Software Engineering- CSC 4350

Spring 2017

An encryption and decryption system for message communication

ADEPT

Amani Konduru

Benjamin Garber (**D**aniel)

Edward Bull

Paul David Utesch

Team

3/7/2017

Table of Contents

I. Rationales

1.1 Use Case Rationales

1.1.1. ADEPT Client Program Use Case Rationales

1.1.1.1 Authenticate

The Authenticate use case is necessary for any client requests to be authenticated. Without authentication, the Check Permissions use case would always return false and requests would not be allowed to succeed. This affects
RTM entry #13.

Use Case Name	Authenticate
Participating Actors:	End User, SQLite Database
Flow Of Events:	1. End User makes an authorization attempt. 2. The client queries the SQLite Database with the End User's credentials. 3. The client responds to the End User's authorization attempt.
Entry Condition:	The End User has provided authorization credentials.
Exit Condition:	The authorization attempt is either approved or denied.
Quality Requirements:	

1.1.1.2 Request Update

The Request Update use case is necessary for the client to retrieve emails from the server. This affects RTM entries #8, #10, and #15.

Use Case Name	Request Update
Participating Actors:	ADEPT Mail Server, SQLite Database
Flow Of Events:	1. The client checks its authentication. 2. The client requests and receives a mailbox update from the ADEPT Mail Server over an IMAP connection. 3. The client stores the new mailbox information in the SQLite Database.
Entry Condition:	The client will perform this action periodically.
Exit Condition:	The SQLite database responds with a success or failure of the update.
Quality Requirements:	1. Email information should be symmetrically encrypted prior to local storage using the End User's credentials as a key. 2. The connection is secured over TLS.

1.1.1.3 View Email

The View Email use case is necessary for an End User to use the client to view particular emails. This affects RTM entry #12.

Use Case Name	View Email
Partiacipating Actors:	End User, SQLite Database
Flow Of Events:	1. End User makes a request to view an email. 2. The client checks its authentication. 3. The client queries the SQLite Database for the email information. 4. The client displays the email to the End User.
Entry Condition:	The End User requests a view of an email.
Exit Condition:	The email is displayed to the End User.
Quality Requirements:	1. Email information should be symmetrically decrypted prior to viewing using the End User's credentials as a key.

1.1.1.4 Manage Emails

The Manage Emails use case is necessary for the capability to send, move, and delete emails. This affects RTM entries #11, #12, and #14.

Use Case Name	Manage Emails
Participating Actors:	End User, ADEPT Mail Server
Flow Of Events:	1. End User makes a request to send, move, or delete an email. 2. The client checks its authentication. 3. The client sends the request to ADEPT Mail Server over an SMTP connection and receives a response. 4. The client displays the status of the request to the End User.
Entry Condition:	The End User requests an email be sent, moved, or deleted.
Exit Condition:	The success or failure of the request is displayed to the End User.
Quality Requirements:	1. The connection is secured over TLS. 2. This use case is inherited by the Send Email, Move Email, and Delete Email use cases. Each use case would send a specific request for steps #1 and #3.

1.1.1.5 Manage Folders

The Manage Folders use case is necessary for the capability to create, move, or delete mailbox folders. This affects RTM entries #12 and #14.

Use Case Name	Manage Folders
Participating Actors:	End User, ADEPT Mail Server
Flow Of Events:	1. End User makes a request to create, move, or delete a mailbox folder. 2. The client checks its authentication. 3. The client sends the request to ADEPT Mail Server over an SMTP connection and receives a response. 4. The client displays the status of the request to the End User.
Entry Condition:	The End User requests that a mailbox folder be created, moved, or deleted.
Exit Condition:	The success of failure of the request is displayed to the End User.
Quality Requirements:	1. The connection is secured over TLS. 2. This use case is inherited by Create Folder, Move Folder, and Delete Folder use cases. Each use case would send a specific request for steps #1 and #3.

1.1.1.6 Check Permissions

The Check Permissions use case is required by any other use case that needs to authenticate its actions prior to completion. This affects RTM entries #9, #10, #11, #12, #14, and #15.

