(2)

Privacy Enhanced Mail
A User's Perspective

Contract No. F19628-88-D-0031

PUBLICATION NO. TM-8626/100/02

Aug 1, 1991

Prepared for:
Unisys Defense Systems
12010 Sunrise Valley drive
Reston, VA 22091

Prepared by:
Unisys Defense Systems
Networks and Information Security Systems
5151 Camino Ruiz
Camarillo, CA 93010

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1, 1991	Technical Manual

4. TITLE AND SUBTITLE
Privac, Enhanced Mail, A User's Perspective

5. FUNDING NUMBERS
F19628-88-D-0031

6. AUTHOR(S)
Unisys Corporation
Networks and Information Security Systems

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Unisys Defense Systems
12010 Sunrise Valley Drive
Reston, VA 22091

8. PERFORMING ORGANIZATION REPORT NUMBER
TM-8626/100/02

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
ESD/AVS
Bldg. 1704
Hanscom AFB, MA 01731-5000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
TM-8626/100/02

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
For general distribution

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document provides a description of the installation and experimentation efforts of Unisys Networks and Information Security Division (NISD), on the Privacy Enhanced Mail version 5.

Privacy Enhanced Mail is based upon the RFCs under development to provide security for internet mail.

14. SUBJECT TERMS
Privacy Enhanced Mail.

15. NUMBER OF PAGES
17

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

1. Introduction

The purpose of this document is to provide a description of the installation and experimentation efforts of Unisys Networks and Information Security Division (NISD), West Coast Operations (WCO), on the Privacy Enhanced Mail version 5.0.

Privacy Enhanced Mail (PEM) is being developed by Trusted Information Systems of Glenwood, Maryland. PEM is based upon the the RFCs under development to provide security for internet mail. While those RFCs are not finalized, TIS, GTE Contel Federal Systems, and others have initiated independent development of PEM products.

Unisys, West Coast Operations, was selected by the STARS program to beta test PEM for its usefulness in the STARS Software Engineering Environment (SEE). The STARS SEE environment is anticipated to be a widely distributed cooperative effort between software developers and their managers. As such, the need for integrity in the transfer of program units and mail is critical. PEM offers the opportunity to transfer information with either integrity or confidentiality protection, and source authentication. These capabilities provide the ability to detect the modification of information during the transfer and to prevent the unauthorized observation of data. The following excerpt from the TIS PEM User's Manual provides an overview of the TIS/PEM Process:

Initially, a user registers with the TIS/PEM system. When registering, the user is assigned an RSA public key/private key pair. The public key is the primary element in the user's certificate that makes verification of electronic signatures possible. By electronically acquiring each others certificates through registered mail messages, the users can communicate using all of TIS/PEM's security services. The DES encryption algorithm is used for encryption of the message contents when requested. A unique DES key is generated for each message, and that key is wrapped with RSA and included in the PEM message header.

2. Installation of PEM

We installed MH and PEM on a Sun 3. The installation went reasonably smoothly. A few problems were encountered in the compilation process. These problems are enumerated in Appendix A along with the approach we used to circumvent them. As delivered, PEM is incorporated into MH, a public domain mail system developed by RAND. As a result, there are a number of options that must be established during the installation. It is essential that the installer understand the way mail is handled to successfully complete the installation. It is not a task for casual users.

Once PEM and MH are installed, the registration of the users can be established. While this is not difficult, the users have to understand that they have to be registered before using PEM. That is not necessarily obvious to them. The site PEM administrator can establish the registration process such that users can register themselves, or it must be done by the administrator.

The programs that perform the registration process are very confusing at first. There is adequate information in the Installation and CAI Manual. However, there is no indication of how to properly sequence the use of cai and mapdir. It took numerous attempts to get the certificates generated properly. The proper procedure is to use mapdir first to establish the user to mailbox and Distinguished Names for the user. Then use cai to create the certificate.

