7

The Graphical Interface for Secure Mail

F. Bračun, B. Jerman-Blažič, T. Klobučar, D. Trček

Laboratory for Open Systems and Networks,
Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia
Tel.: +386 61 177 37 39, Fax.: +386 61 123 21 18
E-mail: franc@e5.ijs.si

Abstract

To achieve broad acceptance for the PEM service it is necessary to offer well designed human interfaces to PEM implementations. In this paper, we describe a graphical interface for secure mail (PEM) to provide easy access to the PEM mechanisms for common users. The interface is embedded in the typical computing environment of our (non-technical) audience.

Keywords

PEM, MIME, Secure Mail, GUI, Motif, Cryptographic mechanisms

1 INTRODUCTION

Within the Internet, e-mail is certainly the most popular application among users, providing them with access to an enormous amount of information and contacts. After introduction of Multipurpose Internet Mail Extensions (MIME) (Borenstein, 1993), users also have the possibility to include richer data types in messages than just simple text, e.g. images, video and audio. However, the majority of the users is still unaware that most of the e-mail systems are not very secure. During the past few years, working groups within the Internet Engineering Task Force (IETF) and the Security Research Group of the Internet Research Task Force (IRTF) have been working on a security-enhancing standard for the Internet e-mail system. The result of this work is Privacy Enhanced Mail (PEM) standard (Linn, 1993) (Kent, 1993) (Balenson, 1993) (Kaliski, 1993) which provides several security services: message integrity, message origin authentication, message confidentiality and non-repudiation of origin. The last version of PEM is designed to support these services in the RFC 822 context, meaning that

R. Posch (ed.), *Communications and Multimedia Security*

they can be applied only to plain text messages. However, there is on-going MIME-PEM integration within the PEM Working Group in IETF.

To date several commercial (e.g. COST-PEM from Sweden) and public domain implementations of PEM have been introduced. Among latter we can mention the implementations that were developed during the PASSWORD (Luehel, 1992) project by the University College London, UK and German Gesellschaft für Mathematik und Datenverarbeitung (GMD). Public implementations of PEM therefore exist but users do not usually use them or they use other less sophisticated solutions for secure electronic mailing such as the Pretty Good Privacy (PGP) application. One of the reasons for the limited use of PEM (apart from the very small number of working certification authorities and simple public directories) is certainly the lack of good graphical interfaces that can provide the security mechanisms of PEM and confirm to the existing standards. Furthermore, the interface should be user friendly, meaning that a user can encipher and decipher messages, verify digital signatures, retrieve chains of certificates from the public directory (e.g. X.500 (CCITT, 1989)) etc., by simple selection of different icons on the screen. The graphical user interface for PEM called Xpem-GUI described in this paper, satisfies these requirements, i.e., it's use is simple and friendly. Xpem-GUI is an electronic mail user agent with an interface to PEM, designed for ease-of-use, while strictly adhering to Internet standards and distributed system architecture principles.

This paper is set out as follows: in the next section, a brief overview of the e-mail systems is given, along with basic security problems related to the electronic mail and a short description of PEM which offers solutions to these problems; our main contribution in this paper can be found in sections 3 and 4 where the Xpem-GUI is described. More detailed description of the interface can be found in (Bračun, 1994).

2 SECURE MAIL SYSTEM

At the moment, there are two standard e-mail systems: X.400 (CCITT, 1988) and SMTP (Postel, 1982). In the X.400 Message Handling System, the mail system can be modelled as a network of Mail Transfer Agents (MTAs) transferring mail to and from the User Agents (UAs). The UA is used by the user to compose mail messages, to view received messages from other users and to send messages to other users. In the Internet, the routing and delivery of the messages is done by use of the Simple Mail Transfer Protocol (SMTP).

Usually, the operation of electronic mail consists of the following steps: the user interacts with the UA and processes a document to be sent to specified user. At the end of the document creation, the user initiates transmission. The message is passed to the MTA responsible for routing of the message. When the recipient uses a UA, the MTA notifies the presence of the message, and the recipient is then able to access the message and process it.

