RSA Sandbox User Manual

Created by Aron Schwartz and Professor John Acken  
Portland State University | Winter 2020 | ECE Department

**Table of Contents**

**1.0 Overview** -------------------------------------------------------------------------------- 3   
 1.1.0 Running the Tool ------------------------------------------------------------- 3  
 1.2.0 Data and Profile Folders ----------------------------------------------------- 4

**2.0 System Data Management** ---------------------------------------------------------- 5

2.1.0 Prime Number Management --------------------------------------------------- 5

2.2.0 Septuple Objects ----------------------------------------------------------------- 6

2.2.1 The Active Septuple ---------------------------------------------------------- 7

2.2.2 Septuple Creation --------------------------------------------------------------8  
 2.2.3 Septuple Creation from Primes-----------------------------------------------9  
2.3.0 Key Management ---------------------------------------------------------------- 9

2.4.0 Displaying System Data ------------------------------------------------------- 11

2.5.0 Saving and Loading System Data--------------------------------------------- 12

**3.0 Encrypting Plaintext and Strings** ------------------------------------------------- 13

3.1.0 Encrypting Plaintext Files------------------------------------------------------ 13  
 3.2.0 Encrypting input strings-------------------------------------------------------- 14

**4.0 Fixed Point Analysis** ----------------------------------------------------------------- 15

4.1.0 Active Septuple Analysis and Septuple Comparison----------------------- 15  
 4.2.0 Key Analysis and Transparency Profile Generation----------------------- 17

**5.0 Help Menu** ---------------------------------------------------------------------- ------ 18

**1.0 RSA Sandbox Overview**

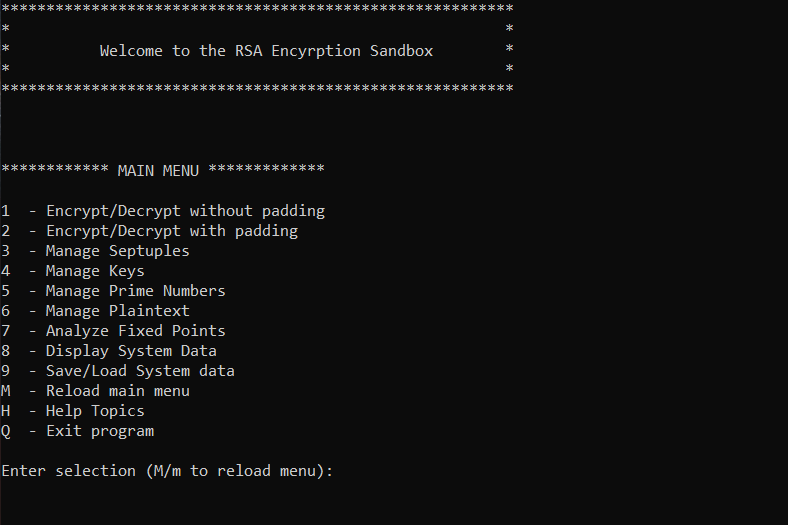
The RSA Sandbox is a research tool that enables users to explore and experiment with the RSA encryption algorithm. The program allows a user to create RSA encryption “septuples”, generate prime numbers, create encryption keys, and combine the three to search for fixed point patterns. In addition, users can encrypt plaintext to file, output results in comma separated format, and save all system data to a user profile to pick up again later.

**1.1 Running the Tool**

The tool is written in Python 3, and is intended to be run from the windows command line. Prior to running the program, all source code can be downloaded from the following repository:

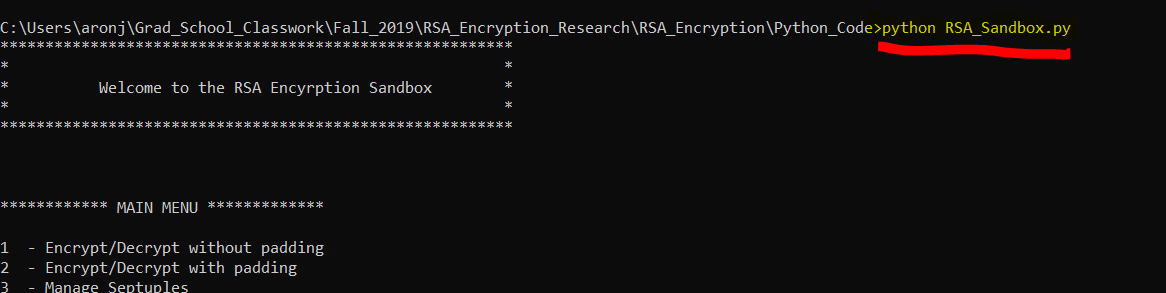
Link to Repository: <https://github.com/aronjschwartz/RSA_Encryption.git>

A screenshot of the main menu of the program can be seen below:



**Figure 1:** The main menu is loaded upon initialization of RSA\_Sandbox.py

To run the program, navigate to the directory housing the repository from the command line and run the RSA\_Sandbox.py file. A screenshot illustrating initialization of the program can be seen below.



