בס"ד

# דוח מסכם- פרויקט אבטחת נתונים


זיו גימני - 207354580
לאה ברודסקי - 318191764
יאנה ענהאל צ'יצ'קין-321271033
מנו ריי -313932832


המאמר שבחרנו- **Hiding data in images using steganography techniques with compression algorithms**

קישור לגיטהאב- **https://github.com/Ziv-Gimani/Steganography_Group10**

במאמר מוצע אלגוריתם משולב DCT ו- LSB על מנת לשפר את יעילות הסטגנוגרפיה.
לכן בקוד שלנו ביצענו שילוב של שני האלגוריתמים על ידי מימוש אלגוריתם DCT על תמונה מסויימת ואז על אותה תמונה שהוצפנה באמצעות DCT ביצענו הצפנה באמצעות LSB.

## השיפורים שהצענו למאמר:

**1. שימוש באלגוריתם RSA כדי לאבטח את ההודעות המועברות בערוץ התקשורת.**
האלגוריתם שמוצע במחקר לא מתייחס לאבטחת ערוץ התקשרות אלא רק בשיטת ההטמעה של המידע הסודי בתמונה. שימוש בRSA לאבטחת ההודעות משפרת את האבטחה.

**2. שימוש במדד SSIM.**
SSIM משמש להערכה של עיוות התמונה בתמונה המקורית בדומה למדדים שבמאמר שלנו PSNR ו-MSE אך מדד זה מראה תוצאות טובות יותר לעין האנושית.

הנוסחה של SSIM:

$$SSIM(x,y)=\frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)}$$

הסבר הנוסחה:
הנוסחה מחשבת את מדד הדמיון המבני בין 2 תמונות נתונות שהוא ערך בין 1 למינוס 1 . ערך של 1 +מציין ששתי התמונות הנתונות דומות מאוד או זהות. ערך 1 - מציין ששתי התמונות הנתונות שונות מאוד.

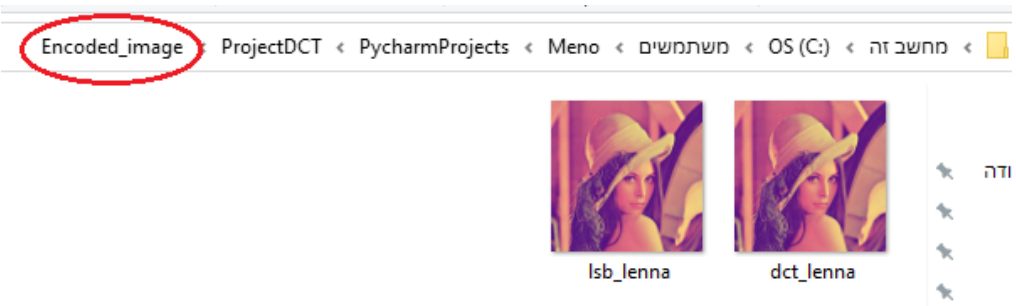
<u>שיפור באמצעות שימוש במדד SSIM שאותו נשלב בקוד הקיים:</u>
<u>לאחר השוואה בין SSIM לבין MSE - המסקנה היא :</u>
אמנם SSIM איטי יותר אך הוא מסוגל לתפוס את השינוי במידע המבני של התמונה – ע"י השוואת אזורים <u>מקומיים</u> של התמונה , בניגוד ל MSE שהוא אומנם מהיר יותר אך הוא מוחל באופן גלובלי.
לכן בחרנו לשפר את MSE בSSIM כי הוא ייתן לנו תוצאות טובות יותר אך נאבד מעט יעילות בביצועים.

3. **<u>שיפור אלגוריתם LSB-</u>** במאמר עושים שימוש באלגוריתם LSB סטנדרטי שבחישוב שלו עושים שימוש במודולו 2 . ניתן לשפר את הנוסחה כך שנממש את האלגוריתם עם מודולו של מספר גדול יותר לדוגמא מודולו 3 . זה יגרום לצמצום מספר השינויים והעיוותים בתמונת הכריכה.שיפור יעילות אלגוריתם LSB באמצעות חישוב של מודולו יותר גדול ממודולו 2.

