מבוא לקריפטוגרפיה / תרגיל בית 1#

אריאל סטולרמן
ודים סטוטלנד

(1)

יהי p ראשוני בן 128 ביטים, $\mathbb{Z}_p = \{0,1,\dots,p-1\}$, $a,b \in \mathbb{Z}_p$ ו-$a \neq 0$. נתונה ההצפנה הבאה להודעה $M \in \mathbb{Z}_p$: $E_{a,b}(M) = aM + b \bmod p$.

(a)

להלן הוכחה כי אם נעשה ב-E שימוש להצפנת הודעה בודדת $M \in \mathbb{Z}_p$ אז המערכת הינה מערכת הצפנה מושלמת:

כדי להראות שהמערכת היא מערכת הצפנה מושלמת, צריך להתקיים: $Pr[plaintext = M] = Pr[plaintext = M|C]$, כאשר $C = E_{a,b}(M)$.

נחשב תחילה את ההסתברות האפריורית לקבלת M: כיוון ש-$M \in \mathbb{Z}_p$ אז ההסתברות לקבלת M בבחירה ראנדומית של מספר כלשהו (הודעה) ב-$\mathbb{Z}_p$ היא כמובן $\frac{1}{p}$. נחשב כעת את ההסתברות לקבלת M בהינתן C ו-E: פונקציית ה-D שלנו היא: $D_{a,b}(C) \coloneqq (C-b)\cdot a^{-1} \bmod p$, כאשר a^{-1} הוא ההופכי הכפלי של a המוגדר לכל $a \neq 0$ כי $\mathbb{Z}_p$ שדה (כי p ראשוני), דהיינו לכל a אפשרי תחת תנאי מערכת ההצפנה. בהינתן C נבחר לו a,b אקראיים כאשר $a \in \mathbb{Z}_p \backslash \{0\}, b \in \mathbb{Z}_p$. למעשה ישנה תלות בין a ו-b, כי לכל $a \neq 0$ קיים $b \in \mathbb{Z}_p$ כך ש-$M = (C-b)\cdot a^{-1}$, ולכן סה"כ הסיכוי לקבל את M הוא:

$$Pr[plaintext = M|\,C] = \underbrace{\frac{1}{p-1}}_{choosing\ \ a} \cdot \underbrace{\frac{1}{p}}_{choosing\ \ b} \cdot \underbrace{(p-1)}_{\#\ of\ \ matc\ hing\ <a,b>} = \frac{1}{p} = Pr[plaintext = M]$$

(b)

נניח משתמשים במערכת להצפנת שתי הודעות M_1, M_2. נניח כי $M_1 = M_2$. אפריורי, הסיכוי לנחש שתי הודעות הוא $\frac{1}{p^2}$, כיוון שהניחוש לכל הודעה נעשה באופן בלתי תלוי בהודעה השניה. לעומת זאת, במקרה בו $M_1 = M_2$, ובהינתן $ciphertext$ עבור שתי ההודעות, רואים מה-$ciphertext$ ששתי ההצפנות זהות, ומכאן גם ההודעות המקוריות זהות, לכן מספיק לפענח את אחת ההודעות בסיכוי כאמור $\frac{1}{p}$.

(c)

יהיו M_1, M_2 שתי הודעות ו-C_1, C_2 ההצפנות של אותן הודעות בהתאמה, אזי מתקיים:

- $C_1 = aM_1 + b \bmod p$
- $C_2 = aM_2 + b \bmod p$

במקרה בו $M_1 \neq M_2$ מתקבלות שתי משוואות שונות עם שני נעלמים, ואז הסיכוי למצוא את a,b הנכונים הוא 1. במקרה בו $M_1 = M_2$, מתקבלת משוואת תלות של a ב-b, ואז הסיכוי למצוא את a,b הנכונים הוא $\frac{1}{p-1}$. סה"כ:

$$\underbrace{\frac{p-1}{p}}_{chance\ \ of\ \ M_1 \neq M_2} \cdot 1 + \underbrace{\frac{1}{p}}_{chance\ \ of\ \ M_1 = M_2} \cdot \underbrace{\frac{1}{p-1}}_{\substack{chance\ \ of\ guessing \\ 'a'\ correctly}} = \frac{p-2}{p-1}$$

וסיכוי זה קרוב ל-1.