Use Case Name	Check Permissions
Participating Actors:	SQLite Database
Flow Of Events:	1. The client queries the SQLite Database for the credentials of the current user. 2. The client compares its current credentials with those from the database and returns the result.
Entry Condition:	The client requests an authentication check.
Exit Condition:	The client is either authenticated or not authenticated.
Quality Requirements:	1. The Check Permissions use case is included by the Request Update, Manage Emails, and Manage Folders use cases.

1.1.2. ADEPT Mail Server Use Case Rationales

1.1.2.1 Authenticate

The Authenticate use case is necessary for any client requests to be authenticated. Without authentication, the Check Permissions use case would always return false and requests would not be allowed to succeed. This affects RTM entry #1, #2, #3, #4, #5, #7, and #8.

Use Case Name	Authenticate
Participating Actors:	ADEPT Client Program, PostgreSQL Database
Flow Of Events:	1. The ADEPT Client Program makes an authentication attempt. 2. The server queries the PostgreSQL database with the credentials provided by the ADEPT Client Program. 3. The server responds to the ADEPT Client Program's request.
Entry Condition:	The ADEPT Client Program makes an authentication attempt.
Exit Condition:	The ADEPT Mail Server responds with an approval or denial of the attempt.
Quality Requirements:	1. The connection is secured over TLS.

1.1.2.2 Receive Email

The Receive Email use case is necessary for the ADEPT Mail Server to receive an incoming email from another ADEPT Mail Server.

Use Case Name	Receive Email
Participating Actors:	ADEPT Mail Server, PostgreSQL Database
Flow Of Events:	1. An external ADEPT Mail Server sends an email to the server. 2. The server queries the PostgreSQL Database to store the email information and receives a confirmation. 3. The server responds to the external ADEPT Mail Server with the status of its request.
Entry Condition:	An external ADEPT Mail Server sends an email to the server.
Exit Condition:	The server responds with the success or failure of the request.
Quality Requirements:	1. The connection is secured over TLS.

1.1.2.3 Send Email

The Send Email use case is necessary for the ADEPT Mail Server to send emails from one hosted client to another hosted client. This affects RTM entries #1, #5, #7, and #8.

Use Case Name	Send Email
Participating Actors:	ADEPT Client Program, PostgreSQL Database
Flow Of Events:	1. ADEPT Client Program sends an e-mail to the server over SMTP. 2. The ADEPT Mail Server checks the authentication of the ADEPT Client Program. 3. The ADEPT Mail Server queries the PostgreSQL database with the email information for storage and receives a response. 4. The ADEPT Mail Server responds to the ADEPT Client Program over SMTP.
Entry Condition:	ADEPT Client program sends an e-mail to the server over SMTP.
Exit Condition:	The server responds to the ADEPT Client Program over SMTP.
Quality Requirements:	1. The connection is secured over TLS.

1.1.2.4 Send External Email

The Send External Email use case is necessary for the ADEPT Mail Server to send emails from a hosted client to a client hosted on another ADEPT Mail Server. This affects RTM entries #5, #7, and #8.

Use Case Name	Send External Email
Participating Actors:	ADEPT Client Program, ADEPT Mail Server (external), PostgreSQL Database (external)
Flow Of Events:	1. ADEPT Client Program sends an e-mail to the server over SMTP. 2. The ADEPT Mail Server checks the authentication of the ADEPT Client Program. 3. The ADEPT Mail Server sends the email to the external ADEPT Mail Server over SMTP. 4. The external ADEPT Mail Server stores the e-mail in its PostgreSQL Database and responds over SMTP. 5. The ADEPT Mail Server responds to the client over SMTP.
Entry Condition:	ADEPT Client program sends an e-mail to the server over SMTP.
Exit Condition:	The server responds to the ADEPT Client Program over SMTP.
Quality Requirements:	1. All connections are secured over TLS.

1.1.2.5 Serve Update

The Serve Update use case is necessary for the ADEPT Mail Server to respond to requests for mailbox updates. This affects RTM entries #4, #7, and #8.

