



ATTRIBUTE-BASED-ENCRYPTION IN DISRUPTION TOLERANT MILITARY NETWORKS TO SECURE DATA ACCESS

A PROJECT REPORT

Submitted by

SONIA.M

Reg. No. 410811104094

*in partial fulfillment of the requirements
for the award of the degree*

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE & ENGINEERING

**GKM COLLEGE OF ENGINEERING & TECHNOLOGY
CHENNAI - 600063**

**ANNA UNIVERSITY
CHENNAI 600 025
APRIL, 2015**

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**ATTRIBUTE-BASED-ENCRYPTION IN DISRUPTION TOLERANT MILITARY NETWORKS TO SECURE DATA ACCESS**” is the bonafide work of **SONIA.M (410811104094)** who carried out the project work under my supervision.

SIGNATURE

Dr.S.Selvakumar, M.E., Ph.D

HEAD OF THE DEPARTMENT

Professor

Department of Computer Science and

Engineering

GKM College of Engineering &

Technology

G.K.M. Nagar

Chennai – 600063.

SIGNATURE

Mr.V.Hemakumar, M.E.

SUPERVISOR

Assistant Professor

Department of Computer Science and

Engineering

GKM College of Engineering &

Technology

G.K.M. Nagar

Chennai – 600063.

**GKM COLLEGE OF ENGINEERING AND TECHNOLOGY
CHENNAI-600 063**

ANNA UNIVERSITY: CHENNAI 600 025

PROJECT VIVA-VOCE EXAMINATION

The viva-voce examination of the project work submitted by,

SONIA.M 410811104094

is to be held on at Computer Science and Engineering
Department .

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Wireless devices carried by soldiers in hostile areas or battle fields are likely to suffer the threats of information leaks and disruptions in connectivity. Disruption-tolerant network (DTN) technologies allow wireless devices carried by soldiers to communicate with each other and access the confidential information or command reliably by exploiting external storage nodes. Some of the most challenging issues in this scenario are the enforcement of authorization policies and the policies update for secure data retrieval. Cipher Text-policy ABE (CP-ABE) provides a scalable way of encrypting data such that the encryptor defines the attribute set that the decryptor needs to possess in order to decrypt the Cipher Text. Thus, different users are allowed to decrypt different pieces of data per the security policy. However, the problem of applying CP-ABE in decentralized DTNs introduces several security and privacy challenges with regard to the attribute revocation, key escrow, and coordination of attributes issued by authority. The existing system says about key generation using multiple authorities. In this paper, we propose a secure data retrieval scheme using CP-ABE for DTNs where key generation is done by automated scheme. The Admin(Sender of message) generates a random one-time-key for the user based on the attributes, which solves the problems of forward secrecy and backward secrecy. In addition it solves key escrow problem since central authority is not explicitly involved in key generation. We demonstrate how to apply the proposed mechanism to securely and efficiently manage the confidential data distributed in the disruption-tolerant military network.

ACKNOWLEDGEMENT

I sincerely thank our respected Chairman, **Dr.G.KATHAMUTHU, M.A.** GKM Group of Educational Institutions for all his efforts and administrations in educating us in his prestigious institution.

I take this opportunity to thank our CEO, **Dr.SUJATHA BALASUBRAMANIAN, M.B.A., Ph.D.**, for the kind cooperation in helping us to complete the project.

I express my sincere thanks to our director, **Dr.K.P.JAGANNATHAN, M.Tech., Ph.D.,(Met. Engg)** for providing appropriate facilities for completing this project.

I would like to express my gratitude to our principal, **Dr.C.CHELLAPPAN, M.E., Ph.D.** for providing the valuable guidance.

I wish to extend my grateful acknowledgement and sincere thanks to my Head of the Department **Dr.S.SELVAKUMAR, M.E., Ph.D.** for his constant motivation, encouragement and criticism in completing the project successfully.

I would endow my respectful and benevolent gratefulness to my project guide **Mr.V.HEMAKUMAR , M.E.** for his regular monitoring and immense help which made this project a huge success.

I thank the entire staff members of our department for helping me by providing valuable suggestions and timely ideas for successful completion of the project.

Last but not the least, my family members and friends have been a great source of inspiration and strength to us during the course of this project work. My sincere thanks to them.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	viii
	LIST OF TABLES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	
	1.1 Network Security	1
	1.2 Disruption Tolerant Network	3
	1.3 Problem Definition	4
	1.4 Methodology	4
2	SYSTEM ANALYSIS	
	2.1 Literature Survey	6
	2.2 Existing System	11
	2.2.1 Disadvantages	11
	2.3 Proposed System	12
	2.3.1 Advantages	12
	2.4 Requirement Analysis	13
	2.4.1 Introduction	13
	2.4.2 Hardware Specification	13
	2.4.3 Software Specification	13
	2.4.4 Technology Overview	13
	2.4.5 Design and Implementation Constraints	15
3	SYSTEM DESIGN	
	3.1 Architecture	16
	3.2 Algorithm	17
	3.2.1 RSA Algorithm	17

	3.2.2 AES Algorithm	19
	3.3 Data Flow Diagram	23
	3.4 UML Diagrams	23
4	SYSTEM IMPLEMENTATION	
	4.1 List of Modules	30
	4.2 Module Description	30
	4.2.1 User and Message creation	30
	4.2.2 Key Transmission	33
	4.2.3 Storage	33
	4.2.4 Retrieving message	33
5	TESTING, PERFORMANCE & COMPARATIVE ANALYSIS	
	5.1 Testing	34
	5.1.1 Unit Testing	34
	5.1.2 Integration Testing	34
	5.1.3 Functional Testing	34
	5.1.4 Manual Testing	35
	5.1.5 Test Case	35
	5.2 Performance	36
	5.3 Comparative Analysis	37
6	CONCLUSION	39
	6.1 Future Work	39
	APPENDIX	40
	Appendix 1 – Sample Source Coding	40
	Appendix 2 – Screen Shots	50
	Appendix 3 – Paper Published	54
	REFERENCES	55

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1	Approaches For Protection Against Security Threats	3
2	Network Model	7
3	Architecture	16
4	RSA Key Generation	18
5	AES Algorithm	19
6	Key Matrix	21
7	AES Encryption	22
8	Data Flow Diagram	23
9	Use Case Diagram	24
10	Class Diagram	26
11	Sequence Diagram	27
12	Activity Diagram	28
13.1	State Diagram for Sender	28
13.2	State Diagram for User	29
14	Component Diagram	29
15	RSA	31
16	AES	32
17	Number of users in an attribute group	38
18	Communication cost in the proposed systems	38
A.1	Admin	50
A.2	Add User	51
A.3	Send Message	51
A.4	Key Authority	52
A.5	Storage Node	52
A.6	(A). Device Login	53
	(B). Request Message	
A.7	(A). Decryption Key	53
	(B). Decrypt Message	

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
1	Test Cases	35
2	Analysis	37

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
CA	Central Authority
IEEE	The Institute of Electrical and Electronics Engineers
DTN	Disruption Tolerant Network
ABE	Attribute-Based-Encryption
KP-ABE	Key Policy Attribute Based Encryption
CP-ABE	Cipher Text Policy Attribute Based Encryption
2PC	Two Party Communication
API	Application Programming Interface
AES	Advanced Encryption Standard
RSA	Rivest-Shamir-Adleman

CHAPTER 1

INTRODUCTION

1.1 NETWORK SECURITY

Security in computer systems is strongly related to the notion of dependability. Informally, a dependable computer system is one that we justifiably trust to deliver its services. Dependability includes availability, reliability, safety, and maintainability. However, if we are to put our trust in a computer system, then confidentiality and integrity should also be taken into account. **Confidentiality** refers to the property of a computer system whereby its information is disclosed only to authorized parties. **Integrity** is the characteristic that alterations to a system's assets can be made only in an authorized way. In other words, improper alterations in a secure computer system should be detectable and recoverable. Major assets of any computer system are its hardware, software, and data. Another way of looking at security in computer systems is that we attempt to protect the services and data it offers against security threats. There are four types of security threats to consider:

1. Interception
2. Interruption
3. Modification
4. Fabrication

Interception refers to the situation that an unauthorized party has gained access to a service or data. A typical example of interception is where communication between two parties has been overheard by someone else. Interception also happens when data are illegally copied, for example, after breaking into a person's private directory in a file system.

An example of interruption is when a file is corrupted or lost. In general, interruption refers to the situation in which services or data become unavailable, unusable, destroyed, and so on. In this sense, denial of service attacks by which someone maliciously attempts to make a service inaccessible to other parties is a security threat that classifies as interruption.

Modifications involve unauthorized changing of data or tampering with a service so that it no longer adheres to its original specifications. Examples of modifications include

intercepting and subsequently changing transmitted data, tampering with database entries, and changing a program so that it secretly logs the activities of its user.

Fabrication refers to the situation in which additional data or activity are generated that would normally not exist. For example, an intruder may attempt to add an entry into a password file or database. Likewise, it is sometimes possible to break into a system by replaying previously sent messages. We shall come across such examples later in this chapter. Note that interruption, modification, and fabrication can each be seen as a form of data falsification.

A security policy describes precisely which actions the entities in a system are allowed to take and which ones are prohibited. Entities include users, services, data, machines, and so on. Once a security policy has been laid down, it becomes possible to concentrate on the security mechanisms by which a policy can be enforced. Important security mechanisms are:

1. Encryption
2. Authentication
3. Authorization
4. Auditing

Encryption is fundamental to computer security. Encryption transforms data into something an attacker cannot understand. In other words, encryption provides a means to implement confidentiality. In addition, encryption allows us to check whether data have been modified. It thus also provides support for integrity checks.

Authentication is used to verify the claimed identity of a user, client, server, and so on. In the case of clients, the basic premise is that before a service will do work for a client, the service must learn the client's identity. Typically, users are authenticated by means of passwords, but there are many other ways to authenticate clients.

After a client has been authenticated, it is necessary to check whether that client is authorized to perform the action requested. Access to records in a medical database is a typical example. Depending on who accesses the database, permission may be granted to read records, to modify certain fields in a record, or to add or remove a record.

Auditing tools are used to trace which clients accessed what, and which way. Although auditing does not really provide any protection against security threats, audit logs can be

extremely useful for the analysis of a security breach, and subsequently taking measures against intruders. For this reason, attackers are generally keen not to leave any traces that could eventually lead to exposing their identity. In this sense, logging accesses makes attacking sometimes a riskier business.

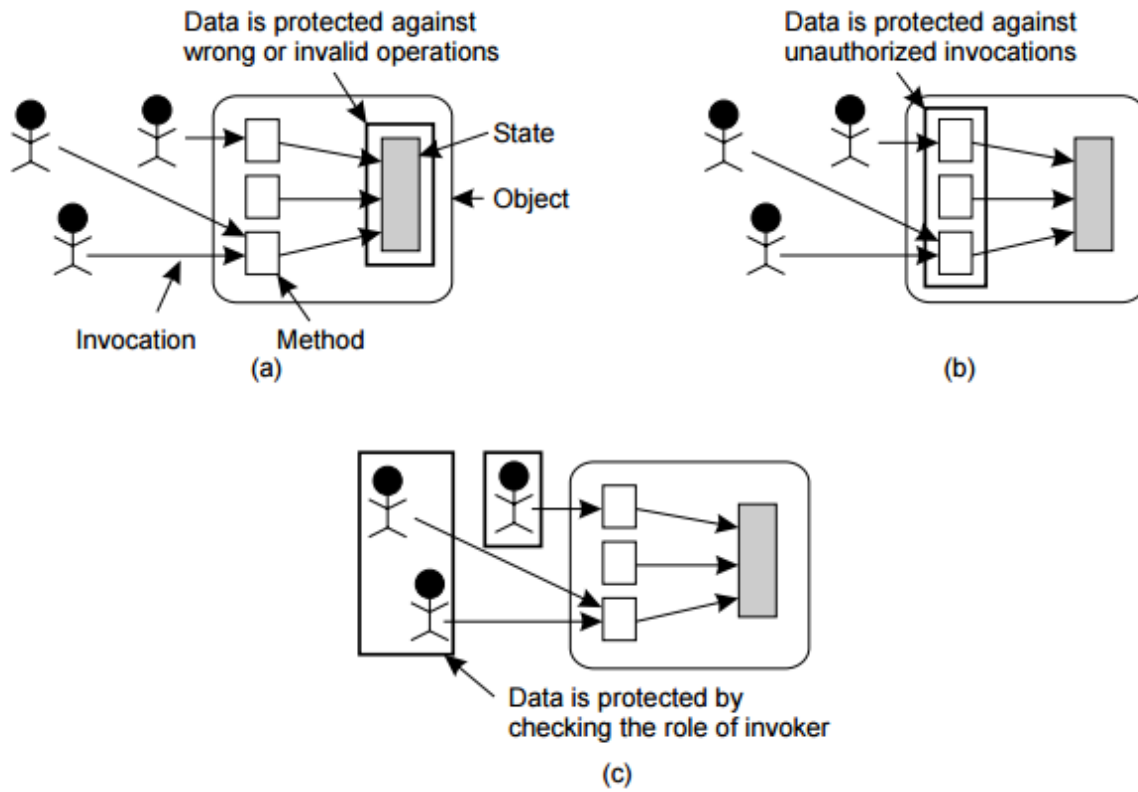


Fig 1: Three approaches for protection against security threats. (a) Protection against invalid operations (b) Protection against unauthorized invocations. (c) Protection against unauthorized users.

1.2 DISRUPTION TOLERANT NETWORK

A disruption-tolerant network (DTN) is a network architecture that reduces intermittent communication issues by addressing technical problems in heterogeneous networks that lack continuous connectivity. Disruption-tolerant network (DTN) technologies are becoming successful solutions that allow nodes to communicate with each other in these extreme networking environments. Typically, when there is no end-to-end connection between a source and a destination pair, the messages from the source node may need to wait in the intermediate nodes for a substantial amount of time until the connection would be eventually established. These nodes use network storage to manage, store, and forward operations over multiple paths and longer periods. Security also protects the infrastructure

from unauthorized use. Roy and Chuah introduced storage nodes in DTNs where data is stored or replicated such that only authorized mobile nodes can access the necessary information quickly and efficiently.

1.3 PROBLEM DEFINITION

DTN Technologies allow wireless devices to efficiently exchange information with each other. CP-ABE provides a scalable way of encrypting data such that the encryptor defines the attribute set that the decryptor needs to possess in order to decrypt the Cipher Text. The User can access the data by decrypting it with a key. The key here is generated by a decentralized network of Key authorities and a Central Authority which gives the limitation of key Escrow Problem and Coordination of attributes; if the key authorities are compromised then the information is compromised. Another problem is with setting up the system of multiple authorities. The system has other limitations like forward secrecy and backward secrecy where the user is able to access data even if he doesn't possess the necessary attributes. I propose a mechanism with Centralized System where the Sender himself generates a random one-time-key for the user and transmits it to the user through Central Authority. Single Authority system is easy to handle than multi-authority system.

1.4 METHODOLOGY

The data owner enforces access control for the outsourced data by combining three cryptographic techniques: selective broadcast, Cipher Text Attribute-Based-Encryption and Key-Pair Generation.

Selective Broadcast – The message will be sent only to the intended devices which satisfy the attributes specification.

Cipher Text Attribute-Based-Encryption – This allows the sender to define policies that must be possessed by the user in order to decrypt the message.

Key-Pair Generator – This method generates Private and public key pair by randomly selecting two prime numbers. The public key is used to encrypt the message. The message can be decrypted only with the corresponding Private Key.

- $\text{RSAKeyPair}(\text{Length})$ – This method takes the length of the keys as parameter. It randomly selects two prime numbers and generates Private Key and Public Key pair with respect to the length. Here ‘Length’ is the length of the Key.
- $\text{RSAEncrypt}(P_{uK}, PT)$ – This method takes the RSA public Key and Plain Text as input. The Plain Text is encrypted with Public Key. Here P_{uK} is RSA Public Key and PT is the Plain Text.
- $\text{RSADecrypt}(P_{rK}, CT)$ – This method takes the RSA Private Key and Cipher Text as input. The Cipher Text is decrypted with the Private Key. Here P_{rK} is the RSA Private Key and CT is the Cipher Text.
- $\text{AESEncrypt}(U_k, P_{rK})$ – This method takes User Key and RSA Private Key as input. The Private Key is encrypted with the User Key. Here U_k is the User Key and P_{rK} is the RSA Private Key.
- $\text{AESDecrypt}(U_k, P_{rK})$ – This method takes the User Key and RSA Private Key as input. The Private Key is decrypted with the User Key. Here U_k is the User Key and P_{rK} is the RSA Private Key.

