Implementation of **S**ecure **A**ccess controlled **F**ile **E**ncryption (SAFE) system

Farrukh Shahzad
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
Email: farrukhshahzad@kfupm.edu.sa

I. INTRODUCTION

Cloud computing has evolved as a popular and universal paradigm for service oriented computing where computing infrastructure and solutions are delivered as a service. The cloud has revolutionized the way computing infrastructure is abstracted and used. Some of the features which makes cloud computing desirable includes; elasticity (the ability to scale on-demand), pay-per-use (which means no/low upfront investment and low time to market) and transfer of risk (from the small application developers to the large service providers). Therefore novel applications/ideas can be tried with minimal risks, an approach that was not feasible in the pre-cloud era. This has resulted in large numbers of applications—of various types, sizes, and requirements—being deployed across the various cloud service providers. Cloud computing not only realizes the dream of computing as a utility but provides opportunity for its adoption and growth. As with any new technology, there are challenges and obstacles. Data confidentiality and security is one of the main obstacles in adopting the cloud at the enterprise level.

The security concerns motivate the authors of [1] to propose a system, called FADE that can enforce access control and assured deletion of outsourced data on the cloud in a fine-grained manner. Access control guaranties that only authorized users can access/download the data on the cloud and assured deletion means that data is permanently inaccessible even if the storage provider (or their sub-contractors) keeps the file upon request of deletion. This work is an extension or simplification of FADE [1] to achieve more practical deployment with less overhead. The Secure Access controlled File Encryption (SAFE) system is an overlay which works seamlessly over the existing cloud storage services without any changes on the cloud side. Furthermore, the implementation only requires basic data access API functions like put (upload) and get (download).

In SAFE, a file is encrypted with a data key by the owner of the file, using the SAFE client. The data key is further encrypted with a secret key which is in turn is encrypted with a control key, based on the access control policy selected by the owner, with the help of a separate key server.

Our contributions are summarized below:

- The design of Secure Access controlled File Encryption (SAFE) system to achieve policy-based access control and assured deletion.
- The development of generic Java implementation to provide encryption/decryption and file upload/ download operations to/from any storage system.
- Implementation of complete generic SAFE client and key server applications which currently support Amazon S3 and dropbox storage services.

II. SAFE DESIGN

The SAFE overview is shown below in figure 1.

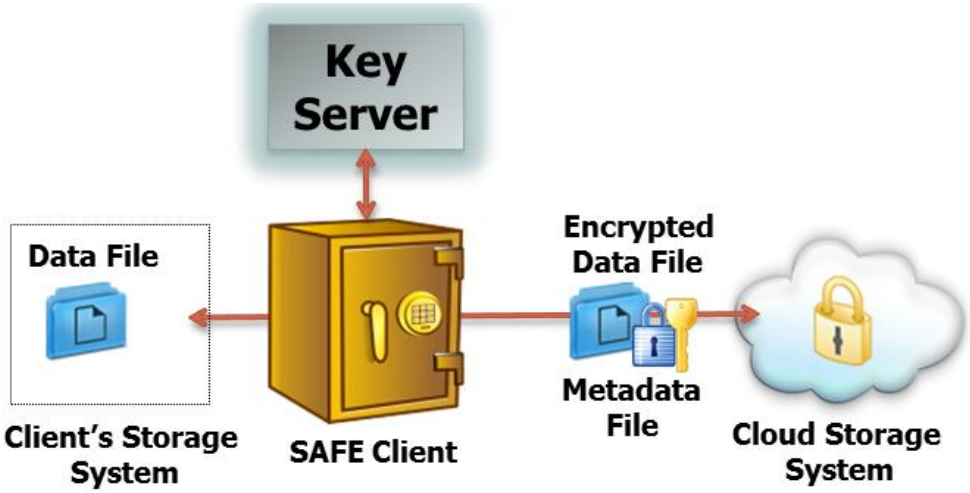

Figure 1: SAFE Overview

A. Cryptographic Keys

SAFE uses three types of cryptographic keys to protect data files stored on the cloud.

