

Această tehnică întărește securitatea deoarece mesajul secret este împărțit în mai multe fragmente, fiind mult mai dificil de spart (mai ales că în lucrarea originală, asupra acestuia a fost aplicat un algoritm de criptare) și, folosind clustering-ul, obținem o imagine cu zgomot, care este mai rezistentă la atacurile potențialilor terți malicioase, aceștia nemaiputând să distingă între poza originală și pixelii care au fost modificați.

Pe de altă parte, culorile nu sunt distribuite egal și ne așteptăm ca unele clustere să fie mai mari decât altele. Astfel, dacă avem un mesaj de o dimensiune mai ridicată și facem împărțirea acestuia în trei bucăți egale, așa cum este descris în algoritmul original, este posibil să avem pierdem din mesajul transmis, de aceea, o posibilă îmbunătățire ar fi să împărțim mesajul în trei bucăți direct proporționale cu dimensiunea celor trei clustere obținute; sau, alternativ, să avem mai multe clustere.

2.3.5. Algoritmul LSB propus

În cele ce urmează, vom propune un algoritm de steganografie bazat pe tehnica LSB. Motivul alegerii acestei metodologii se datorează faptului că, în literatura de specialitate, este bine-cunoscut că un algoritm steganografic folosind LSB este mai vulnerabil decât alte tehnici precum DCT (eng. original: *Discrete Cosine Transform*)⁶ sau DWT (eng. original: *Discrete Wavelet Transform*), doar pentru a numi câteva [5]. Prin urmare, această idee contribuie la steganografia LSB, într-o încercare de a dezvolta un algoritm mai sigur, solid și robust, menținându-și în același timp reputația ca o implementare relativ simplă, în comparație cu alte metode.

Un prim pas logic constă în criptarea mesajului. În acest fel, se face legătura dintre criptografie și steganografie, iar securitatea algoritmului este îmbunătățită substanțial. Pasul de criptare protejează împotriva posibilității extragerii neautorizate a datelor din fișierul de acoperire, prin atacuri de steganaliză⁷. Ne putem gândi că acest aspect servește drept ultimă soluție în cazul în care un atac de steganaliză are succes. Pentru criptarea mesajului secret vom folosi algoritmul AES, un algoritm simetric, rapid și care se realizează fără pierderea anumitor date din memorie.

⁶ DCT este cel mai folosit algoritm de compresie lossy și este utilizat pentru fotografiile de tip .JPEG.

⁷ Steganaliza este studiul detectării mesajelor ascunse folosind tehnici de steganografie și este analogul criptanalizei pentru criptografie. [65]

2.3.5.1. AES

AES (eng. original: *Advanced Encryption Standard*) este un subset al familiei de algoritmi Rijndael [25], algoritmi este de tip bloc (eng. original: *block cipher*). AES prezintă următoarele caracteristici:

- Blocul are lungimea de 128 de biți;
- Cheia de criptare are lungimea variată și poate lua valori din mulțimea {128, 192, 256} (reprezentare pe biți).

AES a fost selectat de guvernul SUA pentru a proteja informațiile clasificate și, în zilele noastre, este folosit pe tot mapamondul cu scopul de a cripta date cu caracter sensibil, fiind standardul pentru securitatea cibernetică. Institutul Național de Standarde și Tehnologii (NIST din eng. original) a început cercetările pentru AES în anul 1997, când, în urma examinărilor și a testelor, a fost observat că DES (eng. original: *Data Encryption Standard*) a început să devină vulnerabil la atacurile cu forță brută [26].

Pentru o securitate optimă, vom folosi cheia de 256 de biți (cea mai mare oferită de standard). AES funcționează, mai degrabă, pe bytes decât pe biți. Algoritmul are lungimea de bloc de 128 de biți, adică 16 bytes, care sunt stocați într-o matrice de $4 \cdot 4$, astfel:

$$\begin{bmatrix} \text{byte}_0 & \text{byte}_4 & \text{byte}_8 & \text{byte}_{12} \\ \text{byte}_1 & \text{byte}_5 & \text{byte}_9 & \text{byte}_{13} \\ \text{byte}_2 & \text{byte}_6 & \text{byte}_{10} & \text{byte}_{14} \\ \text{byte}_3 & \text{byte}_7 & \text{byte}_{11} & \text{byte}_{15} \end{bmatrix}$$

Dimensiunea cheii utilizate specifică numărul de runde de transformări succesive care convertesc textul original, în mesajul criptat. Pentru o cheie de dimensiune 256, ca cea aleasă, sunt necesare 14 runde. Fiecare rundă constă în patru operații (*SubBytes*, *ShiftRows*, *MixColumns*, *AddRoundKey*), iar algoritmul este prezentat așa cum este el descris în [27]:

- 1) *KeyExpansion*: cheile de rundă (eng. original: *round key*) sunt derivate din cheia originală de criptare, fiecare depinzând de cea creată anterior, folosind algoritmul de „AES Key Schedule”.
- 2) *AddRoundKey*: între starea inițială (mesajul clar) și prima cheie de rundă se realizează operația de XOR-are.
- 3) **13 Runde a câte patru Operații:**
 - i. *SubBytes*: fiecare octet este înlocuit de un altul conform tabelii de substituție S-box;