CHAPTER 2

SYSTEM ANALYSIS

2.1 LITERATURE SURVEY

➤ Cipher Text-Policy Attribute-Based Encryption

Cipher Text-Policy Attribute Based Encryption allows for a new type of encrypted access control where user's private keys are specified by a set of attributes and a party encrypting data can specify a policy over these attributes specifying which users are able to decrypt. This system allows policies to be expressed as any monotonic tree access structure and is resistant to collusion attacks in which an attacker might obtain multiple private keys. In KP-ABE, the encryptor only gets to label a Cipher Text with a set of attributes. The key authority chooses a policy for each user that determines which Cipher Texts he can decrypt and issues the key to each user by embedding the policy into the user's key. However, the roles of the Cipher Texts and keys are reversed in CP-ABE. In CP-ABE, the Cipher Text is encrypted with an access policy chosen by an encryptor, but a key is simply created with respect to an attributes set. CP-ABE is more appropriate to DTNs than KP-ABE.^[9]

➤ Serialization in data centre networks

In data center networks, distributed file systems where TCP is used as a communication protocol between a client and a server are very popular. In such distributed systems, a block of data is partitioned into several units called SRUs (Server Request Units) and they are stored into several servers. Thus, when an application on a client tries to read a block, the client sends requests to the servers which have the corresponding SRUs and then the servers almost simultaneously try to send the SRUs to the client. Then, because many packets burstly arrive at the client link, many packets are lost at the port buffer of the client link and consequently some servers have to wait the retransmissions of their lost packets until timeouts occur. By the serialization, the number of packets in the port buffer becomes very small, which leads to Incast avoidance.^[2]

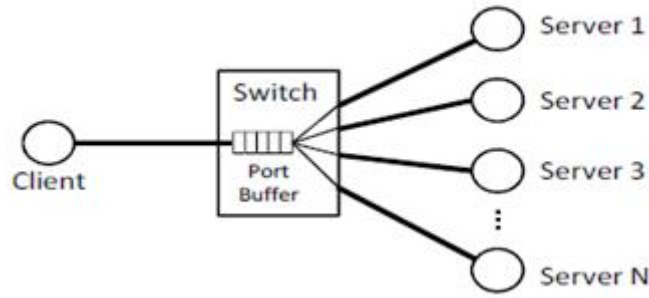


Fig 2: Network Model

➤ Selective group broadcast using dynamic attribute based encryption

Cautious landlords replace the house locks after tenants leave because they worry that tenants might keep copies of the keys. The same concept applies to protecting confidential information. Whenever a user leaves a communication group that has been exchanging and sharing confidential information, the remaining group members will replace the key used to encrypt the messages with a new one. However, given the high cost of key redistribution, this can impact performance especially when the group is made of thousands of users and the group members are likely to move in and out frequently. This work is a follow-up of previous research on Situation Aware Trust. SAT uses descriptive attributes to efficiently perform security policy control. It also builds mechanisms for predicting future trust situations, and; it transforms trust from Internet social communities to VANET trust in order to enhance and promote VANET applications. ABE was used as the basis to integrate the policy control and security services for SAT. One of SAT main contributions was to divide attributes into two categories, static and dynamic, in order to improve efficiency. However, this does not help much since updating is still in terms of (private) keys, instead of attributes. In this paper, we introduce the concept of fading function to CP-ABE, allowing attributes “dynamic”. DABE provides an efficient mechanism for attribute revocation that does not require the reissuing of the entire key. Namely, attributes can be updated independently from entire private keys, making key management in presence of dynamic attributes much more efficient and scalable.^[1]

➤ **Constant-Size Keys for Lightweight Devices**

Lightweight devices, such as radio frequency identification tags, have a limited storage capacity, which has become a bottleneck for many applications, especially for security applications. However, current CP-ABE schemes suffer from the issue of having long decryption keys, in which the size is linear to and dependent on the number of attributes. This drawback prevents the use of lightweight devices in practice as a storage of the decryption keys of the CP-ABE for users. This CP-ABE scheme offers a constant-size decryption key whose length can be as small as 672 bits (80-bit security).^[5]

➤ **Decentralized Key-Policy Attribute-Based Encryption**

Decentralized attribute-based encryption (ABE) is a variant of a multi authority ABE scheme where each authority can issue secret keys to the user independently without any cooperation and a central authority. This is in contrast to the previous constructions, where multiple authorities must be online and setup the system interactively, which is impractical. Hence, it is clear that a decentralized ABE scheme eliminates the heavy communication cost and the need for collaborative computation in the setup stage. Furthermore, every authority can join or leave the system freely without the necessity of reinitializing the system. In contemporary multi authority ABE schemes, a user's secret keys from different authorities must be tied to his global identifier (GID) to resist the collusion attack. However, this will compromise the user's privacy. Multiple authorities can collaborate to trace the user by his GID, collect his attributes, then impersonate him. A privacy-preserving decentralized key-policy ABE scheme where each authority can issue secret keys to a user independently without knowing anything about his GID. Therefore, even if multiple authorities are corrupted, they cannot collect the user's attributes by tracing his GID.^[3]

A Personal Health Records service allows a patient to create, manage, and control her personal health data in one place through the web, which has made the storage, retrieval, and sharing of the medical information more efficient. Especially, each patient is promised the full control of her medical records and can share her health data with a wide range of users, including healthcare providers, family members or friends. Due to the high cost of building and maintaining specialized data centres, many PHR services are outsourced to or provided by third-party service providers, for example, Microsoft HealthVault. However, there have been wide privacy concerns as personal health information could be exposed to those third

party servers and to unauthorized parties. To achieve fine-grained and scalable data access control for PHRs, we leverage attribute-based encryption (ABE) techniques to encrypt each patient's PHR file. Focus is on the multiple data owner scenario, and divide the users in the PHR system into multiple security domains that greatly reduces the key management complexity for owners and users. A high degree of patient privacy is guaranteed simultaneously by exploiting multi authority ABE.^[11]

➤ **Constant sized Cipher Text**

CP-ABE enforces expressive data access policies and each policy consists of a number of attributes. Most existing CP-ABE schemes incur a very large Cipher Text size, which increases linearly with respect to the number of attributes in the access policy. Privacy Preserving Constant CP-ABE (denoted as PP-CP-ABE) significantly reduces the Cipher Text to a constant size with any given number of attributes. Furthermore, PP-CP-ABE leverages a hidden policy construction such that the recipients' privacy is preserved efficiently. Compared to existing Broadcast Encryption (BE) schemes, PP-AB-BE is more flexible because a broadcasted message can be encrypted by an expressive hidden access policy, either with or without explicitly specifying the receivers. Moreover, PP-AB-BE significantly reduces the storage and communication overhead to the order of, where is the system size.^[4]

➤ **Attribute-Based Encryption**

ABE is a public-key based one-to-many encryption that allows users to encrypt and decrypt data based on user attributes. A promising application of ABE is flexible access control of encrypted data stored in the cloud, using access policies and ascribed attributes associated with private keys and Cipher Texts. ABE system with outsourced decryption that largely eliminates the decryption overhead for users. In such a system, a user provides an untrusted server, say a cloud service provider, with a transformation key that allows the cloud to translate any ABE Cipher Text satisfied by that user's attributes or access policy into a simple Cipher Text, and it only incurs a small computational overhead for the user to recover the plaintext from the transformed Cipher Text. Security of an ABE system with outsourced decryption ensures that an adversary (including a malicious cloud) will not be able to learn anything about the encrypted message; however, it does not guarantee the correctness of the transformation done by the cloud. Informally, verifiability guarantees that a user can efficiently check if the transformation is done correctly.^[8]

➤ Two Party Communication Protocol

Adoptions of data sharing paradigm in distributed systems have increased the demands and concerns for distributed data security. Here the data is being stored in external storages. CP-ABE enables data owners to define their own access policies over user attributes and enforce the policies on the data to be distributed. The key generation centre could decrypt any messages addressed to specific users by generating their private keys. This is not suitable for data sharing scenarios where the data owner would like to make their private data only accessible to designated users. In addition, applying CP-ABE in the data sharing system introduces another challenge with regard to the user revocation since the access policies are defined only over the attribute universe. In this study, a novel CP-ABE scheme for a data sharing system is proposed by exploiting the characteristic of the system architecture. The key escrow problem could be solved by escrow-free key issuing protocol, which is constructed using the secure two-party computation between the key generation centre and the data-storing centre.^[7]

2.2 EXISTING SYSTEM

CP-ABE provides a scalable way of encrypting data such that the encryptor defines the attribute set that the decryptor needs to possess in order to decrypt the Cipher Text. Thus, different users are allowed to decrypt different pieces of data per the security policy. The Central Authority grants key to the user with which they retrieve the data from storage node. In the existing scheme, multiple key authorities are likely to manage their own dynamic attributes for soldiers in their deployed regions or echelons, which could be frequently changed (e.g., the attribute representing current location of moving soldiers). Multiple authorities issue and manage their own attribute keys independently as a decentralized DTN. The challenge is the coordination of attributes issued from different authorities. When multiple authorities manage and issue attributes keys to users independently with their own master secrets, it is very hard to define fine-grained access policies over attributes issued from different authorities. It is clear that a decentralized ABE scheme eliminates the heavy communication cost and the need for collaborative computation in the setup stage. Furthermore, every authority can join or leave the system freely without the necessity of reinitializing the system. However, multiple authorities can collaborate to trace the user.

2.2.1 Disadvantages

- Key Revocation – Updation of user Attributes in case of repositioning.
- Key Escrow – Information may be leaked if the Key Authorities are compromised. Multiple authorities can collaborate to trace the user.
- Coordination of Attributes – Key generation is dependent on multiple authorities . Coordinating the Central Authority and multiple key authorities is necessary to generate keys.
- Forward Secrecy – A revoked user will be able to access data until expiration time even if he does not hold the required attribute.
- Backward Secrecy – A user who holds the new attribute might be able to access the previously encrypted data.

2.3 PROPOSED SYSTEM

In the proposed system, we propose an efficient and secure data retrieval method using CP-ABE for centralized DTNs where central authority and sender manage the key generation using automated scheme that generates random one-time-key based on attributes. The Admin himself generates the Key for the User and sends it to CA. The CA then grants the Key to the User. The CA maintains information about the time when the key was sent to the CA from Admin and the time when the message was created by the Admin. In this technique, we are using AES Algorithm for encryption and decryption of Private Key and RSA algorithm for Encryption and Decryption of Message. This solves the problem of forward secrecy and backward secrecy. It also solves Key Escrow problem since central authority is not explicitly involved in key generation. In addition, the system promotes reliability since the central authority keeps tracking on when message was sent and received and also about key reaching the soldier in time. The inherent key escrow problem is resolved such that the confidentiality of the stored data is guaranteed even under the hostile environment where key authorities might be compromised. In addition, the fine grained key revocation can be done for each attribute group.

2.3.1 Advantages

- Key Revocation – Updation of user Attributes in case of repositioning is done by Admin.
- Key Escrow – Central authority is not explicitly involved in key generation.
- Coordination of Attributes – The system is Centralized with only one Central Authority.
- Forward Secrecy – A revoked user will not be able to access data the old data because the admin will remove the access for revoked user.
- Backward Secrecy – A user who holds the new attribute won't be able to access the previously encrypted data since the keys are randomly generated one-time-key. The data can be received only when admin gives access.

2.4 REQUIREMENT ANALYSIS

2.4.1 Introduction

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirement analysis process. This phase involves listing the requirements for a particular software system including functional , performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirements specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software functionality.

2.4.2 Hardware Specification

- Processor : Dual Core or above
- Hard Disk : 40 GB
- Monitor : 15 VGA Colour.
- RAM : 2GB or above

2.4.3 Software Requirements

- Operating system : Windows 7.
- JDK : Java 1.8
- Tool Kit : Eclipse Kepler
- Front End/GUI Tool : Java
- UML tool : Visual Paradigm 12.0

2.4.4 Technology Overview

Java is a high-level programming language and a platform originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Java is a high level, robust, secured and object-oriented programming language.

Platform: Any hardware or software environment in which a program runs, is known as a platform. Since Java has its own runtime environment (JRE) and API, it is called platform.

FEATURES:

- **Platform independent:** Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.
- **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java would be easy to master.
- **Secure:** With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architectural-neutral:** Java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors, with the presence of Java runtime system.
- **Portable:** Being architectural-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary which is a POSIX subset.
- **Robust:** Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded:** With Java's multithreaded feature it is possible to write programs that can do many tasks simultaneously. This design feature allows developers to construct smoothly running interactive applications.
- **Interpreted:** Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light weight process.
- **High Performance:** With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed:** Java is designed for the distributed environment of the internet.
- **Dynamic:** Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

ECLIPSE KEPLER:

Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written

mostly in Java, Eclipse can be used to develop applications. By means of various plug-ins, Eclipse may also be used to develop applications in various other programming languages like C++, COBOL , Perl , PHP, Python , Fortran, JavaScript and also for the development of Android Applications.

2.4.5 Design and Implementation Constraints

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

CHAPTER 3

SYSTEM DESIGN

3.1 ARCHITECTURE

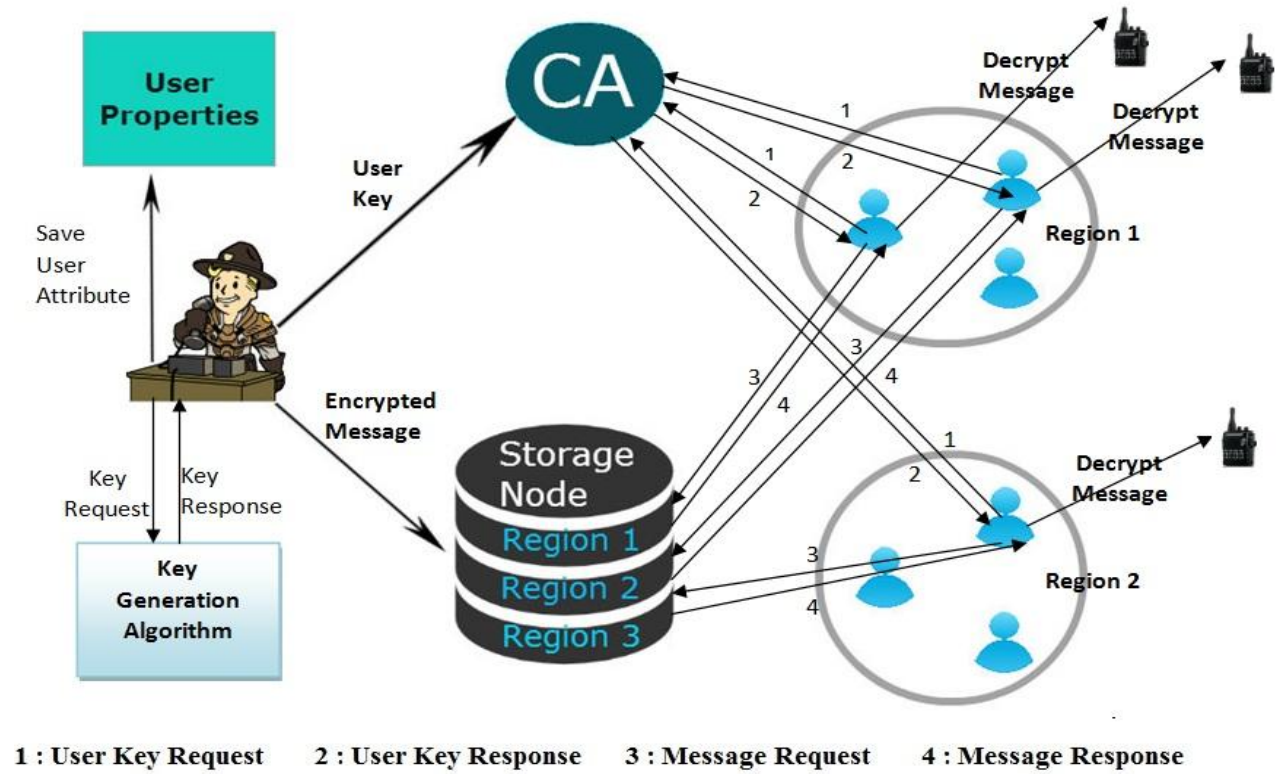


Fig 3 : Architecture

The Sender first decides the batch and the Region for which the message has to be sent. The sender is an entity who owns confidential messages or data (e.g., a commander) and wishes to store them into the external data storage node for ease of sharing or for reliable delivery to users in the extreme networking environments. A sender is responsible for defining (attribute based) access policy and enforcing it on its own data by encrypting the data under the policy before storing it to the storage node. CP-ABE provides a scalable way of encrypting data such that the encryptor defines the attribute set that the decryptor needs to possess in order to decrypt the Cipher Text.

The sender generates the one-time-key for the particular user and passes it through the central authority. RSA Private and Public key pair is used for encryption and decryption of the message. The Sender encrypts the message with RSA Public Key. The message can be decrypted by the user only with the corresponding RSA private key pair. The sender sends

the encrypted message to the Storage Node. Meanwhile the Sender encrypts the RSA Private Key with the User Key using AES algorithm, so that only the user who has got the User Key can decrypt the message.

The central authority receives the key from Admin and renders the key to the user on request. The Central authority keeps record of the time when the key was generated and sent to the Central Authority, the time when the message was created, the User Keys along with the Device ID's and the ID's of the message created for user. The Central Authority takes care of the retransmission of key in case of key loss.

Users are the mobile nodes who want to access the data stored at the storage node (e.g., a soldier). If a user possesses a set of attributes satisfying the access policy of the encrypted data defined by the sender, and is not revoked in any of the attributes, then he will be able to decrypt the Cipher Text and obtain the data. The users request for the User Key from Central Authority. The Central Authority will grant the User Key by verifying the device ID. The user then requests for message from the Storage Node. The message received from the Storage Node will be in the form of Cipher Text. The User has to decrypt the message only if the Key entered by the User matches with the User Key generated by the Admin. Each device can view the message only once. Once the message has been seen by the user, the message access will be removed for the particular user.

Storage Node is an entity that stores data from senders and provide corresponding access to users. It may be mobile or static. Similar to the previous schemes, we also assume the storage node to be semi-trusted that is honest-but-curious. Since the key authorities are semi-trusted, they should be deterred from accessing plaintext of the data in the storage node; meanwhile, they should be still able to issue secret keys to users.

3.2 ALGORITHM

3.2.1 RSA Algorithm

The RSA algorithm is based on public key cryptography that was named after Ron Rivest, Adi Shamir, and Leonard Adleman. It is based on a property of positive integers. Encryption and decryption are carried out using two different keys. The two keys in such a key pair are referred to as the public key and the private key.

When n is a product of two primes, in arithmetic operations modulo n , the exponents behave modulo the totient $\phi(n)$ of n .

