# ASSIGNMENT 01
# ADVANCED ALGORITHMS

# IMPLEMENTING RABIN-KARP STRING MATCHER AND RSA ENCRYPTION

| | |
|---|---|
| PES1201802688 | MANJUNATH S |
| PES1201802296 | SOLAMAN T BABY |

## GOALS ACHIEVED:

Implemented Rabin-Karp string matcher for Text search with alphabet {a-z, A-Z, 0-9}.

Implemented RSA-Encryption with 32-bit Primes for encoding consisting of a 19-digit number , composed from 16-digit Credit card number + 3-digit CVV.

The implementation of both of the algorithms have been done in Python.

### SYSTEM CONFIGURATION:

Model name : MacBook Air
Processor name : Intel Core i5
Processor speed : 1.8GHz
No of processors : 1
No of cores : 2
L2 cache(per core) : 256kb
L3 cache : 3mb
Memory : 8gb

# RABIN-KARP STRING MATCHER :

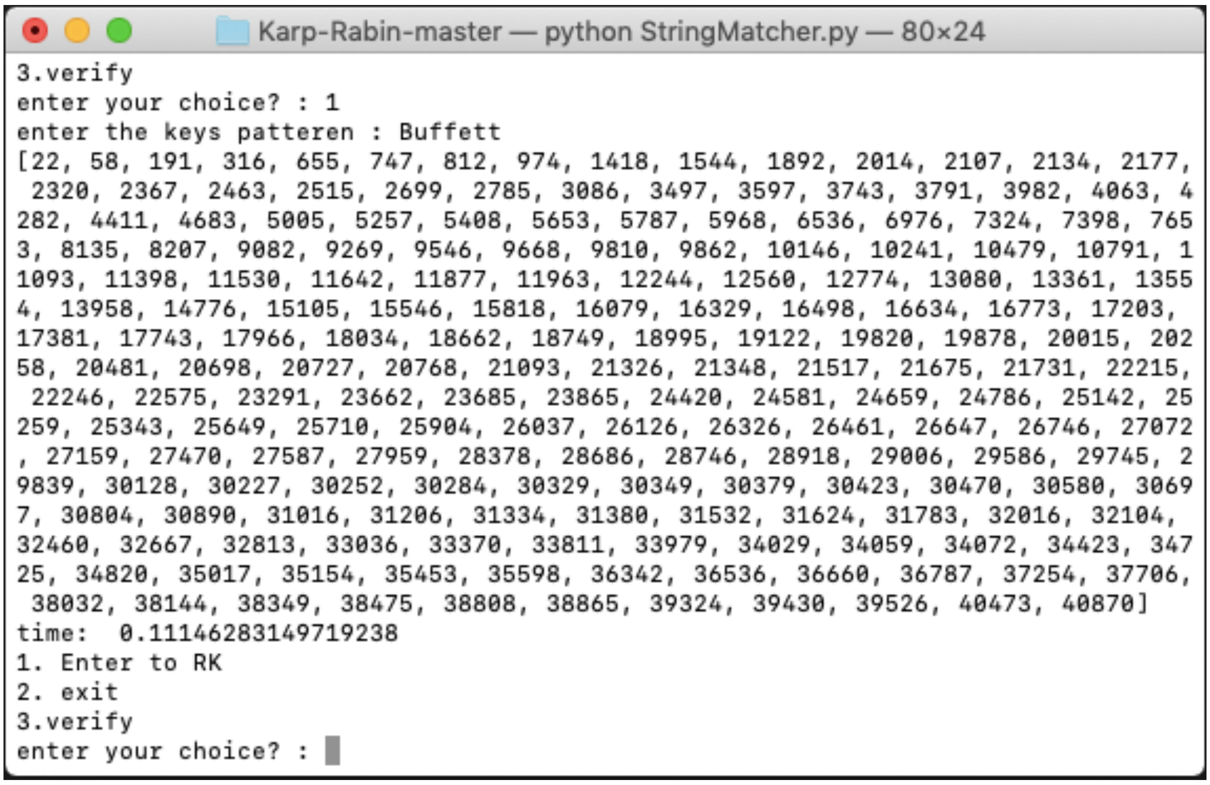
The Rabin Karp string matching algorithm works by calculating the hash value of the text, over the same number of characters as the pattern. This hash is computed for the pattern as well. The hash values of the text and pattern are matched, and if they are equal, then there is a possibility of a match, which is verified by matching individual characters.

