The art of cryptography: Lattices and cryptography, summer 2013

PROF. DR. JOACHIM VON ZUR GATHEN, DR. DANIEL LOEBENBERGER

1. Exercise sheet
Hand in solutions until Sunday, 21 April 2013, 23:59h.

Reminder.

- For the course we remind you of the following dates:
  - Lectures: Monday and Thursday 13:00h-14:30h **sharp**, b-it bitmax.
  - Tutorial: Monday 14:45h-16:15h, b-it bitmax.
- A word on the exercises. They are important. Of course, you know that. In order to be admitted to the exam it is necessary that you earned at least 50% of the credits. You need 50% of the marks on the final exam to pass the course. If you do, then as an additional motivation, you will get a bonus for the final exam if you attended the tutorial regularly **and** earned more than 70% or even more than 90% of the credits.

Note: From the following exercise on, we assume that you have a variant of the basis reduction algorithm running on your computer. Any experiments that you have to do in the sequel will need such a library. It is highly advised that you use C/C++ in the future. If you are not feeling comfortable using this language, you will have use your own implementation or you will have to search on your own for an optimized basis reduction library for the language you have in mind.

Exercise 1.1 (NTL: A Library for doing Number Theory). (5+5 points)

NTL is a high-performance, portable C/C++ library providing data structures and algorithms for manipulating signed, arbitrary length integers, and for vectors, matrices, and polynomials over the integers and over finite fields. It has a highly optimized built in basis reduction algorithm that we will employ frequently for the rest of the lecture. To start, install NTL on your computer and get familar with the NTL-API. Hints how to install NTL and details on the API can be found on `http://www.shoup.net/ntl/doc/tour.html`. Now run the code `lll.cpp` from the course page. To compile it, call for example under UNIX (or Mac OS X) the compiler in the following way: `g++ -o lll lll.cpp -lntl -lm`. You might have to include the headers using the $-I$ flag and the library using the $-L$ flag. Details on that can be found in the `man` page of `g++`. Consider now the lattice spanned by the matrix (written in row notation)

$$\begin{bmatrix} 1 & 10 & 101 \\ 2 & 12 & 121 \\ 3 & 13 & 131 \end{bmatrix}.$$

(i) Hand in the output of the supplied program. 5

(ii) Interpret the result. +5

Exercise 1.2 (The subset sum cryptosystem). (6 points)

In the lecture we studied the subset sum cryptosystem. Let $m = 437$ and $r = 204$. Bob's private key is $s = (2, 6, 10, 26, 68, 161)$.

2 (i) Compute Bob's public key t.

Now Alice wants to send the string $x = (0, 1, 0, 1, 1, 0)$.

1 (ii) Encrypt x with Bob's public key obtaining y.

3 (iii) Describe in detail how Bob will decrypt the encrypted message y and do the decryption.

Exercise 1.3 (Breaking the subset sum cryptosystem). (27+10 points)

Goal of this exercise is to implement the subset sum cryptosystem and the algorithm breaking it.

10 (i) Implement the subset sum cryptosystem in a programming language of your choice and hand in the source code. You will need to implement three functions:

Algorithm. Generate Key Pair.
Input: A positive integer n.
Output: The private key (s, m, r) and the public key t. The private key consists of a superincreasing sequence $s = (s_1, \ldots, s_n)$ with $s_1 = 1$, $s_i \in]S_i, 2S_i]$ for $i > 1$, a value $m \in]S_{n+1}, 2S_{n+1}]$ where $S_i = \sum_{j<i} s_j$ and a value $r \in \mathbb{N}$ with $\gcd(r, m) = 1$. The public key is a sequence $t = (rs_1 \bmod m, \ldots, rs_n \bmod m)$.

Algorithm. Encrypt.
Input: A message $x \in \{0, 1\}^n$. The public key t.
Output: The encrypted message $y = \sum_{i \le n} x_i t_i$.

Algorithm. Decrypt.
Input: A message $y \in \mathbb{N}$. The private key (s, m, r).
Output: The decrypted message x.

In the lecture we have seen an algorithm that computes (sometimes) a solution to the subset sum problem.

10 (ii) Implement the lattice-based solver from the lecture in a programming language of your choice. Hand in the source code.

2 (iii) Assume you have intercepted the message $y = 1147$. Bob's public key is

$$s = (465, 441, 417, 241, 330, 251).$$

Compute the message $x \in \{0, 1\}^6$ that Alice sent to Bob using your algorithm.

5 (iv) Let $n = 6$ and $C = 512$. Run 100 examples with $t_1, \ldots, t_6 \xleftarrow{\$} \{1, \ldots, C\}$ and $x \xleftarrow{\$} \{0, 1\}^6 \setminus \{(0, \ldots, 0)\}$. How often did your algorithm not succeed in finding x?

+10 (v) Experiment with other values of C, maybe much larger ones. You may also want to increase the message length beyond 6 bits. What do you observe? Can you explain the behavior?