Padding oracle attack

בהגשת תרגיל זה מצורף הסקריפט הפייתוני ex1.py הממשש Padding oracle attack כפי שנלמד בתרגול.

הסקריפט מקבל את ה cyphertext , את ה key ואת הבלוק IV.

ה cyphertext הוא בעצם הטקסט המוצפן לפי CBC, הטקסט מחולק לבלוקים וכל בלוק מוצפן בעזרת המפתח (ה key שאותו הסקריפט מקבל), ובעזרת XOR עם הבלוק הקודם. לדוגמא הבלוק ה Ci ב cyphertext הוא בעצם E(Pi ⊕ Ci−1) כך ש Pi מייצג את הבלוק בplaintext , כלומר את תוכן הבלוק המקורי, Ci-1 את הבלוק המוצפן הקודם ו E מייצג את ההצפנה (כמובן לפי המפתח). בנוסף נראה שאת הבלוק הראשון אנו נצפין כך C1 = E(P1 ⊕ IV) ,מכיוון שאין בלוק קודם , ו IV מייצג בלוק כלשהו שבעזרתו אנו מצפינים רק את הבלוק הראשון וכאמור גם אותו הסקריפט מקבל בתחילת ריצתו.

כעת אתאר את ריצתו של הסקריפט, בהינתן ה cyphertext, ה key ו IV . תחילה :

```
Total_plain_text = bytes()
total_blocks = int(len(ciphertext)/8)
# iterating over every block
for block_ind in range(1, total_blocks+1):
    # c = Xj || Ci
    c = b'\x00\x00\x00\x00\x00\x00\x00\x00' + ciphertext[-8:]
    current_block_plaintext = [None] * 8
```

Total_plain_text הוא משתנה שבסוף הריצה יכיל את ערך הPLAINTEXT המקורי בבתים כך שבתחילת הריצה הוא ריק, ובכל איטרציה נשרשר אליו את הבלוק שנחשוף.

לאחר מכן נתחיל לעבור על כל בלוק ב ciphertext בנפרד כך שאנו מתחילים מהבלוק האחרון.

ניצור משתנה c שגודלו יהיה בגודל 2 בלוקים כך שהבלוק הראשון הוא בלוק אפסים, והבלוק השני הוא הבלוק ב cyphertext עליו אנו עוברים באיטרציה הזו.

current_plain_text יהיה מערך שיכיל את הבתים שמייצגים את הטקסט המקורי בבלוק עליו אנו עובדים. בסוף האיטרציה מערך זה יכיל את ערך הבלוק המקורי בשלמותו.

```
if block_ind != total_blocks:
    # iterating over every byte
    for i in range(8):
        # finding Xj[i]
        Xj_i = find_xj(c, key, iv, i)
        # finding the true plaintext byte, P[i] = P'[x]^Ci-1[x]^Xj[x]
        Pj_i = xor(i+1,ciphertext[len(ciphertext)-9-i], Xj_i)
        current_block_plaintext[7-i] = Pj_i
        # iterating over block X from the end each time to adjust my wish, And setting c to the next round
        for j in range(i+1):
            fake_byte = xor(i+2,ciphertext[len(ciphertext)-9-j], int(current_block_plaintext[7-j].hex(), 16))
            c = c[:7-j] + fake_byte + c[8-j:]
    # when finished adding the block to the final message
    Total_plain_text = (b''.join(current_block_plaintext)) + Total_plain_text
    ciphertext = ciphertext[:-8]
```

תחילה, נבדוק האם אנו לא בבלוק הראשון, כזכור הבלוק הראשון מוצפן בעזרת הבלוק הנתון IV ואילו כל בלוק אחר מוצפן בעזרת הבלוק המוצפן שקדם לו – Ci-1 .

נניח שאנו לא בבלוק הראשון ובכך נכנס לתנאי וכעת נעבור בלולאה על כל הבתים בבלוק הנוכחי.

נזכיר ש c = Xj || Ci , כך שהרעיון יהיה למצוא את ערכי Xj כך שההצפנה תהיה חוקית ובכך לחלץ את הplaintext . וכעת נשלח לפונקציה find_xj את c על מנת למצוא Xj[7-i] כך ש c תהיה מוצפנת בצורה חוקית.

```
def find_xj(c, key, iv, i):
    for k in range(256):
        c = c[:7-i] + bytes([k]) + c[8-i:]
        if oracle(c, key, iv):
            return (k)
```

```
def oracle(ciphertext, key, iv):
    try:
        cipher = DES.new(key, DES.MODE_CBC, iv)
        plaintext = unpad(cipher.decrypt(ciphertext), DES.block_size)
        return True
    except (ValueError, TypeError):
        return False
```

נראה שהפונקציה find_xj בעצם רצה על כל האפשרויות בבית מסוים ב Xj , בהתאמה למיקום הבית אותו אנו מחפשים כעת ב cyphertext. כאמור בפעם הראשונה יהיה על הבית האחרון ובכל פעם שנקרא לפונקציה זו, עקב בניית הקוד, הוא יחפש את כל האפשרויות על בית אחד אחורה.

על כל צרוף שכזה ישאל האורקל האם ההצפנה של c חוקית. נראה שמכיוון ש Ci ב c הוא הבלוק האחרון הוא גם יצטרך להיות "כביכול" מרופד על מנת שההצפנה תהיה חוקית. ולכן באיטרציה הראשונה אנו נצפה שהאורקל יחזיר TRUE במצב שבו לאחר פענוח שלו הבית האחרון יהיה : b'x01' עקב הגדרת ההצפנה כפי שהסברתי קודם לכן.