## IMPLEMENTATION  DETAILS:

When Rollinghash is created it receives the text, the pattern, the radix and a 32 bit prime number for the modulus. The default value for the radix is 256, so that all ASCII text is covered , and not just alphanumeric characters.

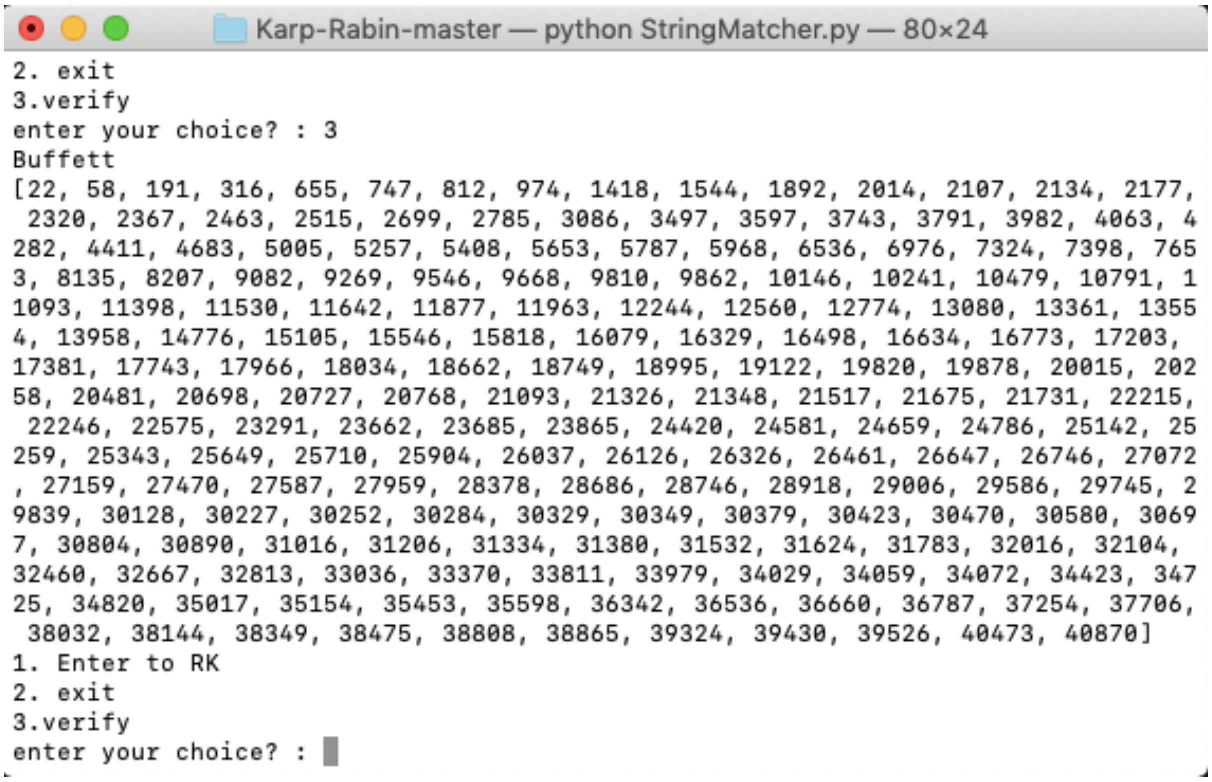
On construction, the hash values for the pattern and the text are calculated and stored as attributes. It  has interfaces that allow callers to shift by one character. This shifting causes the new hash value of the text to be calculated. If this new hash value matches with the pattern's hash value, then there is a hit for that position (shift).

