

המרחב של המספרים Z של n שעבורם ישנו מכפיל הופכי במודולו n ניקרא: Z של n כוכבית. אלה הם גם המספרים מתוך Z של n (השאריות בחלוקה ב- n , כזכור), שהם ראשוניים ביחס ל- n .

קצת נגדיר מהי הפונקציה: $\phi(n)$ של אוילר - זה מספר המספרים הקטנים מ- n , ושהינם ראשוניים ביחס ל- n , כלומר שאין להם מחלקים משותפים. לדוגמה, נתון $n=10$

$$\phi(10) = 4$$

$$\{1, 3, 7, 9\}$$

(ניקראים: Relative prime למשל ל-1, 3, 7, 9 אין מחלק משותף גדול מ-1 עם 10).
זהו למעשה המרחב Z של n כוכבית (כאשר $n=10$)

לא מאמינים? בואו נבדוק: מה ההפך המכפילי של 1 מודולו 10? 1 כמובן. מה עם 3? זהו: 7 (ולכן ההפך המכפילי של 7 הוא 3). מהו ההפך המכפילי של 9? התשובה היא 9, כמובן.

$$1 \cdot 1 \bmod 10 = 1; \quad 3 \cdot 7 \bmod 10 = 1; \quad 7 \cdot 3 \bmod 10 = 1; \quad 9 \cdot 9 \bmod 10 = 1$$

עבור מה אנו צריכים בכלל את: $\phi(n)$? זה ישמש עבורנו טווח לבחירת המפתח הציבורי ומציאת הפרטי (הפך מכפילי מודולו). $\phi(n)$ זה למעשה מספר האיברים בקבוצה Z של n כוכבית. ראינו למשל ש: $n=10$ נתן לנו 4 אפשרויות, אבל אנחנו מעוניינים באפשרויות רבות מאד (כדי שהקוד יהיה קשה מאד לפיצוח) ולכן נלך על מספרים ראשוניים שלהם יש יותר אפשרויות.

ישנה ילמה (Theorem באנגלית) שאומרת שאם P הוא ראשוני, אזי $\phi(p) = p-1$
לא נוכיח זאת כאן מפאת קוצר היריעה, אבל זה די אינטואיטיבי מההגדרה של מספר ראשוני. למשל אם $p=11$ אז מהם המספרים בין 1 ל-10 שאין להם מחלק משותף עם 11? 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

וכמו כן אם p ו- q הם ראשוניים ($n=p \cdot q$), אז מתקיים ש:

$$\phi(n) = \phi(p \cdot q) = (p-1)(q-1)$$

לדוגמה: אם $p=11$ ו- $q=7$ אז:

$$\phi(77) = 60$$

אתם מוזמנים לבדוק זאת בבית אם לא מאמינים לי. או שתבחרו דוגמה פשוטה יותר כמו: 5 ו-3, ואז עבור 15 תגלו שיש 8 מספרים שהם ראשוניים ביחס ל-15 בתחום שבין 1 לפחות מ-15. (1, 2, 4, 7, 8, 11, 13, 14). זהו בעצם Z של 15 כוכבית.
מדוע יש 8 כאלה? כי: $(5-1) \cdot (3-1) = 8$

ישנו משפט שאומר שאם a שייך ל- Z של n כוכבית, אזי: $a^{\phi(n)} \bmod n = 1$
לדוגמה אפשר לנסות 2 בחזקת 60 מודולו 77 - ותסמכו עלי שזה יוצא 1 (אפשר לחשב עם מחשבון על ידי העלאת 2 בחזקת 20, לקיחת מודולו 77, וזה בחזקת 3, ושוב מודולו 77).

בגלל תכונה זו, על מנת למצוא את הזוג של המספרים: e ו- d עבור האלגוריתם RSA שיעבוד, בוחרים את המפתח הציבורי: e ואז מוצאים את ההפך מכפילי שלו $\bmod(\phi(n))$ שיהיה ה- d .

דוגמה קטנה לאיך עובד RSA: אם בחרנו את $p=7$ ו- $q=11$ הרי ש: $n=77$. ואז ראינו כבר ש- $\phi(n)=60$. כלומר יש לנו 59 (בלי 1) ערכים לבחור מהם את e . נניח שאליס בחרה מפתח ציבורי של (77, 13) כי 13 הוא ראשוני ביחס ל 60. ולכן המפתח הפרטי שלה יחושב כהפך המכפילי במוד 60 הוא 37. כיוון ש: $1 = 481 \bmod 60 = 37 \cdot 13 \bmod 60$ ולכן המפתח הפרטי של אליס הוא: (77, 37)

נניח שבוב רוצה לשלוח הודעה x שהיא המספר 5. בוב יחשב את: $y = x^{13} \bmod 77 = 5^{13} \bmod 77$
שזה יוצא: 26 (תנסו) ולכן הוא שולח לאליס 26. אליס תפענח: $z = y^{37} \bmod 77 = 26^{37} \bmod 77$
בואו נחשב זאת:

אפשר לבצע: $26^{10} \pmod{77}$ במחשבון ואז $\pmod{77}$, ואז להכפיל 3 פעמים ואז להכפיל ב- 26^7 , כלומר:
 $26^{37} \pmod{77} = 26^{10} \pmod{77} \times 26^{10} \pmod{77} \times 26^{10} \pmod{77} \times 26^7 \pmod{77}$
 $77 = 23 \times 23 \times 23 \times 5 \pmod{77} = 5$

כעת, בהקשר של פיתוח האלגוריתם של RSA 'אמיתי' דיברנו על **מספרים גדולים**. לאחר מציאת המפתח הציבורי יהיו לנו e ו- n (המודולו). נישתמש בשני הראשוניים p ו- q כאשר $n = p \cdot q$, ולכן

$$\text{PHI}(n) = (p-1) \cdot (q-1)$$

מומלץ עבור כ"א מהראשוניים להיות מספר בעל 512 ספרות בינאריות (שיתורגם לעשרוני עם 154 ספרות), ולכן n (המודולו) בעל 1024 ספרות בינאריות או 309 עשרוניות. כל הקושי של 'ניחוש' ההפך המכפילי טמון בכך שכדי לפרק את n לגורמים ידרש אלגוריתם אקפוננציאלי וכאשר n הוא כזה גדול זה יהיה למעשה בלתי אפשרי.

