PERSONAL ENCRYPTED TALK - SECURING INSTANT MESSAGING WITH A JAVA APPLICATION

David Snogles*
M.S. in Computer Science, Rivier College 2005

Keywords: *Java Encryption Instant Messaging*

Abstract

Most users of mainstream Instant Messaging applications on the Internet don't realize their conversations are being transmitted in clear text and are vulnerable to eavesdropping during transmission. This project report presents a solution to this problem implemented in a final project for CS699 in the spring of 2005 at Rivier College. The project was entitled Personal Encrypted Talk and its primary goal was to secure Instant Messaging communications between two parties on the Internet. Secondary objectives were Java Cryptography Architecture research and the practical experience gained by the author in the development of a scalable Java based Graphical User Interface application. This article summarizes the software engineering steps followed during the implementation of this project.

1 Introduction

In the 1970s UNIX became a popular networked operating system used by programmers. Many of these programmers found it convenient to continue working while simultaneously communicating with fellow programmers using a program called "talk." Talk allowed two users to communicate via typing text in a split window displayed in a local terminal window.

In the 1990s the next generation of UNIX talk was born. It was called Instant Messaging, and it wasn't confined to just UNIX workstations. It was available to any PC connected to the Internet. As its popularity increased major Internet Service Providers such as MSN® and third party software vendors like ICQ® developed their own Instant Messaging services. These services are now used by millions of users around the world.

Most Instant Messaging Services are plagued by the same problem. All the communications between the parties are transmitted in clear text. Anyone with a traffic sniffer on the network or at any hop in between the two end stations can eavesdrop on the conversation. (See the MSN Sniffer and ICQ Sniffer products available from http://www.packet-sniffer.net/ for more details). Another problem is that all the clear text traffic goes through the messaging servers of these companies; therefore at any time they could choose to log all conversations between individuals. This can affect both corporate and personal users alike. Company sales opportunities, intellectual property secrets, information on company projects, marketing information, as well as office gossip between employees can put a company at risk. Personal information between home users could compromise usernames, passwords, credit card numbers, and social security numbers putting the average citizen at risk for identity theft.

Securing network communications is a practical problem facing all interconnected applications. It's not limited to Instant Messaging, but also affects E-mail, FTP, Telnet, and Virtual Desktop applications. The Personal Encrypted Talk (PET) project summarized in this article was implemented as

a final project in CS699 at Rivier College in the spring of 2005 and provides a solution for Internet users still communicating with clear text. The PET system uses open source cryptography technologies to allow end users to form a secure tunnel between two PC's anywhere on the Internet. All communications through this tunnel are encrypted and protected from outside eavesdropping. The PET project is an example of how insecure applications can be made secure using the Java Cryptography Architecture.

2 System Environment

The Personal Encrypted Talk (PET) system was designed to operate on Microsoft® Windows XP and Windows 2000 machines with an active connection to the Internet. The following table (see Fig. 1) illustrates details on the software tools and libraries used in the development of this project.

Development Language:	Java 1.4.2 & 1.5 from Sun Microsystems
Interactive Development Environment:	Borland JBuilder X Personal Edition
Servlet, Applet or Application:	Java Application
User Interface:	Java Frame built using the Swing Library
Communication Transport Mechanism:	Apache XML-RPC.
Cryptography Platform:	Using the Java Cryptography Architecture (JCA), and Java Cryptography Extension (JCE)
Cryptography Ciphers:	Several cipher suites provided by Bouncy Castle and Cryptix were analyzed. The Diffie-Hellman and 3DES implementations provided by Sun in the JCA were used for version 1.0 of PET Project.
UML Tools:	Microsoft Visio with UML Template, Jude UML Modeling Tool (Jude Community V1.4.3) by EIWA System Management.
PC Install Software:	NSIS (Nullsoft Scriptable Install System) V2.06 from: http://nsis.sourceforge.net/
Scheduling Software:	GanttProject 1.9.10

Figure 1: Software Technologies Utilized

3 System analysis issues

This section highlights several of the stages of analysis the PET project went through before the initial coding of the system could begin. The design requirements were developed highlighting what functionality the system was to contain and how the end user would interact with the system. Additional steps in the analysis phase of the project included partitioning the system into Client and Server components, and outlining the interaction the users would have with the system. Each user of the system is called a PetUser and they interact with a PetClient graphical user interface (GUI) to initiate chat sessions with fellow PetUsers. The Server portion of the system is called the PetServer, which contains a list of PetUsers who are logged into the system and are available for chat.