## SCREENSHOTS:



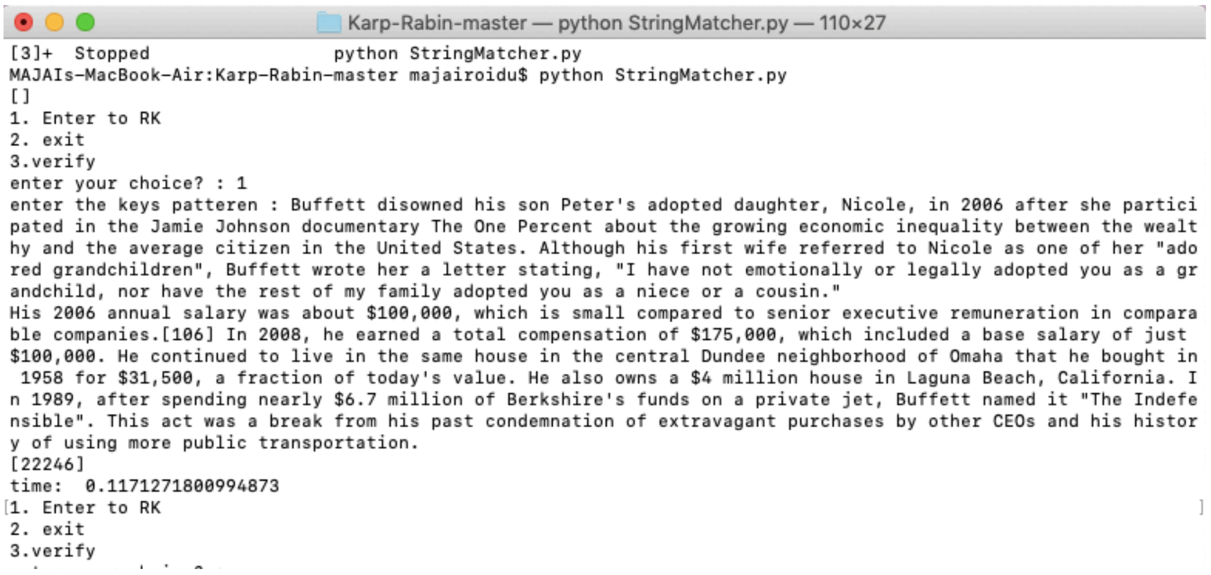
```
3.verify
enter your choice? : 1
enter the keys patteren : Buffett
[22, 58, 191, 316, 655, 747, 812, 974, 1418, 1544, 1892, 2014, 2107, 2134, 2177,
 2320, 2367, 2463, 2515, 2699, 2785, 3086, 3497, 3597, 3743, 3791, 3982, 4063, 4
282, 4411, 4683, 5005, 5257, 5408, 5653, 5787, 5968, 6536, 6976, 7324, 7398, 765
3, 8135, 8207, 9082, 9269, 9546, 9668, 9810, 9862, 10146, 10241, 10479, 10791, 1
1093, 11398, 11530, 11642, 11877, 11963, 12244, 12560, 12774, 13080, 13361, 1355
4, 13958, 14776, 15105, 15546, 15818, 16079, 16329, 16498, 16634, 16773, 17203,
17381, 17743, 17966, 18034, 18662, 18749, 18995, 19122, 19820, 19878, 20015, 202
58, 20481, 20698, 20727, 20768, 21093, 21326, 21348, 21517, 21675, 21731, 22215,
 22246, 22575, 23291, 23662, 23685, 23865, 24420, 24581, 24659, 24786, 25142, 25
259, 25343, 25649, 25710, 25904, 26037, 26126, 26326, 26461, 26647, 26746, 27072
, 27159, 27470, 27587, 27959, 28378, 28686, 28746, 28918, 29006, 29586, 29745, 2
9839, 30128, 30227, 30252, 30284, 30329, 30349, 30379, 30423, 30470, 30580, 3069
7, 30804, 30890, 31016, 31206, 31334, 31380, 31532, 31624, 31783, 32016, 32104,
32460, 32667, 32813, 33036, 33370, 33811, 33979, 34029, 34059, 34072, 34423, 347
25, 34820, 35017, 35154, 35453, 35598, 36342, 36536, 36660, 36787, 37254, 37706,
 38032, 38144, 38349, 38475, 38808, 38865, 39324, 39430, 39526, 40473, 40870]
time:  0.11146283149719238
1. Enter to RK
2. exit
3.verify
enter your choice? : 
```

