CrypDex

DOCUMENTATION

Samarth Bharti

Vaibhav Chauhan

Gaurav Sinha

CONTENTS

1. CERTIFICATE

2. ACKNOWLEDGEMENT

3. INTRODUCTION

3.1. PROJECT OVERVIEW

3.2. PROJECT DELIVERABLES

4. PRODUCT DESCRIPTION

4.1. ASSUMPTIONS

5. REQUIREMENTS

5.1. FUNCTIONAL REQUIREMENTS

5.2. USER INTERFACE REQUIREMENTS

5.3. PERFORMANCE

6. CORE COMPONENTS

7. DESIGN APPROACH

8. ALGORITHM EXPLANATION

9. PACKAGES USED

10. HOW-TO RUN

CERTIFICATE

THIS IS TO CERTIFY THAT THE DOCUMENTATION ENTITLED “CrypDex” IS AN AUTHENTIC WORK CARRIED OUT BY THE INTERNS AT CENTRE FOR ANALYTICAL FINANCE, INDIAN SCHOOL OF BUSINESS UNDER MY GUIDANCE. THE MATTER EMBODIED IN THIS PROJECT HAS NOT BEEN SUBMITTED EARLIER FOR AWARD OF ANY DEGREE OR DIPLOMA TO THE BEST OF MY KNOWLEDGE AND BELIEF.

SIGNATURE :

Mr. **Padmanabhan Balasubramanian**

ACKNOWLEDGEMENT

We would like to extend our gratitude to our project guide Mr. **Padmanabhan Balasubramanian,** for his expert guidance and constant encouragement in the course of the project. We would also like to acknowledge all the other mentors and our colleagues for their valuable suggestions and comments. This project is a group institution, with everyone playing essential roles through the course of the project.

3. INTRODUCTION

This project is designed to allow for encryption of confidential files containing sensitive information. Through the course of this document we will take you through the various key elements that comprise the project. The key functionality of this software “CrypDex” revolves around the open source library ‘RSA’ and ‘PyCrypto’ on the python environment. The essence of this algorithm relies on the very well-known problem in the world of computer science, known as P=NP. An introduction to the algorithm used for encryption can be found later in the document (Section 8).

**3.1 PROJECT OVERVIEW**

This product is a desktop application with the gist revolving around providing encrypting and decrypting functionality to users who wish to secure sensitive documents. The main intended audience is banking firms who deal with highly confidential information in large volumes. The project script as of now is running locally, however we intend to use cloud following this and run our scripts on the externally, as well as use storage facilities from a cloud service available at a premium.

**3.2 PROJECT DELIVERABLES**

The project incorporates a desktop application whose functionality is explained above. We also would be using a third party cloud services. The user can upload their files; select the data columns they wish to encrypt; select the folder to which they intend to save the new encrypted file as well as select a name for the new file. A new encrypted file is generated to maintain the integrity of the original document. Along with the encrypted file, two new files are also generated, containing the public key and the private key respectively.

4. PRODUCT DESCRIPTION

The product is a software that allows user to byte encrypt confidential data pertinent to data columns that might be confidential. The user can select any file (Loan or Savings) data and also define the name of the target file, and location. Accordingly, an encrypted file, a public key and private key files are generated. The public keys can be distributed to multiple clients who can use the software to encrypt files; however decryption which is encapsulated by the private key can only be done by an administrator.

**4.1 ASSUMPTIONS**

It is assumed that the user has a populated data file, has a functioning machine with minimal computing powers. The user need not have python interpreter installed on their machines in case of windows and linux based operating systems. We are still experimenting with how to extend this usability to the MacOS, and will be updating this document soon on the basis of our conclusions.

5. REQUIREMENTS

**5.1 FUNCTIONAL REQUIREMENTS**

**1.** **ADMIN**

a. Have access to the data file.

b. Can generate the public and private key.

c. Distribute public keys.

d. Decrypt files using the Private Key based on:

i. Public key provided to the user.

ii. The file encrypted by user.

iii. User ID if needed for authentication.

**2.** **USER**

a. Can encrypt the data files using a provided public key.

**5.2 USER INTERFACE REQUIREMENTS**

Admin should be provided with the following features:

1. To generate the public and private keys.

2. To select the data file for encrypting/decrypting.

3. To select the columns in the data to be encrypted/decrypted.

4. To generate the encrypted/decrypted data file.

User should be provided with the following features:

1. To select the data file for encrypting using the provided public key.

2. To select the columns in the data that the user wants to encrypt.

3. To generate the encrypted data file.

**1.** **User Interfaces**

**a.** **Home Page(Encryption/ Encryption using a provided public key/Decryption)**

1. The interface would contain three options:
   1. Encryption
   2. Encryption using a provided public key
   3. Decryption

The processes are explained below.

