אבטחת מידע – מטלה 3

אלעזר פיין

1. אלגוריתם: עבור כל מספר מ 1 עד n-1 נבנה את קבוצת החזקות שלו מודולו n ונבדוק אם היא שווה לקבוצת המספרים הזרים ל-n, אם כן הוא שורש פרימיטיבי.

בפייתון:

```
def primitive_roots(n):
    n_coprimes_set = {x for x in range(1, n) if gcd(x, n) == 1}

    p_roots = []
    for x in range(1, n):
        x_powers_mod_set = {pow(x, power, n) for power in range(1, n)}
        if x_powers_mod_set == n_coprimes_set:
            p_roots.append(x)

    return p_roots
```

נקבל את קבוצת המספרים הזרים ל-11:

```
n_coprimes_set for n=11:      {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

תוצאות כל בדיקות כל המספרים בטווח:

1^1 ≡ 1 (mod 11)	2^1 ≡ 2 (mod 11)	3^1 ≡ 3 (mod 11)	4^1 ≡ 4 (mod 11)	5^1 ≡ 5 (mod 11)
1^2 ≡ 1 (mod 11)	2^2 ≡ 4 (mod 11)	3^2 ≡ 9 (mod 11)	4^2 ≡ 5 (mod 11)	5^2 ≡ 3 (mod 11)
1^3 ≡ 1 (mod 11)	2^3 ≡ 8 (mod 11)	3^3 ≡ 5 (mod 11)	4^3 ≡ 9 (mod 11)	5^3 ≡ 4 (mod 11)
1^4 ≡ 1 (mod 11)	2^4 ≡ 5 (mod 11)	3^4 ≡ 4 (mod 11)	4^4 ≡ 3 (mod 11)	5^4 ≡ 9 (mod 11)
1^5 ≡ 1 (mod 11)	2^5 ≡ 10 (mod 11)	3^5 ≡ 1 (mod 11)	4^5 ≡ 1 (mod 11)	5^5 ≡ 1 (mod 11)
1^6 ≡ 1 (mod 11)	2^6 ≡ 9 (mod 11)	3^6 ≡ 3 (mod 11)	4^6 ≡ 4 (mod 11)	5^6 ≡ 5 (mod 11)
1^7 ≡ 1 (mod 11)	2^7 ≡ 7 (mod 11)	3^7 ≡ 9 (mod 11)	4^7 ≡ 5 (mod 11)	5^7 ≡ 3 (mod 11)
1^8 ≡ 1 (mod 11)	2^8 ≡ 3 (mod 11)	3^8 ≡ 5 (mod 11)	4^8 ≡ 9 (mod 11)	5^8 ≡ 4 (mod 11)
1^9 ≡ 1 (mod 11)	2^9 ≡ 6 (mod 11)	3^9 ≡ 4 (mod 11)	4^9 ≡ 3 (mod 11)	5^9 ≡ 9 (mod 11)
1^10 ≡ 1 (mod 11)	2^10 ≡ 1 (mod 11)	3^10 ≡ 1 (mod 11)	4^10 ≡ 1 (mod 11)	5^10 ≡ 1 (mod 11)
6^1 ≡ 6 (mod 11)	7^1 ≡ 7 (mod 11)	8^1 ≡ 8 (mod 11)	9^1 ≡ 9 (mod 11)	10^1 ≡ 10 (mod 11)
6^2 ≡ 3 (mod 11)	7^2 ≡ 5 (mod 11)	8^2 ≡ 9 (mod 11)	9^2 ≡ 4 (mod 11)	10^2 ≡ 1 (mod 11)
6^3 ≡ 7 (mod 11)	7^3 ≡ 2 (mod 11)	8^3 ≡ 6 (mod 11)	9^3 ≡ 3 (mod 11)	10^3 ≡ 10 (mod 11)
6^4 ≡ 9 (mod 11)	7^4 ≡ 3 (mod 11)	8^4 ≡ 4 (mod 11)	9^4 ≡ 5 (mod 11)	10^4 ≡ 1 (mod 11)
6^5 ≡ 10 (mod 11)	7^5 ≡ 10 (mod 11)	8^5 ≡ 10 (mod 11)	9^5 ≡ 1 (mod 11)	10^5 ≡ 10 (mod 11)
6^6 ≡ 5 (mod 11)	7^6 ≡ 4 (mod 11)	8^6 ≡ 3 (mod 11)	9^6 ≡ 9 (mod 11)	10^6 ≡ 1 (mod 11)
6^7 ≡ 8 (mod 11)	7^7 ≡ 6 (mod 11)	8^7 ≡ 2 (mod 11)	9^7 ≡ 4 (mod 11)	10^7 ≡ 10 (mod 11)
6^8 ≡ 4 (mod 11)	7^8 ≡ 9 (mod 11)	8^8 ≡ 5 (mod 11)	9^8 ≡ 3 (mod 11)	10^8 ≡ 1 (mod 11)
6^9 ≡ 2 (mod 11)	7^9 ≡ 8 (mod 11)	8^9 ≡ 7 (mod 11)	9^9 ≡ 5 (mod 11)	10^9 ≡ 10 (mod 11)
6^10 ≡ 1 (mod 11)	7^10 ≡ 1 (mod 11)	8^10 ≡ 1 (mod 11)	9^10 ≡ 1 (mod 11)	10^10 ≡ 1 (mod 11)

ניתן לראות כי המספרים היחידים העונים על הדרישה הם: **2, 6, 7, 8** ולכן הם השורשים הפרימיטיביים של 11.