Maintaining the certificates is not easy. There is no way to determine what is in the certificate files unless you know the specific entry. We did find out, through discussions with the TIS developers, about an undocumented utility that gives you the ability to read the certificates, one at a time. That enabled us to identify the bad ones and delete them. We understand that the next release of PEM will eliminate the cai/mapdir distinction and provide a simpler mechanism for generation and maintenance of certificates. It is definitely needed.

3. PEM Performance

Our Sun 3 does not have any DES hardware support, so we anticipated slow encryption and decryption. That did not turn out to be the case. Encryption and decryption times for a 100kb file were 1 minute, and for a 500kb file were 2 minutes[1]. These seem reasonable for a machine that is in use by a number of other users. The resulting files were compared with the original files and only the following line was inserted by PEM into the message header:

```
X-PEM-Info: INTEGRITY AUTHENTICATION CONFIDENTIALITY
```

PEM is initially based on MH. We found that MH was not suitable for use in the SEE. It is difficult for a non-Unix guru user. Therefore, we examined the possibility of integrating PEM into ELM, a menu based mail handler. That turned out to be fairly straight-forward. With a couple of days effort, we had a demonstration quality integration. This was achieved without any modifications to the PEM code. Complete interoperability between ELM-PEM and MH-PEM was demonstrated. PEM appears to be easily adapted for other mail handlers. With PEM in ELM, we now have a mail system that can be used by a variety of users.

The primary vehicle we used from PEM were the two modules catscc and catrcc. While these are not documented, they provide a shell level interface to the encryption and decryption modules scc and rcc. Catscc requires a short message header with a "to" and "from" line followed by a blank line. Our initial understanding was that line with 8 or more dashes was required, and we used that successfully. Recent information from TIS indicates that a blank line is the preferred separator.

It was straight forward to modify ELM to append these to a file, run catscc and then mail the resulting file. While this approach is not very efficient, it was easily implemented. Approximately five processes are created during the encryption of a message. The clean approach would integrate all the file manipulation and the call to rcc into ELM directly, thus avoiding the process creation overhead. However, this overhead is

[1] All time measurements were made on one specific machine and should only be used for relative reference to each other. Actual times are dependent on the specific processor used and other processes active at the time.

not significant to the user given the fairly long delays caused by the key establishment procedure.

Once a message starts encryption, there is a fairly long delay while the RSA algorithm is used to generate and wrap the transmission encryption key. For example, encrypting a 50 byte message required 20 seconds. Hence, this overhead is very close to 20 seconds for each message. Once that is complete, the actual DES encryption is very quick by comparison.

4. Key Management

Key Management for PEM is a new concept for most users. You just can't send encrypted mail to anyone. The certificates have to be established at both ends of the communication before encryption can be used. This is not difficult to accomplish, but it is not intuitively obvious to users. This situation will exist for any form of protected communications and only requires some understanding on the part of the users. It does not detract from the usefulness of PEM.

There is no easy technique for a user to determine if a certificate exists for a particular user. Cai can show certificates if the proper Distinguished Name is known, or the mailbox address that was used when the certificate was first received. Users frequently use aliases to reduce typing and reliance on remembering long network addresses. It may be difficult for a user to get the desired answer from cai. It would be very helpful if a utility existed that showed the list of certificates.

We understand that TIS desired to integrate PEM with X.500 rather than develop a certificate database system themselves. This is probably a sound approach, however, it does make PEM operations considerably harder for users in the interim.

5. Usefulness for Electronic Mail

PEM is a practical solution to ensuring either integrity or privacy over a network. We tried a number of approaches to modify a message without detection. Every case was detected by PEM and the recipient was notified that the message had been corrupted. We did not try a direct attack on the encryption algorithm. It was too much work for a low probability of success. However, we did try to play with the certificates to see if we could decrypt messages for which we were not a recipient. None of the obvious approaches worked.

We understand that a cryptologic attack did identify a flaw in the key generation process that will be corrected in the next release.

