אבטחת מחשבים ורשתות תקשורת
עבודה 1

גיא זמוסטיאנו – [redacted], ירדן קוריאל - [redacted]

שאלה 1:
ציור כל הקומבינציות בהם תוקף יכול ליישם את ההתקפה.
הערה: בשאלה הזאת אנו מניחים:

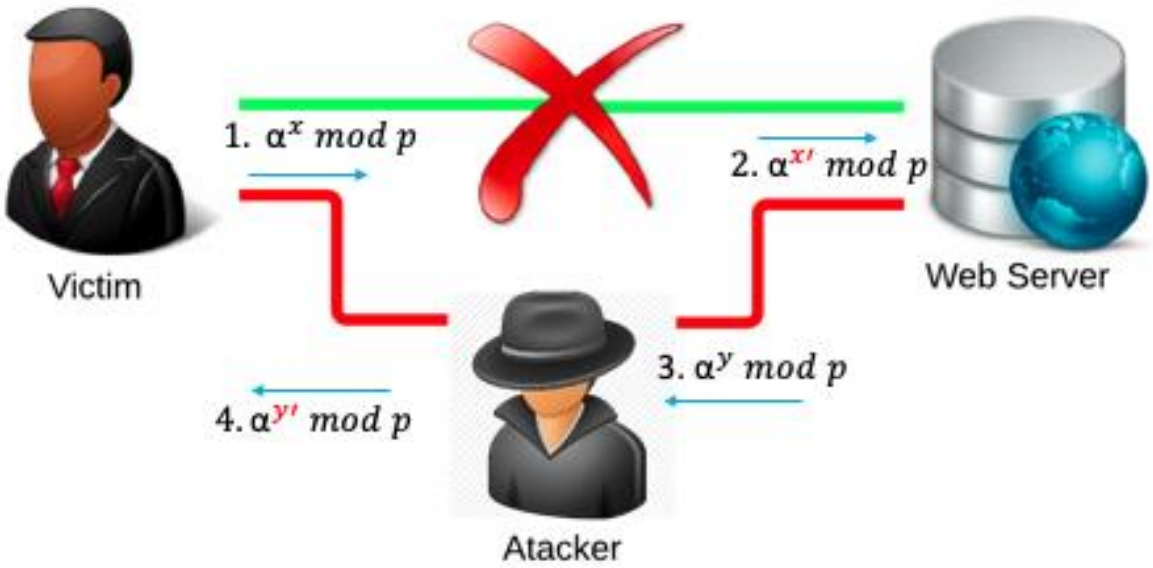
- הלקוח הוא זה ששולח את ההודעה ראשון, הוא היוזם.
- השרת לא יוזם שיחה ובכדי לשלוח הודעה הוא מחויב לקבל פנייה קודם לכן.

במידה והשרת היה מסוגל ליזום, אז היה יכול להיות !3 אפשרויות, יתווספו שלוש האפשרויות הבאות:

- פנייה של הלקוח(הקורבן) ניתור של ההודעה ע"י התוקף, שליחה של הודעה מהשרת ניתור ההודעה ע"י התוקף ושליחה של המפתח החלופי ע"י התוקף ללקוח(קורבן) ולבסוף שליחה של המפתח החלופי ע"י התוקף לשרת.
- פנייה של הלקוח(הקורבן) ניתור של ההודעה ע"י התוקף, שליחה של הודעה מהשרת ניתור ההודעה ע"י התוקף ושליחה של המפתח החלופי ע"י התוקף לשרת ולבסוף שליחה של המפתח החלופי ע"י התוקף ללקוח(קורבן).
- פנייה של הלקוח(הקורבן) ניתור של ההודעה ע"י התוקף, שליחה של המפתח החלופי ע"י התוקף ללקוח(קורבן), שליחה של הודעה מהשרת ניתור ההודעה ע"י התוקף ולבסוף שליחה של המפתח החלופי ע"י התוקף לשרת.

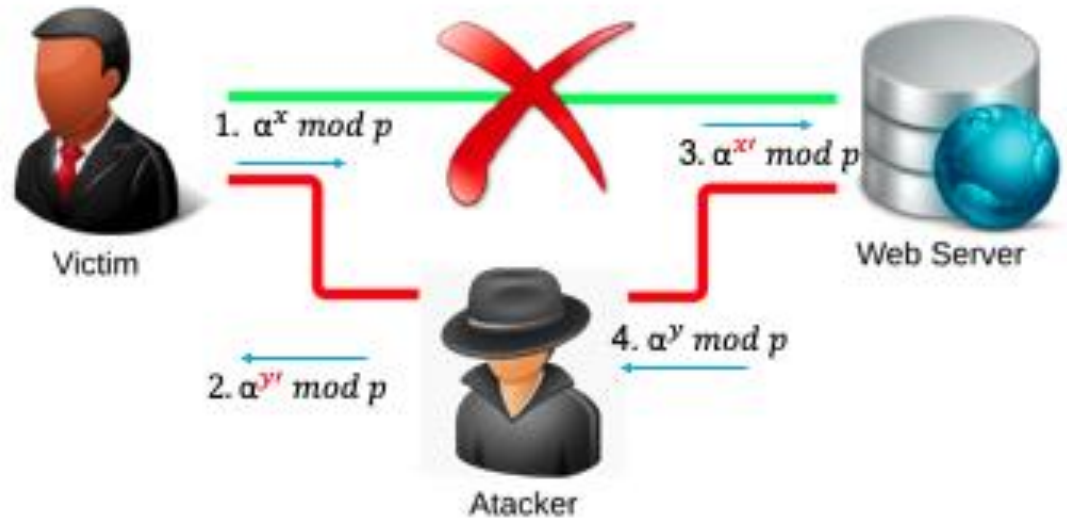
אפשרות 1:

1. לקוח(קורבן) שולח לשרת מפתח A.
2. תוקף מנתר את ההודעה של הלקוח (קורבן).
3. תוקף שולח לשרת הודעה עם מפתח A'.
4. שרת שולח הודעה לתוקף הודעה עם מפתח B.
5. תוקף מקבל את מפתח B מהשרת.
6. תוקף שולח הודעה ללקוח (קורבן) עם מפתח B'.



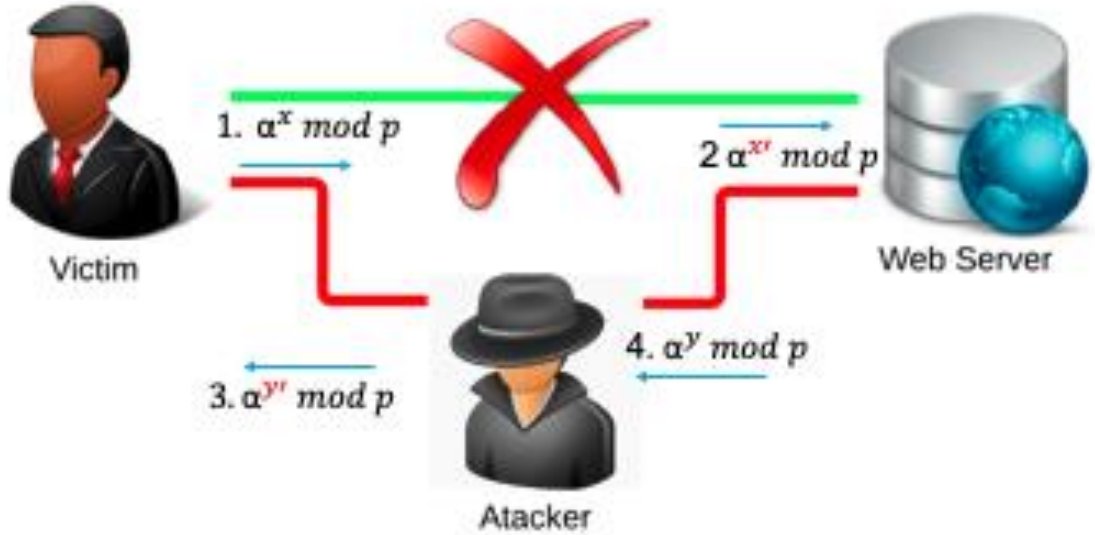
אפשרות 2:

1. לקוח(קורבן) שולח לשרת מפתח A.
2. תוקף מנתר את ההודעה של הלקוח (קורבן).
3. תוקף שולח הודעה ללקוח (קורבן) עם מפתח B’.
4. תוקף שולח לשרת הודעה עם מפתח A’.
5. שרת שולח הודעה לתוקף הודעה עם מפתח B.
6. תוקף מקבל את מפתח B מהשרת.



אפשרות 3:

1. לקוח(קורבן) שולח לשרת מפתח A.
2. תוקף מנתר את ההודעה של הלקוח (קורבן).
3. תוקף שולח לשרת הודעה עם מפתח A’.
4. תוקף שולח הודעה ללקוח (קורבן) עם מפתח B’.
5. שרת שולח הודעה לתוקף הודעה עם מפתח B.
6. תוקף מקבל את מפתח B מהשרת.



שאלה 2:

פסאדו קוד לתקיפה ושבירת המערכת 5-DES :

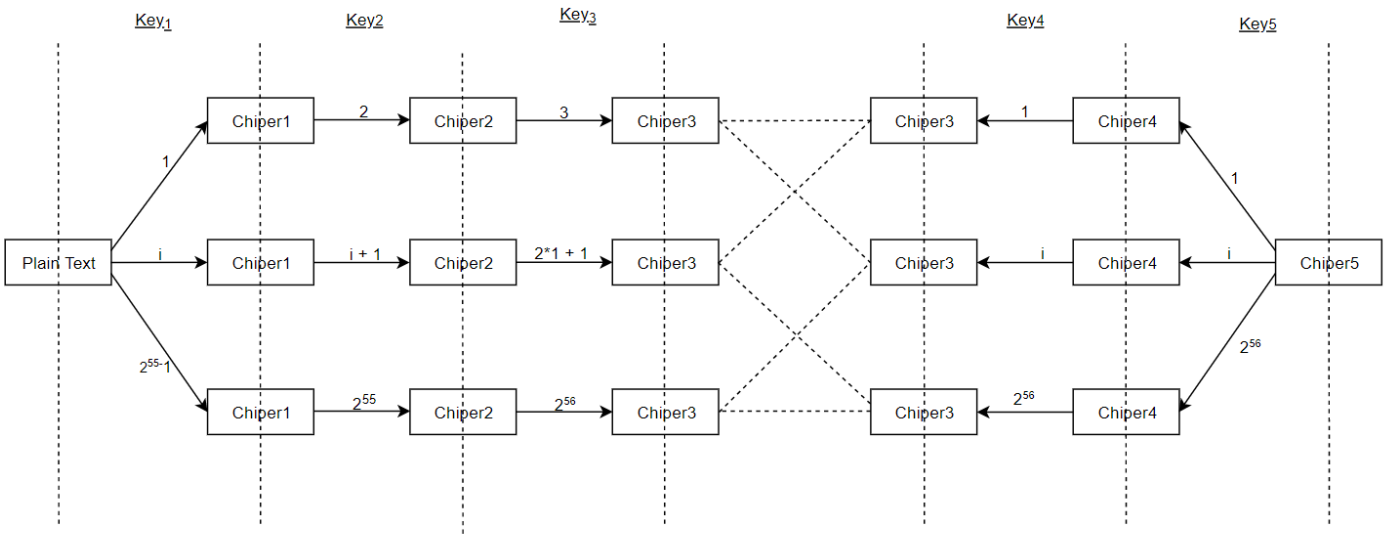
הנחות:

- Known PlainText Attack.
- Given a two pairs of Plain-text and Cypher-text <P1,A1>,<P2,A2>.
- We assume 5-DEC means: Encryption- Encryption- Encryption- Encryption- Encryption.
- Range of the first key is between 0- 2^{55} because the equation of the third key (2X+1). Means we don't ignore overflow bit.

1. Given plaintext P1. For each **key** in range 0 to 2^{55}-1 do the following:
   a. Encrypt P1 with DES using the K_1 (=**key**) and insert to C_1.
   b. Encrypt C_1 with DES using the K_2 (=K_1+1) and insert to C_2.
   c. Encrypt C_2 with DES using the K_3 (=2*K_1+1) and insert to C_3.
   d. Store < C_3, **key** >

2. Given ChiperText A1. For each **key** in range 0 to 2^{56} do the following:
   a. Decrypt A1 with DES twice with **key** as key and insert the result to $C_{3'}$.
   b. Store <$C_{3'}$, **key**>.

3. Compare the encryption outputs with the decryption outputs
   a. Every matching C_3, $C_{3'}$ will reveal a possible key_1, key_5 (=key_4) pair.
   By knowing key_1, key_2 and key_3 will be known as well because key_2 = key_1+1 and key_3 = 2*key_1+, and key_5 =key_4.
   b. Test all possible key-pairs on <P2,A2> to reveal the correct key-pair.



Memory complexity: O(2^{55}+2^{56}) =O(3*2^{55})

Time complexity: O(3*2^{55}+2*2^{56}) =O(7*2^{55})

הערה:

* על פי ההנחה הרביעית כי טווח המפתחות בין אפס ל-2^{55} נבחר חסם הדוק ככל האפשר לפי טווח זה ולכן:
- סיבוכיות הזיכרון מהצפנת שלושת המפתחות הראשונים תדרוש 2^{55} וסיבוכיות הזיכון לפענוח הצופן תהיה 2^{56} (עבור המפתחות הרביעי והחמישי), שעליהן אין הגבלה.
- סיבוכיות זמן הריצה עבור הצפנת ההודעה לטווחים הנ"ל לשלושת המפתחות הראשונים תדרוש $3 * 2^{55}$ וסיבוכיות זמן הריצה עבור הפענוח של הצופן בעזרת המפתחות החמישי והרביעי תדרוש $2 * 2^{56}$
* נשים לב שבכדי להגיע להצפנת ההודעה, טווח המפתח נוכל להשתמש לכל היותר במפתח ראשון בטווח $2^{55} - 1$ (מהנוסחה הנתונה בשאלה), ולכן ההתייחסות לסיבוכיות הזכרון וזמן הריצה היא לפי דרך זאת.

שאלה 3:
סעיף א:

התוקף מנסה להשיג את המפתח למערכת ההצפנה של בוב ואליס.
לפי מה שהבנו, התוקף מנסה להשיג את המפתח החל מהיום השלישי. ההודעות והצופן עבורם הוא מחפש את המפתח הם מהיום השלישי.
כמובן שבהינתן מפתח של יום כלשהו, אז בעזרת הנוסחה בוס ואליס, גם המפתח של כל שאר הימים יהיה ידוע וכך נוכל לפענח הודעות מימים שונים.

פסאדו קוד לתקיפה ושבירת מערכת הצפנה סימטרית עם מפתח K משותף בעל N ביטים.

המתקפה שבחרנו היא Bruth Force Attack כאשר ידוע לנו Plaintext -וCipherText:
Known Plaintext Attack – a set of pairs of messages Pi and their cryptograms Ci from **day three** are known to the attacker: {(P0, C0), (P1, C1),…….}

```
1. Given plaintext P1.
2. for key in range (0…2^N ) that is also have 2 LSB(least significant bit) are '1':
      a. Encrypt P1 using the key
      b.  if (ENC(P1, key) == C1)
                return key
```

הסבר לנכונות האלגוריתם:
על פי הנוסחה של יצירת המפתחות ניתן לדעת שביום השלישי שני הביטים הימניים ביותר (LSB) יהיו אחדות.
הסיבה לכך הינה שבייצוג בינארי המשמעות של הכפלה פי 2 הינה הזחת המספר שמאלה ולפיכך יתווסף ביט אפס.
ובנוסחה גם נתון להוסיף 1 לאחר ההכפלה ולכן בהכרח שני הביטים הראשונים יהיו אחדות. רצף הפעולות הינו הזחה – הוספת אחד – הזחה – הוספת אחד.
בפסדו קוד שרשמנו חיפשנו לכל מפתח קיים אשר 2 הביטים הראשונים שלו הם 1 ובדקנו האם הוא מתאים להצפנה של זוג plaintext ו cipherText , במידה והמפתח מתאים נחזיר אותו.
נוכל למצוא בעזרת זוג הודעות מיום כלשהו את המפתח והדבר היחידי שנצטרך לשנות הוא מספר ביטי היחידות בLSB במספר הימים שעברו.