# <u>הקוד המקורי לפני השיפורים</u>

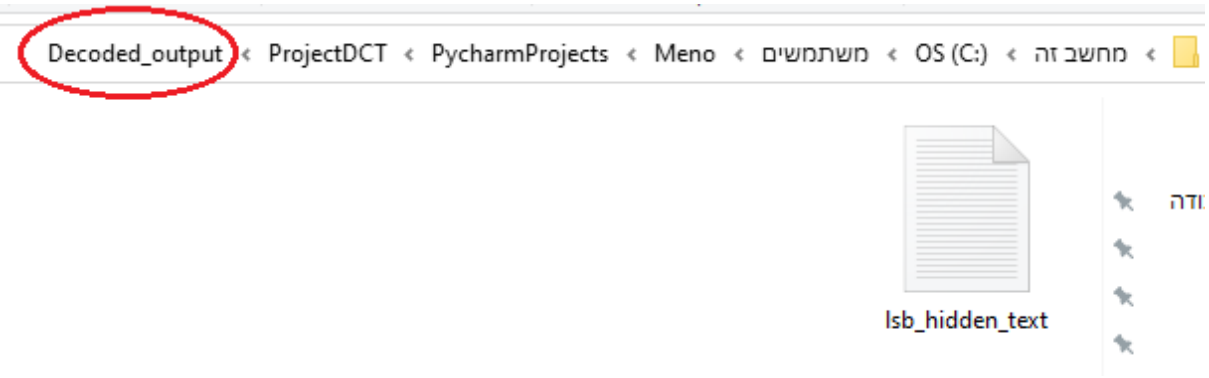
<u>שלבי ההרצה:</u>
- כדי לבצע הצפנה נזין 1
- ואז נכתוב את שם קובץ התמונה (כולל סיומת פורמט) שבה נשתמש
- לאחר מכן נזין את הטקסט אותו נרצה להצפין
- יוצג על המסך אורך ההודעה
- התמונה עם הטקסט המוצפן תישמר בתוך תיקיית "Encoded_image"

```
C:\Users\Meno\ProjectDCT\Scripts\python.exe C:/Users/Meno/PycharmProjects/ProjectDCT/original.py
To encode press '1', to decode press '2', to compare press '3', press any other button to close: 1
Enter the name of the file with extension : lenna.png
Enter the message you want to hide: hello
The message length is:  5
Encoded images were saved!
To encode press '1', to decode press '2', to compare press '3', press any other button to close:
```



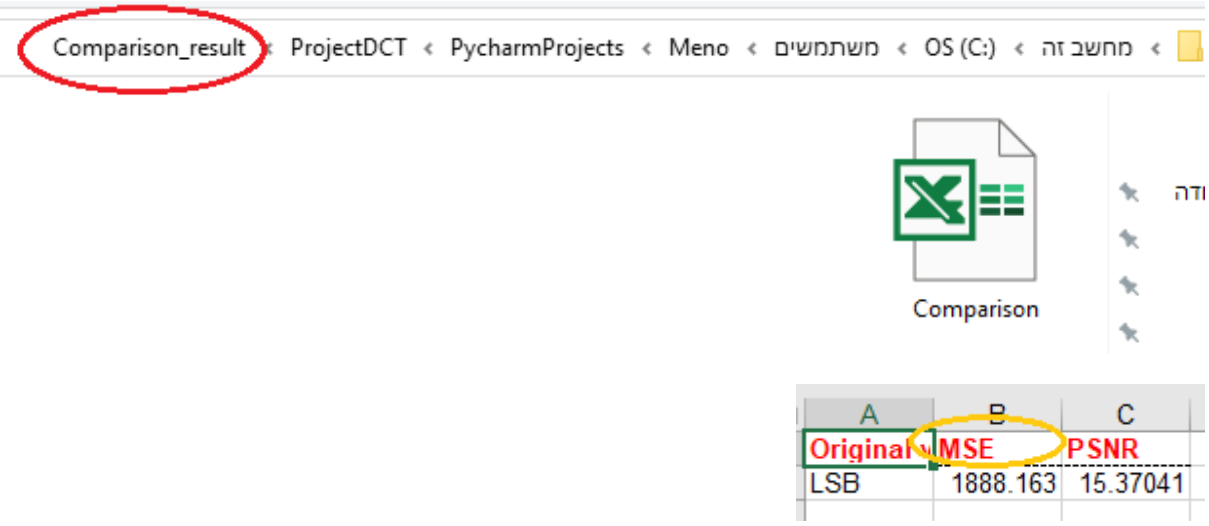

-כדי לבצע פענוח נזין 2
-התוכנית תיגש לתיקייה של התמונות המוצפנות ותפענח את התמונה ששמורה שם.
-הפענוח של הטקסט המוצפן בתמונה יופיע כקובץ טקסט בתוך התיקייה "Decoded_output"

```
To encode press '1', to decode press '2', to compare press '3', press any other button to close: 2
Hidden texts were saved as text file!
```




