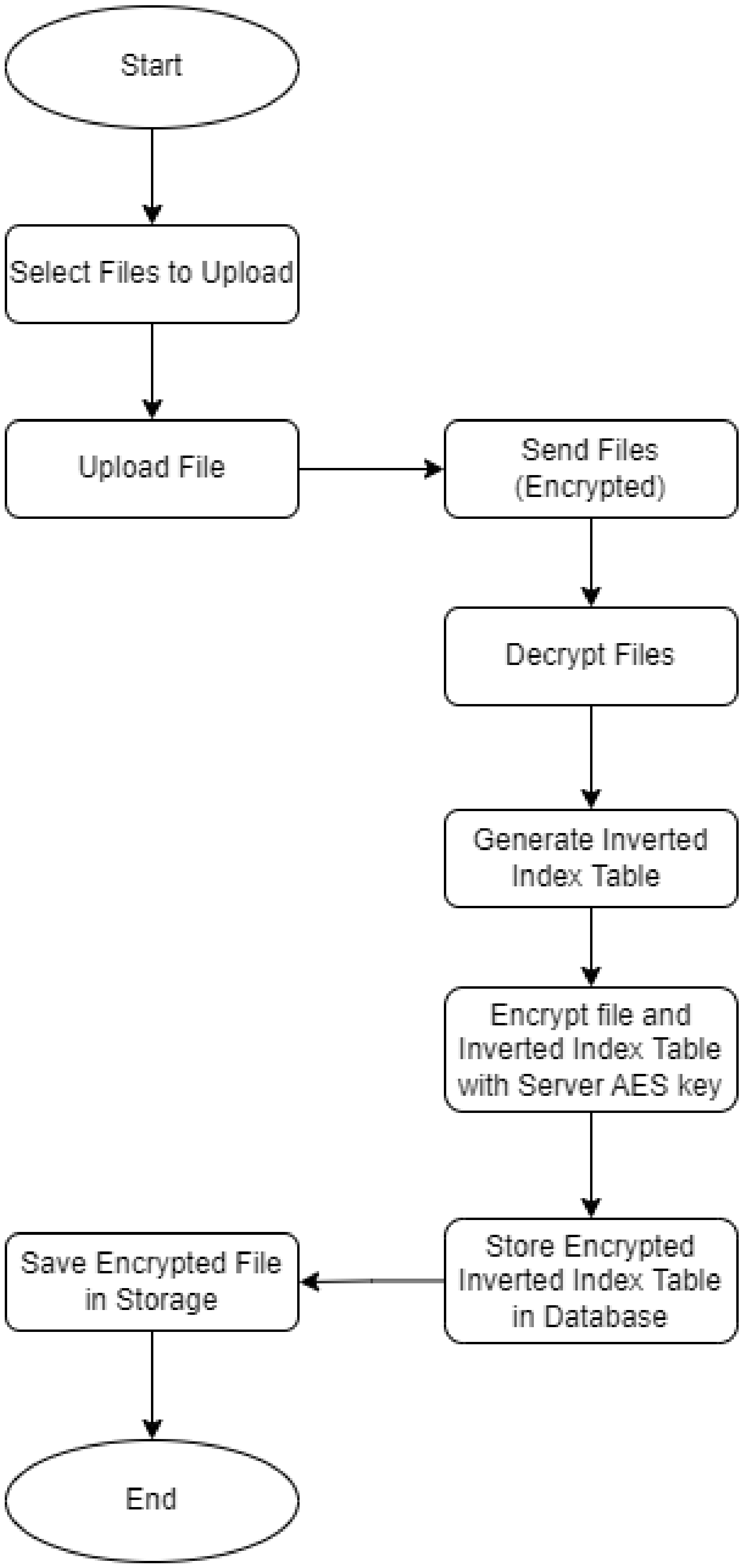

Start
Select Files to Upload
Upload File
Send Files (Encrypted)
Decrypt Files
Generate Inverted Index Table
Encrypt file and Inverted Index Table with Server AES key
Store Encrypted Inverted Index Table in Database
Save Encrypted File in Storage
End

Figure 34: Block Diagram Update File

4.4. Delete File

4.4.1. Use Case Diagram

This use case represents how the user interacts with the system while deleting a file. User can delete the file if and only if he has the access to delete. When the file is deleted the delete query will be sent to the server for further processing.

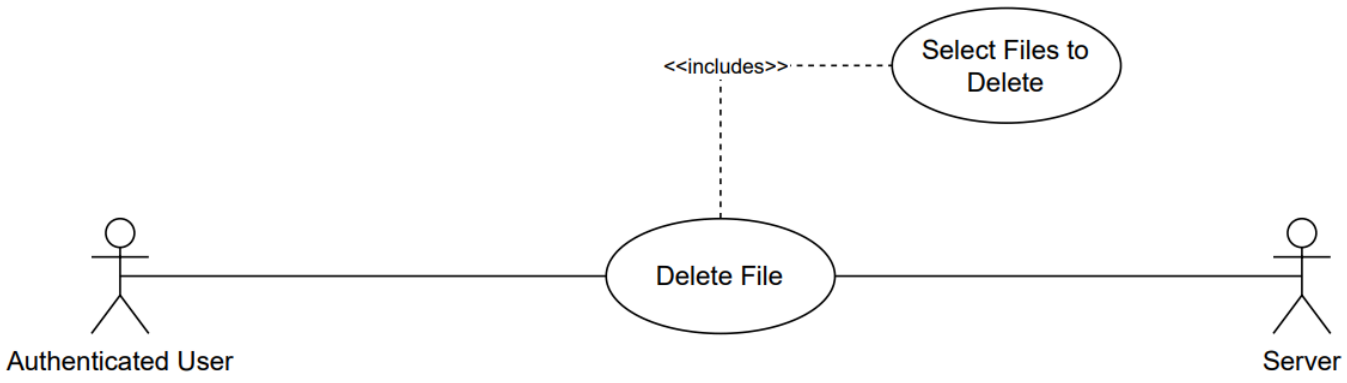


Figure 35: Use Case Diagram Delete File

This use case table outlines the interactions between the user and the system for the Delete File functionality.

Use Case Name	Delete File
Actor	User
Trigger	User clicks on delete file button
Precondition	User must logged into their account
Basic Path	1. User clicks the delete file button for specific file 2. System checks, If user has delete access to that file 3. System will delete the file on the server, If the user has access.
Alternative Path	System responds with an error message, If users don't have download access.

Post Condition	File will be removed from storage
Others	File should be available on storage

Table 9: Use Case Table Delete File

4.4.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the user deletes a file. A delete query will be sent to the server if the user has the access then the file is deleted from the storage and is also removed from all the database tables as well.

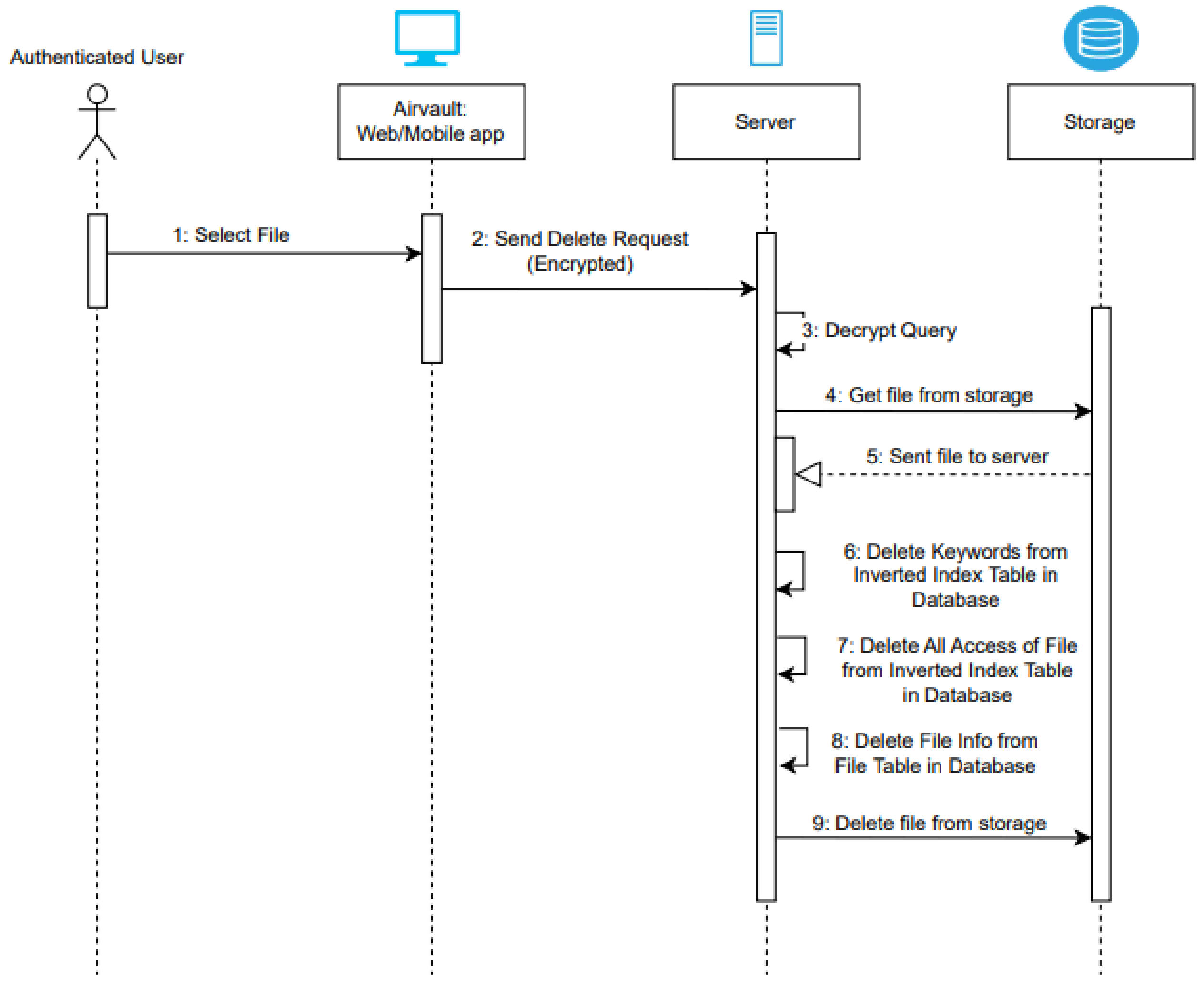

Authenticated User
Airvault: Web/Mobile app
Server
Storage
1: Select File
2: Send Delete Request (Encrypted)
3: Decrypt Query
4: Get file from storage
5: Sent file to server
6: Delete Keywords from Inverted Index Table in Database
7: Delete All Access of File from Inverted Index Table in Database
8: Delete File Info from File Table in Database
9: Delete file from storage

Figure 36: Sequence Diagram Delete File

4.4.3. Activity Diagram

This activity diagram illustrates the series of activities which will occur as the user deletes a file. A delete query will be sent to the server if the user has the access then the file is deleted from the storage and is also removed from all the database tables as well.

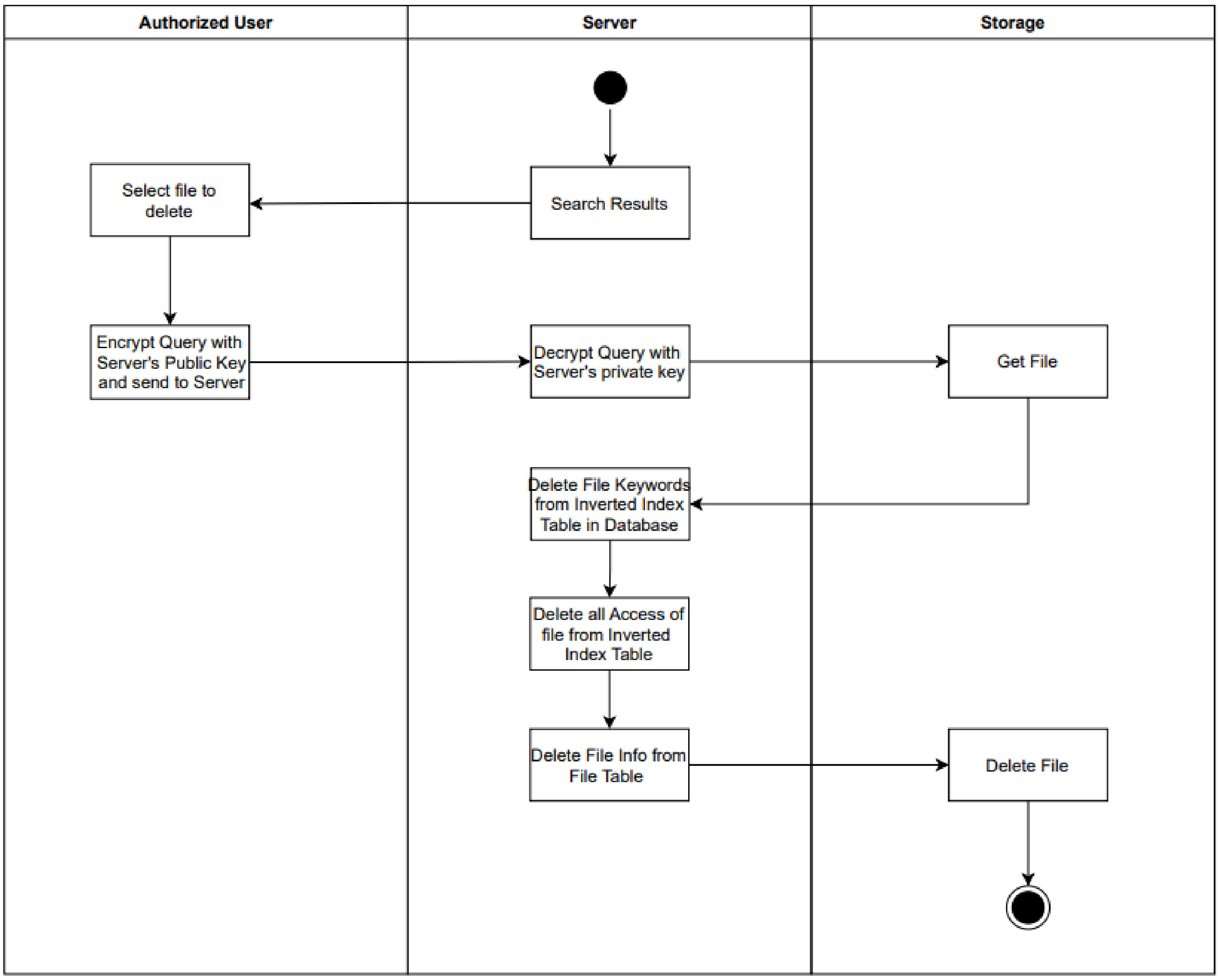

Authorized User
Server
Storage
Search Results
Select file to delete
Encrypt Query with Server's Public Key and send to Server
Decrypt Query with Server's private key
Get File
Delete File Keywords from Inverted Index Table in Database
Delete all Access of file from Inverted Index Table
Delete File Info from File Table
Delete File

Figure 37: Activity Diagram Delete File

4.4.4. Block Diagram

This block diagram illustrates the series of events which will occur as the user deletes a file from the storage.

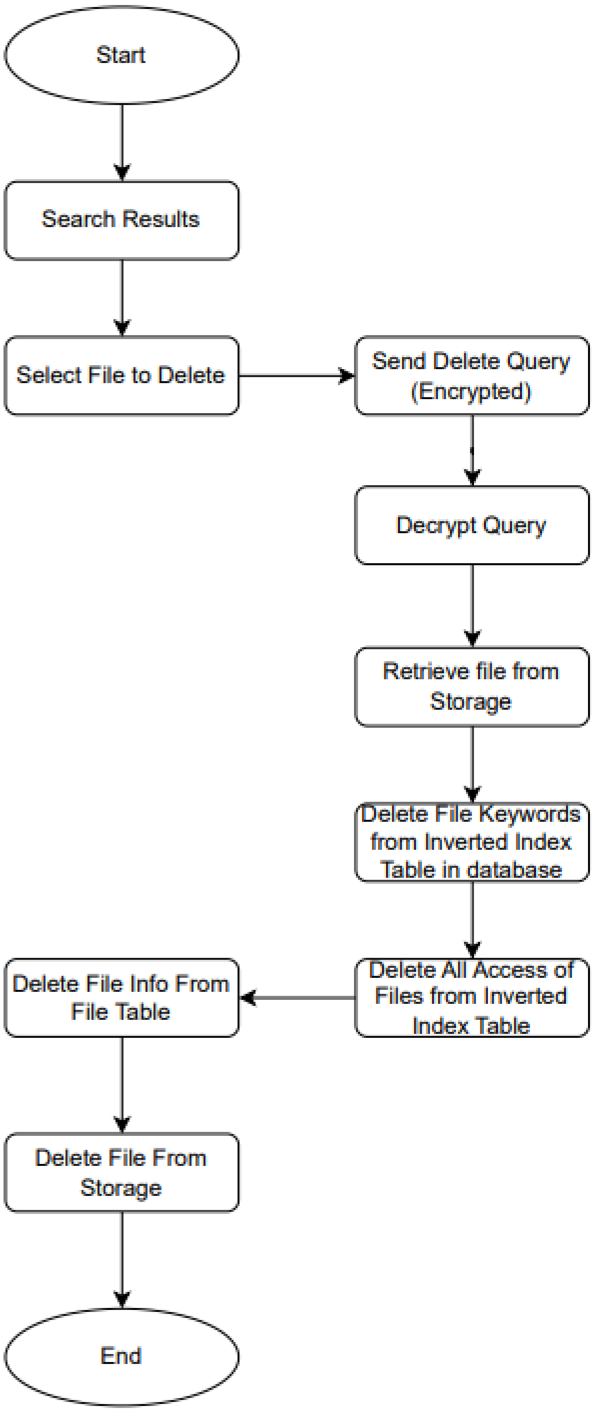

Start
Search Results
Select File to Delete
Send Delete Query (Encrypted)
Decrypt Query
Retrieve file from Storage
Delete File Keywords from Inverted Index Table in database
Delete All Access of Files from Inverted Index Table
Delete File Info From File Table
Delete File From Storage
End

Figure 38: Block Diagram Delete File

4.5. Download Encrypted File

4.5.1. Use Case Diagram

This use case represents how the user interacts with the system while downloading encrypted files.

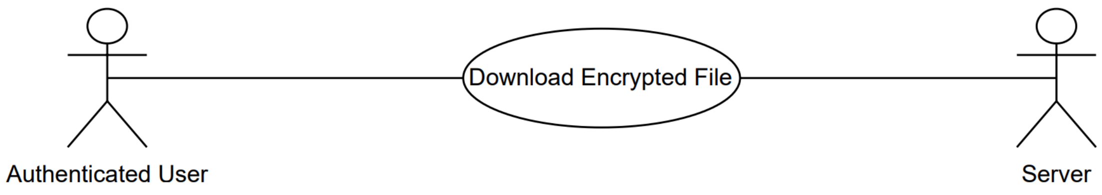


Figure 39: Use Case Diagram Download Encrypted File

This use case table outlines the interactions between the user and the system for the Download Encrypted File functionality.

Use Case Name	Download Encrypted File
Actor	User
Trigger	User clicks on download encrypted file button
Precondition	User must logged into their account
Basic Path	1. User clicks the download encrypted file button for specific file 2. System checks, If user has download access to that file 3. System will download the encrypted file on the user's local machine, If the user has access.
Alternative Path	System responds with an error message, If users don't have download access.
Post Condition	User will have that file on their local machine
Others	File should be available on storage

Table 10: Use Case Table Download Encrypted File

4.5.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the user downloads an encrypted file. A request will be sent to the server and then the file will be retrieved.

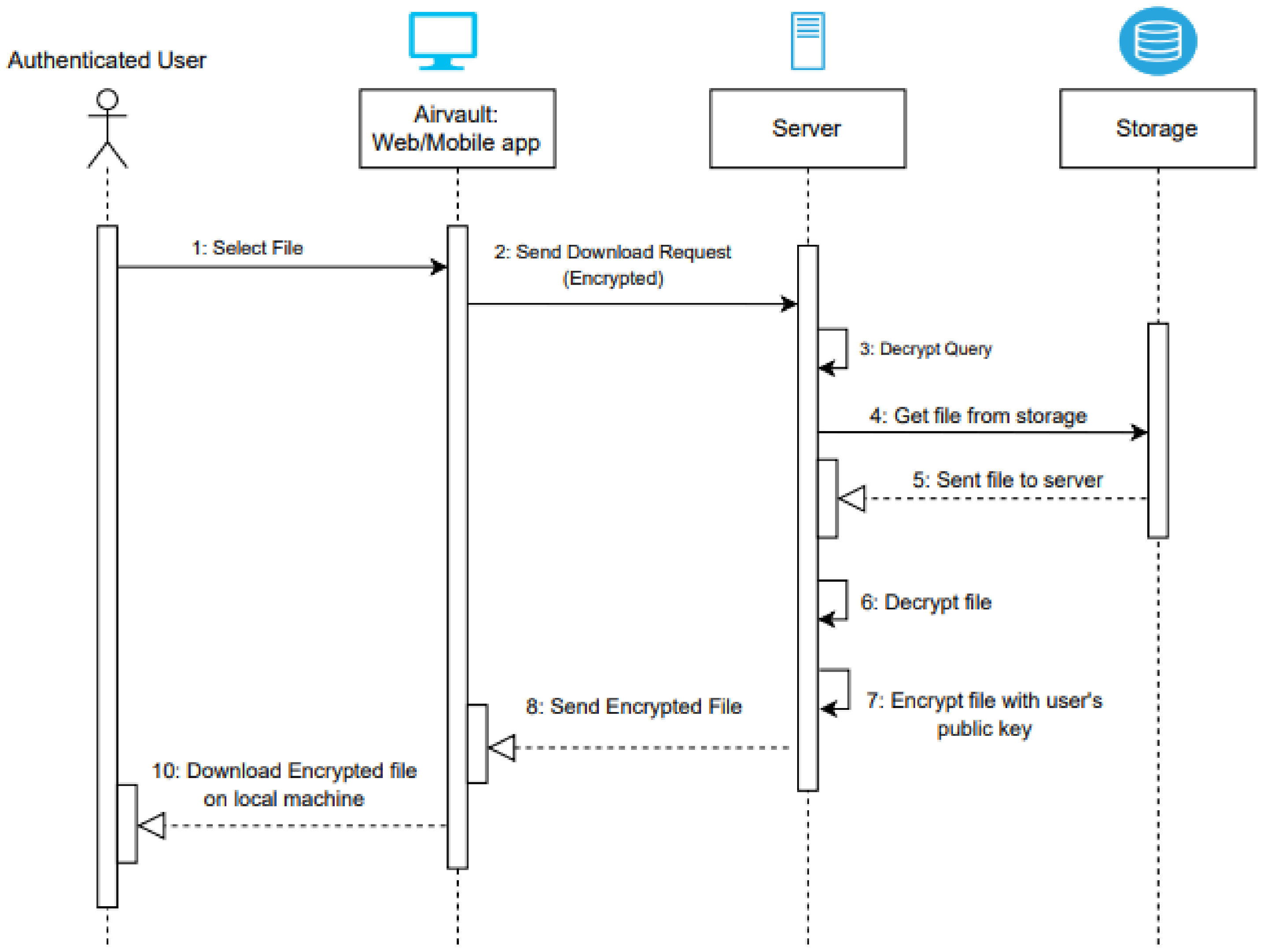


Figure 40: Sequence Diagram Download Encrypted File

4.5.3. Activity Diagram

This activity diagram illustrates the series of activities which will occur at each level of the system as the user downloads an encrypted file. A request will be sent to the server and an encrypted file will be sent which can be downloaded.