- ii. *ShiftRows*: este un pas de transpoziție în care ultimele trei rânduri ale matricei sunt shift-ate cu un anumit număr de pași (dacă începem număratoarea rândurilor de la zero, atunci, valorile primului rând sunt shift-ate la stânga cu o coloană; pe al doilea rând cu două coloane și pe al treilea cu trei, iar rândul 0 rămâne neschimbat):

$$\begin{bmatrix} \text{byte}_0 & \text{byte}_4 & \text{byte}_8 & \text{byte}_{12} \\ \text{byte}_1 & \text{byte}_5 & \text{byte}_9 & \text{byte}_{13} \\ \text{byte}_2 & \text{byte}_6 & \text{byte}_{10} & \text{byte}_{14} \\ \text{byte}_3 & \text{byte}_7 & \text{byte}_{11} & \text{byte}_{15} \end{bmatrix} \rightarrow \begin{bmatrix} \text{byte}_0 & \text{byte}_4 & \text{byte}_8 & \text{byte}_{12} \\ \text{byte}_5 & \text{byte}_9 & \text{byte}_{13} & \text{byte}_1 \\ \text{byte}_{10} & \text{byte}_{14} & \text{byte}_2 & \text{byte}_6 \\ \text{byte}_{15} & \text{byte}_3 & \text{byte}_7 & \text{byte}_{11} \end{bmatrix}$$

- iii. *MixColumns*: fiecare coloană a matricei stare este înmulțită cu

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}, \text{ obținându-se o nouă matrice;}$$

- iv. *AddRoundKey*: XOR-are între cheia de rundă corespunzătoare și matrice.

4) **A 14-a Rundă (Runda Finală):**

- i. *SubBytes*
- ii. *ShiftRows*
- iii. *AddRoundKey*

Cifrările de tip bloc au două proprietăți cheie care îi fac imuni în fața atacatorilor și anume confuzia și difuzia. Formal, confuzia înseamnă că inputul și outputul nu au nicio conexiune între ele și difuzia se referă la faptul că modificări mici ale inputului au un impact gigantic asupra outputului [28]. În AES, confuzia este adăugată la pasul *SubBytes*, iar difuzia se produce la pașii *ShiftRows* și *MixColumns*. De asemenea, AES este eficient și din punct de vedere computațional deoarece criptarea, respectiv decriptarea, nu fac uz de multe resurse.

În continuare, pentru a îmbunătăți securitatea algoritmului LSB, vom lua în considerare faptul că este slab împotriva atacurilor de steganaliză din pricina pixelilor modificați care sunt așezați unul după altul. De aceea, propunem să îi amestecăm, pentru a putea folosi întreaga capacitate a fotografiei de acoperire și a crește gradul de siguranță.

2.3.5.2. Algoritmul modern de amestecare Fisher-Yates (versiunea lui Durstenfeld)

După cum am menționat și anterior, vom genera o secvență unică, aleatoare de pixeli a unei imagini. Secvența poate fi creată manual și de utilizator. Aceasta va acționa drept cheie steganografică, iar fără ea nu vom putea recupera mesajul secret. Adăugăm astfel un nou strat de securitate algoritmului.

Pentru a obține acest lucru, vom folosi versiunea modernă a algoritmului de amestecare Fisher-Yates, introdusă de Richard Durstenfeld în 1964 [29] și popularizată de Donald E. Knuth, în 1969, în cartea sa *The Art of Computer Programming* [30], deoarece aceasta are o complexitate de $O(n)$, față de $O(n^2)$ a implementării naive. Algoritmul de permutare a pixelilor este:

```
Date de Intrare: n = numărul total de pixeli ai imaginii de acoperire;  
                  m = numărul total de bytes ai mesajului secret * 3; (pentru că  
                  avem nevoie de 3 pixeli pentru LSB)  
Date de Ieșire: vectorul P de pixeli amestecați;  
  
w = 0;  
while w ≤ n-1 do  
    aux[w] = w; w++;  
end;  
  
w = 0;  
while w ≤ m-1 do  
    j = valoare random din intervalul [0; m-w-1];  
    swap(aux[w], aux[j]);  
end;  
  
P[m]; w = 0;  
while w ≤ m-1 do  
    P[w] = aux[n-1-w];  
end;
```

În esență, folosim acest algoritm pe post de PRNG (eng. original: *Pseudo-Random Number Generator*). Întrucât secvența numerelor generate aleatoriu depinde, în întregime, de input, acest tip de PRNG nu poate produce mai multe permutări distincte decât numărul maxim de stări posibile distincte. Chiar și atunci când numărul stărilor posibile depășește numărul

permutărilor, natura neregulată a generării produce ca unele permutări să apară mai des decât altele. Astfel, pentru a minimiza bias-ul, numărul total posibil de stări ar trebui să depășească numărul de posibile PRNG-uri cu mai multe ordine [31]. În cazul nostru, ar trebui să ne asigurăm că $Nr.Total\ Pixels / (3 \cdot Nr.Bytes\ Mesaj)$ este suficient de mare, iar, de cele mai multe ori, acest obiectiv este îndeplinit.

Ca să putem folosi pixelii trebuie să cunoaștem locația lor în poză, deci să știm rândul și coloana. Aceste informații se obțin ușor. Dacă considerăm P ca fiind numărul pixelului din secvența generată aleatoriu și N , M reprezintă numărul de linii, respectiv de coloane ale pozei, atunci:

$$linia = \left\lfloor \frac{P}{M} \right\rfloor; coloana = P \% M$$

După ce vom folosi secvența pentru a încorpora mesajul secret în pixelii corespunzători, aceasta va fi codificată folosind AES și va fi transmisă împreună cu poza de acoperire pentru procesul de decodificare.