- כדי לבצע השוואה והצגה של מדדי ה- PSNR וה- MSE נזין 3
- יישמר קובץ EXCEL עם פירוט המדדים בתיקייה "Comparison_result"

```
To encode press '1', to decode press '2', to compare press '3', press any other button to close: 3
Comparison Results were saved as xls file!
To encode press '1', to decode press '2', to compare press '3', press any other button to close:
```




הערכים האופייניים של ה-PSNR בתמונה רועשת ובדחיסת וידאו הם בין 30dB ל-50dB, כאשר PSNR גבוה משקף מצב יותר טוב. ערכים סבירים של PSNR בשידור Wireless הם בין 20dB ל25dB. כאשר שתי תמונות זהות לחלוטין, אז ה-MSE הוא 0, ולכן ה-PSNR הוא אינסופי.

בס"ד

**כך ייראו התיקיות הנ"ל:**

| | | |
|---|---|---|
| Comparison_result | 22/07/2021 17:47 | תיקיית קבצים |
| Decoded_output | 22/07/2021 17:46 | תיקיית קבצים |
| Encoded_image | 22/07/2021 17:45 | תיקיית קבצים |
| Original_image | 22/07/2021 17:00 | תיקיית קבצים |

# הקוד לאחר הוספת השיפורים:

## 1. שימוש בRSA

הוספנו פונקציות הצפנה של טקסט לפי אלגוריתם RSA:

cipher,  decipher, gcd, phi.

```
from PIL import Image
from Stegan import Encode, Decode
import math
letter = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p",
          "q","r","s","t","u","v","w","x","y","z",",",".","!","?"," ","0","1"
           ,"2","3","4","5","6","7","8","9"]
number = ["01","02","03","04","05","06","07","08","09","10","11","12","13",
          "14","15","16","17","18","19","20","21","22","23","24","25","26"
    ,"27","28","29","30","31","32","33","34","35","36","37","38","39","40","41"]
def cipher(num,e):
    for i in range(len(num)):
        X.append((int(num[i])**e)%n)
def decipher(num,d):
    for i in range(len(num)):
        Y.append((int(num[i])**d)%n)
def gcd(a, b):
    while b != 0:
        (a, b)=(b, a % b)
    return a
def phi(n):
    amount = 0
    for k in range(1, n + 1):
        if math.gcd(n, k) == 1:
            amount += 1
    return amount
```

פונקציית Encrypt מקבלת את התמונה המקורית ואת הטקסט שנרצה להצפין בתמונה. הפונקצייה מבצעת הצפנת RSA על הטקסט עם ה-e (מפתח ציבורי) הנתון ומפעילה את פונקציית Encode (שנמצא בקובץ Stegan.py) שמבצעת את ההצפנה של הטקסט בתמונה. ולבסוף מחזירה את התמונה עם הטקסט המוצפן.

```
def Encrypt(original_image_file, plaintext):
# encrypts a plaintext message using the current key
    global numC, j, X
    X = []
    plaintext = (plaintext.lower())
    numC = []
    for i in range(len(plaintext)):
        for j in range(len(letter)):
            if(plaintext[i] == letter[j]):
                numC.append(number[j])
    cipher(numC, e)
    print("Ciphertext:", X)
    img_encoded = Encode(original_image_file, plaintext, X)
    return img_encoded
```

פונקציית Decrypt מקבלת תמונה עם טקסט מוצפן, מבצעת פענוח(Decode שנמצא בStegan.py) של הטקסט המוצפן בתמונה, ואז שולחת את הטקסט עם הערך d (מפתח פרטי) הנתון לפונקציית decipher שמפענחת את הטקסט באמצעות RSA, ולבסוף מחזירה את הטקסט שפוענח. (שמירת הטקסט בקובץ txt נעשית ב-driver).

```
def Decrypt(encoded_image_file):
    global i, j, Y
    Y = []
    hidden_text = Decode(encoded_image_file)
    print("the hidden text \n",hidden_text)
    decipher(hidden_text, d)
    numD = []
    for i in range(len(Y)):
        for j in range(len(number)):
            if(Y[i] == int(number[j])):
                numD.append(letter[j])
    return ''.join(numD)
```

