

מבוא לקריפטוגרפיה / תרגיל בית #2

אריאל סטורמן
ודים סטולנד

(1)

להלן תיאור אלגוריתם לפענוח $DES_{k,k_1}^V(M)$:

יהי k_j ניחוש עבור k ו- (P_i, C_i) זוגות *known plaintext-ciphertext* נתונים. עבור כל ניחוש k_j נעבור על כל הזוגות (P_i, C_i) ונבצע את התהליך הבא:

- נצפין את P_i עם המפתח k_j בהצפנת *DES* רגילה, ונקבל $DES_{k_j}(P_i)$.
 - נחשב את $DES_{k_j}(P_i) \oplus C_i$ ונקבל k'_1 שהוא מועמד עבור k_1 , שכן $DES_{k_j}(P_i) \oplus (DES_{k_j}(P_i) \oplus k'_1) = k'_1$.
 - עבור אותו k_j , אם כל ה- k'_1 (המועמדים ל- k_1) זהים, ניתן לקבוע בהסתברות גבוהה (יפורט בהמשך) כי k_j, k'_1 הם ה- k, k_1 הנכונים.
- ישנם סה"כ 2^{56} ניחושים עבור k_j ; אם נשתמש בשני זוגות (P_i, C_i) שונים, כיוון שניתן להניח כי *DES* היא פונקציה פסאודו-ראנדומית, אז הסיכוי בהינתן $k \neq k_j$ לקבל k'_1 זהה עבור שני הזוגות הוא $\frac{1}{2^{64}}$. כמובן שאם $k_j = k$ אז כל k'_1 המתקבלים אמורים להיות זהים. אם ניקח l זוגות (P_i, C_i) אז הסיכוי לטעות הוא $\frac{1}{2^{64(l-1)}}$, וזהו סיכוי קטן מאוד כבר עבור l קטן.

להלן אלגוריתם לפענוח $DES_{k,k_1}^W(M)$:

תחת אותם סימונים מקודם, עבור כל ניחוש של k_j נעבור על כל הזוגות (P_i, C_i) ונבצע את התהליך הבא:

- נפעיל את $DES_{k_j}^{-1}(C_i)$.
 - נחשב את $DES_{k_j}^{-1}(C_i) \oplus P_i$ ונקבל k'_1 שהוא מועמד עבור k_1 , שכן $DES_{k_j}^{-1}(C_i) \oplus P_i = P_i \oplus k'_1 \oplus P_i = k'_1$.
 - עבור אותו k_j , אם כל ה- k'_1 זהים, ניתן לקבוע בהסתברות גבוהה כי k_j, k'_1 הם ה- k, k_1 הנכונים.
- משיקולים דומים לאלגוריתם הקודם, ההסתברות שבהינתן $k \neq k_j$ ל- l זוגות (P_i, C_i) ההסתברות להגיע למסקנה ש- $k'_1 = k_1$ היא $\frac{1}{2^{64(l-1)}}$.

(2)

נסמן k_1 המפתח שאיתו נצפין ו- k_2 המפתח שאיתו נפענח. נחלק את כל האפשרויות עבור k_2 שהם 2^{56} לבלוקים של 2^{40} , סה"כ 2^{16} בלוקים. עבור כל

בלוק וזוג *plain-text, ciphertext* נבצע את התהליך הרגיל של *Meet in the middle*:

- עם כל המפתחות k_2 בבלוק נפענח את C_i ונשמור בזיכרון שלרשותינו.
- נעבור על כל האפשרויות עבור k_1 , סה"כ 2^{56} , נצפין את P_i ונחפש התאמה לאחד מהפענוחים השמורים.
- אם מצאנו התאמה נשמור את זוג המועמדים עבור k_1, k_2 .