3.1 Design Requirements

The Personal Encrypted Talk (PET) application is designed to allow multiple users to communicate via a secure Instant Messaging tool over the Internet.

- The PetUsers should be able to log into a remote PetServer and find peers to communicate with.
- PetUsers should not be required to have any cryptographic expertise to use the system.
- The PetUsers should be able to initiate a conversation, and all encryption should be done behind the scenes.
- To do encryption behind the scenes, a key exchange algorithm like Diffie-Hellman should be used to create shared private keys for each conversation.
- For the initial Version 1.0 release of the tool, a single encryption Cipher may be used, but hooks for the future support of multiple Ciphers should be implemented.
- The single Symmetric Cipher provided in Version 1.0 should be either 3DES (Triple-DES) or AES (Advanced Encryption Standard).
- When a new user logs into or out of the PetServer, all PET applications should be informed of the updated user (friend) list.
- The user will be required to know the IP Address and Port of the PetServer they will be logging into, but the PET Chat Application should automatically find an open port on the system and utilize that for operations, (i.e., end users will not have to pick a port number manually).
- If a PetUser attempts to talk to a user that has logged out of the PetServer before their local update list is refreshed, the system should not allow a conversation to begin.
- After a user finishes a conversation with a peer and the chat window is closed, the remote peer's chat window should be informed.
- Any network related instabilities in the communications link should be handled gracefully by all components in the system. Minimal user interaction should be required while the system is recovering from exceptions, but the user should be informed that a communication failure has occurred.
- The PetUser should be able to carry on at least one conversation with each user logged into the PetServer. The PET application will not be required to loop back conversations to itself, and if the application does not support this feature it should be prevented.
- For Version 1.0, a console based PetServer application will be required. Status of users logging into and out of the server should be updated to the console in real time. If time permits a GUI window could be provided.
- For Version 1.0 a GUI window should be provided in the PetClient to allow the user to:
    - Log into and out of the PetServer.
    - Get an updated active user list manually via a button click. (This feature is helpful if the user thinks the auto-update refresh is too slow).
    - The ability to initiate a chat (talk) session with a user selected in the active user list.

3.1.1 Requirements for Future Releases:

Future versions of the PET application should provide support for the following features:

- AES support was not provided in the initial release and it should be supported in future releases.

(Note: Version 1.0 supports 3DES).

- Message Digests and Message Authentication should be supported in future releases. This will ensure the integrity of text messages transmitted through the system as well as verify the authenticity of the sender.
- PetUsers were not required to provide a password to log into the PetServer (Version 1.0). Enhancements to the PetServer may include the requirement to log in with an encrypted password.
- A Graphical User Interface (GUI) should be provided for the PetServer. It will provide a simple interface for adding and removing PetUsers from the system. Any usernames and passwords for PetUsers will be stored in a password protected encrypted file on the local disk.
- Add support for PetServer message queuing. With PetServer message queuing, the PetServer will queue encrypted messages of conversations between PetUsers. This will allow the PetUsers to log into the server and retrieve messages on a regular basis. Support for this model will allow users behind a corporate firewall to act as Client Only applications where all messages are posted and received with the PetServer. Since a server isn't running on the client machine, peer PET clients do not require ports to be opened in the firewall.
- Add support for logging a conversation to a file.

3.2 The PET System's Network Layout

The PET System is a Client-Server, Client-Client (Peer-to-Peer) application. As the following diagram illustrates (see Fig. 2), all communication between Java Applications happen over XML-RPC. XML-RPC is a Remote Procedure Call library obtained from Apache. Details on XML-RPC can be found in reference [1]. PetUsers are able to log into and out of the PetServer via XML-RPC function calls. When a client initiates a conversation, it contacts the PetServer to check to see the user is still actively logged in, and get the IP address and port number of the peer it wishes to communicate with. After this information is obtained, the chat session between the two peers is a client-to-client conversation and the PetServer is no longer involved.

