```python
#!/usr/bin/env python
# RSA secret key
# By Olivia Mattsson and Amanda Flote
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_v1_5
from Crypto.Hash import SHA
from Crypto import Random
import pyasn1.codec.der.decoder
import pyasn1.codec.der.encoder
from pyasn1_modules import rfc8017
import base64


def decrypt(m, privKey):
    # From Cryptodome documentations: https://pycryptodome.readthedocs.io/en/latest/src/cipher/pkcs1_v1_5.html
    dsize = SHA.digest_size
    sentinel = Random.new().read(15+dsize)
    cipher = PKCS1_v1_5.new(privKey)
    decrypted_message = cipher.decrypt(m, sentinel)
    return decrypted_message


# RSA private key syntax: https://tools.ietf.org/html/rfc3447#appendix-A.1.2
# Using https://pycryptodome.readthedocs.io/en/latest/src/public_key/rsa.html to construct the private key
# And retrieve information about the RSA components:
def reconstructKey():
    with open('key.pem', 'r') as pem_file:
        k = ''.join(pem_file.readlines()[1:-1])
        pem_key = base64.b64decode(k)
        # Decode using PKCS #1 - the format used in RSA private keys:
        # Found: https://www.programcreek.com/python/example/95764/pyasn1.codec.der.decoder.decode
        decoded_key, rest = pyasn1.codec.der.decoder.decode(pem_key,asn1spec=rfc8017)
        # According to the structure of RS key (https://tls.mbed.org/kb/cryptography/asn1-key-structures-in-der-and-pem),
        # these values can be found on these indexes:
        n, e, d, p, q = decoded_key[1], decoded_key[2], decoded_key[3], decoded_key[4], decoded_key[5]
        # The censored part is the value of n, but it is the same as p * q and our new value can be calculated:
        newN = p * q
        decoded_key[1] = newN
        new_key = RSA.importKey(pyasn1.codec.der.encoder.encode(decoded_key))

    return new_key


if __name__ == '__main__':
    k = reconstructKey()
    encrypted_message = 't2JtdjaM71d67nvC9CZZ5kpumAmY9LrEh8//OdUKX+xvv+UG+9tvM/9P/Aen/tW21FFfNUWPKm+EkuHjecvMa5KqZqVoXKNqVz4Ke4p1fL1eVdUpJ8R
    encrypted_message = base64.b64decode(encrypted_message)
    m = decrypt(encrypted_message, k)
    print(m)
```