TCSS 487 CRYPTOGRAPHY

Cryptographic Library & App
Project Report

Members: Hari Kuduva, Matthew Chan

```
>>>>
Welcome to our app!

Here are your options:
        1) Find a cryptographic hash of any given file or input text
        2) Encrypt or Decrypt a given file symmetrically under a given passphrase
        3) Generate an elliptic key pair file from a given passphrase
        4) Encrypt / Decrypt a data file under a given elliptic public key file.
        5) Sign / Verify a given data file
        6) Exit

Please enter a valid option (invalid option numbers are ignored):
```

Figure 0: The main menu of the app

Objective: Implement (in Java) a library and an app for asymmetric encryption and digital signatures at the 256-bit security level.

This app offers five different options:

1. Compute a plain cryptographic hash of a given file or input text
2. Encrypt a given data file symmetrically under a given passphrase
   Decrypt a given symmetric cryptogram under a given passphrase
3. Generate an elliptic key pair from a given passphrase and write the public key to a file
4. Encrypt a data file under a given elliptic public key file
   Decrypt a given elliptic-encrypted file from a given password
5. Sign a given file from a given password and write the signature to a file
   Verify a given data file and its signature file under a given public key file

Option 1 : Cryptographic Hash

```
Please enter a valid option (invalid option numbers are ignored): 1
        1) Find a cryptographic hash of a file
        2) Find a cryptographic hash of any input text
        3) Back

Please enter a valid option (invalid option numbers are ignored):
```

Figure 1: The menu of option 1

There are two suboptions for option 1:

1. Find a cryptographic hash of a file
2. Find a cryptographic hash of any input text

Suboption 1.1: User can choose a file and obtain its cryptographic hash.

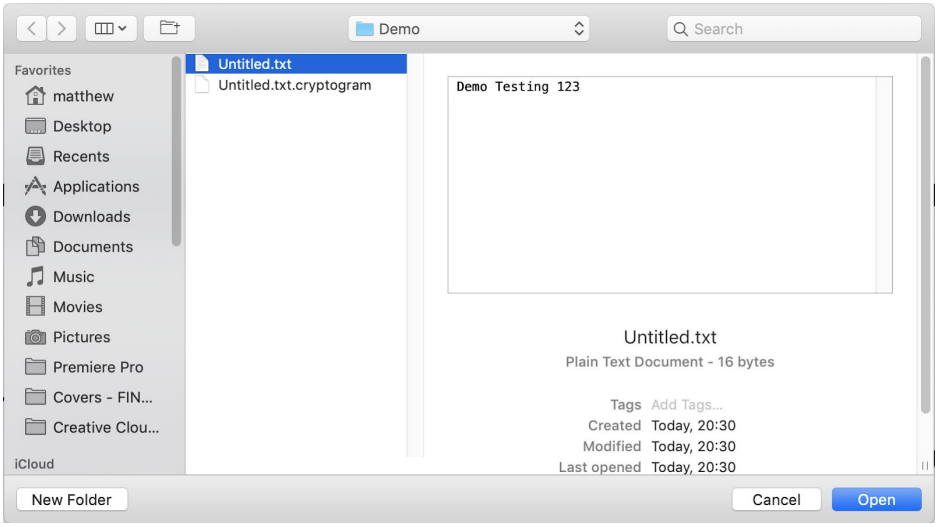


Figure 1.1: The file chooser for choosing a file

Suboption 1.2: User can input any text and obtain its cryptographic hash.

```
Please enter a valid option (invalid option numbers are ignored): 1
        1) Find a cryptographic hash of a file
        2) Find a cryptographic hash of any input text
        3) Back

Please enter a valid option (invalid option numbers are ignored): 2
Please enter the string you want to hash:
Testing
Done! Your text "Testing" hashed to ->  6124d1cde99a48a77384842c3cfa182577884928159d
```

