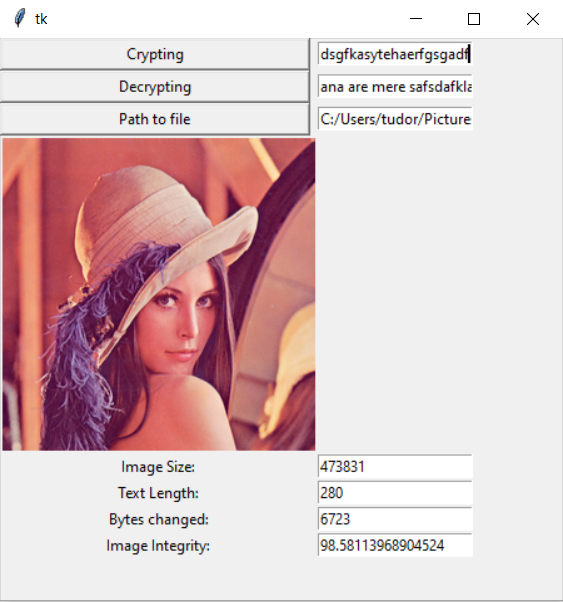
Steganografie

Interfața



Pentru implementarea interfeței am folosit limbajul python și în special biblioteca tkinter. În interfață avem trei butoane (Crypting, Decrypting și Path to file). Fiecare buton are implementată în spate câte o funcție sugestivă.

1. Funcția Browse\_picture:

Este aferentă butonului „Path to file” și are rolul de a prelua path-ul imaginii și a o prezenta pe interfață. Un alt rol este de a afisa dimensiunea imaginii.

1. Funcția Crypt\_process:

Este aferentă butonului „Crypting” și are rolul de a prelua mesajul din entry-ul corespunzătoar butonului și de a executa scriptul de inserare a mesajului. Tot această funcție calculează lungimea textului, numărul de biți schimbați și integritatea imaginii ((1-nr\_octeti\_modif/nr\_total\_octeti)\*100(%)).

1. Decrypt\_process:

Este aferentă butonului „Decrypting” și are rolul de a prelua mesajul în urma rularii procesului de Steganoanaliza.

***Steganografia*** nu trebuie confundata cu ***criptografia***. Acesta din urma face ca un mesaj sa devina indescifrabil, dar existenta lui este vizibila, iar steganografia ascunde existenta mesajului si nu mesajul. Motivele folosirii steganografiei este usurinta detectarii criptarilor de date din interiorul fisierelor (prin intermediul criptografiei ) . Acestea sunt extrem de vizibile avand o entropie mare de caractere in interiorul lor.

Mesaj Criptat sau compresat:



Mesaj sub forma normal:



Algortimul de criptare presupune scrierea unui mesaj intr-o imagine scopul fiind ascunderea de informatii si securitatea. Scopul este inserarea mesajului fara a modifica imaginea din punct de vedere vizual.

Metoda aleasa este “Metoda celui mai putin semnificativ bit”.Inserarea mesajului prin modificarea celui mai putin semnificativ bit este o metoda eficienta dar nu foarte greu de decriptat. Avantajul este ca avem 50% sanse ca la iterarea prin bitii unui character sa nu fie nevoie sa modificam bitul din imagine (el poate fi deja 0 sau 1).

Explicare algoritm:

Algoritmul primeste din Interfata calea absoluta spre imaginea utilizata si mesajul.

Functioneaza doar pentru imagini RGB!!!

* Semnificatia variabilelor folosite:

1. Img -> imaginea retinuta in interiorul unui array
2. Mesaj -> mesajul trimis din interfata
3. MASK -> masca pentru extragerea bitului cel mai nesemnificativ
4. X,Y -> indecsi pentru parcurgerea imaginii
5. Counter -> variabila folosita pentru a aranja corect bitii in octet si pentru a delimita fiecare octet obtinut

Parcurgerea mesajului:

for (auto elem : mesaj)

Extragerea bitilor din caracter:

BYTE bit\_of\_interest = elem & MASK;

bit\_of\_interest = bit\_of\_interest >> (counter - 1);

MASK = MASK >> 1;

Inserarea aceluiasi bit pe Pixel(RGB):

BYTE& oct1 = img.at<BYTE>(y, x);

BYTE& oct2 = img.at<BYTE>(y, x + 1);

BYTE& oct3 = img.at<BYTE>(y, x + 2);

oct1 = ((oct1 >> 1) << 1) | (bit\_of\_interest);

oct2 = ((oct2 >> 1) << 1) | (bit\_of\_interest);

oct3 = ((oct3 >> 1) << 1) | (bit\_of\_interest);

Inserarea unui terminator de sir:

BYTE& oct1 = img.at<BYTE>(y, x);

BYTE& oct2 = img.at<BYTE>(y, x + 1);

BYTE& oct3 = img.at<BYTE>(y, x + 2);

oct1 = 'R';

oct2 = 'C';

oct3 = 'T';

**Steganaliza**

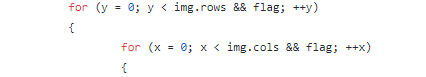
Algoritmul de steganaliza urmareste sa restaureze mesajul inserat in imagine folosind un rationament invers fata de programul de criptare. Astfel, decriptarea presupune extragerea bitilor cei mai nesemnificativi a pixelilor intalniti pana la terminatorul sirului de octeti criptati si gruparea lor in octeti pentru reconstituirea informatiei ascunse.

Explicarea algoritmului:

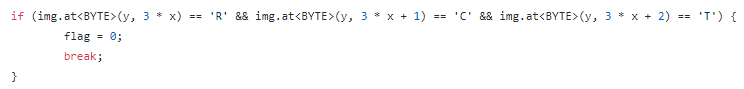
* Semnificatia variabilelor folosite:

1. MASK -> masca pentru extragerea bitului cel mai nesemnificativ
2. counter -> variabila folosita pentru a aranja corect bitii in octet si pentru a delimita fiecare octet obtinut
3. octet, octet\_rezultat -> 2 variabile folosite pentru reconstruirea corecta a octetilor
4. flag -> un flag care opreste parcurgerea pixelilor la intalnirea valorilor cu rol de terminator de sir
5. mesaj\_rezultat -> un string unde va fi la final informatia decriptata

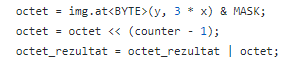
* Parcurgerea pixelilor imaginii



* Verificarea terminatorului de sir



* Extragerea bitului cel mai nesemnificativ din pixel si pozitionarea lui in locul potrivit



* Concatenarea caracterului obtinut la mesajul intreg si reinitializarea unor variabile de control