(2)

(a) פענוח ה-$ciphertext$ הוא לשיר שילדים האנגלי הידוע "*Mary Mary quite Contrary*":

"Mary, Mary, quite contrary,
How does your garden grow?"

(*b*)

להלן תיאור אלגוריתם לפענוח פלטי מערכת ההצפנה המתוארת:

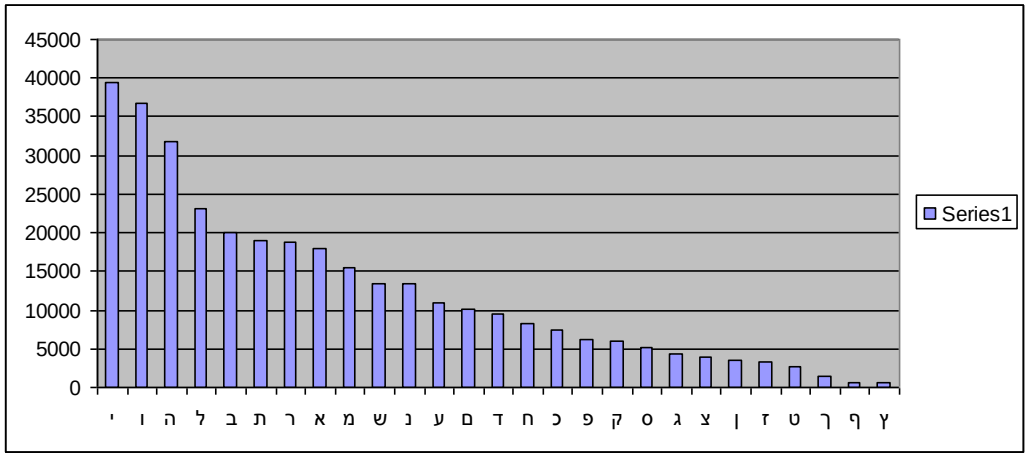
בשלב ראשון יש לבדוק את התפלגות האותיות ב-*ciphertext* ולהשוות להתפלגות האותיות בכל המילים בשפה האנגלית. כיוון שההנחה היא שהטקסט ארוך מספיק, סביר שהחלפת האותיות בהתאם למיקומן בהתפלגות תהיה קרובה מאוד עד מדוייקת. בשלב השני יש לבדוק את כל אופציות חלוקת הטקסט לסגמנטים, ולכל אורך סגמנט לבדוק את כל אופציות כתיבתו כמטריצה (לכל *n,m* אפשריים), אותו אלגוריתם שהופעל בסעיף הקודם (פולינומיאלי). לבסוף יש למצוא את הטקסט המכיל מילים בעלות משמעות, וזאת ניתן לעשות ע"י חיפוש ראנדומי של מילים מילוניות באורך לא קצר מדי בקונטקסט המשוער של ההודעה (למשל "*attack*" באם מדובר בהודעה צבאית מוצפנת) או כל דרך דומה אחרת.

דרך נוספת, אם לא נסתמך על כך שהטקסט המוצפן על אף אורכו יתפלג באותיות המרכיבות אותו כמו כלל השפה האנגלית, היא תחילה לחלק את הטקסט לכל הסגמנטים והמטריצות האפשריות, ולקבל מחרוזות רציפות שאחת מהן עם החלפת האותיות הנכונה היא הטקסט הנכון. כעת ניתן לבצע בדיקת התפלגות על הרצפים שהתקבלו לא רק על אותיות בודדות אלא גם סטטיסטיקת זוגות או רצפים ארוכים יותר. כעת יש לבצע את הבדיקות הסטטיסטיות על התפלגות האותיות (פרמוטציה כלשהי שלהן) על כל הרצפים, כאשר הרצפים שהתפלגות האותיות/זוגות/רצפים ארוכים יותר שלהם דומים עד שווים להתפלגות האמיתית בשפה העברית, הם מועמדים טובים להחלפה.