In today's computer environment, e-mail is a very popular method of communication. The number of documents, video and audio streams that are transferred electronically is steadily increasing, and especially where official documents are concerned, a need has arisen to identify the status of these documents. It is therefore necessary to combine e-mail systems with security services. There are many security approaches that one can imagine for a message handling system. For example, one could attempt to secure the entire message transfer system by securing each message transfer agent and the links between them. However, such an approach has some problems:

- The bootstrapping problem.
- Depending on the security measures required, scaling could be problematic.
- It would not necessarily provide protection once an electronic mail message left the message transfer system.

This has led to a more pragmatic approach whereby the user agents are solely responsible for providing security-related facilities. The following security facilities may be available in the UAs:

- Preventing a third party from being able to examine the body of an e-mail message.
- Preventing a third party from being able to forge some other party's identity in an e-mail message.
- Preventing a third party from replaying, inserting, or deleting e-mail messages.
- Preventing an unauthorised third party from congesting the message transfer system so that UAs are unable to communicate.
- Preventing a third party from altering an e-mail message in transit.
- Preventing the originator of an electronic mail message from falsely denying having it.

We should note that even if the message transfer agent participate in providing security mechanisms, it is not clear whether all of these services could be provided in every MTA that process the message.

Now, let us consider the PEM concept. The User Agent (UA) with PEM can retrieve registered certificates that are stored in a publicly accessible database (e.g. a directory service based on X.500), while confidential information of a user (e.g. private key) is stored in his Personal Secure Environment (PSE). Users can generate PEM messages through their UA using local PEM software and PSE-resident information. These PEM messages are included as a body part in a UA-generated e-mail. The receiving party verifies and deciphers this message using its PSE and consulting the directory service for the certificates and certificate revocation lists.

The following are the security services which PEM offers:

- **Data confidentiality:** This service can be used to protect data from unauthorised disclosure. It can be used against interception of data.
- **Data integrity:** It provides data integrity during communication sessions. It can be used to detect or prevent manipulation of data.
- **Authentication of sender:** This guarantees the identity of a user. In this case, the user is the sender of the message.
- **Non-repudiation of sender:** This service guarantees the integrity and origin of data as far as the sender, not the recipient, is concerned.

All privacy-enhanced messages (PEM) start with the *Proc-Type* header which identifies the PEM version in use and the type of message separated by a comma. There are four types of messages, three of them are described below:

- **encrypted:** (ciphertext form provided with transformation module-base64 (Linn, 1993)); The body contains an e-mail message, in which all four facilities described earlier are provided.

- **mic-clear:** (plain text form represented using NTV ASCII); The body contains an e-mail message, in which originator authority and message integrity are provided. Any UA is able to view the message, but only PEM-capable UA will be able to verify the authenticity and integrity of the message.
- **mic-only:** (plain text form represented using base64); The body contains an e-mail message, in which originator authenticity and message integrity are provided, however only UA that are PEM-capable will be able to automatically view and represent the message.

The syntax of PEM headers follow the LWSP convention from RFC-822 (lengthy header fields can be split onto multiple lines, provided that each subsequent line starts with white space). The use of the PEM security capabilities involves some cryptographic tools. These are used for data encryption, message integrity and key encryption.

2.1 Key Management

Security services based on a cryptographic mechanism (e.g. encipherment) assume cryptographic keys to be distributed to the communication parties prior to secure communications taking place. This causes one of the most subtle problems when integrating cryptographic functions into networks, since key management has to guarantee that the right keys are available at the right time and in the right place. In the Open Systems Interconnection Reference Model, Security Architecture, Part2 (ISO, 1988) key management is defined as "generation, storage, distribution, deletion, archiving and application of keys in accordance with security policy". The purpose of key management, therefore is to provide procedures for handling cryptographic keying material to be used in cryptographic mechanisms (symmetric or asymmetric).