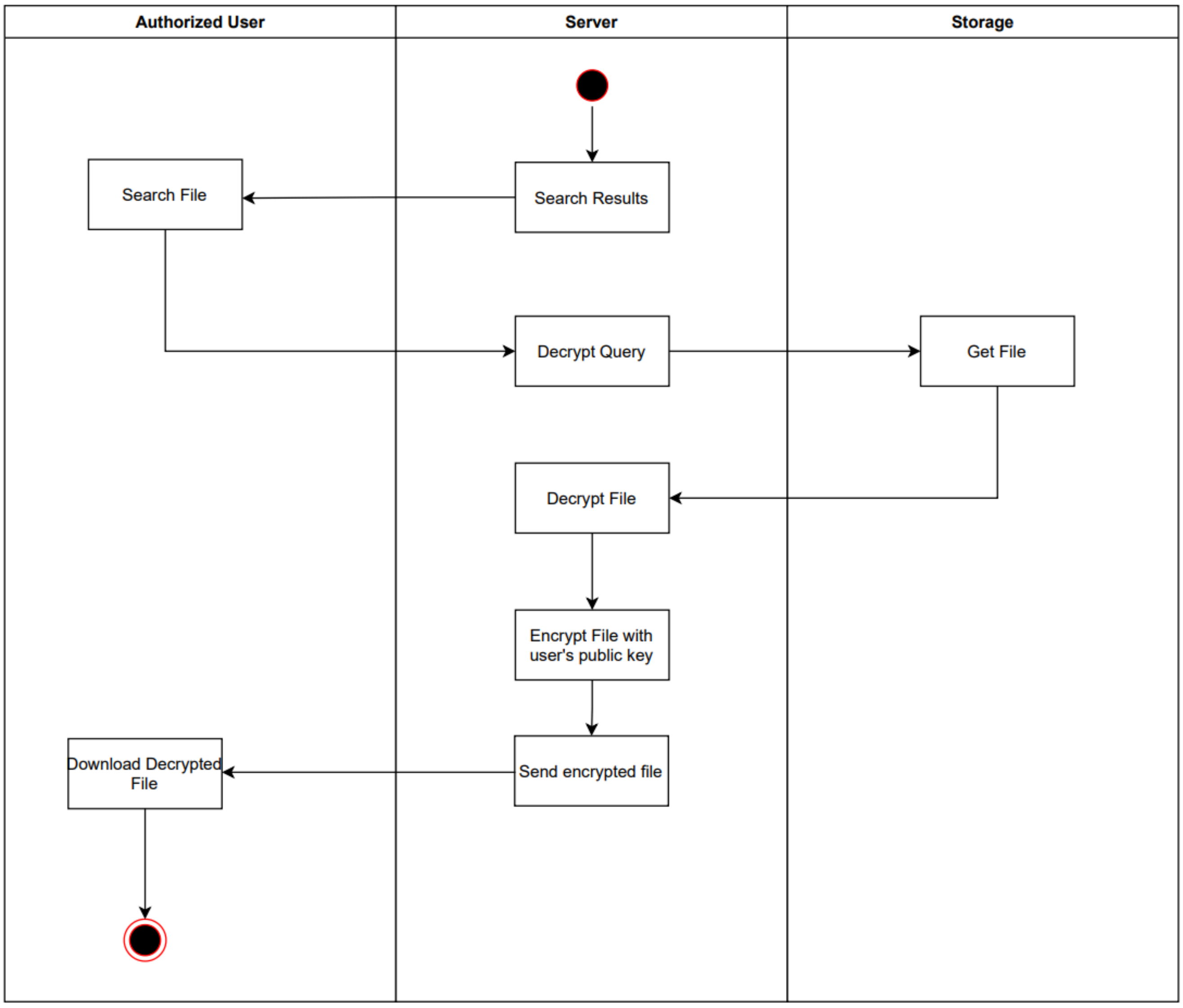


Figure 41: Activity Diagram Download Encrypted File

4.5.4. Block Diagram

This block diagram illustrates the series of events which will occur as the user clicks on the downloaded encrypted file.

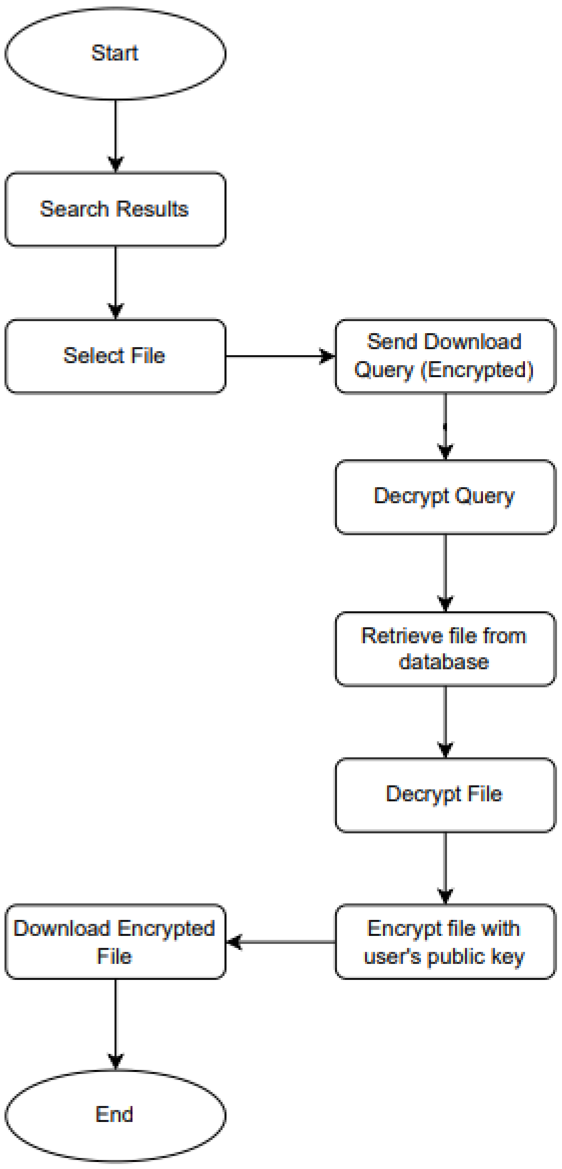

Start
Search Results
Select File
Send Download Query (Encrypted)
Decrypt Query
Retrieve file from database
Decrypt File
Encrypt file with user's public key
Download Encrypted File
End

Figure 42: Block Diagram Download Encrypted File

4.6. Download Decrypted File

4.6.1. Use Case Diagram

This use case represents how the user interacts with the system while downloading decrypted file.

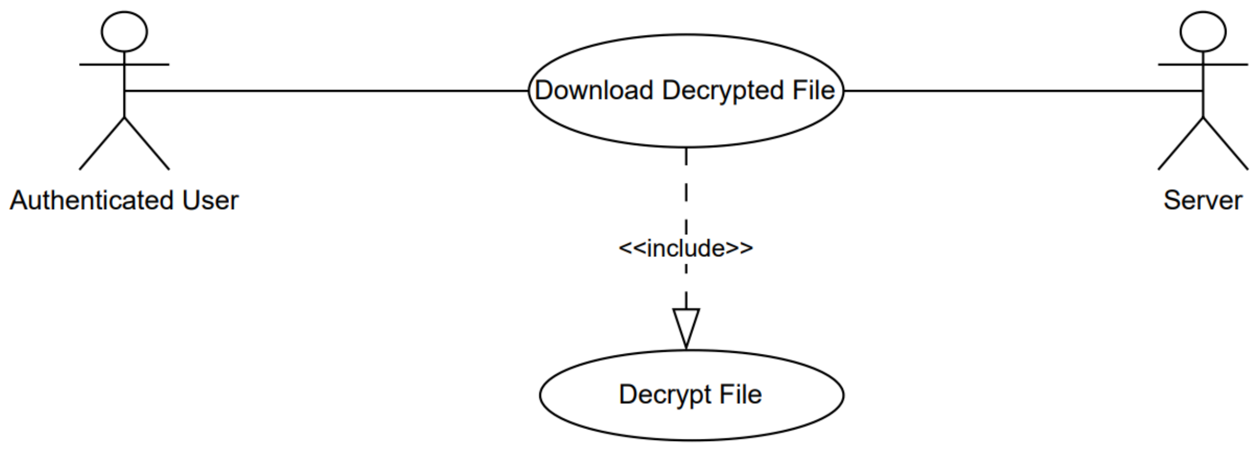


Figure 43: Use Case Diagram Download Decrypted File

This use case table outlines the interactions between the user and the system for the Download Decrypted File functionality.

Use Case Name	Download Decrypted File
Actor	User
Trigger	User clicks on download decrypted file button
Precondition	User must logged into their account
Basic Path	1. User clicks the download decrypted file button for specific file 2. System checks, If user has download access to that file 3. System will decrypt that file on server 4. System download that file on the user's local machine, If the user has access.
Alternative Path	System responds with an error message, If users don’t have download access.

Post Condition	User will have that file on their local machine
Others	File should be available on storage

Table 11: Use Case Table Download Decrypted File

4.6.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the user downloads an decrypted file. A request will be sent to the server and then the file will be retrieved.

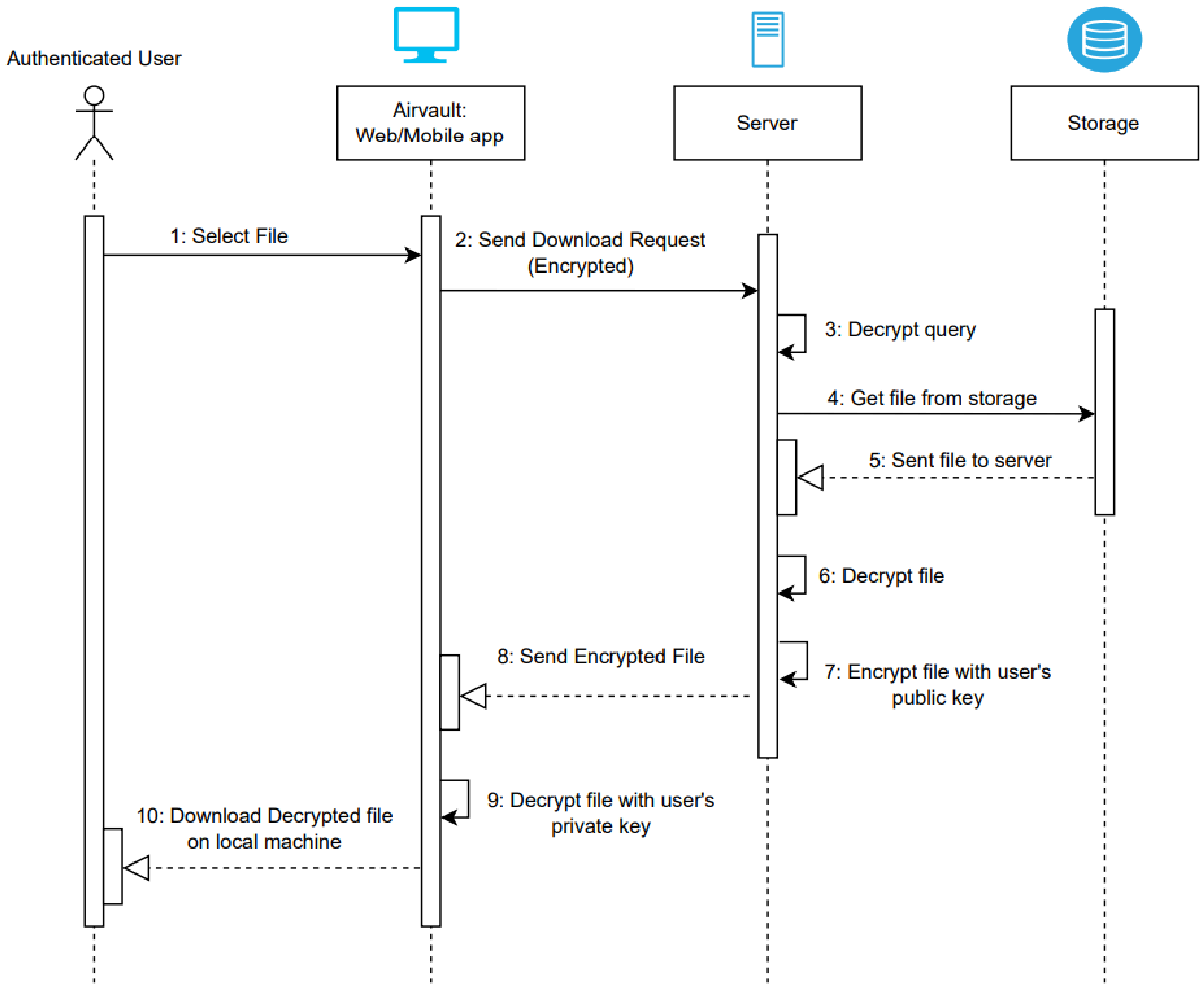


Figure 44: Sequence Diagram Download Decrypted File

4.6.3. Activity Diagram

This activity diagram illustrates the series of activities which will occur at each level of the system as the user downloads a decrypted file. A request will be sent to the server and an encrypted file will be sent which can be downloaded after decryption.

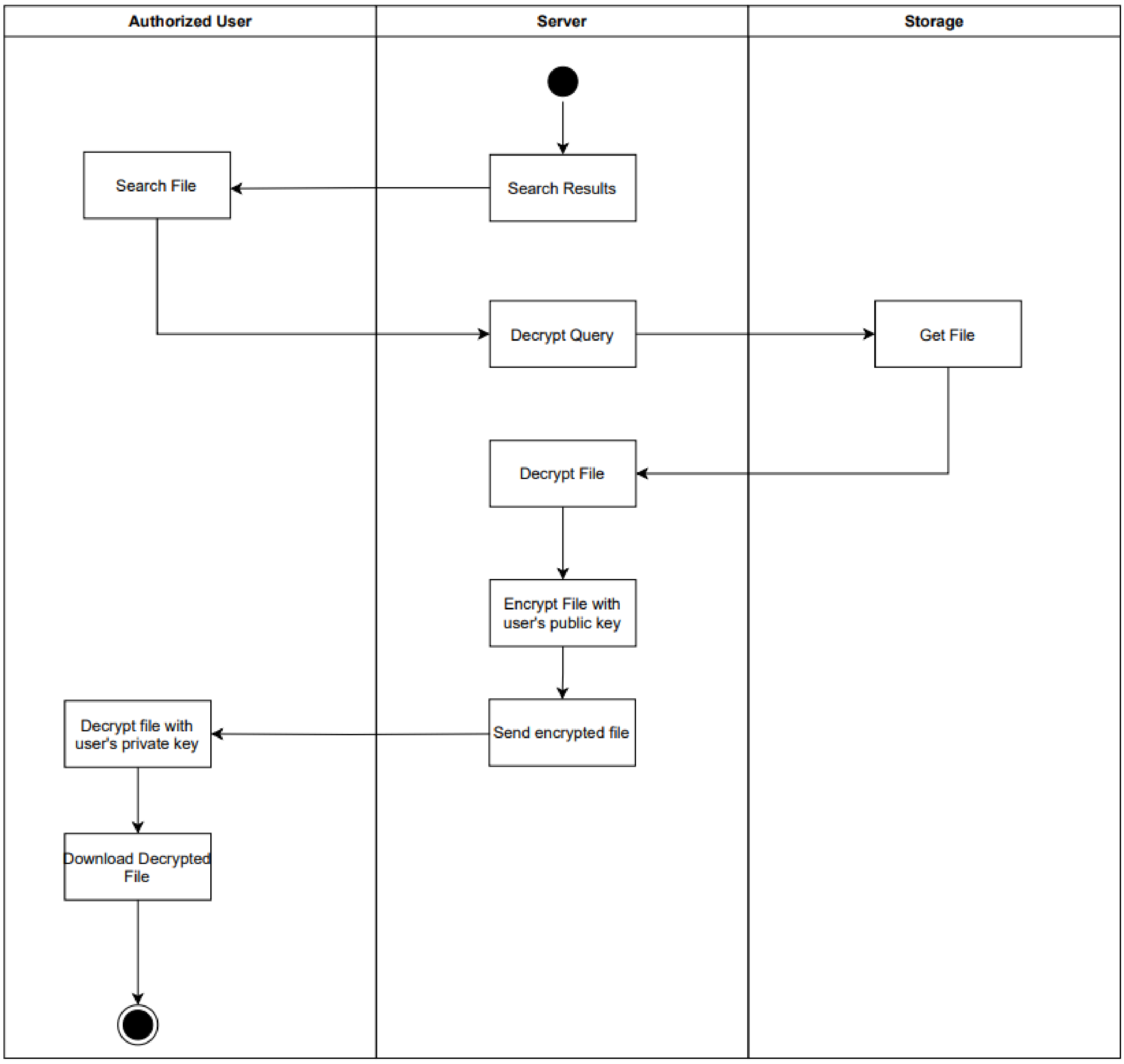


Figure 45: Activity Diagram Download Decrypted File

4.6.4. Block Diagram

This block diagram illustrates the series of events which will occur as the user clicks on the downloaded decrypted file.

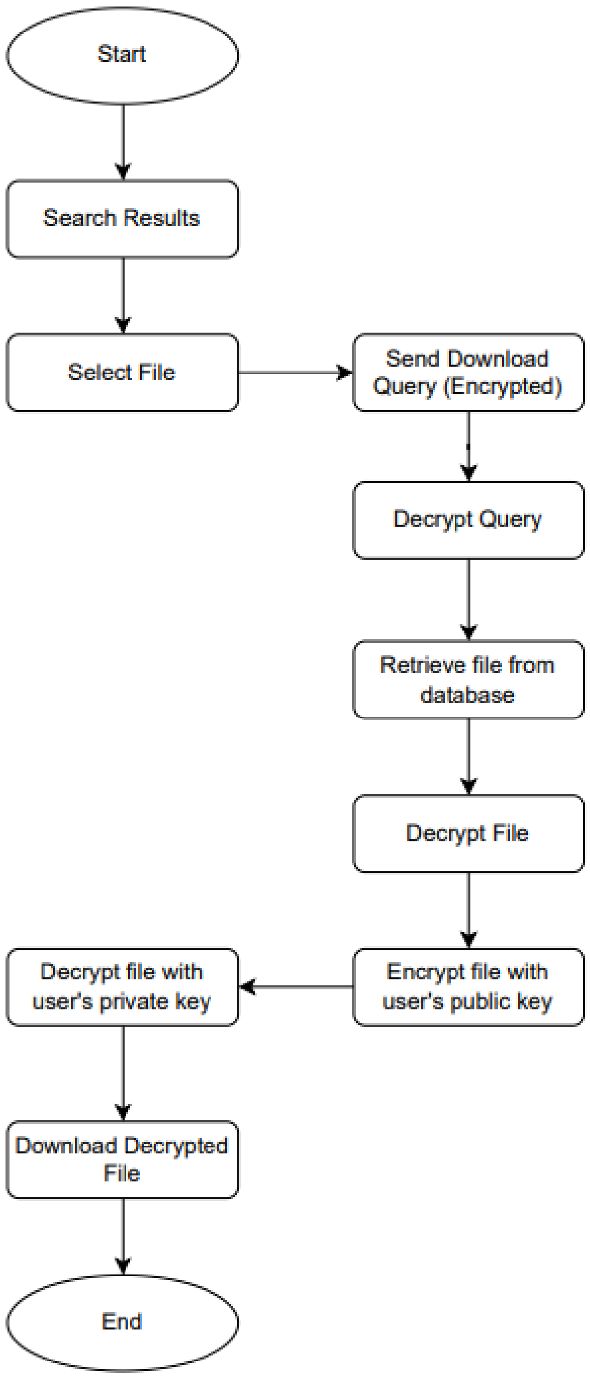

Start
Search Results
Select File
Send Download Query (Encrypted)
Decrypt Query
Retrieve file from database
Decrypt File
Encrypt file with user's public key
Decrypt file with user's private key
Download Decrypted File
End

Figure 46: Block Diagram Download Decrypted File

4.7. Search Files within the Department

4.7.1. Use Case Diagram

This use case illustrates that as the process of searching within the department is initiated users have the option to select the filter (search by file name or search via any keyword within the file). While searching within the department user enters a string on the basis of which user access is checked and results will be shown to the user.

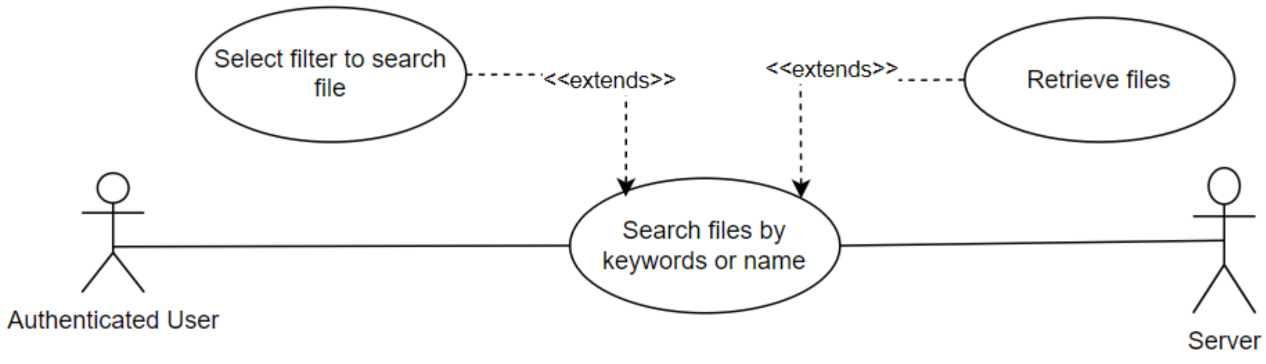

Select filter to search file
<<extends>>
<<extends>>
Retrieve files
Search files by keywords or name
Authenticated User
Server

Figure 47: Use Case Diagram Search files within the Department

This use case table outlines the interactions between the user and the system for the Search any Keyword within Department functionality.

Use Case Name	Search any Keyword within Department
Actor	User
Trigger	User clicks on search page
Precondition	User must logged into their account
Basic Path	1. User redirect to search page 2. User input keyword to search and click search button 3. System search that keyword on all files within department 4. System check access for files 5. System sends files which have keyword and accessible to user 6. User can see a list of files on his screen
Alternative Path	System responds with an error message, If keyword not found

Post Condition	User will have a list of files which have that keyword
Others	System only shows those files which are accessible to user

Table 12: Use Case Table Search files within the Department

4.7.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the user searches for a file within the department. Users can search files by name or by keywords, search results will be displayed to the user after access check.

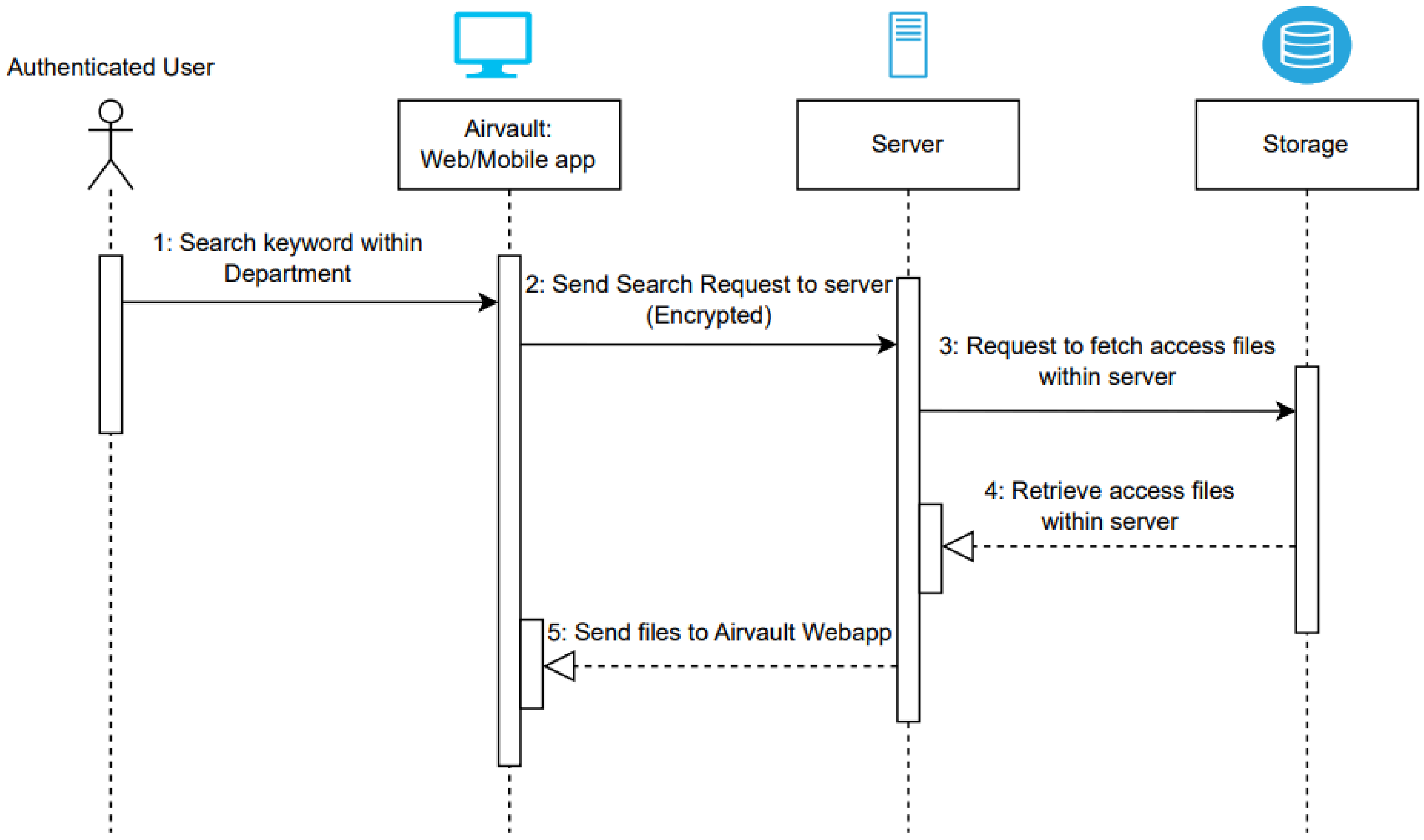

Authenticated User
Airvault: Web/Mobile app
Server
Storage
1: Search keyword within Department
2: Send Search Request to server (Encrypted)
3: Request to fetch access files within server
4: Retrieve access files within server
5: Send files to Airvault Webapp

Figure 48: Sequence Diagram Search files within the Department

4.7.3. Activity Diagram

This activity diagram illustrates the series of activities which will occur as the user searches for a file within the department. Users can search files by name or by keywords, search results will be displayed to the user after access check.