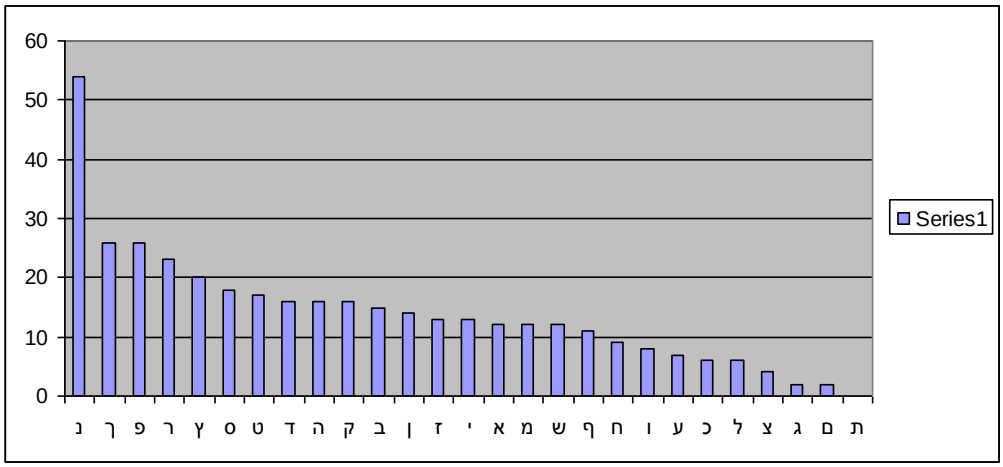
(3)

(*a*)

המסמך שנלקח כבדיקה מהאתר *http://benyehuda.org/* הוא *http://benyehuda.org/carmi_z/mexanex_vedarko.html*. תוצאות התפלגות האותיות במסמך:



(*b*) להלן התפלגות האותיות ב-*cipher1.txt*:



ולהלן הטקסט לאחר החלפת האותיות בהתאם לסטטיסטיקה מסעיף (*a*) (ללא אותיות סופיות, כלומר ״מ״ במקום ״ם״ וכו׳):

בחתנ הדפי ,קפל אמל
:יעקר ,יכלכ קת-הכנשל
בחרמ והו סנדת חש יש
.יעבנ סבמ-הארי

:חתו ינהכ תדפ שיוי
-- יהבג-קהתנ אש פנת ,עמ
,כחרח דגל ערמ וני
.יהביובול כינא

- ׳יכלו ׳יכלו - חיחת התרל
׳בורמל תהכו פאנא
עחו יבזנ פשב אש דצמ
.בוג יאהל דשי

בחתנ הדפי ,קפל אמל
:יעקר ,יכלכ קת-הכנשל
בחרמ והו סנדת חש יש
.יעבנ סבמ-הארי

׳סימצנו סר המל וביז״
׳ימבנג-דהת ,ימפ-עתר
סימג ,כימא בהזבז-תימר
״׳ימיזרל פצתר

,איזו-ונפ ערימ יעימ
,איקדיע יהמהמ ,סימצנוי
,עימרו במהל פיפי
׳ציקמי ממיל

׳סימצנו אש ,יאדצק תש״
׳תגו העדה ,פקלתש
״׳שלה ׳שלה ׳וודג-כיה יו
.תגנ יאיעתל

למרות ההחלפה עדיין לא ניתן לקרוא את הטקסט, ומקור הבעיה הוא כנראה העובדה שהטקסט המוצפן קצר מדי ולכן לא מתאים לשכיחות האותיות בשפה העברית, כפי שניתן לשער שמתקבלת בסעיף (*a*) בו הסטטיסטיקה מתבססת על מסמך בן עשרות אלפי אותיות. גם אם הטקסט היה ארוך, עדיין לא בטוח שהיה ארוך מספיק, שכן אותיות רבות קרובות בהתפלגותן, ובכלל שיטה זו תלויה בגורמים שונים שמשפיעים מאוד על פענוח נכון (כמו תוכן המסמך, שיכול להכיל מילים/מושגים ספציפיים פעמים רבות ובכך להעלות את ה״מיקום״ של אותיות מסויימות ביחס להתפלגותן בשפה העברית). אורך הטקסט הקצר פוסל את האפשרות שבדיקת התפלגות רצפים באורך שתי אותיות או יותר יכולה להניב מידע חשוב בהסתברות גבוהה.