**b.** **Encryption Process**

1. This process is meant for the admin
2. A fresh pair of Public/Private Keys is generated in this case.
3. Workflow:
   1. The admin selects the data file to be encrypted.
   2. The admin selects the columns in the data file that need to be encrypted.
   3. The admin selects the location where he wants to save the encrypted data file and the public and private key files.
   4. The admin inputs the desired name for the encrypted data file.
   5. The admin clicks ‘Next’ to finish the process.

**c.** **Encryption using a provided public key process**

1. This process is meant for the user.
2. The user requires a provided public key to encrypt a data file.
3. Workflow:
   1. The user selects the data file to be encrypted.
   2. The user selects the columns in the data file that need to be encrypted.
   3. The selects the public key file.
   4. The user selects the location where he wants to save the encrypted data file.
   5. The user inputs the desired name for the encrypted data file.
   6. The user clicks ‘Next’ to finish the process.

**d.** **Decryption Process**

1. This process is meant for the admin
2. Only the admin has the authority and access to decrypt an encrypted file
3. Workflow:
   1. The admin selects the data file to be decrypted.
   2. The admin selects the columns in the data file that need to be decrypted.
   3. The admin selects the private key file.
   4. The admin selects the location where he wants to save the decrypted data file.
   5. The admin inputs the desired name for the decrypted data file.
   6. The admin clicks ‘Next’ to finish the process.

**5.3 PERFORMANCE**

**1.** **Scalability**

A cap to regress tests was set to a file containing **1 million rows of data**. Successfully encrypting 1 million rows of data for four data columns. Note, run time may vary depending on the machine. On a **Core i5 4th generation processor with 8Gb RAM**, we were able to perform this rigorous test in a cap time of **10 mins**, and it varied every time depending on the amount of memory and processing power the application could access at that given time.

6. CORE COMPONENTS

1. **HOME PAGE**
   1. Contains buttons for encryption and decryption.
2. **ENCRYPTION PAGE**Contains the following subpages for the encryption process:
   1. Page1:
      1. Option for browsing and selecting the data file.
      2. Option for selecting the columns to be encrypted.
   2. Page2:
      1. Option for encrypting with pre-existing public key or generating a new pair of public-private key.
      2. Option to select the location of the pre-existing public key.
   3. Page3:
      1. Option for selecting the location for the encrypted file and the public & private keys.
      2. Option for inputting the name of the encrypted data file.
      3. Button to start the encryption process.
      4. Button to finish the process.
3. **DECRYPTIONPAGE**Contains the following subpages for the process:
   1. Page1:
      1. Option for browsing and selecting the data file.
      2. Option for selecting the columns to be decrypted.
      3. Option for browsing and selecting the private key.
   2. Page2:
      1. Option for selecting the location for the decrypted file.
      2. Option for inputting the name of the decrypted data file.
      3. Button to start the decryption process.
      4. Button to finish the process.

7. DESIGN APPROACH

We have tried to make the software as user friendly as we could, with very clear demarcations and status bars indicating the user how to proceed on a particular page.

Also, the processing may go on in the background, and the user may feel free to carry on their regular tasks while they are at it.

8. ALGORITHM EXPLANATION

RSA is one of the first practical public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem. A more detailed explanation can be found [here](https://en.wikipedia.org/wiki/RSA_(cryptosystem)).

Principally, lending to one of the most popular problem in the world of computer science known as P=NP, this algorithm relies on the principle that decryption of a file will take non polynomial time or non deterministic polynomial time, and n the other hand it will take polynomial time to run if a private key from the public/private key pair is known.

The keys for the RSA algorithm are generated the following way:

I. Choose two distinct prime numbers p and q. For security purposes, the integers p and q should be chosen at random, and should be similar in magnitude but 'differ in length by a few digits'[2] to make factoring harder.

Ii. Prime integers can be efficiently found using a primality test. Compute n = pq. n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.

Iii. Compute λ(n) = lcm(λ(p), λ(q)) = lcm(p − 1, q − 1), where λ is Carmichael's totient function. This value is kept private. Choose an integer e such that 1 < e < λ(n) and gcd(e, λ(n)) = 1; i.e., e and λ(n) are coprime.

Iv. Determine d as d ≡ e−1 (mod λ(n)); i.e., d is the modular multiplicative inverse of e (modulo λ(n)).

-This is more clearly stated as: solve for d given d⋅e ≡ 1 (mod λ(n)).

-e having a short bit-length and small Hamming weight results in more efficient encryption most commonly 216 + 1 = 65,537.

-However, much smaller values of e (such as 3) have been shown to be less secure in some settings.[14] e is released as the public key exponent.

-d is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and λ(n) must also be kept secret because they can be used to calculate d.

