

5-2017

Optimizing Key Management within a Crypto-System using Aggregate Keys

Vinay Kumar Ananthu
vkananthu@stcloudstate.edu

Follow this and additional works at: https://repository.stcloudstate.edu/msia_etds

Recommended Citation

Ananthu, Vinay Kumar, "Optimizing Key Management within a Crypto-System using Aggregate Keys" (2017). *Culminating Projects in Information Assurance*. 28.
https://repository.stcloudstate.edu/msia_etds/28

This Starred Paper is brought to you for free and open access by the Department of Information Systems at theRepository at St. Cloud State. It has been accepted for inclusion in Culminating Projects in Information Assurance by an authorized administrator of theRepository at St. Cloud State. For more information, please contact rswexelbaum@stcloudstate.edu.

Optimizing Key Management within a Crypto-System using Aggregate Keys

by

Vinay Kumar Ananthu

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of Requirements

for the Degree of

Master of Science

in Information Assurance

May, 2017

Starred Paper Committee:
Dr. Jim Q Chen, Chairperson
Dr. Dennis Guster
Dr. Bala Subramaniam

Abstract

Sharing data with peers is an important functionality in cloud storage. This is a study and analysis of secure, efficient, and flexible sharing of data with other users in cloud storage. The new public key encryptions which produce constant-size ciphertexts in such a way that effective delegation of decryption rights given to any set of ciphertexts are described in this paper. The novelty of the mechanism is that someone can aggregate any number of secret keys and turn them into a small single key, but combine the power of all the keys being grouped. To describe, in other words, the secret key holder could release a constant-size aggregate key for more flexible choices of ciphertext set in cloud storage, but different encrypted files outside of the set remain confidential. The aggregate compact key can be sent to others with ease or saved in a smart card with very less secure storage. In this paper, we discuss various such mechanisms and demonstrate the topic with a project. Some of the papers written by other authors in the area are analyzed in this paper. The project in this paper is a partial implementation of the proposed Crypto System.

Table of Contents

	Page
List of Figures	5
Chapter	
I. Introduction.....	6
Problem Statement	6
The Scope of Work	7
II. Literature Review.....	8
III. Existing System	14
Cryptographic Keys for a Predefined Hierarchy	14
Compact Key in Symmetric-Key Encryption	14
Compact Key in Identity-Based Encryption (IBE).....	14
Attribute-Based Encryption (ABE).....	14
Proxy Re-Encryption (PRE)	15
Disadvantages	15
IV. The Proposed System.....	16
Cryptographic Key Management	17
Compact Key in Symmetric-Key Encryption	19
Compact Key in Identity-Based Encryption (IBE).....	20
Other Encryption Schemes	21
Advantages.....	22
V. System Design	23

Chapter	Page
Proposed Modules.....	23
Identity Token Issuance	23
Identity Token Registration	23
Data Encryption and Uploading.....	23
Data View and Decryption.....	24
Encryption Evolution Management	24
System Design	25
Use Case Diagram.....	26
Class Diagram.....	27
Sequence Diagram	28
Activity Diagram	29
Project	29
Tools and Techniques	29
Data Base	32
Project Screenshots	33
VI. Conclusion	39
Future Work	39
References.....	40
Appendix.....	42

List of Figures

Figure	Page
1. Data Encryption and Decryption	19
2. Flowchart	25
3. Use Case Diagram.....	26
4. Class Diagram.....	27
5. Sequence Diagram	28
6. Activity Diagram	29
7. Folder Structure	32
8. Database Tables	33
9. Welcome Screen	33
10. Signup Screen	34
11. Login Screen	34
12. Home Screen.....	35
13. Inbox Screen	35
14. Decrypted Message Screen.....	36
15. Message Screen.....	36
16. Message Sent Screen.....	37
17. Logout Screen	37
18. Registration Failed Screen	38
19. Logon Failed Screen	38

Chapter I: Introduction

Cloud storage has gained huge popularity lately. The hike in demand for data outsourcing in enterprise settings helps in the strategic management of corporate data. This is used even as the primary technology behind many online web applications and services for personal applications. These days, it is effortless to apply for free email accounts, photo albums, remote access and file sharing, with storage size more than 25 Gigabytes. Combined with the present wireless technology, customers can access nearly all their emails and files with a mobile phone in every corner of the world. Considering security concerns, a conventional way to ensure the privacy of data is to rely on the server to enforce the access-control after authentication. This implies any unexpected privilege escalation will be exposing data.

Problem Statement

In a shared-tenancy distributed cloud computing environment, securing data is more challenging. Information from various clients can be hosted on different virtual machines (VMs) but resides on one physical machine. Information in the target VM can easily be stolen by instantiating another VM co-occupant with the target one. In some corporate sectors, securing private data is of utmost priority. For example, many firms in Healthcare and Pharmacy benefit manager groups keep their data with third-party vendors who are responsible for the security of the data. On the other hand, few cloud users would not believe that cloud-server is doing a great job regarding practicing confidentiality. In the existing system, in applications like photo sharing and social networking which are based on a cloud environment, it is required to share multiple keys for multiple files which require more bandwidth. An encryption solution with proven security relied on number and theoretic-assumptions would be more desirable.

The Scope of Work

Data-sharing is an essential functionality in distributed cloud environment. To illustrate, bloggers usually allow their friends to view a sub-set of their private pictures; a corporate enterprise can grant their employees access to a portion of sensitive data. The biggest concern is how to share encrypted data effectively. Of course, users may download encrypted data from the storage, decrypt it, send them to their friends for sharing, but it loses the significance of a cloud-storage. Users must be allowed to delegate the access rights of the sharing data to other users so that they can access such data directly from the server. However, finding a secure and efficient manner to share partial data in cloud storage is not of less importance.

Chapter II: Literature Review

As indicated by Cui, Liu, and Wang (2015), the capacity of specially encoded information to various clients through openly distributed storage might ease security worries over coincidental information spills in the cloud incredibly. A crucial test to outlining such encryption plans lies in the proficient administration of encryption keys. The coveted adaptability of offering any gathering of chosen reports to any group of clients requests unique encryption keys which are to be utilized for various records. This, likewise, suggests the need of safely conveying to clients an expansive number of keys for both encryption and decryption. Those customers will need to store the keys safely, and similarly present a vast number of catchphrase trapdoors. They should also focus on the end goal to perform and seek the common information (Zhu, Jiang, & Jiang, 2013). The suggested requirement for secure correspondence, stockpiling, and many-sided quality plainly renders the methodology unfeasible.

In his paper, he also addressed this functional issue, which is to a large degree ignored in writing. He proposed the novel idea of key aggregate searchable encryption (KASE) and instantiating the design through a solid KASE plan. In the plan, an information proprietor just needs to circulate a single key to a client for sharing a substantial number of reports. The client just needs to present a single trapdoor to the cloud for questioning the mutual records. The security investigation and execution assessment both affirm that our proposed plans are provably secure and productive.

Ruj, Nayak, and Stojmenovic (2011) proposed another propelled deduplication framework which supports approved copy check. In this new deduplication framework, a half-breed cloud design rightly deals with the challenges. The individual keys for benefits are not

released to clients in a straightforward manner. Instead, they are kept protected by the private cloud server. Here, the clients cannot share their secret keys to their peers in the proposed development, which means that it can sustain the benefit key sharing among customers in a highly-developed environment. To find a document token, the client needs to transmit a solicitation to the private cloud server. To execute the copy, check for some document that the customer demands to make and record which can be copied from the private cloud server. The private cloud server will likewise match the client's key before issuing the document token to the node. The approved copy check for this record can be performed by the customer with people in general cloud before transferring this material. Considering the aftereffects of copy-check, the client either assigns this document or runs PoW.

Lewko and Waters (2011) proposed a Multi-Authority Attribute-Based Encryption (ABE) framework. In our general account, any gathering can turn into power, and there is no requirement for any global coordination other than the output of an underlying system of standard reference parameters. A group can typically go around as an ABE power by hitting an open key and issuing private keys to several clients that mirror their traits. A client can encode information regarding any boolean equation over characteristics released from any special set of abilities. Finally, our framework does not call for any focal power. In developing our framework, our biggest specialized obstacle is to make the arrangement safe. Earlier Attribute-Based Encryption frameworks accomplished agreement resistance when the ABE framework power "tied" together diverse parts (speaking to various tones) of a client's private key by randomizing the key. On the other hand, every part of our framework will develop from a conceivably distinctive power, where we accept no coordination between such forces.

Lewko and Waters (2011) created new methods to tie key segments together and forestall intrigue assaults between clients with various worldwide identifiers. They demonstrated a secure utilization of double frame encryption technique where the security verification works by changing the ciphertext and private keys to a semi-practical structure and later contending security. We assume that after a slow variation of the second framework, an evidence procedure given by Lewko and Waters helps us assemble our framework by utilizing bilinear gatherings of the composite request. We demonstrate security under comparative static suppositions in this paper with the arbitrary prophetic model.

An article by Boneh and Hamburg (2008) gave a general structure to building a character based encryption framework and telecast it. Specifically, we get a general encryption framework called spatial encryption from which various frameworks with a mixture of properties take place. The ciphertext size in each of these frameworks is autonomous of clients included and it has only three gathering components. Once the purpose of these outcomes is given, the principle shows HIBE framework with short ciphertexts. Tele-HIBE takes care of the issue with personality based scrambled email.

Another research paper by Bitar, Gringeri, and Xia (2013) says that cloud today confront a few difficulties when facilitating line-of-business applications in the cloud. Fundamental to many of these challenges is the restricted backing for control over cloud system capacities. For example, the ability to guarantee security, execution sureties or separation and too adaptable middleboxes intervene in application organizations. In this paper, we show the configuration and usage of a new cloud organizing framework called CloudNaaS. Clients can influence CloudNaaS to convey applications expanded with a rich and extensible arrangement of system capacities, for

example, virtual system seclusion, custom tending, separation of administration, and adaptable intervention of different middleboxes. CloudNaaS primitives are precisely executed inside the cloud framework itself by utilizing active programmable system components thus making CloudNaaS very productive. We assess an Open Flow-based model of CloudNaaS and observe that it can be used to instantiate a mixed bag of system capacities in the cloud and that its execution is helpful despite vast quantities of provisioned administrations and disappointments caused by connected gadgets.

According to Ramgovind, Eloff, and Smith (2010, August), cloud computing has raised IT as high as possible by offering the business environment, information stockpiling. It also limits adaptable and versatile energy to match flexible request and supply while lessening capital use. However, the open-door expense of a fruitful execution of Cloud registering is to deal with the security in the cloud applications successfully. Security cognizance and concerns emerge when one starts to run applications past the assigned firewall and tries to move closer to the general population space. The motivation behind the paper is to give a general security point of view of Cloud processing with the expectation to highlight the security worries that ought to be tended appropriately and to figure out how to understand the maximum capacity of Cloud registering. Gartner's rundown on cloud security issues and discoveries from the International Data Corporation venture board are studied here. The cloud dangers given by study will be examined in this paper.

A research by Oberheide, Veeraraghavan, Cooke, Flinn, and Jahanian (2008) said mobile phones keep advancing in their capacities and extensibilities like those of standard desktop PCs. Thus, these gadgets are starting to face many security dangers as desktops. As of now, portable

security arrangements reflect the conventional desktop display in which they run identification benefits on the gadget. This methodology is sophisticated and asset concentrated in both processing and storing.

This paper proposes another model where a versatile antivirus usefulness is moved to an off-gadget system administration utilizing various virtualized malware location motors. Our contention is that, it is conceivable to spend data transfer capacity assets to lessen on-gadget's CPU and Memory essentially. We show how our in-cloud model improves portable security and decreases on-gadget programming unpredictability, while taking into consideration new administrations, for example, behavioral examination motors.

Our benchmarks on Nokia's N800 and N95 cell phones demonstrate that our portable specialists devour a request of greatness, less CPU and memory. Likewise, expending less power in similar situations contrasted with existing on-gadget antivirus programming.

