Mittuniversitetet
MID SWEDEN UNIVERSITY

Laboratory Assignment: Arithmetic with Very Large Integers

Daniel Bosk*

bigint.tex 2169 2015-01-14 22:37:01Z danbos

Contents

1 Introduction

When performing cryptographic operations you usually work with very large numbers; usually in the range of 2^{128} to 2^{256}, in the base of crypto systems based on elliptic curves, or 2^{2048} to 2^{4096}, in the case of crypto systems based on number theory. On the average personal computer the processor instructions usually handle only numbers in the range of 2^{32} to 2^{64}. This laboratory assignment aims to some problems related to this.

2 Aim

After completion of this assignment you will:

- Have insight into the complexity of multiple-precision arithmetic.
- Be able to implement multiple-precision arithmetic.
- Be able to analyse the time complexity of arithmetic operations.

3 Reading instructions

Before commencing with the assignment you should read up on modular arithmetic and exponentiation. This is covered by sections 0 to 3 and 5 to 6 in *Talteori* by Enblom and Sola [1] (in Swedish). There are of course other sources, e.g. Biggs's *Discrete Mathematics* [2].

For the curious student, Knuth covers algorithms on multiple-precision arithmetic in *The Art of Computer Programming* [3, section 4.6], including the algorithm for fast exponentiation.

4 Assignment

This section covers the work to be done and the next section covers how it will be examined, and what to be done to pass it.

For this assignment you should implement multiple-precision arithmetic in C or C++. It is recommended to do this in C++ to take advantage of the object orientation. You should implement multiple-precision arithmetic for integers such that you can perform the standrard integer arithmetic operations; i.e. addition (and subtraction), multiplication, integer division and the modulo operator.

When your arithmetic operations work properly you should implement the algorithm for fast exponentiation. Measure the effectiveness of this algorithm, i.e. compare computing 2^{256} with and without the use of the algorithm.

Finally, use your implementation to decide whether $2^{256} - 1$ and $2^{256} + 1$ are primes or not; if they are not, find their prime factors.

5 Examination

To pass this assignment you must hand in the following:

1. Your implementation for multiple-precision arithmetic.
2. Your measurements comparing the algorithm for fast exponentiation and the trivial way of just multiplying the numbers.
3. Whether $2^{256} - 1$ and $2^{256} + 1$ are prime numbers or not, including their factors as proof if they are not.

References

[1] Andreas Enblom and Alan Sola. "Talteori". KTH:s matematiska cirkel 2008–2009, Kungliga Tekniska högskolan. 2008. URL: `http://www.math.kth.se/cirkel/2008/kompendium.pdf`.

[2] Norman Biggs. *Discrete mathematics*. 2. ed. Oxford: Oxford Univ. Press, 2002. ISBN: 0-19-850717-8 (hft.)

[3] Donald Ervin Knuth. *The Art of Computer Programming*. 3rd ed. Vol. 2, Seminumerical algorithms. Reading, Mass.: Addison-Wesley, 1997. ISBN: 0-201-89684-2.