Figure 1.2: The process of hashing the input text

Option 2 : Symmetrical Encryption / Decryption

```
Please enter a valid option (invalid option numbers are ignored): 2
        1) Encrypt a given data file symmetrically under a given passphrase
        2) Decrypt a given symmetric cryptogram under a given passphrase
        3) Back

Please enter a valid option (invalid option numbers are ignored):
```

Figure 2: The menu of option 2

There are two suboptions for option 2:

1. Encrypt a given data file symmetrically under a given passphrase
2. Decrypt a given symmetric cryptogram under a given passphrase

Suboption 2.1: Encrypt a given data file under a passphrase

```
Please enter a valid option (invalid option numbers are ignored): 2
        1) Encrypt a given data file symmetrically under a given passphrase
        2) Decrypt a given symmetric cryptogram under a given passphrase
        3) Back

Please enter a valid option (invalid option numbers are ignored): 1

        Please enter a password / passphrase: abc
```

Figure 2.1: The process of suboption 2.1

User first enters a password to the console, then a file chooser will appear after the user hits the 'Enter' key. The chosen file will be encrypted based on the password entered. The encrypted file will be generated with the file extension ".cryptogram" in the same directory as the source file afterwards.

Suboption 2.2: Decrypt a given data file under a passphrase

User first enters a password to the console, then a file chooser will appear after the user hits the 'Enter' key. The chosen file will be encrypted based on the password entered. If the password entered is incorrect, the program will reject the decryption process.

```
Please enter a valid option (invalid option numbers are ignored): 2
        1) Encrypt a given data file symmetrically under a given passphrase
        2) Decrypt a given symmetric cryptogram under a given passphrase
        3) Back

Please enter a valid option (invalid option numbers are ignored): 2

        Please enter a password / passphrase: abcd
t' does not equal t. Rejected. Passwords must match.
```

Figure 2.2: The program rejecting the decryption process due to a incorrect password

Option 3 : Generate an Elliptic Key File

```
Please enter a valid option (invalid option numbers are ignored): 3

        Please enter a password / passphrase: testing123

Please select a destination directory for public key output.
```

Figure 3: The menu of option 3

User first enters a password to the console, then a directory chooser will appear after the user hits the 'Enter' key. The public key output file will be generated after choosing the destination directory. After the file being generated, the private key along with the public key X and public key Y will be printed out to the console as well.

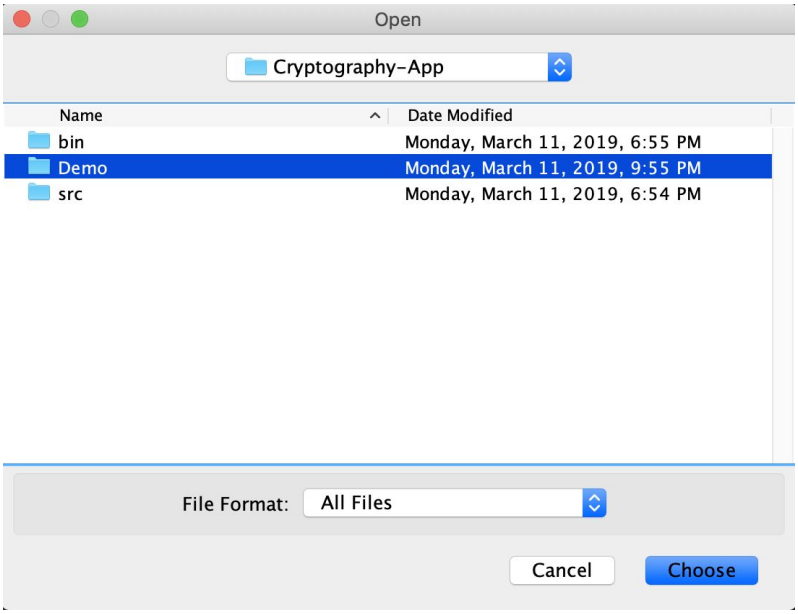


Figure 3.1: The directory chooser for choosing the destination directory

```
Private Key: 869855985818850683792669774229950626139838543785315134462629735670114342077297513893258665553173789900782273384666442767260687868728
Public Key X : 430704156049402307730599059719807948896211886786281057096578301032151934089732712921105508288115087614279859440049502527566769838
Public Key Y : 539337827336313659153198054115706260290745540715576810915026395022348798592433451375156775346219629623425091488897481094471162823

Your public key file has been saved to the destination: /Users/matthew/eclipse-workspace/Cryptography-App/Demo/public_key_file_password=testing123
```