**Figure 2:** Run the program by initializing RSA\_Sandbox.py from the command line

**1.2 Data and Profile Folders**

When the program is initialized, it will check for the existence of four folders in its working directory and create them if they do not exist. They are as follows:

1. **Profiles** – Contains user profile sub-folders containing system data files in .csv format. Profile data can be created and loaded from the system data management menu.
2. **Results** – Contains output .csv files resulting from fixed point analysis
3. **Plaintext** – Contains plaintext files. Users can add text files to this folder to encrypt them
4. **Ciphertext** – Contains ciphertext output resulting from encrypting plaintext files

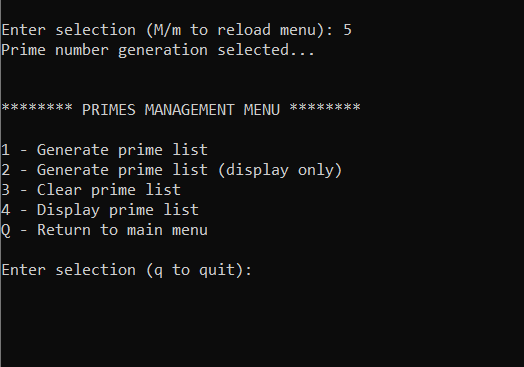
The following sections provide a detailed functionality overview for all components of the RSA sandbox program.

**2.0 System Data Management**

The RSA Sandbox allows creation and manipulation of septuples, encryption keys, and prime numbers. A description of each of these three elements, their respective relationships, and the mechanisms for properly utilizing them can be seen in the following sections.

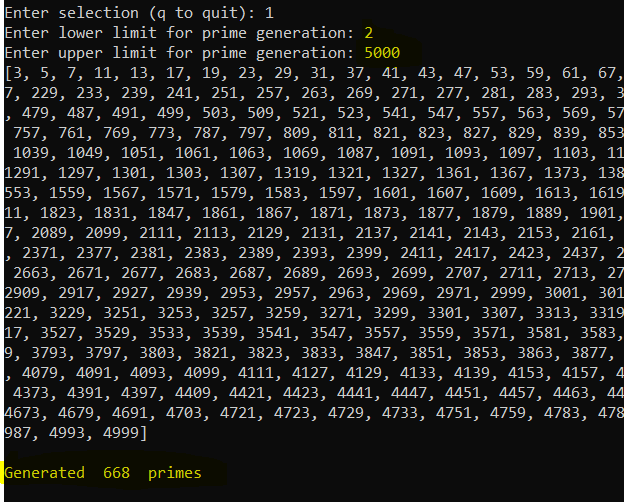
**2.1.0 Prime Number Management**

The program maintains a list of prime numbers that can be used to generate keys and septuples. The prime number management menu can be accessed by pressing ‘5’ from the main menu, as shown below.



**Figure 3:** The prime number management menu allows creation of internal primes list

Both options ‘1’ and ‘2’ will create prime numbers, however option ‘2’ will only display the list and will not append it to the internal data. To generate primes, the user simply inputs a starting and ending value. The program will generate a list of every prime number between the two provided values. The following screenshot shows an example process of generating all prime numbers between 2 and 5000.



**Figure 4:** 668 primes are generated for the input range 2 to 5000

As shown, the system will generate the primes within the given input range. A final message depicting the total number of primes created is also generated, in this case 668 values. As with the septuple menu, other options exist to simply view the prime list or to clear it away entirely.