- Data key. A data key is a random secret that is generated by a SAFE client. It is used for encrypting or decrypting data files via symmetric (AES) key encryption.
- Secret key. Similar to the data key, a secret key is generated by a SAFE client. It is used for encrypting or decrypting the data key via symmetric (AES) key encryption.
- Policy key. This key is associated with a particular policy. It is represented by a public-private key pair, which is maintained by the key server. It is used to encrypt/decrypt the secret key of the file via RSA. To ensure file deletion (inaccessibility), the corresponding policy can be revoked.

Therefore, to successfully decrypt an encrypted file stored on the cloud, the correct combination of data key, secret key and policy key needs to be known; otherwise it will be computationally infeasible to access a SAFE protected file.

B. SAFE Operations

Let F represent the file which needs to be stored securely with access policy P. Each policy corresponds to unique pair of public P_{pub} and private P_{prv} keys which is generated and maintained by the key server. Assume that the encryption and decryption function is represented by $e_K(F)$ and $d_K(F)$ respectively, where K is the key used for encrypting data/file F. Similarly $e_{Ppub}(S)$ and $d_{Pprv}(S)$ represent the encryption of data S with public key P_{pub} and decryption with private key, P_{prv}, respectively.

1) File Upload

The file upload function is shown in Fig. 2. The client first requests the public key P_{pub} of policy P from the key server. Then the client generates two random keys K and S, and sends P, $e_S(K)$, $e_{Ppub}(S)$ (as metadata) and $e_K(F)$ i.e. the encrypted file to the cloud. The client should discard K and S . There will be two objects on the cloud: One the

encrypted client's file and the other the corresponding metadata text file containing policy and related keys (encrypted).

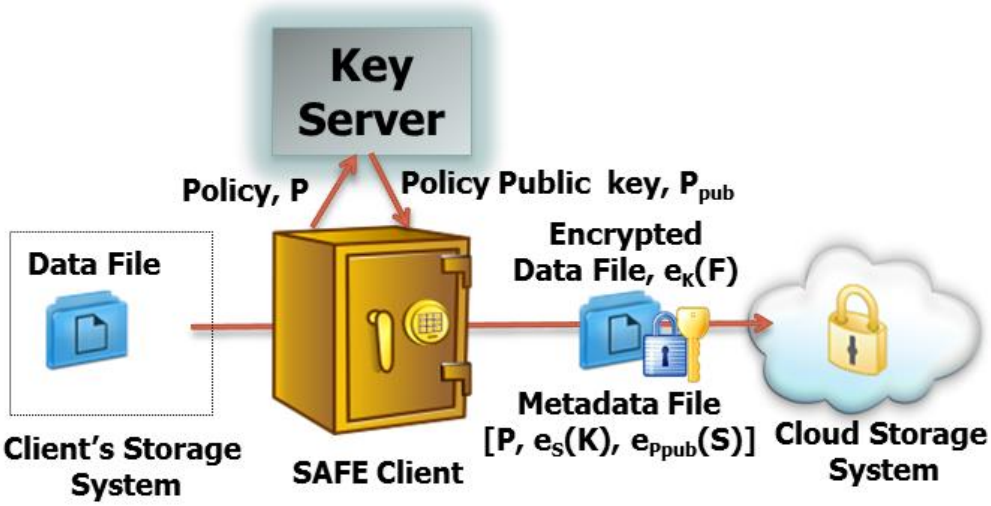

Figure 2: The File upload operation

2) File Downlaod

Fig. 3 show the file download function. The client fetches the metadata file to get P, $e_S(K)$, $e_{Ppub}(S)$ from the storage system. Then the client sends $e_{Ppub}(S)$ to the key server for decryption. The key server decrypts and returns $S = d_{Pprv}(e_{Ppub}(S))$ to the client. The client can now decrypt $e_S(K)$ to get K. The client now fetches the actual encrypted file $e_K(F)$ and decrypt with K to get the original file F.