Cloud computing is generally considered as an appealing administration model which follows the client's responsibilities for venture thus operations are minimized, and expenses are in immediate connection with utilization and interest (Schoo et al., 2011).

When organizing angles for circulated lists are considered, there is little room for back up, and the exertion is frequently disparaged. The venture SAIL tends to form cloud environment as the blend of administration for distributed computing and basic systems administration capacities between disseminated cloud assets are included to enhance the management of both. This position exhibits new security challenges when we consider SAIL for guaranteeing actual utilization of cloud systems, administration assets, and aversion of abuse.

According to Nurmi et al. (2009), distributed computing frameworks on a very basic level give access to expansive pools of information and computational assets. This takes place through a mixture of interfaces comparable in soul to existing lattice and HPC asset administration and programming frameworks. This kind of frameworks offer another programming focus for adaptable application designers. They have picked up prevalence during recent years. In any case, most distributed computing frameworks in operation today are restrictive and they depend upon a base that is undetectable to the exploration group. They are not actually intended to be instrumented and changed by frameworks scientists.

Chapter III: Existing System

Cryptographic Keys for a Predefined Hierarchy

- Cryptographic key assignment plans intend to reduce the expenses in storing and managing private keys for general cryptographic utilization. Using a tree-structure, a key for a given branch can be utilized to determine the keys of its descendant nodes.
- More sophisticated cryptographic key assignment plans support access policy that can be modeled by a cyclic graph or an acyclic graph.

Compact Key in Symmetric-Key Encryption

An encryption plan which is initially proposed for concisely transmitting a huge number of keys in broadcast scenarios.

Compact Key in Identity-Based Encryption (IBE)

IBE is a kind of public key encryption where the public key of a user can be set as an identity string of the user. There is a trusted party known as the private key generator in IBE which holds a master secret key (main key) and generates a secret key to each user in perspective of the user identity. The encryptor may receive the public parameter and concerned user identity to convert (encrypt) a message. The recipient can convert it back (decrypt this ciphertext) with the help of his secret key.

Attribute-Based Encryption (ABE)

ABE allows each ciphertext to be associated with an attribute, and the master secret key holder can extract a private key for a policy of these attributes so that a ciphertext can be decrypted with the use of this key if its associated attribute conforms to the related policy.

Proxy Re-Encryption (PRE)

In order to assign the decryption force of some ciphertexts without sending the secret key (private) to the delegate, a valuable primitive is proxy re-encryption (PRE). A PRE-plan permits the sender to delegate the server's (proxy) ability to convert the ciphertexts encrypted under their public key into plain texts for the receiver.

Disadvantages

- Cryptographic Keys for a predefined hierarchy are usually costlier than symmetric key operations like a pseudorandom function.
- Compact-Key in Symmetric Key Encryption is designed for the symmetric-key setting instead. The encryptor should get the corresponding private or secret keys to encrypt the data, which is usually not suitable for most applications. Since this method is used to produce a secret value rather than a pair of public/private keys, it is unclear about the way in which this idea is planned for public-key encryption.
- IBE mainly increases the expenses of saving and transmitting ciphertexts, which is not very practical in most of the situations such as shared cloud-storage.
- In ABE, either the length of the key often increases linearly with the number of attributes it encompasses, or the length of ciphertext is not constant.
- Using PRE, we can simply move the secure and secret key storage requirement to the proxy from the delegate. So, it is undesirable to let the proxy reside on the storage server.

Chapter IV: The Proposed System

- To learn more about making a decryption key powerful in a way that it allows decryption of any number of ciphertexts, without expanding its size.
- To outline an effective public-key encryption scheme which helps adaptable delegation in such a way that any subset of ciphertexts is decryptable by a decryption key of constant-size.
- Introduce a unique type of public key encryption which is known as key-aggregate cryptosystem (KAC).
- In KAC, users usually encrypt a message with the use of a public-key, as well as an identifier of ciphertext called class. That implies the ciphertexts are further classified into different classes. The key proprietor holds a master secret key, which can be utilized to derive secret keys for various classes.
- More essentially, the derived key can be an aggregate key which is compact like a secret key for a single class, yet aggregates the power of numerous such keys, i.e., the decryption power for some subsets of such ciphertext classes.
- A secure email can be used to send the Aggregate key to the receiver. The encrypted content can be downloaded by the receiver and decrypted by using this aggregate key.

Information can be scrambled through Encryption by any individual who additionally chooses what ciphertext class is connected with the plaintext message to be encoded. The information proprietor can utilize the expert mystery key pair to create a total unscrambling key for an arrangement of ciphertext classes through Extract. The created keys can go to delegates safely through secure messages or secure gadgets Finally, any client with a total key can decode

any ciphertext gave that the ciphertexts class is contained in the total key using Decrypt. Key total encryption plans comprise of five polynomial time calculations as takes after:

1. Setup ($1\lambda, n$): The data owner establishes public system parameter via Setup. On input of a security level parameter 1λ and number of ciphertext classes n , it outputs the public system parameter $param$
2. KeyGen: It is executed by the data owner to randomly generate a public/ master secret key pair (Pk, msk) .
3. Encrypt (pk, i, m) : It is executed by data owner and for message m and index I , it computes the ciphertext as C .
4. Extract (msk, S) : It is executed by data owner for delegating the decrypting power for a certain set of ciphertext classes, and it outputs the aggregate key for set S denoted by Ks .
5. Decrypt (Ks, S, I, C) : It is executed by a delegate who received, an aggregate key Ks generated by Extract. On input Ks , set S , an index I denoting the ciphertext class ciphertext C belongs to and output is decrypted result m (Krishna et al. 2015)

Cryptographic Key Management

Cryptographic key assignment schemes aim to minimize the cost for storage and management of secret keys for regular cryptographic use. By using a tree-structure, a key of a given branch can be utilized to generate the keys of their descendant nodes (but not in the other way around). By simply granting the parent key, all the keys of its descendant nodes are granted implicitly. Sandhu (Dutta & Annappa, 2014) proposed a strategy to derive a tree hierarchy of symmetric keys by utilizing recurring evaluations of pseudorandom function or a block cipher on

a fixed secret. The idea can be generalized to a graph from a tree. More sophisticated cryptographic key assignment schemes help in accessing the policy that can be designed by a cyclic graph or an acyclic graph (Krishna & Prasad, 2015; Wu, Zhou, Wang, Pan, & Chen, 2014). Most of the mentioned schemes generate keys for symmetric key encryptions, even though the key generations may need modular arithmetic as used in public key encryptions, which are usually costlier than symmetric-key operations like “pseudorandom functions.”

Let us take the tree structure to illustrate. Firstly, Alice can categorize the ciphertext classes by their subjects. Every node in the tree represents a private (secret) key, while the keys for individual ciphertext classes is represented by the leaf nodes. The keys for the classes to be delegated are represented by filled circles and the keys to be granted are represented by circles circumvented by dotted lines. Note that the keys of descendant nodes can be derived from every key of the non-leaf node. If Alice likes to share all the files in her personal category, only the key for the node “personal” needs to be granted by her, which automatically grants the keys for all the descendant nodes (“music,” “photo”). This makes it an ideal case, where most classes which must be shared belong to the same branch, and so, their parent key is good enough. In anyways, it is still not easy for usual cases. If Alice wants to share her demo music with a colleague at work who also possesses the rights to view some of her personal data, she needs to provide more keys, leading to an increase in the total size of the key. One can see that this approach is not adaptable when the categorizations are more complex, and she needs to share different sets of documents to different people. For the person in our example, the number of secret keys to be granted becomes the same as the number of different classes. Generally, hierarchical approaches may solve the issues partially if someone intends to share all the files

under a particular branch in their hierarchy. In this way, the number of keys increases as the number of branches increases. It is not easy to come up with some hierarchy that should save a total number of keys to be granted to all the individuals (which have access privileges to a different set of leaf nodes) simultaneously.

Compact Key in Symmetric-Key Encryption

Propelled by the same concern of supporting adaptable hierarchy in decryption power delegation (but with the symmetric key setting), Benaloh et al. (as cited in Reddy & Yadav, 2015) presented a cryptosystem which is initially proposed for concise transmission of some keys in a broadcast scenario. The development is simple, and we quickly review its key generation process here for a solid description of the desired properties we like to achieve. The generation of the key for a set of classes (a subset of all possible ciphertext classes) is as described below: A composite mod $N \approx p \cdot q$ is selected where p and q are two large random prime numbers. A master secret key Y is selected at random using \mathbb{Z} .

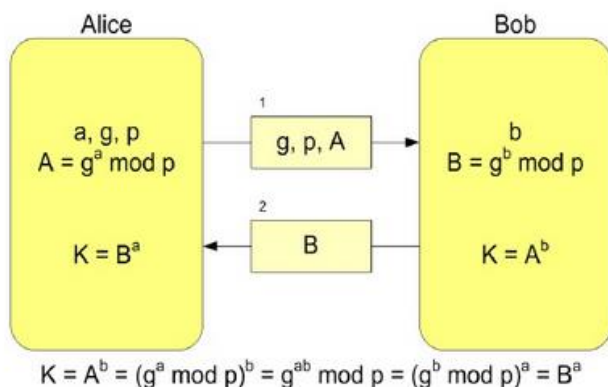


Figure 1. Data Encryption and Decryption

N. Each class relates to a distinct prime i.e. All these prime numbers can be placed in public framework parameter. A steady size key for set S_0 can be created (with the learning of

As a solid illustration, a key for classes represented by $e_1; e_2; e_3$ can be produced as $Y = e_1 \oplus e_2 \oplus e_3$, from which each of $Y \oplus e_1, Y \oplus e_2,$ and $Y \oplus e_3$ can be inferred without much effort (while giving no data about keys to some other class, say, e_4). This approach accomplishes similar properties and performances as our plans. In anyways, it is intended for the symmetric-key setting. The encryptor needs to get the relating mystery keys to encode information, which is not reasonable for some applications. Since their technique is utilized to produce a secret value rather than a couple of public/secret keys, it is vague on how to apply this idea for public key encryption scheme.

At last, we take a note that there are plans which attempt to lessen the key size for accomplishing verification in symmetric-key encryption, for instance (Reddy & Yadav, 2015). In any case, sharing of decryption power is not an issue in these schemes.

Compact Key in Identity-Based Encryption (IBE)

IBE is a sort of open key encryption in which public key of a client can be set as an identity string of the client (e.g., an email address). There is a trusted party called private key generator in IBE which holds a secret master key and issues a secret key to every client as for the client identity. The encryptor can take the public parameter and a client identity to encrypt a message. The beneficiary can decrypt this ciphertext by his secret key. Guo et al. attempted to build IBE with key accumulation (Lewko & Waters, 2011; Reddy & Yadav, 2015). One of their plans assumes random oracles. However, another does not (Ruj, Nayak, & Stojmenovic, 2011). In their plans, key aggregation is compelled in such a way that all keys to be collected must originate from various “identity divisions.” While there are an exponential number of identities and thus secret keys, just a polynomial number of them can be accumulated. In particular, their

key-accumulation (Cui et al., 2015; Reddy & Yadav, 2015) comes to the detriment of O δ nP sizes for both ciphertexts and the public parameter, where n is the number of secret keys which can be aggregated into a consistent size one. This enormously builds the expenses of saving and transmitting cyphertexts, which is unrealistic. For example, shared distributed storage. As we specified, our plans highlight steady ciphertext measure, and their security holds in the standard model. In fluffy, (Reddy & Yadav, 2015) IBE one single reduced secret key can decode ciphertexts encrypted under numerous identities which are shut in a specific metric space, yet not for a subjective arrangement of identities and, therefore, it doesn't coordinate with our concept of key aggregation (Reddy & Yadav, 2015).

Other Encryption Schemes

Attribute-based encryption (ABE) (Cui et al., 2015; Okuhara, Shiozaki, & Suzuki, 2010) permits each cyphertext to relate to quality, and the master secret key holder can separate a secret key for a policy of these attributes so that a ciphertext can be decrypted by this key if its related attribute fits in with the strategy. For instance, with the secret key for the strategy $\delta_2 _ 3 _ 6 _ 8$, one can decrypt ciphertext labeled with class 2, 3, 6, or 8. However, the significant concern in ABE is collusion resistance, however not the compact nature of secret keys. Indeed, the length of the key frequently increments linearly with the quantity of attributes it envelops, or the ciphertext size is not constant.