**2.2.0 Septuple Objects**

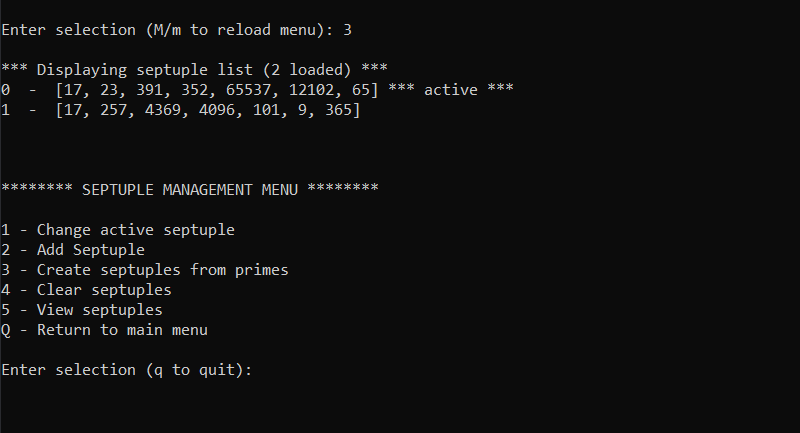
Performing encryption in the RSA Sandbox is accomplished by the creation of septuple objects, a python object that houses the 7 parameters and internal functionality needed to perform RSA encryption and decryption. The 7 parameters of a septuple and a brief description can be seen below.

* **P and Q:** Two initially chosen prime numbers which govern other parameter choices
* **N:** Product of P and Q
* **Totient**: Product of (P-1)\*(Q-1)
* **E:** Encryption exponent, part of the public encryption key **(E, N)**
* **D:** Decryption exponent, part of the private decryption key **(D, N)**
* **K:** Internal parameter related to choosing D

For consistency, septuples are represented in this manual, the RSA Sandbox program, and in output files in the following format

**Septuple format: [P, Q, N, Totient, E, K, D]2.2 The Septuple Management Menu**

The Septuple Management menu allows the creation, viewing, and manipulation of septuple objects. The septuple management menu can be accessed from the main menu by pressing ‘3’ as seen below.



**Figure 5:** Septuple management menu accessed by pressing ‘3’ from main menu

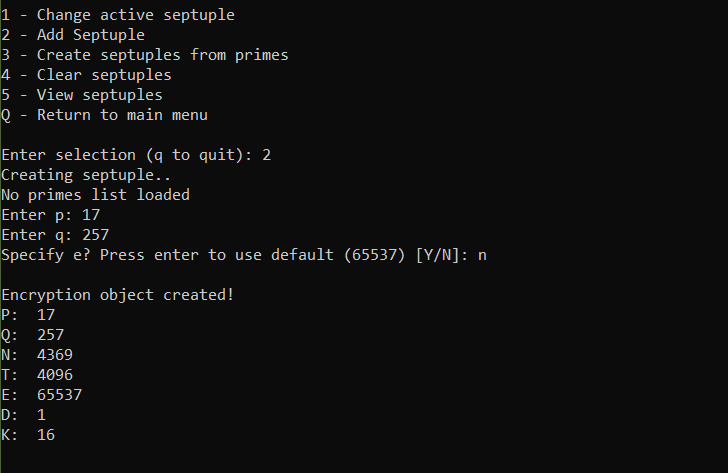
As shown in the screenshot, initializing the septuple menu will initially display all septuples loaded in the program, with a descriptor string accompanying the “active” septuple. This list can also be seen by pressing ‘5’ from the septuple management menu as shown, and the list can be cleared by pressing ‘4’.

**2.2.1 The Active Septuple**

The “active” septuple is the septuple that will be used by default when performing plaintext and/or input string encryption. The user can choose a new active septuple from among the loaded septuples at any time by pressing ‘1’ in the septuple management menu. There is an opportunity to specify a septuple other than the active one when performing plaintext encryption as well.

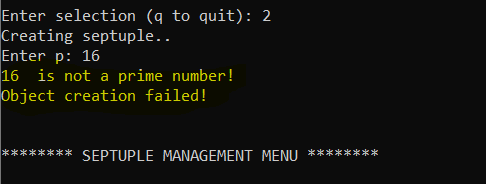
**2.2.2 Septuple Creation**

Option ‘2’ allows a new septuple to be generated from user input and added to system whilst ensuring proper values are provided for P, Q, and E. A screenshot of a typical septuple creation flow can be seen below.



**Figure 6:** Septuples can be created from the command line by inputting P, Q, and E

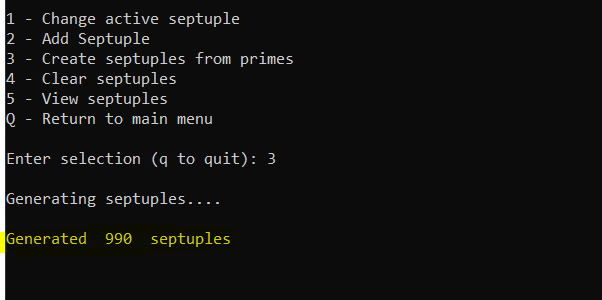
As shown, the program will prompt the user to input P and Q. The user can also input E, or use the default value of 65537. Error messages will be generated and the object creation will fail if bad and/or incompatible values are provided.