Two kinds of encryption algorithms are known - **symmetric** (also known as secret-key algorithms, like DES (ANSI X3.92, 1981)) and **asymmetric** (also known as public-key algorithms, e.g. RSA (Rivest, 1978)). The former use the same key for encryption and decryption, and the latter use one key for encryption (i.e., private key) and another key for decryption (i.e., public key). Asymmetric algorithms have been developed to eliminate the risk of revealing a key while transmitting, which is the main disadvantage of symmetric algorithms. With asymmetric algorithms there is no need to transmit the private key, while the public key can be communicated to anyone. The asymmetric algorithms can also provide a method for digital signature. The main disadvantage of public-key algorithms is their complexity - they are 100 to 10.000 times slower then symmetric algorithms. Generally, the logical conclusion is to use both kinds of algorithms and combinations of them to achieve optimal speeds and security levels.

Key management is a crucial task for the provision of secure networking. Although public key cryptography (e.g., RSA) significantly reduces the complexity of key management over symmetric key cryptography (e.g., DES), a proper authentication of public key(s) remains critical because this is the starting point of every security service if the following is required:

- global (open) and
- lawful solutions (lawful in that sense that electronic documents, which were created by use of security services, can be taken to the court as evidence when, for example, signing a contract via electronic means).

This means that the public key cryptography will be the basis for fulfilling the requirements mentioned above. Verification of the binding between a user and his or her public key is achieved by **certificates** (Balenson, 1993), which are issued by trusted entities, otherwise known as **certification authorities** (CAs). In the certificate the public key of a user is signed by a trusted party. There is also the name of the issuer (i.e., trusted party), the owner's name, serial number of the certificate, validity period and algorithm identifier (formal description can be found in (Balenson, 1993)).

For issuing certificates an established system of CAs is required which provide the users with their certificates. It is reasonable to assume that there will be a large number of CAs in the world, because the task of issuing certificates to millions of Internet users will have to be distributed between CAs. These CAs constitutes the public key infrastructure which defines the CAs, their operation and interrelations.

This infrastructure is expected to be responsible for distribution and cancellation of user certificates. CAs in the system will also issue certificates to one another as well, depending on their (hierarchical) relationship. The sequence of such certificates will enable every Internet user to check the public key of another user.

3 THE GRAPHICAL INTERFACE FOR SECURE MAIL

The user interface is an interface between the application and the user. The goal in developing a GUI was to help a user to interact with an application simply and naturally. Therefore, a user interface was designed in a way that provides an easy use of the security tools. The following tasks were also envisaged in the development of the interface:

- integration with standard UNIX mail programme;
- no changes to the available public code;
- minimal system dependence;
- portability;
- user friendly;
- use of a popular User Agent;
- enhancement on multimedia mail.

The GUI was envisaged as a part of a desktop environment that integrates office automation and productivity applications. Our Xpem-GUI integrates e-mail (with MIME capabilities) and PEM capabilities allowing a user to import multimedia data, sending documents and applications files, and even launch third-party product without ever leaving the GUI.

3.1 The architecture of Xpem-GUI

The architecture of the GUI is presented in Figure 1. The idea during the development of the GUI was to keep an interface flexible, therefore the architecture was split into two modules (platforms). The motivation for doing this was as follows: with improved flexibility the users are enabled to select the best method of accessing a function based on a criterion they choose. The GUI was also designed to be configurable (e.g. one may chose to use the mailer ELM instead of Mail).

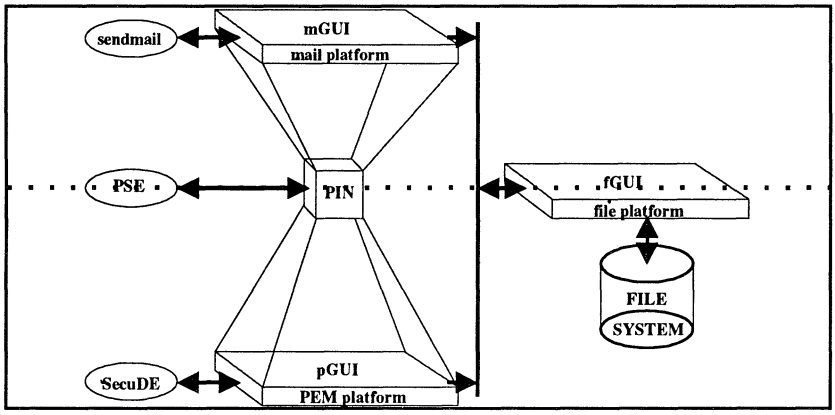

Figure 1 The architecture of Xpem-GUI.