Figure 3.2: The console output after the key file being generated

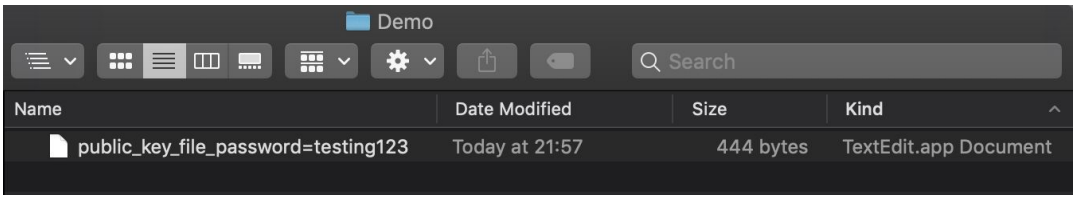


Figure 3.3: The output public key file

Option 4 : Encrypt / Decrypt under Public Key File

```
Please enter a valid option (invalid option numbers are ignored): 4
        1) Encrypt a data file with a public key file
        2) Decrypt a file .CryptogramECC file with a password
        3) Back

Please enter a valid option (invalid option numbers are ignored):
```

Figure 4: The menu of option 4

There are two suboptions for option 4:

1. Encrypt a data file with a public key file
2. Decrypt a file .CryptogramECC file with a password

Suboption 4.1: Encrypt a data file with a public key file

User has to use the public key file generated from option 3 to proceed. The program will prompt the user to choose the public key file first, and then the actual file which the user wants to encrypt.

```
Please enter a valid option (invalid option numbers are ignored): 4
        1) Encrypt a data file with a public key file
        2) Decrypt a file .CryptogramECC file with a password
        3) Back

Please enter a valid option (invalid option numbers are ignored): 1

Please select a public key file you generated from part 3


Please select a file to encrypt

File Encrypted: 44656D6F2054657374696E67
```