Use Case Name	Serve Update
Participating Actors:	ADEPT Client Program, PostgreSQL Database
Flow Of Events:	1. ADEPT Client Program requests a mailbox update over IMAP. 2. The ADEPT Mail Server checks the authentication of the ADEPT Client Program. 3. The ADEPT Mail Server queries the PostgreSQL Database for the requested mailbox information. 4. The ADEPT Mail Server responds to the ADEPT Client Program over IMAP.
Entry Condition:	ADEPT Client Program requests a mailbox update over IMAP.
Exit Condition:	ADEPT Mail Server responds with a mailbox update or failure message over IMAP.
Quality Requirements:	1. The connection is secured over TLS.

1.1.2.6 Edit Emails

The Edit Emails use case is necessary for the ADEPT Mail Server to serve requests to delete or move emails. This affects RTM entries #1, #3, #7, and #8.

Use Case Name	Edit Emails
Participating Actors:	ADEPT Client Program, PostgreSQL Database
Flow Of Events:	1. ADEPT Client Program makes a request to send, move, or delete an email over SMTP. 2. The ADEPT Mail Server checks the authentication of the ADEPT Client Program. 3. The ADEPT Mail Server queries the PostgreSQL Database to fulfill the specified request and receives a response. 4. The ADEPT Mail Server responds to the ADEPT Client Program with a success or failure message over SMTP.
Entry Condition:	ADEPT Client Program makes a request to send, move, or delete an email over SMTP.
Exit Condition:	The ADEPT Mail Server responds to the ADEPT Client Program with a success or failure message over SMTP.
Quality Requirements:	1. The connection is secured over TLS. 2. This use case is inherited by the Delete Email and Move Email use cases. Each use case would send a specific request for steps #1 and #3.

1.1.2.7 Edit Folders

The Edit Folder use case is necessary for the ADEPT Mail Server to serve requests to create, delete, or move folders. This affects RTM entries #2, #7, #8, and #8.

Use Case Name	Edit Folders
Participating Actors:	ADEPT Client Program, PostgreSQL Database
Flow Of Events:	1. ADEPT Client Program makes a request to create, move, or delete a mailbox folder over SMTP. 2. The ADEPT Mail Server checks the authentication of the ADEPT Client Program. 3. The ADEPT Mail Server queries the PostgreSQL Database to fulfill the specified request and receives a response. 4. The ADEPT Mail Server responds to the ADEPT Client Program with a success or failure message over SMTP.
Entry Condition:	ADEPT Client Program makes a request to create, move, or delete a mailbox folder over SMTP.
Exit Condition:	The ADEPT Mail Server responds to the ADEPT Client Program with a success or failure message over SMTP.
Quality Requirements:	1. The connection is secured over TLS. 2. This use case is inherited by the Create Folder, Move Folder and Delete Folder use cases. Each use case would send a specific request for steps #1 and #3.

1.1.2.8 Check Permissions

Use Case Name	Check Permissions
Participating Actors:	PostgreSQL Database
Flow Of Events:	1. The server queries the PostgreSQL Database for the credentials of the current user. 2. The server compares the current credentials with those from the database and returns the result.
Entry Condition:	The server requests an authentication check.
Exit Condition:	The current transaction is either authenticated or not authenticated.
Quality Requirements:	The Check Permissions use case is included by the Send Email, Server Updates, Edit Emails, and Edit Folders use cases.

1.2 Database Rationale

We will use two databases: PostgreSQL and SQLite. The metadata will be used to store the encrypted message in a simple PostgreSQL database, with the owner as a primary key. The client program will have some basic login functionality and a SQLite database will be used to store usernames and a hashed/salted password as well as the user's local emails and mailbox structure. Both are open source and are best for our project from the limited options that are available. PostgreSQL is good for concurrency and will be used for our server to support concurrent connections and operations. SQLite is easy to set up and work with, so it's an ideal solution for our client program.