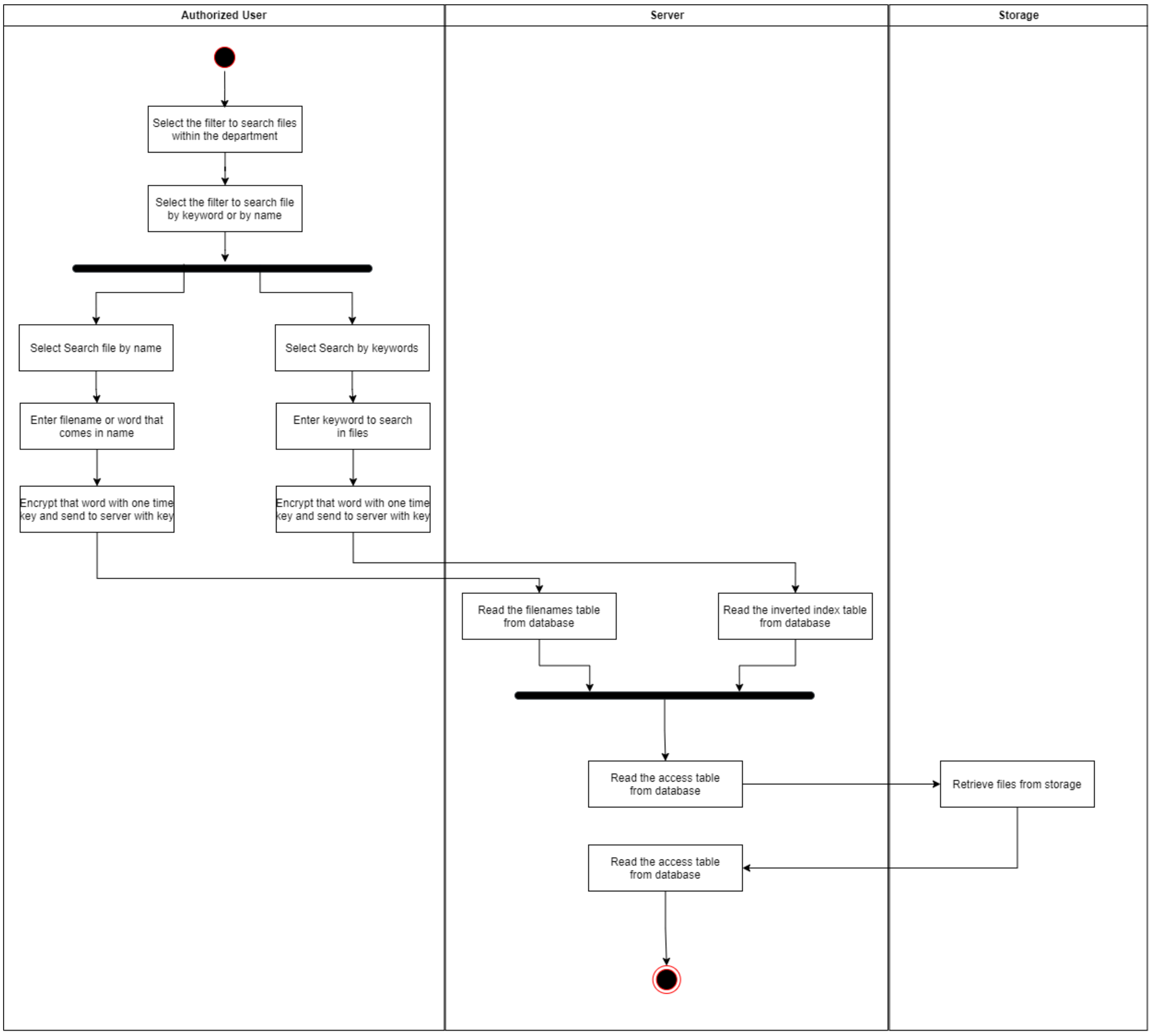

Authorized User
Server
Storage
Select the filter to search files within the department
Select the filter to search file by keyword or by name
Select Search file by name
Select Search by keywords
Enter filename or word that comes in name
Enter keyword to search in files
Encrypt that word with one time key and send to server with key
Encrypt that word with one time key and send to server with key
Read the filenames table from database
Read the inverted index table from database
Read the access table from database
Retrieve files from storage
Read the access table from database

Figure 49: Activity Diagram Search files within the Department

4.7.4. Block Diagram

This block diagram illustrates the series of events which will occur as the user searches for files within the department.

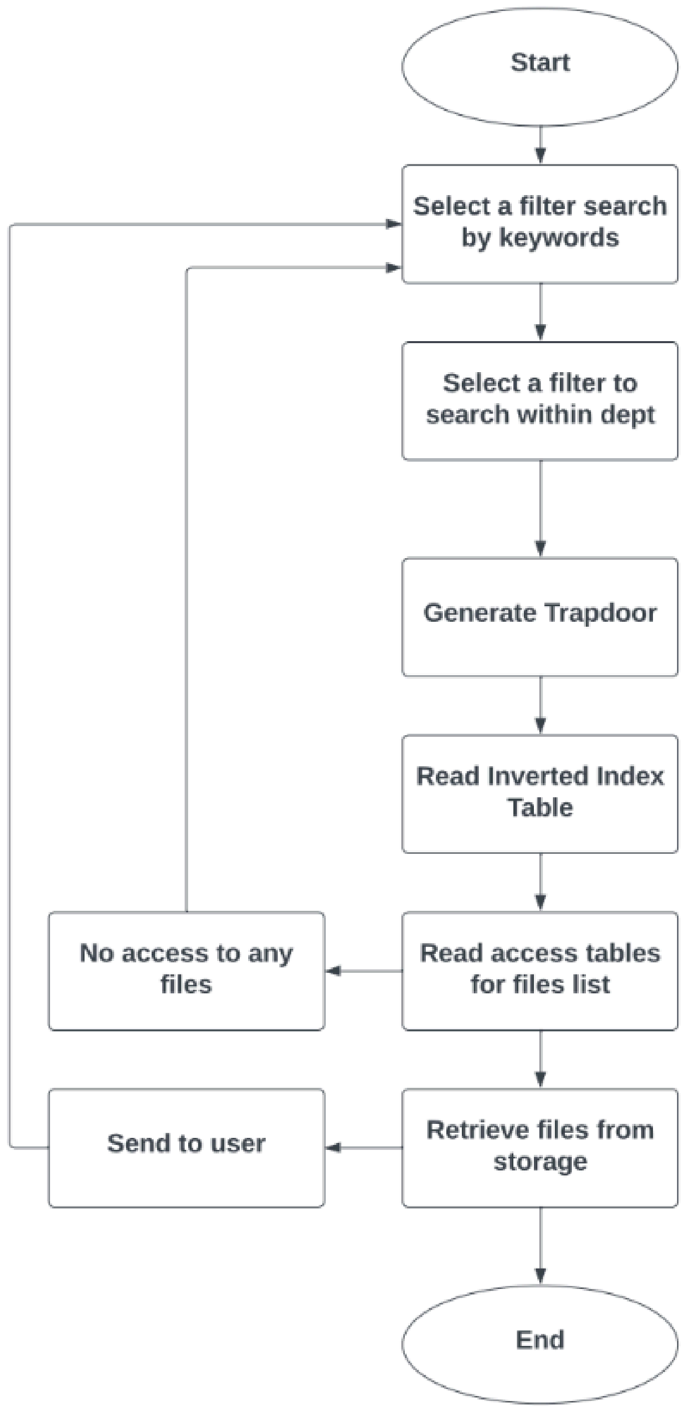

Start
Select a filter search by keywords
Select a filter to search within dept
Generate Trapdoor
Read Inverted Index Table
Read access tables for files list
No access to any files
Retrieve files from storage
Send to user
End

Figure 50: Block Diagram Search files within the Department

4.8. Search Files across the Department

4.8.1. Use Case Diagram

This use case illustrates that as the process of searching across the department is initiated users have the option to select the filter (search by file name or search via any keyword within the file). While searching across the department, the user specifies the department name and enters a string on the basis of which user access is checked and results will be shown to the user.

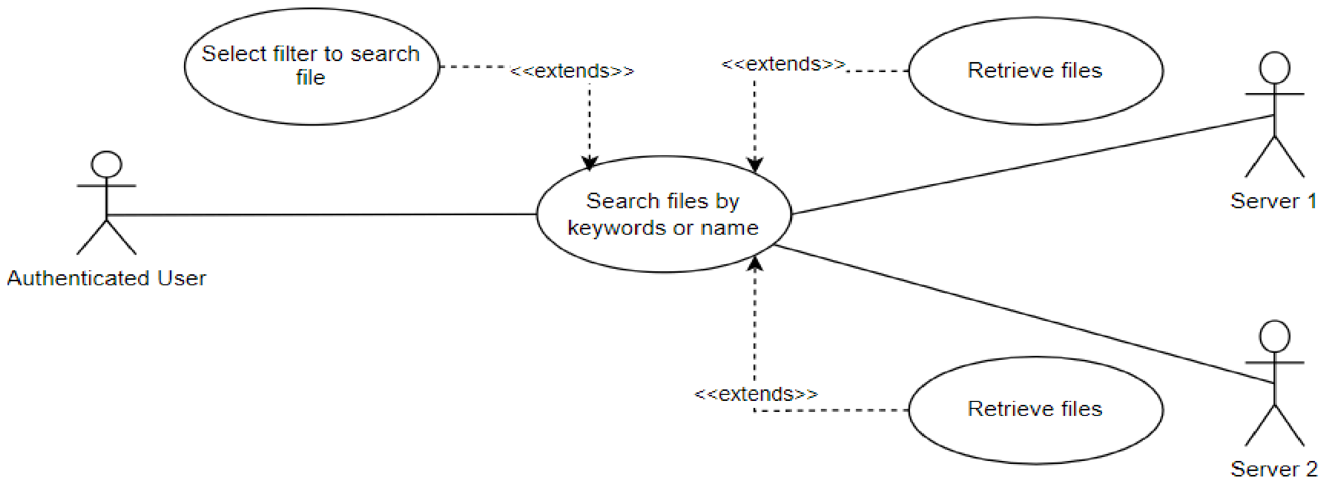


Figure 51: Use Case Diagram Search Files across the Department

This use case table outlines the interactions between the user and the system for the Search any Keyword across the Department functionality.

Use Case Name	Search any Keyword across the Department
Actor	User
Trigger	User clicks on search page
Precondition	User must logged into their account
Basic Path	1. User redirect to search page 2. User input keyword to search and click search button 3. System search that keyword on all files. 4. System check access for files 5. System sends files which have keyword and accessible to user

	6. User can see a list of files on his screen
Alternative Path	System responds with an error message, If keyword not found
Post Condition	User will have a list of files which have that keyword
Others	System only shows those files which are accessible to the user

Table 13: Use Case Table Search Files across the Department

4.8.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the user searches for a file across the department. Users can search files by name or by keywords, requests will be sent to the server and will be forwarded to the specific server for further processing. Search results will be displayed to the user after access check.

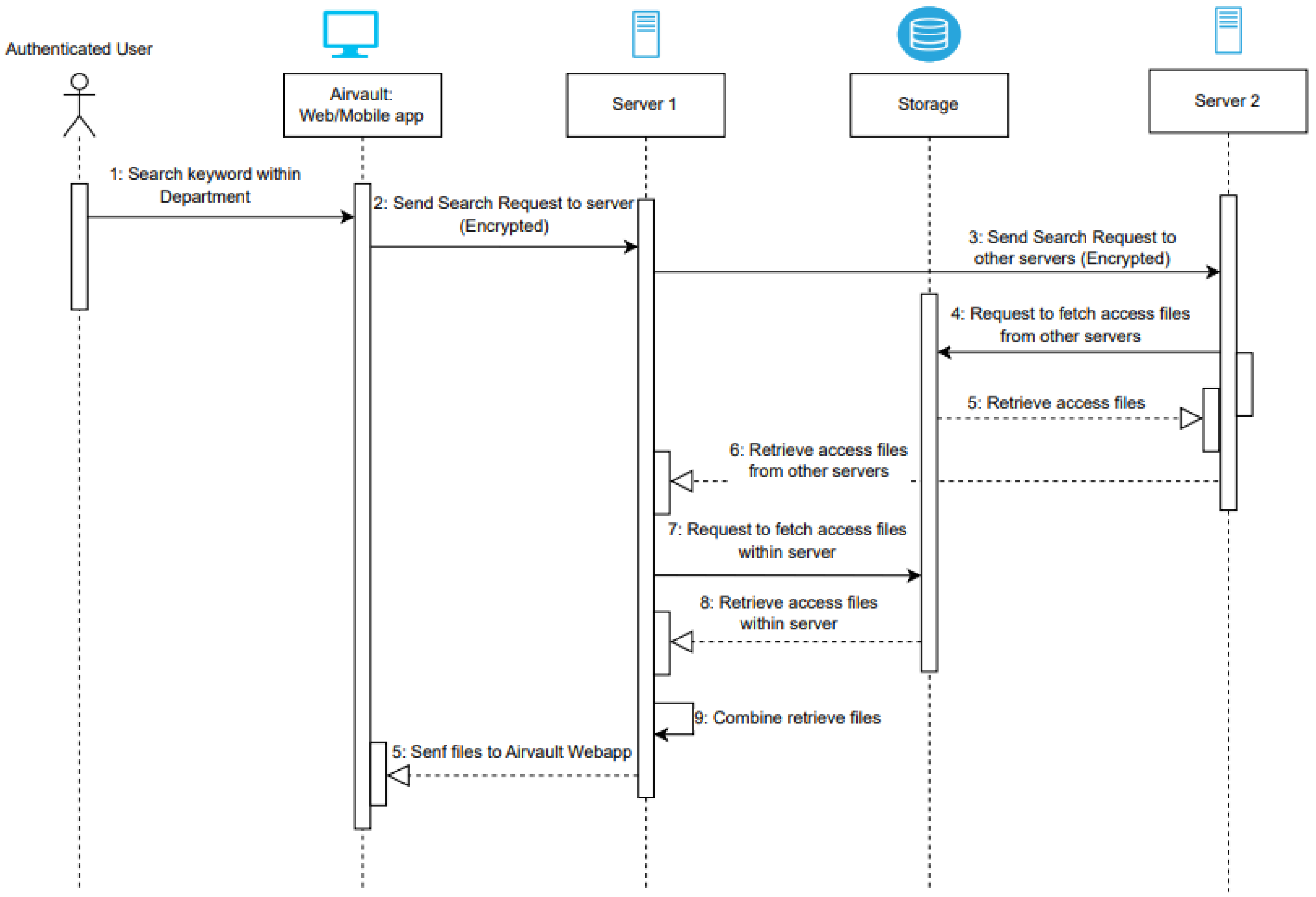

Authenticated User
Airvault: Web/Mobile app
Server 1
Storage
Server 2
1: Search keyword within Department
2: Send Search Request to server (Encrypted)
3: Send Search Request to other servers (Encrypted)
4: Request to fetch access files from other servers
5: Retrieve access files
6: Retrieve access files from other servers
7: Request to fetch access files within server
8: Retrieve access files within server
9: Combine retrieve files
5: Senf files to Airvault Webapp

Figure 52: Sequence Diagram Search Files across the Department

4.8.3. Activity Diagram

This sequence diagram illustrates the series of events which will occur as the user searches for a file across the department. Users can search files by name or by keywords, requests will be sent to the server and will be forwarded to the specific server for further processing. Search results will be displayed to the user after access check.

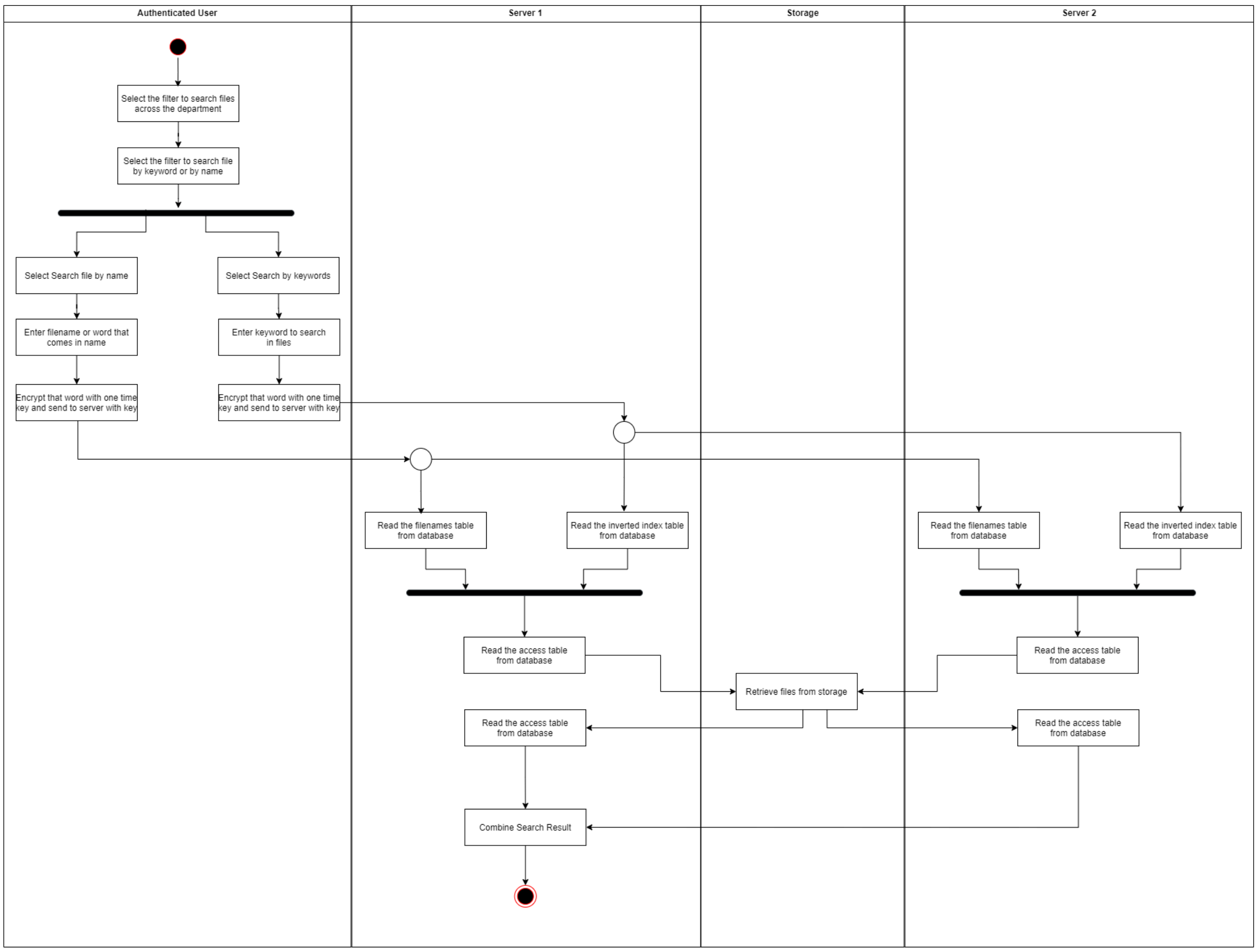

Authenticated User
Server 1
Storage
Server 2
Select the filter to search files across the department
Select the filter to search file by keyword or by name
Select Search file by name
Select Search by keywords
Enter filename or word that comes in name
Enter keyword to search in files
Encrypt that word with one time key and send to server with key
Encrypt that word with one time key and send to server with key
Read the filenames table from database
Read the inverted index table from database
Read the filenames table from database
Read the inverted index table from database
Read the access table from database
Read the access table from database
Retrieve files from storage
Read the access table from database
Read the access table from database
Combine Search Result

Figure 53: Activity Diagram Search Files across the Department

4.8.4. Block Diagram

This block diagram illustrates the series of events which will occur as the user searches for files across the department.

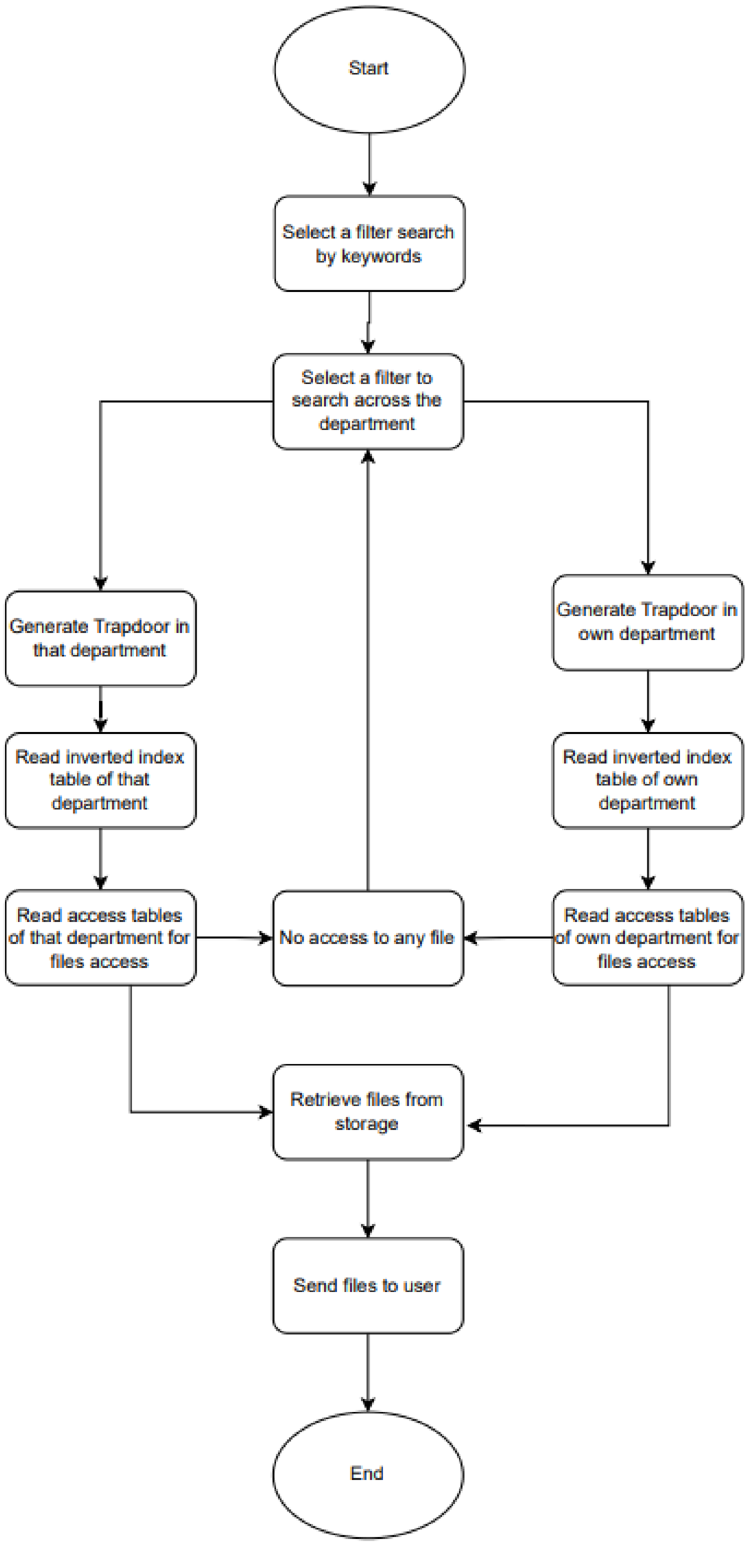

Start
Select a filter search by keywords
Select a filter to search across the department
Generate Trapdoor in that department
Read inverted index table of that department
Read access tables of that department for files access
No access to any file
Generate Trapdoor in own department
Read inverted index table of own department
Read access tables of own department for files access
Retrieve files from storage
Send files to user
End

Figure 54: Block Diagram Search Files across the Department

4.9. Manage Folder

4.9.1. Use Case Diagram

The use case illustrates that the user can manage the folders in the system. The user can add new folder, open folders and change the permission. The functionality is illustrated in the use case.