Figure 4.1: The process of suboption 4.1

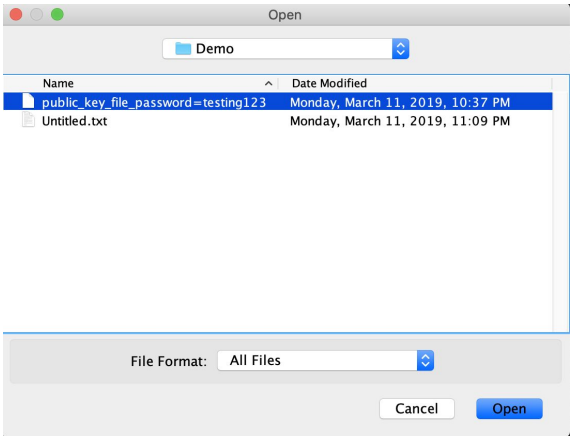


Figure 4.1.1: Choosing the public key file

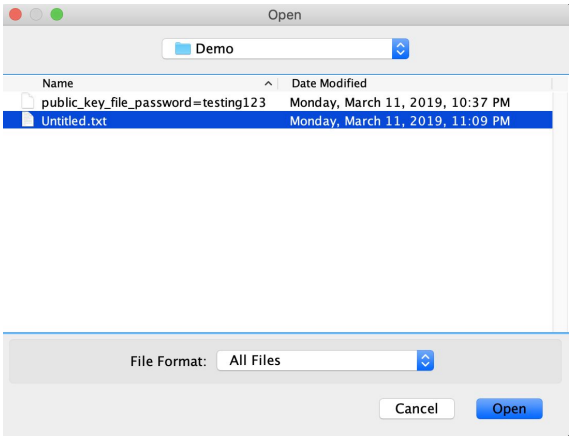


Figure 4.1.2: Choosing the target file

Suboption 4.2: Decrypt a file .CryptogramECC file with a password

User can decrypt a file which has been encrypted from suboption 4.1. The program will first ask for the password from the user. The password has to be matched to the one which is associated to the public key file which being generated from option 3. Otherwise, the program will reject the decryption process.

```
Please enter a valid option (invalid option numbers are ignored): 4
        1) Encrypt a data file with a public key file
        2) Decrypt a file .CryptogramECC file with a password
        3) Back

Please enter a valid option (invalid option numbers are ignored): 2

        Please enter a password / passphrase: testing123
Please select a .CryptogramECC file to decrypt

File decrypted! Contents: 44656D6F2054657374696E67
```

Figure 4.2: The process of suboption 4.2

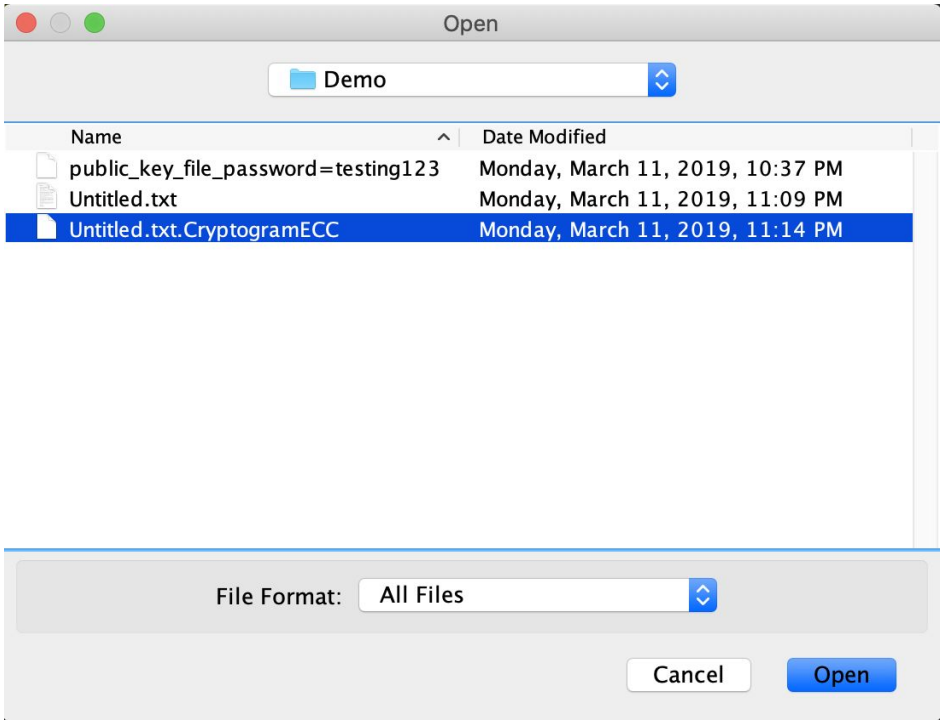


Figure 4.2.1: User choosing the file to be decrypted

Option 5 : Sign / Verify a File

```
Please enter a valid option (invalid option numbers are ignored): 5
    1) Sign a given file from a given password and write the signature to a file
    2) Verify a given data file and its signature file under a given public key file
    3) Back

Please enter a valid option (invalid option numbers are ignored):
```

Figure 5: The menu of option 5

There are two suboptions for option 5:

1. Sign a given file from a password and write to a signature file
2. Verify a data file given a signature and public key file

Suboption 5.1: Sign a file from a password and generate a signature file

```
Please enter a valid option (invalid option numbers are ignored): 5
    1) Sign a given file from a given password and write the signature to a file
    2) Verify a given data file and its signature file under a given public key file
    3) Back

Please enter a valid option (invalid option numbers are ignored): 1

    Please enter a password / passphrase: abc

Please select a data file

A signature file has been generated!
```

Figure 5.1: The process of suboption 5.1

User will first be asked to enter a password for the signature file. After hitting Enter the console will prompt the user to select a data file. A File selector will appear.