Fig1: RK , found Pattern "BUFFET" at different positions



```
2. exit
3.verify
enter your choice? : 3
Buffett
[22, 58, 191, 316, 655, 747, 812, 974, 1418, 1544, 1892, 2014, 2107, 2134, 2177,
 2320, 2367, 2463, 2515, 2699, 2785, 3086, 3497, 3597, 3743, 3791, 3982, 4063, 4
282, 4411, 4683, 5005, 5257, 5408, 5653, 5787, 5968, 6536, 6976, 7324, 7398, 765
3, 8135, 8207, 9082, 9269, 9546, 9668, 9810, 9862, 10146, 10241, 10479, 10791, 1
1093, 11398, 11530, 11642, 11877, 11963, 12244, 12560, 12774, 13080, 13361, 1355
4, 13958, 14776, 15105, 15546, 15818, 16079, 16329, 16498, 16634, 16773, 17203,
17381, 17743, 17966, 18034, 18662, 18749, 18995, 19122, 19820, 19878, 20015, 202
58, 20481, 20698, 20727, 20768, 21093, 21326, 21348, 21517, 21675, 21731, 22215,
 22246, 22575, 23291, 23662, 23685, 23865, 24420, 24581, 24659, 24786, 25142, 25
259, 25343, 25649, 25710, 25904, 26037, 26126, 26326, 26461, 26647, 26746, 27072
, 27159, 27470, 27587, 27959, 28378, 28686, 28746, 28918, 29006, 29586, 29745, 2
9839, 30128, 30227, 30252, 30284, 30329, 30349, 30379, 30423, 30470, 30580, 3069
7, 30804, 30890, 31016, 31206, 31334, 31380, 31532, 31624, 31783, 32016, 32104,
32460, 32667, 32813, 33036, 33370, 33811, 33979, 34029, 34059, 34072, 34423, 347
25, 34820, 35017, 35154, 35453, 35598, 36342, 36536, 36660, 36787, 37254, 37706,
 38032, 38144, 38349, 38475, 38808, 38865, 39324, 39430, 39526, 40473, 40870]
1. Enter to RK
2. exit
3.verify
enter your choice? :
```

Fig2: RK, cross checking the previous result with python lib function.



```
[3]+  Stopped                 python StringMatcher.py
MAJAIs-MacBook-Air:Karp-Rabin-master majairoidu$ python StringMatcher.py
[]
1. Enter to RK
2. exit
3.verify
enter your choice? : 1
enter the keys patteren : Buffett disowned his son Peter's adopted daughter, Nicole, in 2006 after she partici
pated in the Jamie Johnson documentary The One Percent about the growing economic inequality between the wealt
hy and the average citizen in the United States. Although his first wife referred to Nicole as one of her "ado
red grandchildren", Buffett wrote her a letter stating, "I have not emotionally or legally adopted you as a gr
andchild, nor have the rest of my family adopted you as a niece or a cousin."
His 2006 annual salary was about $100,000, which is small compared to senior executive remuneration in compara
ble companies.[106] In 2008, he earned a total compensation of $175,000, which included a base salary of just
$100,000. He continued to live in the same house in the central Dundee neighborhood of Omaha that he bought in
 1958 for $31,500, a fraction of today's value. He also owns a $4 million house in Laguna Beach, California. I
n 1989, after spending nearly $6.7 million of Berkshire's funds on a private jet, Buffett named it "The Indefe
nsible". This act was a break from his past condemnation of extravagant purchases by other CEOs and his histor
y of using more public transportation.
[22246]
time:  0.1171271800994873
1. Enter to RK
2. exit
3.verify
```

fig3: RK, result containing pattern of 1000 characters with timing.

