CS 6500 – Network Security
Prof. Krishna Sivalingam
July-Nov. 2017
Due Date: Sunday, **Sep. 10, 11PM**. On-Line Submission via moodle
NO LATE SUBMISSIONS

1 Project description

The objective of this project is to implement the various components of a security-enhanced EMAIL system similar to PGP (Pretty Good Privacy) and GnuPG which is based on the OpenPGP standard. The cryptographic libraries from the **OpenSSL** (http://www.openssl.org) will be used.

1.1 Encrypted Email

The task is: Given an email message from User A to User B, message digest algorithm, encryption algorithm and the public/private key pairs of users, generate a security-enhanced output message that will handle: (a) Confidentiality ONLY, (b) Authentication/Integrity ONLY and (c) Authentication/Integrity and Confidentiality.

We will assume that for a given set of users, the corresponding public/private key pair is locally available. In reality, the public key information for each user is available from some keyserver and the private key is held securely by each user.

For this project, we will assume that Triple-key 3DES, AES, and Blowfish are available. We will assume that only ECB mode is used – hence no need for Initialization Vector, Counter, etc.

The encryption steps are explained in detail in Stallings' textbook. In order to achieve the different levels of security/privacy, the following mechanisms are used:

Confidentiality ONLY (CONF):

Sender: The email message is encrypted with a randomly generated secret message key (or session key) using a symmetric block cipher algorithm. The message key is encrypted using the receiver's public key and **pre-pended** to the message. The [encrypted key, message] is then written into a file that can be sent using traditional email.

In essence, given input text M from user A to user B, the transmitted message C is:

$$C = \left[E_{KU_b}(K_s, IV_s) \parallel E_{K_s}^{des3}(M)\right]$$

where K_s is the session key (and any needed initialization vector, IV_s) and E_{KU_b} is receiver's public key.

Receiver: The receiver uses its private key to decrypt the secret message key from the message file. This secret key is then fed to the decryption algorithm on the remainder of the message file to extract the plain-text message.

Authentication/Integrity ONLY (AUIN):

Sender: A message digest or hash of the given plain-text message is generated. The digest/hash is then encrypted using the sender's private key. The [encrypted hash, message] is then written into a file that can be sent using traditional email.

In essence, given input text M from user A to user B, the transmitted message C is:

$$C = \left[E_{KR_a}\left(H^{sha1}(M)\right) \parallel M\right]$$

where E_{KR_a} is sender's private key.

Receiver: The receiver reads the encrypted hash from the message file and decrypts using the sender's public key to generate the hash. The digest/hash of the remainder of the message file is computed and compared to the decrypted hash. If they match, success is declared.

Confidentiality and Authentication/Integrity (COAI):

Sender: A message digest or hash of the given plain-text message is generated. The digest/hash is then encrypted using the sender's private key. The [encrypted hash, message] is then encrypted with a randomly generated secret message key (or session key) using a symmetric block cipher algorithm. The message key is encrypted using the receiver's public key and **pre-pended** to the message. The [encrypted key, encrypted hash, message] is then written into a file that can be sent using traditional email.

In essence, given input text M from user A to user B, the transmitted message C is:

$$C = \left[E_{KU_b}(K_s, IV_s) \parallel E_{K_s}^{des3}\left(E_{KR_a}\left(H^{sha1}(M)\right) \parallel M\right)\right]$$

where K_s is the session key (and any needed initialization vector, IV_s), E_{KR_a} is sender's private key and E_{KU_b} is receiver's public key.

Receiver: The receiver performs the reverse operation to verify the sender, integrity and to extract the plain-text message.

1.2 User Interface

The following are the necessary commands:

▷ ./proj2 CreateKeys UserNameListFile

The UserNameListFile will contain a set of users, one per line. For each user, randomly generate an RSA key pair and store private key and public key in user_priv.txt and user_pub.txt respectively.

▷ ./proj2 CreateMail SecType Sender Receiver EmailInputFile EmailOutputFile DigestAlg EncryAlg

SecType: CONF, AUIN, COAI are three possible string values for the three cases listed earlier.

Sender/Receiver are sender and recipient of this message.

EmailInputFile contains the input plain-text file (in ASCII format)

EmailOutputFile contains the output of the encryption algorithms (in binary format)

DigestAlg: One of: sha1, sha256

EncryAlg: bf-ecb, des3, aes-128-ecb

▷ ./proj2 ReadMail SecType Sender Receiver SecureInputFile PlainTextOutputFile DigestAlg EncryAlg

Similar to above.

1.3 Output File Format:

CONF: Line 1: Encrypted Secret Key
Line 2 onwards: Encrypted Data

AUIN: Line 1: Encrypted Hash/Digest
Line 2 onwards: Plaintext Data

COAI: Line 1: Encrypted Secret Key
Line 2: Encrypted [Hash/Digest+MessageData]

2 Starting Point

2.1 System Calls

At the least, the following system calls will be required . On Linux, *man command* or *info command* will provide you with relevant information. You can run 'man evp', 'man rsa', 'man pem' for lots of additional information. You should know or learn how to read function interface descriptions from man pages and use them correctly.

Note: By default, the crypto functions produce binary output. You must choose base64 encoded output format.

3 Sample Session

Assume that you have created the files lab2.c and the corresponding executables in your LAB2 directory.

```
% cd LAB2
% make proj2
...
% pico Usernames.txt
... file will contain a set of 10-15 users
```

```
% ./proj2 CreateKeys Usernames.txt
% ls -l *_priv.txt *_pub.txt
% pico Mail-sample.txt
... file will contain input text message.
% ./proj2 CreateMail COAI alice bob Mail-sample.txt Mail-out.txt sha1 bf-ecb
.. Encrypted File is created.
% ./proj2 CreateMail COAI alice bob Mail-out.txt Mail-decrypt.txt sha1 bf-ecb
.. Decryption operation
% diff Mail-sample.txt Mail-decrypt.txt
.. Any differences?
```

4 What to Submit

Name your project directory as LAB2 (Note: ALL UPPERCASE)

Once you are ready to submit, change directory to the directory above LAB2, and tar all files in the directory with the command:

```
tar czf Lab2-RollNo.tgz LAB2
```

Then, submit it online on Moodle.

The directory should contain the following files:

- ▷ Source Files
- ▷ Makefile
  Typing command 'make' at the UNIX command prompt, should generate all the required executables.
- ▷ A Script file obtained by running UNIX command *script* which will record the way you have finally tested your program.
- ▷ a README file containing instructions to compile, run and test your program. The README should document known error cases and weaknesses with the program.
- ▷ a COMMENTS file which describes your experience with the project, suggestions for change, and anything else you may wish to say regarding this project. This is your opportunity for feedback, and will be very helpful.

5 Help

1. WARNING ABOUT ACADEMIC DISHONESTY: Do not share or discuss your work with anyone else. The work YOU submit SHOULD be the result of YOUR efforts. Any violation of this policy will result in an automatic ZERO on the project, a potential F in the course, and other academic action.
2. Ask questions EARLY. Do not wait until the week before. This project is quite time-consuming.

3. Implement the solutions, step by step. Trying to write the entire program in one shot, and compiling the program will lead to frustration, more than anything else.

   For example, implement the message digest first, then try encryption with secret key, then try encryption with public/private key, etc.

4. You can always verify the intermediate outputs using the openssl command line interface that was demonstrated in class.

6 Grading

- ▷ CONF: 30 points
- ▷ AUIN: 30 points
- ▷ COAI: 40 points

No README/COMMENTS: -5 points; No Script File: -10 points; Incomplete Compilation: -10 points