Memory: O(1).
Time: O(2^{N-2}).

- **Notes**:
  we assume it take no more than one day to break the code.

סעיף ב:

במקרה והתוקף ייעזר בסבלנות הוא יוכל לשפר את זמני הריצה של מציאת המפתח משמעותית.
הסיבה לכך שיוכל לשפר הינה שהמפתח בסופו של דבר יתכנס לצורה שכולו אחדות (בכל יום שיעבור יתווסף למפתח ספרת אחדות נוספת מימין LSB) ואז בהינתן העובדה שהתוקף יודע מהו גודל המפתח אז יוכל תוך ניסיון יחיד למצוא אותו. ידיעת גודל המפתח גם חוסמת את הזמן אותו יצטרך לחכות עד לניסיון הפריצה כי בהכרח ידע שלאחר N ימים מיצירת המפתח, בוודאות כל המפתח יהיה אחדות (הגיוני שההתכנסות תתרחש לפני אך זהו החסם העליון).
ולפיכך סיבוכיות זמן ריצה וזיכרון הינם O(1).

שאלה 4:

פסאדו קוד עבור 2-DEC, לאלגוריתם Meet in the middle:

1. ראשית נבצע לכל **key1** אפשרי בתחום בין 0 ל $2^{56} - 1$:
   a. נחשב את הערך: ENC(key1,plaintext) ונבצע השמה למשתנה C1.
   b. נשמור את הזוג הסדור <C1,key1>

2. שנית, נבצע לכל **key2** אפשרי בתחום בין 0 ל $2^{56} - 1$:
   a. נחשב את הערך: DEC(key2,cipherText) ונבצע השמה למשתנה C2.
   b. נשמור את הזוג הסדור <C2,key2>

3. נבצע השוואה בין כל זוג C2' ל C1 אפשריים, וכאשר נמצא התאמה נבדוק התאמה זאת בעזרת 'plaintext ו- 'cipherText נוספים כדי לוודא את המצאות המפתחות הנכונים.
   a. נחזיר את key1,key2

- Time complexity:
  - Encrypt with all possible keys (in DES = O(2^{56}))
  - Decrypt with all possible keys (in DES = O(2^{56}))
  - Compare results – O(1) using look up table
- Memory: look up table requires O($2^{56} + 2^{56}$)=O(2^{57}) memory
- Note: the complexity of time and memory will be 2^{56} and not 2^{64}.

שאלה 5:

- <u>סעיף א'</u> – Known ChiperText:

נשתמש בהתקפת brute force attack:

על מנת למצוא את המפתחות כאשר רק הצופן ידוע נצטרך לבדוק לכל מפתח k1 אפשרי את כל הקומבינציות עם מפתח k2 אפשרי. עבור כל פענוח, נצטרך לבדוק האם הPLAIN TEXT הינו הגיוני וכך להחליט שהמפתח פוטנציאלי.
לכל אחד מהמפתחות יש 2^{56} אפשרויות ולכן כל הקומבינציות תיצור סיבוכיות זמן ריצה של $2^{56} * 2^{56} = 2^{112}$ ולכן:
<u>Time</u> <u>complexity:</u> $O(2^{112})$
<u>Memory</u> <u>complexity:</u> $O(1)$

<u>Note:</u> we assume from the question that key is really 56bit as we have learned, if the key is really in size 64 the Time will be $O(2^{128})$ and Memory Complexity will be $O(1)$.

- <u>סעיף ב'</u> – Known PlainText:
  - Known Plaintext Attack – a set of pairs of messages Pi and their cryptograms Ci are known to the attacker: {(P0, $C0$), (P1, $C1$),…….}

נשתמש בהתקפת meet in the middle.

1. עבור P1 (PlainText)
   a. עבור על כל מפתח K1 אפשרי:
      i. נחשב P1⊕K1.
      ii. נעביר את התוצר בפונקציה הידועה F.
      iii. את התוצר נשמור תחת זוג סדור <F(P1⊕K1), K1>.

2. עבור C1 (ChiperText)
   a. עבור על כל מפתח K2 אפשרי:
      i. נחשב C1⊕K2.
      ii. את התוצר נשמור תחת זוג סדור <C1⊕K2, K2>.

3. נשווה את כל הערכים F(P1⊕K1) יחד עם הערכים של C1⊕K2 ששמרנו:
   בכל התאמה (שיוויון) נשמור את המפתח הפוטנציאלי ונבדוק האם מתאים לזוג סדור נוסף <P2,C2>, במידה וכן, נחזיר אותו ומצאנו את K1,K2.

לP1 ישנם 2^{56} אפשרויות נבדוק את כולם.
לC1 ישנם 2^{56} אפשרויות נבדוק את כולם.

<u>Time</u> <u>complexity:</u> $O(2^{56})$
<u>Memory</u> <u>complexity:</u> $O(2 * 2^{56}) = O(2^{57})$

- **<u>Note:</u>** we assume from the question that key is really 56bit as we have learned, if the key is really in size 64 the Time will be $O(2^{64})$ and Memory Complexity will be $O(2^{65})$.