We will be testing our server database on a Digital Ocean virtual machine "droplet." While the system is not sufficiently robust to handle large scale operations like you might see with email servers in the real world, it is sufficient for a class project and has a publicly accessible IP. More importantly for our situation, Digital Ocean droplets are inexpensive, reliable, and can be snapshotted for easy rollbacks should something go wrong during development.

Our Digital Ocean instance will be running on Ubuntu for easy terminal access over SSH. This also allows us to use a variety of networking tools on the command line that are native to Linux systems. For development, we will be using GitHub for version control and primarily Eclipse as an IDE.

1.3 Class Diagram Rationale

1.3.1 Server Rationale

1. **Server Controller:** The ServerController object configures the server on startup and spawns the SmtpServer and ImapServer objects.
2. **SSLServer:** The SSLServer abstract class is the foundation for the SmtpServer and ImapServer objects. An SSLServer listens for traffic over a secured SSL socket.
3. **SmtpServer:** The SmtpServer object listens for and accepts SmtpConnection objects.
4. **ImapServer:** The ImapServer object listens for and accepts ImapConnection objects.
5. **Connection:** The Connection abstract class is the foundation for the SmtpConnection and ImapConnection objects. A Connection represents an incoming request over the listening socket, parsed into a byte array.
6. **SmtpConnection:** The SmtpConnection objects input requests into the system and pass those requests to SmtpCmdProc objects.
7. **ImapConnection:** The ImapConnection objects input requests into the system and pass those requests to the ImapCmdProc objects.
8. **CmdProc:** The CmdProc abstract class forms the foundation of the SmtpCmdProc and ImapCmdProc objects. It will process requests and then execute the appropriate action based on the format of the request.
9. **SmtpCmdProc:** The SmtpCmdProc objects parse requests according to the SMTP protocol.
10. **ImapCmdProc:** The ImapCmdProc objects parse requests according to the IMAP protocol.
11. **QueryHandler:** The QueryHandler class constructs queries and receives responses from the PostgreSQL database attached to the server.
12. **ServeUpdate:** The ServerUpdate class executes an IMAP request for an update.
13. **DeleteFolder, MoveFolder, CreateFolder, DeleteEmail, MoveEmail, ReceiveEmail, and SendEmail** classes all execute their respective action by calling the appropriate method in QueryHandler.
14. **SendExternalEmail:** The SendExternalEmail class extends the SendEmail class in the situation that an email destination address is not local. This class sends an email to the destination server by invoking the SmtpClient class and passing the request on.
15. **SmtpClient:** The SmtpClient object connects to an external socket and sends an email in an SMTP formatted request.

1.3.2 Client Rationale

1. **Client:** The Client class starts the program and receives input from ClientUI and translates that input into actions based on its various connected classes.
2. **Authenticate:** The Authenticate class receives credentials from Client and checks those credentials against a query to the SQLiteInterface class.
3. **SQLiteInterface:** The SQLiteInterface class processes and sends queries to the local SQLite database and responds with data as its result
4. **RequestUpdate:** The RequestUpdate class is periodically called by the Client class. It requests updates from a spawned ImapConnection object and then updates the local database with that data by invoking the SQLiteInterface class.
5. **Connection:** The Connection abstract class is the foundation for the ImapConnection and SmtpConnection objects. It opens a connection to an external listening server over the relevant protocol.
6. **ImapConnection:** Used to connect to the server to request updates for the RequestUpdate class.
7. **SmtpConnection:** Used to connect to the server to make requests for the EditMail class.
8. **EditMail:** The EditMail class takes actions from the Client class and calls the appropriate class to execute those actions, then responds with a status to be displayed.
9. **SendEmail, MoveEmail, DeleteEmail, CreateFolder, MoveFolder, DeleteFolder** classes all execute their respective actions by forming and then sending a request to the SmtpConnection class. The response is then sent back to Client.