The two modules of the GUI are PEM platform and Mail platform, with an additional third supporting platform: File platform.

With this separation, the GUI can work either in a mail environment or in a PEM environment independently, so one can use all PEM mechanisms with other documents existing in the system.

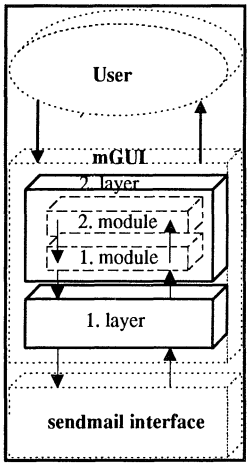

Figure 2 The mGUI module

3.2 The Mail platform (module mGUI)

By starting the programme, the Mail platform is accessed which enables work with an Internet mail UA through a user friendly graphical interface. The Mail platform consists of two layers. The first one is responsible for communication between the GUI and "sendmail interface"* , and the second one is responsible for data processing and passing (or receiving) data to (from) the first layer. Figure 2 represents the structure of the Mail platform.

The first layer

Communication between the second layer and the "sendmail" interface (one can choose one's favourite interface; we use Sun mail MUA - Mail User Agent) is provided with two stream pipes (Figure 3). On one hand, these pipes are connected with the first layer, and on the other with the "sendmail" interface. The system call *write* is responsible for sending commands and data to "sendmail" interface through *mail_in* stream pipe while system call *read* is responsible for receiving data through *mail_out* stream pipe. This is done with two functions: *xpem_mail()* to send data and *xpem_readln()* to receive it.

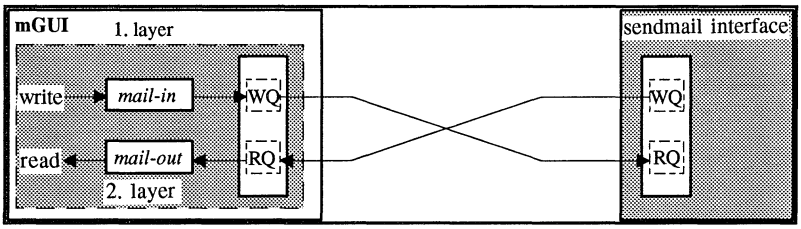

Figure 3 Communication between mGUI and sendmail interface

The second layer

The second layer consists of two modules (see Figure 2). The first module is responsible for exchanging and transferring data between the two layers. This module also transforms actions and data from the second layer in the reasonable form for the first layer.

The second module of the second layer performs communication between the user and the first module. This module, which is a window-oriented interface between user and *mGUI,* is realised with Motif (OSF/Motif, 1993) library. The main window is divided into four main areas:

- menu with pull-down command menus;
- a collection of push buttons for some most commonly used actions;
- a listing (table of contents) of the messages in the open folder; and
- an area which contains some useful information of an open folder and relevant messages.

This module also contains sub-windows for composing, reading, replying and forwarding messages. They are divided into four main areas:

* Sendmail interface is the most common used implementation of a message transfer agent in the Internet.

- menu area with pull-down command menus;
- an area, which contains information on current message header and information on the open folder;
- a view of the selected message in a text widget (window) for reading, forwarding, replying and composing;
- a collection of push buttons for some most commonly used actions.

Connection between two modules is provided with call-back resource functions of the Motif Widget Set Tool Kit.