(*c*)

הימצאות המחרוזת *"YALDI HATZACH"* בטקסט מסייעת בפענוח 6 אותיות מתוך הטקסט ע״י מציאת תבנית המתאימה למחרוזת ״ילדי הצח״ בטקסט המוצפן, אך לא מספיקה כדי לפענח את הטקסט כולו. מחיפוש קל בגוגל ניתן להבין שכנראה טקסט זה הוא תרגומו של אהרון אמיר ״פטעוני״ ל-*"Jabberwocky"* של *Lewis Carrol*; מבנה הטקסט בבתים, סימני הפיסוק ותבניות שונות בו מעידות שאכן זהו הטקסט:

בְּעֵת בָּשָׁק וּשְׁלֵי פַּחְזָר,
בַּאַפְּסֵי חָק סָבְסוּ, מָקְדוּ,
אוֹ אָז חִלְכֵן הָיָה נִמְזַר,
וּמְתֵי עָרָן כִּרְדוּ.

וְהוּא שָׁלַף סִיפּוֹ הָחַז,
נָד, חִפֵּשׂ אֶת פְּחִיק צָרָיו –
וְכֹה עָמַד בְּצֵל זַמְתָם,
תָּפוּס בְּהִרְהוּרָיו.

בְּמֶחִי חָזוּז – הָבֵס! הָבֵס!
תִּכְתֶּךָ הַסַּיִף בְּנִמְהַר!
נָטַל אֶת רֹאשׁ פִּגְרוֹ הַזַּד,
וְאֶל בֵּיתוֹ צָהַר.

בְּעֵת בָּשָׁק וּשְׁלֵי פַּחְזָר,
בְּאַפְּסֵי חָק סָבְסוּ, מָקְדוּ,
אוֹ אָז חִלְכֵן הָיָה נִמְזַר,
וּמְתֵי עָרָן כִּרְדוּ.

״גּוּרָה בָּנַי מִן הַפִּטְעוֹן!
מֵחַד-שִׁנּוֹ, חִיל-צְפִרְנוֹ!
מֵעוֹף-גִּרְגִּיר תָּנוּס, צָעוֹן,
מֵחֶטֶשׁ בְּמִגּוֹנוּ!״

עוֹדוֹ עוֹמֵד שָׁפָה-הָגוּת,
וְהַפִּטְעוֹן, עֵינָיו דּוֹלְקוֹת,
וְשָׁוֵשׁ בַּיָּעָר הַמָּגוּד,
בּוֹעֵעַ וְנָקוֹט!

״אַף קָטַלְתּוֹ, אֶת הַפִּטְעוֹן?
אֲחַבֶּקְךָ, יַלְדִּי הַצָּח!
הוֹ, יוֹם-צְלָהָה! יַבֵּא! יַבֵּא!״
בְּחֶדְוָתוֹ פָּצַח.

בהמשך לסעיף (*b*), ניתן לראות קרבה מסויימת להתפלגות שעלתה מסעיף (*a*), למשל מהמשפט ״הבס! הבס!״ ניתן להבין כי מה שהוצפן לאות ה׳ תורגם בטעות לאות ו׳, כאשר שתי אותיות אלו קרובות בהתפלגותן; כמו כן מהמשפט ״יבא! יבא!״ נראה כי י׳ תורגמה בטעות ל-ה׳, גם כן קרובות בהתפלגותן (ו-י׳ תורגמה בטעות ל-ו׳).