- <u>סעיף ג'</u> – Known PlainText and K1:

נוכל למצוא את המפתח K2 באופן הבא:
C1 ידוע ושווה ל:

$C_1 = F(P_1 \oplus K_1) \oplus K_2$

נתונים: P_1 , K_1, F
ולפיכך נוכל לחשב: $F(P_1 \oplus K_1)$

$C_1 \oplus F(P_1 \oplus K_1) = K_2$

ומכך נוכל לבודד את K_2 ולחשבו.

<u>Time</u> <u>complexity:</u> O(1)
<u>Memory</u> <u>complexity:</u> O(1).

חלק ב׳ – חלק מעשי

שאלה 6:

- סעיף א׳:

פקודה שבה השתמשנו:

ip.src == 132.72.81.121 and not icmp

את הקובץ ייצאנו ל-csv וסיננו IP-ים ייחודיים לפי הקוד הבא:

```
path = "q1.csv"
csv_file = pd.read_csv(path)
unique = csv_file.Destination.unique()
size = len(unique)
```

תשובה: 339.

- הערה: כמו שאפשר לראות בפקודה שביצענו, התעלמנו מהודעות ICMP שהיו לא מהIP הרלוונטי בשאלה.

- סעיף ב׳:

את כלל הרשומות בקובץ pcapng. ייצאנו לcsv ובדקנו IP-ים ייחודיים של יעדים:
ניתן לראות בצילום המסך כי ברשימה נכללו גם שורות לא רלוונטיות ולכן השמטנו אותן מהספירה.
לאחר הבדיקה עבור הערכים הרלוונטיים מצאנו כי מספר יעדי הIP השונים הוא: **353**

	87.233.83.59
	89.46.106.57
	95.100.134.34
	95.101.192.162
	95.168.178.100
	99.86.116.101
	99.86.116.65
	99.86.116.75
	99.86.116.99
	99.86.119.218
	Broadcast
	CDP/VTP/DTP/PAgP/UDLD
	Cisco_51:b4:07
	ff02::1
	ff02::1:2
	ff02::1:3
	ff02::16
	ff02::2
	ff02::c
	ff02::fb
	LLDP_Multicast
	Spanning-tree-(for-bridges)_00
	Grand Total
IP ONLY	353
TOTAL	365

תשובה: 353.

- סעיף ג':

פקודה שבה השתמשנו:

ip.dst == 132.72.81.121 and udp

<u>תשובה</u>: 357,615.

- הערה: כמו שאפשר לראות בפקודה שביצענו, סיננו על-פי הודעות UDP, אך קיבלנו גם הודעות DCP-AF,DNS ו-AR-DRONE שהם כולם פרוטוקולים שמבוססים על UDP.

- סעיף ד':

הפקודה בה השתמשנו:

ip.src == 132.72.81.121 and not tcp

את הקובץ ייצאנו ל-csv וסיננו IP-ים ייחודיים לפי הקוד הבא:

```
path = "../q4.csv"
csv_file = pd.read_csv(path)
# source = csv_file["Source"] == "132.72.81.121"
# without_tcp = csv_file[csv_file["Protocol"] != "TCP"]

count_dict = {}
for size in csv_file.Length.unique():
    count_dict[f"{size}"] = len(csv_file[csv_file["Length"] == size])

sort_dict = {k: v for k, v in sorted(count_dict.items(), key=lambda item: item[1], reverse=True)}

for key, value in sort_dict.items():
print(f"Massage Length:{key} appear {value} times in different records at the csv")
```

פלט התוכנית:

```
Massage Length:75 appear 24134 times in different records at the csv
Massage Length:73 appear 5901 times in different records at the csv
Massage Length:88 appear 1391 times in different records at the csv
Massage Length:71 appear 1214 times in different records at the csv
Massage Length:72 appear 805 times in different records at the csv
Massage Length:70 appear 758 times in different records at the csv
Massage Length:79 appear 668 times in different records at the csv
```

<u>תשובה</u>: רוב הפקטות שנשלחו ממחשב 132.72.81.121 הן בגודל 75 בייט.

- <u>סעיף ה':</u>

הפקודה בה השתמשנו :

ip.src == 132.72.81.121 and dns

את הקובץ ייצאנו ל-csv וסיננו IP-ים ייחודיים לפי הקוד הבא:

```
path = "../q5.csv"
csv_file = pd.read_csv(path)
result = csv_file.Destination.unique()
print(result)
```

פלט התוכנית:

```
['132.72.140.46' '132.72.140.45']
```

תשובה: שרתי ה-DNS הם:

- 132.72.140.45.
- 132.72.140.46.

<u>סעיף ו':</u>

הפקודה בה השתמשנו :

"ip.src == 35.172.73.102 and frame contains "video

<u>תשובה:</u> מצאנו כי ישנם 2 פקטות שונות שמכילות "video" (קשורות לסרטונים) לאחר סינון לפי ipה המבוקש.

<u>סעיף ז':</u>

הפקודה בה השתמשנו:

ip.src == 35.172.73.102 and frame.len < 100

תשובה: קיבלנו כי קיימות **14** פקטות.