3.3 The PEM platform

PEM mechanisms are used at the PEM platform of GUI. The use of these mechanisms requires knowledge of the Personal Identification Number (PIN) code of the PSE. When the PEM button in the GUI window is selected, a dialogue box for login appears on the screen. Xpem-GUI offers security facilities included in PEM and mechanisms for data (messages) processing. In general, it is divided into two modules. The first one is responsible for data processing and the second one is responsible for security tasks. Figure 4 shows the structure of the PEM platform (pGUI).

The module for data processing and text editing enables creating and saving messages by fGUI. It consists of two parts: a text editor with all functions and a file manager, realised by Motif widget set. The second module (security facilities) communicates with PEM interface which performs security tasks. In implementation, the PEM interface (API) from the SecuDE library of GMD (Scheider, 1994) is used.

The module for data processing

This module has all text editor functions and is similar to the second module in the second layer of the mail platform. Processed data is stored in a temporary file which is an input message file for the PEM interface. The result of the PEM tasks is stored in another temporary file. From the file platform, file can be passed to a pGUI for data processing.

Communication with the PEM interface (second module)

The main role of this module is to establish communication between the PEM platform and the PEM interface. It then passes an appropriate command to the PEM interface which is realised with two main functions: *pem_read()* and *pem_write()* (one can use other PEM interface instead of GMD (Scheider, 1994)).

For selected action, the selection of appropriate functions with the right parameters is required. Because a PEM interface is an independent application, the transformation table on Figure 5 is used. Through appropriate selection of the three parameters (*writepem, encr, and clear*) an appropriate operation (*scan, mic-clear, mic-only, encrypted*) is defined. Figure 5 shows the communication between a user and the PEM interface.

pem_write() (mic-clear, mic-only, encrypted)

Creates and writes PEM text, and reports errors to a caller. It reads the input file (/tmp/pem_input), opens the caller's PSE and writes the created PEM message into the output file (/tmp/pem_output).

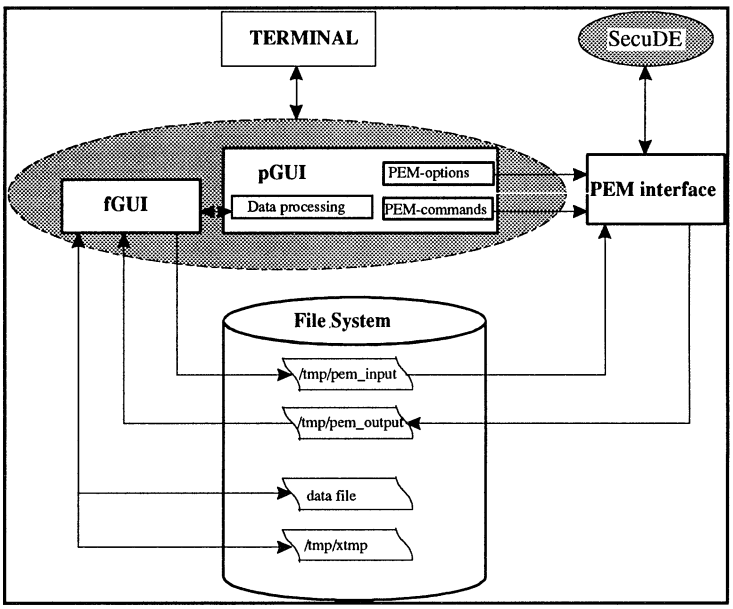

Figure 4 The structure of the PEM platform (pGUI)

pem_read() (scan)
This reads and scans PEM text, and reports errors to a caller. It reads the input file, opens the caller's PSE and writes the (decrypted) PEM body in clear text into the output file.

4 THE IMPLEMENTATION OF XPEM-GUI

The screen presentation of the user interface is split into four areas (menu, collection of icons, table of contents, and information area) (see Figure 6). The menu area contains a list of offered operations. The second area contains a collection of icons representing some most often used operations (e.g. read, save, reply, forward, scan, mic-only, mic-clear, encrypt, delete), while the third area contains a list of messages (Table of Contents -TOC) in the folder that is currently opened. The last area contains some useful information about the opened folder and the selected messages.