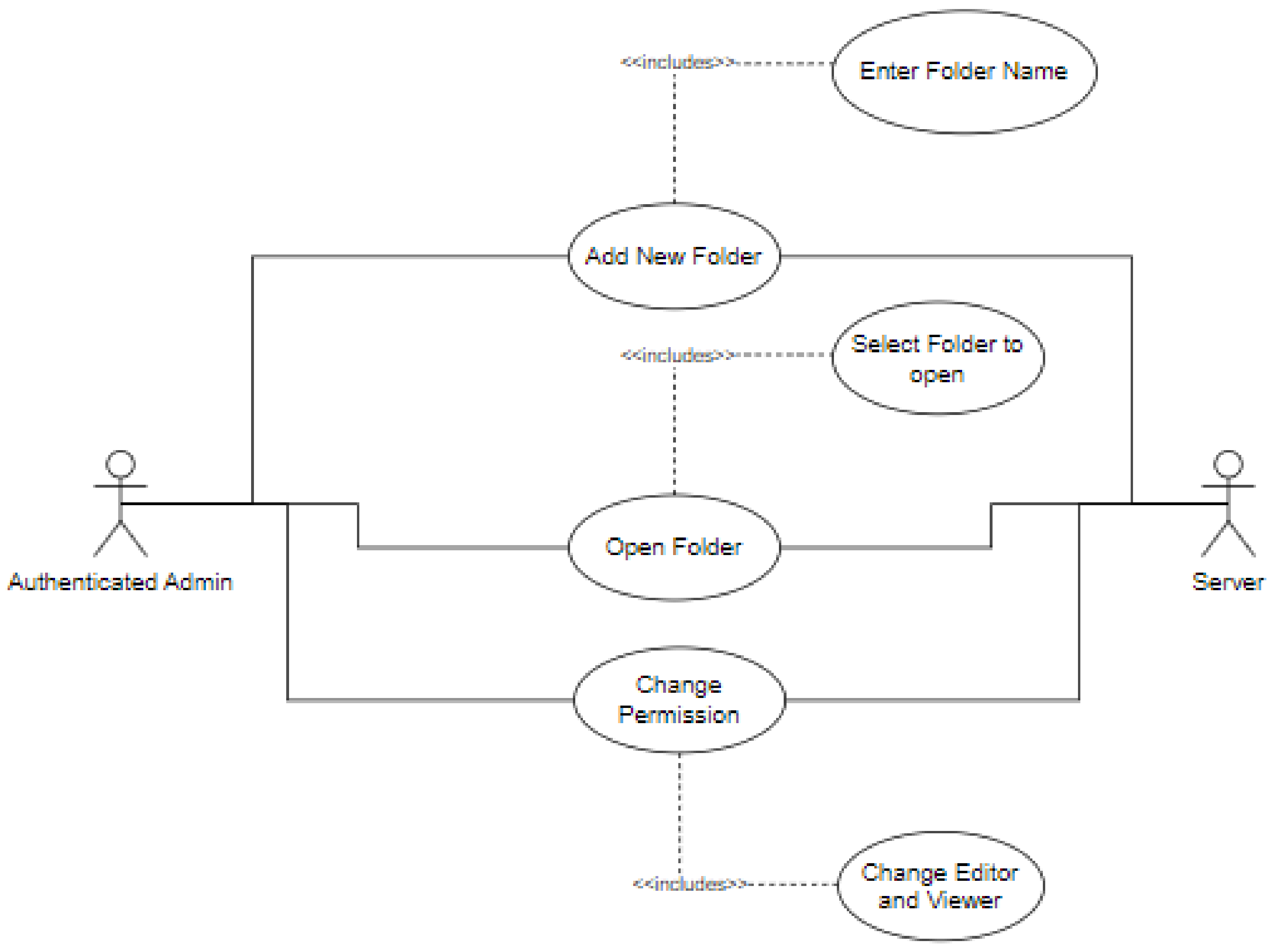


Figure 55: Use Case Diagram Manage Folder

This use case table outlines interactions between user and the system for the Manage Folder functionality.

Use Case Name	Manage Folder
Actor	User
Trigger	User clicks on add folder button, folder name, change permission

	button
Precondition	User must be logged into their account
Basic Path	1. User click on add folder button and send folder name in the add folder request to the server. 2. Clicks on folder. Folder's data will displayed on screen. 3. User click on change permissions button and send the updated editor and viewer array to the server with change permission request.
Alternative Path	System responds with local feedback, If users is selected as editor and viewer both.
Post Condition	Users can see that folder on vault page.
Others	System will not see those files, which are unaccessible to them

Table 14: Use Case Table Manage Folder

4.9.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the user manage folders. All the events of managing folder are illustrated in the figure below.

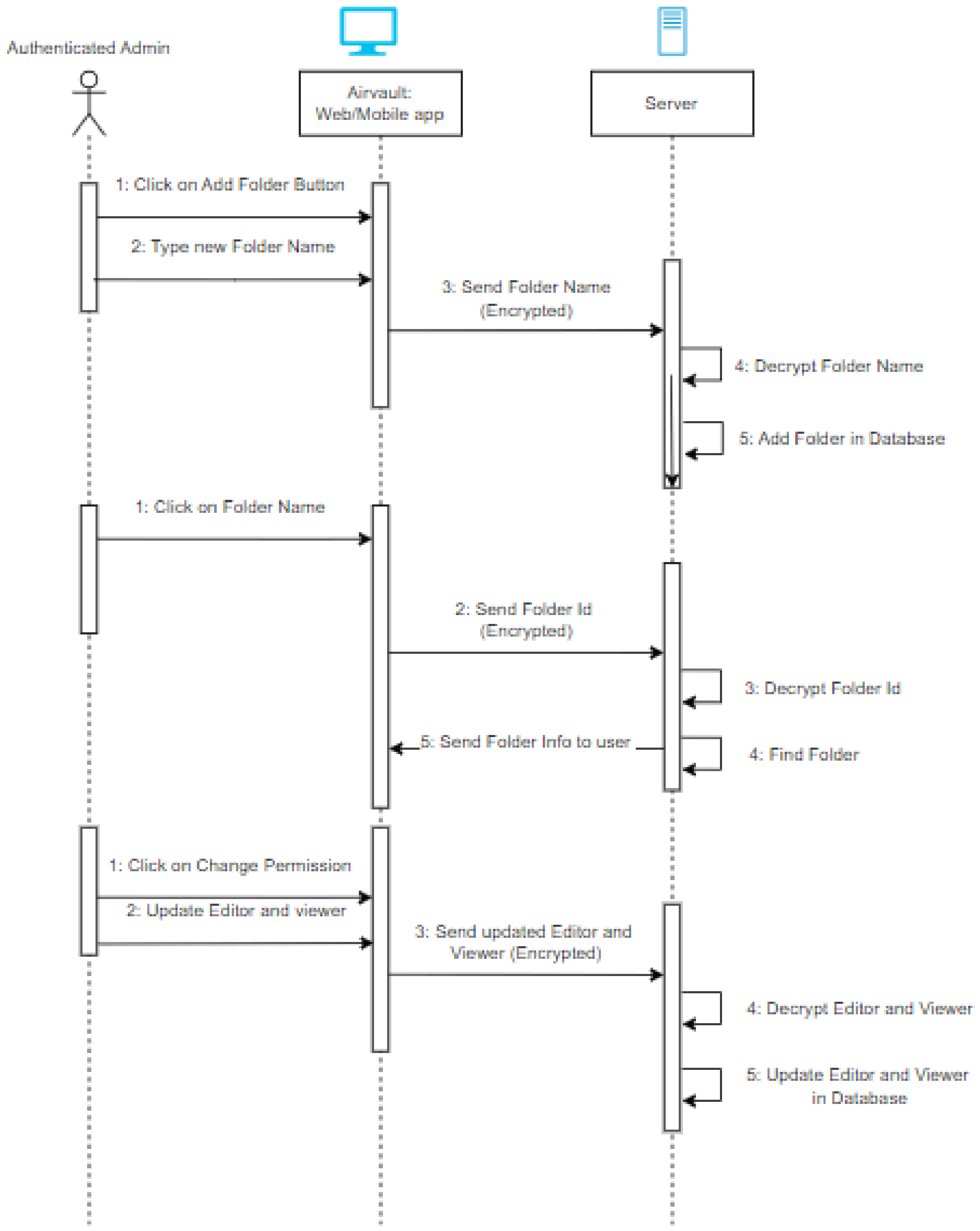

Authenticated Admin
Airvault: Web/Mobile app
Server
1: Click on Add Folder Button
2: Type new Folder Name
3: Send Folder Name (Encrypted)
4: Decrypt Folder Name
5: Add Folder in Database
1: Click on Folder Name
2: Send Folder Id (Encrypted)
3: Decrypt Folder Id
5: Send Folder Info to user
4: Find Folder
1: Click on Change Permission
2: Update Editor and viewer
3: Send updated Editor and Viewer (Encrypted)
4: Decrypt Editor and Viewer
5: Update Editor and Viewer in Database

Figure 56: Sequence Diagram Manage Folder

4.9.3. Activity Diagram

This activity diagram illustrates the series of activities which will occur at each level of the system as the user manage folders. It shows detailed activities of managing folders.

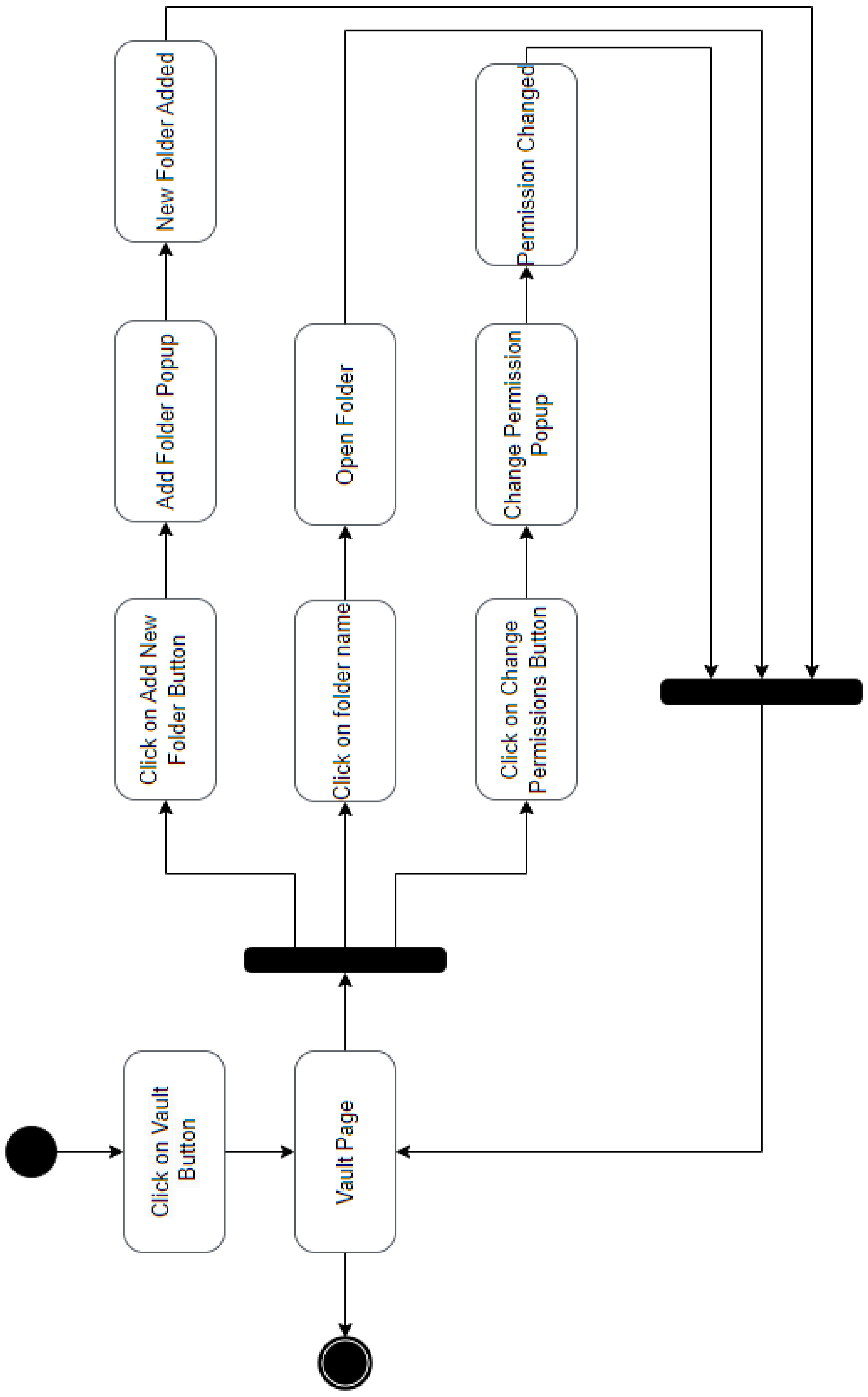


Figure 57: Activity Diagram Manage Folder

4.9.4. Block Diagram

This block diagram illustrates the series of events that will occur as the user manages folder(s).

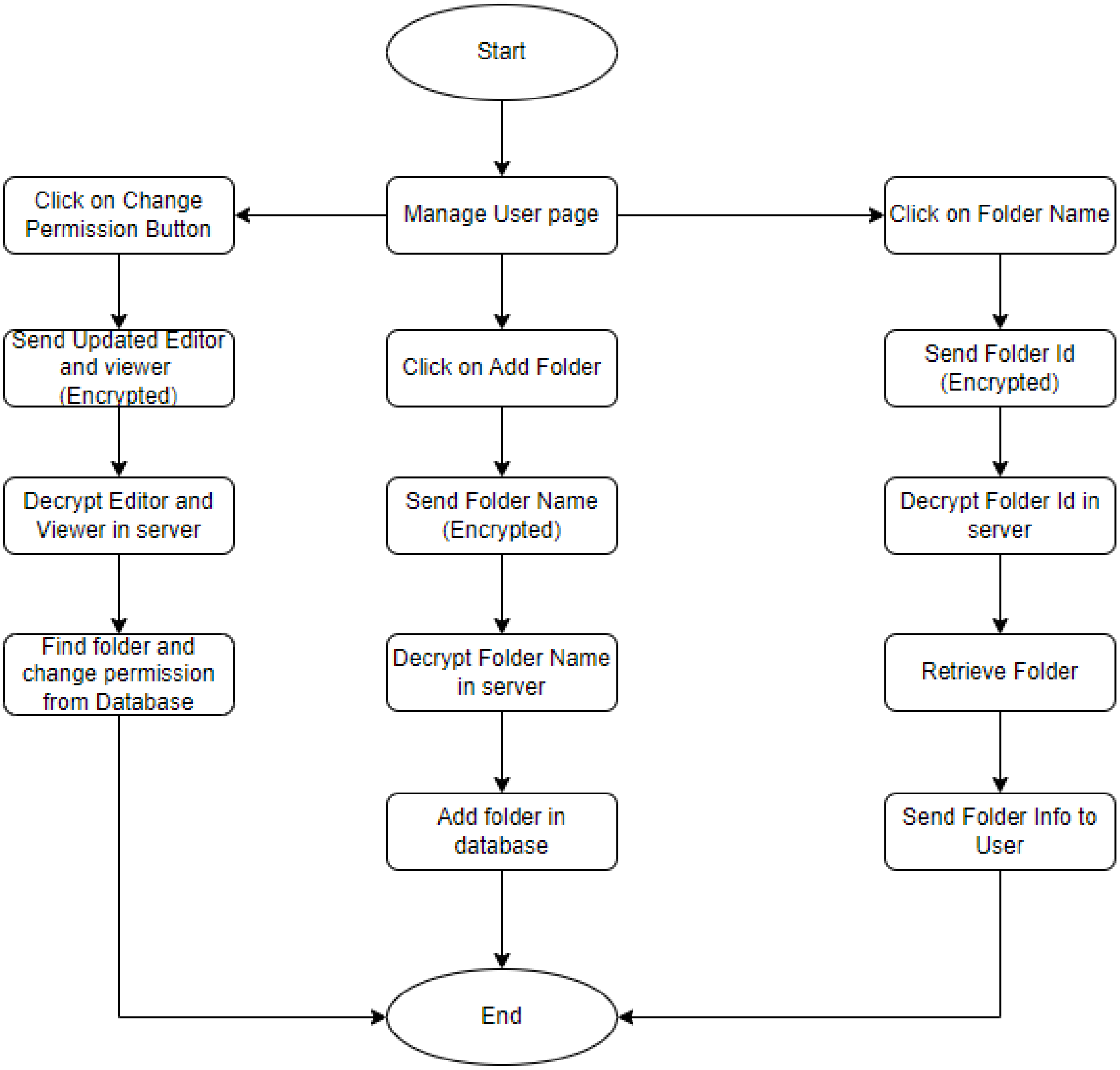

Start
Manage User page
Click on Change Permission Button
Send Updated Editor and viewer (Encrypted)
Decrypt Editor and Viewer in server
Find folder and change permission from Database
Click on Add Folder
Send Folder Name (Encrypted)
Decrypt Folder Name in server
Add folder in database
Click on Folder Name
Send Folder Id (Encrypted)
Decrypt Folder Id in server
Retrieve Folder
Send Folder Info to User
End

Figure 58: Block Diagram Manage Folder

4.10. Manage User

4.10.1. Use Case Diagram

The use case illustrates that the admin can manage the users in the system. The admin can add new user, edit users and delete users. The functionality is illustrated in the use case.

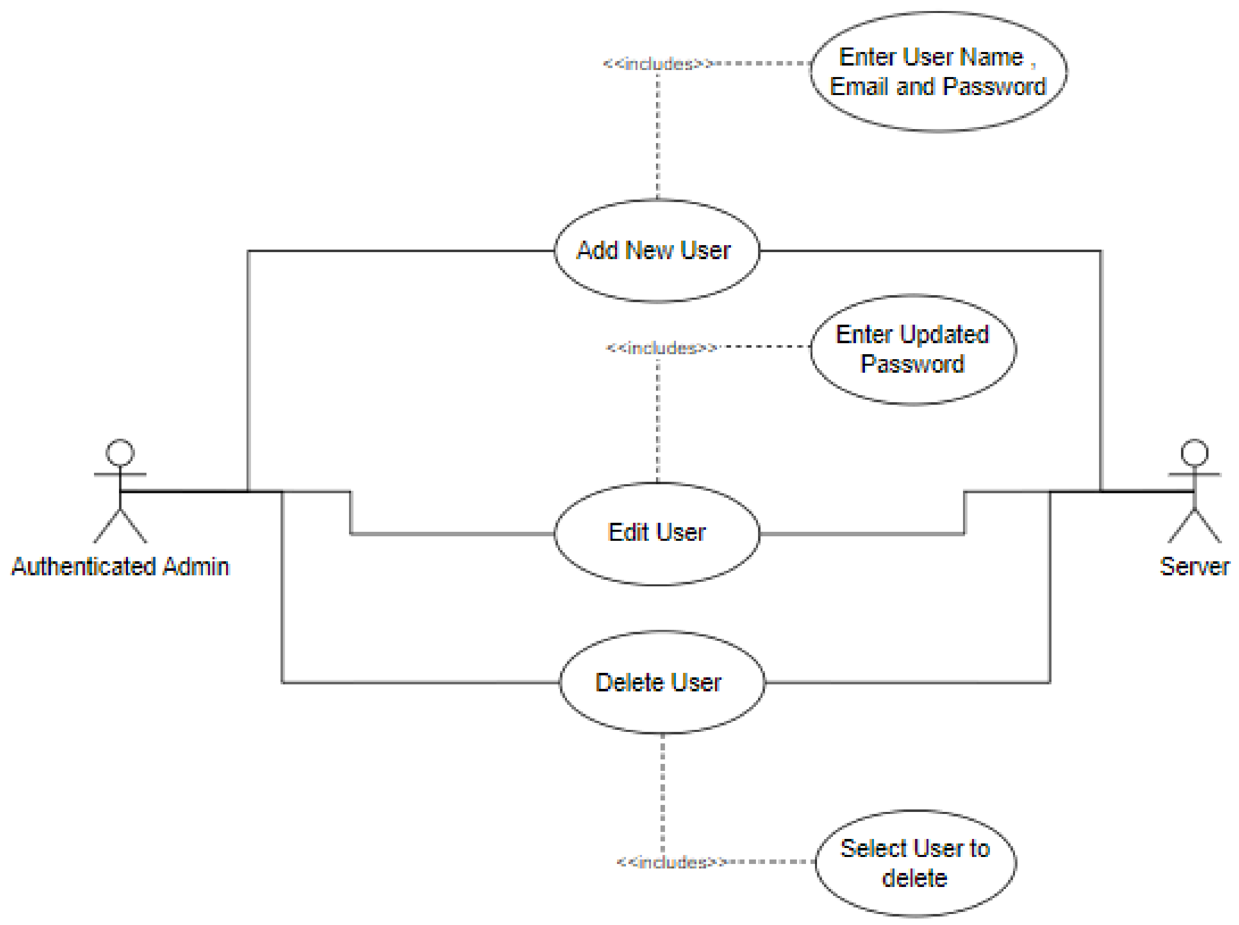


Figure 59: Use Case Diagram Manage User

This use case table outlines the interactions between the user and the system for the Upload File functionality.

Use Case Name	Manage Users
Actor	Admin
Trigger	Admin clicks on Add User button, Edit Icon or Delete Icon.
Precondition	Admin must be logged into their account
Basic Path	1. Admin click on add user button and send user name, email and password in the add user request to the server. 2. Admin click on edit icon and send updated password in the edit user request to the server. 3. Admin click on delete icon and send user id in the delete user request to the server.
Alternative Path	System responds with local feedback, If users email already exists.
Post Condition	Admin can see that users on Manage User page.
Others	System will not add those users, whose email already exists

Table 15: Use Case Table Manage User

4.10.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the admin manage users. All the events of managing user are illustrated in the figure below.

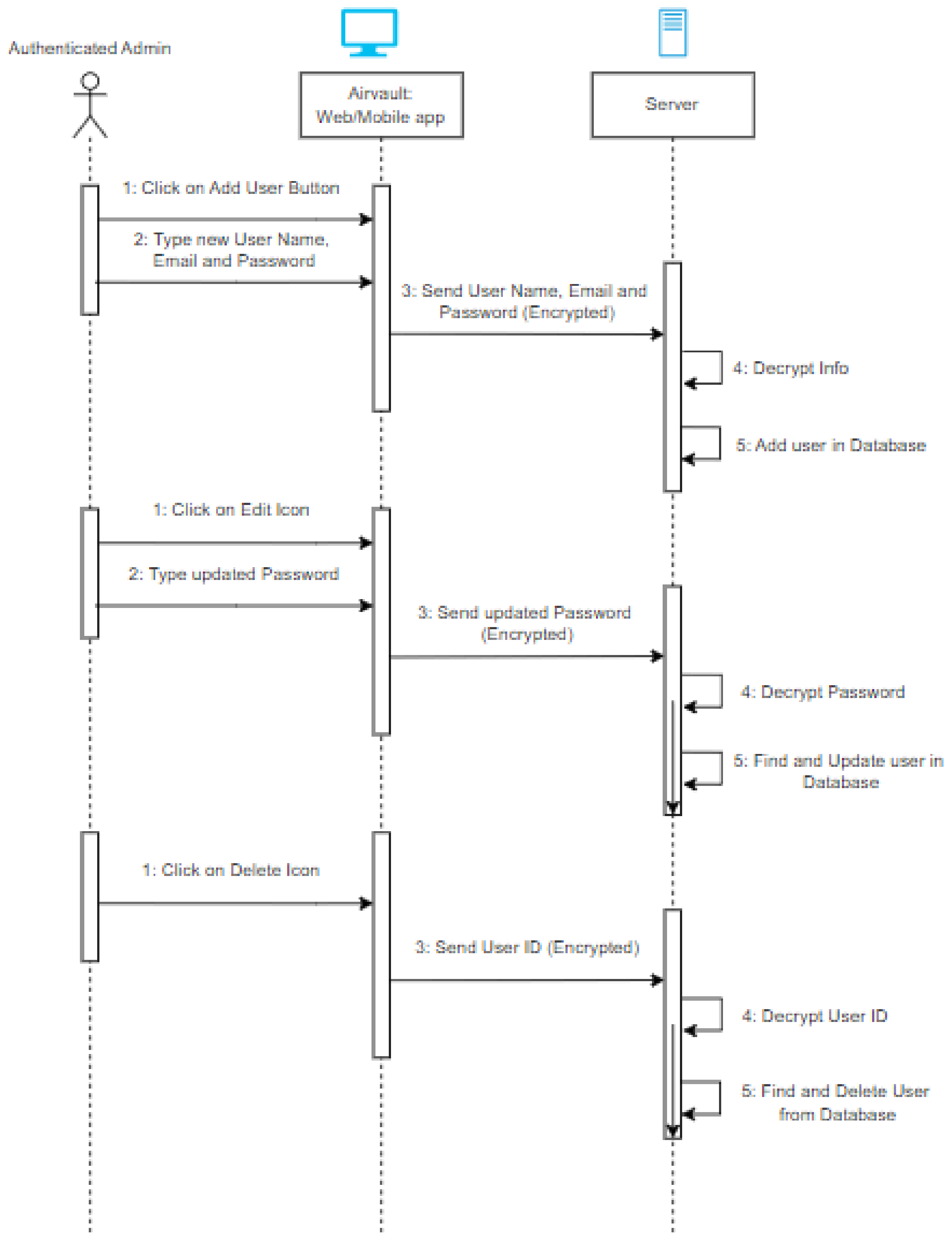

Authenticated Admin
Airvault: Web/Mobile app
Server
1: Click on Add User Button
2: Type new User Name, Email and Password
3: Send User Name, Email and Password (Encrypted)
4: Decrypt Info
5: Add user in Database
1: Click on Edit Icon
2: Type updated Password
3: Send updated Password (Encrypted)
4: Decrypt Password
5: Find and Update user in Database
1: Click on Delete Icon
3: Send User ID (Encrypted)
4: Decrypt User ID
5: Find and Delete User from Database

Figure 60: Sequence Diagram Manage User

4.10.3. Activity Diagram

This activity diagram illustrates the series of activities which will occur at each level of the system as the admin manages users. It shows detailed activities of managing users.