**Figure 7:** System will prevent incorrect P, Q, or E input values

**2.2.3 Septuple Generation from Primes**

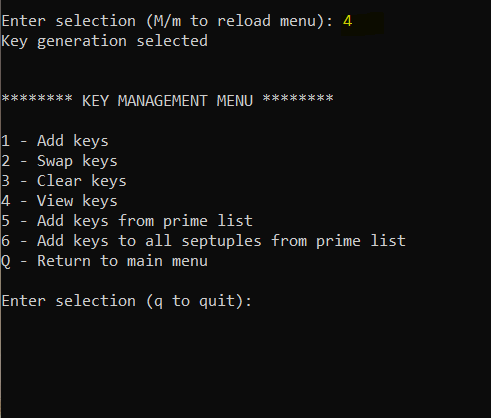
In additional to manually creating septuples one at a time, septuples can be generated from the internal list of primes . Choosing option ‘3’ will take all possible unique combinations of numbers in the internal prime and use those values for P and Q to generate septuples. E values are randomly chosen from among the “public key” list (3, 5, 17, 257, 65537). An example of this mechanism creating all septuple combinations for a prime list between 2 and 200 can be seen below.



**Figure 8:** Septuples can be generated automatically using the internal prime number list

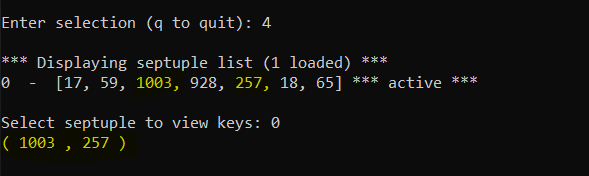
**2.3.0 Key Management**

The program maintains an internal dictionary associating a list of encryption keys with each septuple. The initial key used in septuple creation is first added to the dictionary, and the user can add more keys to any septuple from custom input as well as directly from the internal prime number list. The key management menu can be accessed by pressing ‘4’ from the command line as seen below.



**Figure 9:** The key management menu allows creation and manipulation of public keys

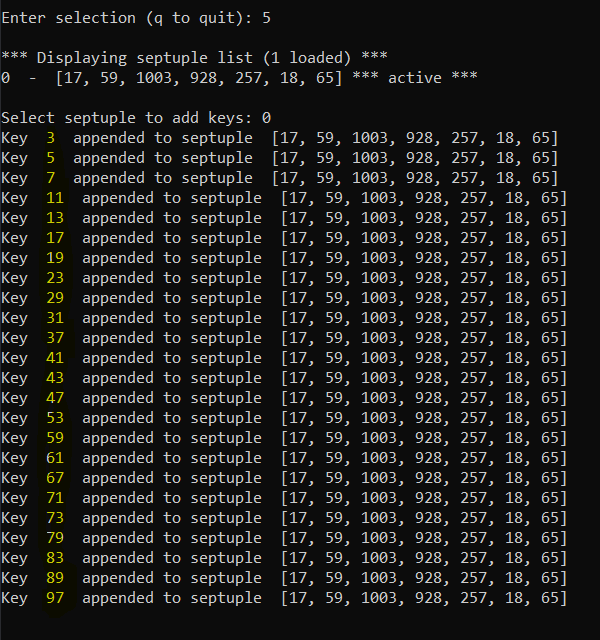
An example of option ‘4’, i.e. “View Keys” following creation of a single septuple is shown below.



**Figure 10:** Single septuple with a single key

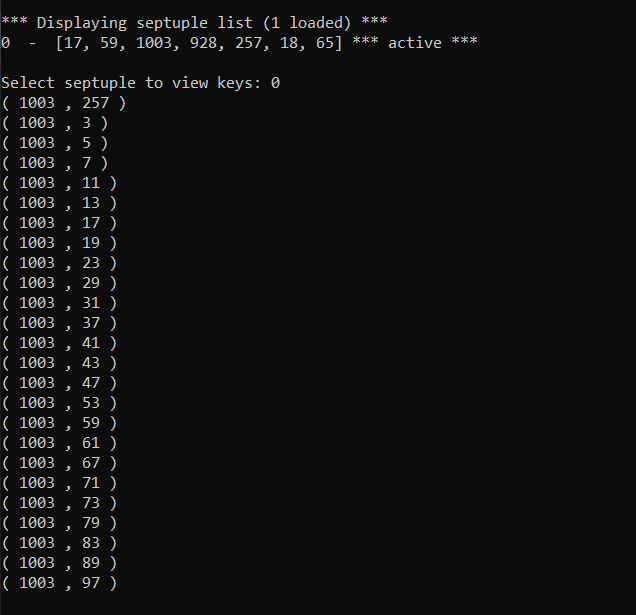
As one can see, the key list is shown in the form (N, E), and in this case the N and E values are identical to the values occupying the septuple object itself since no other keys exist yet.