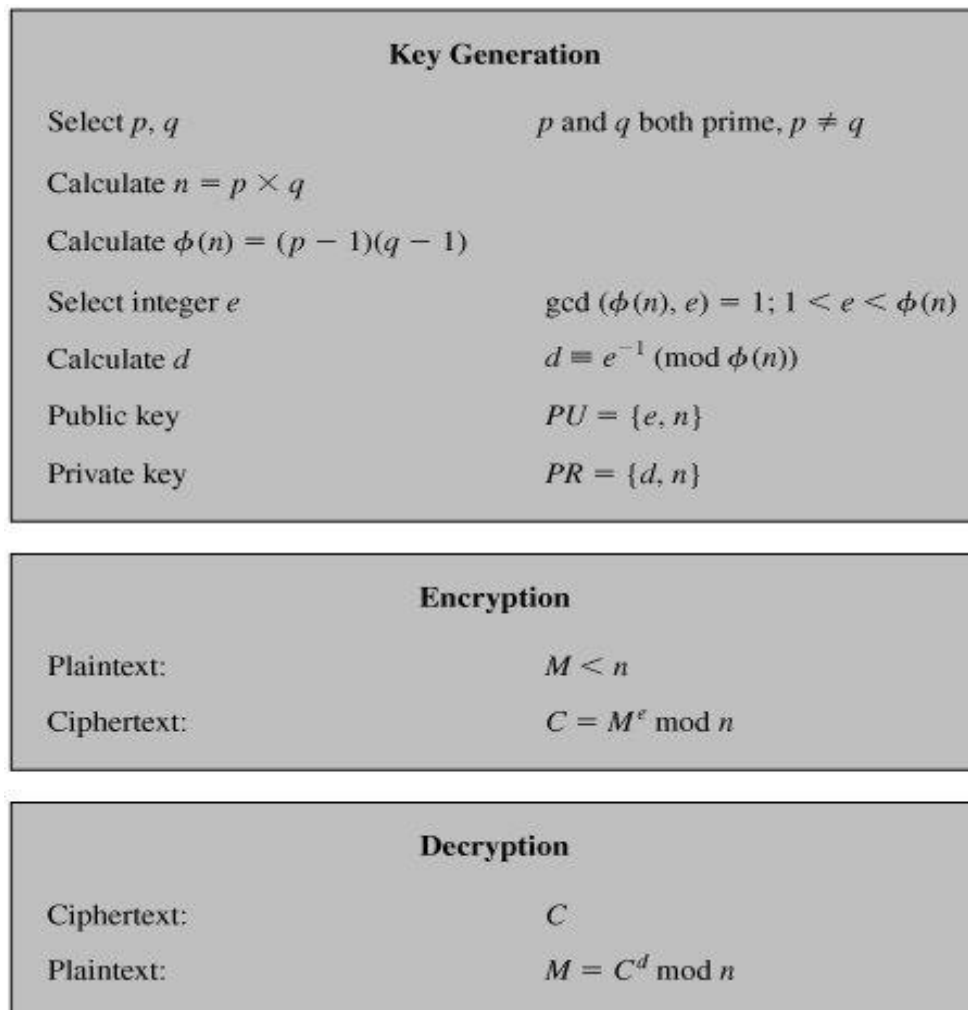


Fig 4 : RSA Key Generation

Considering arithmetic modulo n , let's say that e is an integer that is co-prime to the totient $\phi(n)$ of n . Further, say that d is the multiplicative inverse of e modulo $\phi(n)$. These definitions of the various symbols are listed below for convenience:

n = a modulus for modular arithmetic

$\phi(n)$ = the totient of n

e = an integer that is relatively prime to $\phi(n)$

[This guarantees that e will possess a multiplicative inverse modulo $\phi(n)$]

d = an integer that is the multiplicative inverse of e modulo $\phi(n)$

COMPUTATIONAL STEPS FOR KEY GENERATION IN RSA CRYPTOGRAPHY

The computational steps for key generation are

1. Generate two different primes' p and q .
2. Calculate the modulus $n = p \times q$.
3. Calculate the totient $\phi(n) = (p - 1) \times (q - 1)$.
4. Select for public exponent an integer e such that $1 < e < \phi(n)$ and $\gcd(\phi(n), e) = 1$.
5. Calculate for the private exponent a value for d such that $d = e^{-1} \bmod \phi(n)$.
6. Public Key = $[e, n]$
7. Private Key = $[d, n]$

3.2.2 AES Algorithm

- AES is a block cipher with a block length of 128 bits.
- AES allows for three different key lengths: 128, 192, or 256 bits.
- Encryption consists of 10 rounds of processing for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

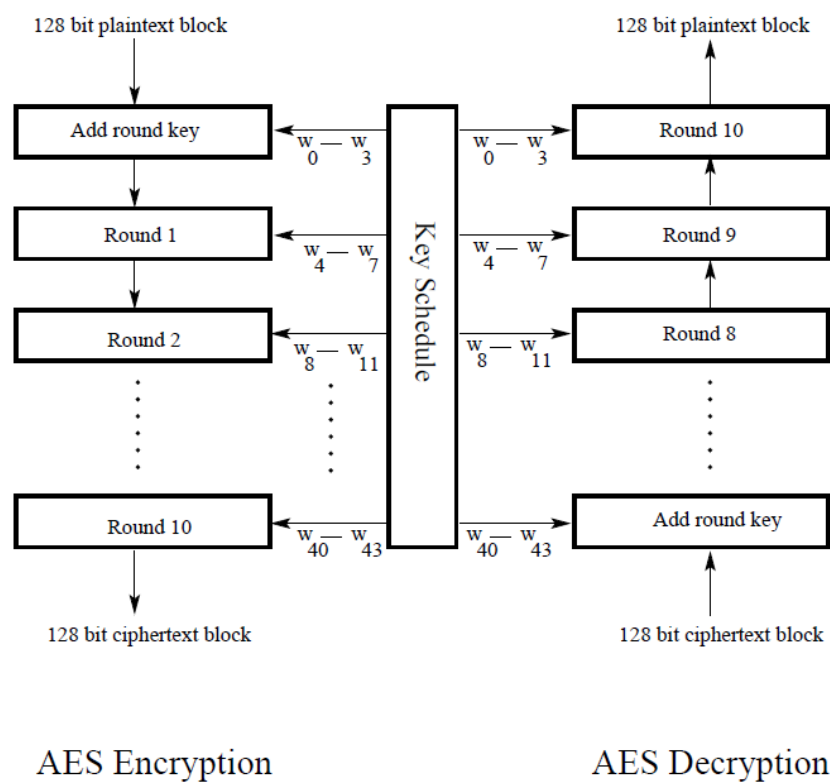


Fig 5 : AES Algorithm

- Except for the last round in each case, all other rounds are identical.
- Each round of processing includes one single-byte based substitution step, a row-wise permutation step, a column-wise mixing step, and the addition of the round key. The order in which these four steps are executed is different for encryption and decryption.
- To appreciate the processing steps used in a single round, it is best to think of a 128-bit block as consisting of a 4×4 matrix of bytes, arranged as follows:

$$\begin{bmatrix} \text{byte}_0 & \text{byte}_4 & \text{byte}_8 & \text{byte}_{12} \\ \text{byte}_1 & \text{byte}_5 & \text{byte}_9 & \text{byte}_{13} \\ \text{byte}_2 & \text{byte}_6 & \text{byte}_{10} & \text{byte}_{14} \\ \text{byte}_3 & \text{byte}_7 & \text{byte}_{11} & \text{byte}_{15} \end{bmatrix}$$

- Therefore, the first four bytes of a 128-bit input block occupy the first column in the 4×4 matrix of bytes. The next four bytes occupy the second column, and so on.
- The 4×4 matrix of bytes shown above is referred to as the state array in AES.
- AES also has the notion of a word. A word consists of four bytes, that is 32 bits. Therefore, each column of the state array is a word, as is each row.
- Each round of processing works on the input state array and produces an output state array.
- Assuming a 128-bit key, the key is also arranged in the form of a matrix of 4×4 bytes. As with the input block, the first word from the key fills the first column of the matrix, and so on.
- The four column words of the key matrix are expanded into a schedule of 44 words. Each round consumes four words from the key schedule.

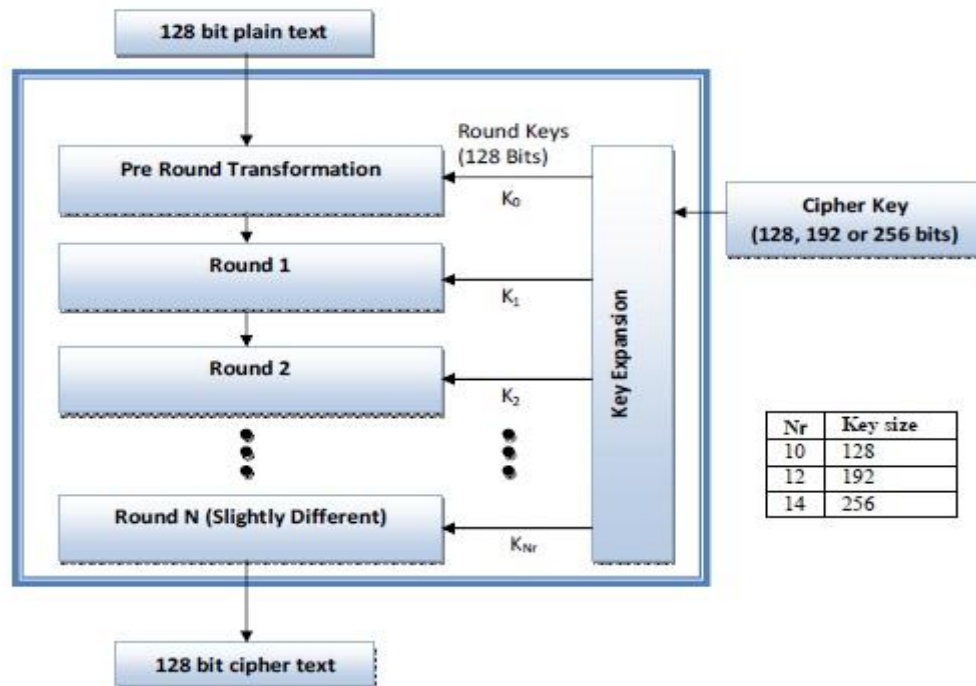


Fig 7 : AES Encryption

For decryption, each round consists of the following four steps:

- 1) Inverse shift rows,
- 2) Inverse substitute bytes,
- 3) Add round key,
- 4) Inverse mix columns. The third step consists of XORing the output of the previous two steps with four words from the key schedule.

The last round for encryption does not involve the “Mix columns” step. The last round for decryption does not involve the “Inverse mix columns” step.

3.3 DATA FLOW DIAGRAM

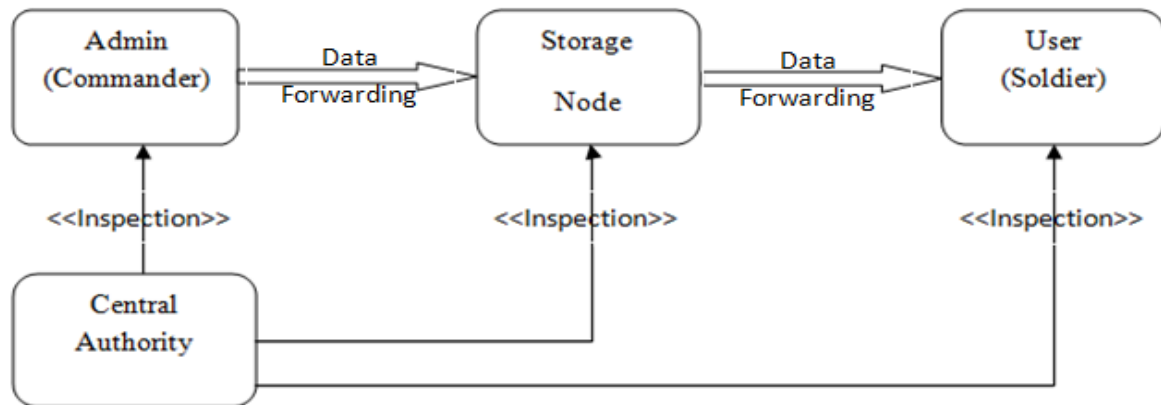


Fig 8 : Data Flow Diagram

3.4 UML DIAGRAMS

The Unified Modelling Language (UML) is a general-purpose modelling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system.

3.4.1 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. The two main components of use case diagram are the use cases and the actors. Use case diagram displays the relationship among the actors and the use cases.

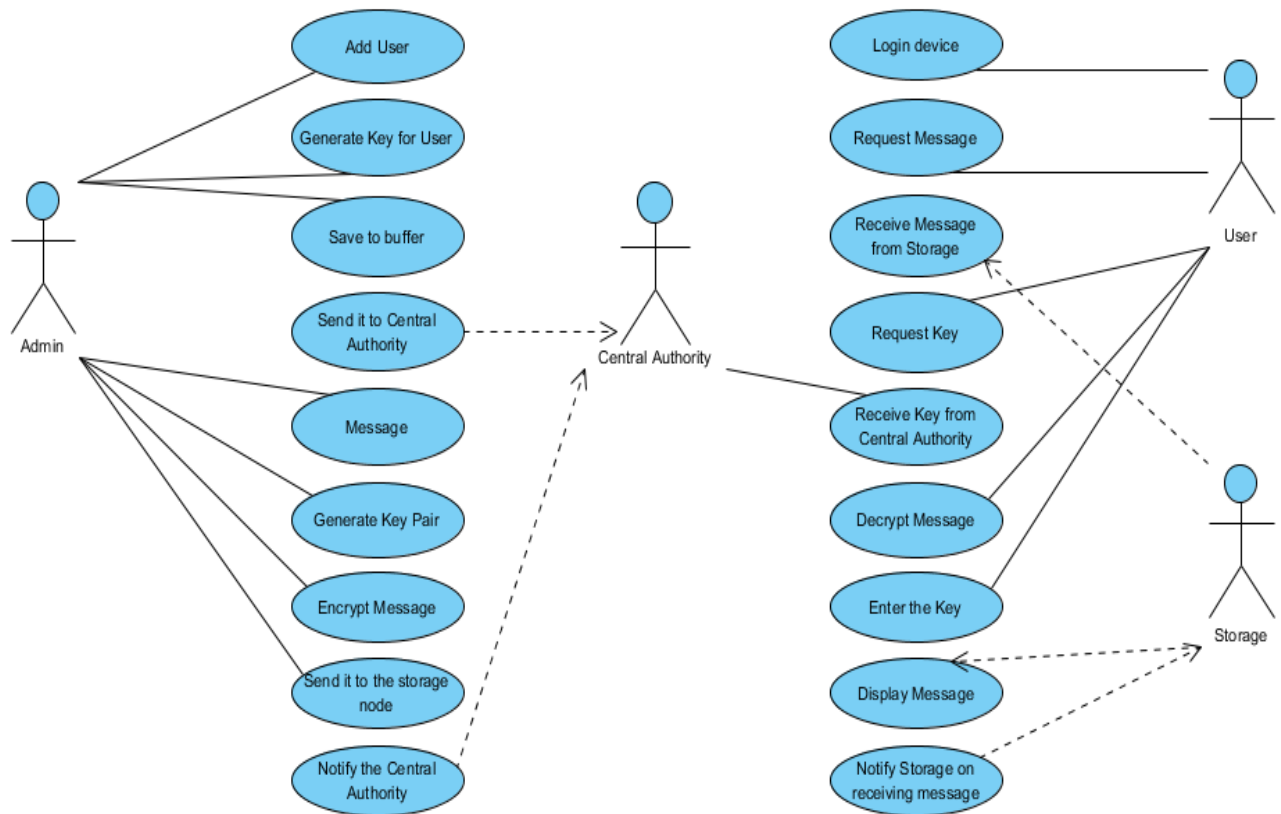


Fig 9 : Use Case Diagram

DOCUMENTATION OF USE CASE DIAGRAM

- The actors in this use case diagram are Sender, Central Authority, Storage Node and the Users.
- The Admin logs in to the system.
- The Admin creates User profile by adding the Device ID, Batch No and Name.
- A User Key will be generated by the Admin for the particular user.
- The user profile will be saved along with the User Key generated.
- The User Key will be sent to the Central authority.
- Now the Admin will enter the message to be conveyed.
- Admin generates a Private and Public Key pair to encrypt and decrypt the message.
- The message is encrypted with Public Key and Sent to the Storage Node.
- The Central Authority will be notified about the message created.
- The corresponding Private Key of the Public Key will be encrypted with the User Key and sent to User Device.

- The User Logs in to the Device with Device ID and Password. The User gets access into device if the Device ID and Password matches.
- The User requests for the message from Storage Node.
- The Storage Node sends the message waiting for the respective device. The message is in the form of Cipher Text.
- The User then requests for the User Key from Central Authority.
- The Central Authority grants the User Key for the respective user.
- The User enters the User Key received and tries to decrypt the message.
- If the entered User Key matches with the Key created by the admin for the corresponding device, then the message is displayed to the user.
- The Storage Node will be notified once the message has been retrieved by the user.
- Each User will be able to view the message only once. The access will be removed for the device once the message has been seen.

3.4.2 Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

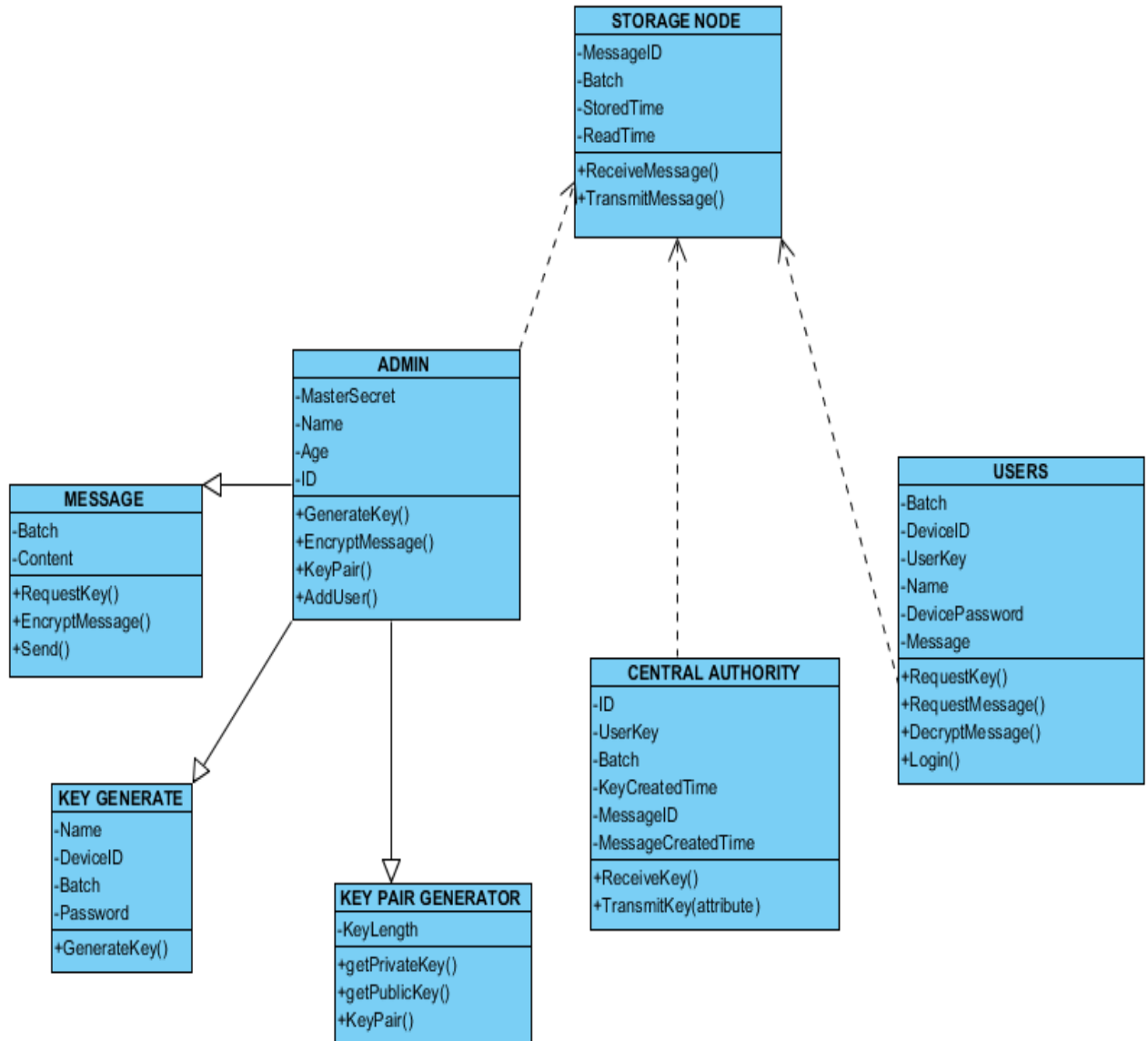


Fig 10 : Class Diagram

3.4.3 Sequence Diagram

Sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between a numbers of lifelines. Sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.