<u>סעיף ח':</u>

הפקודה בה השתמשנו:

ip.src == 35.172.73.102 and ip.dst == 132.72.81.121 and http

תשובה: לא מצאנו אף תוצאות, כלומר לא הייתה תקשורת בין המחשבים הנ"ל בפרוטוקול http.

<u>סעיף ט':</u>

הפקודה בה השתמשנו:

ip.src == 132.72.81.121

את הקובץ ייצאנו ל-csv וסיננו IP-ים ייחודיים לפי הקוד הבא:

```
path = "../q9.csv"
csv_file = pd.read_csv(path)
result = csv_file.Protocol.unique()
print(result)
```

פלט התוכנית:

```
['TCP' 'TLSv1.2' 'ICMP' 'DNS' 'UDP' 'TLSv1.3' 'SSDP' 'TLSv1']
```

תשובה: קיבלנו כי קיימים 8 פרוטוקולים שונים באותם יזם המחשב 132.72.81.121.

סעיף י':

הפקודה בה השתמשנו:

ip.dst == 132.72.81.121 and udp

את הקובץ ייצאנו ל-csv וסיננו IP-ים ייחודיים לפי הקוד הבא:

```
path = "../q10.csv"

data = pd.read_csv(path)
time_feature = data.Time
indexs_feature = data["No."]

min = 1000000
indexMin1 = -1
indexMin2 = -1
max = 0
indexMax1 = -1
indexMax2 = -1

for index in range(10, len(time_feature)-1):

    diff = time_feature[index + 1] - time_feature[index]

    if diff > max:
        max = diff
        indexMax1 = indexs_feature[index]
        indexMax2 = indexs_feature[index + 1]

    if min > diff:
        min = diff
        indexMin1 = indexs_feature[index]
        indexMin2 = indexs_feature[index + 1]

print(f"min difference value: {min}")
print(f"index of Min value is between {indexMin1}-{indexMin2}")
print()
print(f"max difference value: {max}")
print(f"index of Max value is between {indexMax1}-{indexMax2}")
```

הפלט שהתקבל:

```
min difference value: 0.0
index of Min value is between 1119-1120

max difference value: 4.003409999999974
index of Max value is between 424848-425685
```

תשובה: מצאנו כי ההפרש המינימאלי הוא 0.0 (בין האינדקסים 1119-1120) וההפרש המקסימאלי הוא 4.00340999999974 (בין האינדקסים 424,848-424,685).

- הערה: התייחסנו להודעות שונות כהודעות עם No אחר.