Option ‘1’ allows a user to append a new key to a septuple via command line input. For more expansive key generation, option ‘5’ can be used to append the entire prime list to the key list of a chosen septuple, and further still, option ‘6’ can be used to append the prime list to every single septuple loaded in the program. The following screenshots illustrate an example of this mechanism. This assumes that prior to entering the key management menu, the user has already generate an internal prime list of all primes between 2 and 100.



**Figure 11:** Prime list can be appended to key list of a septuple

As shown, the primes between 2 and 100 are added to the chosen septuple. In contrast with **Figure 9**, the following screenshot shows the new output upon viewing the keys for the same septuple.

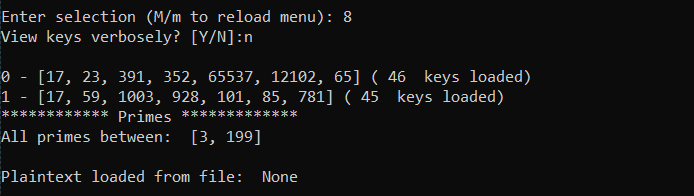


**Figure 12:** Twenty four additional keys now exist in the key list

It is worth noting that the loaded key in the septuple is 257. Option ‘2’ in the key management menu allows choosing a new key from a septuples key list. The septuple object will regenerate the D and K values accordingly upon swapping of the key value. For convenience, a user can enter a key value that does not exist when swapping keys. The program will detect this, ensure co-primality is met, and perform the swap whilst adding the new key to the dictionary.

**2.4.0 Displaying System Data**

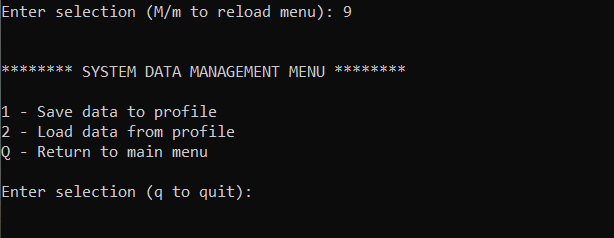
All primes, septuple, key data, and active plaintext file can be viewed by pressing option ‘8’ from the main menu. The program will allow the option of verbosely viewing all key data in addition to septuples, primes, and plaintext data.



**Figure 13:** All system data can be displayed via the display system data menu option

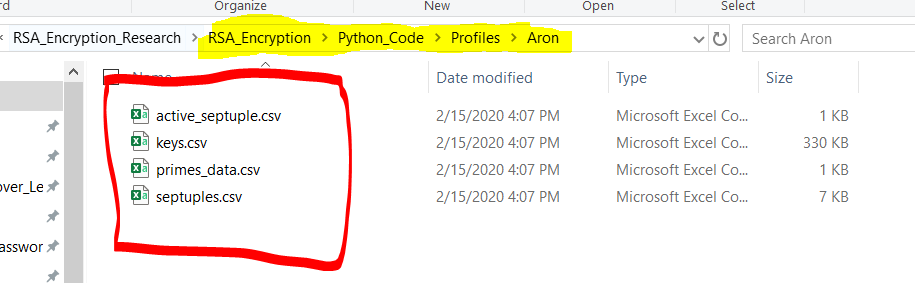
**2.5.0 Saving and Loading System Data**

Users can save all septuples, keys, prime numbers, and active septuple information to a personal profile for later restoration. The data saving/loading menu can be accessed by pressing ‘9’ from the main menu as seen below.