(4)

(*a*)

מערכת *RTAU* יכולה לעבוד באופן הבא: נסמן את קבוצת כלל המשתמשים ב- U (בהתחלה $S = U$). בכל רגע נתון השידור ישודר מוצפן עם המפתח $R_{k_1} \oplus \ldots \oplus R_{k_l}$ כאשר $k_1, \ldots, k_l$ הם המנויים **שאינם** מחוברים לשירות (יתכן שיתחברו במועד מאוחר יותר). עדכון מרכיבי המפתח משידור לשידור יועבר כנתון באופן לא מוצפן לכולם. כאשר מנוי מספר k_{l+1} מתנתק מהשירות, תשודר רשימת מספרי המנויים שהתנתקו (או לא היו מחוברים מעולם) עד כה $k_1, \ldots, k_l, k_{l+1}$, והמפתח החדש יהיה $R_{k_1} \oplus \ldots \oplus R_{k_{l+1}}$. מובן כי בעת תחילת השידור במפתח החדש המשתמש ה- k_{l+1} (וכל מי שאינו מנוי עד כה) אינו מחזיק במספיק אינפורמציה כדי לפענח את התשדורת, שכן באפשרותו לבנות את המפתח $R_{k_1} \oplus \ldots \oplus R_{k_l}$ בלבד. ניתן להסתכל על כך כאילו התשדורת מבחינתו מוצפנת במפתח $R_{k_{l+1}}$ שהוא המפתח היחיד שאין לו.

(*b*)

נתוני השאלה מעידים כי לא ניתן לשחק עם המכשירים של המנויים, ומכך משתמע שלא ניתן להוציא ממכשיר את רשימת המפתחות שבו. בדרך זו, למעשה, אין שום אפשרות לשני מנויים לפענח את השידור. אם נתעלם לרגע מנתון זה, שני מנויים (לשעבר) בהחלט יכולים לפענח את השידור. מרכיבי המפתח משודרים באופן לא מוצפן ולכן שני הפורצים יכולים להאזין למרכבי המפתח בכל עת. אם באפשרותם לשלוף את המפתחות מתוך המכשירים שלהם, אז כמובן שיחד יש להם את רשימת המפתחות המלאה שיש ברשות התחנה, (לאחד יש את המפתח שאין לשני ולהיפך) ולכן יחד יכולים לבנות את המפתח המתאים לכל שידור בכל עת (XOR של כל המפתחות המרכיבים את המפתח הנוכחי המשמש להצפנה).

(5)

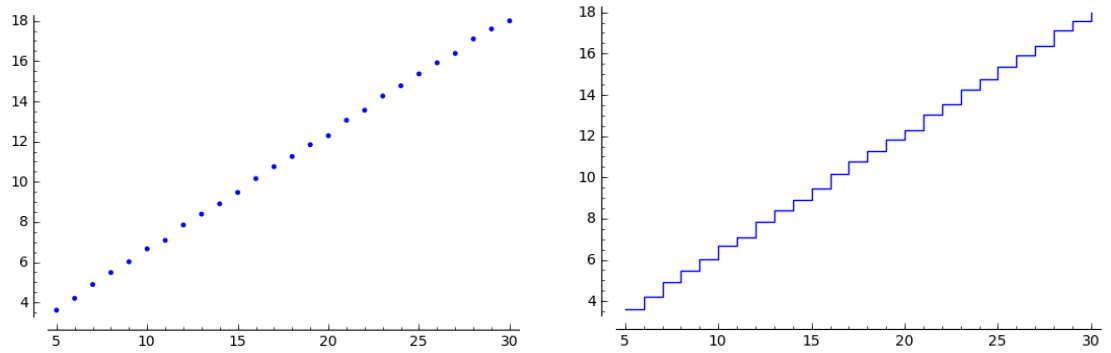
להלן הקוד:

```
from scipy import polyfit

average_arr = []
maxcount_arr = []
max_ab_arr = []

top = 1000
for n in range(5,31):
    sum = 0
    max_count = 0
    max_ab = (0,0)
    low_bound = (2^n)+1
    high_bound = 2^(n+1)
    for i in range(1,top+1):
        a = randint(low_bound,high_bound)
        b = randint(low_bound,high_bound)
        if a<b:
            c = a
            a = b
            b = c
        tmp_ab = (a,b)
        count = 0
        while b<>0:
            t = b
            b = a%b
            a = t
            count +=1
        sum += count
        if max_count < count:
            max_count = count
            max_ab = tmp_ab
    average_arr = average_arr+[(n,sum/top)]
    maxcount_arr = maxcount_arr+[(n,max_count)]
    max_ab_arr=max_ab_arr+[max_ab]

print("average array:")
print(average_arr)
print("max-count array:")
print(maxcount_arr)
print("max-count a's and b's array:")
print(max_ab_arr)

point(average_arr)
plot_step_function(average_arr)
point(maxcount_arr)
plot_step_function(maxcount_arr)

X,Y=zip(*average_arr)
a,b = polyfit(X,Y,1)
print "y =",a,"*x+",b

X,Y=zip(*maxcount_arr)
a,b = polyfit(X,Y,1)
print "y =",a,"*x+",b
```