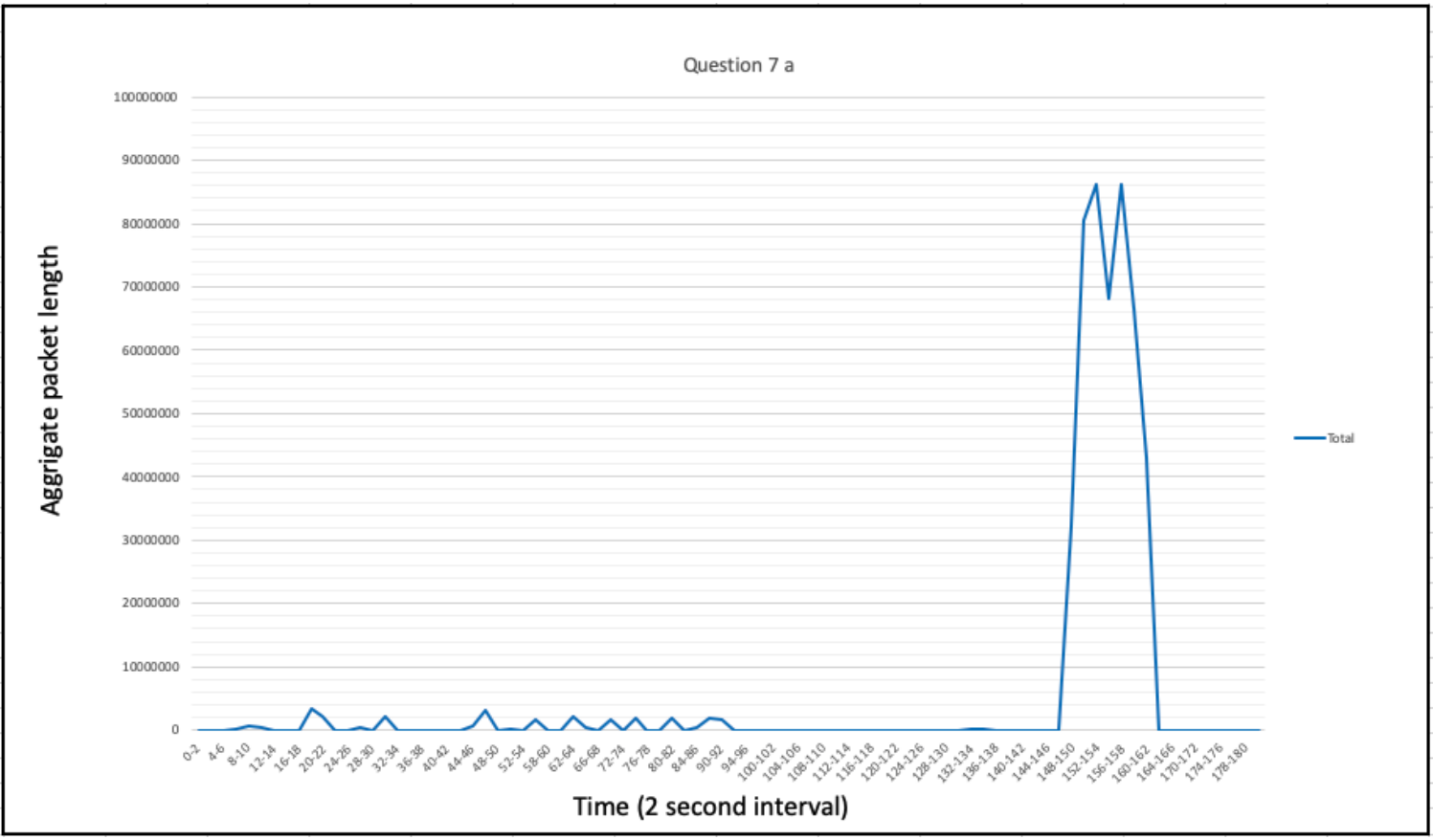
שאלה 7

סעיף א':

הפקודה בה השתמשנו:

ip.dst == 132.72.81.121 and udp

את הקובץ ייצאנו ל-csv ובעזרת pvot table ייצרנו ב-excel את הגרף הבא:



סעיף ב':

כדי לחשב את האינטרוול באורך 12 שבו נשלחת הכי הרבה תעבורה, יצרנו ב-excel עמודה חדשה אשר עושה פונקציית סכימה של כל אינטרוול בגודל 12 שניות ובחרנו את המקסימאלי, ניתן לראות כי האינטרוואל המקסימאלי הוא מ150-162 והוא בעל גודל של **430940112**.

146	144-146	51124	155520
148	146-148	4140	88815
150	148-150	32006596	32094084
152	150-152	80498278	112583850
154	152-154	86261231	198835274
156	154-156	68091233	266912602
158	156-158	86244512	353105990
160	158-160	66618316	419720166
162	160-162	43226542	430940112
164	162-164	644	350442478
166	164-166	717	264181964
	166-168	529	196091260
	170-172	899	109847647
	172-174	1089	43230420
	174-176	1648	5526
	176-178	874	5756
	178-180	706	5745
	180-182	394	=SUM(R93:R98
	Grand Total	**490844213**	430940112

סעיף ג':

ראשית מצאתנו את הזמנים היחסיים שבהם האינטרוול 150-162 מתרחש.

הפקודה בה השתמשנו:

```
frame.time >= "Feb  9, 2020 14:58:14.266603000" and frame.time <= "Feb  9, 2020 14:58:26.266603000"
and ip.dst == 132.72.81.121 and udp
```

את הקובץ ייצאנו ל-csv וסיננו IP-ים ייחודיים, לכל אחד מהם סכמנו את נפח הפקטות לכל אחד מהIP-ים, על-לפי הטבלה הבאה:

Row Labels	Sum of Length
132.72.140.45	1847
132.72.140.46	3136
172.217.168.206	1116
172.217.168.238	1201
172.217.17.106	2301
172.217.17.129	430928680
172.217.17.46	1554
173.194.69.189	215
216.58.208.99	62
Grand Total	**430940112**

תשובה: נשים לב שנפח הפקטות הגדול ביותר הגיע ממחשב שה-IP שלו הוא**172.217.17.129**.

סעיף ד':

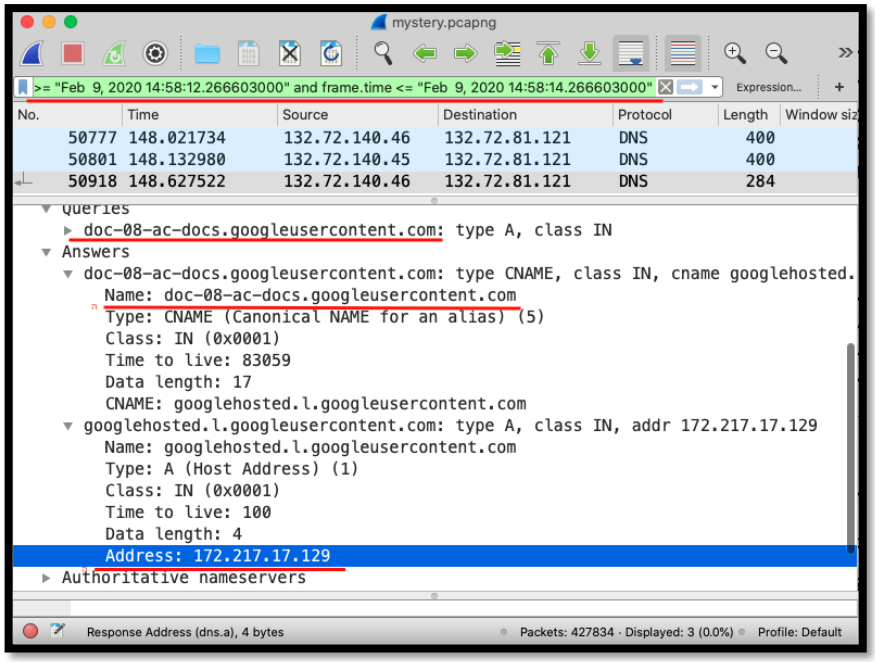
בכדי למצוא מאיזה שירות אחסון קבצים הורד הקובץ אשר מצאנו בסעיף הקודם (ip – 172.217.17.129) חיפשנו הודעת DNS אשר עונה על בקשת DNS לפי כתובת של מסמך, שהתשובה עליה תהיה הכתובת IP אשר אותה מצאנו.

