Homework: big number library and implementation of RSA

Jean-Sébastien Coron

Université du Luxembourg

1 Introduction

The aim of this homework is to implement a big number library in C or C++ and to implement the RSA algorithm based on this library.

2 Big number library

A big integer will be represented using an array of digits in base B. The following `struct` can be used:

```
typedef struct {
  int sign;
  int size;
  int *tab;
} bignum;
```

were `sign` is the sign bit, and `size` is the size of the dynamic array `tab`.

The following functions must be implemented:

`bignum str2bignum(char *str)`
converts a string to a bignum.

`bignum add(bignum a, bignum b)`
adds the integers a and b.

`bignum sub(bignum a,bignum b)`
return $a - b$.

`bignum mult(bignum a,bignum b)`
returns the product of a and b.

`bignum remainder(bignum a,bignum n)`
returns the remainder of the division of a by n. The (inefficient) algorithm given in the course can be used, with a small base B (for example, $B = 10$ or $B = 16$). Alternatively, a more efficient algorithm can be used, as described in [1].

`bignum addmod(bignum a, bignum b,bignum n)`
returns $a + b \mod n$.

`bignum multmod(bignum a,bignum b,bignum n)`
returns $a \cdot b \mod n$.

`bignum expmod(bignum a,bignum b,bignum n)`
returns $a^b \mod n$

`int millerrabin(bignum a,int t)`
performs the Miller-Rabin test on integer a with security parameter t.

`bignum genrandom(int length)`
generates a random integer of size `length` bits.

`bignum genrandomprime(int length)`
generates a random prime of size `length` bits, using the Miller-Rabin primality test.

3 The RSA algorithm

The goal is to implement the RSA algorithm using the previous library. The following functions must be implemented:

`void keygen(bignum *n,bignum *e, bignum *d,int length)`
generates an RSA modulus $n = p \cdot q$, where p and q are two prime integers of size `length` bit. The function also generates the public/private exponent pair (e, d).

`bignum RSAencrypt(bignum m,bignum e,bignum n)`
takes as input a message m, a public exponent e and a RSA modulus n and returns the corresponding ciphertext c.

`bignum RSAdecrypt(bignum c,bignum d,bignum n)`
takes as input a ciphertext c, a private exponent d and a RSA modulus n and returns the corresponding plaintext m.

`void testRSA(int length)`
generates an RSA public-key (e, n) and its corresponding private-key (d, n). It asks the user for a message m to encrypt, and outputs the corresponding ciphertext encrypted with public-key (n, e). It then applies the decryption algorithm with private-key (d, n) and checks that the original message is recovered.

4 Documents to be provided

A document containing the following informations must be provided in .pdf form.

- An overview of the algorithms used to implement the functions and an overview of the implementation.
- The program source code.

The program source code must also be provided separately. It must compile on a Linux machine using the `gcc` or `g++` compiler. The `main` function in the program must call the `testRSA` function with `length=512`. The message that is encrypted must eventually be recovered after decryption.

References

1. V. Shoup, *A Computational Introduction to Number Theory and Algebra*, available at `http://shoup.net/ntb/`.