RSA Sandbox User Manual

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

**Table of Contents** FIX PAGE NUMBERS EVENTUALLY

1. Overview --------------------------------------------------------------------------------- 3
2. System Data Management ------------------------------------------------------------- 4

2.1 Septuple Management ------------------------------------------------------------- 5

2.2 Prime Number Management ------------------------------------------------------ 6  
2.3 Key Management------------------------------------------------------------------- 7  
2.4 Saving/Loading Data -------------------------------------------------------------- 8

1. Saving/Loading Data -----------------------------------------------------
2. Encrypting/Decrypting Plaintext and Strings
3. Fixed Point Analysis
4. Output Files
5. Help Menu

**1.0 RSA Sandbox Overview**

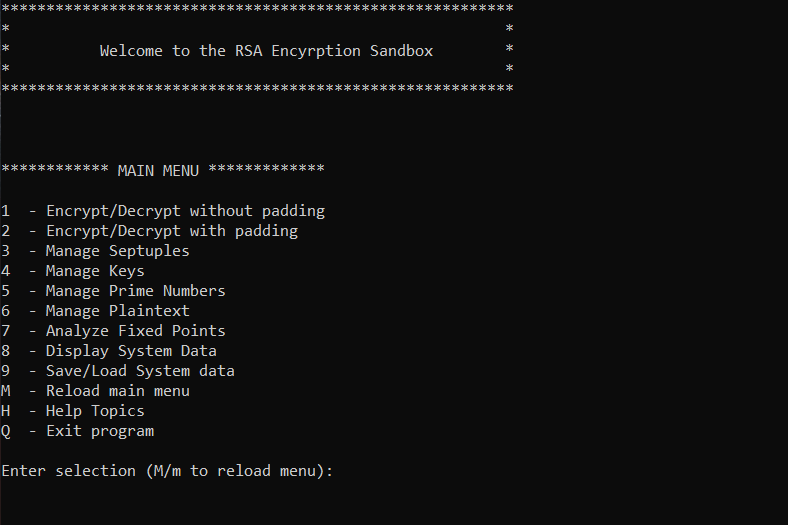
The RSA Sandbox is a research tool that enables users to explore and experiment with the RSA encryption algorithm. The program allows a user to create RSA encryption “septuples”, generate prime numbers, create encryption keys, and combine the three to search for fixed point patterns. In addition, users 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:

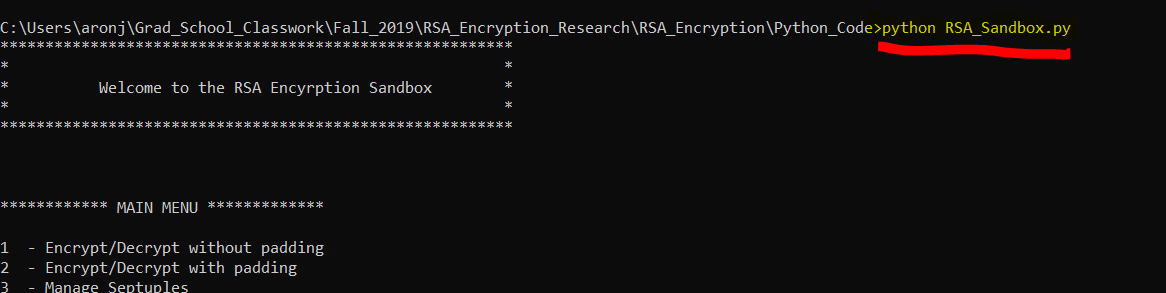
Link to Repository: <https://github.com/aronjschwartz/RSA_Encryption.git>

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

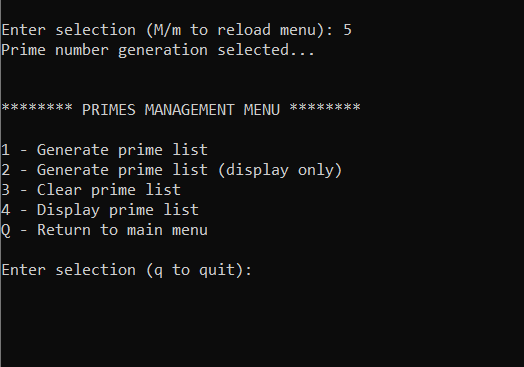
The following sections provide a detailed functionality overview for all components of the RSA sandbox program.