2.3.5.3. Criptarea/Decriptarea

Pe scurt, algoritmul propus se prezintă astfel:

Date de Intrare: I = imaginea; M = mesajul secret; P1 = parolă pentru mesajul secret; P2 = parolă pentru secvența de Pixeli

Date de Ieșire: S = imaginea modificată; Pix = secvența criptată de pixeli

Pas 1: Se criptează mesajul secret (M) cu AES folosind parola secretă, P1, rezultând N.

Pas 2: Se generează secvența aleatoare de pixeli, P, pornind de la lungimea mesajului secret criptat și de la mărimea imaginii de acoperire, I.

Pas 3: Se încorporează N în pixelii generați aleatoriu în P, folosind LSB și obținem S.

Pas 4: Se criptează P cu AES, folosind P2 și obținem Pix.

Pas 5: Returnăm S și Pix.

Pentru decriptare, se parcurg pașii în ordine inversă, astfel:

Date de Intrare: S = imaginea modificată; Pix = secvența criptată de pixeli;
P1 = parolă pentru mesajul secret; P2 = parolă pentru secvența de Pixeli

Date de Ieșire: M = mesajul secret

Pas 1: Se decriptează Pix folosind P2 (AES) și obținem P.

Pas 2: Din S extragem biții din pixelii marcați, din P, și obținem N.

Pas 3: Se decriptează N folosind P1 (AES) și obținem M.

Pas 4: Se returnează M.

Este important ca cele două parole, pentru mesajul secret și pentru secvența aleatoare de pixeli, să fie transmise într-un mod sigur pe canalul de comunicații, altfel algoritmul devine inutil. Ca recomandare, acestea ar trebui să fie diferite ca să nu faciliteze munca unei terțe malițioase. Există mai multe protocoale sigure din care utilizatorul poate alege. Dintre acestea amintim Diffie-Hellman cu semnături digitale sau RSA [32]. Nu vom intra în detalii deoarece nu constituie tema lucrării de față.

Metoda propusă nu este foarte eficientă din punct de vedere al memoriei deoarece trebuie să trimitem secvența criptată de pixeli, alături de imaginea rezultată prin tehnica steganografică, dar facem acest compromis pentru a obține o mai bună securitate.

2.3.6. Algoritmul DCT

Tehnicile steganografice pot fi clasificate în metode aplicate în domeniul spațial și metode aplicate în domeniul de frecvență [33].

Până acum au fost prezentate doar metode care făceau uz de domeniul spațial și anume algoritmi de tip LSB, deoarece procesarea se aplică direct asupra valorilor pixelilor din imaginea de acoperire. Aceste metode sunt utilizate pe scară largă datorită simplității lor. Cu toate acestea, sunt extrem de vulnerabile la modificări, chiar și mici, care au fost aplicate fișierului de acoperire. Un atacator poate executa pur și simplu tehnici de procesare a semnalului pentru a distruge în întregime informațiile secrete. În multe cazuri, chiar și micile modificări rezultate din sistemele de compresie cu pierderi (eng. original: *lossy compression*) produc pierderea mesajului secret.

3. Analiza performanței

Pentru analiza performanțelor, au fost implementați algoritmi descriși în secțiunile 2.3.1., 2.3.3., 2.3.4., 2.3.5., 2.3.6. și anume algoritmi LSB Simplu, LSB folosind conceptul de Liste Înlănțuite, LSB folosind K-Means Clustering, algoritmul propus de LSB și DCT. Deoarece primii 4 algoritmi primesc ca input și pot funcționa doar utilizând imagini de tip .png (sau alt format de tip *lossless*), algoritmul pentru DCT folosește o versiune modificată de Jsteg, unde primește ca input tot fișiere cu extensia .png; de aceea pașii de RLE și codificare Huffman nu au mai fost necesari în implementarea algoritmului, deoarece aceia sunt specifici compresiei de tip JPEG.

3.1. Tehnologii utilizate și justificarea selectării acestora

Algoritmi au fost implementați folosind limbajul Python3 [52], iar ca mediu de dezvoltare, a fost utilizată platforma Google Colab [53].

Python a fost ales deoarece facilitează lucrul cu fișierele, fie ele text sau imagine (librăriile CV2 [54] sau Pillow [55], în funcție de operațiile care sunt necesare asupra fotografiilor; CV2 este mult mai rapidă și complexă, în cadrul căreia există funcții pentru calculul coeficienților DCT, funcție pentru transformarea din RGB în YcbCr și invers etc..., spre deosebire de biblioteca Pillow care este folosită pentru operații de tipul tăiere și redimensionare de fotografii). Algoritmul AES poate fi implementat prin intermediul librăriei Pycrypto [56], iar biblioteca hashlib, care vine preinstalată cu Python, este folosită pentru funcțiile hash, în acest fel verificând validitatea unei parole. Bitstring [57] este un modul Python conceput pentru a ajuta la crearea și analiza datelor binare într-un mod cât mai simplu și natural cu putință.

De asemenea, Python este bine cunoscut ca fiind folosit pentru tehnici de învățare automată, cum este algoritmul K-Means care a fost implementat folosind librăria sklearn [58].

Google Colab, sau, pe scurt, Colab, a fost utilizat ca mediu de dezvoltare deoarece este o unealtă open-source, dezvoltată de Google Corporation, care oferă acces la spațiu Cloud, cu resurse GPU gratuite și puternice, dar și o memorie de disc destul de mare de aproximativ 70 GB.