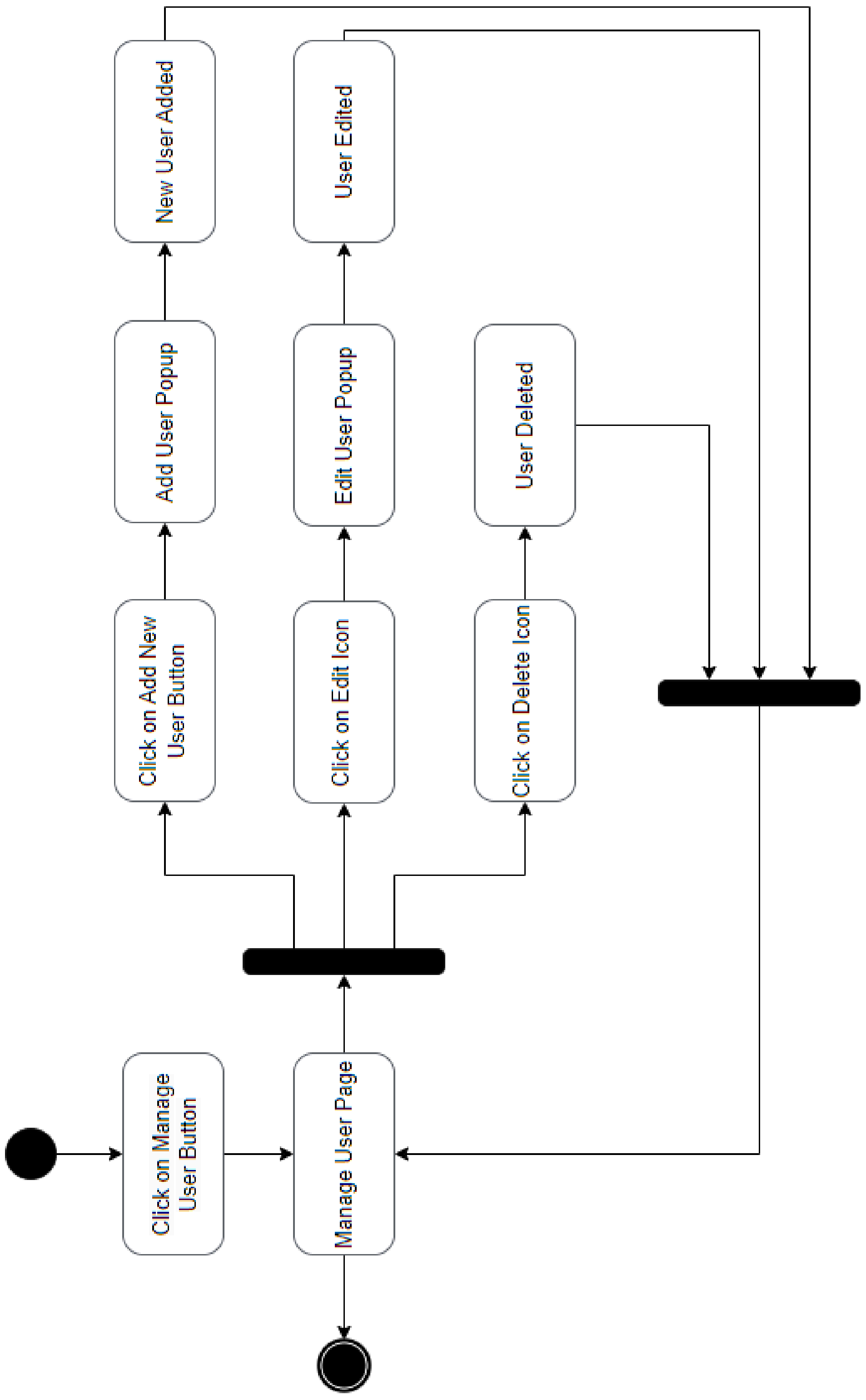


Figure 61: Activity Diagram Manage User

4.10.4. Block Diagram

This block diagram illustrates the series of events that will occur as the admin manages user(s).

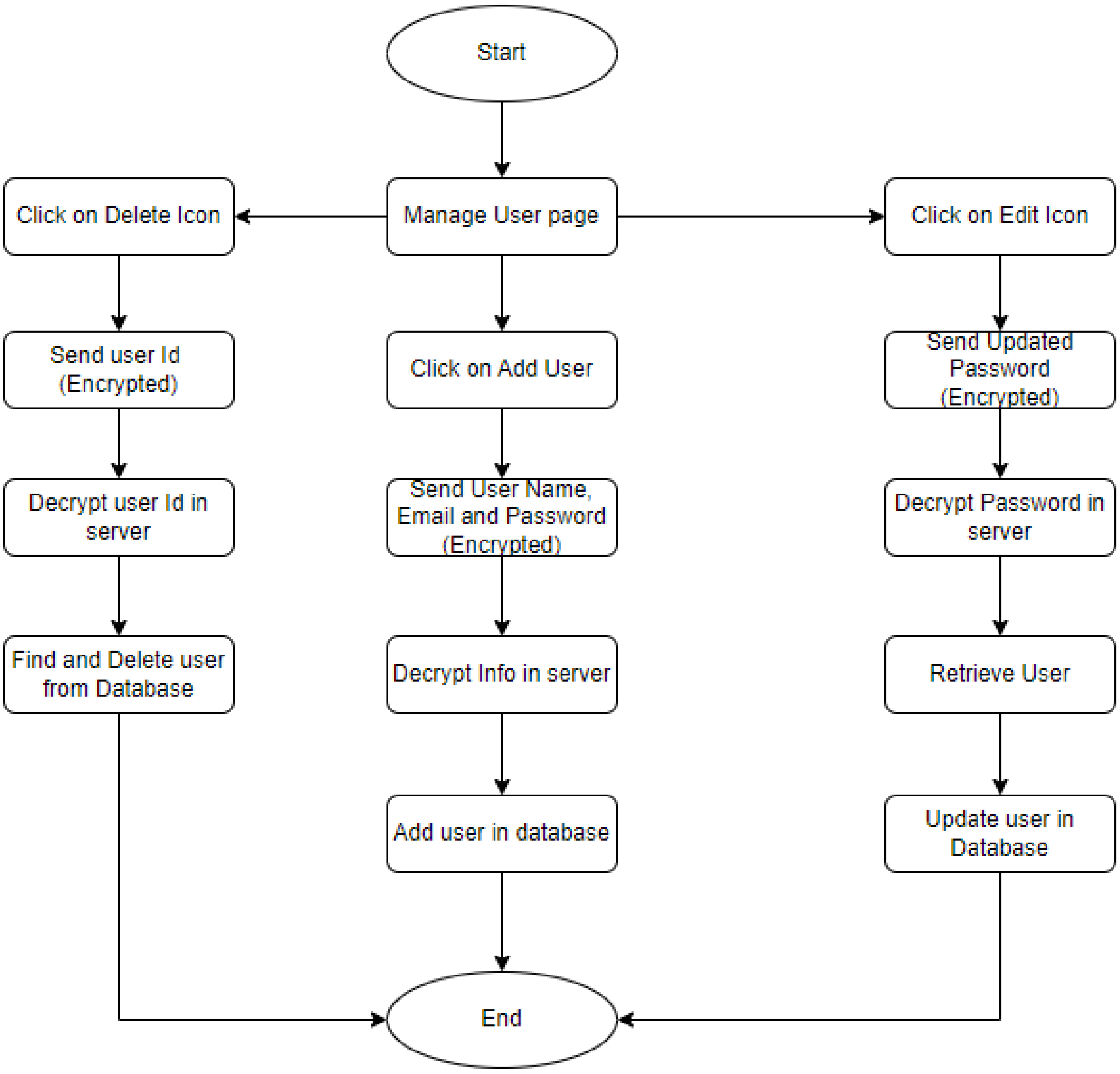


Figure 62: Block Diagram Manage User

4.11. Manage Department

4.11.1. Use Case Diagram

The use case illustrates that the admin can manage the users in the system. The admin can add new departments and edit departments. The functionality is illustrated in the use case.

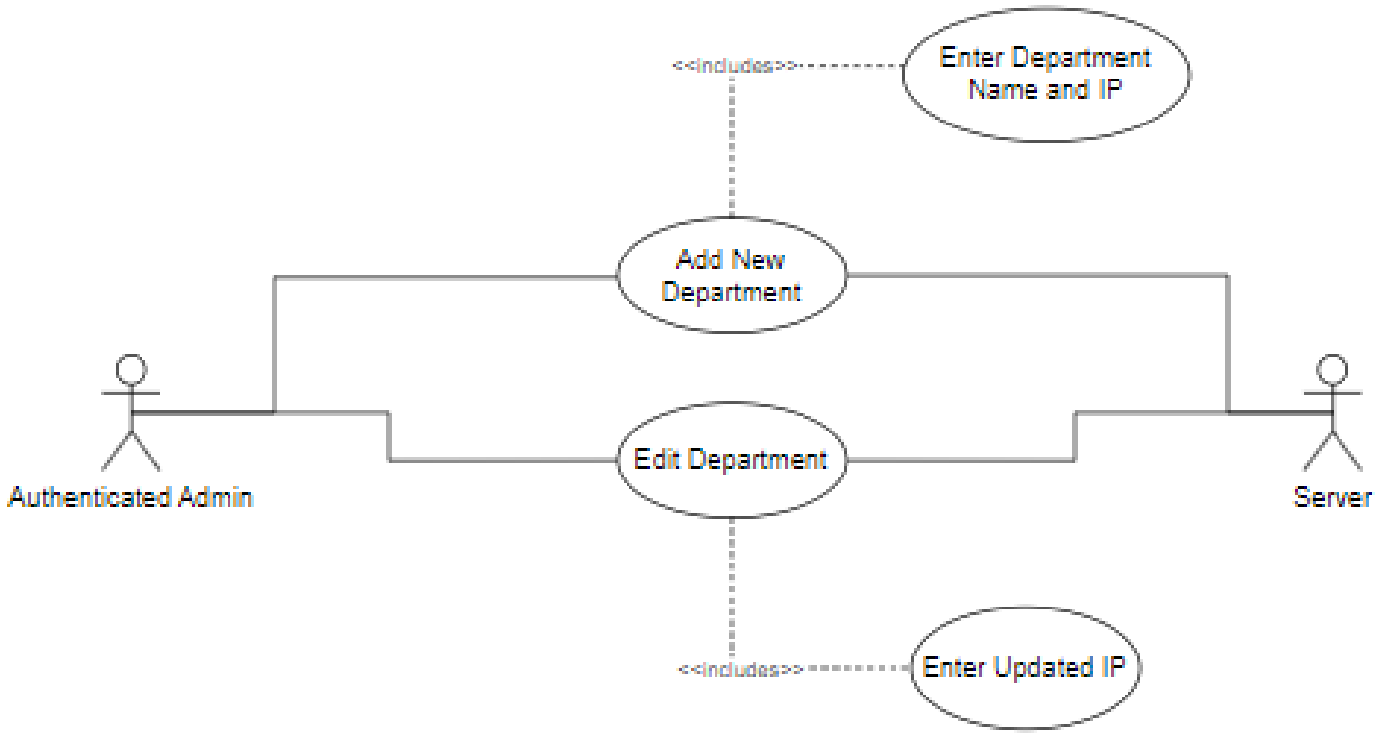


Figure 63: Use Case Diagram Manage Department

This use case table outlines the interactions between the user and the system for the Upload File functionality.

Use Case Name	Manage Department
Actor	Admin
Trigger	Admin clicks on Add Department button and edit icon.
Precondition	Admin must be logged into their account
Basic Path	1. Admin click on add department button and send

	department name and Ip in the add department request to the server. 2. Admin click on edit icon and send updated Ip in the edit department request to the server.
Alternative Path	System responds with local feedback, If department Ip already exists.
Post Condition	Admin can see that department on Manage Department page.
Others	System will not add those department, whose Ip already exists

Table 16: Use Case Table Manage Department

4.11.2. Sequence Diagram

This sequence diagram illustrates the series of events which will occur as the admin manage departments. All the events of managing the department are illustrated in the figure below.

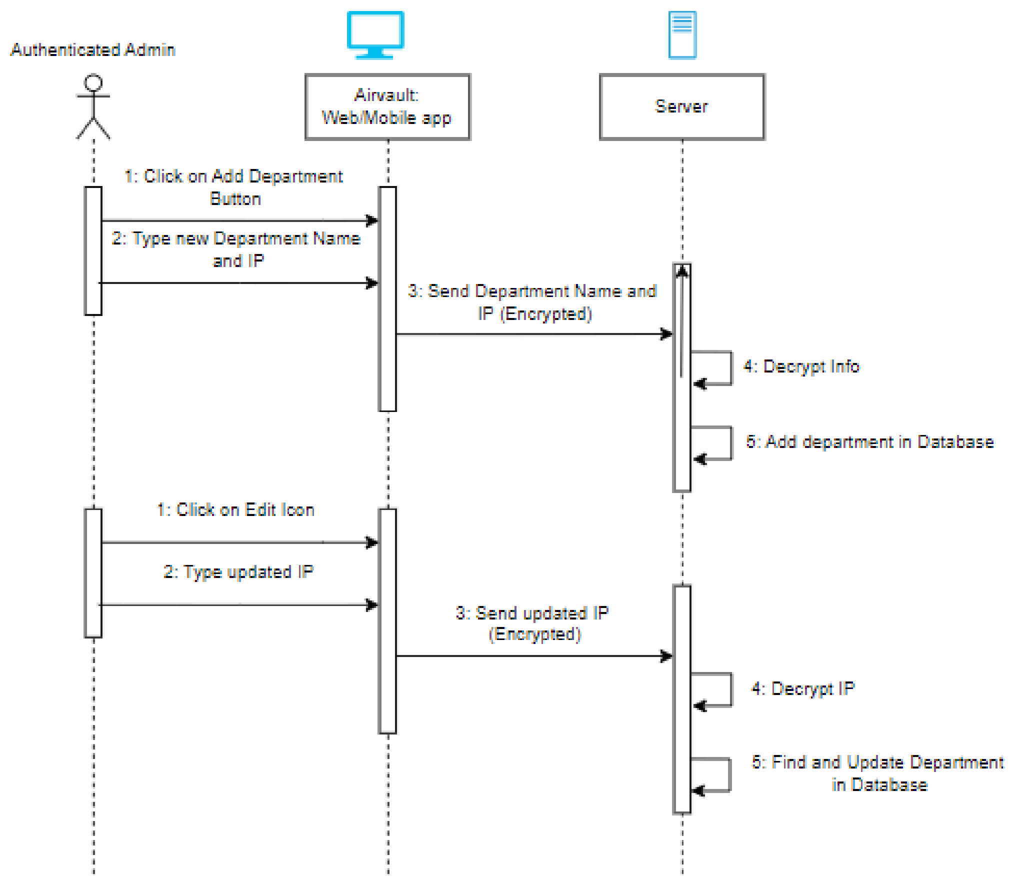

Authenticated Admin
Airvault: Web/Mobile app
Server
1: Click on Add Department Button
2: Type new Department Name and IP
3: Send Department Name and IP (Encrypted)
4: Decrypt Info
5: Add department in Database
1: Click on Edit Icon
2: Type updated IP
3: Send updated IP (Encrypted)
4: Decrypt IP
5: Find and Update Department in Database

Figure 64: Sequence Diagram Manage Department

4.11.3. Activity Diagram

Activity diagram illustrates series of activities which will occur at each level of the system as the admin manages departments. It shows detailed activities of managing departments.

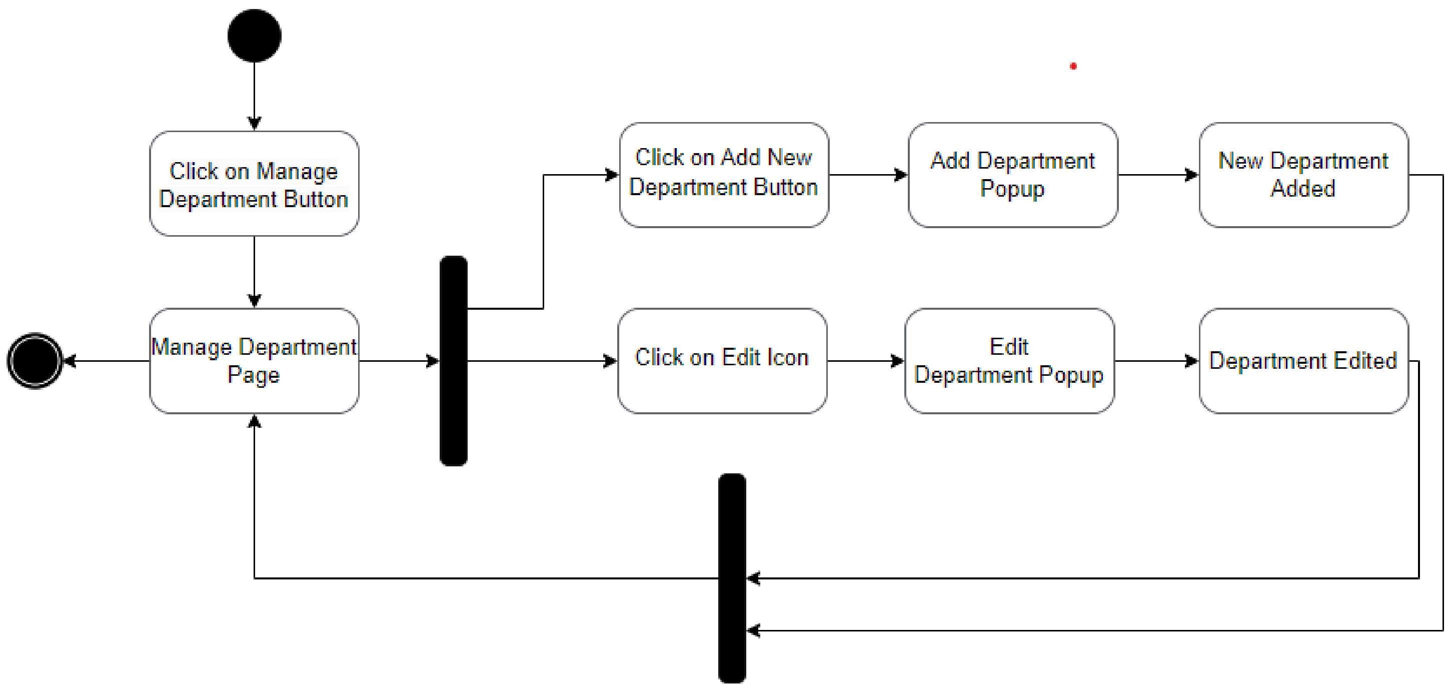


Figure 65: Activity Diagram Manage Department

4.11.4. Block Diagram

Block diagram illustrates series of events that will occur as admin manages department(s)..

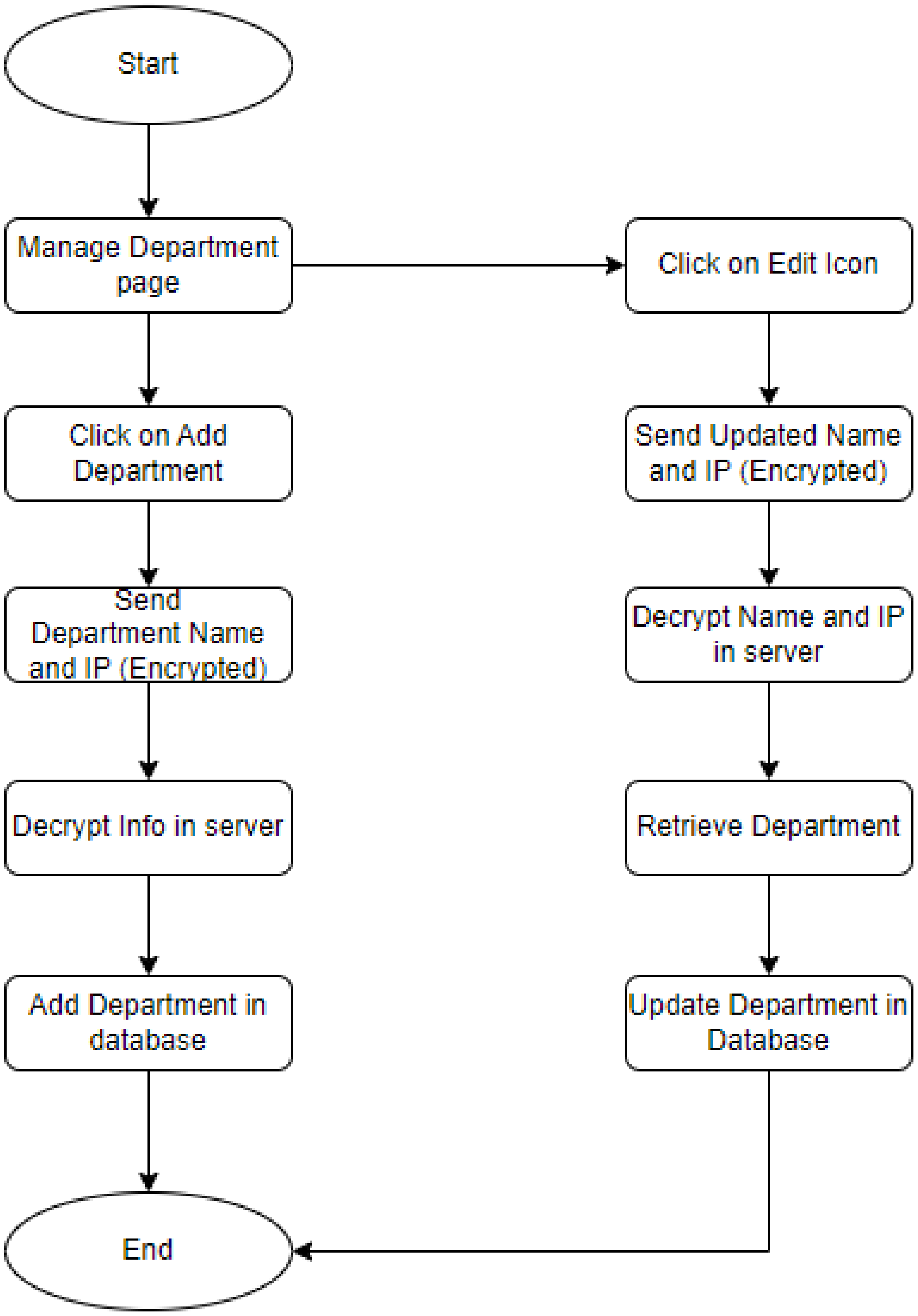

Start
Manage Department page
Click on Add Department
Send Department Name and IP (Encrypted)
Decrypt Info in server
Add Department in database
End
Click on Edit Icon
Send Updated Name and IP (Encrypted)
Decrypt Name and IP in server
Retrieve Department
Update Department in Database

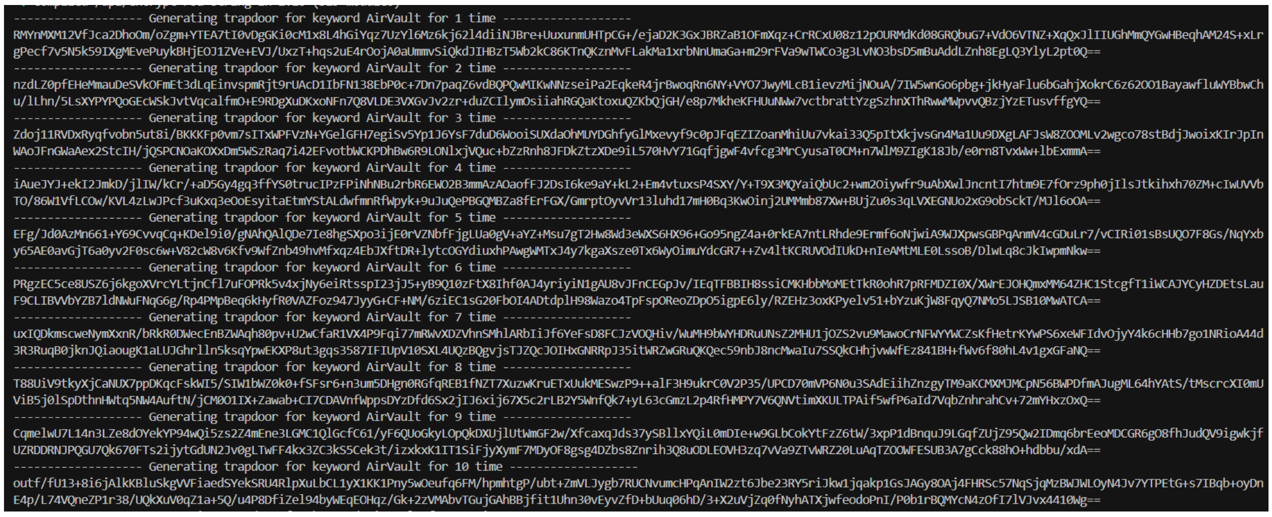
Figure 66: Block Diagram Manage Department

5. Implementation and Testing

In this chapter we have write in detail on generating trapdoors, we also added the results of our performance analysis and it also includes test cases and API specifications.