To designate the decryption-power of some ciphertexts without transmitting the secret key to the delegate, a valuable primitive is proxy re-encryption (PRE) A PRE-plot permits Alice to delegate to the server (intermediary) the capacity to convert the ciphertexts encrypted under her public key into ones for Bob. PRE is well known for having various applications including

cryptographic file system (Ruj et al., 2011). Nevertheless, Alice needs to believe the proxy that it just converts ciphertexts as per her guideline, which is the thing that we need to maintain a strategic distance from at the primary spot. It gets even worse if the proxy connives with Bob—Alice's secret key can be recovered which can decode Alice's (convertible) ciphertexts without Bob's further help. This likewise implies that the transformation key of proxy ought to be much secured. The secure key storage requirement is moved from the delegate to the proxy by using PRE. It is, in this way, undesirable to give the proxy a chance to reside in the capacity server.

Advantages

- Constant key size
- Less overhead for key storage
- High security

Chapter V: System Design

Proposed Modules

Identity Token Issuance

IDPs are trusted third parties that issue identity tokens to Users based on their identity attributes. It should be noted that IDPs need not be online after they issue identity tokens.

Identity Token Registration

Users register their token to obtain secrets to later decrypt the data they are allowed to access. Users register their tokens related to the attribute conditions in ACC with the Owner, and the rest of the identity tokens related to the attribute conditions in ACB/ACC with the Cloud. When Users register with the Owner, the Owner issues them two sets of secrets for the attribute conditions in ACC that are also present in the sub-ACPs in ACPB Cloud. The Owner keeps one set and gives the other set to the Cloud. Two different sets are used to prevent the Cloud from decrypting the Owner encrypted data.

Data Encryption and Uploading

The Owner first encrypts the data based on the Owner's sub-ACPs to hide the content from the Cloud and then uploads them along with the public information generated by the AB-GKM:: KeyGen algorithm and the remaining sub-ACPs to the Cloud. The Cloud, in turn, encrypts the data based on the keys generated using its AB-GKM:: KeyGen algorithm. Note that the AB-GKM:: KeyGen at the Cloud takes the secrets issued to Users and the sub-ACPs given by the Owner into consideration to generate keys.

Data View and Decryption

Users download encrypted data from the Cloud and decrypt twice to access the data. First, the Cloud generated public information tuple is used to derive the OLE key, and then the Owner generated public information tuple is used to derive the ILE key using the AB-GKM::KeyDer algorithm. These two keys allow a User to decrypt a data item only if the User satisfies the original ACP applied to the data item.

Encryption Evolution Management

Over time, either ACPs or user credentials may change. Further, already encrypted data may go through frequent updates. In such situations, data already encrypted must be re-encrypted with a new key. As the Cloud performs the access control enforcing encryption, it simply re-encrypts the affected data without the intervention of the Owner.

Use Case Diagram

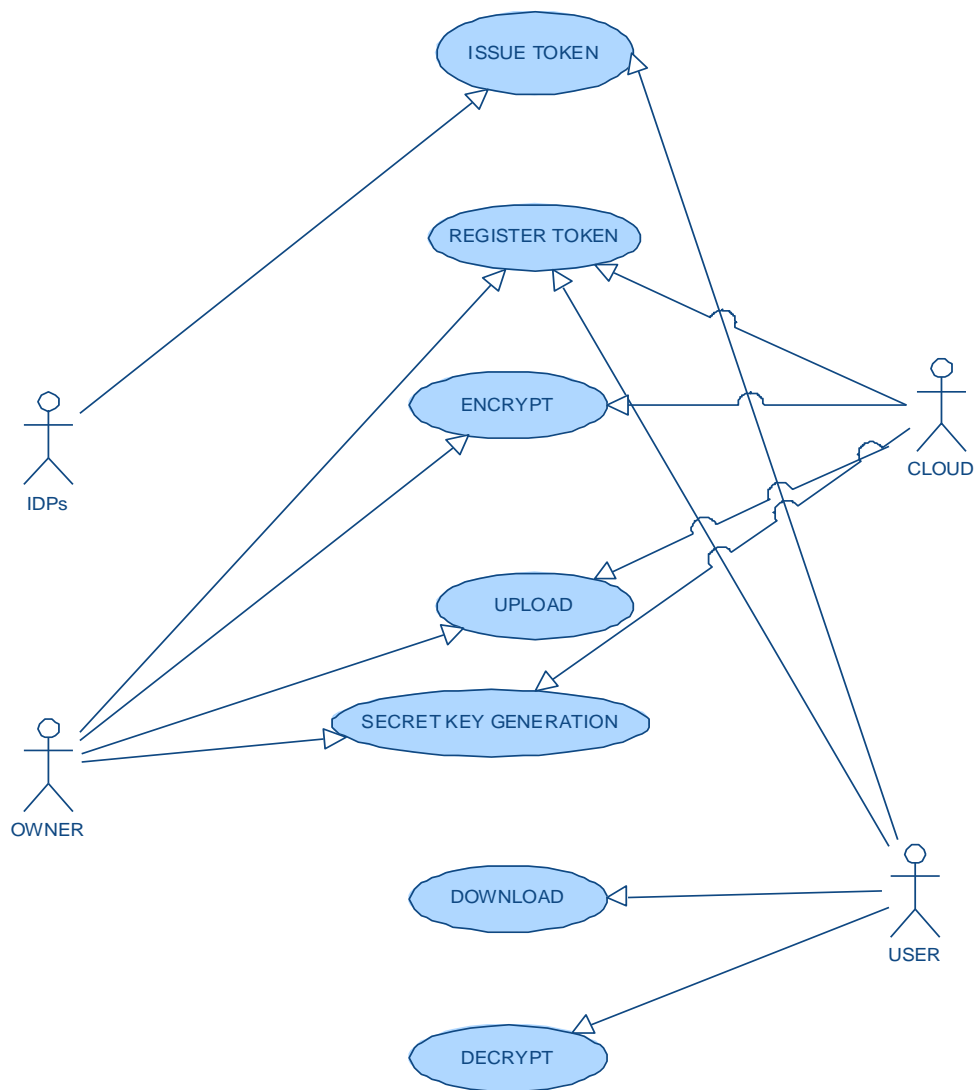
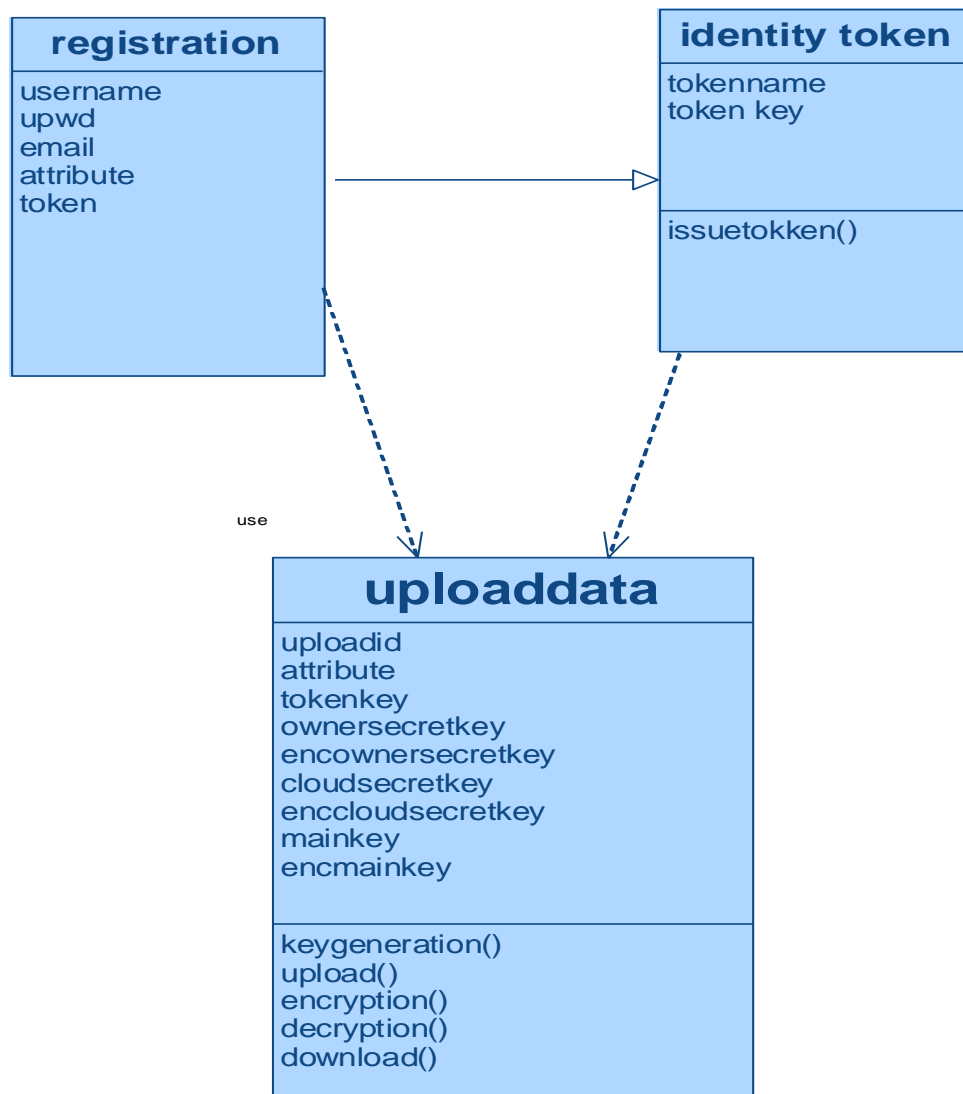