We did find a few small bugs. Appendix B details these problems. There are a wide number of address forms in use throughout the internet community with gateways to other various networks. It is not easy to develop code that can properly deal with all the various forms of possible addresses. We have not been able to test many of these forms because of the limited availability of PEM in the network. Updates recently supplied by TIS have fixed the problems we encountered in address forms.

6. Usefulness for File Transfer

PEM is not optimized for file transfer. You have to manually include the file in a message on transmission, and then extract it on receipt. We used this approach on a number of file types. In all cases, a diff of the original file and the decryped file showed no changes had occurred. An examination of the code for PEM indicates that it would be a fairly easy task to create a tool that automatically created the message header required by PEM, encrypted the message and mailed it. Likewise, a similar tool on receipt could decrypt the message, strip off the header and leave a duplicate copy of the original file. A demonstration version of those tools could be built using shell commands. Likewise, it appears that PEM could be fairly easily incorporated directly into ftp.

7. Conclusions

Although PEM is only in beta test at this time, it appears to be a stable product. The integrity mode will be useful in the SEE for the transfer of files between users. While PEM cannot guarantee that changes are not made during the transfer, it will detect any and notify the recipient so that the transfer can be reinitiated. PEM is also useful for electronic mail. We suspect that the privacy mode will be most useful for mail. By incorporating PEM into a number of different mail handlers, users will not be tied to one handler but will be able to use their favorite. Specialized tools for file transfer can easily be created. Such tools will also aid in the usefulness of PEM in a development environment. One significant advantage to PEM is that the users can develop such tools.

Appendix A
Problems Encountered During MH and PEM Installation

Here are notes regarding the problems encountered in building PEM and MH. They are generally of the form:

```
<Action>
<Errors>
<blank line>
<Fix, if any>
<_____________________________________________>
```

```
cd /home/kronos/steve/pem5.0/mh ; make:
cc -O   -DSUNOS4 -DTTYD -DDUMB -DMHE -DNETWORK -DRPATHS -DSBACKUP='"\043"' -
DSENDMTS -DSMTP -DPEM -DCLOSED -c closefds.c
"closefds.c", line 13: _NFILE undefined
*** Error code 1
make: Fatal error: Command failed for target `closefds.o'
Current working directory /home/kronos/steve/pem5.0/mh/sbr
*** Error code 1
make: Fatal error: Command failed for target `all'

Cut out cpp checks re BSD42, just uses gettabledsize().
```

```
cc -O   -DSUNOS4 -DTTYD -DDUMB -DMHE -DNETWORK -DRPATHS -DSBACKUP='"\043"' -
DSENDMTS -DSMTP -DPEM -DCLOSED -c m_getfld.c
"m_getfld.c", line 243: warning: illegal pointer combination
"m_getfld.c", line 257: warning: illegal pointer combination
"m_getfld.c", line 297: warning: illegal pointer combination
"m_getfld.c", line 304: warning: illegal pointer combination
"m_getfld.c", line 306: warning: illegal pointer combination
"m_getfld.c", line 316: warning: illegal pointer combination
"m_getfld.c", line 352: warning: illegal pointer combination

No action.
```

```
cc -O   -DSUNOS4 -DTTYD -DDUMB -DMHE -DNETWORK -DRPATHS -DSBACKUP='"\043"' -
DSENDMTS -DSMTP -DPEM -DCLOSED -c m_gmsg.c
./../h/local.h: 12: Can't find include file dir.h
*** Error code 2
make: Fatal error: Command failed for target `m_gmsg.o'
Current working directory /home/kronos/steve/pem5.0/mh/sbr
*** Error code 1
make: Fatal error: Command failed for target `all'
```