**Figure 14:** Data can be saved and re-loaded from the system data management menu

As mentioned in section 1.0, a “Profiles” directory is created upon program bootup if it does not exist. Option ‘1’ will prompt a user to enter a name to associate with their data. Once this is entered, all system data is saved to .csv files inside a folder with the provided name. To re-load the data, a user can choose option ‘2’ and enter the name user previously to save the data. The system will find the profile with the same name and load the contents of the .csv files back into the program. A screenshot of a profile and its four internal data files can be seen below



**Figure 15:** CSV files save the four system data components and can be edited directly

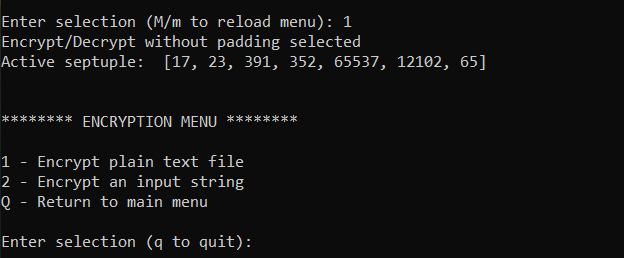
Users can take advantage of this mechanism by editing the data files directly. Upon re-loading the profile, the new data will be re-loaded. This can be used to enter desirable septuples directly into the .csv file for instance and remove the need to create the objects within the program.

**3.0.0 Encrypting Plaintext files and Strings**

The Sandbox can be used to encrypt plaintext files and input strings. An explanation of both mechanisms can be seen in the following sections.

**3.1.0 Encrypted Plaintext Files**

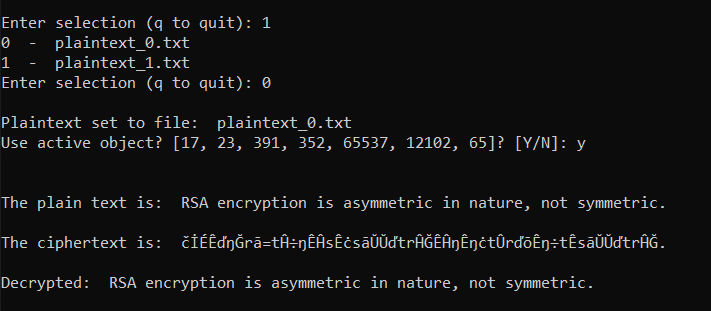
Users can use the active septuple to perform encryption/decryption plaintext files. A screenshot of the non-padding encryption menu can be seen below.



**Figure 16:** The active septuple can be used to encrypt plaintext and input strings

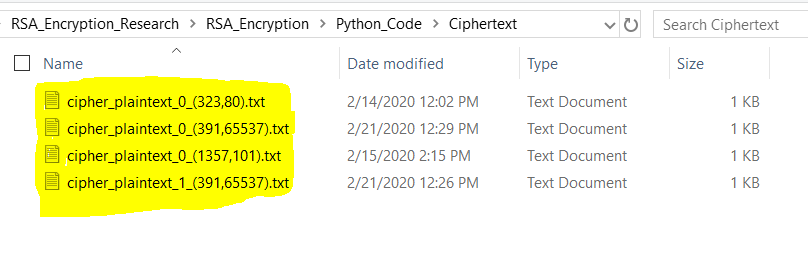
If no active encryption object yet exists, the program will detour and prompt the user to create a valid septuple.

Option one will read all text files in the Plaintext folder and prompt the user to select the plaintext file they wish to encrypt. An example can be seen below.



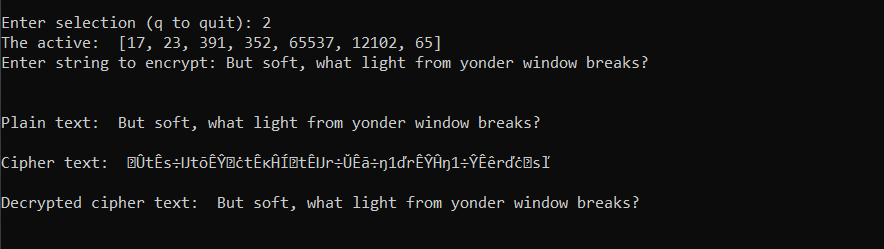
**Figure 17:** User selects plaintext to encrypt, results are shown and output to Ciphertext folder

The resulting ciphertext from the encryption will be appended in a text file format to the Ciphertext folder with the plaintext file and public key used incorporated into the file name, as seen below.

**Figure 18:** Ciphertext results are added to the ciphertext folder with the key used and plaintext file that was encrypted

**3.2.0 Encrypting input strings**

Users can also encrypt/decrypt input strings for quick visualization of the algorithm in action. This process can be done by pressing ‘2’ in the encryption menu as seen below.