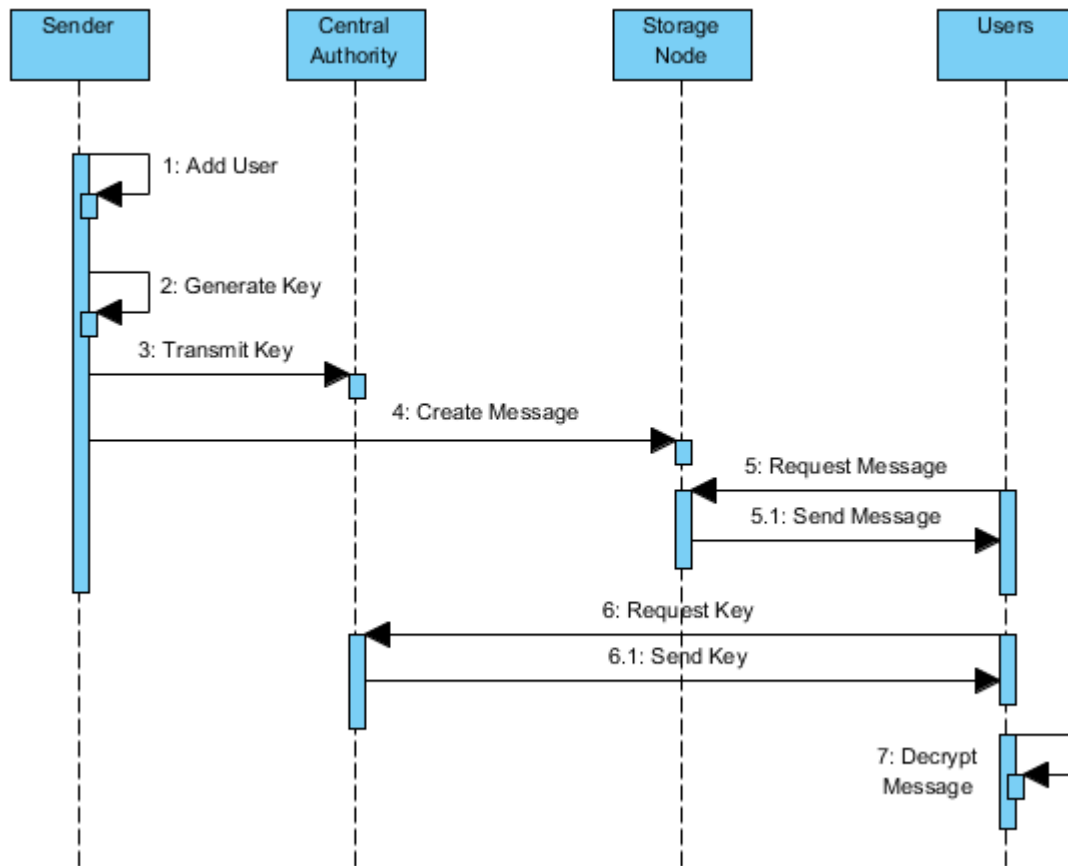


Fig 11: Sequence Diagram

3.4.4 Activity Diagram

Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deal with all types of flow control by using different elements like fork, join etc.

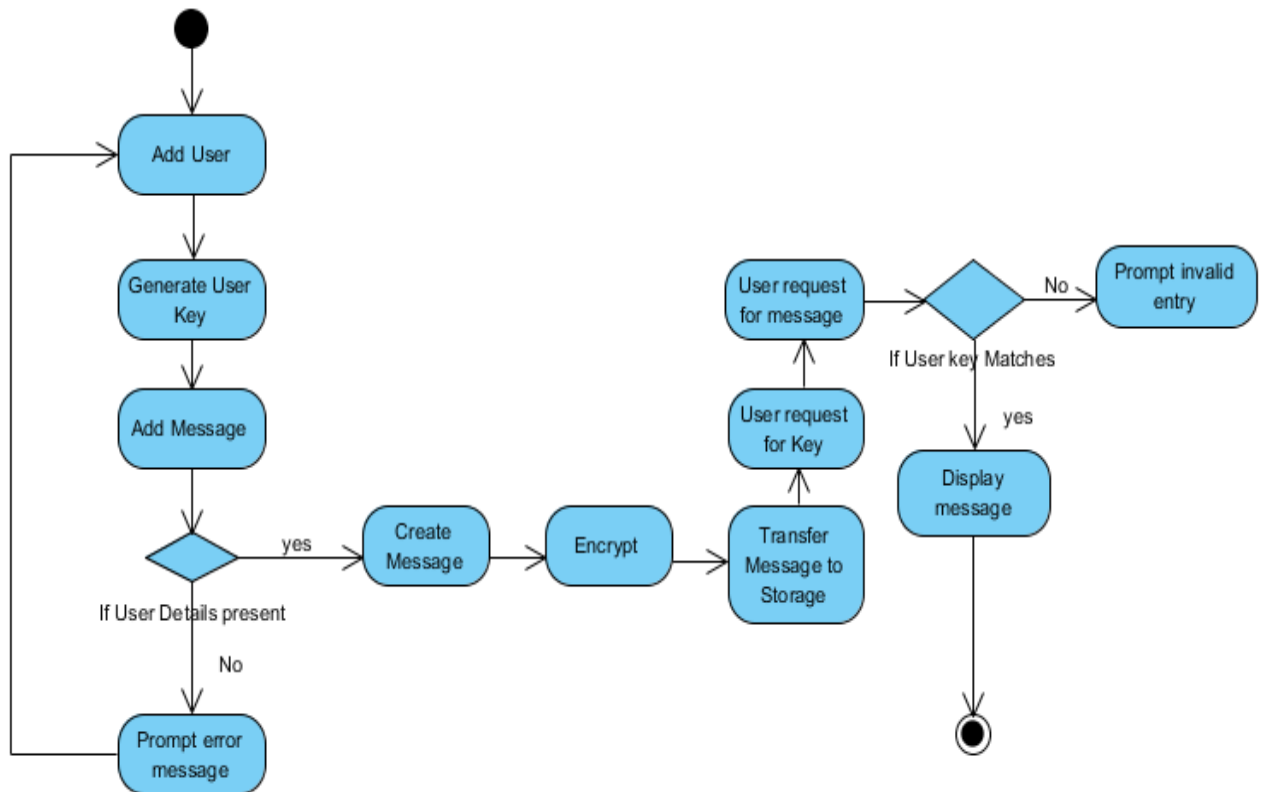


Fig 12 : Activity Diagram

3.4.5 State Diagram

UML state machine diagrams depict the various states that an object may be in and the transitions between those states. A state represents a stage in the behaviour pattern of an object, and like UML activity diagrams it is possible to have initial states and final states. An initial state, also called a creation state, is the one that an object is in when it is first created, whereas a final state is one in which no transitions lead out of. A transition is a progression from one state to another and will be triggered by an event that is either internal or external to the object.

3.4.5.1 State Diagram for Sender

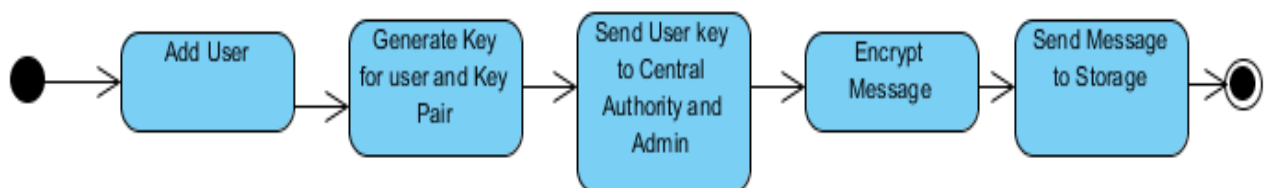


Fig 13.1 State Diagram for Sender

3.4.5.2 State Diagram for User

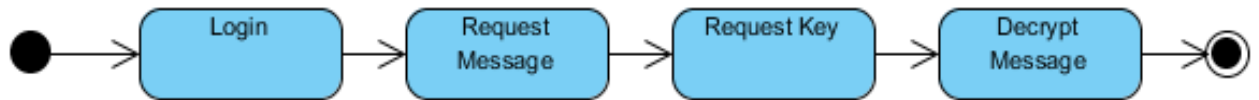


Fig 13.2 State Diagram for User

3.4.6 Component Diagram

A component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. Component diagrams illustrate the pieces of software, embedded controllers, etc., that will make up a system. A component diagram has a higher level of abstraction than a Class Diagram - usually a component is implemented by one or more classes (or objects) at runtime.

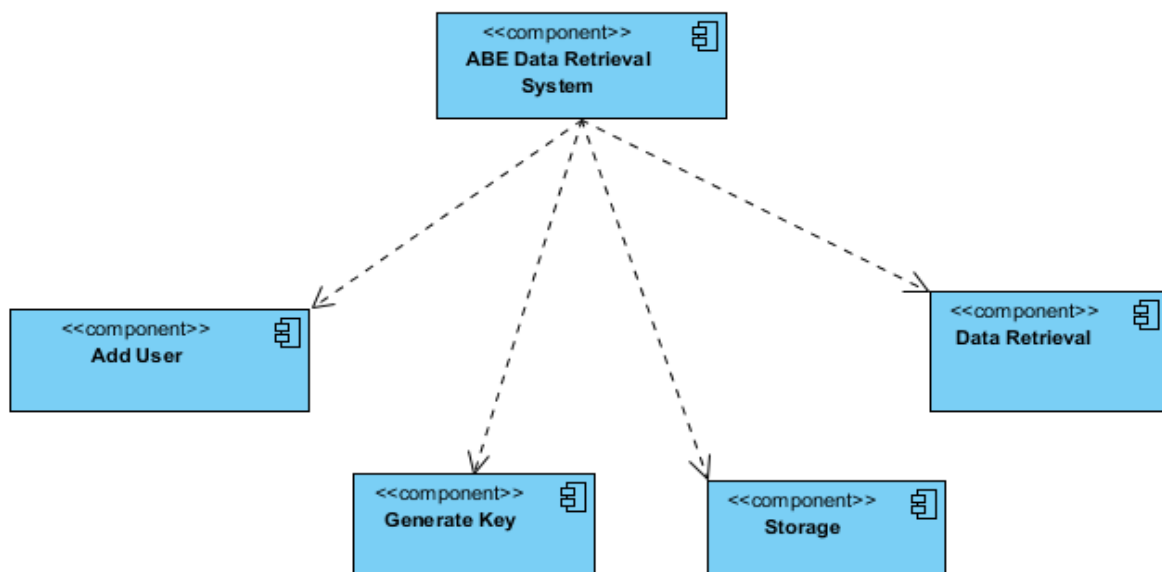


Fig 14 : Component Diagram

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 LIST OF MODULES

1. User and Message creation
2. Key transmission
3. Storage
4. Retrieving message

4.2 MODULE DESCRIPTION

4.2.1 User and Message Creation

This module is responsible for User and Message creation. The admin enter the values of the attributes that the user must possess and generates a random one-time key for User as User Key. The admin creates the required number of users and generates User Keys. The User properties are saved into a buffer. The admin then enters the Message to be sent for the corresponding devices. Then the admin generates a Private and Public key pair using RSA algorithm. The message is then encrypted with the generated RSA Public key. The RSA Private Key is then encrypted with User Key using AES algorithm. Here we follow Symmetric Key Encryption for encryption and decryption of RSA Private Key and Asymmetric Key Encryption for encryption and decryption of Message. The encrypted message is sent to the Storage node. An ID will be assigned to each message created. The message can be decrypted by the corresponding RSA Private Key at the user side.

Symmetric Encryption

Symmetric encryption is the oldest and best-known technique. A secret key, which can be a number, a word, or just a string of random letters, is applied to the text of a message to change the content in a particular way. This might be as simple as shifting each letter by a number of places in the alphabet. As long as both sender and recipient know the secret key, they can encrypt and decrypt all messages that use this key.

Asymmetric Encryption

The problem with secret keys is exchanging them over the Internet or a large network while preventing them from falling into the wrong hands. Anyone who knows the secret key can decrypt the message. One answer is asymmetric encryption, in which there are two related keys--a key pair. A public key is made freely available to anyone who might want to send you a message. A second, private key is kept secret, so that only you know it.

Any message (text, binary files, or documents) that are encrypted by using the public key can only be decrypted by applying the same algorithm, but by using the matching private key. Any message that is encrypted by using the private key can only be decrypted by using the matching public key. This means that you do not have to worry about passing public keys over the Internet (the keys are supposed to be public).

RSA Algorithm

RSA is one of the first practicable public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem.

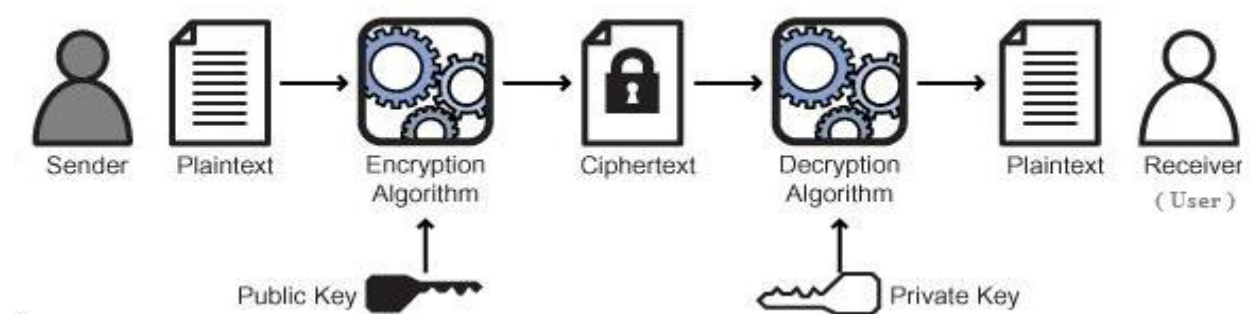


Fig 15 : RSA

There are three ways to use Private and Public Keys

- i) Manually enter two prime numbers and perform factorization to generate Private and Public Key pair.

- ii) Have predefined sets of Private and Public Key in Key Store and use it while performing Encryption and Decryption.
- iii) Randomly select two prime numbers and perform factorization to generate Private and Public Key pair.

Here the Admin uses iii) method to generate Private and Public Key pair.

AES Algorithm

AES is based on a design principle known as a substitution-permutation network, combination of both substitution and permutation, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits.

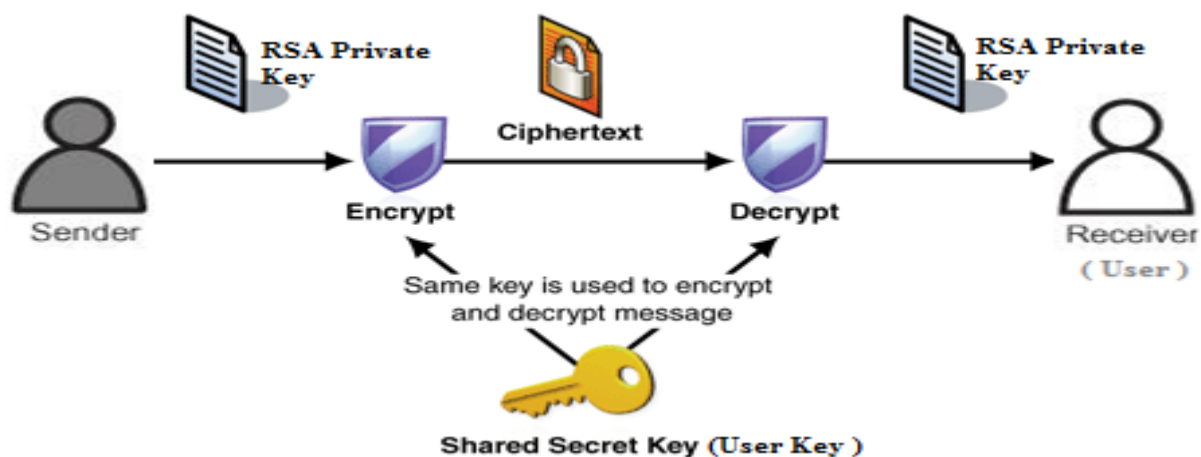


Fig 16 : AES

The Encryption algorithm takes as input the RSA Private Key and encrypts it with the User Key. The Decryption algorithm takes as input the User Key. Decryption takes place only if the entered key matches with the User Key used in Encryption. Here the Sender and receiver share a secret key, which is the User Key.