Cut out cpp checks re BSD42, just uses <sys/dir.h>. I think maybe the division between BSD42 and SUNOS4 is not clean: this could mean lots of ugly problems.

```
cc -O   -DSUNOS4 -DTTYD -DDUMB -DMHE -DNETWORK -DRPATHS -DSBACKUP='"\043"' -
DSENDMTS -DSMTP -DPEM -DCLOSED -c m_sync.c
"m_sync.c", line 69: warning: illegal pointer combination
"m_sync.c", line 70: warning: illegal pointer combination
"m_sync.c", line 71: warning: illegal pointer combination
"m_sync.c", line 72: warning: illegal pointer combination

No action.
```

```
cc -O   -DSUNOS4 -DTTYD -DDUMB -DMHE -DNETWORK -DRPATHS -DSBACKUP='"\043"' -
DSENDMTS -DSMTP -DPEM -DCLOSED -c pidwait.c
"pidwait.c", line 24: warning: illegal pointer combination
"pidwait.c", line 25: warning: illegal pointer combination
"pidwait.c", line 26: warning: illegal pointer combination
"pidwait.c", line 27: warning: illegal pointer combination

No action.
```

```
( In conf/config. )
cc -O    -DSUNOS4 -DTTYD -DDUMB -DMHE -DNETWORK -DRPATHS -DSBACKUP='"\043"' -
DSENDMTS -DSMTP -DPEM -DCLOSED -I..  -target sun3 -c  mts.c
```

```
mts.c, 21: Can't find include file whoami.h
*** Error code 2
make: Fatal error: Command failed for target `mts.o'
Current working directory /home/kronos/steve/pem5.0/mh/zotnet/mts
*** Error code 1
make: Fatal error: Command failed for target `allaux'
Current working directory /home/kronos/steve/pem5.0/mh/zotnet
*** Error code 1
make: Fatal error: Command failed for target `all'

Took out references to SYS5, just uses <sys/utsname.h>.
```

```
( In zotnet/mts. )
cc -O    -DSUNOS4 -DTTYD -DDUMB -DMHE -DNETWORK -DRPATHS -DSBACKUP='"\043"' -
DSENDMTS -DSMTP -DPEM -DCLOSED -I..  -target sun3 -c  mts.c
mts.c: 21: Can't find include file whoami.h
*** Error code 2
make: Fatal error: Command failed for target `mts.o'
Current working directory /home/kronos/steve/pem5.0/mh/zotnet/mts
*** Error code 1
make: Fatal error: Command failed for target `allaux'
Current working directory /home/kronos/steve/pem5.0/mh/zotnet
*** Error code 1
make: Fatal error: Command failed for target `all'

Took out references to SYS5, just uses <sys/utsname.h>.
```

```
cc -I../../h -DDEBUG -DLKM_SERVER='"/eagle1/2.6-
MSD/tmach/home/kronos/steve/pem5.0/mh/new/lib/mh/lkm"' -DSUNOS4 -DTTYD -DDUMB
-DMHE -DNETWORK -DRPATHS -DSBACKUP='"\043"' -DSENDMTS -DSMTP -DPEM -DCLOSED
-target sun3 -c  log.c
"log.c", line 102: O_WRONLY undefined
"log.c", line 102: O_APPEND undefined
"log.c", line 104: O_CREAT undefined
"log.c", line 533: O_WRONLY undefined
"log.c", line 533: O_APPEND undefined
"log.c", line 533: O_TRUNC undefined
```

```
*** Error code 1
make: Fatal error: Command failed for target `log.o'
Current working directory /home/kronos/steve/pem5.0/src/lib/log
*** Error code 1
make: Fatal error: Command failed for target `default'
Current working directory /home/kronos/steve/pem5.0/src/lib
*** Error code 1
make: Fatal error: Command failed for target `all'
Current working directory /home/kronos/steve/pem5.0/src
*** Error code 1
make: Fatal error: Command failed for target `all'
```

At this point examination of log.c showed that definition of the symbol BSD42 was not only affecting the inclusion of .h files, but also the choice of executible statements in the source code. In addition, the file log.c is not on
the MH source code branch, but rather on the PEM source code branch. This cast doubt on the efficacy of the above actions, so I decided to rebuild the entire PEM and MH branches with both symbols BSD42 and SUNOS4 defined. Note that problems due to the non-definition of the symbol SYS5 are still possible. This ambiguity arises from page 8 of the PEM Installation Manual, Draft #1.20.