9. PACKAGES USED

The following libraries have been used:

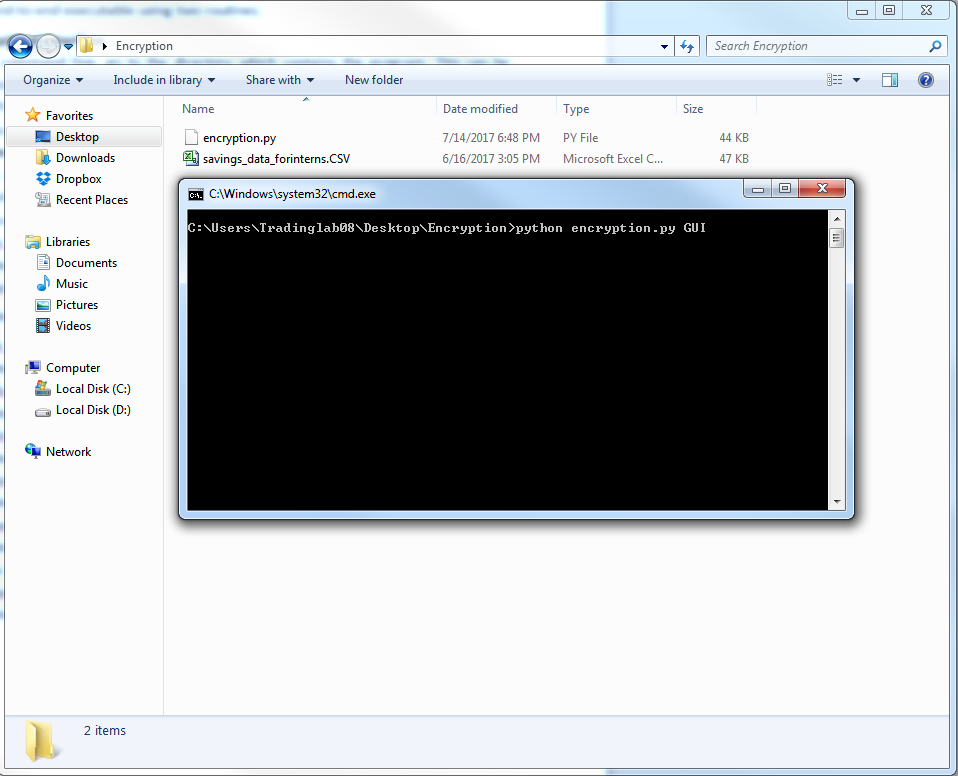
1. [Tkinter](https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwiB4P-3pcnUAhWLo48KHUzvDmQQFggsMAE&url=https%3A%2F%2Fdocs.python.org%2F2%2Flibrary%2Ftkinter.html&usg=AFQjCNHIEqMGV-hkIOHvaZMlEEVrHYE0wg)
2. [PyCrypto](https://pypi.python.org/pypi/pycrypto)
3. [RSA](https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwifttrMpcnUAhVIMI8KHVidDs8QFgglMAA&url=https%3A%2F%2Fpypi.python.org%2Fpypi%2Frsa&usg=AFQjCNG-fVDPMOKGoIiLrQc2tBiBDYjQGg)
4. [Pandas](http://pandas.pydata.org/)
5. [OS](https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwjPusDjpcnUAhUJtI8KHX57CygQFggsMAE&url=https%3A%2F%2Fdocs.python.org%2F3%2Flibrary%2Fos.html&usg=AFQjCNESSwoWQM2MC2qj5Ajo0kfO2LJkPQ)

10. HOW-TO RUN

The program is end-to-end executable using two routines.

1. **Using Graphical Interface**

From the command line, go to the directory which contains the program. This can be done by either navigating to the folder containing the program, or by opening the folder on your PC -> Shift + mouse right click -> Open command window here. Next type python encryption.py GUI and hit enter.



1. **Using Command line interface**

Open the command window and navigate to the folder containing the program in a similar fashion as explained above.

The command line requires the following values to be passed as arguments:

1. CL command

Indicates that program needs to be executed using command line mode.

1. enc\_or\_dec

Enter 1 for encryption, 2 for decryption.

1. loan\_or\_savings

Enter 1 for loan file, 2 for savings file.

1. file\_location

Enter location of the file that needs to be encrypted / decrypted.

1. column\_no

Enter total number of columns that need to be encrypted.

1. existing\_key

Enter 1 if you wish to use pre-existing key, 0 otherwise.

1. existing\_key\_location

Enter Location of existing key if exists, ‘None’ otherwise

1. save\_location

Enter the path of the folder where you wish to store the encrypted / decrypted file along with the keys generated.

1. save\_file\_name

Enter name of the file that is saved.

All these arguments mentioned above should be entered in the same order.

Assume I have a file *savings\_data\_forinterns.CSV*, in the location *C:\Users\Tradinglab08\Desktop\Encryption* and I wish to encrypt it, with a newly generated pair of key. This encrypted file has to be saved back to the same folder with the name *encrypted\_now.CSV.* Here is how that can be done:

