Sicurezza e Privatezza

LAB 1 - Fondamenti di crittografia

The learning objective of this lab is for students to get familiar with elementary concepts in the secret-key encryption. After finishing the lab, students should be able to gain a first-hand experience on basic encryption algorithms as well as hash functions.

Students are invited to solve the following exercises, write the solutions in a .txt file which will contain their name, surname and University ID, and submit it to SPlab@di.unimi.it, with subject: Lab1NameSurname. Messages not conforming to these rules will be automatically rejected.

The first ten students who will respond correctly to the entire lab will receive a bonus of 0.5pt on the final grade.

Exercise #1: Base64

Base64 encoding schemes are commonly used when there is a need to encode binary data that needs be stored and transferred over media that are designed to deal with textual data. This is to ensure that the data remains intact without modification during transport. Base64 is used commonly in a number of applications including email via MIME, and storing complex data in XML or JSON. **Decode the following.**

```
VGhpcyBpcyBiYXNlNjQgZW5jb2RpbmcsIGlzIG5vdCBhIGNoaXBlciBidXQgb25seSBhIHd
heSB0byByZXByZXNlbnQgYmluYXJ5IGRhdGEgaW4gYW4gQVNDSUkgc3RyaW5nLg==
```

Exercise #2: Stream cipher

A **stream cipher** is an encryption algorithm that encrypts 1 bit or byte of plaintext at a time. It uses an infinite **stream** of pseudorandom bits as the key. For a **stream cipher** implementation to remain secure, its pseudorandom generator should be unpredictable and the key should never be reused. The following file crypt.txt (inside 02_stream_chiper folder) has been encrypted using the most famous stream cypher encoding algorithm with the key contained in file key.txt. **Recover the original plaintext.**

Exercise #3: Substitution cipher

It is well-known that monoalphabetic substitution cipher (also known as monoalphabetic cipher) is not secure, because it can be subjected to frequency analysis. In this lab, you are given a cipher-text that is encrypted using a monoalphabetic cipher; namely, each letter in the original text is replaced by another letter, where the replacement does not vary (i.e., a letter is always replaced by the same letter during the encryption). **Your job is to find out the original text using frequency analysis and the encryption key**. It is known that the original text is an Italian poetry. We do keep the spaces between words, so you can still see the boundaries of the words in the ciphertext. In real encryption using monoalphabetic cipher, spaces will be removed. We also keep capital letters.

Guidelines: using the frequency analysis, you can find out the plaintext for some of the characters quite easily. For those characters, you may want to change them back to its plaintext, as you may be able to get more clues. It is better to use capital letters for plaintext, so for the same letter, we know which is plaintext and which is ciphertext.

Hash functions

Exercise #4: Integrity checks

All files recived from an external source can be corrupted, for example when they are downloaded from the internet (due to transmission error), when a storage device faults or -in the worst scenario- when an attackers change it's contents.
One way to ensure that programs or files have not been damaged or tampered is to **calculate their hash value** and then **comparing it with the checksum provided** by the source. If the hash matches, then you have a reasonable degree of confidence that the file you downloaded is exactly the same as the one declared by it's source. If it is not, then the file has been changed in some way.
Inside 04_integrity_checks folder there is a PDF and it's pre-calculated hash, **can you check it's integrity**?

Exercise #5: Hash forcing

Write a python script which is able to determine a value i such that H(i) = `00000xxxxxxxxxxxxxxxx`, where H is the hash function SHA-1. The minimum number of leading zeroes accepted is 3. **The student has to provide the value of i.**

Exercise #6: Short password

Bruteforce attack is the **most powerful cracking mode**, it can try all possible character combinations as passwords. However, cracking with **exhaustive methods will never terminate** because of the number of combinations being too large. In this exercise we have applied some constraints to get the job done in a reasonable time.
The folder 06_short_password_cracking contains some hashed passwords. **Apply a brute force attack in order to recover the passwords plaintexts**, knowing that they are 5 characters long and contains only lowercase characters. The list of recovered passwords has to be sent.

Hints: for solving the above-mentioned problems it can be very useful to use the following python library: **hashlib**

Source code: Lib/hashlib.py
This module implements a common interface to many different secure hash and message digest algorithms. Included are the FIPS secure hash algorithms SHA1, SHA224, SHA256, SHA384, and SHA512 (defined in FIPS 180-2) as well as RSA's MD5 algorithm (defined in Internet ***RFC 1321****). The terms "secure hash" and "message digest" are interchangeable. Older algorithms were called message digests. The modern term is secure hash.*
More info https://docs.python.org/3/library/hashlib.html