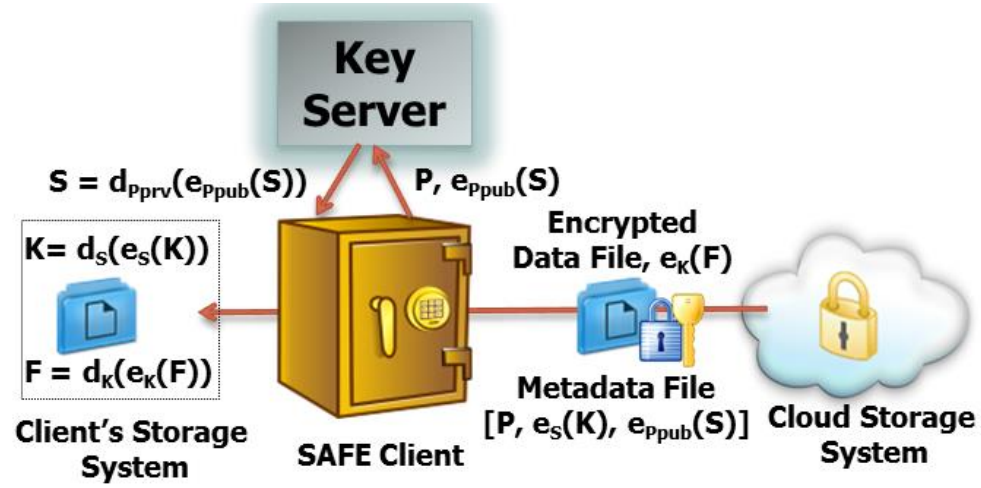

Figure 3: The File download operation

III. IMPLEMENTATION

Secure **A**ccess controlled **F**ile **E**ncryption (SAFE) is a prototype implementation of file upload and download employing AES and RSA cryptography based on FADE framework as presented in the previous section. The SAFE is implemented purely in Java. All the libraries used are third party or built in Java libraries including the following:

- javax.swing.* for GUI
- com.amazonaws.* for amazon S3 APIs
- com.dropbox.* for Dropbox APIs
- org.apache.log4j.* for interactive on-screen and file logging
- javax.crypto and javax.Security for cryptographical operations like AES/RSA encryption/decryption, Key generation, etc.
- Many other built-in libraries for File I/O, SSL socket programming. There are also other external Java libraries which are used by Amazon and Dropbox APIs.

The Java package ***edu.kfupm.ccse.fade*** is created with following files and resources:

DataSourceInterface.java	The interface class implemented by FileSafe.java
FileSafe.java	This super class provides all needed high level functions to support SAFE operations for file storage like upload/download.
S3Safe.java	This class provides all needed high level functions to support SAFE operations for Amazon S3 cloud storage like object (encrypted files and corresponding metadata) upload/download.
DropboxSafe.java	This class provides all needed high level functions to support SAFE operations for Dropbox cloud storage like object (encrypted files and corresponding metadata) upload/download.
Cryptography.java	The class which provides all high-level cryptographical operations like AES/RSA encryption/decryption, Key generation, etc.
Safe.java	The GUI interface
SafeMain.java	The main SAFE Application
KeyServer.java	The main Key Server console application
KeyServerThread.java	The Key server process thread to support multiple clients and provide all policy operations like addition, revocation, policy's public/private key generation, decryption using policy private key and user authentication.
ClientStatus.java.java	The client object/class used by Key Server application.
DBAccess.java.java	The database access utility class used by Key Server application.
EchoClient.java	The test program to test Key Server socket connectivity
SwingLogAppender.java	To allow interactive logging
HelpBrowser.java	To access web pages within the application
PatchedHTMLEditorKit.java	Support files

SimpleLinkListener.java	Support files
KeyServer.bat	The batch file to execute the Key Manager
KeyServerCredentials.properties	User needs to provide Key server IP/port, and authentication details
KeyServerDBCredentials.properties	user needs to provide DB host, username, and password
AwsCredentials.properties	User needs to provide Amazon S3 keys to access his/her account by SAFE application.
DropboxCredentials.properties	User needs to provide Dropbox keys to access his/her account by SAFE application.
Logging_gui.properties	To setup interactive logging
Changelog.txt	A text file to provide some info about the application. This file is accessed by the application to show 'About' content.
Several image files (image folder)	used by the GUI

A. Requirements:

To run the application, user needs all of the above mentioned libraries (available as freeware). User also needs to sign up for Amazon S3 or/and Dropbox cloud storage service (free for some GB of usage). The application is built on Java version 1.7 so JRE ver 1.7 is also required to run the application.