.2

א. **CBC (Cipher-Block Chaining)** הוא אופן הפעלה **(Mode of Operation)**, של צפני בלוקים סימטריים דטרמיניסטיים. במצב זה כל בלוק PLAINTEXT משורשר ע"י חיבור XOR עם תוצאת הצפנת הבלוק הקודם. (הבלוק הראשון משורשר עם IV **(Initial-Vector)** – ווקטור אתחול, אותו הצד המקבל מקבל גם.

בהנחה שווקטור האתחול הוא ערך אקראי וחד-פעמי, העתקת בלוק במקום אחר לא תיתן תוצאה צפויה ו2 בלוקים זהים שהוצפנו ע"י אותו מפתח לא יהיו זהים לאחר ההצפנה. תכונות אלו עוזרות למנוע התקפות REPLAY או התקפות קריטפואנליטיות. בנוסף אם פונקציית ההצפנה מקיימת תכונת **Pseudorandomness** (אקראיות מדומה) אז הצופן יהיה חסין נגד **Chosen-plaintext attack**.

CBC מקנה תכונת התאוששות עצמית – בפענוח, במקרה שבלוק שגוי או חסר רק הבלוק הבא לאחריו ייפגע.

בגלל התלות בין הבלוקים CBC אינו מאפשר הצפנה \ פענוח מקבילים או גישה ישירה.

ב. מנגנון **CBC-MAC:**
כל שלושת המנגנונים להלן נועדו ככלל **לעמידות מפני זיוף קיומי תחת מודל התקפת מסר נבחר** (Existential unforgeability) - למנוע מצב בו צד שלישי יוכל לשלוח הודעה ושהצד המקבל יחשוב שהיא אותנטית.
בכללי האלגוריתם המבטיח זאת זה מסומן כך: $\Pi = (\mathrm{GEN}, \mathrm{MAC}, \mathrm{VERIFY})$
כאשר **GEN** הוא אלגוריתם בחירת המפתח, **MAC** הוא אלגוריתם יצירת החתימה, ו **VERIFY** הוא תהליך האימות ע"י הצד המקבל.

להלן פירוט המנגנונים:

I. מניעת **MAC-FORGE** – מוענקת הבטחה הסתברותית לזניחות סיכויי הצלחה של תוקף לזייף חתימה.

II. עמידות מפריצה של החתימה - ע"מ למנוע פריצה של החתימה ע"י **Extension Attack** ישנן מספר שיטות:
   א. 2 הצדדים מסכימים מראש על אורך קבוע L התלוי בפונקציית הקידוד, ויעבירו אך ורק הודעות באורך של כפולות של L (ונרפד עם אפסים בסוף כשצריך).
   ב. נוסיף בלוק המכיל את אורך המסר לתחילת המסר.
   ג. נשתמש בגרסה של **CBC** הנקראת **ECBC** בה מצפינים את התוצאה הסופית במפתח נוסף.
   לאבטחת החתימה, **CBC-MAC** הוא דטרמיניסטי עם פלט יחיד - הערך האחרון.

III. אימות מסרים \ שולח - המקבל יכול לאמת את המסרים שהוא מקבל ע"י חישוב מחדש של החתימה והשוואה עם החתימה אותה הוא מקבל בנוסף למסר.

ג. **Initial-Vector -** ווקטור אתחול, בשרשור בלוקים בצופן זרם מהווה לבלוק הראשון בלוק מקדים איתו הוא משורשר, ופותר את בעיית הדטרמיניסטיות של הצפנת בלוקים שונים ע"י אותו מפתח, כלומר ע"י שימוש בIV ניתן להצפין מסרים ארוכים עם אותו מפתח K ללא חשש שבלוקים זהים יוצפנו באופן זהה (כל עוד

הוא מוחלף עבור כל מסר) ובכך מפשט את תהליך הכנת המפתח (מכיוון שמספיק מפתח אחד לכל הבלוקים).

ד. נסמן את פונקציית ההצפנה לפי המפתח K בסימון E_K.