האורקל פשוט מנסה לפענח את ה ciphertext לפי CBC, מפתח ההצפנה ו IV , במידה והצליח מחזיר True אחרת False . נציין שלאורקל בלבד יש את פונקציית הפענוח decrypt וכמובן שאם גם לנו כתוקפים היה אותה היינו יכולים ישירות לפענח את ה cyphertext .

כעת נתבונן על קטע הקוד הבא :

```
# finding the true plaintext byte, P[x] = P'[x]^Ci-1[x]^Xj[x]
  Pj_i = xor(i+1,ciphertext[len(ciphertext)-9-i], Xj_i)
```

כפי שהסברתי קודם לכן, מהגדרת ההצפנה המשוואה הזו מתקיימת :

P[7-i] = P'[7-i]^Ci-1[7-i]^Xj[7-i]

כך ש 'P בעצם מייצג את מה שאילצנו את הPLAINTEXT להיות על מנת שהאורקל יחזיר TRUE .

כאמור את P'[7-i] אנו יודעים בהתאם לאיטרציה, באיטרציה הראשונה נרצה שיהיה 1 מכיוון שאנו מסתכלים רק על הבלוק האחרון ונצפה שרק הוא ירופד ב1, באיטרציה השנייה נרצה ששני הבתים יהיו 2 וכן הלאה...

Ci-1[7-i] נתון לנו, ו Xj[7-i] זה הערך שהפונקציה find_xj החזירה לנו. וכעת, מכיוון ששלושת הערכים הללו נתונים לנו, אנו יכולים להשתמש בפונקציית הXOR שבנינו ולחלץ את הPLAINTEXT המקורי במקום ה 7-i.

בנוסף, נכניס את Pj_i למערך current_block_plaintext במקום ה 7-i וכעת נרצה להכין את c לסיבוב הבא. נשים לב למשוואה הבאה:

$$P'2[x] = Pi[x] \oplus Ci-1[x] \oplus Xj[x]$$

זו בעצם אותה משוואה מלפני כמה שורות רק שהחלפנו בין P ל 'P , חילוף זה הוא חוקי עקב חוקי XOR . נתבונן בלולאה זו :

```
for j in range(i+1):
    fake_byte = xor(i+2,ciphertext[len(ciphertext)-9-j], int(current_block_plaintext[7-j].hex(), 16))
    c = c[:7-j] + fake_byte + c[8-j:]
```

נראה שאנו מממשים את המשוואה שהצגתי כעת ומטרתנו היא בעצם להכין את c לסיבוב הבא כך שיהיה חוקי עד לבית אותו אנו מחפשים.

לדוגמא, אם אנו מנסים לפענח את הבית החמישי, נראה שההצפנה שתהיה חוקית היא ההצפנה שאם נפענח אותה עד המקום החמישי היא תהיה מרופדת ב b'x03' בכל שלושת הבתים האחרונים.

מיד לאחר מכן נחזור בלולאה לבית הבא אותו אנו רוצים לפענח, שהוא בעצם הבית הקודם, וכמובן רצף הפעולות יתבצע באופן דומה

בסיום מעבר על כל הבתים בבלוק הנוכחי, המערך current_block_plaintext יהיה מלא בכל שמונת ערכיו ויכיל את ה plaintext המקורי ואותו נוסיף ל/Total_plain_text.

בנוסף נמחק את שמונת הבתים האחרונים ב cyphertext מכיוון שכבר פענחנו אותם ובכל איטרציה c נבנית בעזרת שמונת הבתים האחרונים של ה cyphertext ונרצה שכעת הם יהיו הבלוק הבא.

לאחר מכן נחזור בלולאה לעבור על שאר הבלוקים וכמובן התהליך יתבצע באופן דומה עבור כל בלוק.

נשים לב שבאיטרציה האחרונה, שבה בעצם אנו מפענחים את הבלוק הראשון, יתבצע תהליך זהה לחלוטין רק בכל פעם שביצענו פעולת XOR עם Ci-1 נחליף אותו ב IV מהסיבות המפורטות קודם לכן.

פענוח הבלוק הראשון כך שהשינויים מפענוח בלוק שאינו הראשון מסומנים :

```
else:
    # doing the same thing but instead using Ci-1 we will use iv
    for i in range(8):
        Xj_i = find_xj(c, key, iv, i)
        Pj_i = xor(i+1,iv[7-i], Xj_i)
        current_block_plaintext[7-i] = Pj_i

        for j in range(i+1):
            fake_byte = xor(i+2,iv[7-j], int(current_block_plaintext[7-j].hex(), 16))
            c = c[:7-j] + fake_byte + c[8-j:]
    Total_plain_text = (b''.join(current_block_plaintext)) + Total_plain_text
```

ובנוסף, אין סיבה למחוק את שמונת הבתים האחרונים של ה cyphertext מכיוון שסיימנו לעבור על כולו.

שורת הקוד האחרונה תהיה :

```
print(unpad(Total_plain_text, DES.block_size).decode())
```

והיא תדפיס כמבוקש את הPLAINTEXT המקורי בתצורה טקסטואלית, ובנוסף תבצע unpad שיוריד את הריפוד המלאכותי שהתווסף לפני ההצפנה.