***ערכים אלו של הRSA נקבעו באופן שרירותי על ידינו, וניתן לשנות אותם.**

```
n = 2537
e = 13
d = 937
```

בס"ד

## 2. שימוש במדד SSIM במקום MSE:

הוספנו את הפונקציה ssim למחלקה Compare. הפונקצייה מבצעת חישוב של הSSIM.

```
def ssim(self, img1, img2):
    C1 = (0.01 * 255) ** 2
    C2 = (0.03 * 255) ** 2

    img1 = img1.astype(np.float64)
    img2 = img2.astype(np.float64)
    kernel = cv2.getGaussianKernel(11, 1.5)
    window = np.outer(kernel, kernel.transpose())

    mu1 = cv2.filter2D(img1, -1, window)[5:-5, 5:-5]  # valid
    mu2 = cv2.filter2D(img2, -1, window)[5:-5, 5:-5]
    mu1_sq = mu1 ** 2
    mu2_sq = mu2 ** 2
    mu1_mu2 = mu1 * mu2
    sigma1_sq = cv2.filter2D(img1 ** 2, -1, window)[5:-5, 5:-5] - mu1_sq
    sigma2_sq = cv2.filter2D(img2 ** 2, -1, window)[5:-5, 5:-5] - mu2_sq
    sigma12 = cv2.filter2D(img1 * img2, -1, window)[5:-5, 5:-5] - mu1_mu2

    ssim_map = ((2 * mu1_mu2 + C1) * (2 * sigma12 + C2)) / ((mu1_sq + mu2_sq + C1) *
                                                            (sigma1_sq + sigma2_sq + C2))
    return ssim_map.mean()
```

כעת בטבלת ה excel יופיע מדד SSIM ולא MSE.

| A | B | C |
|---|---|---|
| Original v | SSIM | PSNR |
| Encoded | 0.771185 | 15.36798 |

### 3. שימוש במודולו גדול מ-2 באלגוריתם ה-LSB

השתמשנו במודולו 3 במקום ב2, להלן הפונקציות של הLSB החדש:

```
class LSB():
    def encode(image_name, secret_data):
        # read the image
        image = cv2.imread (image_name)
        # maximum bytes to encode
        n_bytes = image.shape[0] * image.shape[1] * 3 // 8
        print ("[*] Maximum bytes to encode:", n_bytes)
        if len (secret_data) > n_bytes:
            raise ValueError ("[!] Insufficient bytes, need bigger image or less data.")
        print ("[*] Encoding data...")
        # add stopping criteria
        secret_data += "======"
        data_index = 0
        # convert data to binary
        binary_secret_data = ternary(binaryToDecimal(to_bin (secret_data)))
        # size of data to hide
        print (binary_secret_data)
        data_len = len (binary_secret_data)
        for row in image:
            for pixel in row:
                # convert RGB values to binary format
                r, g, b = to_bin (pixel)
                # modify the least significant bit only if there is still data to store
                if data_index < data_len:
                    # least significant red pixel bit
                    pixel[0] = int (r[:-1] + binary_secret_data[data_index], 3)
                    data_index += 1
                if data_index < data_len:
                    # least significant green pixel bit
                    pixel[1] = int (g[:-1] + binary_secret_data[data_index], 3)
                    data_index += 1
                if data_index < data_len:
                    # least significant blue pixel bit
                    pixel[2] = int (b[:-1] + binary_secret_data[data_index], 3)
                    data_index += 1
                # if data is encoded, just break out of the loop
                if data_index >= data_len:
                    break
        return image
```

*על מנת לממש את הפונקצייה עם מודולו 3 עשינו מספר שינויים והתאמות:
<u>בהצפנה</u>: תחילה המרנו את הטקסט לבינארי, ואז המרנו לבסיס דצימלי ולבסוף המרנו לבסיס טרנרי (מודולו 3)