II. Requirements Traceability Matrix (RTM)

Entry #	System Specification text	Type	Build	Use Case Name	Category
1	The Adept Mail Server shall store user e-mails in a database.	SW	1	Send Email, Edit Emails	Server
2	The Adept Mail Server shall move user e-mails between user-designated mailboxes upon an authenticated request from that user.	SW	2	Edit Folders	Server
3	The Adept Mail Server shall delete user-designated e-mails from its database upon an authenticated request from that user.	SW	2	Delete Emails	Server
4	The Adept Mail Server shall serve user data when authenticated requests are received from the Adept Mail Client via a minimally compliant IMAP protocol.	SW	1	Serve Updates	Server
5	The Adept Mail Server shall send user emails from other Adept Mail Servers upon an authenticated request from that user via a minimally compliant SMTP protocol.	SW	2	Send Email, Send External Email	Server
6	The Adept Mail Server shall receive user emails from other Adept Mail Servers via a minimally compliant SMTP protocol.	SW	2	Receive Email	Server
7	The Adept Mail Server shall encrypt all incoming and outgoing connections using the TLS 1.2 standard.	SW	1	Receive Email, Send External Email, Serve Updates, Edit Emails, Edit Folders, Authenticate	Server
8	The Adept Mail Server shall support multiple concurrent connections.		1	Receive Email, Serve Updates, Edit Emails, Edit folders	Server
9	The Adept Mail Client shall request user email data from the Adept Mail Server via a minimally compliant IMAP protocol.	SW	1	Request Update	Client
10	The Adept Mail Client shall store user email data locally in a local database.	SW	1	Request Update	Client
11	The Adept Mail Client shall send user	SW	1	Send Email	Client

	emails to the Adept Mail Server via a minimally compliant SMTP protocol.				
12	The Adept Mail Client shall provide a graphical user interface to allow users to generate requests and view their emails.	SW	2	Authenticate, view Email, Manage Emails, Manage Folders	Client
13	The Adept Mail Client shall require local authentication from any user before executing local requests.	SW	1	Authenticate	Client
14	The Adept Mail Client shall provide remote authentication to the Adept Mail Server prior to executing any requests.	SW	1	Manage Emails, Manage Folders	Client
15	The Adept Mail Client shall locally encrypt and decrypt the subject and body of every email it sends and receives, respectively, using symmetric-key block encryption based on a user provided password.	SW	2	Request Update	Client

III. WSD

Name	**Role**
Amani Konduru	Project manager and tester, Document handler, Java Programmer, Database Structure (PostgresSQL schemas)
Benjamin Garber (Daniel)	Code the GUI that utilizes Ed's code base and document the code for JavaDocs
Edward Bull	Code the Server and Client prototypes into classes that can will be used in the GUI program and document with JavaDocs
Paul David Utesch	Create testing scenarios for testing the GUI functionality and J-Unit

IV. Gantt Chart

Create V1 Stories	**S-01001**	**Ed Bull,akonduru**	**Done**	**1**
Document 2 - Requirements Elicitation	S-01017	akonduru	Done	1
Learn how to use VersionOne	S-01022	Ed Bull,bgarber,akonduru	Done	1
Set up group Github	S-01018	Ed Bull	Done	1
Plan Server Database And Write CreateTables SQL Script	S-01019	Putesch,akonduru	Future	1
Set up PostgresSQL test server	S-01009	akonduru	In Progress	1
Implement Test Database	S-01020	akonduru	Future	1
Server ServerController	S-01002	Ed Bull	In Progress	1
Server SmtpServer	S-01003	Ed Bull	In Progress	1
Server ImapServer	S-01004	Ed Bull	In Progress	1
Server SmtpConnection	S-01005	Ed Bull	In Progress	1
Server ImapConnection	S-01006	Ed Bull	In Progress	1
Server CmdProcessor	S-01007	bgarber	In Progress	1
Server QueryGenerator	S-01008		Future	1
Server SmtpClient	S-01021		Future	1
Document 2 Title Page	S-01037	akonduru	Done	1
Document 2 Problem Statements	S-01038	Ed Bull	Done	1
Document 2 RTM	S-01039	Ed Bull	Done	1
Document 2 WSD	S-01040	akonduru	Done	1
Document 2 Gantt	S-01041	akonduru	Done	1
Document 2 Dictionary	S-01042	Ed Bull,bgarber	Done	1
Set up PostgresSQL test server	S-01043	akonduru	In Progress	2
Implement Test Database	S-01044	akonduru	Future	2
Server ServerController	S-01045	Ed Bull	In Progress	2
Server SmtpServer	S-01046	Ed Bull	In Progress	2
Server ImapServer	S-01047	Ed Bull	In Progress	2