Prior to completely rebuilding, I ran find on the all the .c and .h files to search for the patterns BSD, SUNOS4, and SYS5. Using the results of this, I was
unable to find any efforts of the form " #if ... #else #if ... " to explicitly segregate BSD, SUNOS4, and SYS5 dependencies. Since any more thorough effort to
find errors in the use of these dependencies would be time-consuming, I began the complete rebuild.

```
cc -I../h -DDEBUG -DLKM_SERVER='"/home/kronos/steve/pem5.0/mh/new/lib/mh/lkm"'
-DBSD42 -DSUNOS4 -DTTYD -DDUMB -DMHE -DNETWORK -DRPATHS -DSBACKUP='"\043"' -
DSENDMTS -DSMTP -DPEM -DCLOSED -DCAI  -target sun3 -c  write_user.c
make: Fatal error: Don't know how to make target `../lib/libbsafe.a'
Current working directory /home/kronos/steve/pem5.0/src/cai
*** Error code 1
```

```
make: Fatal error: Command failed for target `all'
Current working directory /home/kronos/steve/pem5.0/src
*** Error code 1
make: Fatal error: Command failed for target `all'

Changed libbsafe.a to libbsafe-mc68020.a in src/cai/Makefile.pre.

-------------------------------- ----------------

cc -I../h -DDEBUG -DLKM_SERVER='"/home/kronos/steve/pem5.0/mh/new/lib/mh/lkm"'
-DBSD42 -DSUNOS4 -DTTYD -DDUMB -DMHE -DNETWORK -DRPATHS -DSBACKUP='"\043"' -
DSENDMTS -DSMTP -DPEM -DCLOSED -DLKM  -target sun3 -c  process.c
make: Fatal error: Don't know how to make target `../lib/libbsafe.a'
Current working directory /home/kronos/steve/pem5.0/src/lkm
*** Error code 1
make: Fatal error: Command failed for target `all'
Current working directory /home/kronos/steve/pem5.0/src
*** Error code 1
make: Fatal error: Command failed for target `all'

Running "gen " in "catpem"
make: Fatal error: Don't know how to make target `../../lib/libbsafe.a'
Current working directory /home/kronos/steve/pem5.0/src/interfaces/catpem
*** Error code 1
make: Fatal error: Command failed for target `default'
Current working directory /home/kronos/steve/pem5.0/src/interfaces
*** Error code 1
make: Fatal error: Command failed for target `all'
Current working directory /home/kronos/steve/pem5.0/src
*** Error code 1
make: Fatal error: Command failed for target `all'

Changed libbsafe.a to libbsafe-mc68020.a in
src/interfaces/catpem/Makefile.pre.
Changed libbsafe.a to libbsafe-mc68020.a in src/tools/regroot/Makefile.pre.
Changed libbsafe.a to libbsafe-mc68020.a in src/tools/dumpcfile/Makefile.pre.

-----------------------------------------

cc -O   -DBSD42 -DSUNOS4 -DTTYD -DDUMB -DMHE -DNETWORK -DRPATHS -
```

```
DSBACKUP='"\043"' -DSENDMTS -DSMTP -DPEM -DCLOSED  -target sun3 -c  rcc.c
./../pem/h/rcodes.h: 25: NOTOK redefined
make: Fatal error: Don't know how to make target `../pem/lib/libbsafe.a'
Current working directory /home/kronos/steve/pem5.0/mh/uip
*** Error code 1
make: Fatal error: Command failed for target `all'
```

Changed libbsafe.a to libbsafe-mc68020.a in mh/uip/Makefile.