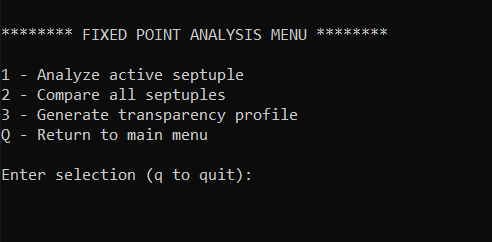
**Figure 19:** Input strings can also by encrypted/decrypted for quick visualization and algorithm testing

**4.0.0 Fixed Point Analysis**

The RSA Sandbox features several mechanisms for exploring the relative strength and occurrence of fixed points for any septuple/key combination. An overview of the fixed point analysis mechanisms and their functionality can be seen in the following sections.

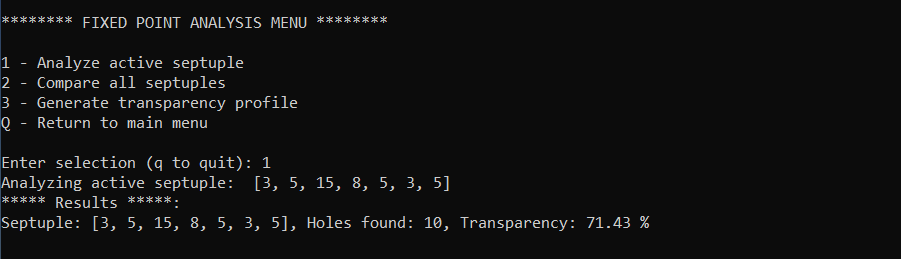
**4.1.0 Active Septuple Analysis and Septuple Comparison**

A screenshot of the fixed point analysis menu can be seen below.

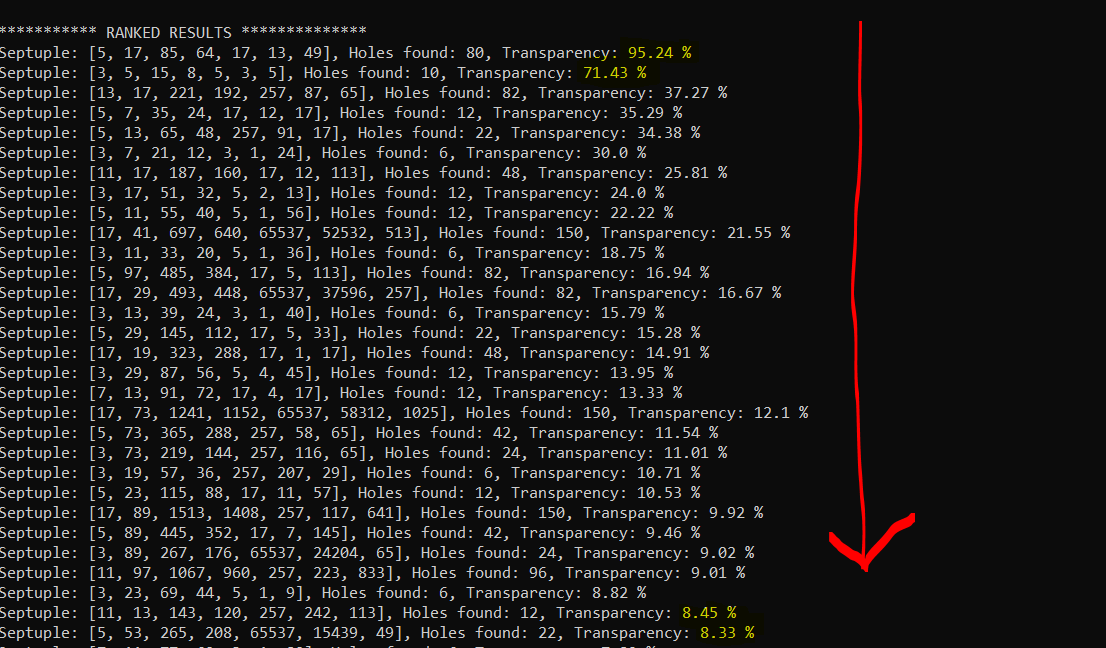


**Figure 20:** Fixed point analysis allows septuple and key comparison transparency analysis

The user can choose to analyze the strength of the active septuple by choosing option ‘1’. They can also compare the strength of every septuple loaded in the program with option ‘2’. Screenshots of these mechanisms in action can be seen below.