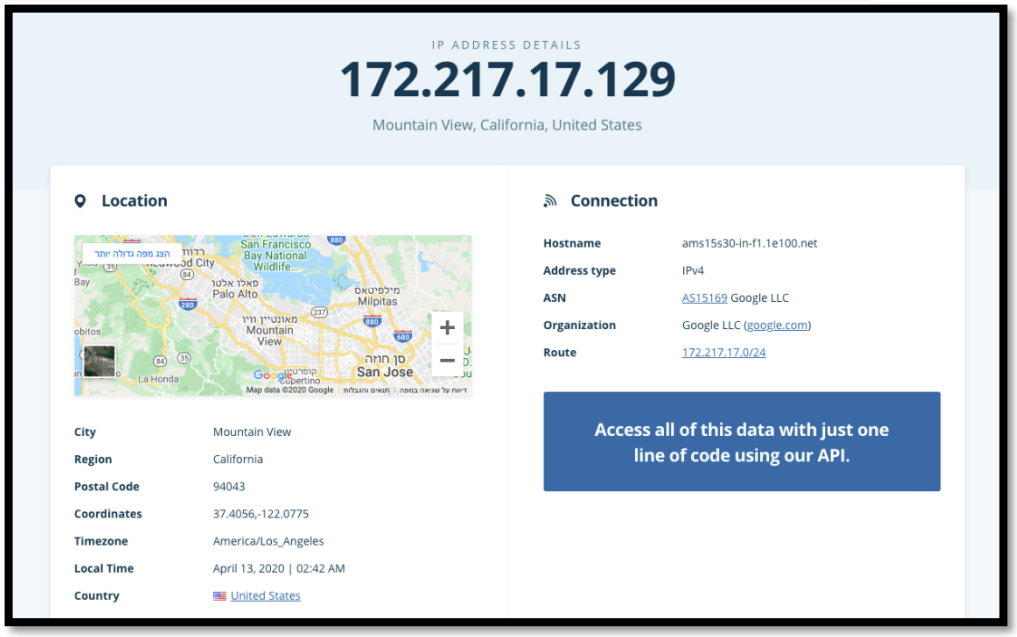
הפקודה בה השתמשנו:

ip.dst == 132.72.81.121 and dns and frame.time >= "Feb 9, 2020 14:58:12.266603000" and frame.time <= "Feb 9, 2020 14:58:14.266603000"

לאחר מכן קיבלנו 3 תוצאות של תשובות שהגיעו ל-IP 132.72.81.121 ושמנו לב שבפקטה האחרונה הגיעה תשובה לגבי בקשה לכתובת ip של "doc-08-ac-docs.**google**usercontent.com" והתשובה שהגיע משרת הdns היא 172.217.17.129, הכתובת שאיתה קיימנו חיבור ארוך.
ניתן לראות שזהו השירות האחסון של **google**.

מצורפות תמונות המחשה:





שאלה 8

סעיף א':
פתרון תיאורטי לשיטה:

על מנת לפענח את הצופן הנתון, נפתח את הביטוי ונפשט אותו בצורה מתמטית באופן הבא:

$$AES_3^*\{M\}_{K1,K2\ ,K3} = AES_1^*\{AES_1^*\{AES_1^*\{M\}_{k1}\}_{k2}\}_{k3} = C$$

$$ShiftC(ShiftC(ShiftC(Message), \oplus Key_1)\oplus Key_2)\oplus Key_3 = Cipher$$

$$ShiftC(ShiftC(ShiftC(Message)))\oplus ShiftC(ShiftC(Key_1))\oplus ShirftC(Key_2)\oplus Key_3 = Cipher$$

$$ShirftC(ShirftC(Key_1))\oplus ShirftC(Key_2)\oplus Key_3 = Cipher\oplus ShiftC(ShiftC(ShiftC(Message)))$$

בהינתן **Message** ו-**Cipher** נוכל לדעת את הביטוי הימני במשוואה, נציב:

$$\boldsymbol{A} = Cipher\oplus ShiftC(ShiftC(ShiftC(Message)))$$

נקבל:

$$ShirftC\big(ShirftC(Key_1)\big)\oplus ShirftC(Key_2)\oplus Key_3 = A$$

מתכונות ה*XOR* נוכל לבחור שרירותית 2 מספרים שייצגו את $ShirftC\big(ShirftC(Key_1)\big)$ ואת $ShirftC(Key_2)$. ובכך למצוא את המפתח Key_3 בהינתן המשוואה:

$$K_1''\oplus K_2'\oplus Key_3 = A \rightarrow\ Key_3 = K_1''\oplus K_2'\oplus A$$

לאחר מכן מה שיישאר לעשות זה את הפעולה ההפוכה לShiftC בכדי למצוא את המפתח Key1 ו-Key2.

$$ShirftC^{-1}(ShiftC^{-1}(K_1'') = \ Key_1$$
$$ShiftC^{-1}(K_2') = \ Key_2$$

בכך בהינתן Message-וCipher נוכל לדעת מהם שלושת המפתחות Key1,Key2-וKey3 אשר שימשו להצפנת AES_3^*.

*ניתן לפשט את מציאת המפתחות על ידי בחירת מפתחות אשר אינם רגישים לSHIFT כמו מפתח שכולו אחדות ומפתח שכולו אפסים.