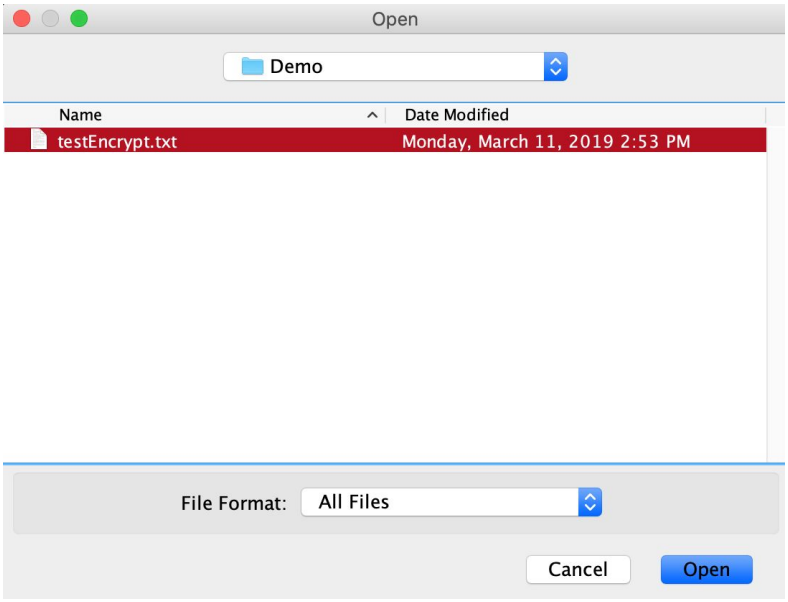


Figure 5.1.1: The file chooser prompting the user for an input file

After hitting “Open” a signature file will be automatically generating in the same location as the source file