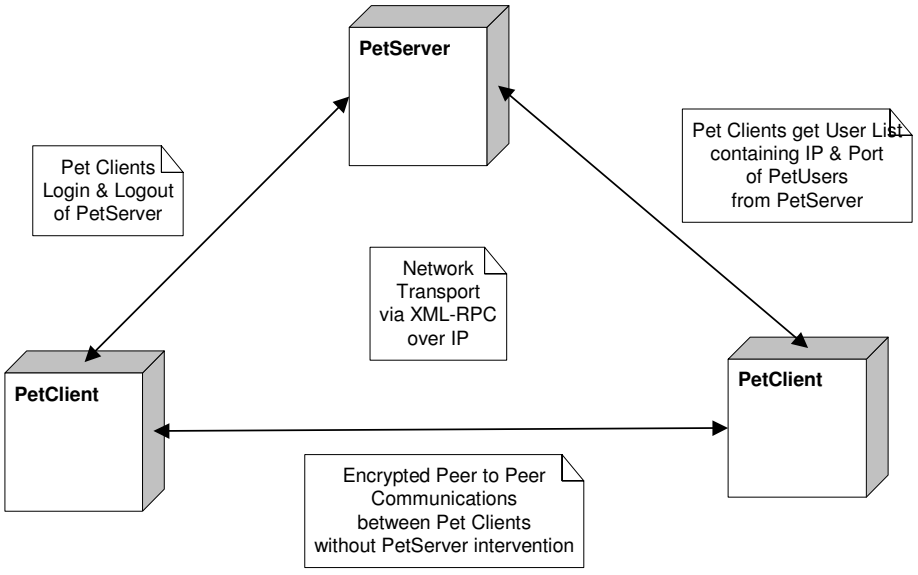

Figure 2: PET System Network Layout

3.3 State Chart for the PET System User Operations

The following state chart (Fig. 3) provides more detail into the interaction of multiple users with the PET application, and what's going on behind the scenes within the PET system. Notice the exchange of public keys to agree on a shared secret key. This exchange is an asymmetric key agreement protocol called Diffie-Hellman. It sets up the shared secret between the two PetUsers, which is used to create the shared 3DES secret key. Cryptographic research utilized references [2] & [3] and more details on the JCA and JCE are available from those sources.