הנכונות נובעת מנכונות ההתקפה הישירה של *meet in the middle*, שכן התהליך נשאר אותו דבר למעט סיבוכיות הזמן והמקום שהשתנתה. סה"כ

decryptions בתהליך: $2^{56} = 2^{40} \cdot \frac{2^{16}}{\text{#blocks}}$; סה"כ *encryptions*: $2^{72} = 2^{56} \cdot 2^{16}$. כלומר, סה"כ מספר ההצפנות שצריך לבצע בתהליך גדל, בעוד

מספר הפענוחים נשאר כמו בהתקפה המקורית. מספר זוגות ה- (P_i, C_i) הנדרשים לא שונה מההתקפה המקורית, בגלל שימור התהליך כמו בהתקפה המקורית (רק פרסנו אותו על גבי יותר פעולות).

(3)

(a) תהי A קבוצה ריקה. נבצע את התהליך הבא:

- נגדיל מחרוזת m_i באורך 128 ביטים.
- נצפין את h_0 תחת m_i כמפתח: $h_1^{m_i} := AES_m(h_0)$.
- נבדוק האם קיים j כך ש- $h_1^{m_j} \in A$ וגם $h_1^{m_i} = h_1^{m_j}$ (ו- $m_i \neq m_j$ כמובן). אם כן, אז m_i, m_j הן הודעות המתנגשות עבור H . אם לא, נוסיף את $h_1^{m_i}$ ל- A ונמשיך.

מספר האיטרציות שנדרש אליהם עד מציאת התנגשות בהסתברות גבוהה הוא כשורש מרחב ההודעות (בדומה לפרדוקס היומולדת), כלומר $\sqrt{2^{128}} = 2^{64}$.

(b) בהינתן הודעה m באורך n בלוקים, נסמן $H(m) = h_n$ כאשר h_n היא מחרוזת בגודל בלוק (128 ביטים). נמצא הודעה $m' := m_1 m_2$ באורך 2 בלוקים המקיימת $H(m') = H(m)$ באמצעות *meet-in-the-middle attack*, באופן הבא:

- נאתחל קבוצה A וקבוצה B , שתיהן ריקות.
- נגדיל m' הודעה באורך שני בלוקים עד שנקבל $m' \neq m$ כזו.
- נבדוק האם $AES_{m_1}(\vec{0}) = AES_{m_2}^{-1}(h)$, או האם $AES_{m_1}(\vec{0})$ מתאים לאיבר מ- B או האם $AES_{m_2}^{-1}(h)$ מתאים לאיבר מ- A . אם כן, סיימנו. אחרת נוסיף את $AES_{m_1}(\vec{0})$ לקבוצה A ואת $AES_{m_2}^{-1}(h)$ לקבוצה B ונמשיך.

הגרלת הודעה m' באורך שני בלוקים שקולה להגרלת שתי הודעות בנות 128 ביטים מתוך מרחב בגודל 2^{128} הודעות. בדומה לפרדוקס היומולדת, לאחר $\sqrt{2^{128}} = 2^{64}$ ניסיונות נמצא בסיכוי גבוה שתי הודעות המקיימות זאת (תחת הנחה ש- AES היא פרמוטציה פסאודו ראנדומית התלויה במפתח לפיו פועלת). הודעה m' המקיימת זאת כמובן מקיימת $h = H(m') = AES_{m_2}(AES_{m_1}(\vec{0}))$, ולכן m' מהווה התנגשות עם m עבור H .

(4)

(a) להלן אלגוריתם למציאת $(w, MAC_k(w))$ עבור $w \notin \{z_1, \dots, z_s\}$ בזמן פולי בגודל הבלוק n :

עבור z_1, z_2 מילים בגודל בלוק יחיד (n ביטים), בהינתן $(z_1, MAC_k(z_1))$ ו- $(z_2, MAC_k(z_2))$ כאשר $z_2 = MAC_k(z_1)$, הזוג $(w, MAC_k(z_2))$ הוא זוג טוב כאשר $w = z_1 \circ \vec{0}$ הוא בלוק אפסים בגודל n ביטים:

- נשים לב כי $C_1 = MAC_k(z_1) = E_k(z_1 \oplus \vec{0})$.
- כמו כן $C_2 = MAC_k(z_2) = MAC_k(C_1) = E_k(C_1 \oplus \vec{0})$.
- מתקיים: $C_2 = MAC_k(z_1 \circ \vec{0})$ כיוון שהבלוק הראשון הוא z_1 , ומופעל $E_k(z_1 \oplus \vec{0})$ שהוא ידוע - C_1 , ועל הבלוק השני, שהוא $\vec{0}$, מופעל $E_k(C_1 \oplus \vec{0})$ שהוא גם ידוע - C_2 . לכן הזוג $(w, MAC_k(z_2))$ הוא זוג חוקי ו- $w \notin \{z_1, z_2\}$.

(b) להלן אלגוריתם למציאת $(w, MAC(w))$ עבור $w \notin \{z_1, \dots, z_s\}$ בזמן פולי בגודל הבלוק n :

נסמן $MAC_k(m) = t_m$. בהינתן הזוגות:

- (z_1, t_{z_1})
- (z_2, t_{z_2})
- עבור $(z', t_{z'})$ עבור $z' = z_1 \circ 0 \dots 01 \circ z_3$ n bits

כאשר z_i מילים שרירותיות בגודל n ו- $z_1 \neq z_2$, נראה כיצד למצוא את (w, t_w) עבור $w = z_2 \circ 0 \dots 01 \circ [t_{z_1} \oplus t_{z_2} \oplus z_3]$ נפתח תחילה את $t_{z'}$:

1. $\vec{0}, z_1 \rightarrow E_k(\vec{0} \oplus z_1) = E_k(z_1)$
2. $E_k(z_1), 0 \dots 01 \rightarrow E_k(0 \dots 01 \oplus E_k(z_1))$
3. $E_k(0 \dots 01 \oplus E_k(z_1)), z_3 \rightarrow E_k(z_3 \oplus E_k(0 \dots 01 \oplus E_k(z_1)))$
4. Adding block size (3): $E_k(z_3 \oplus E_k(0 \dots 01 \oplus E_k(z_1))), 0 \dots 011 \rightarrow E_k(0 \dots 011 \oplus E_k(z_3 \oplus E_k(0 \dots 01 \oplus E_k(z_1))))$
 $= E_k(0 \dots 011 \oplus E_k(z_3 \oplus t_{z_1})) = t'$

כמו כן מתקיים כי $t_{z_2} = E_k(0 \dots 01 \oplus E_k(z_2))$ ו- $t_{z_1} = E_k(0 \dots 01 \oplus E_k(z_1))$.

כעת נפתח את t_w :

1. $\vec{0}, z_2 \rightarrow E_k(\vec{0} \oplus z_2) = E_k(z_2)$
2. $E_k(z_2), 0 \dots 01 \rightarrow E_k(0 \dots 01 \oplus E_k(z_2)) = t_{z_2}$
3. $t_{z_2}, [t_{z_1} \oplus t_{z_2} \oplus z_3] \rightarrow E_k(t_{z_1} \oplus t_{z_2} \oplus z_3 \oplus t_{z_2}) = E_k(t_{z_1} \oplus z_3)$
4. Adding block size (3): $E_k(t_{z_1} \oplus z_3), 0 \dots 011 \rightarrow E_k(0 \dots 011 \oplus E_k(t_{z_1} \oplus z_3))$

כאשר הביטוי האחרון שקיבלנו הוא t' , כלומר כבר ברשותנו מהנתונים. כיוון ש- $z_1 \neq z_2$ או $z_1 \in \{z_1, z_2, z'\}$, $w \notin \{z_1, z_2, z'\}$ וברשותינו $(w, MAC_k(w))$, כנדרש.
 (c) להלן אלגוריתם למציאת $(w, MAC(w))$ עבור $w \in \{z_1, \dots, z_s\}$ בזמן פוליני בגודל הבלוק n :