The following occured while running make inst-all in mh:

```
cp ali.man /home/kronos/steve/pem5.0/mh/man/man1/ali.1
cp: /home/kronos/steve/pem5.0/mh/man/man1/ali.1: No such file or directory
*** Error code 1
make: Fatal error: Command failed for target
`/home/kronos/steve/pem5.0/mh/man/man1/ali.1'
Current working directory /home/kronos/steve/pem5.0/mh/doc
*** Error code 1
make: Fatal error: Command failed for target `inst-all'
```

Forgot to create man/man1, did so.

At this point installation was apparently complete.

Appendix B
Problems Encountered with PEM Operation

The following problems were encountered in the operation of PEM:

If you specify integrity option and then mess with a character in the MCC field, PEM properly detects that and then reports the error to you when you try and decrypt it. However, if you change a character in the Certificate and then decrypt it, you get nothing. No error or text. Note - using catrcc to decrypt. I don't think that matters, it is very hard to mess with the messages otherwise.

The header fields "to" and "from" do not accept addresses of the form:

doug (Doug Hardie)

PEM tells you when it encounters such that there is no "to" or "from" field in the message. Not only is the message misleading since the field is there, but it should accept such forms.

At several points in the execution of both mapdir and cai, an input field for Distinguished Name was displayed, but the program skipped over the field and did not allow me to enter any value.

The PEM Installation Manual in section 4.4 indicates that the commands for the .events file can be found in file /lkm/lkm_init.c. That file does not exist in the version 5.0 distribution. Using the "ALL" command generates about 1.5 mbytes for simple commands. We discovered that the "ERRORS" command is much more useful. It only produces the error messages.

An encrypted message was received from another site where the receiver's address in the message certificate did not match the certificate here. Rcc did not report any errors, but did not produce any text output either.

Appendix C
Changes to ELM to Incorporate PEM

diff elm_distribution/src/elm.c elm2.3/src/elm.c

```
2c2
< static char rcsid[] = "@(#)$Id: elm.c,v 4.1 90/04/28 22:42:54 syd Exp $";
---
> static char rcsid[] = "@(#)$Id: elm.c,v 4.1.pem 90/04/28 22:42:54 syd Exp $";
5c5
<  *  The Elm Mail System  -  $Revision: 4.1 $   $State: Exp $
---
>  *  The Elm Mail System  -  $Revision: 4.1.pem $   $State: Exp $
53a54,55
>     char buffer[SLEN];
>
204a207,219
>                     break;
>
>         case 'z' :  PutLine0(LINES-3, strlen("Command: "),
>                                 "Decrypt Mail");
>                     ClearScreen();
>                     sprintf (buffer, "%s -f %s -h %d | catrcc | more",
>                         readmsg,
>                         (folder_type == NON_SPOOL ? cur_folder : cur_tempfolder),
>                         headers[current-1]->index_number);
>                     system (buffer);
>                     PutLine0(LINES,0,"Please Press any key to return.");
>                     (void) ReadCh();
>                     redraw = 1;
```