**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.

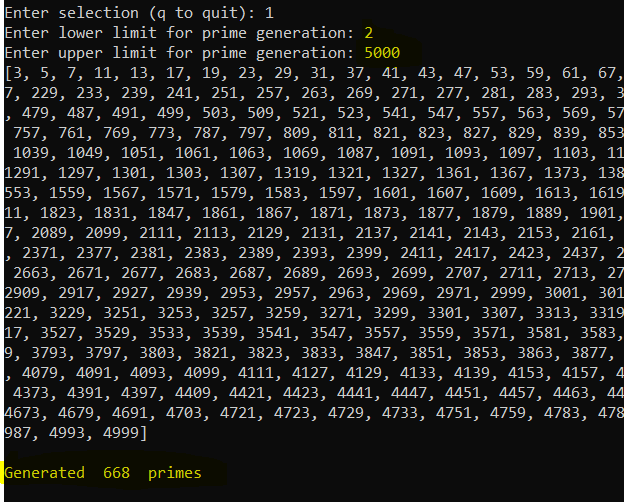
**2.1.0 Prime Number Management**

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 6:** 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 7:** 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**

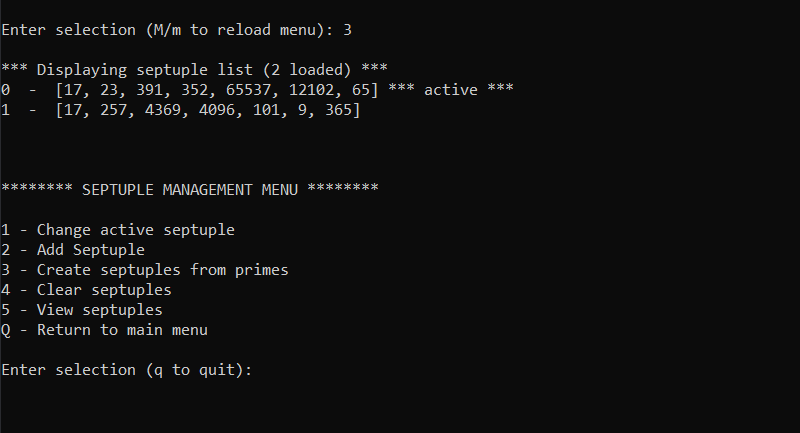
Performing encryption in the RSA Sandbox is accomplished by the creation of septuple objects, a python object that houses the 7 parameters and internal functionality needed to perform RSA encryption and decryption. The 7 parameters of a septuple and a brief description can be seen below.

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

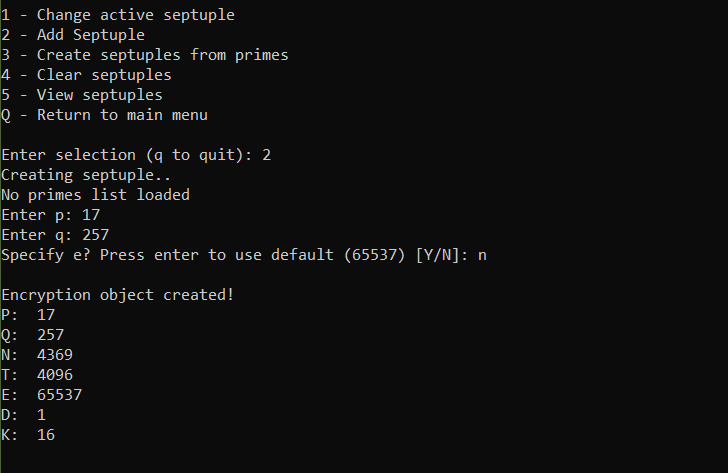
As shown in the screenshot, initializing the septuple menu will initially display all septuples loaded in the program, with a descriptor string accompanying the “active” septuple. This list can also be seen by pressing ‘5’ from the septuple management menu as shown, and the list can be cleared by pressing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

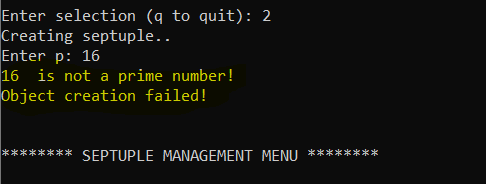
**2.2.2 Septuple Creation**

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 4:** 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.