3.2. Măsurarea performanței

Pentru testele de performanță au fost selectate următoarele fotografii, des utilizate în studiul asupra imaginilor digitale:

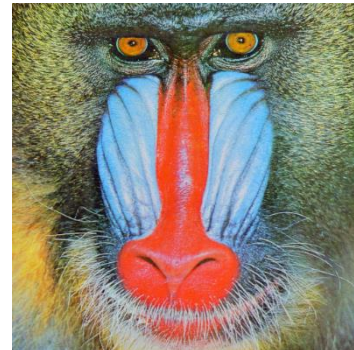
Figură 3.2. 1 - Lenna.png [12]



Figură 3.2. 2 – Pepper.png [59]



Figură 3.2. 3 – Baboon.png [59]



Prima fotografie are dimensiunea 225x225, a doua are dimensiunea 320x320 și ultima 512x512. Prima fotografie a fost selectată special cu dimensiuni nedivizibile la 8 pentru a vedea cum se compară imaginea rezultată din algoritmul DCT modificat cu restul.

Mesajul secret a fost ales să aibă dimensiunea de aproximativ 2.5 KB, pentru a observa cum se descurcă algoritmi în fața unui mesaj secret de dimensiuni ridicate.

Din punct de vedere vizual, obținem următoarele rezultate:

- LSB Simplu:



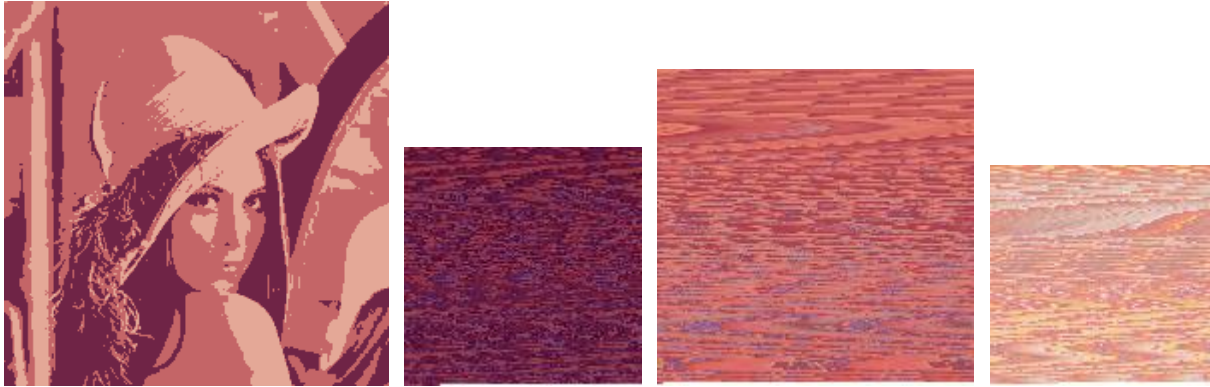
Figură 3.2. 4 – Stego LSB Simplu

- LSB Liste Înlănțuite:



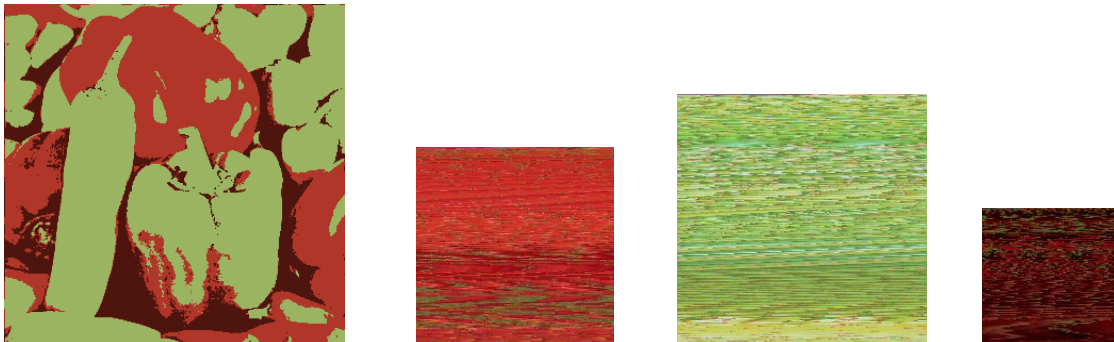
Figură 3.2. 5 – Stego LSB Liste

- LSB K-Means Clustering (după cum am menționat, vom obține 3 fotografii pentru fiecare imagine introdusă):
 - Pentru Lenna.png:



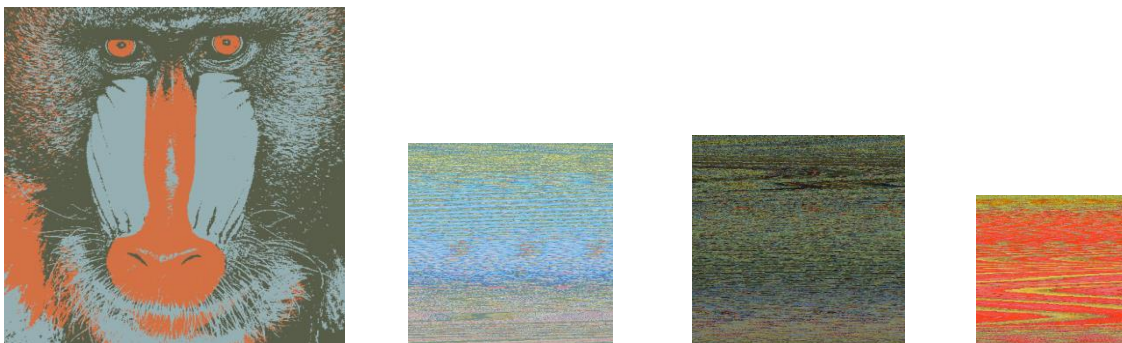
Figură 3.2. 6 – LSB K-Means Lenna.png

- Pentru Pepper.png:



Figură 3.2. 6 – LSB K-Means Pepper.png

- Pentru Baboon.png:



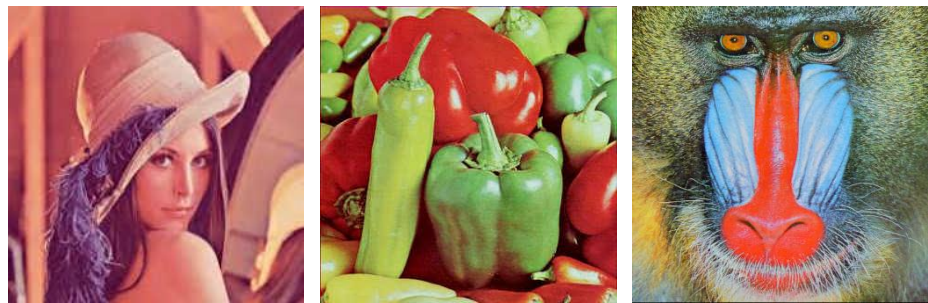
Figură 3.2. 7 – LSB K-Means Fruits.png

- LSB Algoritm Propus:



Figură 3.2. 8 – Stego LSB Original

- DCT:



Figură 3.2. 9 – Stego DCT

După cum se poate observa, cu excepția LSB-ului prin K-Means care returnează pozele clusterelor și DCT-ului atunci când este dat ca input o fotografie cu dimensiuni nedivizibile cu 8, caz în care algoritmul va redimensiona fișierul de acoperire, adăugând padding, ceea ce rezultă într-o poză mai neclară, ceilalți algoritmi întorc imagini identice cu cele oferite ca input, deci, obiectivul steganografic de a rămâne vizual nedetectabil, este îndeplinit.

Dacă luăm în calcul capacitatea de încorporare a mesajului, algoritmiile LSB Simplu, LSB cu Liste Înlănțuite și algoritmul LSB propus, au trecut testul cu brio: aceștia au înglobat întregul mesaj secret în imaginea de acoperire, fără pierderi de date. În contrast, LSB folosind K-Means Clustering nu a reușit să codeze tot fișierul text în poza Pepper.png, deoarece ultimul cluster era prea mic. Dacă, în schimb de 3 cluster, folosim 5, reușim să rezolvăm această problemă. În mod similar, dacă împărțim mesajul în funcție de dimensiunile celor trei cluster, evităm problema de capacitate. DCT are aceeași problemă: pentru Lenna.png nu reușește să codeze tot mesajul secret.

Ca timp de execuție, algoritmi au următoarele performanțe:

Algoritmi	Imagini	Codare	MedieC	Decodare	MedieD
LSB Simplu	<i>Lenna.png</i>	0.04	0.066	0.011	0.013
	<i>Pepper.png</i>	0.07		0.014	
	<i>Baboon.png</i>	0.09		0.015	
LSB Liste	<i>Lenna.png</i>	0.15	0.28	0.047	0.148
	<i>Pepper.png</i>	0.24		0.048	
	<i>Baboon.png</i>	0.45		0.053	
LSB K-Means	<i>Lenna.png</i>	0.58	1.61	0.014	0.018
	<i>Pepper.png</i>	1.35		0.019	
	<i>Baboon.png</i>	2.92		0.022	
LSB Propus	<i>Lenna.png</i>	0.29	0.49	0.172	0.223
	<i>Pepper.png</i>	0.41		0.213	
	<i>Baboon.png</i>	0.78		0.285	
DCT	<i>Lenna.png</i>	1.57	2.87	0.687	4.698
	<i>Pepper.png</i>	2.96		1.588	
	<i>Baboon.png</i>	4.09		2.423	

Figură 3.2. 10 – Tabel Timpi Execuție

Se poate observa că metoda propusă este destul de rapidă în faza de codare, ocupând mai mult timp în etapa de decodare. Datorită operațiilor de criptare și permutare a pozițiilor pixelilor ne așteptăm la o creștere a timpului de execuție, dar remarcăm că se compară destul de bine cu timpii celorlalți algoritmi.

PSNR (eng. original: *Peak Signal to Noise Ratio*) este metrica primară utilizată pentru a măsura cât de perceptibil a fost zgomotul rezultat, cauzat de algoritm în rezultatele colectate [60]. PSNR este o metrică de eroare și este considerat a fi raportul dintre forța maximă a unui semnal digital și cantitatea de *noise* nedorit, care îl afectează. O valoare mai mare de PSNR implică o distorsiune mai mică. Formula de calcul este:

$$PSNR = 20 \cdot \log_{10} \frac{MAX}{\sqrt{MSE}} \quad (\text{Ecuația 3.2.1})$$

unde, *MAX* este numărul maxim de pixeli al imaginii de acoperire.

MSE (eng. original: *Mean Square Error*) măsoară media pătratului „erorii”; eroarea fiind suma cu care estimatorul diferă față de cantitatea de estimat [61]. Cu cât MES are o valoare mai mică, cu atât eroarea este mai mică. Dată o imagine I , fără zgomot, de dimensiune $m \times n$ și aproximarea ei zgomotoasă K , MSE este calculat folosind:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (\text{Ecuatia 3.2.2})$$

SSIM (eng. original: *Structural Similarity*) este folosit pentru a măsura asemănarea dintre două fotografii, din punct de vedere al percepției. Diferența dintre această metodă și tehnicile menționate anterior este aceea că PSNR și MSE calculează erori absolute, în schimb, SSIM este un model bazat pe percepție, care consideră degradarea imaginii ca o schimbare percepută în informațiile structurale, încorporând totodată fenomene perceptive importante, printre care enumerăm atât termeni de mascare a luminozității, cât și termeni de mascare a contrastului. Informația structurală este ideea că pixelii au interdependențe puternice, mai ales atunci când sunt apropiați spațial. Aceste dependențe poartă informații importante despre structura obiectelor din scena vizuală. SSIM poate lua valori între -1 și 1: un scor egal cu 1 înseamnă că cele două fotografii sunt foarte asemănătoare și -1 reprezintă contrariul. Indicele SSIM este calculat pe diferite blocuri ale unei imagini. Măsura dintre două astfel de blocuri x și y , cu dimensiunea comună de $N \times N$, este [62]:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (\text{Ecuatie 3.2.3})$$

$$c_1 = (k_1L)^2; c_2 = (k_2L)^2; L = 2^{bits \text{ per pixel}} - 1; k_1 = 0.01; k_2 = 0.03$$

unde, μ_x este media lui x , μ_y a lui y , σ_x^2 și σ_y^2 varianța lui x , respectiv y , σ_{xy} este covariația lui x și y , c_1 și c_2 sunt două variabile pentru stabilizarea diviziunii cu numitor slab, iar L reprezintă intervalul dinamic al valorilor pixelilor.

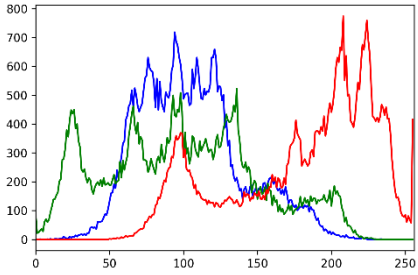
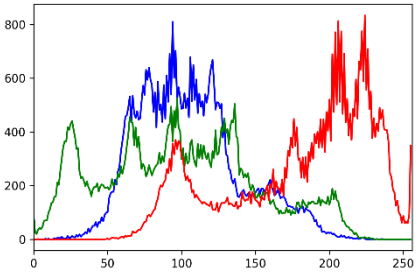
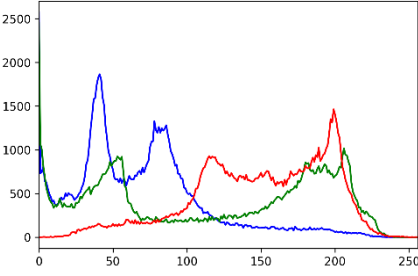
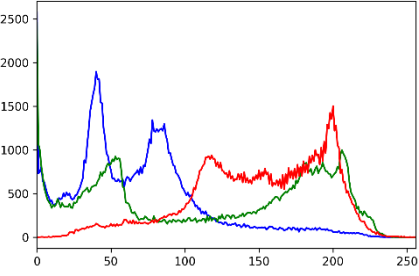
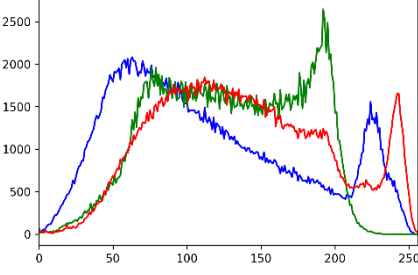
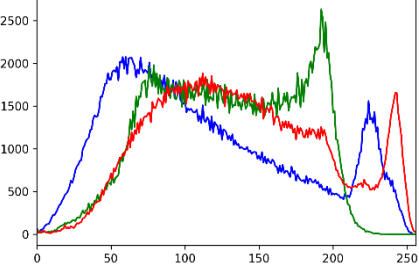
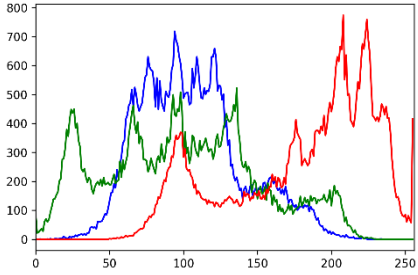
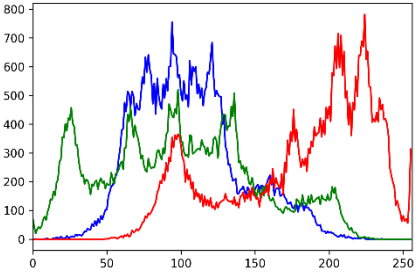
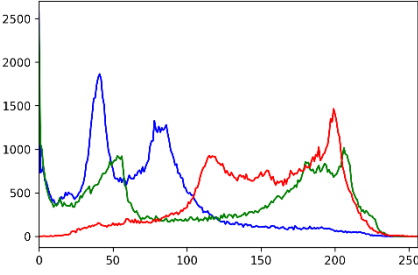
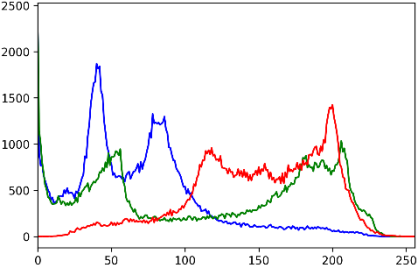
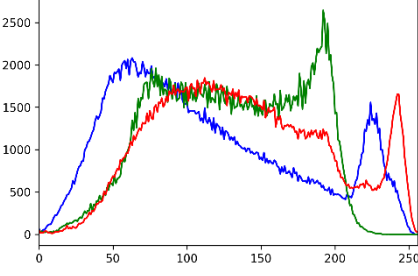
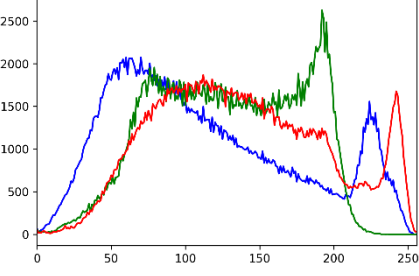
Algoritmi	Fotografii		MSE	Medie	PSNR	Medie	SSIM	Medie
LSB Simplu	<i>Lenna.png</i>		1.760	0.603	45.673	58.206	0.994	0.997
	<i>Pepper.png</i>		0.037		62.419		0.999	
	<i>Baboon.png</i>		0.014		66.527		0.999	
LSB Liste	<i>Lenna.png</i>		1.886	0.674	45.371	55.238	0.993	0.997
	<i>Pepper.png</i>		0.100		58.122		0.999	
	<i>Baboon.png</i>		0.038		62.222		0.999	
LSB K-Means	<i>Lenna.png</i>	<u>Cluster1</u>	0.050	0.047	61.083	62.521	0.999	0.998
		<u>Cluster2</u>	0.091		58.516		0.999	
		<u>Cluster3</u>	0.105		57.906		0.998	
	<i>Pepper.png</i>	<u>Cluster1</u>	0.072		59.514		0.997	
		<u>Cluster2</u>	0.023		64.458		0.998	
		<u>Cluster3</u>	0.038		62.257		0.999	
	<i>Baboon.png</i>	<u>Cluster1</u>	0.012		67.098		0.998	
		<u>Cluster2</u>	0.011		67.443		0.999	
		<u>Cluster3</u>	0.023		64.430		0.998	
LSB Propus	<i>Lenna.png</i>		1.835	0.646	45.493	56.111	0.994	0.997
	<i>Pepper.png</i>		0.074		59.397		0.999	
	<i>Baboon.png</i>		0.029		63.444		0.999	
DCT	<i>Lenna.png</i>		-	41.634	-	32.537	-	0.877
	<i>Pepper.png</i>		21.162		34.875		0.924	
	<i>Baboon.png</i>		62.106		30.199		0.830	