Figure 3. Use Case Diagram

Class Diagram*Figure 4.* Class Diagram

Sequence Diagram

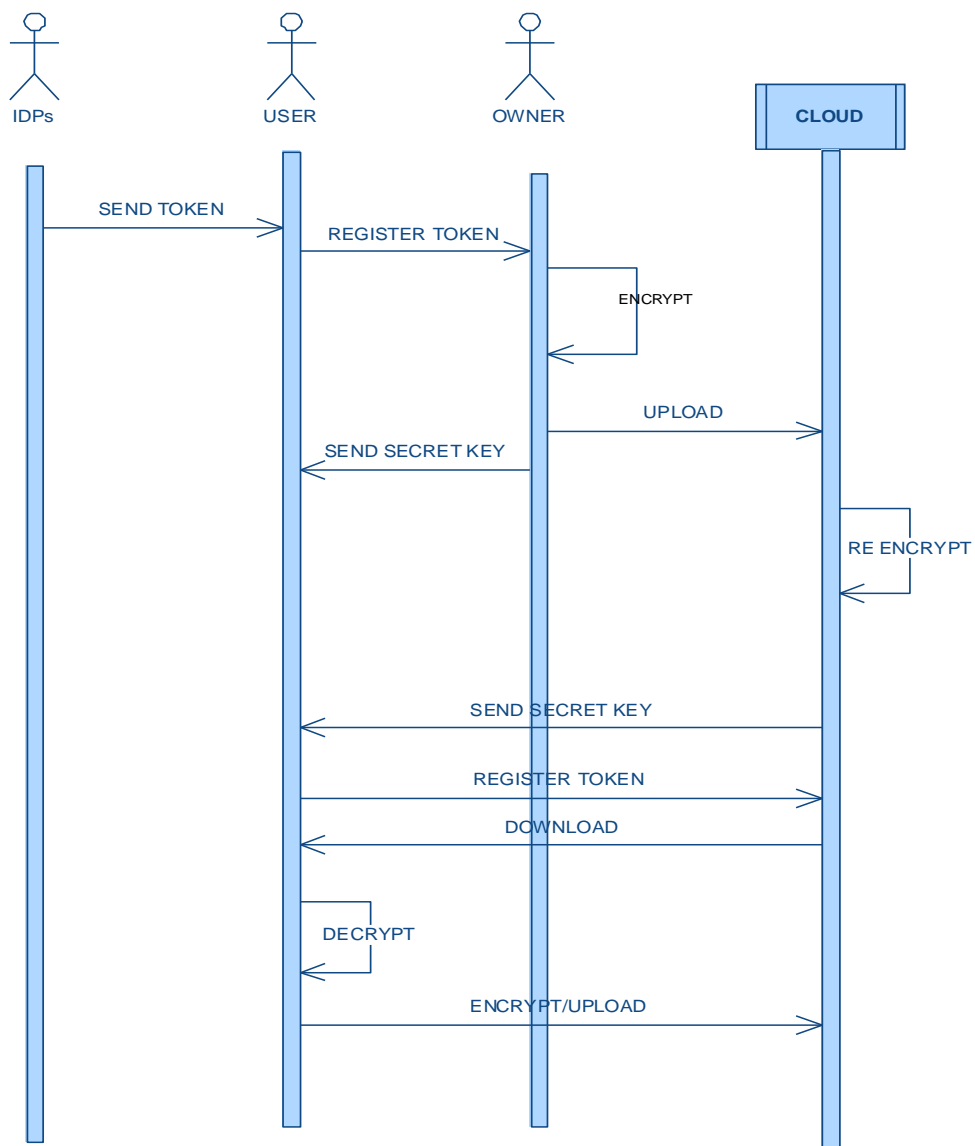


Figure 5. Sequence Diagram

Activity Diagram

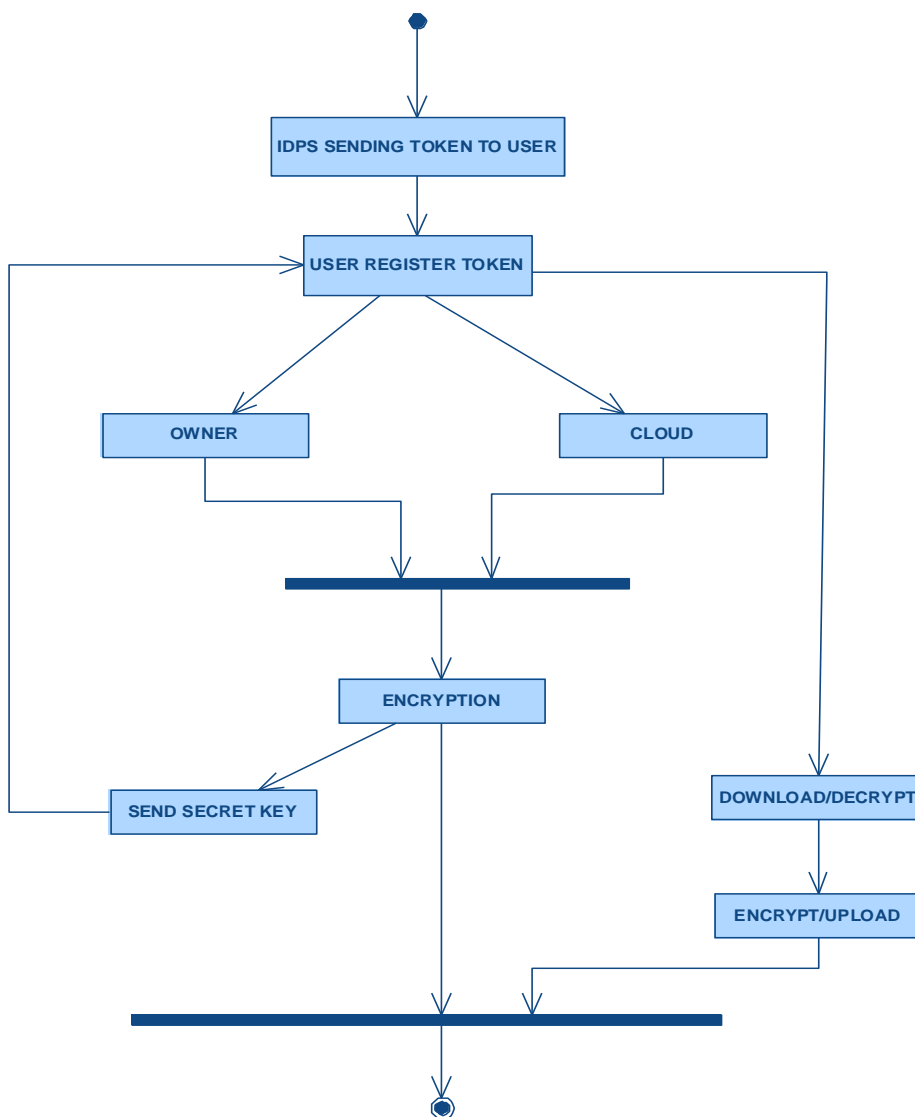


Figure 6. Activity Diagram

Project

Tools and Techniques

Hardware requirements:

Processor

Any Processor above 500 MHz

Ram

256Mb.

Hard Disk	10 GB.
Compact Disk	650 Mb.
Input device	Standard Keyboard and Mouse.
Output device	VGA and High-Resolution Monitor.

Software requirements:

Operating System	Windows Family.
Language	JDK 1.5 or above
Database server	MySQL 5.0
ORM Tool	Hibernate 5.4
Front End	HTML, JSP, JavaScript
Front End CSS	LibraryBootstrap
Encryption Algorithm	Caesar Cipher
Application Server	Apache Tomcat

Following are the reasons I chose the above techniques.

- Java: Because Java is a secure language and portable across all platforms. Java's code is (each program) is translated to Java Byte code which is robust. As there can be a JVM (Java Virtual Machine) in each machine, it can execute Java byte code.
- MySQL: Because it is an open source database which has handy server to store back end data. Database is the back bone of any project.
- Hibernate ORM: Hibernate is a very cool ORM (Object relational Mapping) tool. It makes us free of JDBC driver connections and Statement/Prepared Statement classes to query. It also prevents SQL Injection without any effort.

- Modified Caesar Cipher: Caesar cipher is one of the basic and old encryption techniques, But, I just wanted to start the project with this as it is just a beginning and at a later point, I would modify the project with some more effective encryption techniques.
- Bootstrap: Bootstrap is very good library to have when we want to design a good looking web application. It gives us a large variety of inbuilt CSS libraries, which give us a wide variety of navigation bars, buttons, texting styles etc., with very less effort.
- Besides all these, Eclipse IDE can integrate all these libraries and servers and make the work easy.

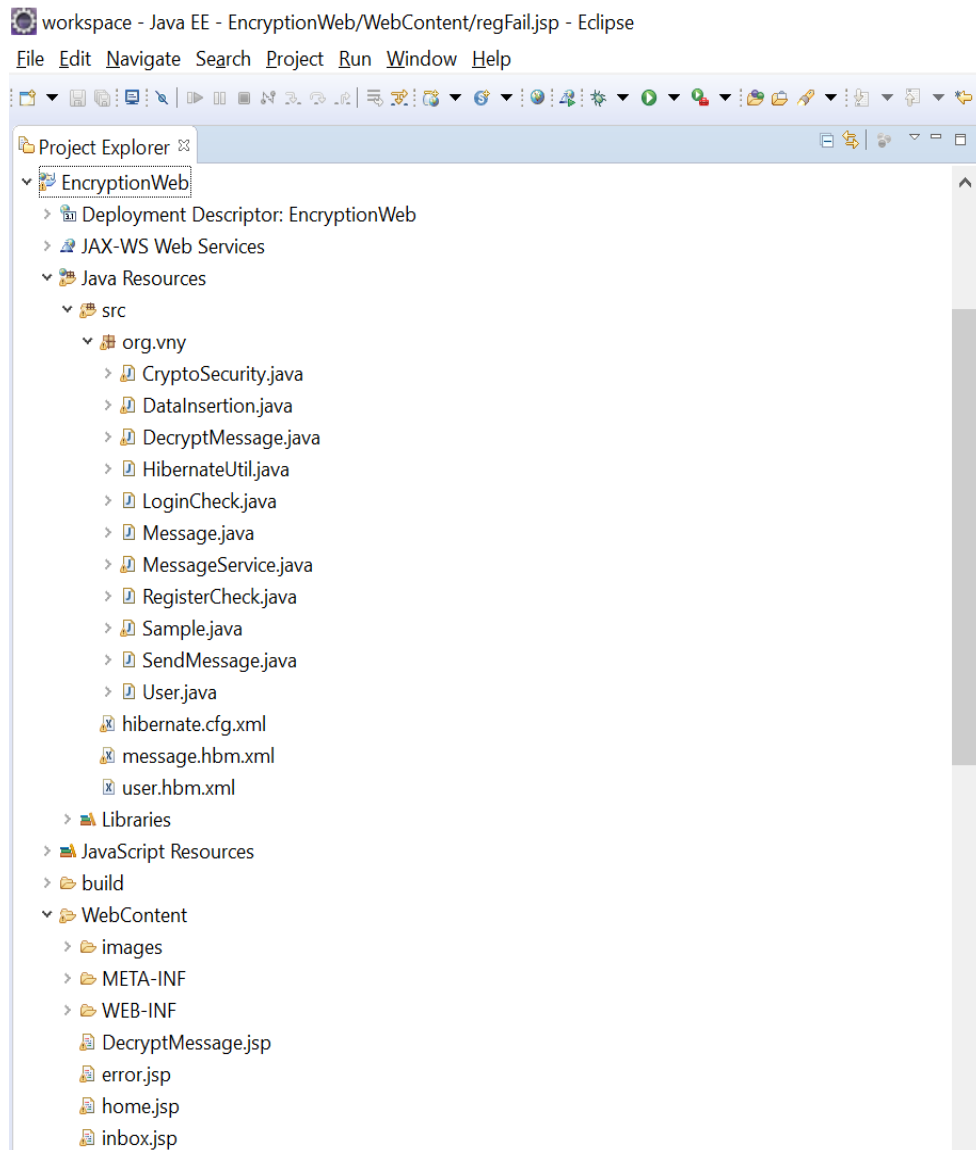


Figure 7. Folder Structure

Data Base

There are two major tables.

1. `user_info`:

To store the users information like first name, last name.

Model class: `User.java`

2. message_info:

To store the conversations between the users.

Model class: Message.java

```
mysql> Describe message_info;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| msg_id     | int(11)   | NO   | PRI | NULL    | auto_increment |
| message    | char(200) | NO   |     | NULL    |              |
| fromUname  | varchar(50) | NO   |     | NULL    |              |
| toUname    | varchar(50) | NO   |     | NULL    |              |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.10 sec)

mysql> Describe user_info;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)   | NO   | PRI | NULL    | auto_increment |
| name      | varchar(50) | NO   |     | NULL    |              |
| password  | varchar(50) | NO   |     | NULL    |              |
| fname     | varchar(50) | YES  |     | NULL    |              |
| lname     | varchar(50) | YES  |     | NULL    |              |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.09 sec)

mysql>
```

Figure 8. Database Tables

Project Screenshots

Following are the Project screenshots.