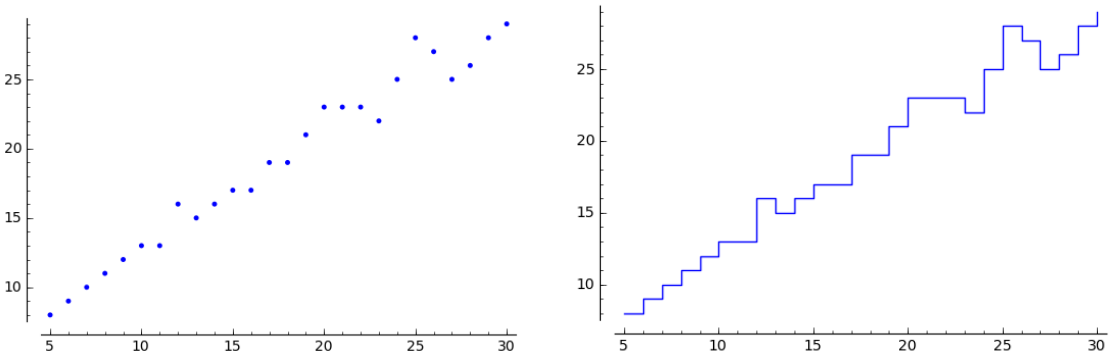
פלט:

```
average array:
[(5, 3613/1000), (6, 2101/500), (7, 1223/250), (8, 1371/250), (9,
6023/1000), (10, 6663/1000), (11, 3543/500), (12, 7851/1000), (13,
1049/125), (14, 1781/200), (15, 9469/1000), (16, 5079/500), (17,
10749/1000), (18, 1407/125), (19, 11843/1000), (20, 12289/1000), (21,
13061/1000), (22, 6779/500), (23, 14263/1000), (24, 14773/1000), (25,
3841/250), (26, 3979/250), (27, 2048/125), (28, 2138/125), (29,
17599/1000), (30, 18003/1000)]
max-count array:
[(5, 8), (6, 9), (7, 10), (8, 11), (9, 12), (10, 13), (11, 13), (12,
16), (13, 15), (14, 16), (15, 17), (16, 17), (17, 19), (18, 19), (19,
21), (20, 23), (21, 23), (22, 23), (23, 22), (24, 25), (25, 28), (26,
27), (27, 25), (28, 26), (29, 28), (30, 29)]
max-count a's and b's array:
[(55, 34), (123, 89), (199, 144), (466, 289), (971, 759), (1999, 1269),
(3677, 2684), (7064, 4381), (16001, 9802), (31159, 27925), (65221,
41290), (121744, 74201), (198434, 137876), (439165, 339786), (1040330,
760997), (1692015, 1325132), (4068051, 2855264), (5576584, 4844617),
(16345832, 13490569), (33432149, 24045008), (63684470, 40123912),
(111720309, 80842369), (257137156, 160174519), (456210703, 403036874),
(1014730957, 557493457), (2109457971, 1212265509)]
```

Averages graph:



Maximum count graph:




```
Estimates:
average: y = 0.579774358974 *x+ 0.811641025641
max_count: y = 0.830427350427 *x+ 4.50598290598
```

(6)

- פולינום אי פריק מעל השדה $GF(2)$: $x^4 + x + 1$

```
x = polygen(GF(2)) # make x the variable of the polynomial field
f = x^4 + x + 1 # insert your favorite polynomial here
assert f.is_irreducible()
print(f.is_irreducible())
print(factor(f))
output:
> True
> x^4 + x + 1
```