All operations in Xpem-GUI use the concept of a 'current message'. With a click on a message in the TOC, the user replaces the current message with the selected one which is then used for the next operation. A double-click on a selected message initiates a view of that message. The user can select more than one message in a TOC by clicking on different messages and holding Ctrl key. All selected messages can be saved or deleted, however only the first selected message (i.e. current message) can be read.

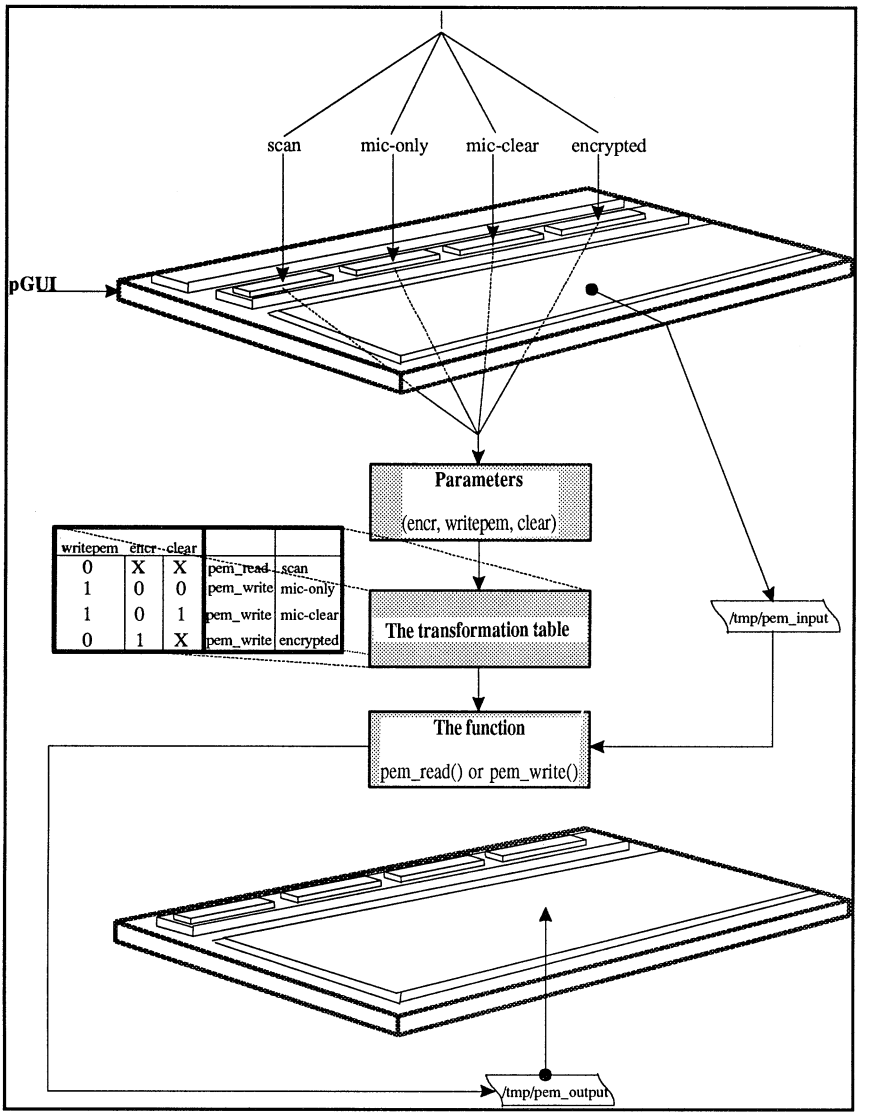

Figure 5 Communication between a user and the PEM interface

4.1 Processing my mail

In this section, we define the concept of the selected message, current message, selected folder and current folder, and introduce some common used commands.

The selected folder

The selected folder is whichever folder name appears in the bar under the TOC. We can change the selected folder by pressing the File menu button and selecting the Folder Manager item. This will initiate the dialogue window for selecting folder.