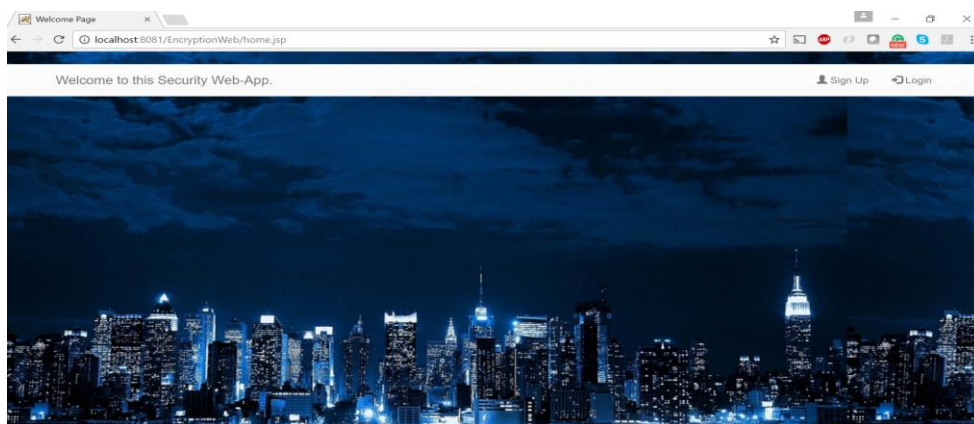
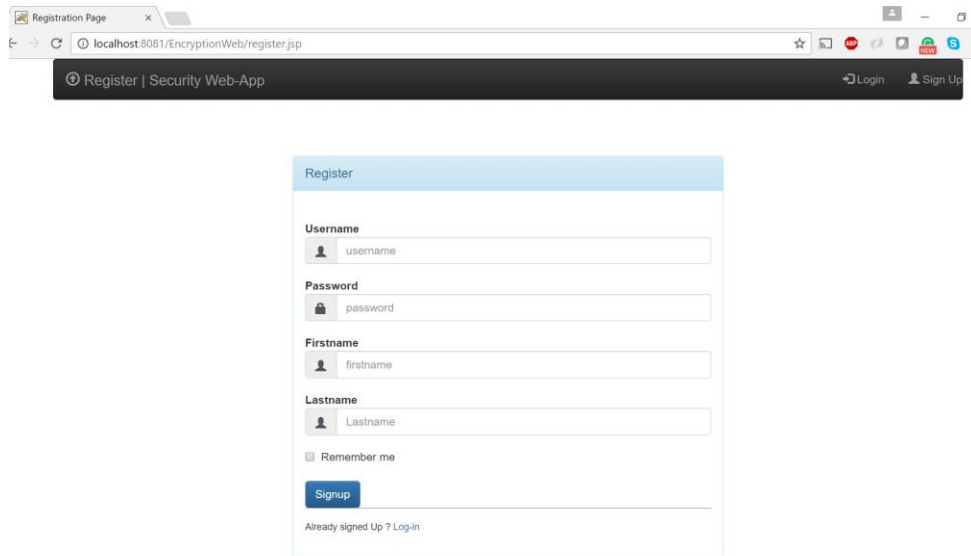


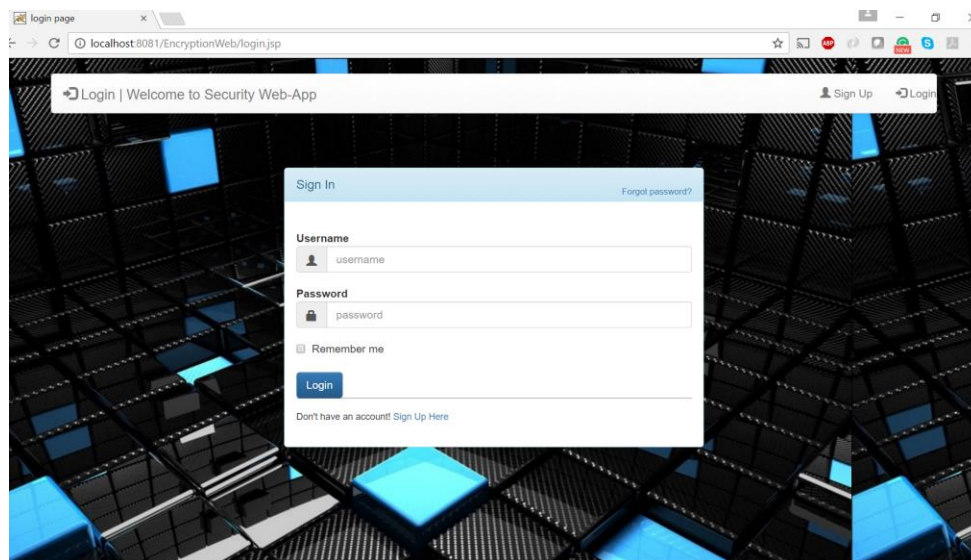
Figure 9. Welcome Screen



The screenshot shows a web browser window with the address bar displaying "localhost:8081/EncryptionWeb/register.jsp". The page title is "Register | Security Web-App". The main content is a registration form with the following fields and elements:

- Register** (Section Header)
- Username**: Input field with placeholder "username"
- Password**: Input field with placeholder "password"
- Firstname**: Input field with placeholder "firstname"
- Lastname**: Input field with placeholder "Lastname"
- Remember me
- Signup** (Button)
- Already signed Up ? [Log-In](#)

Figure 10. Signup Screen



The screenshot shows a web browser window with the address bar displaying "localhost:8081/EncryptionWeb/login.jsp". The page title is "Login | Welcome to Security Web-App". The main content is a login form with the following fields and elements:

- Sign In** (Section Header)
- Username**: Input field with placeholder "username"
- Password**: Input field with placeholder "password"
- Remember me
- Login** (Button)
- [Forgot password?](#)
- Don't have an account? [Sign Up Here](#)

Figure 11. Login Screen

localhost:8081/EncryptionWeb/member.jsp

Home Page | Security Web-App Inbox Logout

Welcome Vinay Ananthu

List of Friends:

User ID	First Name	Last Name	Options
vny.anant	Kumar	Abhi	Message
praveen.p	Praveen	Pasupu	Message
abhinay.anan	Abhinay	A	Message
denis.guster	Denis	Guster	Message

Figure 12. Home Screen

localhost:8081/EncryptionWeb/inbox.jsp

Home | Security Web-App. Back to Home Logout

Here are your messages:

??

— denis.guster

Key: [decrypt](#)

Figure 13. Inbox Screen

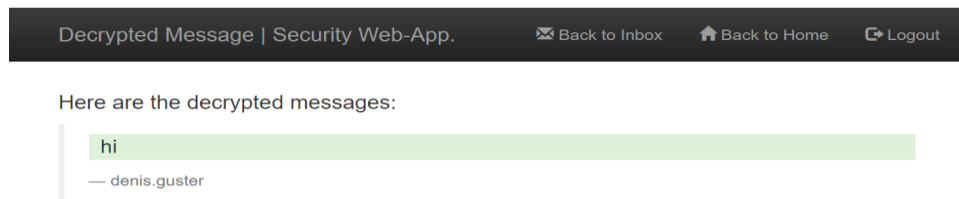


Figure 14. Decrypted Message Screen

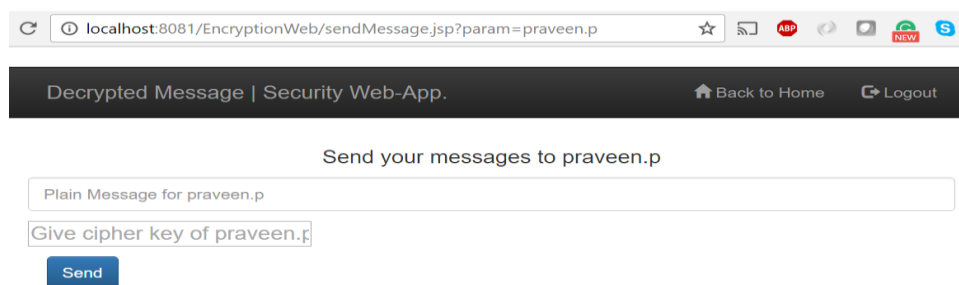


Figure 15. Message Screen

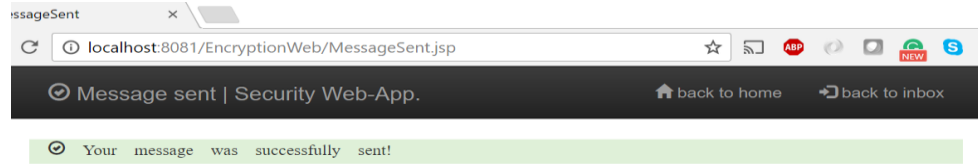


Figure 16. Message Sent Screen

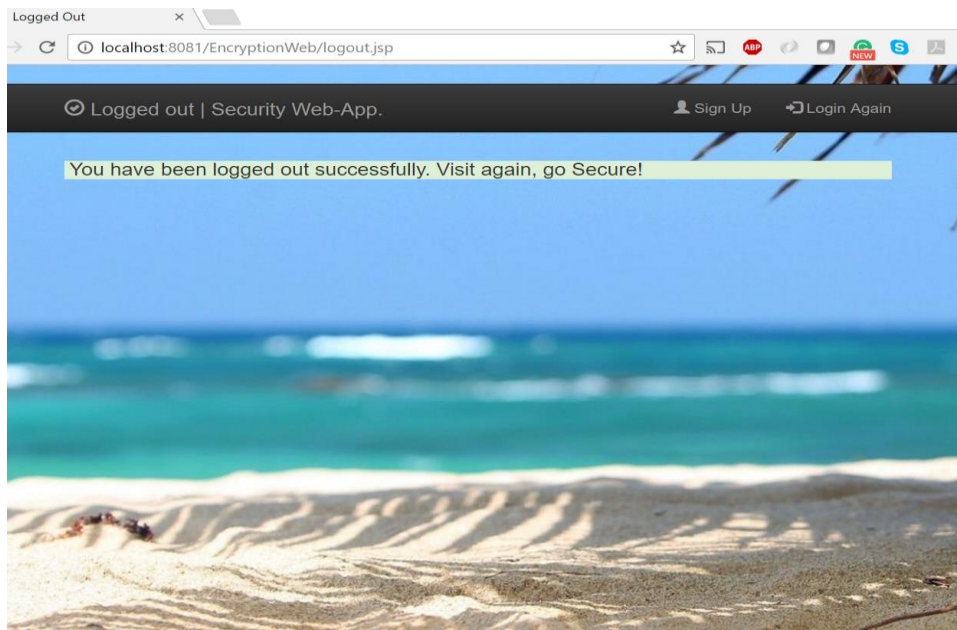


Figure 17. Logout Screen



Figure 18. Registration Failed Screen

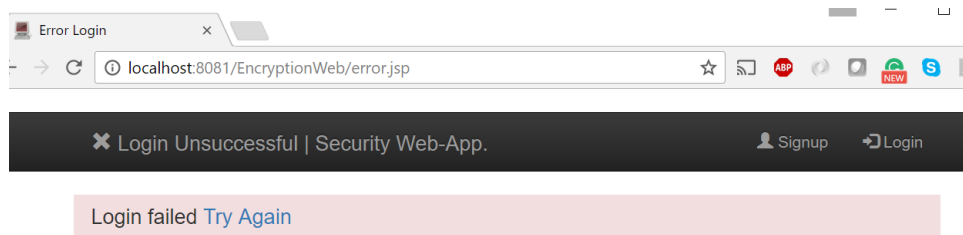


Figure 19. Logon Failed Screen

Chapter VI: Conclusion

Protecting user's data is the main concern of cloud computing today. As more and more cryptographic and mathematical tools are being used to generate multiple keys to access single application, they also become more complex. So the idea of compressing these multiple keys into a single master key obtained from different secret keys for various ciphertext classes in cloud is a considerable alternative. The main advantage of this concept is that the aggregate master key would be of constant size. This concept is more adaptable than canonical key distribution because it saves lot of space when all key-owners need same set of user privileges.

Future Work

In the current project, I used a simple and easy cipher key generation technique which is a derivation of Caesar cipher. In the future, the encryption technique can be more complicated, likely a choice from RSA or SHA key encryption techniques. Also, the aggregation of keys is not implemented in the current project. There is a scope to create aggregation of keys in future.