B. Safe Client

The SAFE client is GUI based application developed Java Swing framework. It implements the basic file upload/download (including corresponding metadata) to the cloud provider using FADE model as described in the paper [1]. The application connects to the Key server at initialization through SSL TCP/IP socket connection (need Key server IP address: port). User need to login to the key server. There are two types of account:

User account: This user can perform basic operations like uploading or downloading to/from cloud with policies he/she is authorized to use.

Admin Account: This user can perform admin operations like adding a new policy, revoking a policy or adding a user to a policy, apart from basic user level operations

The SAFE client is a menu driven application with three menus:

- **Cloud** Options to Upload, download, get the object list in the cloud and to exit the application.
- **Key Server** Options to connect or disconnect to/from Key server. Add policy and revoke policy is only available for admin user.
- **Help** The help and about option.

The main screen is spilt horizontally into left and right panes. The right pane is used for interactive logging for user to see underlying processing/information during application execution. The left pane show the logo on top and it consists of three tabs for home, upload and download. The upload tab shows the machine's file system for user to select the file he/she wants to upload to the cloud. The download tab show the list of existing objects(files) on the cloud for user to select the object he/she wants to download to his/her machine. The application will perform all the necessary encryption, decryption, metafile generation and any other related task transparently (with the help of Key server).

C. *Metafile*

Here is an example of a metadata file generated after an upload to the cloud:

```
SAFE0001
6B6C379A35A8A17CF005F8CE850D0F45A24C86747DB1D83E167A46ADBBF8CF03
4A31EAF4FFC824ADD69D327D551705F2CB164D23AC47D0B85E47D1BCFEBA342F7C886C3292DBDB590348FC900F210D5
6DEC21E1177A0CFC17138ACB41193AC9DEECCC74D0B72A1599026A3FD1A0BEBA1E08DA716CE7C58BA77BD79E42E1E85
033EA1F1A2B785F939F47BE421A9A2EA82005AFB81B50D628ABDA43AEFC989B788
```

This metafile is saved along with the encrypted file on the cloud with extension '.safe'.

First line is the policy name/Id(Pi). The second line is the file's AES key (K) encrypted with the user generated policy AES key (Si). The rest is the user generated Key (Si) encrypted with public key for policy Pi. Notice that K and Si are generated every time user need to upload a file.

D. *File Upload*

Here is a sample log of file upload. Figure 4 shows the GUI for upload operation.

```
14:22:36   File will be uploaded from: C:\Users\Farrukh\Documents
14:22:36   Encrypting..
14:22:36   Uploading a new object to S3 from a file in to SAFE bucket
14:22:38   Uploading the corresponding metadata object to S3 in to SAFE bucket
14:22:39   Uploaded file:  cover.docx      Done.
14:22:39   Listing objects..
```