The TOC lists the messages in the viewed sequence of messages within the viewed folder. The file command menu contains commands of a global nature: Open Folder, Create Folder and Delete Folder , which are part of the folder manager.

The selected messages are highlighted. The current message is indicated by pressing '>' next to the message number. It usually corresponds to the message currently being viewed. When a message is viewed (in a new window), the text widget above the view will identify the message. We select a new current message by typing the message number in this text widget and pressing the Enter key.

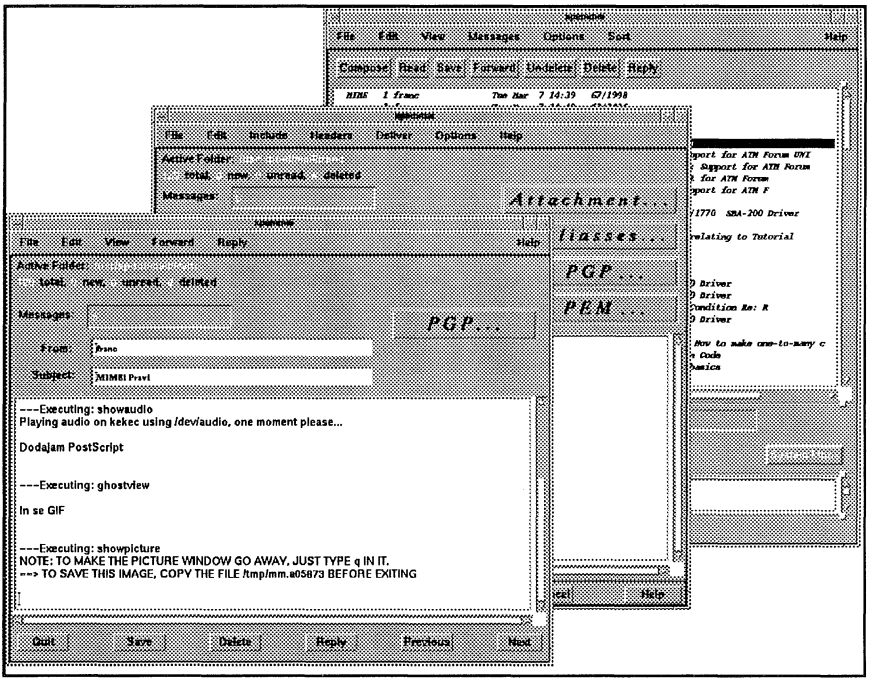

Figure 6 User interface for Secure Mail

Table of contents commands

The TOC command menu (e.g. Edit, View) contains commands which operate on the opened or viewed folder. The name of each command button indicates the meaning of the operation; e.g. save, delete or preserve a message. These commands are relevant for all selected messages.

Message commands

These are the commands which operate on selected messages, e.g. read message, compose message (a new window will be brought up for composition), view next message, view previous message, reply (create a composition window in reply to first selected message), forward (create a composition window whose body is initialised to be the contents of the selected messages), print (print the selected messages).

View commands

Commands of view windows (which result from message command read) correspond in functionality to the commands from the message command menu, but operate on the viewed message rather than the selected or current message: quit (close window), reply, forward, save, edit (copy, cut, paste), print. There is also a scan command which is used for reading a message which includes message privacy or message integrity services.

Composition window

Aside from the normal text editing functions, there are some command buttons associated with a composition window:

- ***Quit*** Close this composition window;
- ***Save Message*** Save this composition in your drafts folder;
- ***Send*** Send this composition;
- ***Insert*** Insert a related message into the composition.

There are also three command buttons for composing messages which include either message privacy or message integrity services:

- ***Mic-only;***
- ***Mic-clear;***
- ***Encrypt.***

Beside these functions, there is a menu button with options for selecting a message integrity check algorithm (MIC), data encryption algorithm, data encrypting key algorithm and some other parameters that are needed for PEM.

Help

For each function and parameter, the corresponding recommendation is indicated to advise the user. The Xpem-GUI Help gives a user the possibility to easily learn more about PEM framework.