## RESULTS:

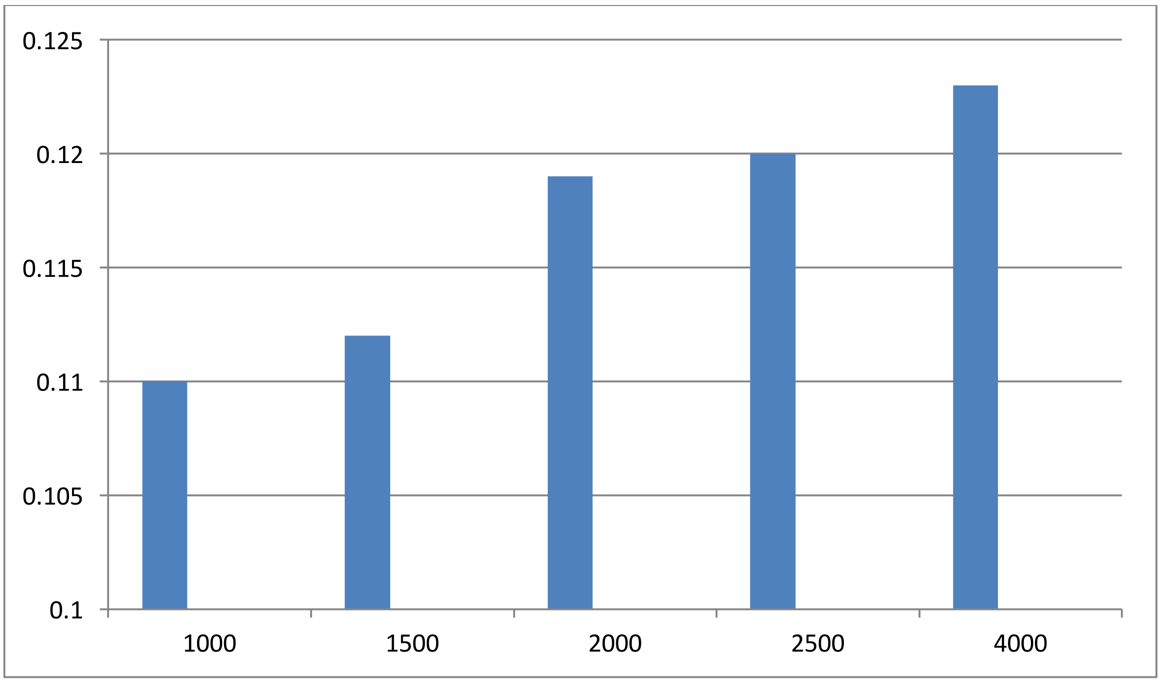


Fig4: Performance of Rabin-karp over varying pattern size
X axis : length of pattern and Y axis : time(secs)

## RSA ENCRYPTION :

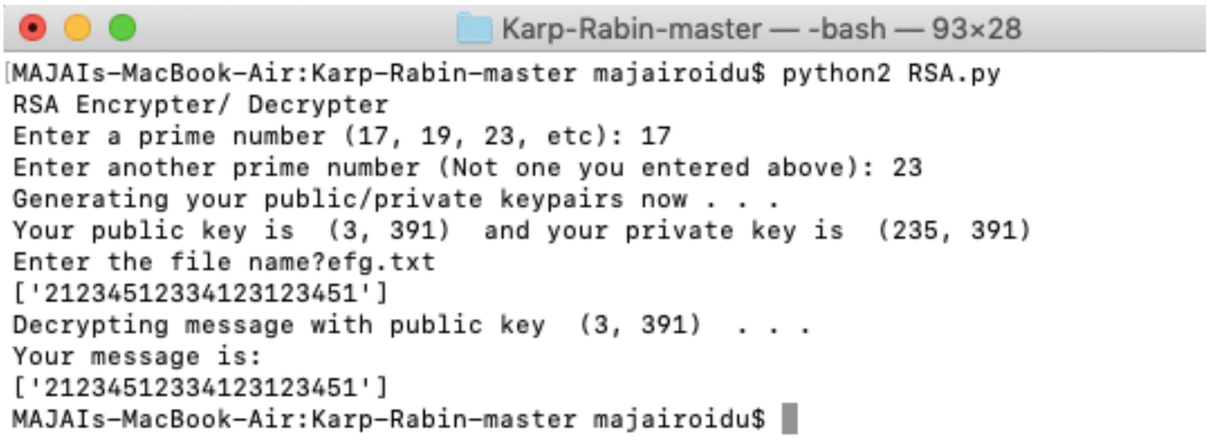
The RSA cryptosystem mainly consists of three parts – generation of the key, encrypting the data, and decrypting the data.