$$CBC-MAC(M,K) = E_K(m_3 \oplus E_K(m_2 \oplus E_K(m_1 \oplus 0)))$$

(ניתן גם לכתוב ללא האפס מכיוון ש $m_1 \oplus 0 = m_1$)

ה. 1. בוב יצפין את M ע"פ אותה שיטה CBC-MAC ואותו מפתח שמשותף בינו לבין אליס, ויקבל תג (Authenticator) t2.

2. בוב ישווה את t2 עם התג t1 שקיבל בנוסף לM, אם צד שלישי כלשהו שינה חלק מהמסר אז התגים יהיו שונים, ולכן בוב יבחין שהמקור ו\או התוכן לא אותנטיים.

.3

א. הצפנה אסימטרית שמתבססת על חישוב חזקה מודולו של מספר גדול.

ב. הבטחון שלה – חוזק הצפנה, מכיוון שמשתמשת ב2 מפתחות – ציבורי ופרטי, ובנוסף הקושי של פעולת פירוק לגורמים של מספר גדול המורכב ממספרים ראשוניים גדולים.

ג. +

ד. +

ה. :

$$N = p * q = 23 * 5 = 115$$
$$\varphi(N) = (p-1)(q-1) = 22 * 4 = 88$$
$$GCD(19,88) = GCD(19,12) = GCD(12,7) = GCD(7,5) = GCD(5,2) = GCD(2,1) = 1$$

קיבלנו ש E זר ל $\varphi(N)$.

$$d * e \equiv 1 \; mod \; 88$$

אלגוריתם אוקלידס המורחב:

$$d * e + k * \varphi(N) = GCD(19,88)$$
$$88k + 19d = GCD(88,19)$$

c	d	q	r	r = c - q*d	r(a, b)
88	19	4	12	12 = c - 4d	**12 = a - 4b**
19	12	1	7	7 = c - d	7 = b - (a - 4b)= **-a + 5b**
12	7	1	5	5 = c - d	5 = (a - 4b) - (-a + 5b)= **2a - 9b**
7	5	1	2	2 = c - d	2 = (-a + 5b) - (2a - 9b)= **-3a + 14b**
5	2	2	1	1 = c - 2d	1 = (2a - 9b) - 2(-3a + 14b)= **8a - 37b**
2	1	1	0	0 = c - d	0 = (-3a + 14b) - (8a - 37b)
					0 = -11a + 51b

$$k = -11, \qquad d = 51, \qquad GCD(88,19) = 1$$

Public = {19,115} Private = {51,115}

ו. הCIPHERTEXT חייב להיות מספר בייצוג בינארי קטן מ N, נסמן את ההודעה המקורית בm ואת המוצפנת בc, ונקבל כי ההצפנה לפי המפתח הפומבי היא:

$$c = m^e \; mod \; n = m^{19} \; mod \; 115$$

והפענוח יהיה על בסיס המפתח הפרטי:

$$m = c^d \; mod \; n = c^{51} \; mod \; 115$$

ז. לפי מיקום אלפביתי: א = 1, ש = 21 -> PLAINTEXT = 121.

$$C_1 = 1^{19} \; mod \; 115 = 1$$

19	$c_0 = 1$
1	$c_1 \equiv c_0^2 \; * 21^1 = 1^2 \; * 21 = 21 = 21 \; mod \; 115$
0	$c_2 \equiv c_1^2 \; * 21^0 = 21^2 \; * 1 = 441 \; \equiv 96 \; mod \; 115$
0	$c_3 \equiv 96^2 \; * 21^0 = 9216 \; \equiv 99 \; mod \; 115$
1	$c_4 \equiv 99^2 \; * 21^1 = 205821 \equiv 86 \; mod \; 115$
1	$c_5 \equiv 86^2 \; * 21^1 = 155316 \equiv 66 \; mod \; 115$

⇨ $C_2 = \; 21^{19} \; mod \; 115 = 66$

ח. מכיוון ש 166 גדול מ 115 נחלק ל 2 בלוקים.

ט. $m = c^d \; mod \; N = c^{51} \; mod \; 115$

.4

א. פונקציות גיבוב לרוב מהירות יותר מפונקציות הצפנה. בנוסף, אין צורך לפענח את הסיסמא ולכן גם לא דרוש מפתח.

ב. הוספת מלח לסיסמא מונע התקפות מילון או קשת מכיוון שאי אפשר לזהות סיסמאות נפוצות או זהות.

ג. לא ניתן להפוך את ערך הגיבוב.

השתמשו בפונקציית גיבוב SHA-256.

השתמשתי באתר https://crackstation.net שמחפש במאגרים של ערכים מחושבים מראש של כל מילה אפשרית וכל סיסמא הייתה ברשימת סיסמאות שהודלפו לרשת בין אם נפוצה או לא. האתר מצא שערך הגיבוב מתאים לסיסמא Security1.

Hash	Type	Result
00225b3d0d8dd59287998962ba4bd9e01f6e8b7314d806474de84062e2c527fd	sha256	Security1