4.2.2 Key Transmission

The generated User Key is then transferred to the Central Authority. The Central Authority grants the User with respective User Key on request. The Central Authority holds information about the Key creation time, Message Creation time, the Batch for which the message was generated and the corresponding keys of Users. Here the Central Authority is not explicitly involved in the key generation process. So it solves the Key Escrow problem where if the Central Authority is compromised then the information is compromised. The Central Authority is responsible for the re-transmission of key in case of Loss in the transmission line.

4.2.3 Storage

The Sender encrypts the data with a policy. The policy includes the attributes that must be possessed by a user in order to decrypt the data. The admin sends the encrypted data to the Storage Node. Storage Node is an entity that stores data from senders and provide corresponding access to users. It may be mobile or static. Since the key authorities are semi-trusted, they should be deterred from accessing plaintext of the data in the storage node; meanwhile, they should be still able to issue secret keys to users.

4.2.4 Retrieving Message

Each user in the Military network is given a device. Each device is associated with a device ID, Region and Batch. The User Logs in the Device through Batch number and device password. The user then requests for message from the Storage Node. The storage Node sends encrypted message to the Device in response. The message is in the form of Cipher Text. The user then requests for the User Key. The central Authority renders the user with a User Key. The User enters the key to decrypt the message. If the key matches with the key generated by admin, then AES decryption is performed on Private Key and the message is decrypted with the Private Key. The message is finally displayed to the user.

CHAPTER 5

TESTING, PERFORMANCE AND COMPARATIVE ANALYSIS

5.1 TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test.

5.1.1 Unit Testing

Unit testing is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output. In procedural programming a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class.

5.1.2 Integration Testing

Integration testing is a logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit. In a realistic scenario, many units are combined into components, which are in turn aggregated into even larger parts of the program. The idea is to test combinations of pieces and eventually expand the process to test your modules with those of other groups. Eventually all the modules making up a process are tested together. Beyond that, if the program is composed of more than one process, they should be tested in pairs rather than all at once.

5.1.3 Functional Testing

Functional testing means testing the application against business requirements. Functional testing is executed using the functional specifications given by the client or by the design specifications according to use cases given by the design team. Role of functional testing is to validating the behaviour of an application.

Functional testing is more important because it always verifies that your system is fixed for release. The functional tests define your working system in a useful manner. In functional testing tester has to validate the application to see that all specified requirements of the client whatever we have said in SRS have been incorporated or not.

5.1.4 Manual Testing

Manual Testing is a process carried out to find the defects. In this method the tester plays an important role as end user and verify all features of the application to ensure that the behaviour of the application. The Manual Testing is very basic type of testing which helps to find the bugs in the application under test. It is preliminary testing, must be carried out prior to start automating the test cases and also needs to check the feasibility of automation testing. The Test Plan is created & followed by the tester to ensure that the comprehensiveness of testing while executing the test cases manually without using automation testing tool.

5.1.5 Test Cases

Test Case No.	Task Description	Expected Outcome	Actual Outcome	Test Result
1	Sender enters username or password	Enters into Admin profile if username and password are correct. Otherwise send Error Message	Proceeds further	If (username and password are correct) Then Passed Else Failed
2	Sender enters the details of user to be added and generate key	Key generated with respect to attributes of user	Key generated and sent to CA	Passed
3	Sender enters the message and the	Message Created	Plain Text is created	Passed

	batch for which to send it			
4	Sender encrypts the message	Message Encrypted	Encrypts the Message with RSA Public key and transfer the Cipher Text to Storage Node	Passed
5	User enters the batch number and device password	Enter into user profile if batch and password are correct. Otherwise send Error Message.	Proceeds Further	If (batch and password are correct) Then Passed Else Failed
6	User requests for message	Receive Message.	Cipher Text received from Storage Node	Passed
7	User requests for Key	Received Key	Private Key is sent to User Device	Passed
8	User decrypts the message with key	Displays message if the Key is correct. Otherwise send Error Message	Decrypt the message with the RSA private key if the entered User Key is correct	If(Key is correct) Then Display message Else Display Error Message

Table No 1 : Test Cases

5.2 PERFORMANCE

The proposed scheme enables the authorized users to ensure that they are receiving the most recent version of the outsourced data. Moreover, in case of dispute regarding data integrity/ newness, Admin is able to determine the dishonest party. The data owner enforces access control for the outsourced data by combining three cryptographic techniques: broadcast encryption, Cipher Text Attribute-Based-Encryption and Key-Pair Generation. We have studied the security features of the proposed scheme. We have investigated the

overheads added by our scheme when incorporated into a storage model for *static* data with only *confidentiality* requirement. The storage overhead is $\approx 0.4\%$ of the outsourced data size, the communication overhead due to block-level dynamic changes on the data is $\approx 1\%$ of the block size, and the communication overhead due to retrieving the data is $\approx 0.2\%$ of the outsourced data size. For a large organization with 105 users, performing dynamic operations and enforcing access control add about 63 milliseconds of overhead. Therefore, important features of data storage can be supported without excessive overheads in storage, communication, and computation.

5.3 COMPARATIVE ANALYSIS

Scheme	Authority	Expressiveness	Key Escrow	Revocation
BSW[9]	Single	-	Yes	Periodic attribute revocation
HV[13]	Multiple	AND	Yes	Periodic attribute revocation
RC[14]	Multiple	AND	Yes	Immediate System level revocation
Distributed Multiple Key Authority	Multiple	Any monotone access structure	No	Immediate attribute level revocation
Proposed	Single	AND	No	Immediate System level revocation

Table No. 2 : Analysis

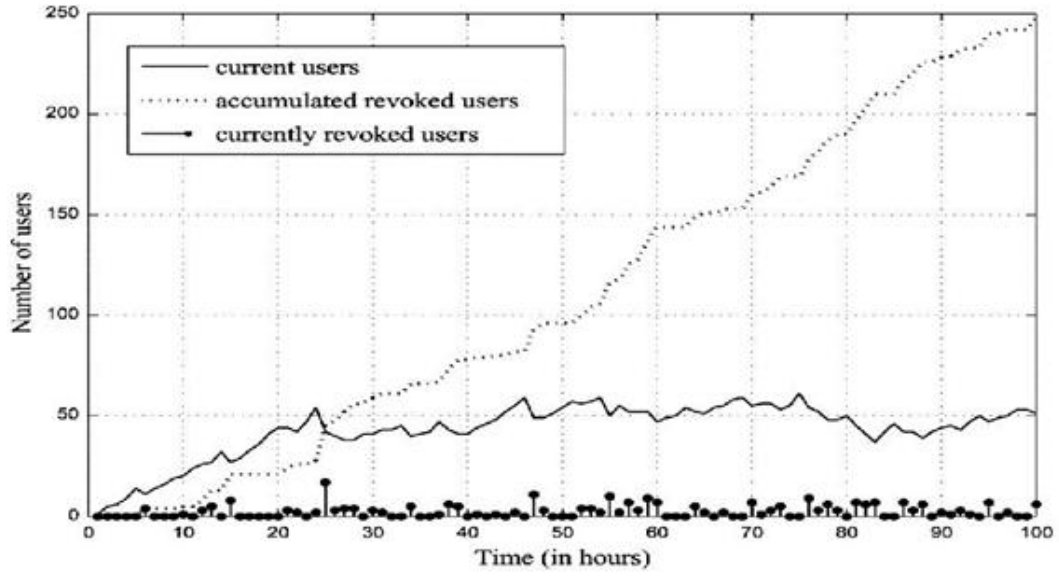


Fig. 17. Number of users in an attribute group.

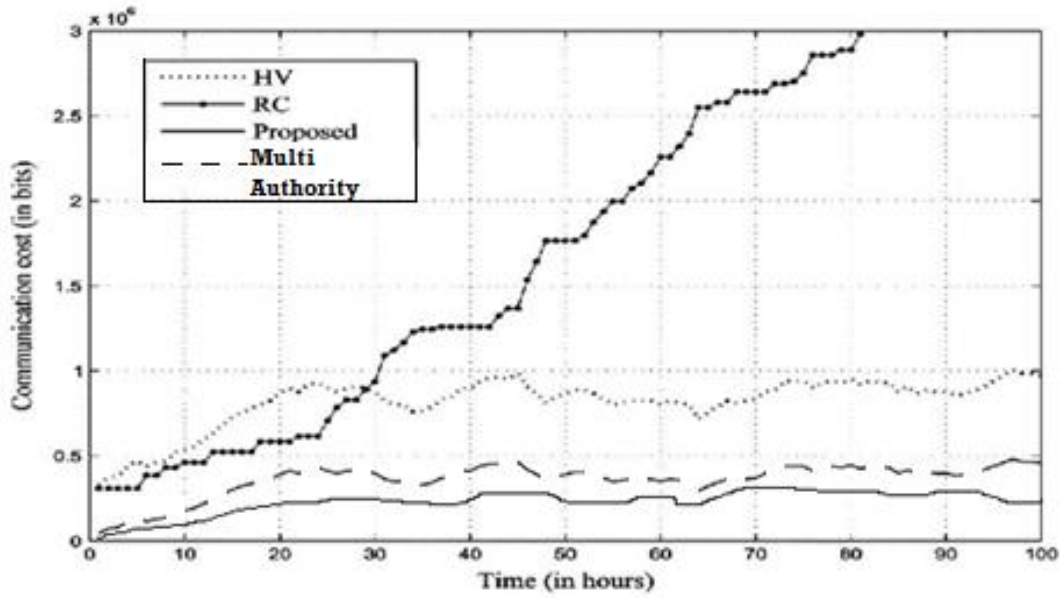


Fig. 18. Communication cost in the proposed CP-ABE systems

CHAPTER 6

CONCLUSION

DTN technologies are becoming successful solutions in military applications that allow wireless devices to communicate with each other and access the confidential information reliably by exploiting external storage nodes. CP-ABE is a scalable cryptographic solution to the access control and secure data retrieval issues. In this paper, we proposed an efficient and secure data retrieval method using CP-ABE for centralized DTNs where central authority and sender manage the key generation using automated scheme that generates random one-time-key based on attributes. This solves the problem of forward secrecy and backward secrecy. It also solves Key Escrow problem since central authority is not explicitly involved in key generation. In addition, the system promotes reliability since the central authority keeps tracking on when message was sent and received and also about key reaching the soldier in time. The inherent key escrow problem is resolved such that the confidentiality of the stored data is guaranteed even under the hostile environment where key authorities might be compromised. In addition, the fine-grained key revocation can be done for each attribute group. This ABE (Attribute-Based-Encryption) method can be further used in other hostile networks and private concerns like Hospitals, Research Centre to preserve data.

6.1 Future Work

- i) **Deal with Break-glass Access-** Sender may need to have temporary view at the data stored in Storage Node in case when the Sender System is hacked. The Sender may want to analyse the effects of the acts by intruder. Break-glass provides temporary access under emergency scenarios.
- ii) **Serialization in storage node-** By the serialization, the size of the data becomes very small which reduces the overheads required in Storage Node.
- iii) **Constant-sized keys-** Lightweight devices, such as radio frequency identification tags, have a limited storage capacity, which has become a bottleneck for many applications, especially for security applications. A novel CP-ABE scheme with constant-size decryption keys independent of the number of attributes can be used with size as small as 672 bits.

APPENDIX

Appendix 1- Sample Source Code

Sender of the Message :

```
package com.secure.ui;
import java.awt.Color;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.io.IOException;
import java.security.GeneralSecurityException;
import java.util.HashMap;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPasswordField;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.UIManager;

import org.jvnet.substance.SubstanceLookAndFeel;

import com.secure.algorithm.RSACipher;
import com.secure.bo.Message;
import com.secure.bo.User;
import com.secure.impl.AdminReceiver;
import com.secure.impl.Sender;
import com.secure.utility.SecureUtil;

public class Admin implements MouseListener {
    public static String text;
    public static User user;
    public static Message msg;
    public JFrame mainform;
    public JLabel mainLabel;
    public JLabel messageLabel,soldierLabel,exitLabel;

    //register
    public JLabel register;
    public JLabel nameLabel, batchnoLabel, deviceidLabel, passwordLabel,
    keyLabel;
    public static JTextField nameText,batchnoText,deviceidText,keyText;
    public JPasswordField passwordText;
    public JLabel save,generateUserKey;

    //sender
    public JLabel sender;
    public JLabel sendernoLabel,contentLabel;
    public JTextField sendernoText;
    public JTextArea contentText;
    public JLabel keyRequest,encrypt,send;
```

```

        public Sender sendObj=new Sender();
public static HashMap<String, User> userMap=new HashMap<String, User>();
public static HashMap<Integer, Message> msgMap=new HashMap<Integer, Message>();
        public Admin() {
            new SecureUtil();
            new AdminReceiver(SecureUtil.ADMIN_PORT);
        }

        static {
            try {
                SubstanceLookAndFeel
.setCurrentWatermark("org.jvnet.substance.watermark.SubstanceBinaryWatermark");
                SubstanceLookAndFeel
.setCurrentTheme("org.jvnet.substance.theme.SubstanceInvertedTheme");
                SubstanceLookAndFeel
.setCurrentGradientPainter("org.jvnet.substance.painter.SpecularGradientPainter");
                SubstanceLookAndFeel
.setCurrentButtonShaper("org.jvnet.substance.button.ClassicButtonShaper");
                UIManager.setLookAndFeel(new SubstanceLookAndFeel());
            } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
            }
        }

public void UI(){

    ImageIcon icon=new ImageIcon("images/home.jpg");
    mainform=new JFrame();

    mainLabel=new JLabel(icon);
    mainLabel.setLayout(null);

    messageLabel=new JLabel();
    messageLabel.setBounds(3,400,100,30);
    messageLabel.setBorder(BorderFactory.createLineBorder(Color.GRAY));
    messageLabel.addMouseListener(this);
    mainLabel.add(messageLabel);

        JLabel msg1=new JLabel(new ImageIcon("images/message.png"));
        msg1.setBounds(3,3,24,24);
        messageLabel.add(msg1);

        JLabel msg2=new JLabel("Message");
        msg2.setBounds(35,3,100,24);
        msg2.setForeground(new Color(66,192,251));
        messageLabel.add(msg2);

    soldierLabel=new JLabel();
    soldierLabel.setBounds(136,400,120,30);
    soldierLabel.setBorder(BorderFactory.createLineBorder(Color.GRAY));
    soldierLabel.addMouseListener(this);
    mainLabel.add(soldierLabel);

        JLabel sol1=new JLabel(new ImageIcon("images/soldier.jpg"));

```

```

        sol1.setBounds(3,3,24,24);
        soldierLabel.add(sol1);

        JLabel sol2=new JLabel("Add soldier");
        sol2.setBounds(35,3,100,24);
        sol2.setForeground(new Color(66,192,251));
        soldierLabel.add(sol2);

        exitLabel=new JLabel("  Exit  ");
        exitLabel.setBounds(286,400,50,30);
        exitLabel.setBorder(BorderFactory.createLineBorder(Color.GRAY));
        exitLabel.setForeground(new Color(66,192,251));
        exitLabel.addMouseListener(this);
        mainLabel.add(exitLabel);

        mainform.add(mainLabel);
        mainform.setSize(600,450);
        mainform.setLocation(300,200);
        mainform.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainform.setUndecorated(true);
        mainform.setVisible(true);
    }

    public JLabel registerUI(){
        register=new JLabel(new ImageIcon("images/back.jpg"));
        register.setBounds(50,40,500,350);
        register.setBorder(BorderFactory.createLineBorder(Color.GRAY));

        JLabel title=new JLabel(new ImageIcon("images/title.png"));
        title.setBounds(1,1,499,50);
        register.add(title);

        nameLabel=new JLabel("Name");
        nameLabel.setBounds(100,70,100,25);
        register.add(nameLabel);

        nameText=new JTextField();
        nameText.setBounds(200,70,175,25);
        register.add(nameText);

        batchnoLabel=new JLabel("Batch No");
        batchnoLabel.setBounds(100,120,100,25);
        register.add(batchnoLabel);

        batchnoText=new JTextField();
        batchnoText.setBounds(200,120,175,25);
        register.add(batchnoText);

        deviceidLabel=new JLabel("Device ID");
        deviceidLabel.setBounds(100,170,100,25);
        register.add(deviceidLabel);

        deviceidText=new JTextField();
        deviceidText.setBounds(200,170,175,25);
        register.add(deviceidText);
    }

```

```

passwordLabel=new JLabel("Password");
passwordLabel.setBounds(100,220,100,25);
register.add(passwordLabel);

passwordText=new JPasswordField();
passwordText.setBounds(200,220,175,25);
register.add(passwordText);

keyLabel=new JLabel("Key");
keyLabel.setBounds(100,270,100,25);
register.add(keyLabel);

keyText=new JTextField();
keyText.setBounds(200,270,175,25);
keyText.setEditable(false);
register.add(keyText);

generateUserKey=new JLabel();
generateUserKey.setBounds(150,310,120,30);

generateUserKey.setBorder(BorderFactory.createLineBorder(Color.BLACK));
register.add(generateUserKey);
generateUserKey.addMouseListener(this);

        JLabel key1=new JLabel(new ImageIcon("images/key.png"));
        key1.setBounds(3,3,24,24);
        generateUserKey.add(key1);

        JLabel key2=new JLabel("Generate Key");
        key2.setForeground(Color.BLACK);
        key2.setBounds(30,0,90,30);
        generateUserKey.add(key2);

save=new JLabel("Save");
save.setBounds(300,310,100,30);
save.setBorder(BorderFactory.createLineBorder(Color.BLACK));
register.add(save);
save.addMouseListener(this);

        JLabel save1=new JLabel(new ImageIcon("images/save.png"));
        save1.setBounds(3,3,24,24);
        save.add(save1);

        JLabel save2=new JLabel("Save");
        save2.setForeground(Color.BLACK);
        save2.setBounds(40,0,70,30);
        save.add(save2);

return(register);
}

public JLabel senderUI(){
    sender=new JLabel(new ImageIcon("images/back.jpg"));
    sender.setBounds(50,40,500,350);