References

- Bitar, N., Gringeri, S., & Xia, T. J. (2013). Technologies and protocols for data center and cloud networking. *Communications Magazine, IEEE, 51*(9), 24-31.
- Boneh, D., & Hamburg, M. (2008). Generalized identity based and broadcast encryption schemes. In J. Peiprzyk (Ed.), *Advances in cryptology-ASIACRYPT 2008* (Vol. 5350). Berlin Heidelberg, Germany: Springer
- Cui, B., Liu, Z., & Wang, L. (2015). Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage. *Computers, IEEE Transactions On, PP*(99), 1-1.
- Dutta, R., & Annappa, B. (2014). Protection of data in unsecured public cloud environment with open, vulnerable networks using threshold-based secret sharing. *Network Protocols and Algorithms, 6*(1), 58-75.
- Krishna, N. S. R., & Prasad, Y. V. (2015). A novel system for scalable data sharing in cloud storage using key-aggregate. *IJSEAT, 3*(9), 461-465.
- Lewko, A., & Waters, B. (2011). Decentralizing attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011* (pp. 568-588). Berlin Heidelberg, Germany: Springer.
- Nurmi, D., Wolski, C., Grzegorzczak, G., Obertelli, S., Soman, L., . . . Zagordonov, D. (2009). *The eucalyptus open-source cloud-computing system*. Proceedings of 2009 ACM/IEEE International Conference on Grid Computing (pp. 124-131).
- Oberheide, J., Veeraraghavan, K., Cooke, E., Flinn, J., & Jahanian, F. (2008). *Virtualized in-cloud security services for mobile devices*. Proceedings of the First Workshop on Virtualization in Mobile Computing (pp. 31-35).

- Okuhara, M., Shiozaki, T., & Suzuki, T. (2010). Security architecture for cloud computing. *Fujitsu Sci. Tech. J*, 46(4), 397-402.
- Ramgovind, S., Eloff, M. M., & Smith E. (2010). *The management of security in cloud computing*. Retrieved from <http://uir.unisa.ac.za/bitstream/handle/10500/3883/ramgovind.pdf?sequence=1>
- Reddy, M. S. K., & Yadav, T. S. (2015). Public-key patient-controlled encryption for flexible data sharing in cloud storage. *International Journal of Innovative Technologies*, 3(1), 0005-0007.
- Ruj, S., Nayak, A., & Stojmenovic, I. (2011, November). Dacc: Distributed access control in clouds. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on* (pp. 91-98). IEEE.
- Schoo, P., Fusenig, V., Souza, V., Melo, M., Murray, P., Debar, H., . . . Zeglache, D. (2011). *Challenges for cloud networking security*. Paper presented at the International Conference on Mobile Networks and Management (pp. 298-313). Berlin Heidelberg, Germany: Springer.
- Wu, T. Y., Zhou, C., Wang, E. K., Pan, J. S., & Chen, C. M. (2014). Towards time-bound hierarchical key management in cloud computing. In *Intelligent Data Analysis and Its Applications, Volume I* (pp. 31-38). New York, NY: Springer International Publishing.
- Zhu, Z., Jiang, Z., & Jiang, R. (2013). *The attack on Mona: Secure multi-owner data sharing for dynamic groups in the cloud*. Proceedings of the 2013 International Conference on Information Science and Cloud Computing Companion (pp. 514-519).

Appendix

hibernate.cfg.xml:

```

<?xml version="1.0" encoding="UTF-8"?>
<hibernate-configuration>
  <session-factory>

<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="connection.url">jdbc:mysql://localhost:3306/mydb</property>
  <property name="connection.username">root</property>
  <property name="connection.password">Password@123</property>

  <!-- JDBC connection pool (use the built-in) -->
  <property name="connection.pool_size">1</property>

  <!-- SQL dialect -->
  <property name="dialect">
    org.hibernate.dialect.MySQLDialect
  </property>
  <!-- Enable Hibernate's automatic session context management -->
  <property name="current_session_context_class">thread</property>

  <property
name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>
  <!-- Echo all executed SQL to stdout -->
  <property name="show_sql">>true</property>
  <!--<property name="dialect">org.hibernate.dialect.MySQLDialect</property> -->
  <!-- Drop existing tables and create new one -->
    <property name="hbm2ddl.auto">update</property>
  <!--<mapping class="org.vnyseries.hibernate.Student_Info"/>-->
    <mapping resource="user.hbm.xml"/>
    <mapping resource="message.hbm.xml"/>

  </session-factory>

</hibernate-configuration>

```

message.hbm.xml:

```

<?xml version="1.0" encoding="UTF-8"?>

<hibernate-mapping>

```

```

<class name="org.vny.Message" table="message_info">
  <id name="msg_id" column="msg_id">
    <generator class="assigned" />
  </id>

  <property name="message" column="message" />
  <property name="fromUname" column="fromUname" />
  <property name="toUname" column="toUname" />
</class>
</hibernate-mapping>

```

user.hbm.xml:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class name="org.vny.User" table="user_info">
    <id name="user_id" column="id">
      <generator class="assigned" />
    </id>

    <property name="user_name" column="name" />
    <property name="password" column="password" />
    <property name="fname" column="fname" />
    <property name="lname" column="lname" />
  </class>
</hibernate-mapping>

```

CryptoSecurity.java:

```
package org.vny;
```

```
import java.security.Security;
```

```
import javax.crypto.Cipher;
```

```
import javax.crypto.spec.SecretKeySpec;
```

```
public class CryptoSecurity {
  public static void main(String[] args) throws Exception {
    String cipher = new CryptoSecurity().encryptString("Hello 123 ????",12);

```

```

System.out.println(cipher);
String plain = new CryptoSecurity().decryptString(cipher,12);
System.out.println(plain);
}

```

```

public String encryptString(String plainText,int key){
    StringBuffer input = new StringBuffer(plainText);
    for(int i=0;i<input.length();i++){
        int temp=0;
        temp=(int)input.charAt(i);
        temp=temp*key;
        input.setCharAt(i, (char)temp);
    }
    String decipher = input.toString();
    return decipher;
}

```

```

public String decryptString(String cipherText,int key){
    System.out.println("Decrypting "+cipherText+" with "+key);
    StringBuffer input = new StringBuffer(cipherText);
    for(int i=0;i<input.length();i++){
        int temp=0;
        temp=(int)input.charAt(i);
        temp=temp/key;
        input.setCharAt(i, (char)temp);
    }
    String plain = input.toString();
    System.out.println("Decrypted Text:"+plain);
    return plain;
}
}

```

DataInsertion.java:

```

package org.vny;

```

```

import java.util.ArrayList;
import java.util.List;

```

```

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import org.hibernate.query.Query;

```

```

public class DataInsertion {

    public static void main(String[] args) {
    }

    public void insertInfo(User user) {
        Session session = HibernateUtil.openSession();
        Transaction tx = null;
        try {
            tx = session.getTransaction();
            tx.begin();
            session.saveOrUpdate(user);
            tx.commit();
        } catch (Exception ex) {
            System.out.println(ex);
            if (tx != null) {
                tx.rollback();
            }
        } finally {
            session.close();
        }
    }

    public User getUserInfo() {
        Session session = HibernateUtil.openSession();
        Transaction tx = null;
        User user = null;
        try {
            tx = session.getTransaction();
            tx.begin();
            user = session.get(User.class, 2);
            tx.commit();
        } catch (Exception ex) {
            if (tx != null) {
                tx.rollback();
            }
        } finally {
            session.close();
        }

        return user;
    }
}

```

```

public Boolean validatePassword(String usn, String pwd) {
    User user = getUserByUserName(usn);
    if (user != null && user.getUser_name().equals(usn) &&
user.getPassword().equals(pwd)) {
        return true;
    } else {
        return false;
    }
}

public Boolean checkUserNameExists(String usn) {
    Session session = HibernateUtil.openSession();
    Transaction tx = null;
    List<User> users = null;
    try {
        tx = session.getTransaction();
        tx.begin();
        Query query = session.createQuery("from User where user_name = " +
"" + usn + "");
        users = (List<User>) query.list();
        tx.commit();
    } catch (Exception ex) {
        if (tx != null) {
            tx.rollback();
        }
    } finally {
        session.close();
    }

    if (users.size() != 0) {
        return false;
    } else {
        return true;
    }
}

public User getUserByUserName(String uname) {
    Session session = HibernateUtil.openSession();
    Transaction tx = null;
    List<User> users = null;
    try {
        tx = session.getTransaction();
        tx.begin();

```

```

        """ + uname + """);
        Query query = session.createQuery("from User where user_name = " +
            users = (List<User>) query.list();

        session.getTransaction().commit();
    } catch (Exception ex) {
        if (tx != null) {
            tx.rollback();
        }
    } finally {
        session.close();
    }
    return users.get(0);
}

public List<User> getListOfUsers() {
    List<User> list = new ArrayList<User>();
    Session session = HibernateUtil.openSession();
    Transaction tx = null;
    try {
        tx = session.getTransaction();
        tx.begin();
        list = session.createQuery("from User").list();
        tx.commit();
    } catch (Exception e) {
        if (tx != null) {
            tx.rollback();
        }
        e.printStackTrace();
    } finally {
        session.close();
    }
    return list;
}
}

```

```

}

```

DecryptMessage.java

```

package org.vny;

```

```

import java.io.IOException;

```

```

import javax.servlet.ServletException;

```



```

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class DecryptMessage
 */
@WebServlet("/DecryptMessage")
public class DecryptMessage extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public DecryptMessage() {
        super();
        // TODO Auto-generated constructor stub
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
     * response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        MessageService messageService = new MessageService();
        CryptoSecurity crypto = new CryptoSecurity();
        int key = Integer.parseInt(request.getParameter("key"));
        String cypherText = (String) request.getParameter("param");
        String decryptMessage = crypto.decryptString(cypherText, key);
        request.getSession().setAttribute("decryptMessage", decryptMessage);
        response.sendRedirect("DecryptMessage.jsp");
    }
}

```

HibernateUtil.java:

```

package org.vny;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static Session openSession() {
        return sessionFactory.openSession();
    }
}

```

LoginCheck.java

```

package org.vny;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class LoginCheck
 */
@WebServlet("/LoginCheck")
public class LoginCheck extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**

```

```

    * @see HttpServlet#HttpServlet()
    */
    public LoginCheck() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     * response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
     * response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        // doGet(request, response);
        User user = null;
        String uname = request.getParameter("uname");
        String password = request.getParameter("password");
        DataInsertion data = new DataInsertion();
        if (!data.checkUserNameExists(uname) && !password.equals(null) &&
!uname.equals(null)) {
            user = data.getUserByUserName(uname);
            if (data.validatePassword(uname, password)) {
                request.getSession().setAttribute("user", user);
                response.sendRedirect("member.jsp");
            } else {
                response.sendRedirect("error.jsp");
            }
        } else {
            response.sendRedirect("error.jsp");
        }
    }
}

```

Message.java:

```

package org.vny;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "message_info")
public class Message {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int msg_id;
    private String message;
    private String fromUname;
    private String toUname;

    public int getMsg_id() {
        return msg_id;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public void setMsg_id(int msg_id) {
        this.msg_id = msg_id;
    }

    public String getFromUname() {
        return fromUname;
    }

    public void setFromUname(String fromUname) {
        this.fromUname = fromUname;
    }

    public String getToUname() {
        return toUname;
    }
}

```

```

    }
    public void setToUname(String toUname) {
        this.toUname = toUname;
    }
}

```

MessageService.java:

```

package org.vny;

import java.util.List;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.query.Query;

public class MessageService {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Message msg = new Message();
        msg.setFromUname("vny.anan");
        msg.setToUname("praveen.p");
        // msg.setMessage("Hi Praveen");
        new MessageService().deleteMessage(msg);
    }

    public void insertMessage(Message msg) {
        Session session = HibernateUtil.openSession();
        Transaction tx = null;
        try {
            tx = session.getTransaction();
            tx.begin();
            session.saveOrUpdate(msg);
            tx.commit();
        } catch (Exception ex) {
            System.out.println(ex);
            if (tx != null) {
                tx.rollback();
            }
        } finally {
            session.close();
        }
    }
}

```