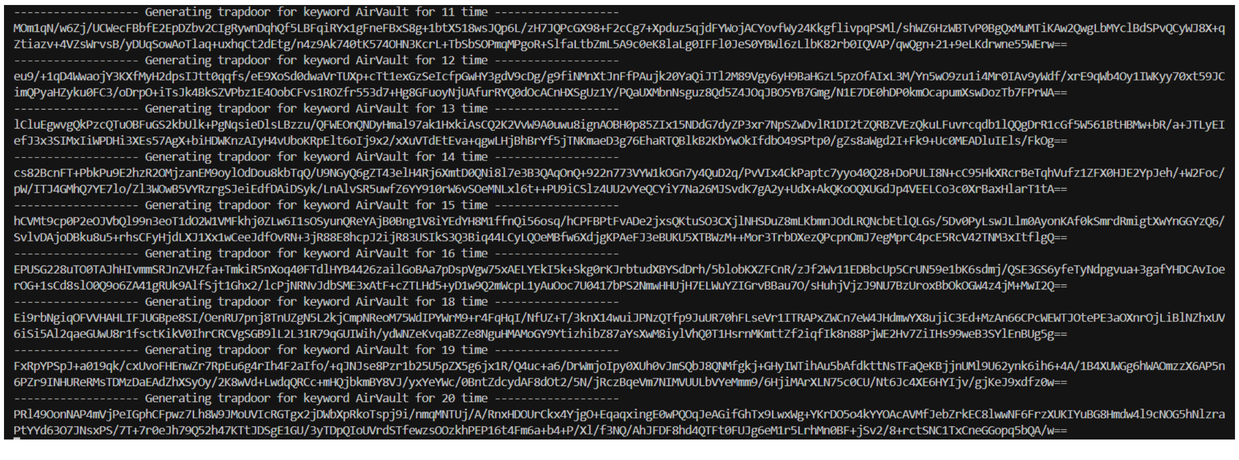
5.1. Keyword Trapdoor Generating

The goal of AirVault is to create a system in which the trapdoor linked to a specific keyword has no obvious connection to the keyword itself. The main goal is to make sure that a trapdoor appears completely different every time it is generated for the same keyword. This method greatly improves the security of implementing AirVault. With regard to the keyword "AirVault," twenty different trapdoors have been created; this illustrates how completely different the trapdoors are from one another even though they share the same keyword.



```
------------------ Generating trapdoor for keyword AirVault for 1 time ------------------
RMYnMXM12VfJca2DhoOm/oZgm+YTEA7tI0vDgGKi0cM1x8L4hGiYqz7UzYl6Mz6kj62l4diiNJBre+UuxunmUHTpCG+/ejaD2K3GxJBRZaB1OFmXqz+CrRCxU08z12pOURMdKd08GRQbuG7+VdO6VTNZ+XqQxJlIIUGhMmQYGwHBeqhAM24S+xLr
gPecf7v5N5k59IXgMEvePuykBHjEOJ1ZVe+EVJ/UxzT+hqs2uE4rOojA0aUmmvSiQkdJIHBzT5Wb2kC86KTnQKznMvFLakMa1xrbNnUmaGa+m29rFVa9wTWCo3g3LvNO3bsD5mBuAddLZnh8EgLQ3YlyL2pt0Q==
------------------ Generating trapdoor for keyword AirVault for 2 time ------------------
nzdLZ0pfEHeMmauDeSVkOFmEt3dLqEinvspmRjt9rUAcD1IbFN138EbP0c+7Dn7paqZ6vdBQPQwMIKwNNzseiPa2EqkeR4jrBwoqRn6NY+VYO7JwyMLcB1ievzMijNOuA/7IW5wnGo6pbg+jkHyaFlu6bGahjXokrC6z62OO1BayawfluWYBbwCh
u/lLhn/5LsXYPYPQoGEcWSkJvtVqcalfmO+E9RDgXuDKxoNFn7Q8VLDE3VXGvJv2zr+duZCIlymOsiiahRGQaKtoxuQZKbQjGH/e8p7MkheKFHUuNWw7vctbrattYzgSzhnXThRwwMWpvvQBzjYzETusvffgYQ==
------------------ Generating trapdoor for keyword AirVault for 3 time ------------------
Zdoj11RVDxRyqfvobn5ut8i/BKKKFp0vm7sITxWPFVzN+YGelGFH7egiSv5Yp1J6YsF7duD6WooiSUXdaOhMUYDGhfyGlMxevyf9c0pJFqEZIZoanMhiUu7vkai33Q5pItXkjvsGn4Ma1Uu9DXgLAFJsW8ZOOMLv2wgco78stBdjJwoixKIrJpIn
WAoJFnGWaAex2StcIH/jQSPCNOaKOXxDm5WSzRaq7i42EFvotbWCKPDhBw6R9LONlxjVQuc+bZzRnh8JFDkZtzXDe9iL570HvY71GqfjgwF4vfcg3MrCyusaT0CM+n7WlM9ZIgK18Jb/e0rn8TvxWw+lbExmmA==
------------------ Generating trapdoor for keyword AirVault for 4 time ------------------
iAueJYJ+ekI2JmkD/jlIW/kCr/+aD5Gy4gq3ffYS0trucIPzFPiNhNBu2rbR6EWO2B3mmAzAOaofFJ2DsI6ke9aY+kL2+Em4vtuxsP4SXY/Y+T9X3MQYaiQbUc2+wm2Oiywfr9uAbXwlJncntI7htm9E7fOrz9ph0jIlsJtkihxh70ZM+cIwUVVb
TO/86W1VfLCOw/KVL4zLwJPcf3uKxq3eOoEsyitaEtmYStALdwfmnRfWpyk+9uJuQePBGQMBZa8fErFGX/GmrptOyvVr13luhd17mH0Bq3KwOinj2UMMmb87Xw+BUjZu0s3qLVXEGNUo2xG9obSckT/MJl6oOA==
------------------ Generating trapdoor for keyword AirVault for 5 time ------------------
EFg/Jd0AzMn661+Y69CvvqCq+KDel9i0/gNAhQAlQDe7Ie8hgSXpo3ijE0rVZNbfFjgLUa0gV+aYZ+Msu7gT2Hw8Wd3eWXS6HX96+Go95ngZ4a+0rkEA7ntLRhde9Ermf6oNjwiA9WJXpwsGBPqAnmV4cGDuLr7/vCIRi01sBsUQO7F8Gs/NqYxb
y65AE0avGjT6a0yv2F0sc6w+V82cW8v6Kfv9WfZnb49hvMfxqz4EbJXftDR+lytcOGYdiuxhPAwgWMTxJ4y7kgaXsze0Tx6WyOimuYdcGR7++Zv4ltKCRUVOdIUkD+nIeAMtMLE0LssoB/DlwLq8cJkIwpmNkw==
------------------ Generating trapdoor for keyword AirVault for 6 time ------------------
PRgzEC5ce8USZ6j6kgoXVrcYLtjnCfl7uFOPRk5v4xjNy6eiRtsspI23jJ5+yB9Q10zFtX8Ihf0AJ4yriyiN1gAU8vJFnCEGpJv/IEqTFBBIH8ssiCMKHbbMoMEtTkR0ohR7pRFMDZI0X/XWrEJOHQmxMM64ZHC1StcgfT1iWCAJYCyHZDEtsLau
F9CLIBVVbYZB7ldNWuFNqG6g/Rp4PMpBeq6kHyfR0VAZFoz947JyyG+CF+NM/6ziEC1sG20FbOI4ADtdplH98Wazo4TpFspOReoZDpO5igpE6ly/RZEHz3oxKPyelv51+bYzuKjW8FqyQ7NMo5LJSB10MWATCA==
------------------ Generating trapdoor for keyword AirVault for 7 time ------------------
uxIQDkmscweNymXxnR/bRkR0DWecEnBZWAqh80pv+U2wCfaR1VX4P9Fqi77mRWvXDZVhnSMhlARbIiJf6YeFsD8FCJzVOQHiv/WuMH9bWYHDRuUNsZ2MHU1jOZS2vu9MawoCrNFWYYWCZsKfHetrKYwPS6xeWFIdvOjyY4k6cHHb7go1NRioA44d
3R3RuqB0jknJQiaougK1aLUJGhrlln5ksqYpwEKXP8ut3gqs3587IFIUpV10SXL4UQzBQgvjsTJZQcJOIHxGNRRpJ35itWRZwGRuQKQec59nbJ8ncMwaIu7SSQkCHhjvwWfEz841BH+fWv6f80hL4v1gxGFaNQ==
------------------ Generating trapdoor for keyword AirVault for 8 time ------------------
T88UiV9tkyXjCaNUX7ppDKqcFskWI5/SIW1bWZ0k0+fSFsr6+n3um5DHgn0RGfqREB1fNZT7XuzwKruETxUukMESwzP9++alF3H9ukrC0V2P35/UPCD70mVP6N0u3SAdEiihZnzgyTM9aKCMXMJMCpN56BWPDfmAJugML64hYAtS/tMscrcXI0mU
ViB5j0lSpDthnHWtq5NW4AuftN/jCM0O1IX+Zawab+CI7CDAVnfWppsDYzDfd6Sx2jIJ6xij67X5c2rLB2Y5WnfQk7+yL63cGmzL2p4RfHMPY7V6QNVtimXKULTPAif5wfP6aId7VqbZnhrahCv+72mYHxzOxQ==
------------------ Generating trapdoor for keyword AirVault for 9 time ------------------
CqmelwU7L14n3LZe8dOYekYP94wQi5zs2Z4mEne3LGMC1QlGcfC61/yF6QUoGkyLOpQkDXUjlUtWmGF2w/XfcaxqJds37ySBllxYQiL0mDIe+w9GLbCokYtFzZ6tW/3xpP1dBnquJ9LGqfZUjZ95Qw2IDmq6brEeoMDCGR6gO8fhJudQV9igwkjf
UZRDDRNJPQGU7Qk670FTs2ijytGdUN2Jv0gLTwFF4kx3ZC3kS5Cek3t/izxkxK1IT1SiFjyXymF7MDyOF8gsg4DZbs8Znrih3Q8uODLEOVH3zq7vVa9ZTvWRZ20LuAqTZOOWFESUB3A7gCck88hO+hdbbu/xdA==
------------------ Generating trapdoor for keyword AirVault for 10 time ------------------
outf/fU13+8i6jAlkKBluSkgVVFiaedSYekSRU4RlpXuLbCL1yX1KK1Pny5wOeufq6FM/hpmhtgP/ubt+ZmVLJygb7RUCNvumcHPqAnIW2zt6Jbe23RY5riJkw1jqakp1GsJAGy8OAj4FHRSc57NqSjqMzBWJWLOyN4Jv7YTPEtG+s7IBqb+oyDn
E4p/L74VQneZP1r38/UQkXuV0qZ1a+5Q/u4P8DfiZel94byWEqEOHqz/Gk+2zVMAbvTGujGAhBBjfit1Uhn30vEyvZfD+bUuq06hD/3+X2uVjZq0fNyhATXjwfeodoPnI/P0b1rBQMYcN4zOfI7lVJvx4410Wg==
```

Figure 67: Trapdoor generation 1



```
------------------ Generating trapdoor for keyword AirVault for 11 time ------------------
MOm1qN/w6Zj/UCWecFBbfE2EpDZbv2CIgRywnDqhQf5LBFqiRYx1gFneFBxS8g+1btX518wsJQp6L/zH7JQPcGX98+F2cCg7+Xpduz5qjdFYWojACYovfWy24KkgflivpqPSMl/shWZ6HzWBTvP0BgQxMuMTiKAw2QwgLbMYclBdSPvQCyWJ8X+q
Ztiazv+4VZsWrvsB/yDUqSowAoTlaq+uxhqCt2dEtg/n4z9Ak740tK574OHN3KcrL+TbSbSOPmqMPgoR+SlfaLtbZmL5A9c0eK8laLg0IFFl0JeS0YBWl6zLlbK82rb0IQVAP/qwQgn+21+9eLKdrwne55WErw==
------------------ Generating trapdoor for keyword AirVault for 12 time ------------------
eu9/+1qD4WwaojY3KXfMyH2dpsIJtt0qqfs/eE9XoSd0dwaVrTUXp+cTt1exGzSeIcfpGwHY3gdV9cDg/g9fiNMnXtJnFfPAujk20YaQiJTl2M89Vgy6yH9BaHGzL5pzOfAIxL3M/Yn5wO9zu1i4Mr0IAv9yWdf/xrE9qWb4Oy1IWKyy70xt59JC
imQPyaHZyku0FC3/oDrpO+iTsJk4BkSZVPbz1E4OobCFvs1ROZfr553d7+Hg8GFuoyNjUAfurRYQ0dOcACnHXSgUz1Y/PQaUXMbnNsguz8Qd5Z4JOqJBO5YB7Gmg/N1E7DE0hDP0kmOcapumXswDozTb7FPrWA==
------------------ Generating trapdoor for keyword AirVault for 13 time ------------------
lCluEgwvgQkPzcQTuOBFuGS2kbUlk+PgNqsieDlsLBzzu/QFWEOnQNDyHmal97ak1HxkiAsCQ2K2VvW9A0uwu8ignAOBH0p85ZIx15NDdG7dyZP3xr7NpSZwDvlR1DI2tZQRBZVEzQkuLFuvrcqdb1lQQgDrR1cGf5W561BtHBMw+bR/a+JTLyEI
efJ3x3SIMxIiWPDHi3XEs57AgX+biHDWKnzAIyH4vUboKRpElt6oIj9x2/xXuVTdEtEva+qgwLHjBhBrYf5jTNKmaeD3g76EhaRTQBlkB2KbYwOkIfdbO49SPtp0/gZs8aWgd2I+Fk9+Uc0MEADluIEls/FkOg==
------------------ Generating trapdoor for keyword AirVault for 14 time ------------------
cs82BcnFT+PbkPu9E2hzR2OMjzanEM9oylOdDou8kbTqQ/U9NGyQ6gZT43elH4Rj6XmtD0QNi8l7e3B3QAqOnQ+922n773VYW1kOGn7y4QuD2q/PvVIx4CkPaptc7yyo40Q28+DoPULI8N+cC95HkXRcrBeTqhVufz1ZFX0HJE2YpJeh/+W2Foc/
pW/ITJ4GMhQ7YE7lo/Zl3WOwB5VYRzrgSJeiEdfDAiDSyk/LnAlvSR5uwfZ6YY910rW6vSOeMNLxl6t++PU9iCSlz4UU2vYeQCYiY7Na26MJSvdK7gA2y+UdX+AkQKoOQXUGdJp4VEELCo3c0XrBaxHlarT1tA==
------------------ Generating trapdoor for keyword AirVault for 15 time ------------------
hCVMt9cp0P2eOJVbQl99n3eoT1dO2W1VMFkhj0ZLw6I1sOSyunQReYAjB0Bng1V8iYEdYH8M1ffnQi56osq/hCPFBPtFvADe2jxsQKtuSO3CXjlNHSDuZ8mLKbmnJOdLRQNcbEtlQLGs/5Dv0PyLswJLlm0AyonKAf0kSmrdRmigtXwYnGGYzQ6/
SvlvDAjoDBku8u5+rhsCFyHjdLXJ1Xx1wCeeJdfOvRN+3jR88E8hcpJ2ijR83USIkS3Q3Biq44LCyLQOeMBfw6XdjgKPAeFJ3eBUKU5XTBWzM++Mor3TrbDXezQPcpnOmJ7egMprC4pcE5RcV42TNM3xItflgQ==
------------------ Generating trapdoor for keyword AirVault for 16 time ------------------
EPUSG228uTO0TAJhHIvmmSRJnZVHZfa+TmkiR5nXoq40FTdlHYB4426zailGoBAa7pDspVgw75xAELYEkI5k+Skg0rKJrbtudXBYSdDrh/5blobKXZFCnR/zJf2Wv11EDBbcUp5CrUN59e1bK6sdmj/QSE3GS6yfeTyNdpgvua+3gafYHDCAvIoe
rOG+1sCd8slO0Q9o6ZA41gRUk9AlfSjt1Ghx2/lcPjNRNvJdbSME3xAtF+cZTLHd5+yD1w9Q2mWcpL1yAuOoc7U0417bPS2NmwHHUjH7ELWuYZIGrvBBau7O/sHuhjVjzJ9NU7BzUroxBbOkOGW4z4jM+MwI2Q==
------------------ Generating trapdoor for keyword AirVault for 18 time ------------------
Ei9rbNgiqOFVVHAHLIFJUGBpe8SI/OenRU7pnj8TnUZgN5L2kjCmpNReoM75WdIPYWrM9+r4FqHqI/NfUZ+T/3knX14wuiJPNzQTfp9JuUR70hFLseVr1ITRAPxZWCn7eW4JHdmwYX8ujiC3Ed+MzAn66CPcWEWTJOtePE3aOXnrOjLiBlNZhxUV
6iSi5Al2qaeGUwU8r1fsctKikV0IhrCRCVgSGB9lL2L31R79qGUIWih/ydWNZeKvqaBZZe8NguHMAMoGY9YtizhibZ87aYsXwM8iylVhQ0T1HsrnMKmttZf2iqfIk8n88PjWE2Hv7ZiIHs99weB3SYlEnBUg5g==
------------------ Generating trapdoor for keyword AirVault for 19 time ------------------
FxRpYPSpJ+a019qk/cxUvoFHEnwZr7RpEu6g4rIh4F2aIfo/+qJNJse8Pzr1b25U5pZX5g6jx1R/Q4uc+a6/DrWmjoIpy0XUh0vJmSQbJ8QNMfgkj+GHyIWTihAu5bAfdkttNsTFaQeKBjjnUMl9U62ynk6ih6+4A/1B4XUWGg6hWAOmzzX6AP5n
6PZr9INHUReRMsTDMzDaEAdZhXSyOy/2K8wVd+LwdqQRCc+mHQjbkmBY8VJ/yxYeYWc/0BntZdcydAF8dOt2/5N/jRczBqeVm7NIMVUULbVYeMmm9/6HjiMArXLN75c0CU/Nt6Jc4XE6HYIjv/gjKeJ9xdfz0w==
------------------ Generating trapdoor for keyword AirVault for 20 time ------------------
PRl49OonNAP4mVjPeIGphCFpwz7Lh8W9JMoUVIcRGTgx2jDwbXpRkoTspj9i/nmqMNTUj/A/RnxHDOUrCkx4YjgO+EqaqxingE0wPQOqJeAGifGhTx9LwxWg+YKrDO5o4kYYOAcAVMfJebZrkEC8lwwNF6FrzXUKIYuBG8Hmdw4l9cNOG5hNlzra
PtYYd63O7JNsxPS/7T+7r0eJh79Q52h47KTtJDSgE1GU/3yTDpQIoUVrdSTfewzsOOzkhPEP16t4Fm6a+b4+P/Xl/f3NQ/AhJFDF8hd4QTFt0FUJg6eM1r5LrhMn0BF+jSv2/8+rctSNC1TxCneGGopq5bQA/w==
```

Figure 68: Trapdoor generation 2

5.2. Performance Analysis

This section presents performance analysis of AirVault, Performance analysis is done on the basis of time system took to complete a task.

5.2.1. Upload Files

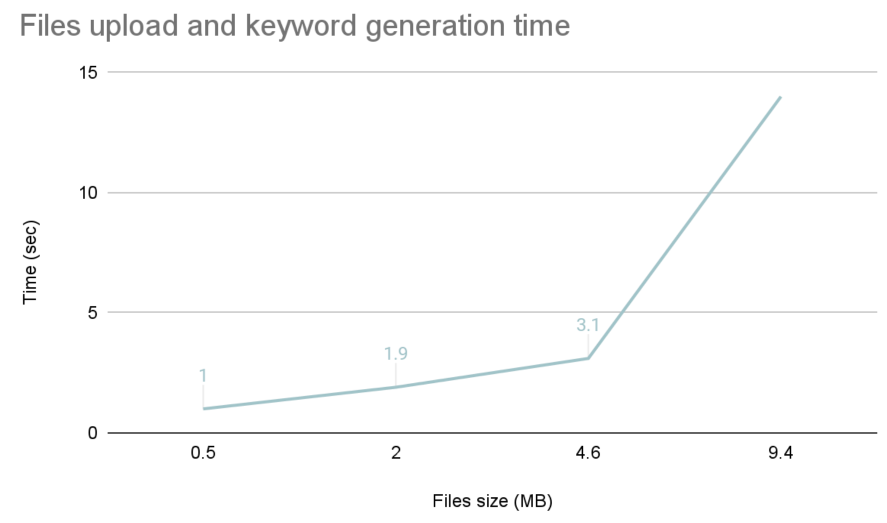


Figure 69:Files Upload and keyword generation time

5.2.2. Download File

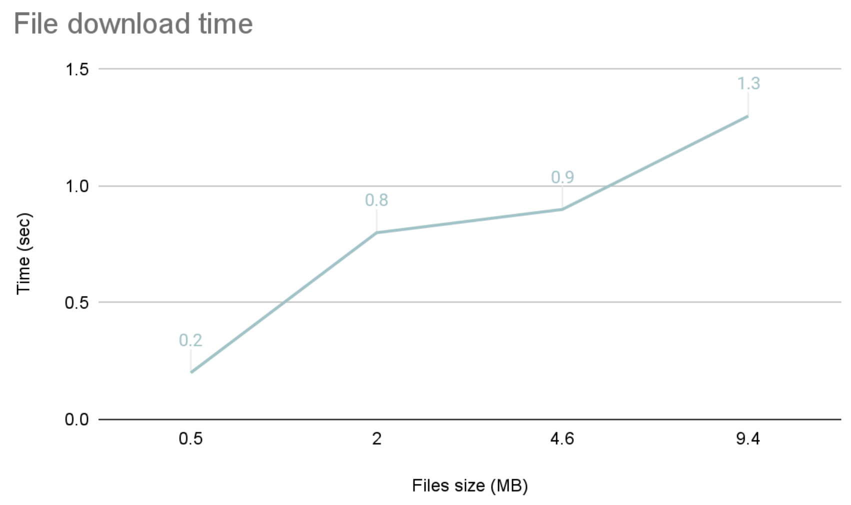


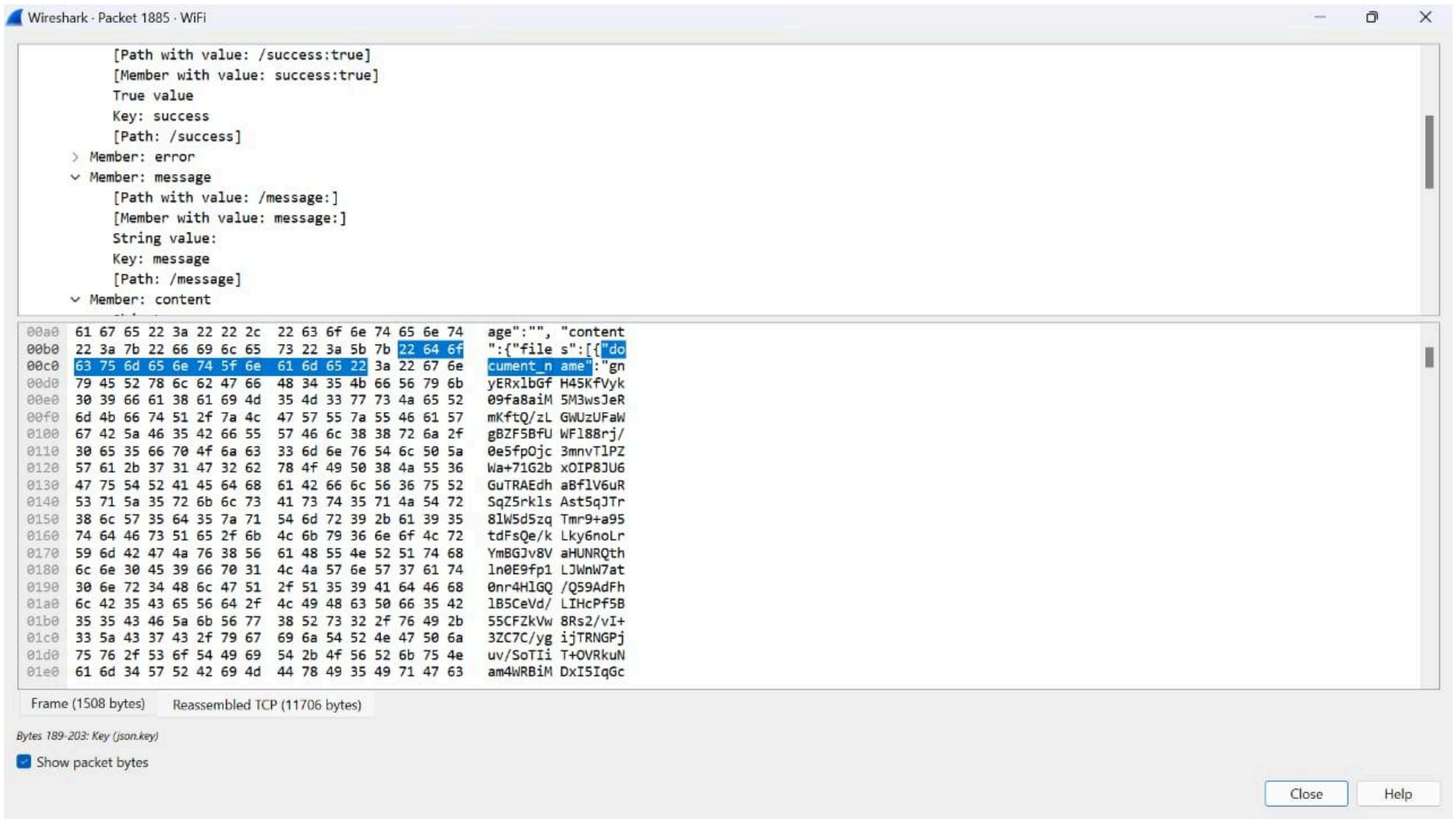
Figure 70: File Download time graph

5.2.3. Search Keyword

It took 0.75 seconds to accurately search for a keyword across all keywords in our inverted index table and display the relevant results.