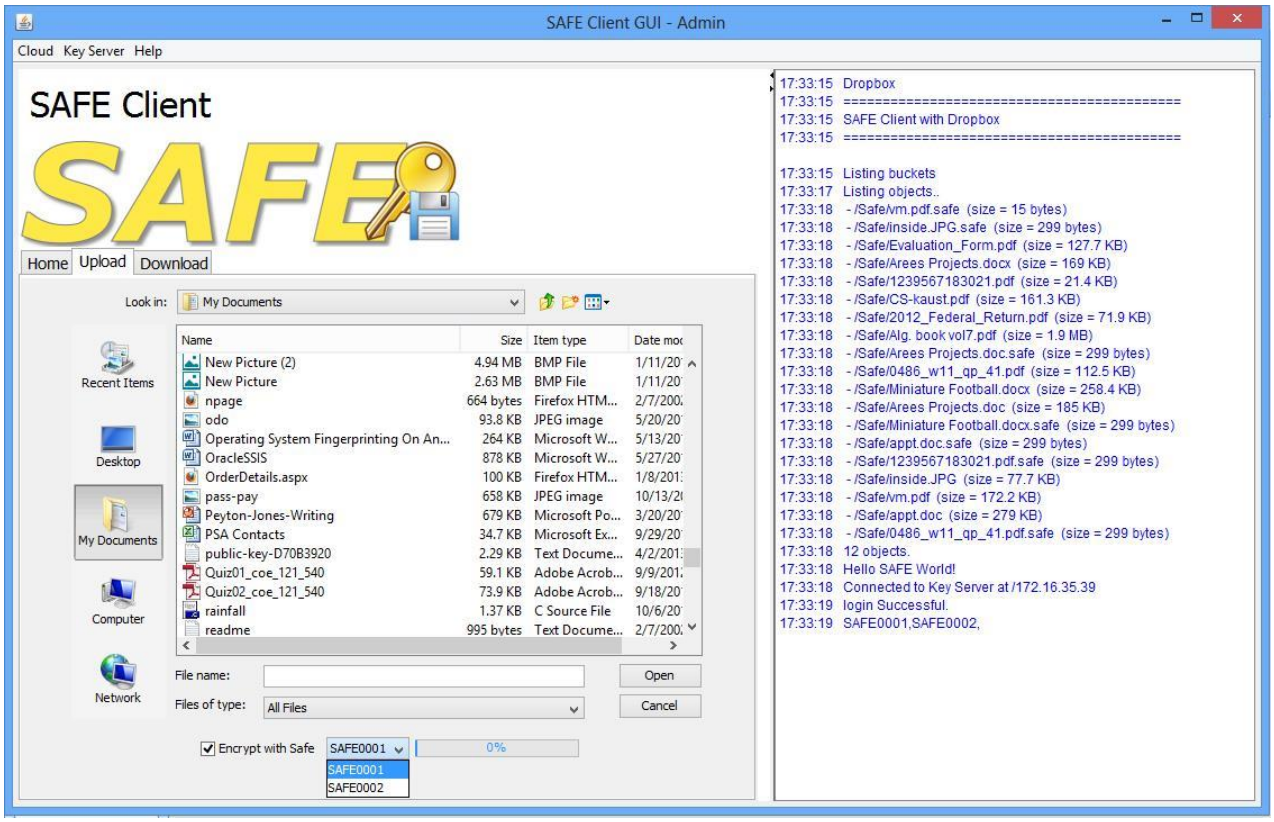

Figure 4: Upload operation

E. File Download

Here is a sample log of file download. Figure 5 shows the GUI for download operation.

```
2013-05-22 05:51:26 Downloading the object metadata.
2013-05-22 05:51:24 Downloading the object
2013-05-22 05:51:26  File Name: walterp-gridsim.pdf
2013-05-22 05:51:26  Content-Type: application/octet-stream
2013-05-22 05:51:27 Decrypting ..
2013-05-22 05:52:13 File will be saved to: C:\Users\Farrukh\Documents\walterp-gridsimu.pdf
```