diff elm_distribution/src/mailmsg2.c elm2.3/src/mailmsg2.c

```
2c2
< static char rcsid[] = "@(#)$Id: mailmsg2.c,v 4.1 90/04/28 22:43:28 syd Exp $";
---
> static char rcsid[] = "@(#)$Id: mailmsg2.c,v 4.1.pem 90/04/28 22:43:28 syd Exp $";
5c5
<  *  The Elm Mail System  -  $Revision: 4.1 $   $State: Exp $
---
>  *  The Elm Mail System  -  $Revision: 4.1.pem $   $State: Exp $
79a80
>         char buffer[SLEN];
85a87
>
364d365
<
493a495
>             Modified to include PEM support
497,498c499,500
<      char ch, buffer[SLEN], fname[SLEN];
<      int x_coord, y_coord;
---
>      char ch, buffer[SLEN], fname[SLEN], tname[SLEN];
>      int x_coord, y_coord, err;
508c510
<          PutLine0(LINES-2,0,
---
>          PutLine0(LINES-3,0,
512c514
<          Centerline(LINES-1,
---
>          Centerline(LINES-2,
513a516,517
>             Centerline (LINES-1,
>               "p)rivacy, i)ntegrity.");
515c519
<          PutLine0(LINES-2, 0, "And now: s");
---
>          PutLine0(LINES-3, 0, "And now: s");
530,531c534,537
<          strcat(buffer, "h)eaders, c)opy file, s)end, or f)orget.");
<          Centerline(LINES-1, buffer);
---
>          strcat(buffer, "h)eaders, c)opy file, s)end, f)orget");
>          Centerline(LINES-2, buffer);
>             Centerline (LINES-1,
>               "p)rivacy, i)ntegrity.");
575a582,631
>
>              case 'i'  : Write_to_screen("Integrity",0);
>                          ClearScreen();
>
```

```
>                       sprintf(fname, "%sELM_pem_%d", temp_dir, getpid());
>                       sprintf(tname, "%sELM_tmp_%d", temp_dir, getpid());
>                       sprintf(buffer, "echo To: '%s' >> %s", to, tname);
>                  system(buffer);
>                  sprintf (buffer,"echo From: %s >> %s", username,tname);
>                  system (buffer);
>                  sprintf (buffer, "echo ------------ >> %s", tname);
>                  system (buffer);
>                  sprintf (buffer,"more %s >> %s", filename, tname);
>                  system (buffer);
>
>                       sprintf(buffer, "catscc -integrity %s > %s",
>                                                       tname,fname);
>                       err=system (buffer);
>                  if (err ==0) {
>                       sprintf (buffer, "mv %s %s", fname, filename);
>                       system (buffer);
>                  }
>                  unlink (fname);
>                  unlink (tname);
>                       break;
>
>             case 'p'  : Write_to_screen("Privacy",0);
>                       ClearScreen();
>
>                       sprintf(fname, "%sELM_pem_%d", temp_dir, getpid());
>                       sprintf(tname, "%sELM_tmp_%d", temp_dir, getpid());
>                       sprintf(buffer, "echo To: '%s' >> %s", to, tname);
>                  system(buffer);
>                  sprintf (buffer,"echo From: %s >> %s", username,tname);
>                  system (buffer);
>                  sprintf (buffer, "echo ------------ >> %s", tname);
>                  system (buffer);
>                  sprintf (buffer,"more %s >> %s", filename, tname);
>                  system (buffer);
>
>                       sprintf(buffer, "catscc -privacy  %s > %s",
>                                                       tname,fname);
>                       err=system (buffer);
>                  if (err ==0) {
>                       sprintf (buffer, "mv %s %s", fname, filename);
>                       system (buffer);
>                  }
>                  unlink (fname);
>                  unlink (tname);
>                       break;
```

diff elm_distribution/src/screen.c elm2.3/src/screen.c

```
2c2
< static char rcsid[] = "@(#)$Id: screen.c,v 4.1 90/04/28 22:44:04 syd Exp $";
---
> static char rcsid[] = "@(#)$Id: screen.c,v 4.1.pem 90/04/28 22:44:04 syd Exp $";
5c5
<  *  The Elm Mail System  -  $Revision: 4.1 $   $State: Exp $
---
>  *  The Elm Mail System  -  $Revision: 4.1.pem $   $State: Exp $
88a89,90
>       Centerline(LINES-4,
>   "z = decrypt Privacy Enhanced Mail");
96a99,100
>     Centerline(LINES-4,
>   "z = decrypt Privacy Enhanced Mail");
```

diff elm_distribution/src/utils.c elm2.3/src/utils.c

```
212,213c212,214
<       MoveCursor(LINES,0);
<       NewLine();
---
> /*    MoveCursor(LINES,0);
>       NewLine();   */
>       ClearScreen();
```

diff elm_distribution/hdrs/defs.h elm2.3/hdrs/defs.h

```
30c30
< # define VERSION         "2.3"                       /* Version number... */
---
> # define VERSION         "2.3.pem"                   /* Version number... */
```