5 CONCLUDING REMARKS

The lack of very good user interfaces to PEM is certainly one of the reasons why PEM was not widely used by users without a technical education. In this paper we have presented our solution to this problem, a graphical interface called Xpem-GUI. Current experiences with the tool have shown that:

- the graphical interface considerably reduces the complexity of use of secure mail;
- the response times of the interface are sufficiently short;
- the users are very satisfied with the ease of listing, reading, forwarding, replying and performing of PEM operations.

We hope that our contribution will help in further wide spreading of security enhanced e-mail systems.

6 REFERENCES

American National Standards Institute, Data Encryption Algorithm, ANSI X3.92, New York, 1981.
Balenson D., Privacy Enhancement for Internet Electronic Mail: Algorithms, Modes and Identifiers, RFC 1423, February 1993.
Borenstein N., Freed N: MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies, RFC 1521, September 1993.
Bračun F. Trček D., Jerman-Blažič B., Grafični vmesnik za PEM, IJS DP 7090, October 1994.
CCITT Recommendations X.400, 1988.
CCITT Recommendation X.509, The Directory - Authentication Framework, 1989.
International Standard Organisation, Information Processing Systems, Open Systems Interconnection Reference Model - Security Architecture, ISO 7498-2, July 1988.
Kaliski B., Privacy Enhancement for Internet Electronic Mail: Key Certification and Related Services, RFC 1424, February 1993.
Kent S., Privacy Enhancement for Internet Electronic Mail: Certificate-Based Key Management , RFC 1422, February 1993.
Linn J., Privacy Enhancement for Internet Electronic Mail: Message Encipherment and Authentication Procedures , RFC 1421, February 1993.
Luehe J., R2.6: User Requirements, Ver. 1.0, PASSWORD Project, November 1992.
OSF/Motif, Style Guide, Prentice Hall, New Jersey, 1993.
Postel J. B., Simple Mail Transfer Protocol, RFC 821, 1982.
Rivest R.L., Shamir A., Adleman L., A Method for Obtaining Digital Signatures and Public Key Cryptosystems, Comm. ACM, Vol. 21, No. 2, February 1978, pp. 120-126.
Scheider W. et al., SecuDE 4.3, GMD, Darmstadt, Germany, 1994.

7 BIOGRAPHY

Franc Bračun was born in Novo mesto, Slovenia in 1968. He received his B.S. degree in electrical engineering in 1993 at the University of Ljubljana. That same year he joined the Laboratory for Open Systems and Networks, Jožef Stefan Institute, Slovenia. His primary research interest is in computer and network security, and in the general area of the integration of graphics and communications services. Currently he is working for a master's degree.

Borka Jerman-Blažič is a chair of the Laboratory for Open Systems and Networks at Jožef Stefan Institut, Ljubljana, Slovenia. She is teaching postgraduate course on

Telecommunication Services at the Faculty for Economics, University of Ljubljana. She is a member of the ARNES (Slovenian academic network) Steering Committee and a member of RARE Technical Committee. She is the convenor of RARE WG on Character Sets and national representative to ISO JTC1 SC2, JTC1 SC22WG20 and CEN TC 304. Her research currently focuses on networks applications and issues related to internationalisation of software applications. She is also interested in development of security services and CA infrastructure. She is author of more then 150 published scientific papers.

Tomaž Klobučar was born in Ljubljana, Slovenia in 1967. In 1992 he graduated in mathematics from the University of Ljubljana. Next year he joined the Laboratory for Open Systems and Networks, Jožef Stefan Institute, Slovenia. His primary research interest is computer and network security (especially key management). Currently he is working for his master's degree.

Denis Trček is with the Laboratory for Open Systems and Networks, Jožef Stefan Institute, Slovenia. As a scientist he has been involved in the field of security in computer networks for more than four years. He has received his M. Sc. from University of Ljubljana, Slovenia, and is currently finishing his PhD thesis on security in communications networks.