בס"ד

בפענוח: נדרשנו לבצע את תהליך ההמרה ההפוך, תחילה המרנו מטרנרי לדצימלי ולאחר מכן המרנו מדצימי לבינארי.

```python
def decode(image_name):
    print ("[+] Decoding...")
    # read the image

    print ("*")
    print (type (image_name))
    print ("*")

    print ("___")
    binary_data = ""
    for row in image_name:
        for pixel in row:
            r, g, b = dec_to_bin (convertToDecimal (pixel))
            binary_data += r[-1]
            binary_data += g[-1]
            binary_data += b[-1]
    # split by 8-bits
    all_bytes = [binary_data[i: i + 8] for i in range (0, len (binary_data), 8)]
    # convert from bits to characters

    decoded_data = ""
    for byte in all_bytes:
        # decoded_data += byte
        decoded_data += chr (int (byte, 2))
        if decoded_data[-5:] == "=====":
            break
    return decoded_data[:-5]
```

**ההרצה של הקוד עם השיפורים:**

```
C:\Users\Meno\ProjectDCT\Scripts\python.exe "C:/Users/Meno/PycharmProjects/ProjectDCT/new 15.7.py"
To encode press '1', to decode press '2', to compare press '3', press any other button to close: 1
Enter the name of the file with extension : lenna.png
Enter the message you want to hide: hello
The message length is:  5
Ciphertext: [156, 205, 2497, 2497, 2116]
Encoded images were saved!
```

**ניתן לראות שהטקסט שהוכנס (hello) הומר לצופן cipher תחילה ורק אז הוצפן בתמונה.**
**את שאר השינויים שהגיעו בעקבות שינוי הקוד והוספת השיפורים, ניתן לראות במדדים בקובץ ה excel,**
**לאחר הזנת 3 בתפריט, לביצוע השוואה.**

# מסקנות

- הוספת אלגוריתם RSA להצפנת הטקסט  תרמה לאבטחת ההודעה המועברת בערוץ תקשורת.
- שימוש במדד SSIM במקום MSE, משפר את התוצאות הנראות לעין האנושית.
- שימוש במודולו שגדול ממודולו 2 (מודולו 3 במקרה שלנו) תורם לצמצום השינויים והעיוותים בתמונה.

## הוראות הרצת הקוד:

- הספריות הנדרשות הן:

```
import numpy as np
import os
import xlwt
import shutil
import cv2
import sys
import math
import itertools
from PIL import Image
from pathlib import Path
from scipy import signal
from PIL import Image
from Stegan import Encode, Decode
import math
```

בס"ד

- ישנם 2 קבצי פייתון:

DataSecurityProject.py
Stegan.py

- צריך לשמור את תיקיית התמונות "Original_image" בתיקייה של הפרוייקט של פייתון.
- לבסוף לבצע הרצה:
  - **להזין 1 להצפנה**
  - ואז יש לרשום את שם הקובץ של התמונה כולל סיומת
  - לאחר מכן להזין את הטקסט אותו נרצה להצפין בתמונה
  - התמונה המוצפנת תישמר בתיקייה "Encoded_image"
  - **כעת נזין 2 לפענוח**
  - התוכנית תבצע פענוח של הטקסט המוצפן בתמונה שנמצאת בתיקייה "Encoded_image"
  - קובץ TXT שבו הטקסט שהוצפן בתמונה, יישמר בתיקייה "Decoded_output"
  - **כעת נזין 3 להשוואת מדדים**
  - התוכנית תבצע השוואה במדדים
  - יישמר קובץ excel עם נתוני המדדים

```
C:\Users\Meno\ProjectDCT\Scripts\python.exe C:/Users/Meno/PycharmProjects/ProjectDCT/DataSecurityProject.py
To encode press '1', to decode press '2', to compare press '3', press any other button to close: 1
Enter the name of the file with extension : pepper.png
Enter the message you want to hide: hello
The message length is:  5
Ciphertext: [156, 205, 2497, 2497, 2116]
Encoded images were saved!
To encode press '1', to decode press '2', to compare press '3', press any other button to close: 2
the hidden text
 [156, 205, 2497, 2497, 2116]
Hidden texts were saved as text file!
To encode press '1', to decode press '2', to compare press '3', press any other button to close: 3
Comparison Results were saved as xls file!
To encode press '1', to decode press '2', to compare press '3', press any other button to close:
```