SSN Lab Assignment: Crypto Math [Bonus]*

P. Prjevara Zhiyang Wang Lucas Hecht†

Feedback deadline:
Monday September 23, 2024 10:00 CET

1 Introduction

In this assignment we aim to deepen your understanding of the math described within the lecture today - especially the first part!

2 Modulo Arithmetic

Questions

1. With your own words, explain the following items on your wiki.[1]
   - Remainder
   - Modulus
   - Equivalence class
   - Congruence

2. Try to find the answer to the following without using a computer or calculator for the calculations.
   - 127 mod 15
   - -127 mod 15
   - 145 mod 22
   - 395 mod 42

3. Are the following statements true or false? Explain your answers, and if false, write down the correct answer.
   - 5 is congruent to 5 in modulo 5
   - 1 is congruent to 6 in modulo 5
   - 17 is congruent to 395 in modulo 42
   - 1 mod 5 and -4 mod 5 are in the same equivalence class
   - 2 mod 5 and 28 mod 5 are in the same equivalence class
   - A is congruent to 17 (mod 5). The value of A is...?

*September 15, 2024.

†lucas@os3.nl,yang@os3.nl

[1] Main source: `https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/what-is-modular-arithmetic`

3 Modular Addition, Substraction

4. Answer the following quesitons. Show your thought process by writing on your wiki.

    - (918 + 335) mod 30
    - If A mod 9 is 7, then write the result of (A - 21) mod 9

4 The Greatest Common Divisor

In the lecture, it was discussed how to find the Greatest Common Divisor of a number.

5. Try to find the answer to the following without using a computer or calculator for the calculations. Describe the process that you followed on your wiki.

    - Find the Greatest Common Divisor (GCD) of 400 and 200.
    - Find the GCD of 292 and 76.
    - Find the GCD of 30 and 260.
    - Find the GCD of 12 and 56.

5 The Modular Inverse of a Integer

You can probably recall that B is the inverse of A, if it's true that A * B = 1. This statement is also true for the modular inverse of any number: if you want to find B, which is the modular inverse of A mod C, you should satisfy the statement (A * B) mod C = 1.

6. Try to find the modular inverse fo the following numbers. Show your thought process. Feel free to use tools to achieve this (Libre Office Calc can work well). Note B must be an integer as well.

    - Find the inverse of 3 modulo 8.
    - 12 modulo 35.
    - 19 modulo 73.
    - Lastly, what about 12 modulo 56?

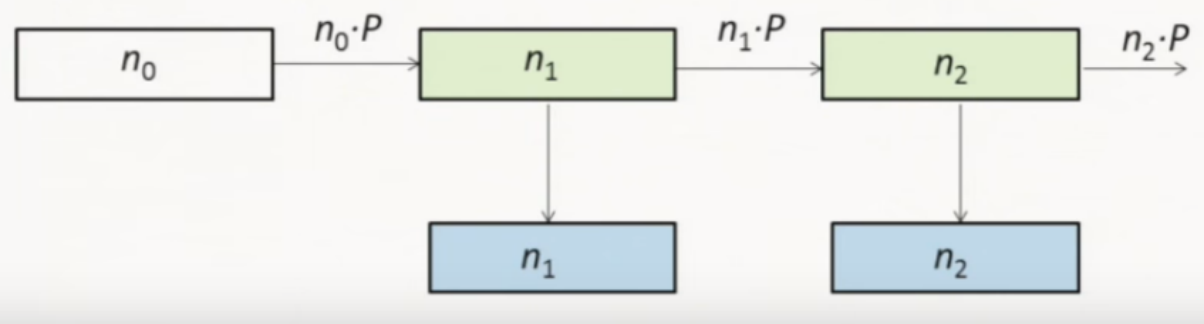

Figure 1: Schematic of a random number generator based on Elliptic Curve Cryptography (ECC)

6 The Discrete Logarithm Problem

7. In a short paragraph, explain the discrete logarithm problem, and how is it used in modern cryptography. Show at least two different examples.

A Random Number Generator Based on Elliptic Curve Cryptography (ECC)

8. Observe the above schematic of a random number generator, that is based on ECC. n_0 is the seed that is repeatedly used as a multiplier for point P, taking the x-coordinate of the result. Answer the following quesitons:

   - Does the discrete logarithm problem apply here?
   - What is the security of this random number generator dependant on? Is it secure?

7 Integer factorisation

As you have learned in the lecture, every integer can be written down as a product of primes. Integer factorisation (or prime decomposition) is the process of determining the primes that produce the integer. So far, there hasn't been an efficient algorithm devised to find these primes, thus this problem is considered hard. In the following exercise, you will experiment with this issue yourself.

7.1 Cython and Python

The algorithms you are going to play with can get extremely slow if interpreted by Python on the fly, especially when using large numbers. For this reason you will first compile your Python code using a library called Cython into a C Shared Object. You will cover C and Shared Objects as part of CIA in more detail, so for now it is enough if you follow the steps outlined below. If you have spare time, you can read more about Cython at `https://cython.readthedocs.io/en/latest/src/tutorial/cython_tutorial.html`. The scripts are implemented using Python 2.7.

1. Download and extract the `crypto_math.tar.xz` from `http://software.os3.nl/SSN/`.
2. Install all the requirements specified in the requirements.txt.
3. In a terminal window, navigate to the folder where you have extracted the contents of the .zip file to, and from there run "python setup.py build_ext –inplace". This will create a C shared object form the `num_factorisation.pyx` script. You can import the script using the below instructions and run the functions described within, using the python interpreter.

4. Now, have a look at the `num_factorisation.pyx` script, and try to understand what it does and how to use it.

Usage Instructions

Once you have created the C Shared object, you can access the functions defined in the Python script the following way:

1. Enter the Python interpreter, by typing ”python”
2. Type ”import num_factorisation”, then press enter.
3. Try the given algorithm by typing : `num_factorisation.prime_factors1(15)`

Questions

9. Explain what the `num_factorisation.pyx` code does. Write your answer on your wiki. Experiment with the factorisation algorithm `prime_factors1()` using your own integers.

10. Implement your own factorisation algorithm in the `prime_factors2()` function. Show that it works. BONUS for a more efficient algorithm!

11. Checkout the `randRun(30)` function. Using the given script, measure the time it takes to factorise pseudo random numbers up 30 digits, with both functions. Run the experiment 3 times, and plot the results using your favourite plotting software (Hint: “`gnuplot`”). Upload the plots on your wiki.