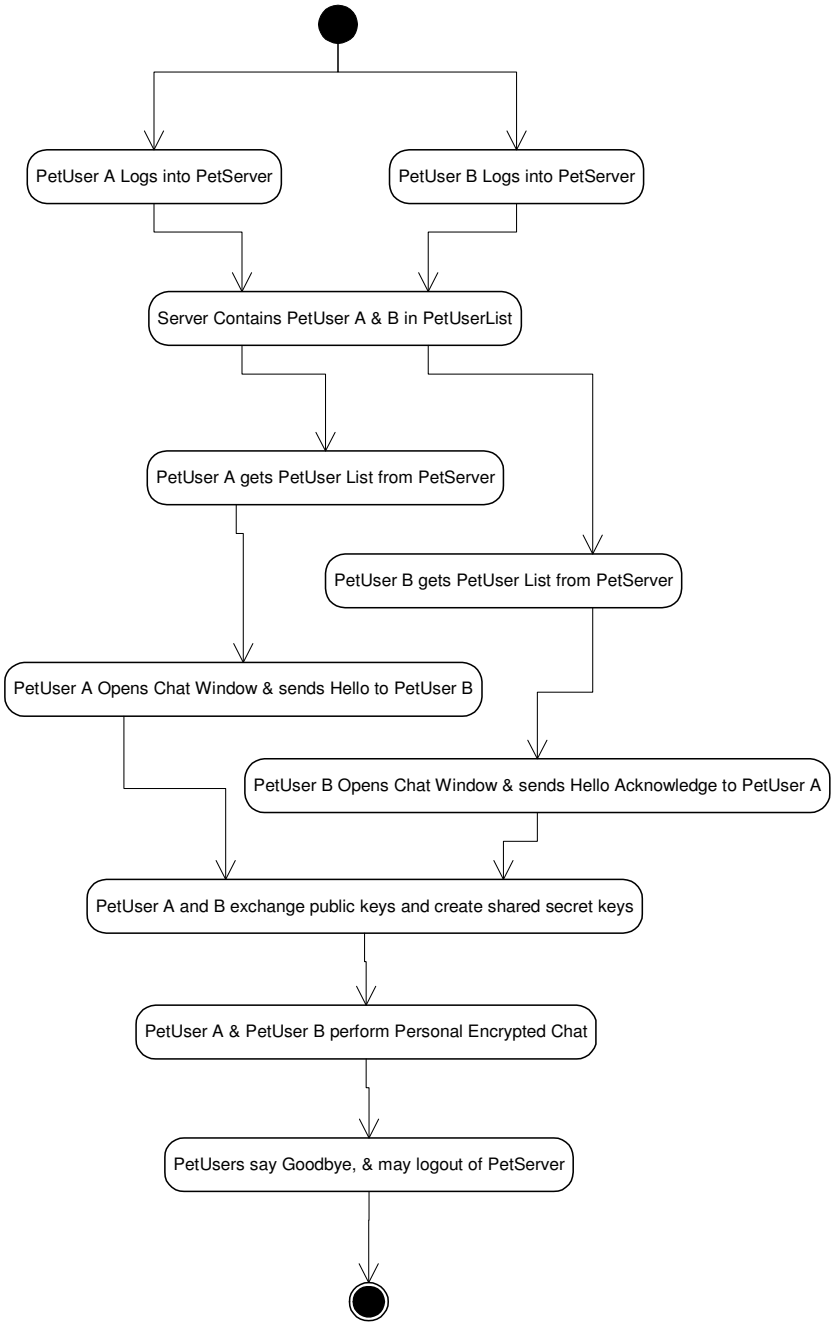

Figure 3: State Chart for PET System User Operations

4 System design issues

This section highlights several of the stages of development the PET project went through before the initial prototype was ready for a system demonstration. Included in this section is an example of a sequence diagram, and examples of several UML (Unified Modeling Language) class diagrams. The sequence diagrams together with several state machine diagrams and CRC cards (CRC: Classes, Responsibilities, and Collaborators) were used to formalize eleven Java objects in the final design. These Java objects were specified in UML prior to writing any Java code for the system.

During development Unit, Integration, and System Level test plans were developed. As blocks were finished the test plan was executed and the system debugged until all the tests in the System Level test plan passed and the features in the functional specification were operational.

Reference [4] was heavily utilized for the UML system analysis and design processes followed in the development of this project. Java language references utilized during the programming of the system included [5], [6], and [7].

4.1 PET System Sequence Diagram

The following Figure 4 contains the initial sequence diagram developed during the software architecture phase of the design. Sequence diagrams like this were used to solidify the interaction of the main components of the system and the end users of the system.