```

```

sender.setBorder(BorderFactory.createLineBorder(Color.GRAY));

JLabel title=new JLabel(new ImageIcon("images/title1.png"));
title.setBounds(1,1,499,50);
sender.add(title);

sendernoLabel=new JLabel("Batch No");
sendernoLabel.setBounds(100,70,100,25);
sender.add(sendernoLabel);

sendernoText=new JTextField();
sendernoText.setBounds(200,70,175,25);
sender.add(sendernoText);

contentLabel=new JLabel("Content");
contentLabel.setBounds(100,120,100,25);
sender.add(contentLabel);

contentText=new JTextArea();
contentText.setBounds(200,120,175,100);
sender.add(contentText);

keyRequest=new JLabel();
keyRequest.setBounds(150,250,120,30);
keyRequest.setBorder(BorderFactory.createLineBorder(Color.BLACK));
sender.add(keyRequest);
keyRequest.addMouseListener(this);

        JLabel key1=new JLabel(new ImageIcon("images/key.png"));
        key1.setBounds(3,3,24,24);
        keyRequest.add(key1);

        JLabel key2=new JLabel("Request Key");
        key2.setForeground(Color.BLACK);
        key2.setBounds(30,0,90,30);
        keyRequest.add(key2);

encrypt=new JLabel();
encrypt.setBounds(300,250,100,30);
encrypt.setBorder(BorderFactory.createLineBorder(Color.BLACK));
encrypt.addMouseListener(this);
sender.add(encrypt);

        JLabel e1=new JLabel(new ImageIcon("images/encrypt.png"));
        e1.setBounds(3,3,24,24);
        encrypt.add(e1);

        JLabel e2=new JLabel("Encrypt");
        e2.setForeground(Color.BLACK);
        e2.setBounds(40,0,70,30);
        encrypt.add(e2);

send=new JLabel();
send.setBounds(220,310,100,30);
send.setBorder(BorderFactory.createLineBorder(Color.BLACK));

```

```

send.addMouseListener(this);
sender.add(send);

        JLabel s1=new JLabel(new ImageIcon("images/send.png"));
        s1.setBounds(3,3,24,24);
        send.add(s1);

        JLabel s2=new JLabel("Send");
        s2.setForeground(Color.BLACK);
        s2.setBounds(40,0,70,30);
        send.add(s2);

    return(sender);
}

public static void main(String args[]){
    new Admin().UI();
}

@SuppressWarnings("deprecation")
@Override
public void mouseClicked(MouseEvent e) {
    // TODO Auto-generated method stub

    Object obj=e.getSource();
    if(obj==exitLabel){
        System.exit(0);
    }else if(obj==messageLabel){
        if(register!=null){
            register.setVisible(false);
        }
        mainLabel.add(senderUI());
        mainLabel.repaint();
    }else if(obj==soldierLabel){
        if(sender!=null){
            sender.setVisible(false);
        }
        mainLabel.add(registerUI());
        mainLabel.repaint();
    }else if(obj==generateUserKey){
        if(nameText.getText().trim().equalsIgnoreCase("") ||
        batchnoText.getText().trim().equalsIgnoreCase("") ||
        passwordText.getText().trim().equalsIgnoreCase("") ||
        deviceidText.getText().trim().equalsIgnoreCase("")){
            SecureUtil.msgg("Empty field not allowed");
        }else{
            User user=new User();
            user.name=nameText.getText();
            user.batchno=batchnoText.getText();
            user.password=passwordText.getText();
            user.deviceId=deviceidText.getText();
            user.port=SecureUtil.generatePort();
            user.type="admin";

            sendObj.sendUserKeyReq(SecureUtil.KEY_IP,SecureUtil.KEY_PORT,user);

```

```

    }
    }else if(obj==save){
        String res=SecureUtil.save(Admin.user);
        if(res.equalsIgnoreCase("success")){
            SecureUtil.msg("Successfully Saved");
        }
    }else if(obj==keyRequest){
        if(batchnoText.getText().trim().equalsIgnoreCase("") ||
contentText.getText().trim().equalsIgnoreCase("")){
            SecureUtil.msg("Empty field not allowed");
        }else{
            Message msg=new Message();
            msg.batchno=batchnoText.getText();
            msg.content=contentText.getText();
            msg.id=msgMap.size()+1;
            msg.userkey=userMap.get(msg.batchno).key;
            msg.type="admin";
            sendObj.send(SecureUtil.KEY_IP,SecureUtil.KEY_PORT, msg);
        }
    }else if(obj==encrypt){
        try {
            RSACipher cip=new RSACipher();
            Admin.msg.Cipher
TextContent=cip.encrypt(Admin.msg.content,Admin.msg.publicKey,
SecureUtil.TRANSFORMATION,SecureUtil.ENCODING);
            Admin.msgMap.put(Admin.msg.id,Admin.msg);
            SecureUtil.msg("Encrypted successfully");
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (GeneralSecurityException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }else if(obj==send){
        Admin.msg.type="admin";
sendObj.send(SecureUtil.STORAGE_IP,SecureUtil.STORAGE_PORT,Admin.msg);
        SecureUtil.msg("Message sent successfully");
    }
}

@Override
public void mouseEntered(MouseEvent e) {
    // TODO Auto-generated method stub
}
@Override
public void mouseExited(MouseEvent e) {
    // TODO Auto-generated method stub
}
@Override
public void mousePressed(MouseEvent e) {
    // TODO Auto-generated method stub
}

```



```

        @Override
        public void mouseReleased(MouseEvent e) {
            // TODO Auto-generated method stub
        }
    }
}

```

Key Pair Generator :

```

package com.secure.algorithm;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.GeneralSecurityException;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;

public final class RSAKeyPair {
    private int keyLength;
    private PrivateKey privateKey;
    private PublicKey publicKey;

    public RSAKeyPair(int keyLength)throws GeneralSecurityException {
        this.keyLength = keyLength;
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
        keyPairGenerator.initialize(this.keyLength);
        KeyPair keyPair = keyPairGenerator.generateKeyPair();
        privateKey = keyPair.getPrivate();
        publicKey = keyPair.getPublic();
    }

    public final PrivateKey getPrivateKey() {
        return privateKey;
    }

    public final PublicKey getPublicKey() {
        return publicKey;
    }

    public final void toFileSystem(String privateKeyPathName, String publicKeyPathName)
        throws IOException {
        FileOutputStream privateKeyOutputStream = null;
        FileOutputStream publicKeyOutputStream = null;

        try {

            File privateKeyFile = new File(privateKeyPathName);
            File publicKeyFile = new File(publicKeyPathName);

            privateKeyOutputStream = new FileOutputStream(privateKeyFile);
            privateKeyOutputStream.write(privateKey.getEncoded());

            publicKeyOutputStream = new FileOutputStream(publicKeyFile);

```

```

        publicKeyOutputStream.write(publicKey.getEncoded());
    } catch(IOException ioException) {
        throw ioException;
    } finally {
        try {

            if (privateKeyOutputStream != null) {
                privateKeyOutputStream.close();
            }
            if (publicKeyOutputStream != null) {
                publicKeyOutputStream.close();
            }

        } catch(IOException ioException) {
            throw ioException;
        }
    }
}
}
public static void main(String args[]){ }
}

```

RSA Encryption and Decryption :

```

package com.secure.algorithm;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.IOUtils;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.GeneralSecurityException;
import java.security.KeyFactory;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import javax.crypto.Cipher;

public class RSACipher {

    public String encrypt(String rawText, String publicKeyPath, String transformation, String
encoding)
        throws IOException, GeneralSecurityException {

        X509EncodedKeySpec x509EncodedKeySpec = new
X509EncodedKeySpec(IOUtils.toByteArray(new FileInputStream(publicKeyPath)));

        Cipher cipher = Cipher.getInstance(transformation);
        cipher.init(Cipher.ENCRYPT_MODE,
KeyFactory.getInstance("RSA").generatePublic(x509EncodedKeySpec));

        return Base64.encodeBase64String(cipher.doFinal(rawText.getBytes(encoding)));
    }

    public String decrypt(String Cipher Text, String privateKeyPath, String transformation, String
encoding)
        throws IOException, GeneralSecurityException {

```

```

        PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new
        PKCS8EncodedKeySpec(IOUSils.toByteArray(new FileInputStream(privateKeyPath)));

        Cipher cipher = Cipher.getInstance(transformation);
        cipher.init(Cipher.DECRYPT_MODE,
        KeyFactory.getInstance("RSA").generatePrivate(pkcs8EncodedKeySpec));

        return new String(cipher.doFinal(Base64.decodeBase64(Cipher Text)), encoding);
    }
}

```

AES Encryption and Decryption :

```

package com.secure.alogithm;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class CryptoUtils {
    private static final String ALGORITHM = "AES";
    private static final String TRANSFORMATION = "AES";

    public static String encrypt(String key, File inputFile, File outputFile){
        return(doCrypto(Cipher.ENCRYPT_MODE, key, inputFile, outputFile));
    }

    public static String decrypt(String key, File inputFile, File outputFile){
        return(doCrypto(Cipher.DECRYPT_MODE, key, inputFile, outputFile));
    }

    private static String doCrypto(int cipherMode, String key, File inputFile,
        File outputFile) {
        try {
            Key secretKey = new SecretKeySpec(key.getBytes(), ALGORITHM);
            Cipher cipher = Cipher.getInstance(TRANSFORMATION);
            cipher.init(cipherMode, secretKey);

            FileInputStream inputStream = new FileInputStream(inputFile);
            byte[] inputBytes = new byte[(int) inputFile.length()];
            inputStream.read(inputBytes);

            byte[] outputBytes = cipher.doFinal(inputBytes);

            FileOutputStream outputStream = new FileOutputStream(outputFile);
            outputStream.write(outputBytes);

            inputStream.close();
            outputStream.close();

        } catch(Exception e) {

```

```

        e.printStackTrace();
        return("failure");
    }
    return("success");
}
public static void main(String args[]){ }
}

```

Appendix 2 – Screen Shots

Module 1 : Admin



Fig A.1 : Admin

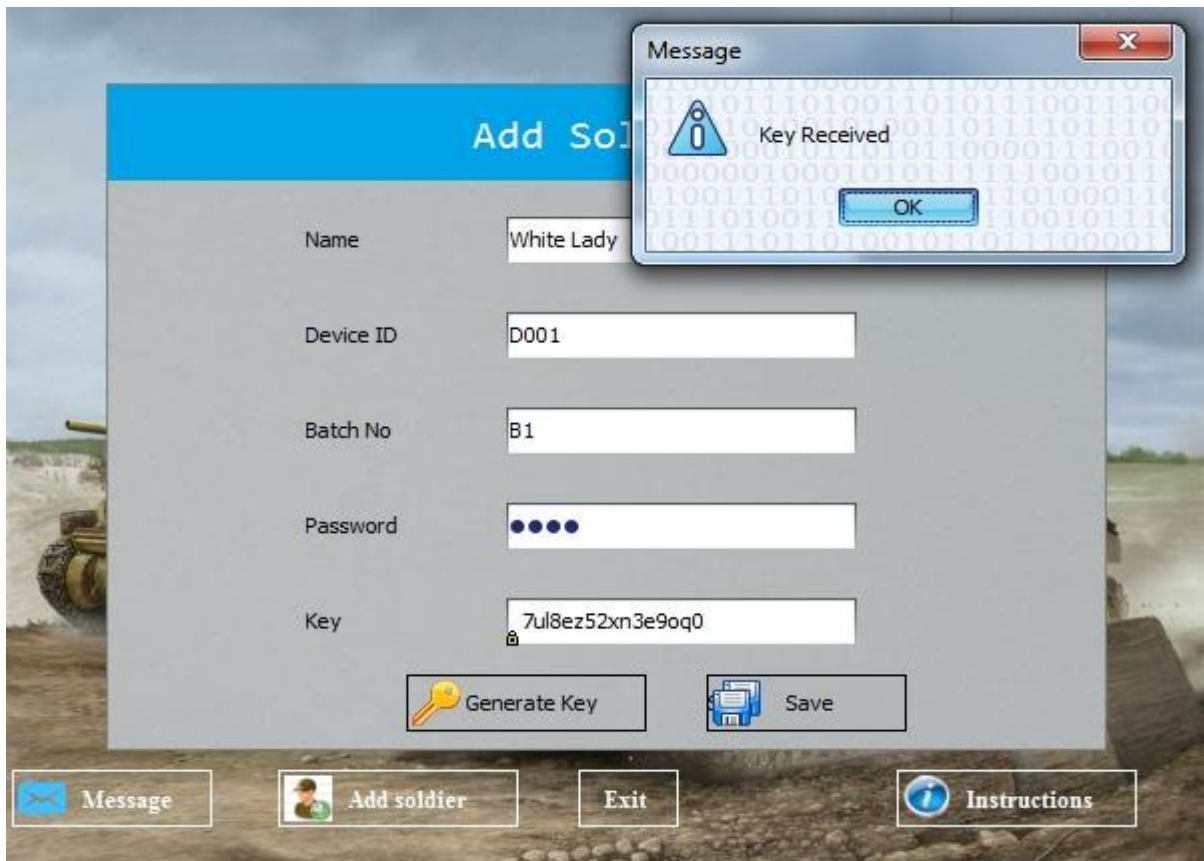


Fig A.2 : Add User

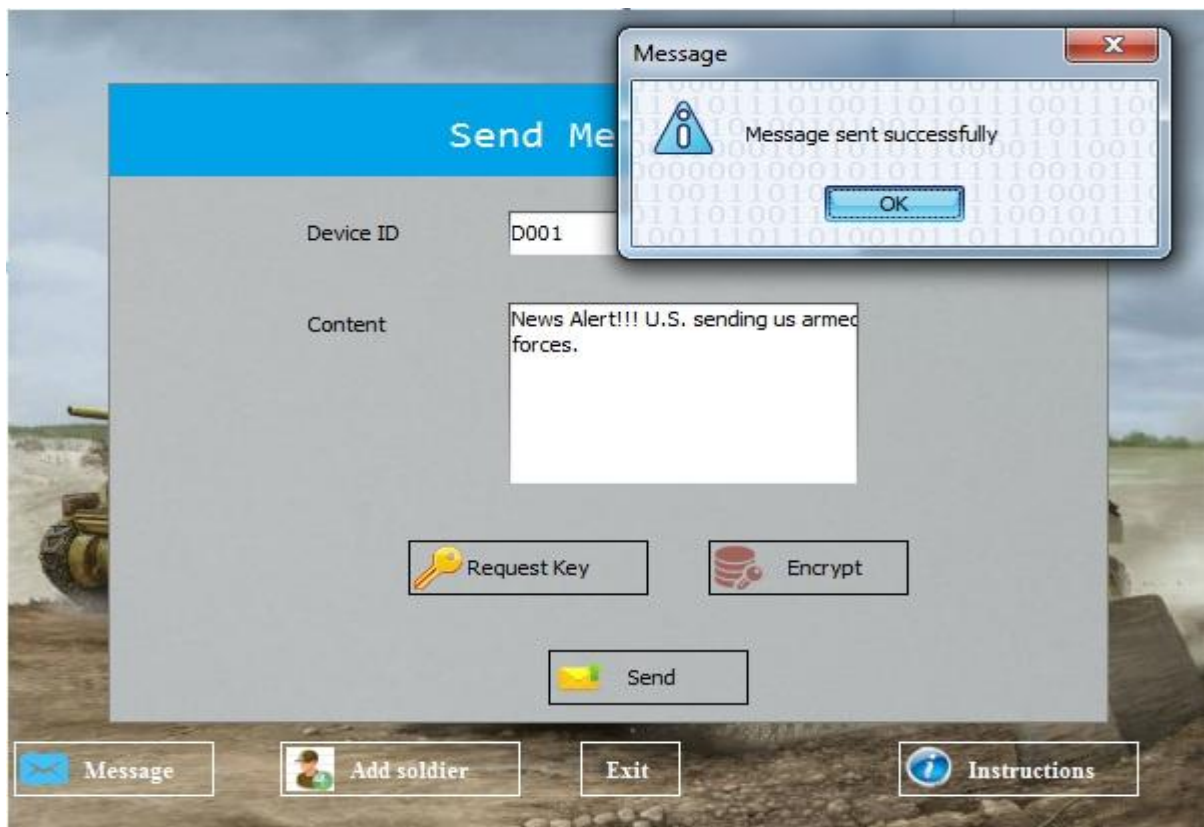
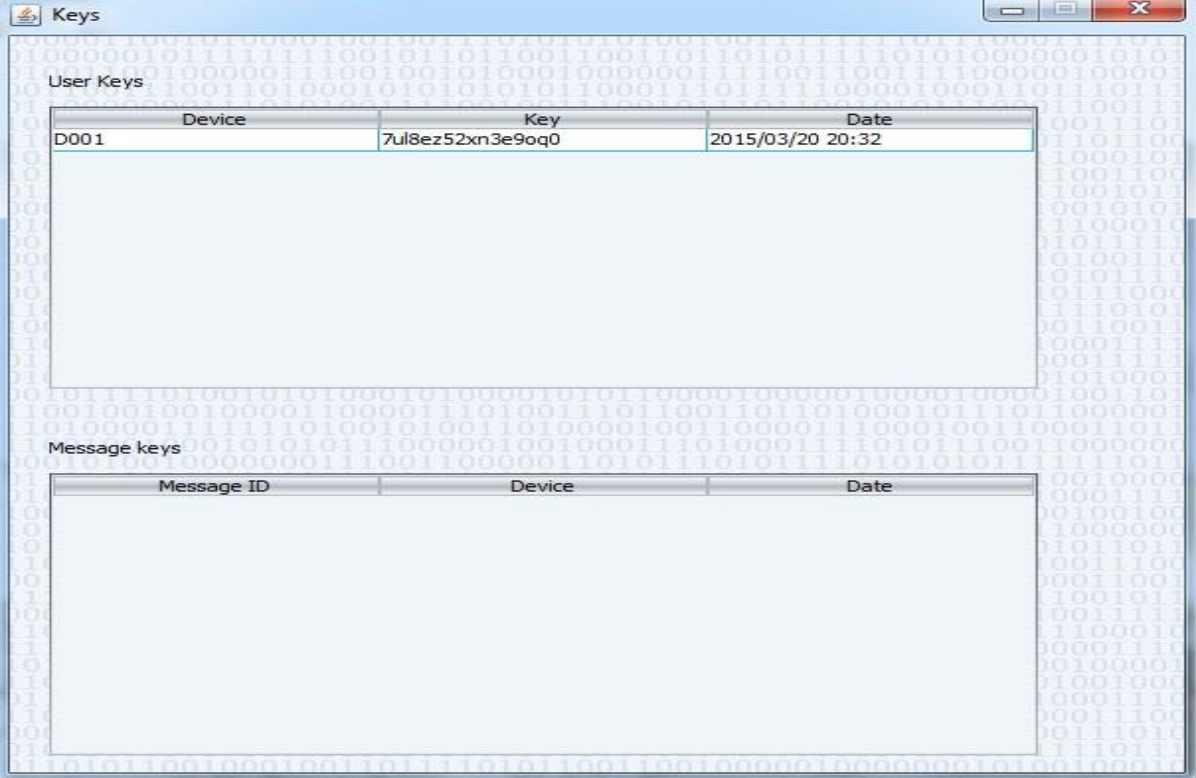