```

}

public void deleteMessage(Message msg) {
    Session session = HibernateUtil.openSession();
    Transaction tx = null;
    List<Message> mesgs = null;
    Message mesg = null;
    try {
        tx = session.getTransaction();
        tx.begin();
        Query query = session.createQuery("from Message where fromUname
=“ + “““ + msg.getFromUname()
                                + ““ AND toUname =“ + “““ + msg.getToUname() + ““
AND message =“ + “““ + msg.getMessage() + “““);
        mesgs = (List<Message>) query.list();
        mesg = session.get(Message.class, mesgs.get(0).getMsg_id());
        session.delete(mesg);
        tx.commit();
    } catch (Exception ex) {
        System.out.println(ex);
        if (tx != null) {
            tx.rollback();
        }
    } finally {
        session.close();
    }
}

public List<Message> getMessagesByUserName(String toUSN) {
    Session session = HibernateUtil.openSession();
    Transaction tx = null;
    List<Message> messages = null;
    try {
        tx = session.getTransaction();
        tx.begin();
        Query query = session.createQuery("from Message where toUname =“ +
“““ + toUSN + “““);
        messages = (List<Message>) query.list();

        session.getTransaction().commit();
    } catch (Exception ex) {
        if (tx != null) {
            tx.rollback();
        }
    }
}

```

```

        } finally {
            session.close();
        }
        return messages;
    }
}

```

RegisterCheck.java:

```

package org.vny;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/RegisterCheck")
public class RegisterCheck extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public RegisterCheck() {
        super();
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     * response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String uname = request.getParameter("uname");
        String password = request.getParameter("password");
        String fname = request.getParameter("fname");
        String lname = request.getParameter("lname");
        User user = new User();
        // user.setUser_id(1);
    }
}

```

```

        user.setUser_name(uname);
        user.setPassword(password);
        user.setLname(lname);
        user.setFname(fname);
        DataInsertion data = new DataInsertion();
        if (data.checkUserNameExists(uname) && !uname.equals("") &&
!password.equals("")) {
            data.insertInfo(user);
            response.sendRedirect("regSuccess.jsp");
        } else {
            response.sendRedirect("regFail.jsp");
        }
    }
}

```

SendMessage.java:

```

package org.vny;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class SendMessage
 */
@WebServlet("/SendMessage")
public class SendMessage extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public SendMessage() {
        super();
        // TODO Auto-generated constructor stub
    }

}

```



```

* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
*   response)
*/
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // TODO Auto-generated method stub
    response.getWriter().append("Served at: ").append(request.getContextPath());
}

/**
* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
*   response)
*/
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // TODO Auto-generated method stub
    // doGet(request, response);
    String msg = (String) request.getParameter("message");
    System.out.println("Message:" + msg);
    MessageService messageService = new MessageService();
    Message message = new Message();
    int key = 1;
    CryptoSecurity cryptoSec = new CryptoSecurity();
    if (!msg.equals("")) {
        System.out.println((String) request.getParameter("toUSN"));
        System.out.println((String) request.getParameter("fromUSN"));
        message.setFromUname((String) request.getParameter("fromUSN"));
        message.setToUname((String) request.getParameter("toUSN"));
        String mesg = request.getParameter("message");
        if (request.getParameter("key") != null) {
            key = Integer.parseInt(request.getParameter("key"));
        } else {
            key = 1;
        }
        if (key < 1) {
            key = 1;
        }
        message.setMessage(cryptoSec.encryptString(mesg, key));
        // message.setSkey(key);
        messageService.insertMessage(message);
        response.sendRedirect("MessageSent.jsp");
    } else {
        response.sendRedirect("error.jsp");
    }
}

```

```

    }
}

```

User.java:

```

package org.vny;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name="user_info")
public class User {
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
    private int user_id;
    private String user_name;
    private String password;
    private String fname;
    private String lname;
    public int getUser_id() {
        return user_id;
    }
    public void setUser_id(int user_id) {
        this.user_id = user_id;
    }
    public String getUser_name() {
        return user_name;
    }
    public void setUser_name(String user_name) {
        this.user_name = user_name;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getFname() {
        return fname;
    }
}

```

```

    public void setFname(String fname) {
        this.fname = fname;
    }
    public String getLname() {
        return lname;
    }
    public void setLname(String lname) {
        this.lname = lname;
    }
}

```

Front End Code:

DecryptMessage.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@page import="java.util.List"%>
<%@page import="org.vny.*"%>
<%@page import="java.util.Date"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
<link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
    integrity="sha384-
BVYüSiFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
    crossorigin="anonymous">
<link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
    integrity="sha384-
rHyoN1iRsVXV4nD0JutlNGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
    crossorigin="anonymous">
<script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
    integrity="sha384-
Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA712mCWNIpG9mGCD8wGNlCPD7Txa"
    crossorigin="anonymous"></script>
</head>
<body>

```

```

try{
    <%
    if(session.getAttribute("user")!=null)
    {
    %>
    <br />
    <nav class="navbar navbar-inverse">
    <div class="container">
        <div class="navbar-header">
            <a class="navbar-brand" href="#"> Decrypted Message | Security
            Web-App.</a>
        </div>
        <ul class="nav navbar-nav navbar-right">
            <li><a href="inbox.jsp"><span
                class="glyphicon glyphicon-envelope"></span> Back to
            Inbox</a></li>
            <li><a href="member.jsp"><span
                class="glyphicon glyphicon-home"></span> Back to
            Home</a></li>
            <li><a href="logout.jsp"><span
                class="glyphicon glyphicon-log-out"></span>
            Logout</a></li>
        </ul>
    </div>
    </nav>
    <div class="container">
        <h4>Here are the decrypted messages:</h4>
    </div>
    <%
    //String plainMessage = (String)session.getAttribute("decryptMessage");
    MessageService messageService = new MessageService();
    CryptoSecurity crypto = new CryptoSecurity();
    int skey = Integer.parseInt(request.getParameter("skey"));
    User user = (User) session.getAttribute("user");
    List<Message> list =
    messageService.getMessagesByUserName(user.getUserName());
    String message = "";
    for (Message m : list) {
    %>
    <div class="container">

```

```

<blockquote>
  <p class="bg-success">
    &nbsp;
    <%=crypto.decryptString(m.getMessage(), skey)%>

    <%-- <%
    int skey= Integer.parseInt(request.getParameter("key"));
    CryptoSecurity crypto = new CryptoSecurity();
    String decryptedMsg = crypto.decryptString(m.getMessage(), skey);
    %> --%>
  </p>

  <footer><%=m.getFromUname()%></footer>
</blockquote>
</div>

  <%
  _____}
  _____}
  _____else{
  _____response.sendRedirect("login.jsp");
  _____}
  _____}catch(Exception E){
  _____response.sendRedirect("login.jsp");
  _____}
  _____//String decryptMessage= crypto.decryptString(cypherText, key);
  _____%>
  _____<br />
</body>
</html>

```

error.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Error Login</title>
<link rel="stylesheet"
  href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"

```

```

    integrity="sha384-
BVYiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
    crossorigin="anonymous">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
integrity="sha384-
rHyoNIrSvXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
crossorigin="anonymous">
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-
Tc5Iqib027qvyjSMfHjOMaLkfuWVxZxUPnCJA712mCWNIpG9mGCD8wGNlCPD7Txa"
crossorigin="anonymous"></script>
</head>
<body>
<br />
    <nav class="navbar navbar-inverse">
    <div class="container">
        <div class="navbar-header">
            <a class="navbar-brand" href="#"><span
                class="glyphicon glyphicon-remove"></span> Login
Unsuccessful / Security
                Web-App. </a>
        </div>
        <ul class="nav navbar-nav navbar-right">
            <li><a href="register.jsp"><span
                class="glyphicon glyphicon-user"></span> Signup
            </a></li>
            <li><a href="login.jsp"><span
                class="glyphicon glyphicon-log-in"></span> Login
            </a></li>
        </ul>
    </div>
</nav>
<div class="container bg-danger">
    <h4>Login failed <a href="login.jsp"> Try Again</a></h4>
</div>
</body>
</html>

```

home.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
    integrity="sha384-
BVYüSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
    crossorigin="anonymous">
<link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
    integrity="sha384-
rHyoNIiRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
    crossorigin="anonymous">
<script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
    integrity="sha384-
Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIPg9mGCD8wGNlCPD7Txa"
    crossorigin="anonymous"></script>

<link rel="shortcut icon" href="/images/BG.png" type="image/x-icon" />

<title>Welcome Page</title>
</head>
<body background="images/BG7.jpg">
    <br />
    <nav class="navbar navbar-default">
    <div class="container">
        <div class="navbar-header">
            <a class="navbar-brand" href="#">Welcome to this Security
                Web-App. </a>
        </div>

        <ul class="nav navbar-nav navbar-right">
            <li><a href="register.jsp"><span
                class="glyphicon glyphicon-user"></span> Sign
                Up</a></li>
            <li><a href="login.jsp"><span

```

```

class="glyphicon glyphicon-log-in"></span>
Login</a></li>
    </ul>
</div>
</nav>
</body>
</html>

```

inbox.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="org.vny.*"%>
<%@ page import="java.util.List"%>
<%@ page import="org.vny.*"%>
<%@ page import="java.util.Date"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Inbox</title>
<link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
    integrity="sha384-
BVYiSIFeKIdGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
    crossorigin="anonymous">
<link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
    integrity="sha384-
rHyoNIiRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
    crossorigin="anonymous">
<script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
    integrity="sha384-
Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7I2mCWNIPG9mGCD8wGNICPD7Txa"
    crossorigin="anonymous"></script>
<script type="text/javascript">
function load_DecryptMsg(clicked_id)
{
    value1=document.getElementById("skey").value;

```



```

window.location="http://localhost:8081/EncryptionWeb/DecryptMessage.jsp?param="+click
d_id+"&skey="+value1;
}
</script>
</head>
<%try{
    if(session.getAttribute("user")!=null)
    {
%>

<body>
    <br />

    <nav class="navbar navbar-inverse">
    <div class="container">
        <div class="navbar-header">
            <a class="navbar-brand" href="#"><span
                class="glyphicon glyphicon-envelope"></span> Home | Security
                Web-App. </a>
        </div>

        <ul class="nav navbar-nav navbar-right">
            <li><a href="member.jsp"><span
                class="glyphicon glyphicon-home"></span> Back to
Home</a></li>
            <li><a href="logout.jsp"><span
                class="glyphicon glyphicon-log-out"></span>
Logout</a></li>
        </ul>
    </div>
</nav>

    <div class="container">
        <%
            User user = (User) session.getAttribute("user");
            MessageService messageService = new MessageService();
            List<Message> list =
            messageService.getMessagesByUserName(user.getUser_name());
            String message = "";
            if(list.isEmpty()){%>
                <h4>Your inbox is empty.</h4>
            <%
                }

```


login.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>login page</title>
</head>
<body background="images/BG9.jpg">
    <br />
    <div class="container">
        <nav class="navbar navbar-default">
            <div class="navbar-header">
                <a class="navbar-brand" href="#"> <span
class="glyphicon glyphicon-log-in"></span> Login | Welcome to Security
Web-App</a>
            </div>
            <ul class="nav navbar-nav navbar-right">
                <li><a href="register.jsp"><span
class="glyphicon glyphicon-user"></span> Sign Up</a></li>
                <li><a href="login.jsp"><span
class="glyphicon glyphicon-log-in"></span> Login</a></li>
            </ul>
        </nav>
        <div id="loginbox" style="margin-top: 50px;"
class="mainbox col-md-6 col-md-offset-3 col-sm-8 col-sm-offset-2">
            <div class="panel panel-info">
                <div class="panel-heading">
                    <div class="panel-title">Sign In</div>
                </div>
                <div style="float: right; font-size: 80%; position: relative; top: -10px">
                    <a href="#">Forgot password?</a>
                </div>
            </div>
            <div style="padding-top: 30px" class="panel-body">
                <form class="form-horizontal" method="post" action="LoginCheck">
                    <fieldset>
                        <label class="control-label" for="uname">Username</label>
                        <div style="margin-bottom: 10px" class="input-group">
                            <span class="input-group-addon"><i>