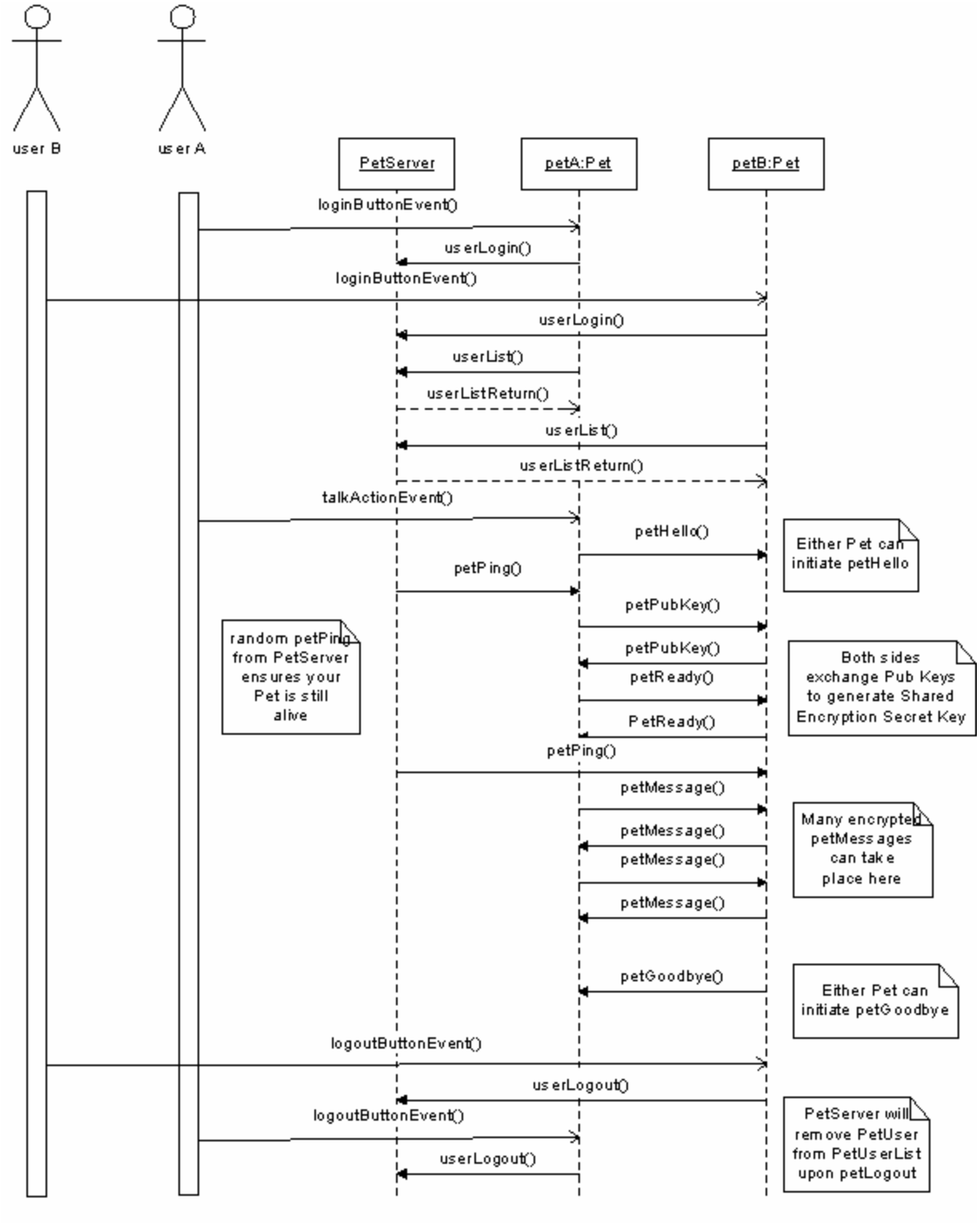

Figure 4: PET System Sequence Diagram

4.2 Examples of UML (Unified Modeling Language) Class Diagrams

The PetServer and PetClient applications are contained in the com.snogles.pet Java package. This package contains eleven Java objects implementing the PET system. UML diagrams were developed for all eleven classes and integrated into a system level UML class diagram. This section contains two example classes, PetCipherEngine and PetChatFrame, from the final project report.

4.2.1 UML Diagram for the PetCipherEngine

The PetCipherEngine is the heart of the Java encryption features implemented in the project (see Fig. 5). The arrows pointing away from the PetCipherEngine illustrate other Java objects this class encapsulates and interacts with. The lower half of the PetCipherEngine diagram lists functions this class contains. Two functions of interest are the encrypt( ) and decrypt( ) methods. These are used by the PetChatFrame while communicating with its peer PetChatFrame during an encrypted conversation.