Server SmtpConnection	S-01048	Ed Bull	In Progress	2
Server ImapConnection	S-01049	Ed Bull	In Progress	2
Server CmdProcessor	S-01050	bgarber	In Progress	2
Server QueryGenerator	S-01051		Future	2
Server SmtpClient	S-01052		Future	2
Collate Document 3	S-01024	akonduru	In Progress	2
Document 3 Title Page	S-01025	akonduru	Accepted	2
Document 3 RTM (5 columns)	S-01027	Ed Bull	Accepted	2
Document 3 Use Cases and Int. Diagrams	S-01028	Ed Bull	Done	2
Document 3 Function Point Analysis	S-01029	bgarber	Done	2
Document 3 Database To Be Used	S-01030	akonduru	Future	2
Document 3 Updated WSD	S-01031	akonduru	Done	2
Document 3 Updated Gantt Chart	S-01032	Ed Bull	In Progress	2
Document 3 Dictionary	S-01033	Ed Bull	Accepted	2
Document 3 Use Cases Rationale	S-01034	Ed Bull	Done	2
Document 3 horizontal prototype	S-01035	Putesch	In Progress	2
Doc4 #2	S-01063	akonduru	In Progress	2
Doc4 #4	S-01064	Putesch	Done	2
Doc4 #5	S-01065	Ed Bull	In Progress	2
Doc4 #9	S-01066	EdBull,Putesch,akonduru	In Progress	2
Doc5	S-01067	Ed Bull,akonduru	In Progress	2

V. Glossary

1. **Encryption:** the process of converting data into a code, to prevent unauthorized access. Encryption is the process of transforming data into an unreadable, encrypted form. The transformation is done using one of several cryptographic algorithms that leverage computationally difficult mathematical problems to make reversing the transformation difficult if not effectively impossible

2. **Symmetric Encryption**: Symmetric Encryption uses a key or set of keys to both encrypt and decrypt data. If data is to be shared between two parties, they must both have the key or keys to decrypt or encrypt the data.

3. **Asymmetric Encryption:** Asymmetric Encryption, also known as Public Key Encryption, is a type of encryption where anyone in possession of a public key can encrypt a message. That message can then only be decrypted with a private key. This method is often used for identity authentication because it is computationally expensive. Once authentication is completed, communications will then often transition into symmetric encryption after generating a symmetric encryption key.

4. **End to End Encryptio**n: Only the communicating users can read the messages.

5. **SSL/TLS (Secure Sockets Layer / Transport Layer Protocol):** TLS and the now- deprecated SSL it is based on are network security protocols meant to secure client-server connections using both symmetric encryption for data transfer and asymmetric encryption for identity authentication. While there are many options that can be set in an SSL/TLS session, the foundation of the protocols lie in using encryption to authenticate the identities of the connected parties and to secure the privacy of the data transferred between them.

6. **Server:** a server program awaits and fulfills requests from client programs, which may be running in the same or different computers.

7. **Client:** requesting program or user.

8. **Socket:** Is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

9. **SMTP protocol:** Simple Mail Transfer Protocol. It is an Internet standard for electronic mail (email) transmission. SMTP was first defined by RFC 821 and updated in RFC 5321.

10. **IMAP protocol:** Internet Message Access Protocol. Itis an Internet standard protocol used by e-mail clients to retrieve e-mail messages from a mail server over a TCP/IP connection. IMAP is defined by RFC 3501.

11. **TCP/IP :** IP (Internet Protocol) is the basic communication language or protocol of the ozInternet. It can also be used as a communications protocol in a private network (either an intranet or an extranet). TCP (Transmission Control Protocol) is layered on top of IP to provide certain network control and data validation features for many internet communications.