Lab 09 - Cryptography (1)

1. Programming Setup
Use whatever OS and programming language you prefer.

2. PROBLEM – Monoalphabetic substitution using a pseudorandom permutation

All monoalphabetic substitution encryption algorithms do the same thing: compute a permutation of the alphabet somehow; they then apply that permutation to the message to compute the cryptogram.

Example: Caesar's cipher computes the permutation of the alphabet by circularly shifting the alphabet to the left with K positions (K is the key here, an integer, typical value is K=3); thus A will be replaced (substituted) by D, B by E etc.

Mathematically, this actually computes a permutation of the alphabet (considering the English alphabet A = 0, B = 1, … , Z = 25) by addition modulo 26 (the size of the alphabet):

$y = (x+K) \bmod 26$

Other, more complex methods exist, like the Affine cipher for example, which uses the following formula (the key here is the pair of integers (a,b)):

$y = (a*x+b) \bmod 26$

Clearly the best cipher in this class is the one that uses a RANDOM permutation of the alphabet.
(no formula for computing the permutation of the alphabet ! how long is the key, then?)
The problem here is that we need a TRNG to compute the permutation.

Next best option is to use a PSEUDORANDOM permutation of the alphabet; we must compute this permutation using a PRNG that is seeded with the "keyword" used for encryption/decryption.

2.1. The alphabet and the messages
Let us consider the alphabet of all possible bytes, consisting of 256 symbols (from 0x00 to 0xFF).
The [plaintext/cleartext] messages then, are any strings of bytes.
Let us consider the message to be a file (so a binary file, actually).

2.2. The key
Let us consider the most complex form of monoalphabetic substitution, using a random key.
The key K will be a random permutation of the alphabet.
We will generate the key starting from a "keyword" that is easy to remember.
The "keyword" is usually any string of printable characters.
We compute a "seed" for some PRNG using the "password" somehow; show some creativity here…
We set the seed of the PRNG to the computed "seed".
We generate a random permutation of the alphabet K using exactly 1024 random bytes obtained from the PRNG. Show some creativity here too…
K is the key to be used for encryption.

2.3. The Encryption
Take the message file M, open it in binary mode, read all of it into memory; we assume the input file is smaller than 1 MB.
Use the key K to apply the substitution, then write the result of the encryption into binary file C
User should be able to choose the filenames involved.

2.4. The Decryption
Compute the key K starting from the “keyword”.
Compute the inverse of the permutation K, let’s call this IK.
Take the input file C, open it in binary mode, read all of it into memory; we assume the input file is smaller than 1 MB.
Use the key IK to apply the inverse substitution, then write the result of the decryption into binary file M
User should be able to choose the filenames involved.

3. DELIVERABLES

3.1. Please prepare a file (DOCX, PDF) containing the following sections:

1. The process/algorithm you used to compute the seed for your PRNG from the keyword
2. The process/algorithm you used to compute the key from the first 1024 bytes obtained from the PRNG.
3. A screenshot of the running executable; Use your FULL NAME as the keyword, for this assignment.
   Also, display the KEY that was used (the pseudorandom permutation of the alphabet)
4. The Message is this file, “Lab_09.pdf” (Please provide a Hex view, the beginning only please)
5. The Cryptogram “Lab_09.enc”, after encrypting this Message with the key (Please provide a Hex view, the beginning only please)
6. The decrypted file “Lab_09_dec.pdf”, after decrypting “Lab_09.enc” with the same key (Please provide a Hex view, the beginning only please)

3.2. The source code