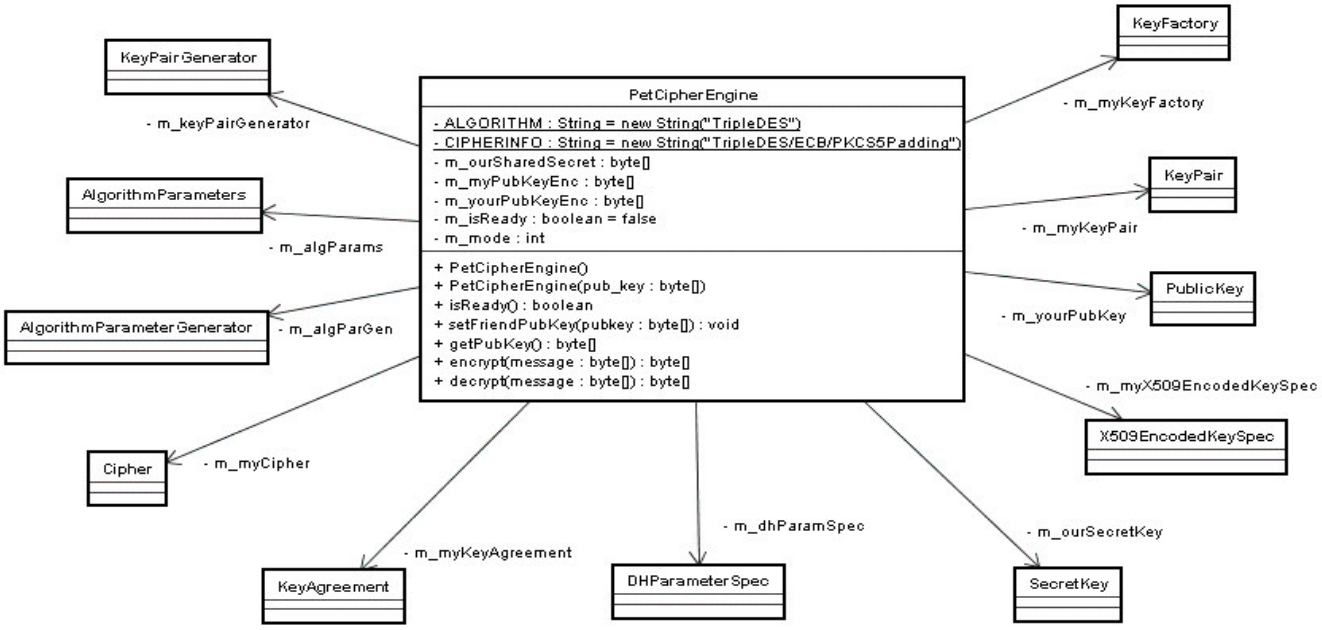

Figure 5: UML Class Diagram for PetCipherEngine

4.2.2 UML Diagram for the PetChatFrame

The PetChatFrame (see Fig. 6) contains the GUI (Graphical User Interface) for the chat window used when two PetUsers communicate in the PET system. Notice it contains an instance of the PetCipherEngine used for encrypting and decrypting messages as well as the JTextArea's used for displaying and typing messages.

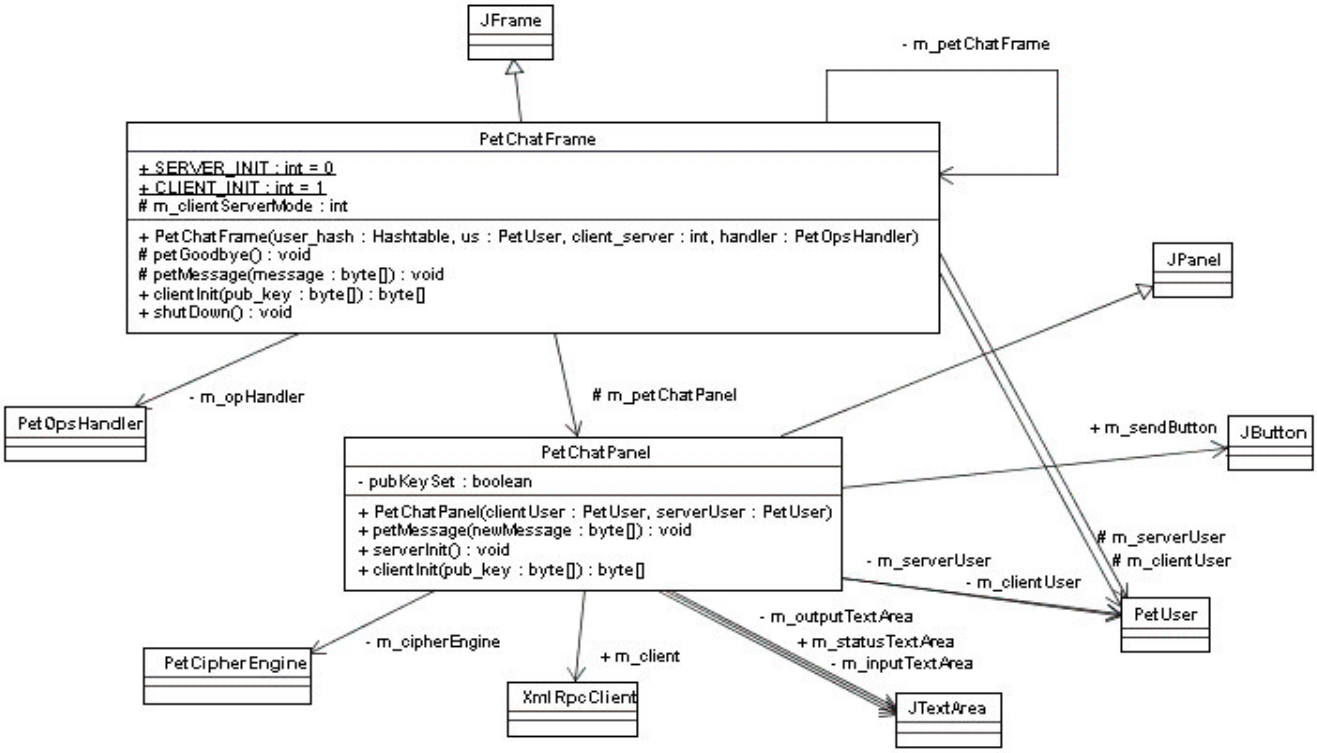

Figure 6: UML Class Diagram for PetChatFrame

5 Implementation issues

The implementation issues overcome in this project all revolved around the use of Java encryption in the System. Before encryption technology could be used in the system, research into the field of cryptography and into the Java Cryptography Architecture (JCA) and Java Cryptography Extension (JCE) in particular were required. Several weeks of experimentation using the JCA and JCE led to the final implementation of the PetCipherEngine Java class in the project. The symmetric cipher chosen for Version 1.0 of PET system was 3DES (aka: Triple-DES). The asymmetric key (public key) exchange protocol used to create the 3DES secret key was Diffie-Hellman.