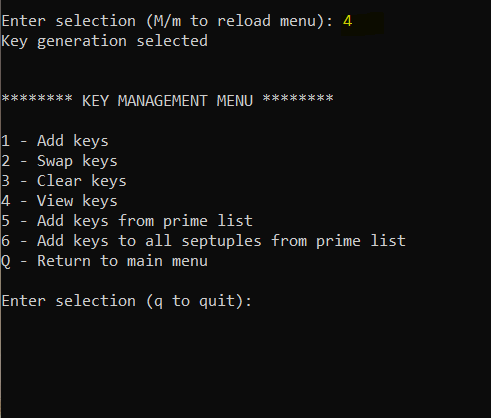
**Figure 5:** System will prevent incorrect P, Q, or E input values

**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. Keys 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.

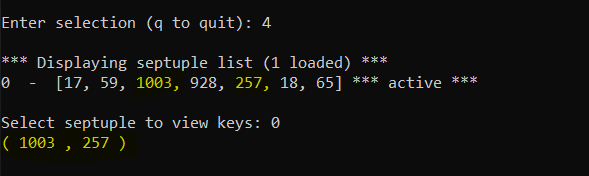
**2.3.0 Key Management**

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

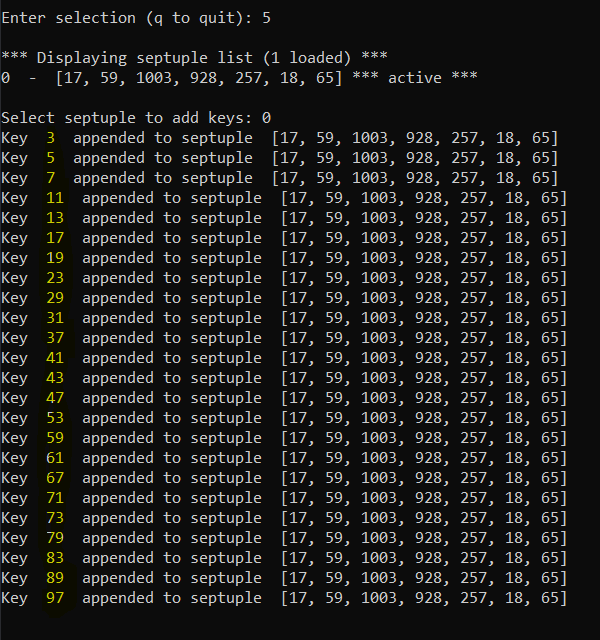
An example of option ‘4’, i.e. “View Keys” following creation of a single septuple is shown below.



**Figure 9:** Single septuple with a single key

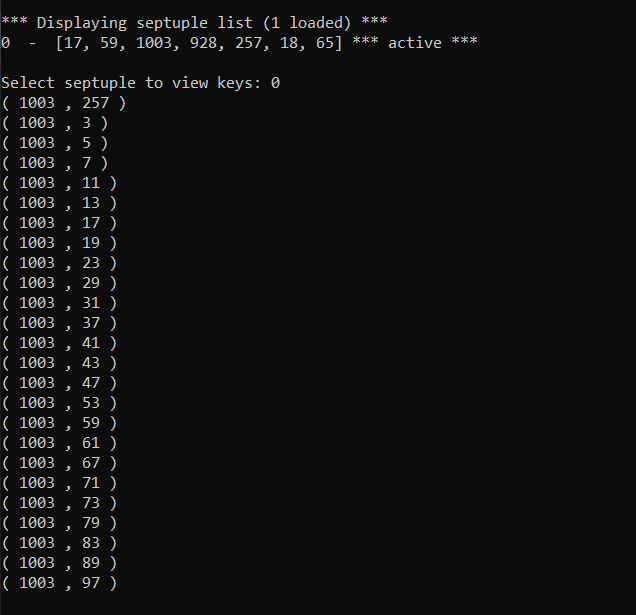
As one can see, the key list is shown in the form (N, E), and in this case the N and E values are identical to the values occupying the septuple object itself since no other keys exist yet.

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 10:** 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.

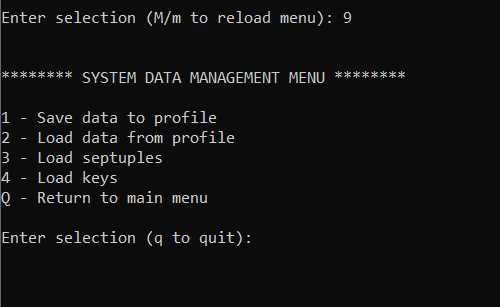


**Figure 11:** Twenty four additional keys now exist in the key list

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.

**3.0 Saving and Loading 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 12:** Data can be saved and re-loaded from the system data management menu

Upon bootup of the program, a “Profiles” directory is created 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.