Library:

Python-RSA is a pure-Python RSA implementation. It supports encryption and decryption, signing and verifying signatures, and key generation according to PKCS#1 version 1.5. It can be used as a Python library as well as on the commandline. The code was mostly written by Sybren A. Stüvel.

File : rsa_model.py

First function to genenrator keys

```
4
5  def generatorKeys():
6      (pubKey, privKey) = rsa.newkeys(1024)
7      return pubKey,privKey
8
9
```

Second function

Encrypt files but on binary level to speed up the process and ability. As we know we cant encrypt more than key size the trick here is encrypt line by line each line must be 117 byte because key size 1024bit. Takes file path and path to save and public key

Line 18-20:

If size of file equal key size encrypt direct

Line 23-39:

Split file to part each part 117 byte and encrypt then store data and move to other part.

Finally save file

```
9
10 def rsa_encrypt_binfile(file_path,save_path,pub_key):
11
12     with open(file_path, 'rb') as f:
13       message = f.read()
14
15     length = len(message)
16     default_length = 117
17
18     result = []
19     if length <= default_length:
20       result.append(base64.b64encode(rsa.encrypt(message,pub_key)))
21
22
23     offset = 0
24     while length - offset > 0:
25       if length - offset > default_length:
26         result.append(base64.b64encode(rsa.encrypt(message[offset:offset+default_length],pub_key)))
27       else:
28         result.append(base64.b64encode(rsa.encrypt(message[offset:],pub_key)))
29       offset += default_length
30
31     with open(save_path,"ab") as w:
32       for ciphertext in result:
33         ciphertext += b"\n"
34         w.write(ciphertext)
```

Third function :

Here the process is simple we need to just to decrypt each line of a file the encrypt function implement all work spilt etc.

```
35
36  def rsa_decrypt_binfile(file_path,save_path,priv_key):
37
38      with open(file_path,"rb") as f:
39          line = f.readline()
40          while line:
41              message = base64.b64decode(line.strip(b"\n"))
42
43              plaintext = rsa.decrypt(message, priv_key)
44              with open(save_path, 'ab') as w:
45                  w.write(plaintext)
46              line = f.readline()
47
48
```

Forth function:

Signature function here the trick is hashing file and sign it to speed up the process

```
49
50  def sign_sha1(file_path,save_path, privkey):
51      with open(file_path, 'rb') as f:
52          message = f.read()
53
54      sign =  base64.b64encode(rsa.sign(message, privkey, 'SHA-1'))
55
56      with open(save_path,"wb") as f :
57          f.write(sign)
58
59
```

Last function

Verify function like sign function but here we pass two file and check

```
60
61  def verify_sha1(file_path, signature_path, pubkey):
62      try:
63          with open(signature_path, 'rb') as f:
64              signature =base64.b64decode( f.read())
65
66          with open(file_path, 'rb') as f:
67              message =f.read()
68
69          return rsa.verify(message, signature, pubkey) == 'SHA-1'
70      except:
71          return False
72
73
74
```