Cryptography lab

Weeks 01 / 02

Guillermo Zecua

October 1, 2021

Contents

About the lecture material

- Cryptography: classical (*)
- Number theory: congruences $\rightarrow$ primality tests (*)
- Cryptography: public key RSA (*)
- Number theory: quadratic residues $\rightarrow$ finite fields
- Cryptography: public key ElGamal (*)
- Cryptography: key transport protocols

About this lab

- Computer science students have their lab every two weeks (odd and even weeks).
- There are going to be 4 lab activities during the semester. Each of the activity will be graded and you get points $1 + 1 + 1 + 1.5$ for the final grade.

Schedule:

- Lab 1 due: week 5 (odd groups) or 4 (even groups)
- Lab 2 due: week 9 (odd groups) or 8 (even groups)
- Lab 3 due: week 13 (odd groups) or 12 (even groups)
- Lab 4 due: week 13 (odd groups) or 14 (even groups)

What are the learning objectives of the lab?

- NOT "to learn how to code": you know that already
- NOT "to translate an algorithm into code": soon this will be automated (program synthesis)

What are the learning objectives of the lab?

What we really want you to learn:

1. To understand and explain the question or the problem to solve
2. To devise a plan for a solution, and to justify and illustrate why is this a good plan
3. To put your plan in action with a program and describe concisely your implementation
4. To be critical of your own work; to verify the correctness and validity of your solution

How are the labs going to be graded?

Active participation means:

- I am goint to ask questions to everybody
- Questions about your program but also related to the problem
- Explanation over implementation

How are the labs going to be graded?

Active participation is mandatory

If you submit the program *without* previous active participation, there will be *no* points.

- More important than the ability to write a program is your capacity to discuss and describe what it does and how it fits into the context of a given problem.

What can go wrong?

- Second lab: nobody wants to answer questions and participate...
- ... Third lab (deadline): there is not enough time to take a careful look at everybody
- Consequence: those who could not be heard on the deadline lab will get no points. Sorry.

What can go wrong?

Do not be afraid to participate!

- I will do my best to manage time so that everybody has a chance to present their work.
- You need to have some material prepared before the deadline.
- The best way to be prepared to give good and precise answers is to *understand* what you are doing.
- Still, I will be there to help you get the best out of your work.
- It is *not necessary* that you have a complete program to get the full points!
- Be prepared to present before the deadline! (especially important for lab 4)

Lab attendance

Attendance

You can have at most one absence during the semester.

First lab activity

The greatest common divisor

- Implement in Python or Haskell three essentially different algorithms for computing the greatest common divisor $\gcd(a, b)$ of two natural numbers a and b. Perform a comparative running-time or multiplication-number analysis of these algorithms for a set of at least 10 inputs. *Do not use the modulo operator in Python / Haskell.*

or, alternatively,

- Implement in Python or Haskell a single algorithm for computing the greatest common divisor $\gcd(a_1, a_2, \ldots, a_k)$ of a k-set of natural numbers $a_1, \ldots, a_k$ for any $k > 1$. *Do not use the recursion rule* $\gcd(a, b, c) = \gcd(\gcd(a, b), c)$ nor any of its variants, and do not use linear search.

"essentially different"

- What are "essentially different" algorithms?
- Repeated substraction and division are the same.
- Two things are "essentially the same" when are related by group-theoretic operations (Pólya theorem)

About GUI-UI

User interfaces

Do not use user interfaces, neither command-line nor graphical.

- We will work primarily on the code directly; the inputs for the programs will be written directly on code, and we will run the interpreter every time.
- This will allows to write directly expressions as `2**1020` for 2^{1020} without problems.

Guidelines

- Code only in Python or Haskell
- You can also use SageMath, and I warmly encourage you to use it
- You can use Jupyter (default modus in SageMath)
- Write the program in a single file. (Easier to follow on screen)
- Literate Haskell most welcome

Haskell

This is a great opportunity to learn the basics of Haskell!

Lab activity submission

Moodle

The source files for the programs will be uploaded to the MS Teams platform, in the assignment section created for every lab activty.

- Only the source file (the "text file"), not any compiled file.

Camera on

- In this lab we are working with mathematical questions, of the kind of paper and pencil.
- It really helps me to have some visual feedback when we are talking on a particular question or problem.
- You can answer the questions in English or Romanian

When camera on?

It is enough to have the camera on only when I am asking you a question, or when you are explaining something.

Cheating

Zero tolerance on cheating

If I discover that there are two identical (or almost) identical programs, then those persons will lose *all* their lab points.