The majority of ideas for the PetCipherEngine class came from Section 3.5 "The *KeyAgreement* Engine" in Reference [2]: *Java Cryptography Extensions* by Jason Weiss.

6 Prototype properties and demonstration

The PET system project was presented on May 3rd 2005 to fellow students in the Rivier College CS699 Professional Seminar lab and was open to the student body, Computer Science professors and invited guests.

An overview of the project was presented, followed by a demonstration of the Version 1.0 prototype of the PET system. The following list contains highlights of the demonstration:

- An install of Version 1.0 of the PET System was performed on a Windows 2000 laptop.
- A PetServer and three PetClient Applications were started.
- All three PetClient applications logged into the PetServer.
- Each PetClient's friend list was updated as new users logged into the PetServer.
- An encrypted conversation was started between the users.
- Encrypted messages were sent between the Pet chat windows of each PetUser.
- Examples were provided of how conversations were terminated from the chat window, the main PetClient window, or from the peer connection.
- Corner case testing of PetServer login failures and exception handling for PetClient network disruptions were demonstrated.
- An overview of the Java code developed during the project and steps taken to create the NSIS Installer were finally presented.

6.1 Screen Shots of the Prototype

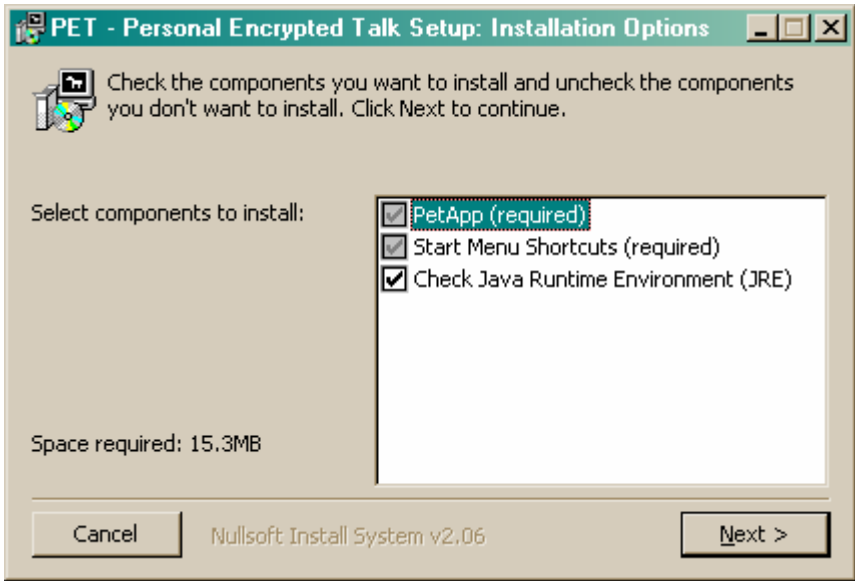

Figure 7: PET Installation Options Window

Figure 8: PetServer window with login and logout status updates

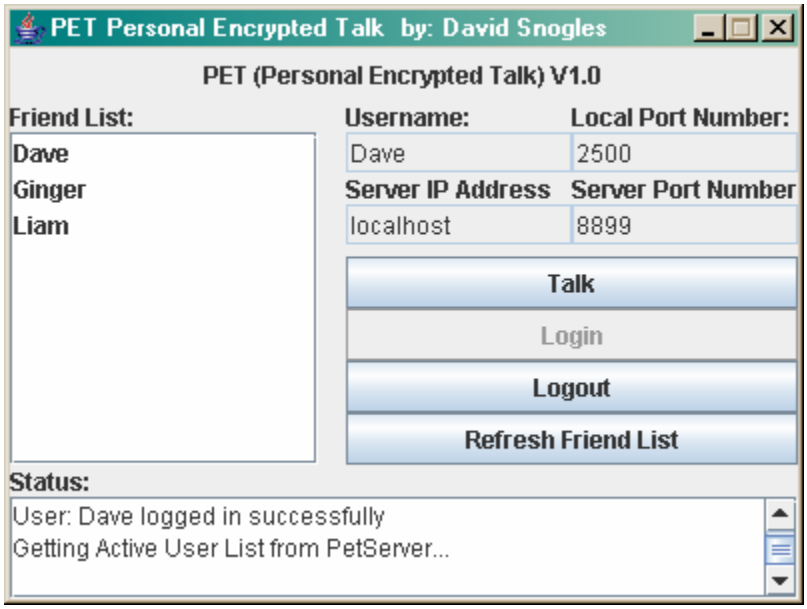

Figure 9: PetClient application login window

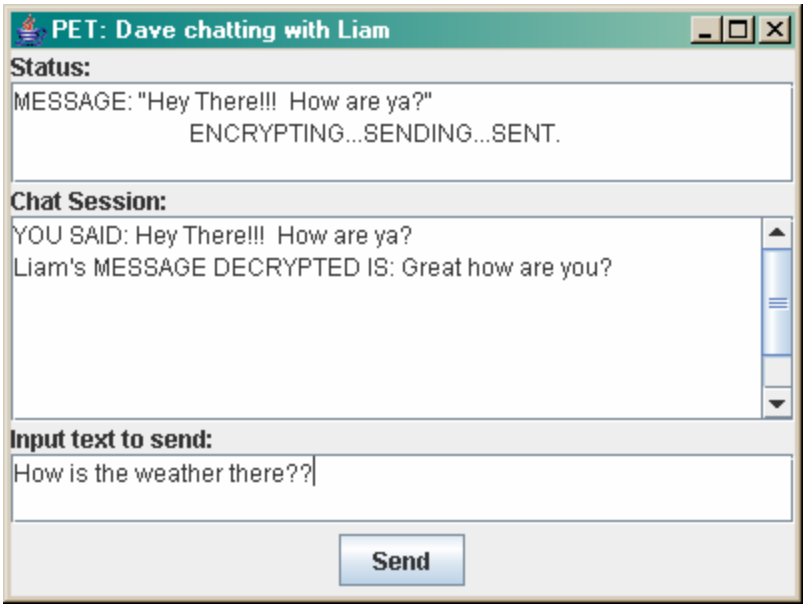

Figure 10: PetClient example chat window between user Dave and user Liam

7 Conclusions

Implementing the Personal Encrypted Talk application provided an excellent opportunity to put into practical use many of the software design skills the author developed over the last 2 years at Rivier College. The application meets the functional specifications for Version 1.0 as outlined in the Design Requirements section of this report. It meets the primary objective of securing Instant Messaging between two