5.3. Encrypted data flow

Here are attached screenshots from wireshark which shows that all our data is traveling in encrypted form over the network for a more secure channel to transfer sensitive information.

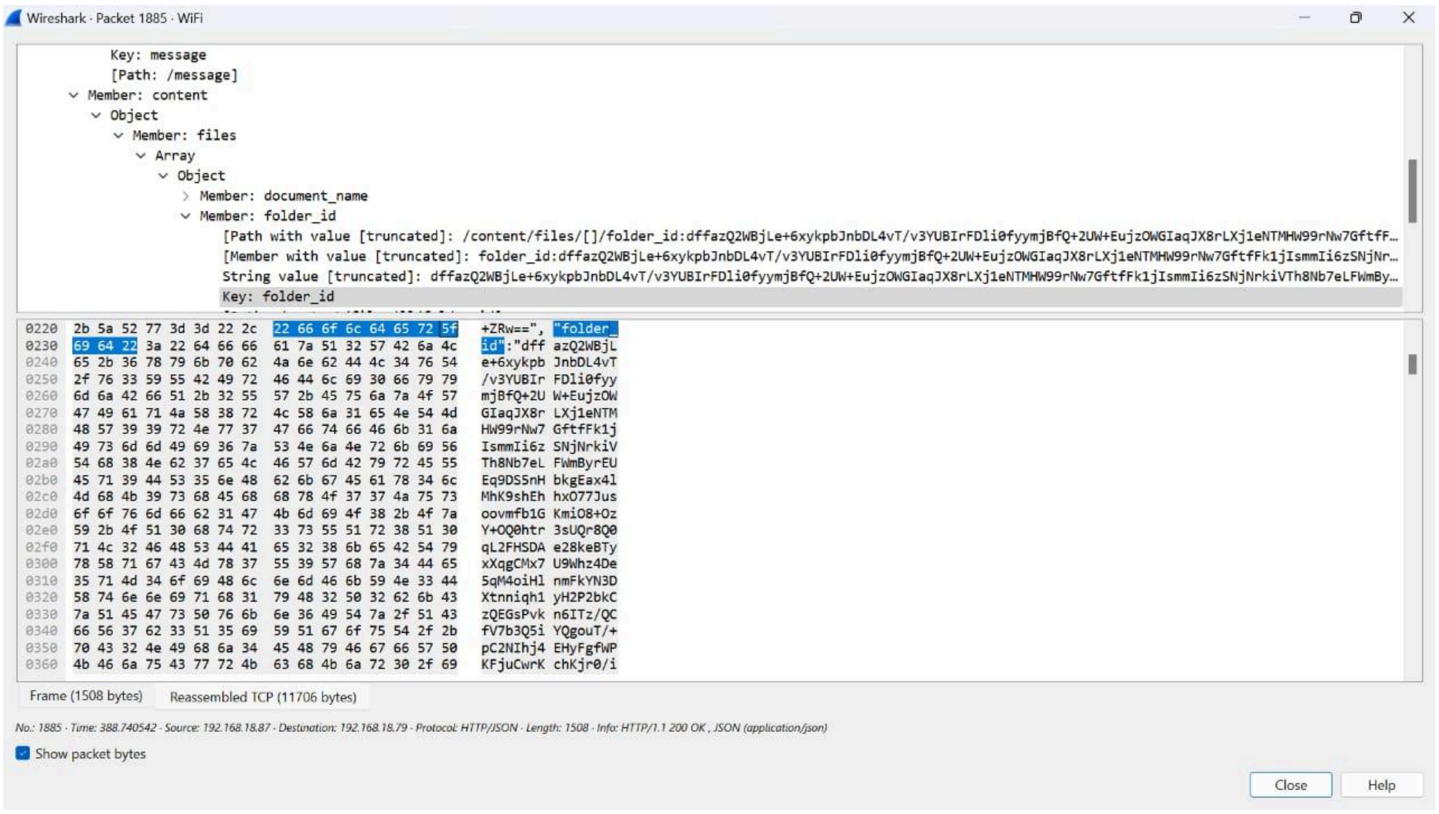

Wireshark · Packet 1885 · WiFi

```
        [Path with value: /success:true]
        [Member with value: success:true]
        True value
        Key: success
        [Path: /success]
    > Member: error
    ˅ Member: message
        [Path with value: /message:]
        [Member with value: message:]
        String value:
        Key: message
        [Path: /message]
    ˅ Member: content
```

```
00a0  61 67 65 22 3a 22 22 2c  22 63 6f 6e 74 65 6e 74   age":"", "content
00b0  22 3a 7b 22 66 69 6c 65  73 22 3a 5b 7b 22 64 6f   ":{"file s":[{"do
00c0  63 75 6d 65 6e 74 5f 6e  61 6d 65 22 3a 22 67 6e   cument_n ame":"gn
00d0  79 45 52 78 6c 62 47 66  48 34 35 4b 66 56 79 6b   yERxlbGf H45KfVyk
00e0  30 39 66 61 38 61 69 4d  35 4d 33 77 73 4a 65 52   09fa8aiM 5M3wsJeR
00f0  6d 4b 66 74 51 2f 7a 4c  47 57 55 7a 55 46 61 57   mKftQ/zL GWUzUFaW
0100  67 42 5a 46 35 42 66 55  57 46 6c 38 38 72 6a 2f   gBZF5BfU WFl88rj/
0110  30 65 35 66 70 4f 6a 63  33 6d 6e 76 54 6c 50 5a   0e5fpOjc 3mnvTlPZ
0120  57 61 2b 37 31 47 32 62  78 4f 49 50 38 4a 55 36   Wa+71G2b xOIP8JU6
0130  47 75 54 52 41 45 64 68  61 42 66 6c 56 36 75 52   GuTRAEdh aBflV6uR
0140  53 71 5a 35 72 6b 6c 73  41 73 74 35 71 4a 54 72   SqZ5rkls Ast5qJTr
0150  38 6c 57 35 64 35 7a 71  54 6d 72 39 2b 61 39 35   8lW5d5zq Tmr9+a95
0160  74 64 46 73 51 65 2f 6b  4c 6b 79 36 6e 6f 4c 72   tdFsQe/k Lky6noLr
0170  59 6d 42 47 4a 76 38 56  61 48 55 4e 52 51 74 68   YmBGJv8V aHUNRQth
0180  6c 6e 30 45 39 66 70 31  4c 4a 57 6e 57 37 61 74   ln0E9fp1 LJWnW7at
0190  30 6e 72 34 48 6c 47 51  2f 51 35 39 41 64 46 68   0nr4HlGQ /Q59AdFh
01a0  6c 42 35 43 65 56 64 2f  4c 49 48 63 50 66 35 42   lB5CeVd/ LIHcPf5B
01b0  35 35 43 46 5a 6b 56 77  38 52 73 32 2f 76 49 2b   55CFZkVw 8Rs2/vI+
01c0  33 5a 43 37 43 2f 79 67  69 6a 54 52 4e 47 50 6a   3ZC7C/yg ijTRNGPj
01d0  75 76 2f 53 6f 54 49 69  54 2b 4f 56 52 6b 75 4e   uv/SoTIi T+OVRkuN
01e0  61 6d 34 57 52 42 69 4d  44 78 49 35 49 71 47 63   am4WRBiM DxI5IqGc
```

Frame (1508 bytes) Reassembled TCP (11706 bytes)

Bytes 189-203: Key (json.key)



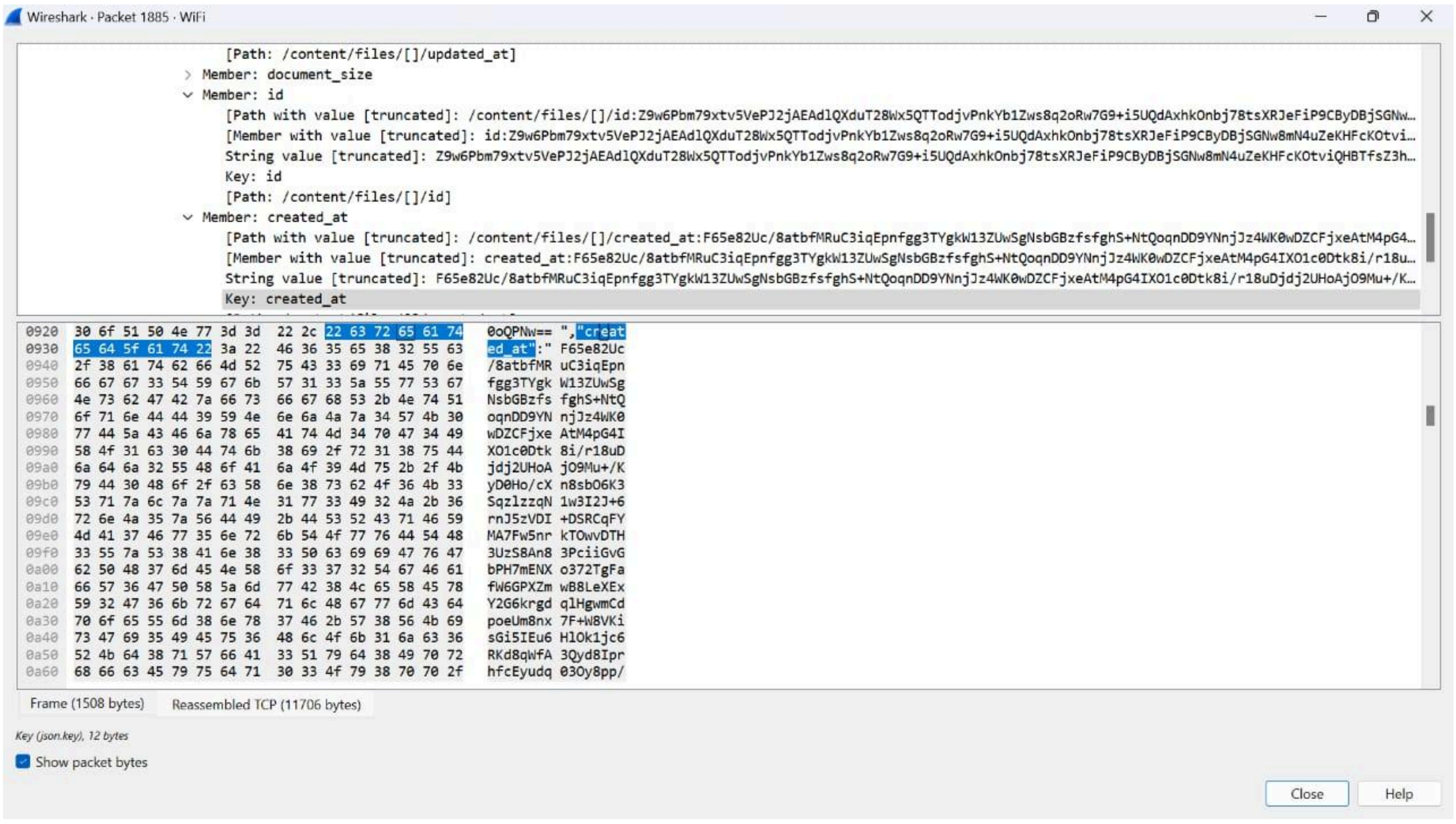
```
Key: message
[Path: /message]
Member: content
  Object
    Member: files
      Array
        Object
          Member: document_name
          Member: folder_id
            [Path with value [truncated]: /content/files/[]/folder_id:dffazQ2WBjLe+6xykpbJnbDL4vT/v3YUBIrFDli0fyymjBfQ+2UW+EujzOWGIaqJX8rLXj1eNTMHW99rNw7GftfF…
            [Member with value [truncated]: folder_id:dffazQ2WBjLe+6xykpbJnbDL4vT/v3YUBIrFDli0fyymjBfQ+2UW+EujzOWGIaqJX8rLXj1eNTMHW99rNw7GftfFk1jIsmmIi6zSNjNr…
            String value [truncated]: dffazQ2WBjLe+6xykpbJnbDL4vT/v3YUBIrFDli0fyymjBfQ+2UW+EujzOWGIaqJX8rLXj1eNTMHW99rNw7GftfFk1jIsmmIi6zSNjNrkiVTh8Nb7eLFWmBy…
            Key: folder_id
```

```
0220  2b 5a 52 77 3d 3d 22 2c  22 66 6f 6c 64 65 72 5f   +ZRw==", "folder_
0230  69 64 22 3a 22 64 66 66  61 7a 51 32 57 42 6a 4c   id":"dff azQ2WBjL
0240  65 2b 36 78 79 6b 70 62  4a 6e 62 44 4c 34 76 54   e+6xykpb JnbDL4vT
0250  2f 76 33 59 55 42 49 72  46 44 6c 69 30 66 79 79   /v3YUBIr FDli0fyy
0260  6d 6a 42 66 51 2b 32 55  57 2b 45 75 6a 7a 4f 57   mjBfQ+2U W+EujzOW
0270  47 49 61 71 4a 58 38 72  4c 58 6a 31 65 4e 54 4d   GIaqJX8r LXj1eNTM
0280  48 57 39 39 72 4e 77 37  47 66 74 66 46 6b 31 6a   HW99rNw7 GftfFk1j
0290  49 73 6d 6d 49 69 36 7a  53 4e 6a 4e 72 6b 69 56   IsmmIi6z SNjNrkiV
02a0  54 68 38 4e 62 37 65 4c  46 57 6d 42 79 72 45 55   Th8Nb7eL FWmByrEU
02b0  45 71 39 44 53 35 6e 48  62 6b 67 45 61 78 34 6c   Eq9DS5nH bkgEax4l
02c0  4d 68 4b 39 73 68 45 68  68 78 4f 37 37 4a 75 73   MhK9shEh hxO77Jus
02d0  6f 6f 76 6d 66 62 31 47  4b 6d 69 4f 38 2b 4f 7a   oovmfb1G KmiO8+Oz
02e0  59 2b 4f 51 30 68 74 72  33 73 55 51 72 38 51 30   Y+OQ0htr 3sUQr8Q0
02f0  71 4c 32 46 48 53 44 41  65 32 38 6b 65 42 54 79   qL2FHSDA e28keBTy
0300  78 58 71 67 43 4d 78 37  55 39 57 68 7a 34 44 65   xXqgCMx7 U9Whz4De
0310  35 71 4d 34 6f 69 48 6c  6e 6d 46 6b 59 4e 33 44   5qM4oiHl nmFkYN3D
0320  58 74 6e 6e 69 71 68 31  79 48 32 50 32 62 6b 43   Xtnniqh1 yH2P2bkC
0330  7a 51 45 47 73 50 76 6b  6e 36 49 54 7a 2f 51 43   zQEGsPvk n6ITz/QC
0340  66 56 37 62 33 51 35 69  59 51 67 6f 75 54 2f 2b   fV7b3Q5i YQgouT/+
0350  70 43 32 4e 49 68 6a 34  45 48 79 46 67 66 57 50   pC2NIhj4 EHyFgfWP
0360  4b 46 6a 75 43 77 72 4b  63 68 4b 6a 72 30 2f 69   KFjuCwrK chKjr0/i
```

Frame (1508 bytes) Reassembled TCP (11706 bytes)

No.: 1885 · Time: 388.740542 · Source: 192.168.18.87 · Destination: 192.168.18.79 · Protocol: HTTP/JSON · Length: 1508 · Info: HTTP/1.1 200 OK , JSON (application/json)

Show packet bytes



```
            [Path: /content/files/[]/updated_at]
          Member: document_size
          Member: id
            [Path with value [truncated]: /content/files/[]/id:Z9w6Pbm79xtv5VePJ2jAEAdlQXduT28Wx5QTTodjvPnkYb1Zws8q2oRw7G9+i5UQdAxhkOnbj78tsXRJeFiP9CByDBjSGNw…
            [Member with value [truncated]: id:Z9w6Pbm79xtv5VePJ2jAEAdlQXduT28Wx5QTTodjvPnkYb1Zws8q2oRw7G9+i5UQdAxhkOnbj78tsXRJeFiP9CByDBjSGNw8mN4uZeKHFcKOtvi…
            String value [truncated]: Z9w6Pbm79xtv5VePJ2jAEAdlQXduT28Wx5QTTodjvPnkYb1Zws8q2oRw7G9+i5UQdAxhkOnbj78tsXRJeFiP9CByDBjSGNw8mN4uZeKHFcKOtviQHBTfsZ3h…
            Key: id
            [Path: /content/files/[]/id]
          Member: created_at
            [Path with value [truncated]: /content/files/[]/created_at:F65e82Uc/8atbfMRuC3iqEpnfgg3TYgkW13ZUwSgNsbGBzfsfghS+NtQoqnDD9YNnjJz4WK0wDZCFjxeAtM4pG4…
            [Member with value [truncated]: created_at:F65e82Uc/8atbfMRuC3iqEpnfgg3TYgkW13ZUwSgNsbGBzfsfghS+NtQoqnDD9YNnjJz4WK0wDZCFjxeAtM4pG4IXO1c0Dtk8i/r18u…
            String value [truncated]: F65e82Uc/8atbfMRuC3iqEpnfgg3TYgkW13ZUwSgNsbGBzfsfghS+NtQoqnDD9YNnjJz4WK0wDZCFjxeAtM4pG4IXO1c0Dtk8i/r18uDjdj2UHoAjO9Mu+/K…
            Key: created_at
```

```
0920  30 6f 51 50 4e 77 3d 3d  22 2c 22 63 72 65 61 74   0oQPNw== ","creat
0930  65 64 5f 61 74 22 3a 22  46 36 35 65 38 32 55 63   ed_at":" F65e82Uc
0940  2f 38 61 74 62 66 4d 52  75 43 33 69 71 45 70 6e   /8atbfMR uC3iqEpn
0950  66 67 67 33 54 59 67 6b  57 31 33 5a 55 77 53 67   fgg3TYgk W13ZUwSg
0960  4e 73 62 47 42 7a 66 73  66 67 68 53 2b 4e 74 51   NsbGBzfs fghS+NtQ
0970  6f 71 6e 44 44 39 59 4e  6e 6a 4a 7a 34 57 4b 30   oqnDD9YN njJz4WK0
0980  77 44 5a 43 46 6a 78 65  41 74 4d 34 70 47 34 49   wDZCFjxe AtM4pG4I
0990  58 4f 31 63 30 44 74 6b  38 69 2f 72 31 38 75 44   XO1c0Dtk 8i/r18uD
09a0  6a 64 6a 32 55 48 6f 41  6a 4f 39 4d 75 2b 2f 4b   jdj2UHoA jO9Mu+/K
09b0  79 44 30 48 6f 2f 63 58  6e 38 73 62 4f 36 4b 33   yD0Ho/cX n8sbO6K3
09c0  53 71 7a 6c 7a 7a 71 4e  31 77 33 49 32 4a 2b 36   SqzlzzqN 1w3I2J+6
09d0  72 6e 4a 35 7a 56 44 49  2b 44 53 52 43 71 46 59   rnJ5zVDI +DSRCqFY
09e0  4d 41 37 46 77 35 6e 72  6b 54 4f 77 76 44 54 48   MA7Fw5nr kTOwvDTH
09f0  33 55 7a 53 38 41 6e 38  33 50 63 69 69 47 76 47   3UzS8An8 3PciiGvG
0a00  62 50 48 37 6d 45 4e 58  6f 33 37 32 54 67 46 61   bPH7mENX o372TgFa
0a10  66 57 36 47 50 58 5a 6d  77 42 38 4c 65 58 45 78   fW6GPXZm wB8LeXEx
0a20  59 32 47 36 6b 72 67 64  71 6c 48 67 77 6d 43 64   Y2G6krgd qlHgwmCd
0a30  70 6f 65 55 6d 38 6e 78  37 46 2b 57 38 56 4b 69   poeUm8nx 7F+W8VKi
0a40  73 47 69 35 49 45 75 36  48 6c 4f 6b 31 6a 63 36   sGi5IEu6 HlOk1jc6
0a50  52 4b 64 38 71 57 66 41  33 51 79 64 38 49 70 72   RKd8qWfA 3Qyd8Ipr
0a60  68 66 63 45 79 75 64 71  30 33 4f 79 38 70 70 2f   hfcEyudq 03Oy8pp/
```

Frame (1508 bytes) Reassembled TCP (11706 bytes)

Key (json.key), 12 bytes

Show packet bytes

5.4. Test Cases

5.4.1. Login User

Enumerate the test cases for the acceptance criteria

Acceptance Criteria	Test Type	Preconditions	Steps	Expectations
Users must enter the email and password and submit a login form.	Positive	• User must have access to login page	• User enter email address and password • User clicks the login button	• User's login request sent to server successfully
	Negative	• User must have access to login page	• User didn't enter credentials • User clicks the login button	• User will see an error message indicating "Email and Password is required"
		• User must have access to login page	• User enter email address • User clicks the login button	• User will see an error message indicating "Password is required"
		• User must have access to login page	• User enter password • User clicks the login button	• User will see an error message indicating "Email address is required"
System verify user's credential in database.	Positive	• User must sent request for login	• System verify the credentials from database • User's credentials	• User will be redirected to Dashboard page

			are valid	
	Negative	• User must sent request for login	• System verify the credentials from database • User with given email address not exists	• User will see an error message indicating “User doesn’t exist with given email address”
		• User must sent request for login	• System verify the credentials from database • User’s password in not valid	• User will see an error message indicating “Invalid password”
System redirect to Dashboard, If credentials are correct.	Positive	• User must sent login request	• User is authorized from database	• User will be redirected to Dashboard page
	Negative	• User must sent login request	• User is unauthorized from database	• User will not allow to access dashboard

Table 17: Test Case User Login

5.4.2. Upload File

Enumerate the test cases for the acceptance criteria

Acceptance Criteria	**Test Type**	**Preconditions**	**Steps**	**Expectations**
User selects the file to upload on server	Positive	• User must be login to Airvault	• User select the file from local machine • User clicks button	• User’s selected file uploaded on server

	Negative	• User must be login to Airvault	• User didn't select the file from local machine • User clicks the submit button	• User will see an error message indicating "Please select a file"
		• User must be login to Airvault	• User select the file with with extension which is not allowed to upload on Airvault	• User will see an error message indicating "File is not in correct format"
System encrypt that file and store it on storage	Positive	• User must upload the file on the server	• System encrypts that file and creates an index table	• System stores that encrypted file in airvault storage
	Negative	• User must upload the file on the server	• System failed to encrypt that file due to server issue	• User will see an error message indicating "File not uploaded"
System will respond to the user with a snackbar on file uploads.	Positive	• User must upload the file on the server	• System encrypts that file and store it in airvault storage	• User will see an success message indicating "File uploaded successfully"
	Negative	• User must upload the file on the server	• System encrypts that file but failed to store it in airvault storage	• User will see an error message indicating "File not uploaded"

Table 18: Test Case Upload File

5.4.3. Download Decrypted File

Enumerate the test cases for the acceptance criteria

Acceptance Criteria	**Test Type**	**Preconditions**	**Steps**	**Expectations**
User clicks the download decrypted file button for specific file	Positive	• User must login to Airvault and has access to file	• User clicks the download file button	• System will send request on server
	Negative	• User must be login to Airvault	• User don't have access to file	• User is unable to send request for download
System will decrypt that file on server	Positive	• User must login to Airvault and have access to file	• System send request on server • Request verifies	• File is decrypted on server
	Negative	• User must login to Airvault and have access to file	• System send request on server • System failed to decrypt file on server	• User will see an error message indicating "Failed to download file"
System download that file on the user's local machine, If the user has access.	Positive	• User must be login to Airvault and have access to file	• System send request on server • System verifies the request • Server decrypt file	• Decrypted file is downloaded on user's local machine
	Negative	• User must be login to Airvault and have access to file	• System send request on server • System failed to decrypt file on server	• User will see an error message indicating "Failed to download file"

Table 19: Test Case Download Decrypted File

5.4.4. Delete File

Enumerate the test cases for the acceptance criteria

Acceptance Criteria	Test Type	Preconditions	Steps	Expectations
User clicks the delete file button for specific file	Positive	• User must be login to Airvault and has access to file	• User clicks the delete file button	• System will send request on server
	Negative	• User must be login to Airvault	• User don't have access to file	• User is unable to send request on server
System checks, If user has delete access to that file	Positive	• User must be login to Airvault and have access to file	• System send request on server • System verifies the request	• System checks for access
	Negative	• User must be login to Airvault and have access to file	• System send request on server • System didn't verifies the request	• User will see an error message indicating "Failed to delete file"
System will delete the file on the server, If the user has access.	Positive	• User must be login to Airvault and have access to file	• System send request on server • System verifies the request • User has delete access	• File will be delete from storage and send success message to user indicating "File deleted"
	Negative	• User must be login to Airvault	• System send request on server	• User will see an error message

		and have access to file	• User don't have delete access	indicating "Failed to delete file"