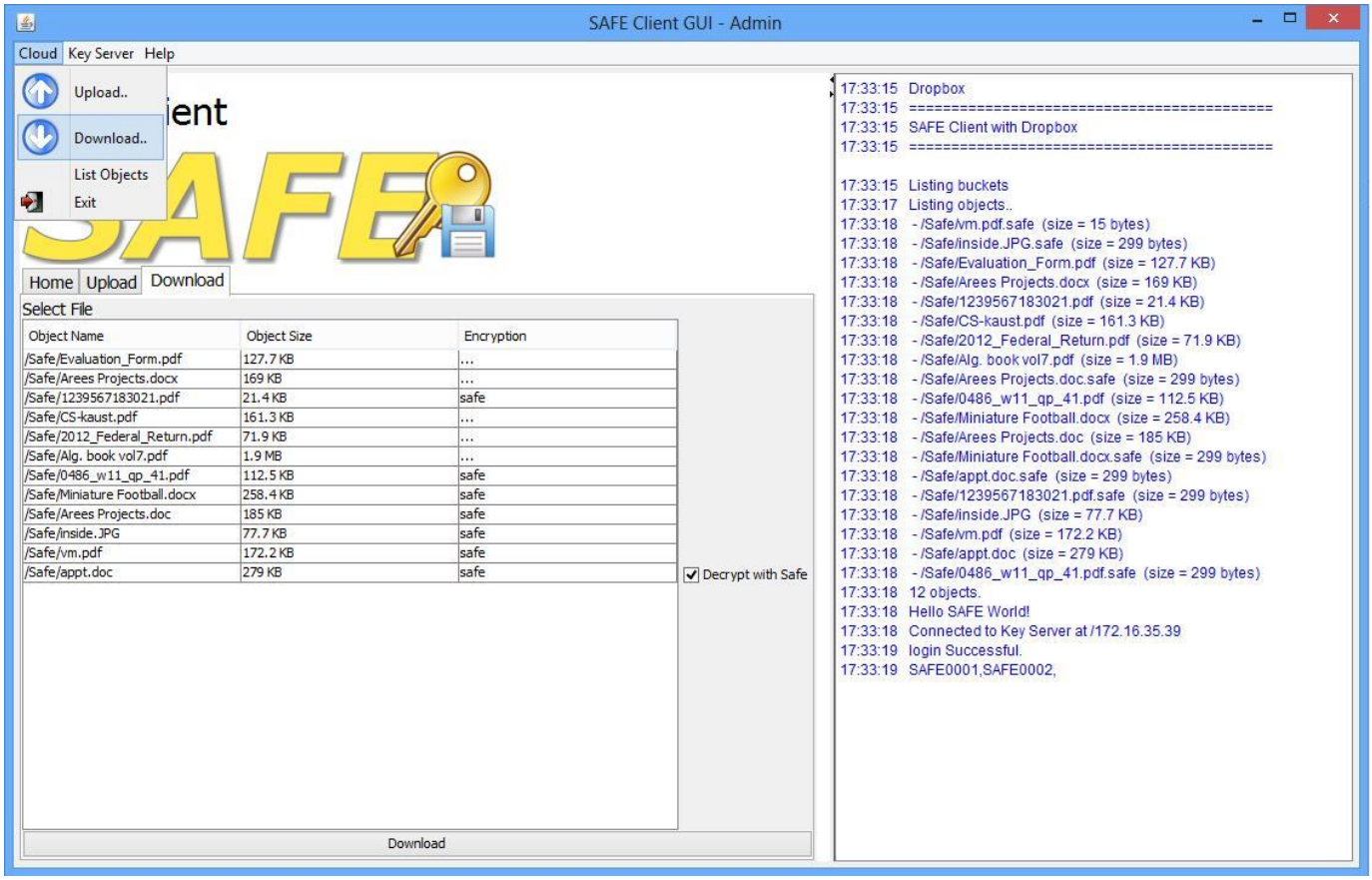

Figure 5: Download operation

F. Key Server

The Key server is a multithreaded console application written in Java. It uses the SSL TCP/IP socket to communicate to the clients (SAFE application). It listens on a certain port for clients. Multiple clients can connect at the same time. It can be run on any machine on the network as long as its IP address is known to client. It provides following services on client request:

- Policy addition (admin account)
- Policy revocation (admin account)
- Policy's public/private key generation

- Transmission of public key
- Decryption using policy private key.
- User authentication at connection initiation.