- להלן רשימת כל האלמנטים הפרמיטיביים ב-$GF(2^4)$:

```
G16.<u> = GF(2^4, modulus=f) # u is the generator's name
for a4 in range(0,2):
    for a3 in range(0,2):
        for a2 in range(0,2):
            for a1 in range(0,2):
                for a0 in range(0,2):
                    y=a4*u^4+a3*u^3+a2*u^2+a1*u + a0
                    z=y
                    i=1
                    while z<>1 & i!=20:
                        z=z*y
                        i=i+1
                    if i==2^4-1:
                        print(y)
output:
> u
> u + 1
> u^2
> u^2 + 1
> u^3 + 1
> u^3 + u + 1
> u^3 + u^2 + 1
> u^3 + u^2 + u
…
```

ישנם $2^3 = 8$ איברים פרמיטיביים ב-$GF(2^4)$.

- עבור $GF(2^5)$ השתמשנו באותו קוד לעיל (פרט להחלפות היכן שהיה צריך). מצאנו כי הפולינום $y = x^5 + x^3 + x^2 + x + 1$ הוא אי-פריק בשדה זה וקיבלנו 30 איברים פרימיטיביים שהם כל הפולינומים בשדה פרט לפולינומים הקבועים 0 ו-1. אם נסתכל על הקבוצה הכפלית $GF^*(2^5)$ (ללא האיבר 0), נקבל שיש בה 31 איברים – מספר ראשוני. כיוון שסדר כל איבר בקבוצה מחלק את גודלה, אזי סדר כל איבר בקבוצה הוא או 1 או 31. היחיד שסדרו 1 הוא כמובן 1 (והוא מן הסתם לא התקבל בבדיקה כי הוא יוצר רק את עצמו), בעוד ששאר האיברים חייבים להיות מסדר 31. כיוון שהם מסדר 31, הם חייבים לפרוש את כל איברי הקבוצה בחזקות עד 31, אחרת היו "מתאפסים" ביחס לשדה ואז סדרם היה קטן מ- 31, בסתירה למשפט לגרנג'. לכן במקרה זה קיבלנו את כל הפולינומים פרט ל-1 כאיברים פרימיטיביים ב-$GF^*(2^5)$. ב-$GF^*(2^4)$ המקרה אינו כזה כיוון שמספר האיברים בקבוצה הוא 15, שהוא כמובן אינו ראשוני והנ"ל לא מתקיים.
- בהפעלת אלגוריתם gcd על שני מספרים ראשוניים באופן יחסי $a = 15162, b = 993631$ קיבלנו 11 פעולות mod, וההופכי כפלי של a בשדה $\mathbb{Z}_b$ הוא 975478.

```
a = randint(10000,100000)
print(factor(a))
print(a)
b = randint(100000,1000000)
print(factor(b))
print(b)
i = 0
a_tmp=a
b_tmp=b
while b_tmp!=0:
    t = b_tmp
    b_tmp = a_tmp%b_tmp
    i = i+1
    a_tmp = t
print(i)
print(a_tmp)    #gcd(a,b)
print(gcd(a,b))

a_tmp=a
b_tmp=b
x=0
y=1
last_x=1
last_y=0
while b_tmp<>0:
    quotient = (a_tmp-Integer(mod(a_tmp,b_tmp)))/b_tmp
    temp = b_tmp
    b_tmp = Integer(mod(a_tmp,b_tmp))
    a_tmp = temp

    temp = x
    x = last_x-quotient*x
```

```
    last_x = temp

    temp = y
    y = last_y - quotient*y
    last_y = temp
ans = (a_tmp,last_x,last_y)
print(ans)
print(xgcd(a,b))

c=Integer(mod(last_x,b))
print(c)
print(mod(c*a,b))

output:
=======
2 * 3 * 7 * 19^2
15162                   #a
229 * 4339
993631                  #b
11                      # number of mod operations in gcd(a,b)
1                       # gcd algorithm above result
1                       # sage's gcd result
(1, -18153, 277)        # xgcd algorithm above result
(1, -18153, 277)        # sage's xgcd result
975478          # a's inverse in Zb
1                       # check that...
```