נסמן $t_m = MAC_{k,k'}(m)$. יהיו z_1, z_2 שתי מילים שרירותיות באורך n (גודל בלוק יחיד) שלא שתיהן $\vec{0}$. בהינתן זוגות המילים וה- MAC שלהן למילים:

$$\begin{aligned} & z_1 \circ z_1 \quad \bullet \\ & z_2 \quad \bullet \\ & z_1 \circ z_1 \circ [t_{z_2} \oplus t_{z_1 \circ z_1}] \quad \bullet \end{aligned}$$

נראה כיצד למצוא את (w, t_w) עבור $w = z_2 \circ \vec{0}_{n \text{ bits}}$. נפתח תחילה את ה- MAC של המילים הנתונות:

עבור $z_1 \circ z_1$:

1. $\vec{0}, z_1 \rightarrow E_k(\vec{0} \oplus z_1) = E_k(z_1)$
2. $E_k(z_1), z_1 \rightarrow E_k(z_1 \oplus E_k(z_1))$
3. Adding k' : $E_k(z_1 \oplus E_k(z_1)) \oplus k'$

עבור z_2 :

1. $\vec{0}, z_2 \rightarrow E_k(\vec{0} \oplus z_2) = E_k(z_2)$
2. Adding k' : $E_k(z_2) \oplus k'$

עבור $z_1 \circ z_1 \circ [t_{z_2} \oplus t_{z_1 \circ z_1}]$:

1. $\vec{0}, z_1 \rightarrow E_k(\vec{0} \oplus z_1) = E_k(z_1)$
2. $E_k(z_1), z_1 \rightarrow E_k(z_1 \oplus E_k(z_1))$
3. $E_k(z_1 \oplus E_k(z_1)), [t_{z_2} \oplus t_{z_1 \circ z_1}] \rightarrow E_k([t_{z_2} \oplus t_{z_1 \circ z_1}] \oplus E_k(z_1 \oplus E_k(z_1))) = E_k(E_k(z_2) \oplus k' \oplus E_k(z_1 \oplus E_k(z_1)) \oplus k' \oplus E_k(z_1 \oplus E_k(z_1))) = E_k(E_k(z_2) \oplus k' \oplus k') = E_k(E_k(z_2))$
4. Adding k' : $E_k(E_k(z_2)) \oplus k'$

נראה כי עבור $w = z_2 \circ \vec{0}$ מתקיים כי t_k גם כן שווה ל- $E_k(E_k(z_2)) \oplus k'$:

1. $\vec{0}, z_2 \rightarrow E_k(\vec{0} \oplus z_2) = E_k(z_2)$
2. $E_k(z_2), \vec{0} \rightarrow E_k(\vec{0} \oplus E_k(z_2)) = E_k(E_k(z_2))$
3. Adding k' : $E_k(E_k(z_2)) \oplus k'$

כיוון ש- z_1, z_2 אינן שתיהן $\vec{0}$ או $\{z_1 \circ z_1, z_2, z_1 \circ z_1 \circ [t_{z_2} \oplus t_{z_1 \circ z_1}]\}$ $w \notin \{z_1 \circ z_1, z_2, z_1 \circ z_1 \circ [t_{z_2} \oplus t_{z_1 \circ z_1}]\}$ וברשותינו (w, t_w) , כנדרש.

(6)

(a) לכל $p > 2$ ראשוני מתקיים כי p הוא אי זוגי ולפיכך לכל $k > 1$ גם p^k אי זוגי. מכאן ש- $p^k - 1 = 2r$ כלומר $r = \frac{p^k - 1}{2}$.

לכל $x \in GF^*(p^k)$ מתקיים $x^{p^k - 1} \equiv 1$ ב- $GF^*(p^k)$ ולכן: $x^{p^k - 1} = x^{2r} = (x^2)^r \equiv 1$, וכיוון שסדר כל איבר מחלק את סדר החבורה, הסדר של

(x^2) הוא לכל היותר $r = \frac{p^k - 1}{2} < p^k$ בתחום המבוקש. לפיכך $(x^2) \in GF^*(p^k)$ אינו פרימיטיבי.