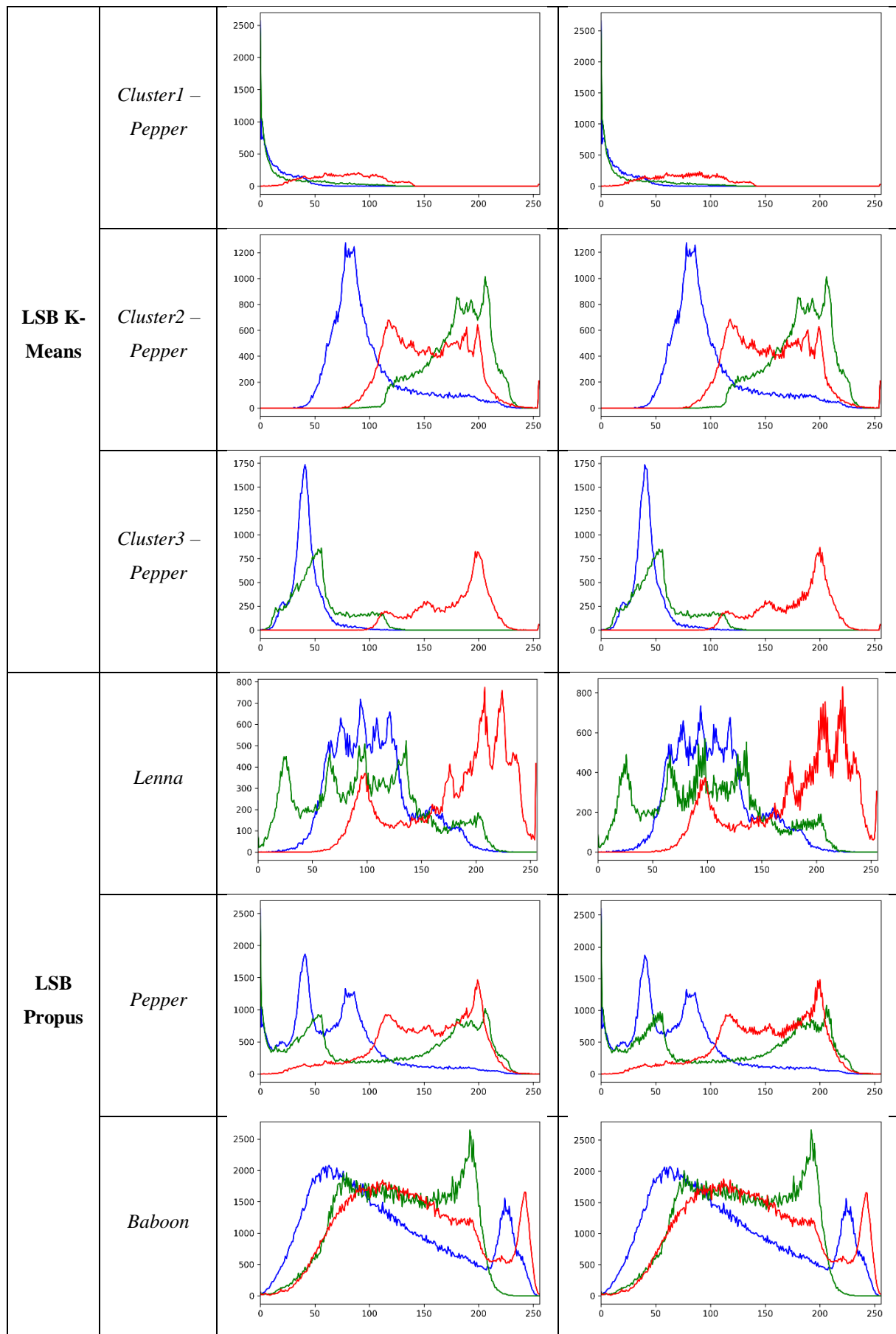
Figură 3.2. 11 – Tabel Performanțe Execuție

