# Simple Documentation for RSA Implementation

Abraham Xiao

December 9, 2013

## 1 Introduction

For convenience, we cite some facts and description from [1] without much more mentioning. We hope this will not intrigue intelligence property issues.

**Definition 1** *The* RSA problem *is the following: given a positive integer $n$ that is a product of two distinct odd primes $p$ and $q$, a positive integer $e$ such that $gcd(e, (p-1)(q-1)) = 1$, and an integer $c$, find an integer $m$ such that $m^e \equiv c \mod n$.*

In other words, the RSA problem is that of finding $e^{th}$ roots modulo a composite integer $n$. The condition imposed on the problem parameters $n$ and $e$ ensure that for each integer $c \in 0, 1, \ldots, n-1$ there is exactly one $m \in 0, 1, \ldots, n-1$ such that $m^e \equiv c \mod n$. Equivalently, the function $f : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ defined as $f(m) = m^e \mod n$ is a *permutation*.

## 2 Implementation

### 2.1 Data Structure

A special data structure containing two primes $p$ and $q$, the multiplication of $(p-1)(q-1)$ as well as public, private key pairs is defined as follows:

```
typedef struct RSA_PARAM_Tag
{
```

```
unsigned __int64 p, q;  // p and q are two primes
unsigned __int64 f; // f=(p-1)*(q-1)
unsigned __int64 n, e; // n=pq; gcd(e,f)=1 public keys
unsigned __int64 d; // private key, ed=1(mod f), gcd(n,d)=1
}RSA_PARAM;
```

A class containing a private data as well as public data, method is defined as follows:

```
class RandNumber
{
private:
unsigned __int64 randSeed;
public:
RandNumber(unsigned __int64 s = 0);
unsigned __int64 Random(unsigned __int64 n);
};
```

For the rest part we itemize features in our implementation.

- An array of small prime table is created to speed-up the process of identifying if a large number is a prime or composite.
- The seed used to generate large random number is taken from current calendar time to ensure enough randomness.
- A random number is generated in a way of multiplying a large enough number and then add another one.
- Rabin-Miller primality test is implemented. And the testing loop is adjustable.
- Both the Euclidean algorithm and binary algorithm for calculating *greatest common divisor* are implemented.
- The whole RSA algorithm is implemented neatly.

# 3 Examples

We use a toy sample to conclude this simple documentation. Up to now, the string with spaces is not supported. We are sorry for that, indeed.

```
abrahamx91@debian:~/Professional/Git/CIS612-Composition/Codes$
␣./a.out
p=47911
q=38839
f=(p-1)*(q-1)=1860728580
n=p*q=1860815329
e=46387
d=1574922403

␣Please␣enter␣your␣plaintext:␣Abraham-Xiao-Keep-Moving!

␣Ciphertext␣is:␣b58c31a␣6d4c7761␣15dafa09␣17a7e101␣2c02bb80
␣17a7e101␣650e1f0c␣64dc1f07␣2c3b1738␣1189bc8c␣17a7e101␣19873f79
␣64dc1f07␣5596ced9␣38a8ee68␣38a8ee68␣9bb7fbf␣64dc1f07␣49bec0cc
␣19873f79␣52d47daf␣1189bc8c␣2dd5496b␣13442502␣2bec903d␣0

Decipher:␣␣You␣plaintext␣should␣be:␣Abraham-Xiao-Keep-Moving!

abrahamx91@debian:~/Professional/Git/CIS612-Composition/Codes$


abrahamx91@debian:~/Professional/Git/CIS612-Composition/Codes$
./rsa.out
p=55901
q=34763
f=(p-1)*(q-1)=1943195800
n=p*q=1943286463
e=3501
d=155966301

Please␣enter␣your␣plaintext:␣Dr.␣Zeyar␣works␣hard␣at␣Masdar
Institute␣of␣Science␣and␣Technology,␣Abu␣Dhabi,␣54224,␣UAE.

Ciphertext␣is:␣adade57␣32897eda␣52a1a30␣4c829245␣14a4abb3
a1d1f32␣43b55255␣1495f338␣32897eda␣4c829245␣59ab7cb0␣173246d5
32897eda␣3a9f6276␣f252cab␣4c829245␣5f319e1c␣1495f338␣32897eda
71f4e4c6␣4c829245␣1495f338␣1c5adab7␣4c829245␣578c2244␣1495f338
f252cab␣71f4e4c6␣1495f338␣32897eda␣4c829245␣4ad4cdcd␣3f9710a
```

```
f252cab␣1c5adab7␣6ab94d57␣1c5adab7␣3d3bb9af␣1c5adab7␣a1d1f32
4c829245␣173246d5␣1de4b8d6␣4c829245␣37fa50a5␣846acf6␣6ab94d57
a1d1f32␣3f9710a␣846acf6␣a1d1f32␣4c829245␣1495f338␣3f9710a
71f4e4c6␣4c829245␣49347697␣a1d1f32␣846acf6␣5f319e1c␣3f9710a
173246d5␣14e39337␣173246d5␣69fa7dd7␣43b55255␣315c3799␣4c829245
1a4c5703␣1f0ec954␣3d3bb9af␣4c829245␣adade57␣5f319e1c␣1495f338
1f0ec954␣6ab94d57␣315c3799␣4c829245␣6cd1e5fd␣6a0f76e3␣1ef30397
1ef30397␣6a0f76e3␣315c3799␣4c829245␣4949bf92␣1a4c5703␣4345300e
52a1a30␣0

Decipher:␣␣You␣plaintext␣should␣be:␣Dr.␣Zeyar␣works␣hard␣at
Masdar␣Institute␣of␣Science␣and␣Technology,␣Abu␣Dhabi,␣54224,
UAE.

abrahamx91@debian:~/Professional/Git/CIS612-Composition/Codes$
date
Mon␣Dec␣␣9␣12:07:15␣GST␣2013
abrahamx91@debian:~/Professional/Git/CIS612-Composition/Codes$
```

Some parts are manually modifies due to page space issues.

# Change Log

In this part we document the changes we made after the beta version.

20131209. As required, we replaced the `cin << str` with `getline()` to patch the defect that only non-space-separated input can be encrypted. Right now it works fine if a single line is typed in. We plan to deal with chunks of input, say "happy.txt" file and so on if time permitting.

# Acknowledgment

I would like to thank Dr. Zeyar for preparing high quality lectures throughout the whole Fall 2013 semester. In addition, I am extremely grateful for being able to carry out *care-free* research at Masdar Institute of Science and

Technology, especially as a late applicant last year[1].

# References

[1] Menezes, A. J., Vanstone, S. A., and Oorschot, P. C. V. *Handbook of Applied Cryptography*, 1st ed. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[1] I submitted my full application just 2 weeks before the deadline. But I got the offer pretty fast.