INGI2347 : Exercises
Lab session 3 *

Xiao Chen, Marco Canini

March 7, 2016

Exercise 1: Hash Functions and the Birthday Paradox

The SHA-1 hash function generates 160-bit digital fingerprints which are typically used when signing public keys. Suppose we decide to create a public key for each person on Earth (6×10^9 people).

1. Calculate the probability that at least one fingerprint for public key is the same as Gérard Mansoiffe's

`0x11c42333 330debe6 63d722a5 f34388c8 b88520bb`

(in hexadecimal notation), using the fact that $1 - x \approx e^{-x}$, for x close to 0.

2. Calculate the probability that at least two people have identical SHA-1 fingerprints.

Exercise 2: RSA Algorithm

This exercise deals with the details of the RSA public-key algorithm.
1. Detail out the procedure to be followed to generate a pair (public key, private key).

2. Encrypt the message "16" with the public key $(17, 55)$. The calculation can be easily done by hand after noticing that $16^5 \equiv 1 \pmod{55}$.

3. Decrypt the message "8" with this private key $(33, 55)$ in order to retrieve the clear message.

4. Why can we not encrypt the message "66" with the public key $(17, 55)$?

5. How can we use RSA to compute signature of a message that has an arbitrary length ?

Exercise 3: RSA Vulnerabilities

Previous exercise uses the RSA algorithm as is presented in the introduction to cryptography manuals: in practice, we should *never* use it as it is! The RSA algorithm is, in this form, vulnerable to many attacks. To convince ourselves, let us study one amongst them: show that the product of the cyphertext of two messages (constructed using the same private key) is equal to the cyphertext of the product of the two messages.

Exercise 4: Exhaustive Search for Asymmetric Keys

Knowing that $\pi(n)$, the number of prime numbers smaller than n, can be approached by

$$\pi(n) \approx \frac{n}{\ln n},$$

calculate an approximation of the worst case number of trials that a naive cryptanalyst would require to factorize a 1024-bit RSA public key using an exhaustive factors search.

Exercise 5: Authenticated Encryption and Compression

1. In *authenticated encryption* we want to transmit a message that is both encrypted and authenticated. How to achieve this ?
2. Unrelated to that, if we want to both encrypt and compress a message, in what order should we do it ?

Exercise 6: One-Way Hash Function and MAC

For the practical part of this lab session, we will be using *stacktile*, a Web-based application that offers virtualized environments in a browser as a service.

The learning objective of this exercise is for students to get familiar with one-way hash functions and Message Authentication Code (MAC). After finishing the exercise, in addition to gaining a deeper understanding of the concepts, students should be able to use tools and write programs to generate one-way hash value and MAC for a given message.

Let's get started! Head out to this address `https://stacktile.io/org/marco`.

To get familiar with stacktile, we recommend that you first select the **INGI2347 on stacktile** introductory workflow.

Afterwards, please select the workflow titled **One-Way Hash Function and MAC** under the **Lab 3 - Cryptography** heading.

Please note that stacktile limits to 25 minutes the use of the service for unregistered visitors. Simply signup for service (it's free!) at `https://stacktile.io/` to overcome this limit.