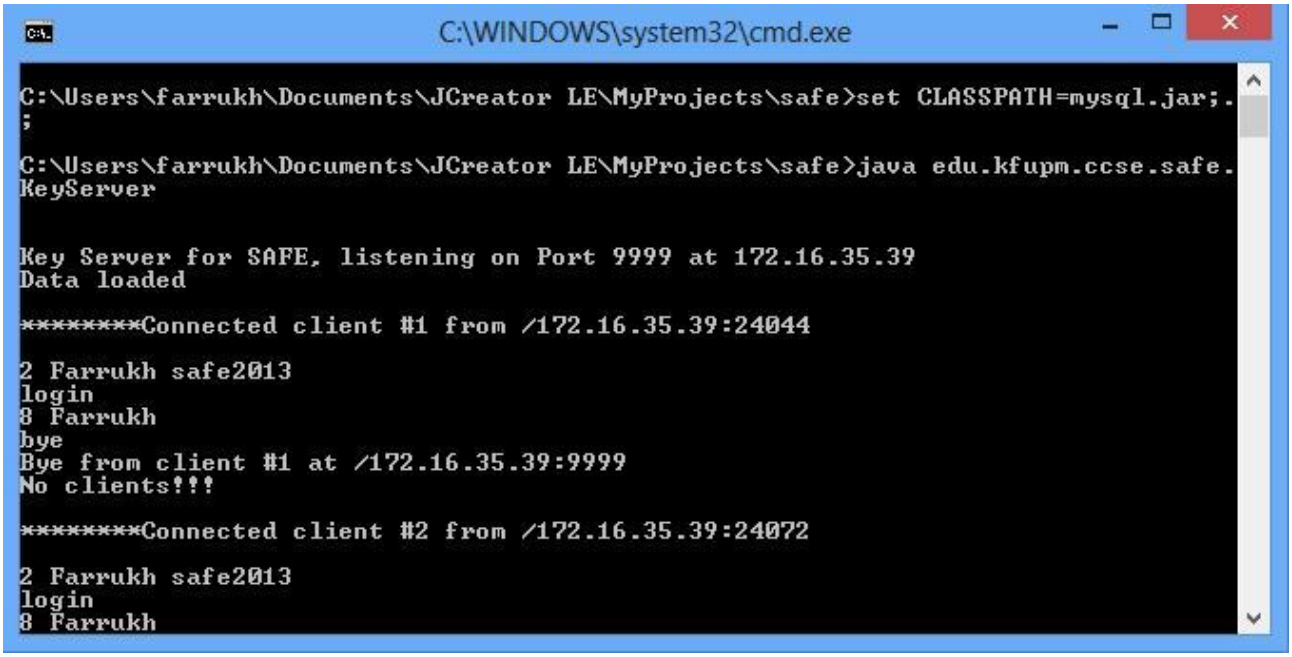

Figure 6: The Key Server application

REFERENCES

[1] Yang Tang; Lee, P.P.C.; Lui, J.C.S.; Perlman, R., "Secure Overlay Cloud Storage with Access Control and Assured Deletion," Dependable and Secure Computing, IEEE Transactions on , vol.9, no.6, pp.903,916, Nov.-Dec. 2012 doi: 10.1109/TDSC.2012.49.

[2] Amazon S3, http://aws.amazon.com/s3, 2010

[3] http://blog.fileburst.com/personal-cloud-storage-explained/

[4] R. Geambasu, T. Kohno, A. Levy, and H.M. Levy, "Vanish: Increasing Data Privacy with Self-Destructing Data," Proc. 18th Conf. USENIX Security Symp, Aug. 2009.

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," Proc. 13th ACM Conf. Computer and Comm. Security (CCS), 2006.

[6] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," Proc. IEEE Symp. Security and Privacy, May 2006.

[7] R. Perlman, "File System Design with Assured Delete," Proc. Network and Distributed System Security Symp. ISOC (NDSS), 2007.

[8] http://www.rsa.com/rsalabs/node.asp?id=2339

[9] http://www.cs.tau.ac.il/~bchor/Shamir.html

[10] http://ansrlab.cse.cuhk.edu.hk/software/fade

[11] Dropbox, http://www.dropbox.com, 2010.

[12] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," Proc. IEEE Symp. Security and Privacy, May 2006.

[13] http://ansrlab.cse.cuhk.edu.hk/software/fade/