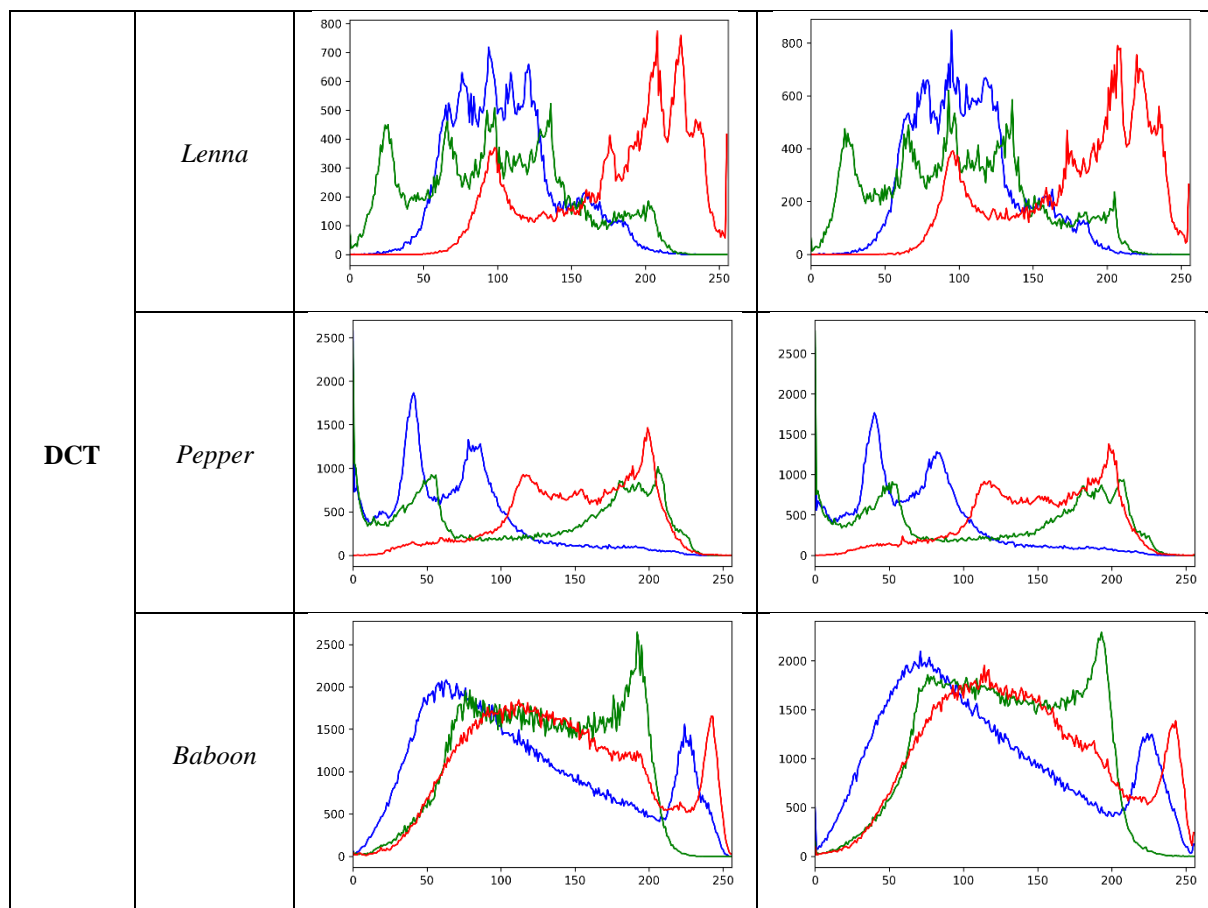
La DCT, nu am putut verifica metricile de performanță pe *Lenna.png* deoarece algoritmul redimensionează orice fotografie care nu are dimensiunile divizibile cu opt și nu puteam să comparăm două imagini cu proporții diferite. În general, algoritmul DCT oferă performanțele cele mai slabe. Era și de așteptat în condițiile în care acesta realizează o compresie a imaginilor date ca input.

Algoritmul propus are performanțe bune în comparație cu celelalte metode prezentate. Remarcăm că performanțele acestuia sunt direct proporționale cu dimensiunea fotografiei și a mesajului secret introdus.

În final, analizăm histogramele:

Algoritm	Fotografii	Poză Originală	Poză Stego
LSB Simplu	<i>Lenna</i>		
	<i>Pepper</i>		
	<i>Baboon</i>		
LSB Liste	<i>Lenna</i>		
	<i>Pepper</i>		
	<i>Baboon</i>		





Figură 3.2. 12 – Tabel Histograme

După cum putem remarca, histogramele prezintă o deviere minimală din perspectiva modificărilor aduse pixelilor în urma aplicării algoritmilor. Algoritmul propus are performanțe bune alături de celelalte metode prezentate. Remarcăm că performanțele acestuia sunt direct proporționale cu dimensiunea fotografiei și a mesajului secret introdus.