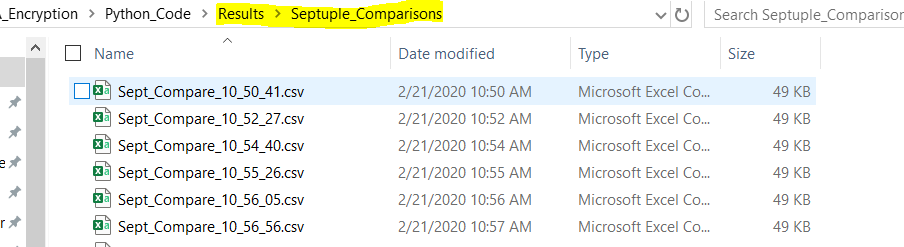


**Figure 21:** Option ‘1’ will analyze the active septuple



**Figure 22:** Option ‘2’ will compare all septuples and sort by transparency

Performing comparison of all septuples will display the results and also output them to the “Results” folder as mentioned in section 1.0. Septuple comparison results are stored in the subfolder ‘Septuple Comparison’ within the results folder as shown below and have a timestamp in their file title to differentiate them.



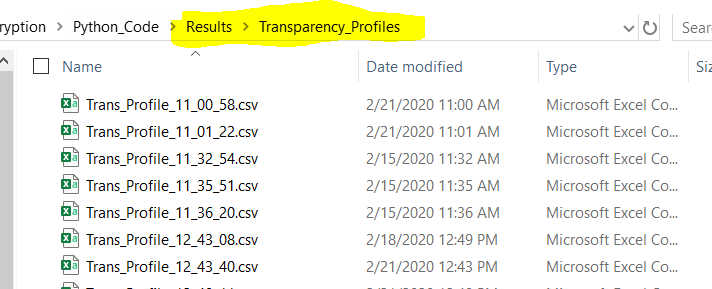
**Figure 23:** Septuple comparison results are stored in the ./Results/Septuple\_Comparisons directory

**4.2.0 Key Analysis and Transparency Profile Generation**

Option ‘3’ can be used to analyze all keys associated with a given septuple and rank them by strength. This is referred to as a “transparency profile” for a given septuple and key set. An example of the output of this mechanism can be seen below.

**Figure 24:** Option ‘3’ will compare all keys loaded for a given septuple

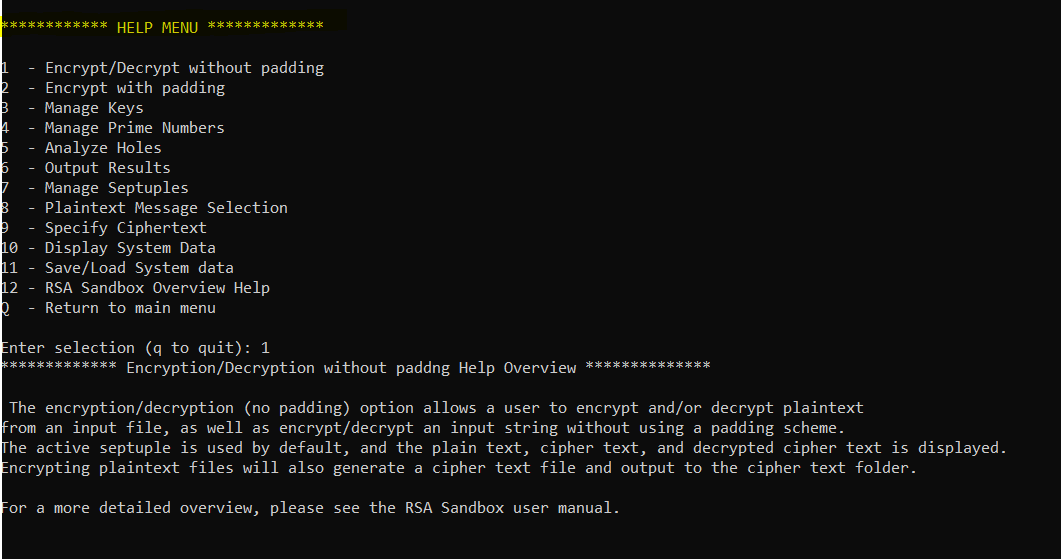
Similar to septuple comparisons, key analysis results will be appended to the “Transparency Profiles” directory within the results folder.



**Figure 25:** Transparency profile results are stored in ./Results/Transparency\_Profiles directory

**5.0.0 The Help Menu**

The help menu can be accessed from the main menu by pressing ‘h’. Users can select system options for a brief functionality overview.



**Figure 26:** The help menu can be used for functionality descriptions while running the Sandbox