Fig A.3 : Send Message

Module 2 : Central Authority



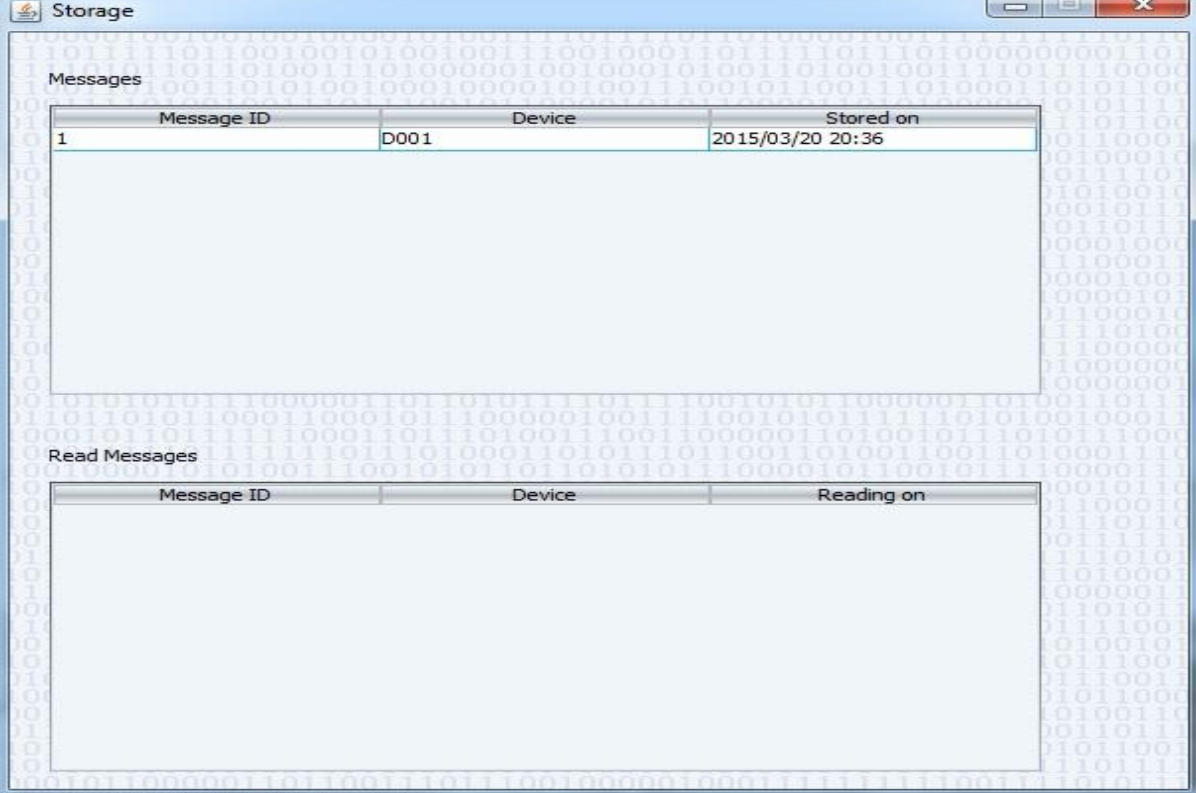
The screenshot shows a window titled 'Keys' with two table sections. The 'User Keys' section contains one entry with Device 'D001', Key '7ul8ez52xn3e9oq0', and Date '2015/03/20 20:32'. The 'Message keys' section is currently empty.

Device	Key	Date
D001	7ul8ez52xn3e9oq0	2015/03/20 20:32

Message ID	Device	Date
------------	--------	------

Fig A.4 Key Authority

Module 3 : Storage Node



The screenshot shows a window titled 'Storage' with two table sections. The 'Messages' section contains one entry with Message ID '1', Device 'D001', and Stored on '2015/03/20 20:36'. The 'Read Messages' section is currently empty.

Message ID	Device	Stored on
1	D001	2015/03/20 20:36

Message ID	Device	Reading on
------------	--------	------------

Fig A.5 : Storage Node

Module 4 : User Device

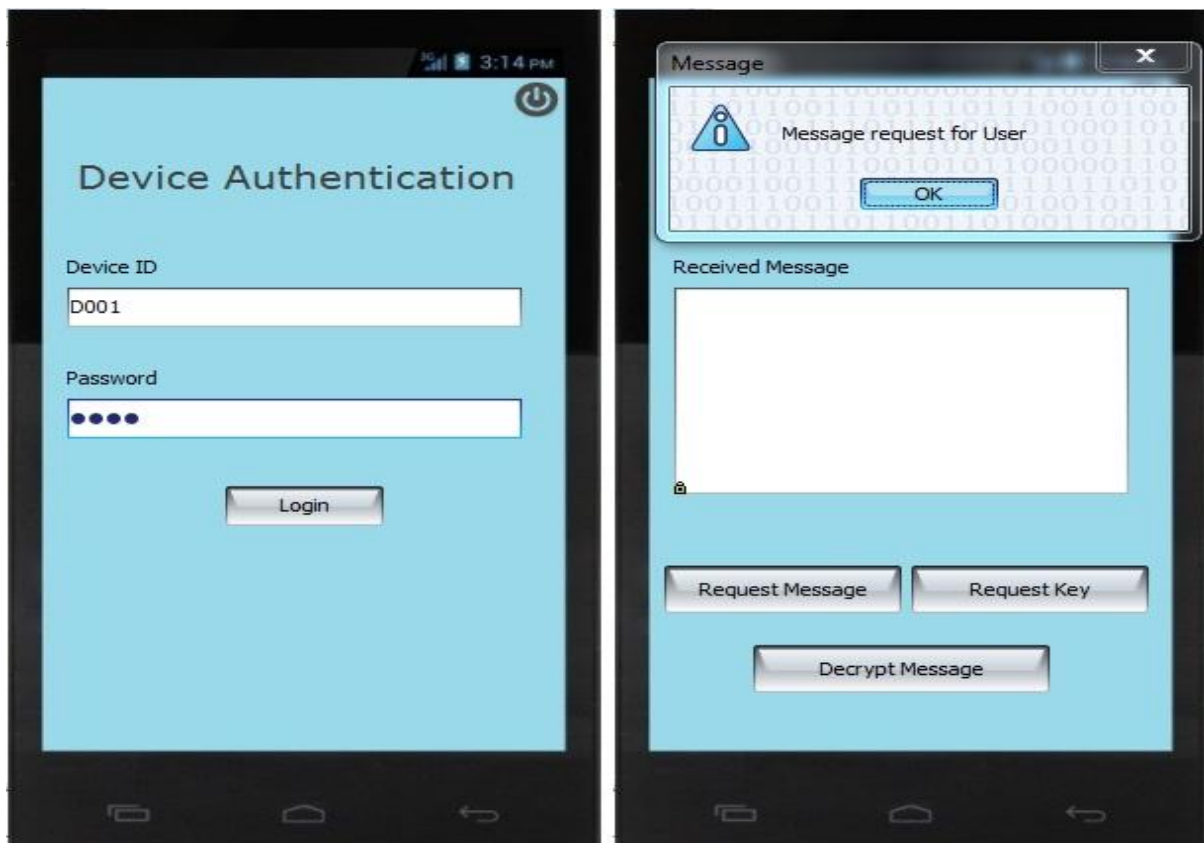


Fig A.6 : (a) Device Login (b) Request Message



Fig A.7 : (a) Decryption Key (b) Decrypt Message

Appendix 3 – Paper Published



Attribute-Based-Encryption in Disruption Tolerant Military Networks to secure data access

¹Sonia.M, ²Hemakumar.V

Department of Computer Science & Engineering

GKM College of Engineering & Technology

G.K.M.Nagar, New Perungalathur (Near Tambaram), Chennai-600063, TamilNadu , India.

Contact no. : ¹09952051209, ²09841411732

¹imsonia1993@gmail.com

²hemu.be1989@gmail.com

Abstract—Wireless devices carried by soldiers in hostile areas or battle fields are likely to suffer the threats of information leaks and disruptions in connectivity. Disruption-tolerant network (DTN) technologies allow wireless devices carried by soldiers to communicate with each other and access the confidential information or command reliably by exploiting external storage nodes. Some of the most challenging issues in this scenario are the enforcement of authorization policies and the policies update for secure data retrieval. Ciphertext-policy ABE (CP-ABE) provides a scalable way of encrypting data such that the encryptor defines the attribute set that the decryptor needs to possess in order to decrypt the ciphertext. Thus, different users are allowed to decrypt different pieces of data per the security policy. However, the problem of applying CP-ABE in decentralized DTNs introduces several security and privacy challenges with regard to the attribute revocation, key escrow, and coordination of attributes issued by authority. The existing system says about key generation using multiple authorities. In this paper, we propose a secure data retrieval scheme using CP-ABE for DTNs where key generation is done by automated scheme. The admin(Sender of message) generates a random one-time-key for the user based on the attributes, which solves the problems of forward secrecy and backward secrecy. In addition it solves key escrow problem since central authority is not explicitly involved in key generation. We describe how to apply the proposed mechanism.

Keywords-security, CP-ABE, Military Networks, Disruption Tolerant Network, One-time-key

I. INTRODUCTION

In many military network scenarios, connections of wireless devices carried by soldiers may be temporarily disconnected by jamming, environmental factors, and mobility, especially when they operate in hostile environments. Disruption-tolerant network (DTN) technologies are becoming successful solutions that allow nodes to communicate with each other in these extreme networking environments [1]. Typically, when there is no end-to-end connection between a source and a destination pair, the messages from the source node may need to wait in the intermediate nodes for a substantial amount of time until the connection would be eventually established.

Roy [2] and Chuah [3] introduced storage nodes in DTNs where data is stored such that only authorized mobile nodes can access the necessary information quickly and efficiently. Many military applications require increased protection of confidential data including access control methods that are cryptographically enforced [4], [5]. In many cases, it is desirable to provide differentiated access services such that data access policies are defined over user attributes or roles, which are managed by the key authorities. For example, in a disruption-tolerant military network, a commander may store confidential information at a storage node, which should be accessed by members of “Battalion 1” who are participating in “Region 2.”

II. CURRENT METHODOLOGY

In this case, it is a reasonable assumption that multiple key authorities are likely to manage their own dynamic attributes for soldiers in their deployed regions or echelons, which could be frequently changed (e.g., the attribute representing current location of moving soldiers) [2], [6]. Multiple authorities issue and manage their own attribute keys independently as a decentralized DTN. The challenge is the coordination of attributes issued from different authorities. When multiple authorities manage and issue attributes keys to users independently with their own master secrets, it is very hard to define fine-grained access policies over attributes issued from different authorities.

A. Issues

The concept of attribute-based encryption (ABE) [7]–[9] is a promising approach that fulfils the requirements for secure data retrieval in DTNs. ABE features a mechanism that enables an access control over encrypted data using access policies and ascribed attributes among private keys and ciphertexts. Especially, ciphertext-policy ABE (CP-ABE) provides a scalable way of encrypting data such that the encryptor defines the attribute set that the decryptor needs to possess in order to decrypt the ciphertext [8]. Thus, different users are allowed to decrypt different pieces of data per the security policy.

However, the problem of applying the ABE to DTNs introduces several security and privacy challenges. Since some users may change their associated attributes at some point (for example, moving their region), or some private keys might be compromised, key revocation (or update) for each attribute is necessary in order to make systems secure. However, this issue is even more difficult, especially in ABE systems, since each attribute is conceivably shared by multiple users (henceforth, we refer to such a collection of users as an attribute group). This implies that revocation of any attribute or any single user in an attribute group would affect the other users in the group. For example, if a user joins or leaves an attribute group, the associated attribute key should be changed and redistributed to all the other members in the same group for backward or forward secrecy. It may result in bottleneck during rekeying procedure or security degradation due to the windows of vulnerability if the previous attribute key is not updated immediately.

Another challenge is the key escrow problem. In CP-ABE, the key authority generates private keys of users by applying the authority's master secret keys to users' associated set of attributes. Thus, the key authority can decrypt every ciphertext addressed to specific users by generating their attribute keys. If the key authority is compromised by adversaries when deployed in the hostile environments, this could be a potential threat to the data confidentiality or privacy especially when the data is highly sensitive. The key escrow is an inherent problem even in the multiple-authority systems as long as each key authority has the whole privilege to generate their own attribute keys with their own master secrets. Since such a key generation mechanism based on the single master secret is the basic method for most of the asymmetric encryption systems such as the attribute-based or identity-based encryption protocols, removing escrow in single or multiple-authority CP-ABE is a pivotal open problem.

The last challenge is the coordination of attributes issued from different authorities. When multiple authorities manage and issue attribute keys to users independently with their own master secrets, it is very hard to define fine-grained access policies over attributes issued from different authorities. For example, suppose that attributes "role 1" and "region 1" are managed by the authority A, and "role 2" and "region 2" are managed by the authority B. Then, it is impossible to generate an access policy ("role 1" OR "role 2") AND ("region 1" or "region 2")) in the previous schemes because the OR logic between attributes issued from different authorities cannot be implemented. This is due to the fact that the different authorities generate their own attribute keys using their own independent and individual master secret keys.

B. Related Work

ABE comes in two flavors called key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In KP-ABE, the encryptor only gets to label a ciphertext with a set of attributes. The key authority chooses a policy for each user

that determines which ciphertexts he can decrypt and issues the key to each user by embedding the policy into the user's key. However, the roles of the ciphertexts and keys are reversed in CP-ABE. In CP-ABE, the ciphertext is encrypted with an access policy chosen by an encryptor, but a key is simply created with respect to an attributes set. CP-ABE is more appropriate to DTNs than KP-ABE because it enables encryptors such as a commander to choose an access policy on attributes and to encrypt confidential data under the access structure via encrypting with the corresponding public keys or attributes[2],[5],[10].

1) *Attribute Revocation*: Bethencourt *et al.* [8] and Boldyreva *et al.* [11] first suggested key revocation mechanisms in CP-ABE and KP-ABE, respectively. Their solutions are to append to each attribute an expiration date (or time) and distribute a new set of keys to valid users after the expiration. The periodic attribute revocable ABE schemes [6], [8], [11], [12] have two main problems.

The first problem is the security degradation in terms of the backward and forward secrecy [13]. It is a considerable scenario that users such as soldiers may change their attributes frequently, e.g., position or location move when considering these as attributes [2]. Then, a user who newly holds the attribute might be able to access the previous data encrypted before he obtains the attribute until the data is reencrypted with the newly updated attribute keys by periodic rekeying (backward secrecy). For example, assume that at time t_i , a ciphertext is encrypted with a policy that can be decrypted with a set of attributes a_i (embedded in the users keys) for users with a_i . After time, say t_j , a user newly holds the attribute set. Even if the new user should be disallowed to decrypt the ciphertext C for the time instance t_j , he can still decrypt the previous ciphertext C until it is reencrypted with the newly updated attribute keys. On the other hand, a revoked user would still be able to access the encrypted data even if he does not hold the attribute any more until the next expiration time (forward secrecy). For example, when a user is disqualified with the attribute a_i at time t_j , he can still decrypt the ciphertext of the previous time instance unless the key of the user is expired and the ciphertext is reencrypted with the newly updated key that the user cannot obtain. We call this uncontrolled period of time windows of vulnerability.

The other is the scalability problem. The key authority periodically announces a key update material by unicast at each time-slot so that all of the non-revoked users can update their keys. This results in the "1-affects-" problem, which means that the update of a single attribute affects the whole non-revoked users who share the attribute. This could be a bottleneck for both the key authority and all non-revoked users.

The immediate key revocation can be done by revoking users using ABE that supports negative clauses [2], [9]. To do so, one just adds conjunctively the AND of negation of revoked

user identities (where each is considered as an attribute here). However, this solution still somewhat lacks efficiency performance. This scheme will pose overhead $O(R)$ group elements additively to the size of the ciphertext and $O(\log M)$ multiplicatively to the size of private key over the original CP-ABE scheme of Bethencourt *et al.* [8], where M is the maximum size of revoked attributes set. Golle *et al.* [14] also proposed a user revocable KP-ABE scheme, but their scheme only works when the number of attributes associated with a ciphertext is exactly half of the universe size.

2) *Key Escrow*: Most of the existing ABE schemes are constructed on the architecture where a single trusted authority has the power to generate the whole private keys of users with its master secret information [7], [8], [9], [15]–[17]. Thus, the key escrow problem is inherent such that the key authority can decrypt every ciphertext addressed to users in the system by generating their secret keys at any time. Chase *et al.* presented a distributed KP-ABE scheme that solves the key escrow problem in a multi authority system. In this approach, all (disjoint) attribute authorities are participating in the key generation protocol in a distributed way such that they cannot pool their data and link multiple attribute sets belonging to the same user. One disadvantage of this fully distributed approach is the performance degradation. Since there is no centralized authority with master secret information, all attribute authorities should communicate with each other in the system to generate a user's secret key. This results in $O(N^2)$ communication overhead on the system setup and the rekeying phases and requires each user to store $O(N^2)$ additional auxiliary key components besides the attributes keys, where N is the number of authorities in the system.