```

logout.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Logged Out</title>
</head>
<body background="images/BG10.jpg" class="img-responsive">
  <br />
  <nav class="navbar navbar-inverse">
    <div class="container">
      <div class="navbar-header">
        <a class="navbar-brand" href="#"><span
          class="glyphicon glyphicon-ok-circle"></span> Logged
out | Security
          <span class="glyphicon glyphicon-user"></span> Sign
          <span class="glyphicon glyphicon-log-in"></span> Login Again</a></li>
        <li><a href="register.jsp"><span
          class="glyphicon glyphicon-user"></span> Sign
          <span class="glyphicon glyphicon-log-in"></span> Login Again</a></li>
        <li><a href="login.jsp"><span
          class="glyphicon glyphicon-log-in"></span> Login Again</a></li>
      </ul>
    </div>
  </nav>
  <%
    session.setAttribute("user",null);
    session.invalidate();
  %>
  <div class="container">
    <div class="bg-success">
      <h4> &nbsp;   You have been logged out successfully. Visit again, go Secure!</h4>
    </div>
  </div>
</body>
</html>

```

Member.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@page import="org.vny.*"%>

```

```

<%@page import="java.util.List"%>
<%@page import="org.vny.*"%>
<%@page import="java.util.Date"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome page</title>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-
BVYüSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
integrity="sha384-
rHyoNIiRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
crossorigin="anonymous">
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-
Tc5Iqib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7I2mCWNIPg9mGCD8wGNIcPD7Txa"
crossorigin="anonymous"></script>
<script type="text/javascript">
function load_SendMsg(clicked_id) {
window.location =
"http://localhost:8081/EncryptionWeb/sendMessage.jsp?param=" +
clicked_id;
}
</script>
<style type="text/css">
.black-background {
background-color: #000000;
}

.white {
color: #ffffff;
}
</style>
</head>
<body>
<%
try {

```

```

        if (session.getAttribute("user") != null) {
%>

    <br />
    <nav class="navbar navbar-inverse">
    <div class="container">
        <div class="navbar-header">
            <a class="navbar-brand" href="#"><span
                class="glyphicon glyphicon-home"></span> Home Page |
Security
                Web-App </a>
        </div>

            <ul class="nav navbar-nav navbar-right">
                <li><a href="inbox.jsp"><span
                    class="glyphicon glyphicon-envelope"></span>
Inbox</a></li>
                <li><a href="logout.jsp"><span
                    class="glyphicon glyphicon-log-in"></span>
Logout</a></li>
            </ul>
        </div>
    </nav>

%>
    User user = (User) session.getAttribute("user");
%>

    <div class="container" id="container">

        <div class="row">

            <div class="col-md-9">
                <h3>
                    <u>Welcome <%=user.getFname() + " " +
user.getLname()%></u></h3>
                </div>
            </div>
            <div class="row">

                <div class="col-md-9">
                    <h3>
                        <small>List of Friends:</small>

```

```

        </h3>
    </div>
    <!-- <div class="col-md-3"><%=new Date()%></br></div> --%>
</div>
</div>
<div class="container">
    <table class="table table-hover">
        <thead>
            <tr>
                <th>User ID</th>
                <th>First Name</th>
                <th>Last Name</th>
                <th>Options</th>
            </tr>
        </thead>
        <tbody>
            <%
                DataInsertion data = new DataInsertion();
                List<User> list = data.getListOfUsers();
                for (User u : list) {
                    if
(!u.getUser_name().equals(user.getUser_name())) {
                %>
                <tr>
                    <td><%=u.getUser_name()%></td>
                    <td><%=u.getFname()%></td>
                    <td><%=u.getLname()%></td>
                    <%
                        session.setAttribute("fromUSN",
user.getUser_name());
                    %>
                    <td>
                        <button id="<%=u.getUser_name()%>"
                            class="btn btn-primary black-background
white"
                            onclick="load_SendMsg(this.id)">Message</button>
                    </td>
                </tr>
                <%
                    }
                %>
            %>
        </tbody>
    </table>
</div>

```

```

        <tbody>
    </table>
    <br />
</div>
<%
    } else {
        response.sendRedirect("login.jsp");
    }
    } catch (Exception E) {
        response.sendRedirect("login.jsp");
    }
%>
</body>
</html>

```

MessageSent.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>MessageSent</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-
theme.min.css" integrity="sha384-
rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
crossorigin="anonymous">
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-
Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7I2mCWNIpG9mGCD8wGNlCPD7Txa"
crossorigin="anonymous"></script>

</head>
<%if(session.getAttribute("user")!=null){%>
<body>
    <nav class="navbar navbar-inverse">
    <div class="container">
        <div class="navbar-header">
            <a class="navbar-brand" href="#"><span

```



```

class="glyphicon glyphicon-ok-circle"></span> Message
sent / Security
Web-App. </a>
</div>
<ul class="nav navbar-nav navbar-right">
  <li><a href="member.jsp"><span
class="glyphicon glyphicon-home"></span> back to home</a></li>
  <li><a href="inbox.jsp"><span
class="glyphicon glyphicon-log-in"></span> back to
inbox</a></li>
</ul>
</div>
</nav>
<div class="container bg-success">
  <div class="bg-success">
    <p><span class="glyphicon glyphicon-ok-circle"> Your message was successfully
sent!</span></p>
  </div>
</div>
</body>
<%> else{
    response.sendRedirect("login.jsp");
} %>
</html>

```

regFail.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Registration failed</title>
</head>
<%
    try {
%>
<body>
  <br />
  <div class="container">

    <nav class="navbar navbar-inverse">

```

```

        <div class="navbar-header">
            <a class="navbar-brand" href="#"><span
                class="glyphicon glyphicon-remove"></span> Security Web-
App</a>
        </div>
    </nav>
</div>
<div class="container">
    <div>
        <h4>
            <dl>
                <dt>
                    <span class="glyphicon glyphicon-remove"></span>Registration
failed due to one of the following reasons:
                </dt>
                <small><dd>- Username already exists</dd>
                <dd>- Username is invalid.</dd>
                <dd>- Password is invalid.</dd></small>
            </dl>
        </h4>
    </div>
    <div>
        <form method="post" action="register.jsp">
            <button type="submit" class="btn">Try Again</button>
        </form>
    </div>
</div>
</body>
<%
    } catch (Exception E) {
        response.sendRedirect("login.jsp");
    }
%>
</html>

```

Register.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>

```

```

<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Registration Page</title>

<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
      integrity="sha384-
BVYüSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
      crossorigin="anonymous">
<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
      integrity="sha384-
rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
      crossorigin="anonymous">
<script
      src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
      integrity="sha384-
Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7I2mCWNIpG9mGCD8wGNlCPD7Txa"
      crossorigin="anonymous"></script>

</head>
<body>
<div class="container">

<nav class="navbar navbar-inverse">
  <div class="navbar-header">
    <a class="navbar-brand" href="#"><span
      class="glyphicon glyphicon-upload"></span> Register | Security
    Web-App</a>
  </div>
  <ul class="nav navbar-nav navbar-right">
    <li><a href="login.jsp"><span
      class="glyphicon glyphicon-log-in"></span> Login</a></li>
    <li><a href="register.jsp"><span
      class="glyphicon glyphicon-user"></span> Sign Up</a></li>
  </ul>
</nav>
<div id="loginbox" style="margin-top: 50px;"
class="mainbox col-md-6 col-md-offset-3 col-sm-8 col-sm-offset-2">
  <div class="panel panel-info">
  <div class="panel-heading">
  <div class="panel-title">Register</div>
</div>

```

```

<div style="padding-top: 30px" class="panel-body">
<form class="form-horizontal" method="post" action="RegisterCheck">
<fieldset>
  <label class="control-label" for="uname">Username</label>
  <div style="margin-bottom: 10px" class="input-group">
<span class="input-group-addon"><i
class="glyphicon glyphicon-user"></i></span> <input
                                                                    id="login-
username" type="text" class="form-control
name="uname" value="" placeholder="username">
</div>

<label class="control-label" for="password">Password</label>
<div style="margin-bottom: 10px" class="input-group">
<span class="input-group-addon"><i
class="glyphicon glyphicon-lock"></i></span> <input
id="login-password" type="password" class="form-control"
name="password" placeholder="password">
</div>

<label class="control-label" for="uname">Firstname</label>
<div style="margin-bottom: 10px" class="input-group">
  <span class="input-group-addon"><i
class="glyphicon glyphicon-user"></i></span> <input
id="login-username" type="text" class="form-control"
name="fname" value="" placeholder="firstname">
</div>

<label class="control-label" for="uname">Lastname</label>
<div style="margin-bottom: 10px" class="input-group">
<span class="input-group-addon"><i
class="glyphicon glyphicon-user"></i></span> <input
id="login-username" type="text" class="form-control"
name="lname" value="" placeholder="Lastname">
</div>
<div class="input-group">
<div class="checkbox">
<label> <input id="login-remember" type="checkbox"
name="remember" value="1"> Remember me
</label>
</div>
</div>
</div>
<br />
<div class="controls">

```

```

<button class="btn btn-primary">Signup</button>
</div>
</fieldset>
<div class="form-group">
<div class="col-md-12 control">
<div
style="border-top: 1px solid #888; padding-top: 15px; font-size: 85%">
Already signed Up ? <a href="login.jsp"
onClick="$('#loginbox').hide(); $('#signupbox').show()">
</a>
</div>
</div>
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</body>
</html>

```

Log-in

regSuccess.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@page import="org.vny.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Successful Registration</title>
</head>
<%try{ %>
<body>
<br />
<div class="container">
<nav class="navbar navbar-inverse">
<div class="navbar-header">

```

```

        <a class="navbar-brand" href="#"><span
            class="glyphicon glyphicon-ok"></span> Security Web-App</a>
    </div>
</nav>
</div>
<div class="container">
    <div>
        <h4>
<span class="glyphicon glyphicon-ok"></span> Registration Successful.

        </h4>
    </div>
    <div>
        <form method="post" action="login.jsp">
            <button type="submit" class="btn">Log in</button>
        </form>
    </div>
</div>
</body>
<%>catch(Exception E){
    response.sendRedirect("login.jsp");
}<%>
</html>

```

sendMessage.jsp:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Send Message</title>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-
BVYüSiFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-
theme.min.css" integrity="sha384-
rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
crossorigin="anonymous">

```

```

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-
Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7I2mCWNIpG9mGCD8wGNiCpD7Txa"
crossorigin="anonymous"></script>
</head>
<body>
<%
try{

    if(session.getAttribute("user")!=null)
    {

%>
<br/>
<nav class="navbar navbar-inverse">
<div class="container">
    <div class="navbar-header">
        <a class="navbar-brand" href="#"> Decrypted Message | Security
        Web-App.</a>
    </div>

    <ul class="nav navbar-nav navbar-right">
        <li><a href="member.jsp"><span
            class="glyphicon glyphicon-home"></span> Back to
Home</a></li>
        <li><a href="logout.jsp"><span
            class="glyphicon glyphicon-log-out"></span>
Logout</a></li>
    </ul>
    </div>
</nav>

<div class="container">
<form method="post" action="SendMessage">
<%
    String var = request.getParameter("param");
    session.setAttribute("toUSN", var);
%>
<h4 class="text-center">Send your messages to <%=session.getAttribute("toUSN") %></h4>

    <div class="row">
        <div class="col-mid-6" style="margin-bottom: 10px">
            <input id="msg" type="text" class="form-control"
name="message" placeholder="Plain Message for <%=session.getAttribute("toUSN") %>">

```

```

</div>

<div class="col-mid-2" style="margin-bottom: 10px">
    <input id="key" type="text" class="" name="key"
placeholder="Give cipher key of <%=session.getAttribute("toUSN") %>">
</div>
</div>
<input type="hidden" name="toUSN"
value="<%=session.getAttribute("toUSN")%>">
<input type="hidden" name="fromUSN"
value="<%=session.getAttribute("fromUSN")%>">
<input type="submit" class="btn btn-primary" value =
"Send">

</form>
</div>
<%
}
else{
    response.sendRedirect("login.jsp");
}
}catch(Exception E){
    response.sendRedirect("login.jsp");
}
%>
</body>
</html>

```