remote clients while bypassing an intermediate server. The project was designed with the ability to quickly implement advanced functionality in the future. One of the first advanced features that will be implemented in the next version is a Graphical User Frame for the PetServer with the ability to create user accounts with encrypted passwords. Enhancements to the Personal Encrypted Talk application after Version 1.0 will provide an excellent test bench for further study into the Java Cryptography Architecture and Java Cryptography Extensions.

8 References

[1] Simon St. Laurent, Joe Johnston & Edd Dumbill, Dave Winer, *Programming Web Services with XML-RPC.* O'Reilly & Associates, Inc., 2001

[2] Jason Weiss, *Java Cryptography Extensions: Practical Guide for Programmers*. Morgan Kaufmann Publications, 2004.

[3] Jonathan Knudsen, *Java Cryptography.* O'Reilly & Associates, Inc., 1998.

[4] Alan Dennis, Barbara Haley Wixom, David Tegarden, *Systems Analysis & Design An Object-Oriented Approach with UML.* John Wiley & Sons, Inc., 2002.

[5] Cay S. Horstmann, Gary Cornell, *Core Java 2 Volume I – Fundamentals.* Sun Microsystems Press, 2003.

[6] Cay S. Horstmann, Gary Cornell, *Core Java 2 Volume II – Advanced Features.* Sun Microsystems Press, 2005.

[7] Ralph Morelli, *Object-Oriented Problem Solving Java, Java, Java*, Second Edition. Prentice Hall, 2003.

[8] Wayne C. Booth, Gregory G. Colomb, Joseph M. Williams, *The Craft of Research*, Second Edition. The University of Chicago Press, 2003.

---

* **DAVID SNOGLES** is a Senior Hardware Engineer specializing in the development and verification of Application Specific Integrated Circuits (ASIC's) for networking applications. He received his Bachelor of Science degree in Computer Engineering Technology, summa cum laude, from the State University of New York at Utica in 1996. Graduate work was done at UMASS Lowell and Rivier College in Nashua, New Hampshire. He received his Master of Science in Computer Science degree from Rivier in 2005. David's research interests include: Electronic Design Automation (EDA) Technologies, Hardware Description and Verification Languages, Constrained Random Testing, Encryption Technologies and Object-Oriented Programming.