Some key features of the implementation are shown below:

• The keys can be generated by specifying the number of bits in the key.
• Once p and q are manually selected, n is calculated, and the value of e is chosen random. “d” is calculated to b the inverse of e.
• Since there is no communication involved, any key (public or private) can be used for encryption / decryption. For this implementation, the public key (e,n) is always used for encryption, and private key (d,e) is always used for decryption.
• The implementation offers simple interfaces to generate a dataset which consists of “n”entries of 19 digits each, and save the output to a file. This dataset will be used to test the encryption and decryption using the RSA Cryptosystem.

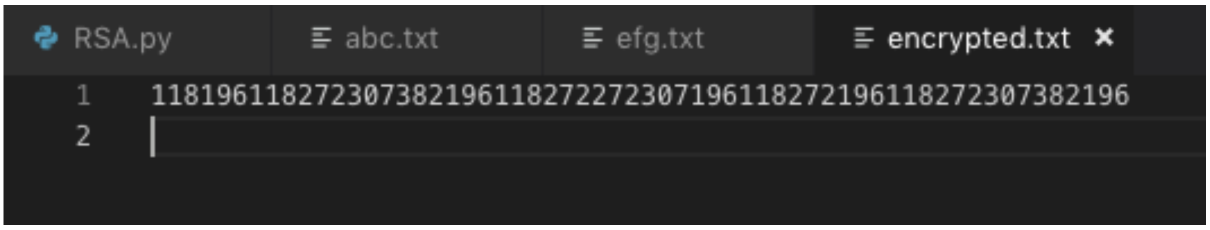
• Setting the key size to 32 bits seemed to cause issues in the modular exponentiation – the decryption of the encrypted number didn't result in the original number. This was because the message M (19 digit number) was equal to or greater than the modulus n. Hence, a key size of 64 bit was used.

SCREEN SHOTS:



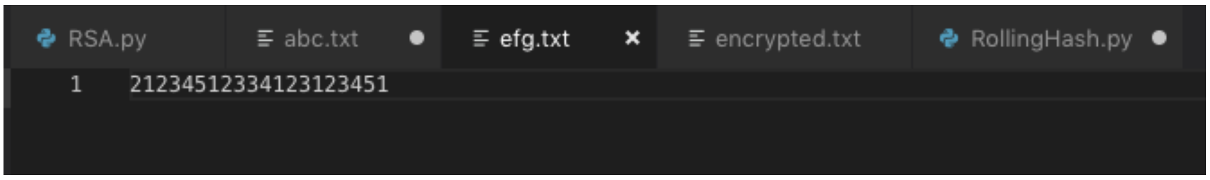
```
MAJAIs-MacBook-Air:Karp-Rabin-master majairoidu$ python2 RSA.py
RSA Encrypter/ Decrypter
Enter a prime number (17, 19, 23, etc): 17
Enter another prime number (Not one you entered above): 23
Generating your public/private keypairs now . . .
Your public key is  (3, 391)  and your private key is  (235, 391)
Enter the file name?efg.txt
['2123451233412312345 1']
Decrypting message with public key  (3, 391)  . . .
Your message is:
['2123451233412312345 1']
MAJAIs-MacBook-Air:Karp-Rabin-master majairoidu$
```

Fig5: RSA, execution with decrypted text.



```
1181961182723073821961182722723071961182721961182723073821 96
```

Fig6: RSA, encrypted values.



```
2123451233412312345 1
```

Fig7: RSA, Decrypted values in a text file.

## CONCLUSION :

In RK, with increasing length of pattern size the time is almost constant and small values of prime number may cause several spurious hits which degrades the performance.

In RSA, we find that encryption takes much longer if the number of digits are prime and if we take small bit length its easy to break the encrypted values.