Table 20: Test Case Delete File

5.4.5. Search any Keyword

Enumerate the test cases for the acceptance criteria

Acceptance Criteria	Test Type	Preconditions	Steps	Expectations
User input keyword to search and click search button	Positive	• User must be login to Airvault • User must be on search page	• User write the keywords in input field	• System will send encrypted request on server
	Negative	• User must be login to Airvault • User must be on search page	• User didn't write any keyword	• Files will not be fetched
System search that keyword on all files within department	Positive	• User must be login to Airvault • User requested to search for a keyword	• System decrypt the request • System search for keyword on index table within department	• System select those file which has searched keyword
	Negative	• User must be login to Airvault • User requested to search for a keyword	• System failed to decrypt the request	• User will see an error message indicating "File not found for searched keyword"
		• User must be	• System decrypt the	• User will see an

		login to Airvault • User requested to search for a keyword	request • System search for keyword on index table • Keyword doesn't exist in any file	error message indicating "File not found for searched keyword"
System check access for files	Positive	• User must be login to Airvault • User requested to search for a keyword	• Keyword found on some files • System select those file which has searched keyword	• System check for access to selected file
	Negative	• User must be login to Airvault • User requested to search for a keyword	• Keyword found on some files • System failed to select files with keyword	• User will see an error message indicating "File not found for searched keyword"
System sends those files which have that keyword and also accessible to user	Positive	• User must be login to Airvault • User requested to search for a keyword	• System select those file which has searched keyword	• User will see a list of files which has searched keyword and accessible to user
	Negative	• User must be login to Airvault • User requested to search for a keyword	• System failed to select those file which has searched keyword	• User will see an error message indicating "File not found for searched keyword"

Table 21: Test Case Search any Keyword within Department

5.5. API Specification

5.5.1. User

POST

API to create new user

URL: /api/user/create

- Body:

```
{
name: string, // "Test Name"
email: string, // "test@gmail.com"
password: string, // "ABC.#123"
server_name: string, // "Accounts Department"
server_ip: string, // "192.168.10.2"
is_admin: boolean, // true
created_at: Date, // 12-10-2023
}
```

- Success

```
{
status: number, // 200
message: string, // "Admin created successfully"
}
```

- Failure

```
{
status: number, // 200
message: string, // "Admin created successfully"
}
```

- Fails On
    - User already exist
    - Incorrect data is sent for name
    - Incorrect data is sent for email
    - Incorrect data is sent for password
    - Incorrect data is sent for department
    - Incorrect data is sent for server_ip
    - Incorrect data is sent for is_admin
- Precondition
    - Admin should be login to system
- Postcondition: User will be created and added in database
- Usage: To create a new user

GET

API to get user

URL: /api/user/<str:user_id>

- Body: None
- Success

```
{
status: number, // 200
message: string, // "User found"
data:{
name: string,     // "Sami Asghar"
email: sting, // "sami@gmail.com"
server_name: string, // "IT Dept"
server_ip: string, // "192.168.10.1"
is_admin: boolean, // True/False
created_at: Date, // 12-10-2023
}
}
```

- Failure

```
{
status: number, // 404
message:string, // "User not found"
}
```

- Fails On
  - User doesn’t exist
- Precondition
  - Admin should be login to system
- Postcondition: None
- Usage: To get user value

UPDATE

API to update user

URL: /api/user/<str:user_id>

- Body:

```
{
name: string, // "Test Name"
password: string, // "ABC.#123"
server_name: string, // "Accounts Department"
server_ip: string, // "192.168.10.2"
is_admin: boolean, // true
}
```

- Success

```
{
status: number, // 200
message: string, // "Admin Updated successfully"
}
```

- Failure

```
{
status: number, // 404
message:string, // "User not found"
}
```

- Fails On
  - User doesn’t exist
  - Admin is not authorized to update user

- Precondition
    - Admin should be login to system
- Postcondition: User information will be updated in database
- Usage: To update a user

DELETE

API to delete user

URL: /api/user/<str:user_id>

- Body: None
- Success

```
{
status: number, // 200
message: string, // "Admin Deleted successfully"
}
```

- Failure

```
{
status: number, // 200
message: string, // "Admin Deleted successfully"
}
```

- Fails On
    - User doesn’t exist
    - Admin is not authorized to delete user

- Precondition
    - Admin should be login to system
- Postcondition: User will be deleted from database
- Usage: To delete a user

5.5.2. File

POST

API to store file

URL: /api/file

- Body:

```
{
file: File,

folder_id: string
}
```

- Success

```
{
status: number, // 200
message: string, // "File uploaded successfully"
}
```

- Failure

```
{
status: number, // 409
message:string, // "File already exist"
}
```

- Fails On
    - File already exist
    - Incorrect file type
    - User is not authorized to upload file
- Precondition
    - User should be login to system
- Postcondition: File will be added in storage
- Usage: To store a file

GET

API to get file

URL: /api/file/<str:file_id>

- Body: None
- Success

```
{
status: number, // 200
message: string, // "File uploaded successfully"

file: File
}
```

- Failure

```
{
status: number, // 409
message:string, // "File already exist"
}
```

- Fails On
  - File doesn’t exist
  - User is not authorized to read file
- Precondition
  - User should be login to system
- Postcondition: None
- Usage: To get a file

UPDATE

API to update file

URL: /api/file/<str:file_id>

- Body:

```
{
file: File
}
```

- Success

```
{
status: number, // 200
message: string, // "File updated successfully"
}
```

- Failure

```
{
status: number, // 404
message:string, // "File doesn't exist"
}
```

- Fails On
    - File doesn’t exist
    - Incorrect file type
    - User is not authorized to upload file
- Precondition
    - User should be login to system
- Postcondition: File will be updated in storage
- Usage: To update a file

DELETE

API to delete file

URL: /api/file/<str:file_id>

- Body: None
- Success

```
{
status: number, // 200
message: string, // "File deleted successfully"
}
```

- Failure

```
{
status: number, // 404
message:string, // "File doesn't exist"
}
```

- Fails On
    - File doesn’t exist
    - User is not authorized to delete file
- Precondition
    - User should be login to system
- Postcondition: File will be deleted from storage
- Usage: To delete a file

5.5.3. Access

UPDATE

API to update access

URL: /api/file-access/<str:file_id>/user/user-id?=<str:user_id>

- Body:

```
{
viewers: string[], // [user_id1, user_id2]

editors: string[], // [user_id3, user_id4]
}
```

- Success

```
{
status: number, // 200
message: string, // "File access updated successfully"
}
```

- Failure

```
{
status: number, // 404
message:string, // "File doesn't exist"
}
```

- Fails On
  - File doesn’t exist
  - Incorrect data for access
  - Admin is not authorized to update access
- Precondition
  - Admin should be login to system
- Postcondition: File access will be updated
- Usage: To update file access

GET

API to get access

URL: /api/file-access/user/user-id?=<str:user_id>

- Body: None

- Success

```
{
status: number, // 200
message: string, // "File access found"

viewers: string[], // [user_id1, user_id2]

editors: string[], // [user_id3, user_id4]
}
```

- Failure

```
{
status: number, // 404
message:string, // "File access doesn't exist"
}
```

- Fails On
  - File access doesn’t exist
  - Admin is not authorized to get access
- Precondition
  - Admin should be login to system
- Postcondition: None
- Usage: To get file access

5.5.4. Search File/Keyword

GET

API to search file/keyword

URL: /api/search/<str:user_id>/filter?search_by=<str:search_filter>&across_department=<bool:across_dept>

- Body: None
- Success

```
{
status: number, // 200
message: string, // "Files found"
}
```

- Failure

```
{
status: number, // 404
message:string, // "Keyword doesn't exist"
}
```

- Fails On
    - Keyword doesn’t exist
    - Incorrect data for search_by filter
    - Incorrect data for across department
- Precondition
    - User should be login to system
- Postcondition: None
- Usage: To get files which has searched keyword

6. Conclusion and Future Work

This chapter includes conclusion of all functionalities of AirVault, it also explain about our future research and work which we plan to do on AirVault.

6.1. Summary of AirVault's Contributions and Significance

A revolutionary project called Airvault addresses the actual problem of safe data management in air-gapped systems. Though they are totally isolated off from other networks, these isolated environments provide better security but make data management more difficult. To solve these issues, AirVault creates a secure system for searchable encryption integration with personal data storage, identity based access management, and retrieval of data in air-gapped systems.

Challenges of Air-Gapped Systems

The security requirements of air-gapped systems are not intended for use with conventional data management solutions. This may result in several problems which an organization has to face:

- Security vulnerabilities

  Within the air-gapped environment itself, traditional solutions might not offer sufficient protection against unauthorized access or data breaches.

- Limited searchability

  To search specific data in a large storage of encrypted information can be a difficult task without proper search functionalities.

- Access control concerns

  There's a risk of unauthorized users gaining access to sensitive data due to inadequate access control mechanisms.

Solution provided by Airvault

Airvault steps in to address these challenges with a diverse approach:

Data Security and Protection

Airvault puts in place strict measures to ensure that data on the isolation chamber also referred to as the air-gapped system is safe. This includes data access control on confidentiality and data integrity, superior encryption mechanisms like AES-256 for safe storage, secure identity-based users.

Efficient Data Retrieval with Searchable Encryption

It is a special type of encryption used by Airvault given that it offers searchable encryption within an air-gapped environment. Thus, users can freely use search voting on encrypted data while not disclosing the data to the search engine. To simplify this, imagine of a way to look for information in the catalog of a certain library without going through an exercise of scanning through the entire books! This improves the access and data security within the air-gapped systems significantly.

Enhanced Access Control

To mitigate the risk of unauthorized access, Airvault employs identity-based access control which grants access based on user identity therefore prioritizes data protection. By doing this, the risk of the user or other parties getting unauthorized access to important data is limited.

Who can benefit from AirVault

Organizations dealing with sensitive information in air-gapped network will definitely be the beneficiaries of Airvault's services. This encompasses organizations that implement air-gapped systems in their operations and require absolute security of data including; government departments, military contractors, healthcare, and financial organizations among others.

Taking all into consideration, Airvault can be considered a significant contribution to the management of data in the framework of the air-gapped system. In this paper, I have demonstrated that with the provision of proper security, AES Full Disk Encryption Mechanism, and Broad Access Authorities, Airvault can enable businesses properly store, access, and retrieve crucial data inside these secure environments.

6.2. Limitations and Areas for Improvement

Despite Airvault provides a way to store data in a very secure manner but there is still a lot of area of improvements, following are some of the limitations in AirVault system

Limitations

Limited Search Capabilities

The search abilities of searchable encryption can be limited based on the particular implementation. For example, it only searches one keyword at a time. Improving search functionality while keeping security is something that needs to be done. Multi words searching is not implemented yet.

Areas for improvement

Disaster Recovery

Any system has the potential of losing data due to hardware malfunctions or other unexpected events. It would be useful to investigate alternatives for safe data backup as well as recovery in an air-gapped setting.

6.3. Future Research Directions and Development Opportunities

Here is an overview of the features we plan to work on in the future, all of which build on Airvault's fundamental features.

Multi-keyword Search

With the help of this feature, users can do multiple keyword searches for data, which improves efficiency and improves search results. Imagine trying to search for specific documents by searching for "financial reports" AND "2024".

Benefits

Greater accuracy as well as efficiency in retrieving relevant data inside the air-gapped system.

Development Considerations

In order to ensure appropriate search results while keeping data security, integrating multi-keyword search with Airvault's existing searchable encryption required careful design.

Built-in Document Viewer

The functionality of an intrinsic document viewer likes to google docs will be integrated in Airvault to view different kinds of documents while they are in the system. This scrubs out the necessity of exporting and comes with the added bonus of preventing exposure to potential data loss.

Benefits

Documents do not have to be closed to give users a preview, which brings out the issue of user interface and convenience in a secure Airvault environment.

Development Considerations

Overall, the people’s awareness of safety remains paramount, to this date. Special attention should be paid to the integration of the document viewer with the tech document since the viewer can pose as a threat and lead to unauthorized access or modification of the document. It might also require additional computer programming to handle multiple types of documents.

Secure Chat Feature

Subscribers would also be able to communicate in specific channels in the isolated partitioned space by using Airvault’s safely integrated chat. This will ensure that data is protected while at the same time ensure that people share the information that they need.

Benefits

A theoretical increase of the authorized user’s collaboration and interconnectivity on the air-gapped system.

Development Considerations

Security is crucial. The last one is fairly obvious and refers to the need for solid algorithms in the chat function in order to ensure messages’ security and to prevent any third-party access. In the same context, additional controls whereby users’ permission based involvement would apply would be required where interests are limited to select channels.

Storing Pictures and Other Formats

Airvault's file storage would be increased by adding support for images and other unconventional data formats. This serves businesses with air-gapped systems that handle a variety of data kinds.

Benefits

It will help Airvault to be more efficient and flexible, and that it will result in a more inclusive method of data management.

Development Considerations

In today’s world, numerous kinds of data necessitate the creation of effective architectures for both its retrieval and storage. Furthermore, there is an appreciation that unconventional data often poses operational security risks and placing the data in an air-gapped environment could be risky.

Enhanced User Interface and Training

The aim of this feature is to improve the awareness of the specific system used in Airvault through user experiences. It includes two primary components to it:

1. Improving the members' satisfaction in using the current Airvault through the redesign of the easier to understand interface.
2. Designing a training course to enable the customers to learn on how to manage their data safely with Airvault and other airvault secure systems.

Benefits

- User acceptance is improved as is efficiency due to simplified and more streamlined User Interface.
- Increased user activity implied by greater user knowledge of how the model operates and improved data security habits.
- Users no longer depend on IT help for daily operations because they learn how to fix and perform tasks independently.

Development Considerations

- Researching the competitor apps and their interface to assess their strength and weaknesses interfering with their usability.
- Working with UX designers and developing an interface in the format that will allow using necessary software and at the same time will be convenient for the user given the conditions of the air-gapped system.
- It is for this reason that training should be developed in formats that include Audio, Video, Power Point, and flipped classroom. g. Here are the best practices of writing tutorials: Tutorial is any communicative aid, informative and educational material that explains how to do something to its recipients (in this case, students) Oral and textual combination – when writing tutorials, it is useful to use all possible media tools to explain what is to be done (lectures, video instructions, written guidelines) to address the students with different learning styles.

7. Requirements Matrix

The Requirements Matrix outlines essential user interactions, security measures, file handling, and administrative functions for the system.

ID	Requirement	Component
R1	The user should have access to the web or mobile application.	Client
R2	The user should be able to enter credentials and submit the login form.	View
R3	When a user logs in to the system for the first time, the key is generated automatically and saved on the specified path.	System
R4	Users must have access to upload textual files on the portal.	Client
R5	Keywords should be extracted from the uploaded file.	Server
R6	Once the keyword list is generated, the index table should be updated with the new keywords.	Server
R7	Inverted index table and data file should be encrypted.	Server
R8	After encryption, the table will be stored on the server and files will be stored in storage.	Server
R9	Users should have access to generate an encrypted search query	Client

	using any keyword.	
R10	Users should be able to send the generated query to the server.	Client
R11	Server should search for the relevant files in the index table using the search query.	Server
R12	Results should be sent back to the user.	Server
R13	Users must have access to download any file from the search results. The downloaded file should be decrypted locally.	Client, Server
R14	User can only upload text, pdf, docx, doc, xlsx, csv files	Client
R15	System will follow the English dictionary.	System
R16	It must be clear to the user how to use the system with minimal instructions.	View
R17	Users can act in a minimal number of steps. No more than 2 or 3.	View
R18	Admin must have access to add new users.	Server
R19	Admin must have access to edit access for any user.	Server

Table 22: Requirements Matrix

8. References

1. Kui Ren, Cong Wang, and Qian Wang, "Security challenges for the public cloud", IEEE Internet Computing, v. no. 1, pp. 69–73, 2012
2. Sunitha .S. Buchade, Prof. P. R. Devale, "A study on searchable encryption schemes", International Journal of Engineering & Technology, 2018, v. 7, no. 2, p. 4, doi: 10.14419/ijet.v7i2.9254
3. Christoph Bosch, Pieter Hartel, Willem Jonker, and Andreas Peter, "A survey of provably secure searchable encryption," ACM Comput. Surv., vol. 47, no. 2, p. 51, Aug. 2014, doi: http://dx.doi.org/10.1145/2636328
4. Shahzaib Tahir, Muttukrishnan Rajarajan, Ali Sajjad, "A ranked searchable encryption scheme for encrypted data hosted on the Public Cloud", 2017 International Conference on Information Networking (ICOIN), January 2017, p. 7, doi: 10.1109/ICOIN.2017.7899512
5. "Software Testing Help", Accessed on May 20, 2022. [Online]. Available: https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/The-number-of-users-for-top-cloud-storage-providers..png
6. "Duality Technologies Inc.", Accessed on: Dec. 11, 2021. [Online]. Available: https://dualitytech.com/
7. "City Defend. Cloud Confidence." Accessed on: Dec 11, 2021. [Online]. Available: https://citydefend.com/
8. "Enveil | Encrypted Veil", Accessed on: Dec 11, 2021. [Online]. Available: https://www.enveil.com/
9. Ronald L Rivest and Alan T Sherman, "Randomized encryption techniques", Springer, pages 145–163, 1983, doi: https://doi.org/10.1007/978-1-4757-0602-4_14
10. Md Iftekhar Salam , Wei-Chuen Yau, Ji-Jian Chin , Swee-Huay Heng , Huo-Chong Ling , Raphael C-W Phan , Geong Sen Poh , Syh-Yuan Tan and Wun-She Yap, "Implementation

of searchable symmetric encryption

11. Du, M., Wang, Q., He, M., and Weng, J., “Privacy preserving indexing and query processing for secure dynamic cloud storage”, IEEE Transactions on Information Forensics and Security, v. 13, no. 9, pp. 2320–2332, Sep 2018, doi: 10.1109/TIFS.2018.2818651
12. “Brief Explanation of Inverted Index Table”, Accessed on: Dec 1, 2021. [Online]. Available: https://medium.com/@igorkopanev/a-brief-explanation-of-the-inverted-index-f082993f8605
13. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. 102 Public key encryption with keyword search. In International conference on the theory and applications of cryptographic techniques, pages 506–522. Springer, 2004.
14. Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000, pages 44–55. IEEE, 2000.
15. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano. Public key encryption with keyword search. In International conference on the theory and applications of cryptographic techniques, pages 506–522. Springer, 2004.
16. Geong Sen Poh, Ji-Jian Chin, Wei-Chuen Yau, Kim-Kwang Raymond Choo, and Moesfa Soeheila Mohamad. 2017. Searchable symmetric encryption: Designs and challenges. ACM Comput. Surv. 50, 3, Article 40 (May 2017), 37 pages. DOI: http://dx.doi.org/10.1145/3064005
17. Park, J.; Yoo, J.; Yu, J.; Lee, J.; Song, J. A Survey on Air-Gap Attacks: Fundamentals, Transport Means, Attack Scenarios and Challenges. Sensors 2023, 23, 3215. https://doi.org/10.3390/s23063215
18. Bulbul, S.S.; Abduljabbar, Z.A.; Najem, D.F.; Nyangaresi, V.O.; Ma, J.; Aldarwish, A.J.Y. Fast Multi-User Searchable Encryption with Forward and Backward Private Access Control. J. Sens. Actuator Netw. 2024, 13, 12. https://doi.org/10.3390/jsan13010012
19. M. Guri, G. Kedma, A. Kachlon and Y. Elovici, "AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies," 2014 9th International Conference on Malicious

and Unwanted Software: The Americas (MALWARE), Fajardo, PR, USA, 2014, pp. 58-67, doi: https://doi.org/10.1109/MALWARE.2014.6999418

20. Arendt, D. (2016). Medical-Grade Network Security - Air-Gap Isolation and PossibleWeak Points. Journal of Applied Computer Science, 24(3), 7-19. https://doi.org/10.34658/jacs.2016.24.3.7-19
21. Okamoto, Tatsuaki, and David Pointcheval. "The gap-problems: A new class of problems for the security of cryptographic schemes." Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001 Cheju Island, Korea, February 13–15, 2001 Proceedings 4. Springer Berlin Heidelberg, 2001.

9. Appendices

9.1. Encrypt file:

This function takes in path of the file that is to be encrypted and then encrypts the file at the end it removed unencrypted file, this process takes place in a split of a second.

```python
async def encrypt_with_ot_aes_key(path):
    try:

        iv, key = await generate_ot_aes_key()
        with open(path, "rb") as f:
            plaintext = f.read()
            padding_length = 16 - len(plaintext) % 16
            plaintext += bytes([padding_length]) * padding_length
            cipher = Cipher(
                algorithms.AES(key), modes.CBC(iv), backend=default_backend()
            )
            encryptor = cipher.encryptor()

            ciphertext = encryptor.update(plaintext) + encryptor.finalize()
            content = base64.b64encode(ciphertext).decode("utf-8")
            iv = base64.b64encode(iv).decode("utf-8")
            key = base64.b64encode(key).decode("utf-8")
            f.close()
        os.remove(path)
        return content, iv, key
    except Exception as e:
        print("Error in encrypting file", e)
        return None, None, None
```

Figure 71: Code Snippet Encrypt File Function

9.2. Decrypt file:

This function takes in path of the file and name of file that is to be decrypted and then decrypt the file it temporarily store file as a (spooled file) and at the end it removed unencrypted file.

```python
async def decrypt_file(file_path, file_name):
    write_path = os.path.join(ENCRYPTED_FILES, file_name)
    try:
        with open(KEY_PATH, "rb") as key_file:
            key = key_file.read()
            with open(IV_PATH, "rb") as iv_file:
                iv = iv_file.read()
            print(file_path)
            with open(file_path, "rb") as f:
                ciphertext = f.read()
                cipher = Cipher(
                    algorithms.AES(key), modes.CBC(iv), backend=default_backend()
                )  # Use CBC mode
                decryptor = cipher.decryptor()
                plaintext = decryptor.update(ciphertext) + decryptor.finalize()
                # Remove padding
                padding_length = plaintext[-1]
                plaintext = plaintext[:-padding_length]
                print(write_path)
                with open(write_path, "wb") as plain:
                    plain.write(plaintext)
                return write_path
    except Exception as e:
        print("Error in decrypting file received", e)
```

Figure 72: Code Snippet Decrypt File Function

9.3. Encryption using RSA:

This function takes RSA public key and data as arguments and encrypts the data. It returns the encrypted data.

```
async def encrypt_from_rsa_public_key_string(key: str, data: str):
    try:
        data = str(data)
        data = data.encode()
        key = base64.b64decode(key)
        public_key = serialization.load_pem_public_key(key, backend=default_backend())
        ciphertext = public_key.encrypt(
            data,
            padding.OAEP(
                mgf=padding.MGF1(algorithm=hashes.SHA256()),
                algorithm=hashes.SHA256(),
                label=None,
            ),
        )
        ciphertext = base64.b64encode(ciphertext).decode()
        return ciphertext
    except Exception as e:
        print("Error in encrypting from rsa key", e)
        return None
```

Figure 73: Code Snippet Encrypt String by RSA Function

9.4. Decryption using RSA:

This function takes data as argument and decrypts the data from the private key. It returns the decrypted data.

```
async def decrypt_from_rsa_private_key(data: str):
    try:
        data = base64.b64decode(data)

        private_key = read_private_key_from_file()
        plain_text = private_key.decrypt(
            data,
            padding.OAEP(
                mgf=padding.MGF1(algorithm=hashes.SHA256()),
                algorithm=hashes.SHA256(),
                label=None,
            ),
        )
        plain_text = plain_text.decode()

        return plain_text
    except Exception as e:
        print("Error in decrypting from rsa key", e)
        return ""
```

Figure 74: Code Snippet Decrypt String by RSA Function

9.5. Keywords Generation

These functions is used for keyword generation and storing them in database after encryption. It uses nltk library for keyword generation, stop words are removed from the system, words are encrypted and then stored in inverted index table.

```
async def extract_keywords_and_encrypt(
    text: str, folder_id: int, file_name: str, db: Session
):
    try:
        tokens = word_tokenize(text.lower())
        stop_words = set(stopwords.words("english"))
        filtered_tokens = [
            word for word in tokens if word.isalnum() and word not in stop_words
        ]
        unique_keywords = set(filtered_tokens)
        encrypted_keywords = []
        for keyword in unique_keywords:
            keyword = keyword.lower()
            encrypted_keyword = await encrypt_string_from_server_aes_key(keyword)
            encrypted_keyword = encrypted_keyword.decode("utf-8", "ignore")
            encrypted_keyword = base64.b64encode(encrypted_keyword.encode())
            encrypted_keywords.append(encrypted_keyword)

        await store_keywords_in_db(encrypted_keywords, folder_id, file_name, db)

        return
    except Exception as e:
        print(e, "Error in extrating keywords from file")
        return
```

Figure 74: Code Snippet of Keyword Generation Function