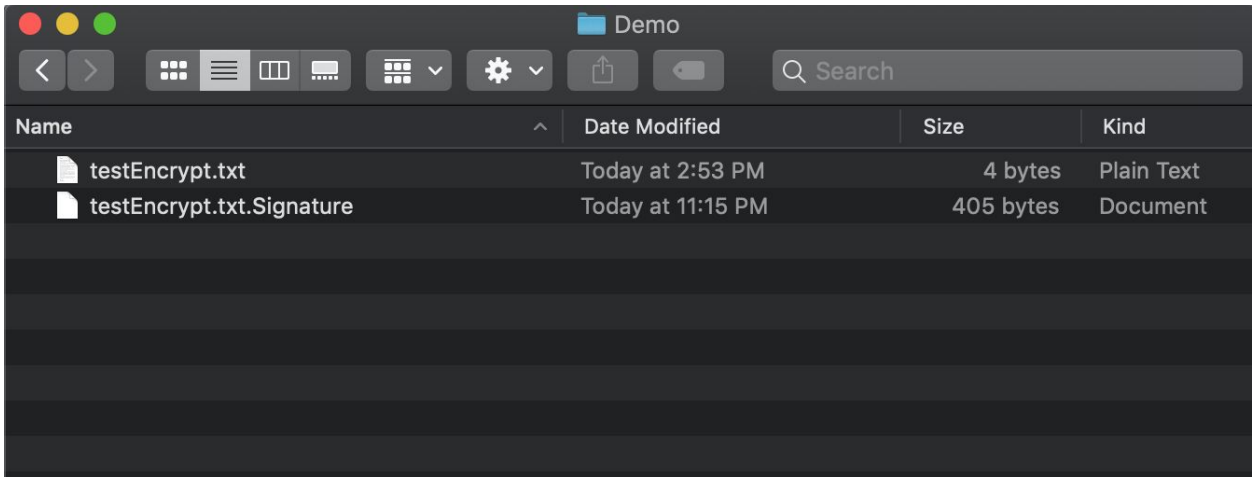


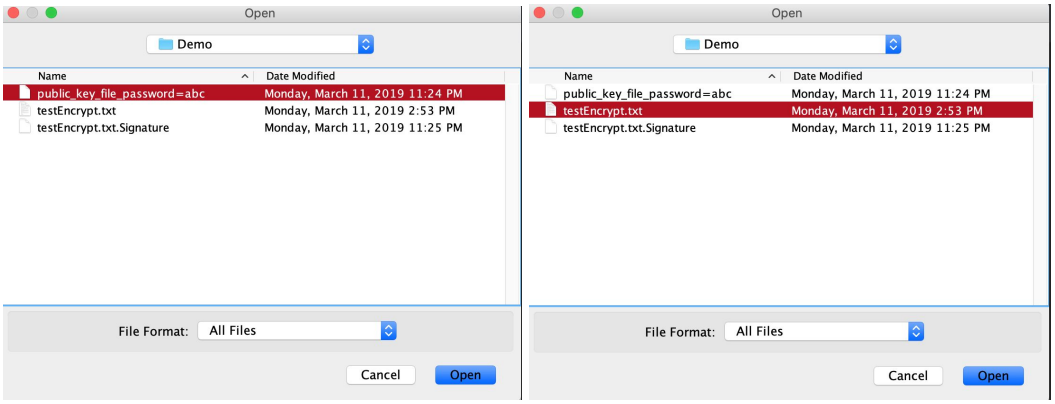
Figure 5.1.2: The signature file outputted into the directory of the source file

Suboption 5.2: Verify a given signature file

```
Please enter a valid option (invalid option numbers are ignored): 5
    1) Sign a given file from a given password and write the signature to a file
    2) Verify a given data file and its signature file under a given public key file
    3) Back

Please enter a valid option (invalid option numbers are ignored): 2

Please select a public key file you generated from part 3


Please select a data file


Please select a Signature file

Illegitimate signature
```

Figure 5.2: The verification process for signed files

For this part, the user will be directed to select three files: The public key file, the data source file, and the signature file that was generated from during the part 5 option 1. Three file choosers will pop up one by one prompting the user to select the corresponding files mentioned above.



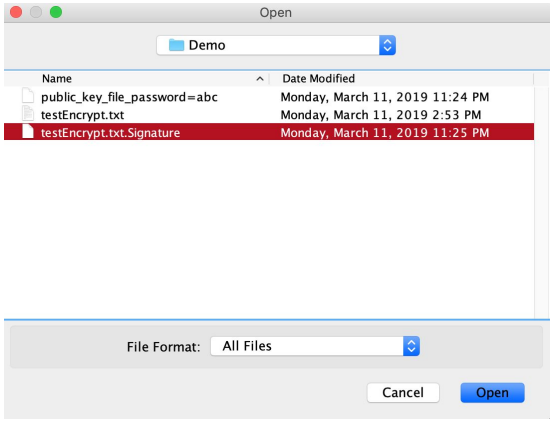


Figure 5.2.1: Three file choosers showing up 1 by 1 to make the user select the files

After all the files have been selected, the program will output whether the verification process passed or failed.

** BUG: Our verification always outputs “Illegitimate Signature” **

Unresolved Bugs:

1. Signature verification always outputs “Illegitimate Signature” even if the verification passed.
2. JFileChooser occasionally does not show up. Simply rerun the program if this happens.

Work Cited:

1. SHAKE java file was written by Paulo Barretto and Markku-Juhani Sarrinen whose original Model.SHAKE implementation was in C.