3) *Decentralized ABE*: Huang *et al.* and Roy *et al.* [2] proposed decentralized CP-ABE schemes in the multi authority network environment. They achieved a combined access policy over the attributes issued from different authorities by simply encrypting data multiple times. The main disadvantages of this approach are efficiency and expressiveness of access policy. For example, when a commander encrypts a secret mission to soldiers under the policy ("Battalion 1" AND ("Region 2" OR "Region 3")), it cannot be expressed when each "Region" attribute is managed by different authorities, since simply multi encrypting approaches can by no means express any general "out-of-" logics (e.g., OR, that is 1-out-of-). For example, let A_1, \dots, A_N be the key authorities, and a_1, \dots, a_N be attributes sets they independently manage, respectively. Then, the only access policy expressed with a_1, \dots, a_N is a_1 AND ... AND a_N , which can be achieved by encrypting a message with a_1 by A_1 , and then encrypting the resulting ciphertext c_1 with a_2 by A_2 (where c_1 is the ciphertext encrypted under a_1), and then encrypting resulting ciphertext c_1 with a_3 by A_3 , and so on, until this multi encryption generates the final ciphertext c_N . Thus, the access logic should be only AND, and they require N iterative encryption operations where N is the number of attribute authorities. Therefore, they are somewhat restricted

in terms of expressiveness of the access policy and require $O(N)$ computation and storage costs. Chase [18] and Lewko *et al.* [10] proposed multi authority KP-ABE and CP-ABE schemes, respectively. However, their schemes also suffer from the key escrow problem like the prior decentralized schemes.

III. NETWORK ARCHITECTURE

In this section, we describe the DTN architecture and define the security model.

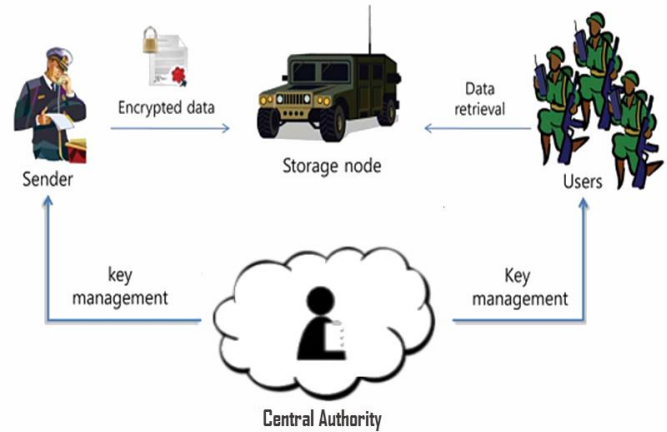


Fig 1: Architecture of secure data retrieval in a disruption-tolerant military network.

A. System Description and Assumptions

Fig. 1 shows the architecture of the DTN. As shown in Fig. 1, the architecture consists of the following system entities.

- 1) *Central Authority(Key Authority)*: The role of central authority is to track the reliability of the key and message generated and reaching the users at the exact time.
- 2) *Storage node*: This is an entity that stores data from senders and provide corresponding access to users. It may be mobile or static [2], [3]. Similar to the previous schemes, we also assume the storage node to be semi-trusted, that is honest-but-curious.
- 3) *Sender(Admin)*: This is an entity who owns confidential messages or data (e.g., a commander) and wishes to store them into the external data storage node for ease of sharing or for reliable delivery to users in the extreme networking environments. A sender is responsible for defining (attribute based) access policy and enforcing it on its own data by encrypting the data under the policy before storing it to the storage node. The sender generates the one-time key for the particular user and passes it through the central authority.
- 4) *User*: This is a mobile node who wants to access the data stored at the storage node (e.g., a soldier). If a user possesses a

set of attributes satisfying the access policy of the encrypted data defined by the sender, and is not revoked in any of the attributes, then he will be able to decrypt the ciphertext and obtain the data.

Since the key authorities are semi-trusted, they should be deterred from accessing plaintext of the data in the storage node; meanwhile, they should be still able to issue secret keys to users.

B. Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- 1) What data should be given as input?
- 2) How the data should be arranged or coded?
- 3) The dialog to guide the operating personnel in providing input.
- 4) Methods for preparing input validations and steps to follow when error occur.

C. Contribution

In this paper, we propose an attribute-based secure data retrieval scheme using CP-ABE for decentralized DTNs. The proposed scheme features the following achievements. First, immediate attribute revocation enhances backward/forward secrecy of confidential data by reducing the windows of vulnerability. Second, encryptors can define a fine-grained access policy using any monotone access structure under attributes issued from any chosen set of authorities. Third, the key escrow problem is resolved by restricting the functionality of central authority. The key issuing protocol generates and issues user secret keys by performing a secure two-party computation (2PC) protocol between the Central Authority and the Sender. The role of the Central Authority is limited to passing the key generated by the admin to the corresponding users. The Central authority has to maintain the reliability in the system. The 2PC protocol deters the Central authority from obtaining any master secret information such that he could not determine the key generated for user. Thus, users are not required to fully trust the authority in order to protect their data to be shared. The data confidentiality and privacy can be cryptographically enforced against any curious key authority or data storage nodes in the proposed scheme.

D. Analysis

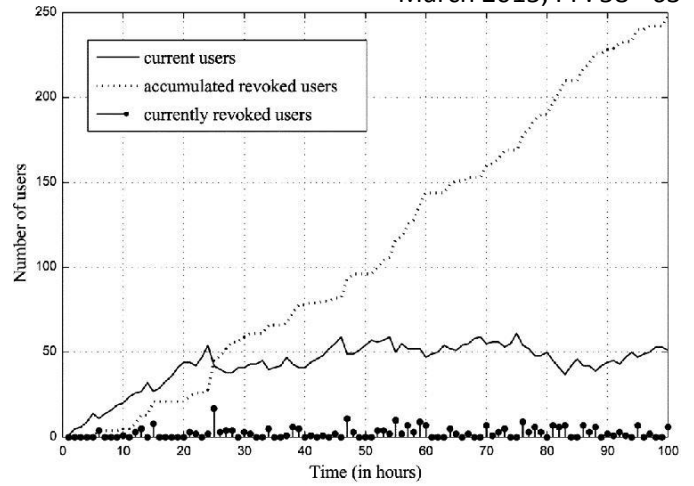


Fig. 2. Number of users in an attribute group.

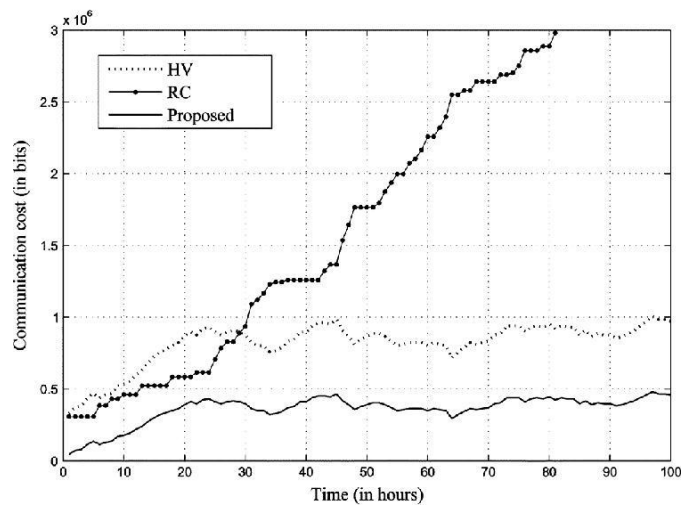


Fig. 3. Communication cost in the multi authority CP-ABE systems.

IV. PROPOSED SCHEME

An ciphertext-policy attribute based encryption scheme consists of four fundamental algorithms: Setup, Encrypt, KeyGen, and Decrypt.

- i) $Setup(\lambda, U)$: takes as input a security parameter and attribute universe description U . It outputs the public parameters PK and a master key MK .
- ii) $Encrypt(PK, M, A)$: The encryption algorithm takes as input the public parameters PK , a message M , and an access structure A over the universe of attributes. The algorithm will encrypt M and produce a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains A .

- iii) *KeyGeneration*(*MK*, *S*): The key generation algorithm takes as input the master key *MK* and a set of attributes *S* that describe the key. It outputs a private key *SK*.
- iv) *Decrypt*(*PK*, *CT*, *SK*): The decryption algorithm takes as input the public parameters *PK*, a ciphertext *CT*, which contains an access policy *A*, and a private key *SK*, which is a private key for a set *S* of attributes. If the set *S* of attributes satisfies the access structure *A* then the algorithm will decrypt the ciphertext and return a message *M*.

A. Mathematical Model

1) Encryption

- i. Input: Attribute Value (Attr).
- ii. Get Byte [](*B1*) of that Attr.
- iii. Generate Public Key(*Pk*).
- iv. Perform Encryption on *B1*.
- v. Convert *B1* into string(*EAttr*).

2) Decryption

- i. Input: Encrypted attribute value(*EAttr*)
- i. Convert *EAttr* into byte [](*B2*).
- ii. Generate Private Key.
- iii. Perform Decryption on *B2*.
- iv. Convert *B2* into string(*DAttr*).

V. SECURITY

In this section, we prove the security of our scheme with regard to the security requirements discussed in Section II.

A. Collusion Resistance

In CP-ABE, the secret sharing must be embedded into the ciphertext instead to the private keys of users. There are less chances of collusion between the keys generated for the soldiers with the same attributes or with different attributes since the algorithm deals with random generation. It is hard to find two inputs that hash to the same output; that is, two inputs *a* and *b* such that $H(a) = H(b)$, and $a \neq b$.

B. Data Confidentiality

Unauthorized users who do not have enough credentials satisfying the access policy should be deterred from accessing the plain data in the storage node. In addition, unauthorized access from the storage node or key authorities should be also prevented. Data confidentiality on the stored data against unauthorized users can be trivially guaranteed. If the set of attributes of a user cannot satisfy the access tree in the ciphertext, he cannot recover the desired value during the

decryption process, where is a random value uniquely assigned to him. In order to decrypt a node for an attribute, the user needs to pair from the ciphertext and from its private key.

C. Backward and Forward Secrecy

In the context of ABE, backward secrecy means that any user who comes to hold an attribute (that satisfies the access policy) should be prevented from accessing the plaintext of the previous data exchanged before he holds the attribute. On the other hand, forward secrecy means that any user who drops an attribute should be prevented from accessing the plaintext of the subsequent data exchanged after he drops the attribute, unless the other valid attributes that he is holding satisfy the access policy.

D. Reliability

Since the entire key generation process is governed by admin or sender of message, there are less chances of information leak. The central authority is responsible for rendering the key to the user on time.

VI. SUGGESTIONS

- i) *Deal with Break-glass Access*- Sender may need to have temporary view at the data stored in Storage Node in case when the Sender System is hacked. The Sender may want to analyse the effects of the acts by intruder. Break-glass provides temporary access under emergency scenarios.
- ii) *Serialization in storage node*- By the serialization, the size of the data becomes very small which reduces the overheads required in Storage Node.
- iii) *Constant-sized keys*- Lightweight devices, such as radio frequency identification tags, have a limited storage capacity, which has become a bottleneck for many applications, especially for security applications. A novel CP-ABE scheme with constant-size decryption keys independent of the number of attributes can be used with size as small as 672 bits.

VII. CONCLUSION

DTN technologies are becoming successful solutions in military applications that allow wireless devices to communicate with each other and access the confidential information reliably by exploiting external storage nodes. CP-ABE is a scalable cryptographic solution to the access control and secure data retrieval issues. In this paper, we proposed an efficient and secure data retrieval method using CP-ABE for centralized DTNs where central authority and sender manage the key generation using automated scheme that generates random one-time-key based on attributes. This solves the problem of forward secrecy and backward secrecy. It also solves Key Escrow problem since central authority is not explicitly involved in key generation. In addition, the system promotes reliability since the central authority keeps tracking

on when message was sent and received and also about key reaching the soldier in time. The inherent key escrow problem is resolved such that the confidentiality of the stored data is guaranteed even under the hostile environment where key authorities might be compromised. In addition, the fine-grained key revocation can be done for each attribute group. This ABE(Attribute-Based-Encryption) method can be further used in other hostile networks and private concerns like Hospitals , Research Centre to preserve data.

ACKNOWLEDGEMENT

I express my sincere thanks to my project guide **Asst. Prof. V.Hemakumar** and **Head Of Department, Dr.S.Selvakumar** of Computer Science Engineering Department at GKM College of Engineering & Technology, for their valuable guidance, suggestion and support through the project work, who have given co-operation for the project with personal attention in spite of their busy schedule. I would like to thank ETIC-2015 team , Department of Computer Science and Engineering , S.A. Engineering College for providing me this opportunity.

REFERENCES

[1] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption tolerant networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–11.
 [2] S. Roy and M. Chuah, "Secure data retrieval based on ciphertext policy attribute-based encryption (CP-ABE) system for the DTNs," Lehigh CSE Tech. Rep., 2009.
 [3] M. Chuah and P. Yang, "Performance evaluation of content-based information retrieval schemes for DTNs," in *Proc. IEEE MILCOM*, 2007, pp. 1–7.

[4] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," in *Proc. Conf. File Storage Technol.*, 2003, pp. 29–42.
 [5] L. Ibraimi, M. Petkovic, S. Nikova, P. Hartel, and W. Jonker, "Mediated ciphertext-policy attribute-based encryption and its application," in *Proc. WISA*, 2009, LNCS 5932, pp. 309–323.
 [6] N. Chen, M. Gerla, D. Huang, and X. Hong, "Secure, selective group broadcast in vehicular networks using dynamic attribute based encryption," in *Proc. Ad Hoc Netw. Workshop*, 2010, pp. 1–8.
 [7] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Eurocrypt*, 2005, pp. 457–473.
 [8] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute based encryption," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 321–334.
 [9] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. ACM Conf. Comput. Commun. Security*, 2007, pp. 195–203.
 [10] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proc. ASIACCS*, 2010, pp. 261–270.
 [11] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. ACM Conf. Comput. Commun. Security*, 2008, pp. 417–426.
 [12] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute based systems," in *Proc. ACM Conf. Comput. Commun. Security*, 2006, pp. 99–112.
 [13] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *Comput. Surv.*, vol. 35, no. 3, pp. 309–329, 2003.
 [14] P. Golle, J. Staddon, M. Gagne, and P. Rasmussen, "A content-driven access control system," in *Proc. Symp. Identity Trust Internet*, 2008, pp. 26–35.
 [15] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. ACM Conf. Comput. Commun. Security*, 2007, pp. 456–465.
 [16] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute-based encryption," in *Proc. ICALP*, 2008, pp. 579–591.
 [17] X. Liang, Z. Cao, H. Lin, and D. Xing, "Provably secure and efficient bounded ciphertext policy attribute based encryption," in *Proc. ASIACCS*, 2009, pp. 343–352.
 [18] M. Chase, "Multi-authority attribute based encryption," in *Proc. TCC*, 2007, LNCS 4329, pp. 515–534.

REFERENCES

- [1]. Nanxi Chen, Dijiang Huang and Mario Gerla , “Secure, selective group broadcast in vehicular networks using dynamic attribute based encryption” in Ad Hoc Networking Workshop (Med-Hoc-Net), 2010 The 9th IFIP Annual Mediterranean , 2010, pp. 1 – 8.
- [2]. Osada, S. , Kajita, K. , Fukushima, Y. , Yokohira, T. , “TCP incast avoidance based on connection serialization in data centre networks” in Communications (APCC), 2013 19th Asia-Pacific Conference , 2013 , pp. 142 – 147.
- [3]. Jinguang Han, Willy Susilo, Yi Mu, and Jun Yan , “Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption”, Parallel and Distributed Systems, IEEE Transactions on (Volume:23 , Issue: 11) , 2012 , pp. 2150 – 2162.
- [4]. Zhibin Zhou, Dijiang Huang, and Zhijie Wang, “Efficient Privacy-Preserving Ciphertext-Policy Attribute-Based-Encryption and Broadcast Encryption” , Computers, IEEE Transactions on (Volume:64 , Issue: 1) , 2013 , pp. 126 – 138.
- [5]. Fuchun Guo, Yi Mu, Willy Susilo, Duncan S. Wong, and Vijay Varadharajan, “CP-ABE With Constant-Size Keys for Lightweight Devices”, Information Forensics and Security, IEEE Transactions on (Volume:9 , Issue: 5) , 2014 , pp. 763 – 771.
- [6]. Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou, “Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption”, Parallel and Distributed Systems, IEEE Transactions on (Volume:24 , Issue: 1) , 2012, pp. 131 – 143.
- [7]. Junbeom Hur , “Improving Security and Efficiency in Attribute-Based Data Sharing”, Knowledge and Data Engineering, IEEE Transactions on (Volume:25 , Issue: 10) , 2013 , pp. 2271 – 2282.
- [8]. Jin Junzuo Lai, Robert H. Deng, Chaowen Guan, and Jian Weng , “Attribute-Based Encryption With Verifiable Outsourced Decryption”, Information Forensics and Security, IEEE Transactions on (Volume:8 , Issue: 8) , 2013 , pp. 1343 – 1354.
- [9]. J.Bethencourt, A.Sahai and B.Waters, “Ciphertext-Policy Attribute-Based Encryption”, in Proc. IEEE symp. Security Privacy , 2007, pp. 321-334.
- [10]. Jin Li, Jingwei Li, Xiaofeng Chen, Chunfu Jia, and Wenjing Lou, “Identity-Based Encryption with Outsourced Revocation in Cloud Computing”, Computers, IEEE Transactions on (Volume:64, Issue: 2) , 2013 , pp. 425 – 437.
- [11]. Soumya Parvatikar, Puja Prakash, Richa Prakash, Pragati Dhawale, S.B. Jadhav , “Secure sharing of personal health records using multi authority attribute based encryption in cloud computing” , International Journal of Technical Research and Applications, Volume 1, Issue 5 (Nov-Dec 2013), PP. 50-52.
- [12] M. Chase, “Multi-authority attribute based encryption,” in *Proc. TCC*, 2007, LNCS 4329, pp. 515–534.
- [13] D. Huang and M. Verma, “ASPE: Attribute-based secure policy enforcement in vehicular ad hoc networks,” *Ad Hoc Netw.*, vol. 7, no. 8, pp. 1526–1535, 2009.
- [14] S. Roy and M. Chuah, “Secure data retrieval based on ciphertext policy attribute-based encryption (CP-ABE) system for the DTNs,” *Lehigh CSE Tech. Rep.*, 2009.