וכעת לאלגוריתם עצמו של RSA

```
select 2 large prime numbers p and q where p != q
p*q -> n
(p-1)(q-1) -> PHI(n)
select e such that 1 < e < PHI(n) and e is co-prime with PHI(n)
find d multiplicative inverse of e mod PHI(n) //Extended Euclidean
(e,n) -> public_key
d-> private_key //secret
```

כדי להצפין טקסט נישתמש במפתח הציבורי (e, n) אם הטקסט ארוך מ- n נחלקו לבלוקים קטנים מ- n . ההצפנה תעשה כך (ע"י שימוש באלגוריתם של חזקות מהירות - Fast exponentiation) שנוכל לראות מאוחר יותר, אם הזמן יאפשר זאת:

Fast_exponentiation(P, e, n) -> C // $P^e \pmod{n}$ - P plain text, C cipher text

פיענוח של C יעשה כך:

Fast_exponentiation(C, d, n) -> P // $C^d \pmod{n}$ - P plain text, C cipher text

בואו נראה דוגמה כמעט טריביאלית: ניבחר לדוגמה את 7 ו-11 כ- p ו- q .

$$n = 7 \cdot 11 = 77, \text{PHI}(77) = (11-1) \cdot (7-1) = 60$$

ניבחר כעת את e להיות 13 ונימצא את ההפך המכפילי שלו מודולו 60 (בעזרת האלגוריתם המורחב של אוקלידס). אחסוך לכם שבירת הראש כרגע, זהו 37 ($37 \cdot 13 \pmod{60} = 481 \pmod{60} = 1$) כלומר $d = 37$

נניח שנירצה לשלוח טקסט פשוט כמו המספר 5:

$$C = 5^{13} \pmod{77} = 1220703125 \pmod{77} = 26$$

ולכן הטקסט המוצפן שלנו יהיה 26.

ע"מ לפענח אותו, נשתמש במפתח הפרטי: 37

$$P = 26^{37} \pmod{77} = 5$$

האלגוריתם שעדיין חסר לנו הוא: **Fast exponentiation** והוא מבוסס על ריבוע והכפלה. רוצים לחשב את: $y = a^x$ נהפוך את החזקה (x) למספר בינארי עם k ספרות למשל: $X_5 X_4 X_3 X_2 X_1 X_0$ כאשר כל X הוא 0 או 1 ובמקרה זה $k=5$ ואז ניתן לייצג את המספר כאוסף של ספרות בינאריות המיוצגות על ידי מבנה: $\text{num}(X_5, X_4, \dots, X_0)$

$$x = \text{num}(X_5, X_4, \dots, X_0)$$

$$y = a^x = a^{\text{num}(X_5, X_4, \dots, X_0)}$$

כעת נבדוק שני מקרים אפשריים עבור החזקה הבינארית (ונראה דוגמה). מקרה אחד שהחזקה הבינארית מסתיימת ב-0, ומקרה שני שהיא מסתיימת ב-1. למשל 1010 (זוגי) ו-1011 (אי זוגי) בהתאמה. אני טוען שאם נשמיט את הספרה הימנית שהיא 0, כאילו חילקנו בשתיים (ואין שארית) ולכן:

$$a^{\text{num}(X_5, X_4, \dots, X_0)} = a^{\text{num}[(X_5, X_4, \dots, X_1) * 2]} = \{a^{\text{num}[(X_5, X_4, \dots, X_1)]}\}^2$$

ואם נשמיט את הספרה הימנית שהיא 1, כאילו חילקנו בשתיים עם שארית 1 ולכן:

$$a^{\text{num}(X_5, X_4, \dots, X_0)} = a^{\text{num}[(X_5, X_4, \dots, X_1) * 2 + 1]} = a * \{a^{\text{num}[(X_5, X_4, \dots, X_1)]}\}^2$$

בדוגמה הנ"ל: $a^{1010} = a^{[101] * 2} = [a^{101}]^2$ וכמו כן: $a^{1011} = a^{[101] * 2 + 1} = [a^{101}]^2 * a$

וכך ניתן לחתוך שוב סיפרה מימין ולהמשיך בתהליך. מכאן שהאלגוריתם יעבוד בשיטה של ריבוע התוצאה הקודמת ואם הסיפרה הימנית הייתה 1, גם נכפיל ב- a , ואם 0, לא נכפיל. כל פעם ניקח מודולו כדי לא לקבל מספרים גדולים מידי כי כבר אמרנו ש:

$$a * b \bmod n = [(a \bmod n) * (b \bmod n)] \bmod n$$

```
//n - mod, a - base, x - exponent
turn x to bit string (binary) -> xbits
length(xbits) -> k
1 -> y //Initializing with 1 for multiplications
for(int i=0; i<k; ++i)
{
  y=y*y%n; //square previous
  if(xbits[i] == '1')
    y=a*y%n; //also multiply by a
}
return y;
```

הערה: האלגוריתם עובד עבור חזקות גם בלי המודולו.