(b) אם נסתכל על ייצוג $GF^*(p^k)$ מעל שדה הפולינומים, אז כל פולינום מדרגה 0 שישומו $f(x) = i \cdot x^0$ עבור $2 \leq i \leq p - 1$ הוא איבר לא

פרימיטיבי, כיוון שמתקיים: $(a \cdot x^0)^{p-1} = a^{p-1} = 1 \pmod{p}$ ב- \mathbb{Z}_p^* כיוון שאיברי $GF^*(p^k)$ הם פולינומים מעל $GF(p)$, העלאת $p - 2$

הפולינומים הנ"ל ב- $p - 1$ תתן את הפולינום הקבוע 1. לפיכך סדר אותם איברים הוא $p - 1 < p^k - 1$ ולכן איברים אלו אינם פרימיטיביים.

(7)

(a)

```
p = 249*(2^249)-1
is_prime(p)
> True
factor(p-1)
> 2 * 708211533214392631 * 49321227993155495937119743 * 3224344192787689425940157396942471
```

(b)

```
def is_primitive_root(p, g):
    if is_prime(p) == False:
        print "p is not prime"
    else:
        f = factor(p-1)
        for i in range(len(f)):
            exp = (int)((p-1)/f[i][0])
            pow_mod = power_mod(g,exp,p)
            if pow_mod == 1:
                return False
        return True
```

```
# print is_primitive_root(p,g)
# mod(g,p).multiplicative_order()
```

הפונקציה נבדקה ע"י שתי השורות האחרונות בקוד לעיל עבור p, g שונים. כאשר פלט הפונקציה היה $True$, הסדר שהוחזר היה $p - 1$, וכאשר פלט הפונקציה היה $False$, הסדר שהוחזר היה קטן ממנו.

(c)

```
p = 249*(2^249)-1
g = randint(10^7+1,10^8)
print(g)
f = factor(p-1)
```

```
def is_primitive_root(p, g, f):          # adding factor(p-1) to the input
    if is_prime(p) == False:
        print "p is not prime"
    else:
        for i in range(len(f)):
            exp = (int)((p-1)/f[i][0])
            pow_mod = power_mod(g,exp,p)
            if pow_mod == 1:
                return False
        return True
```

```
is_primitive_root(p,g,f)
is_primitive_root(p,g+1,f)
print p
print mod(g,p).multiplicative_order()
print mod(g+1,p).multiplicative_order()
> 34845892          # = g
> True            # g is a primitive element of Z*p
> False          # g+1 is not a primitive element of Z*p
> 225251798594466661409915431774713195745814267044878909733007331390393510002687
> 225251798594466661409915431774713195745814267044878909733007331390393510002686 # = p-1, as expected
> 112625899297233330704957715887356597872907133522439454866503665695196755001343 # != p-1, as expected
```

```

q1 = next_prime(2^400)
p1 = 2*q1 + 1
while is_prime(p1) == False:
    q1 = next_prime(q1)
    p1 = 2*q1 + 1
print q1
print p1
>
2582249878086908589655919172003011874329705792829223512830659356540647622016841194629645353280137831435903171972
747524733
>
5164499756173817179311838344006023748659411585658447025661318713081295244033682389259290706560275662871806343945
495049467

q2 = next_prime(q1)
p2 = 2*q2 + 1
while is_prime(p2) == False:
    q2 = next_prime(q2)
    p2 = 2*q2 + 1
print q2
print p2
>
2582249878086908589655919172003011874329705792829223512830659356540647622016841194629645353280137831435903171972
747631671
>
5164499756173817179311838344006023748659411585658447025661318713081295244033682389259290706560275662871806343945
495263343

```

(8)

(a) להלן הוכחה כי עבור $f_0(x) := g^x \bmod p, f_1(y) = a \cdot g^y \bmod p$, כאשר $a, g, x, y \in \mathbb{Z}_p^*$, יוצר של \mathbb{Z}_p^* ראשוני, מתקיים כי f_0, f_1 הן *claw-free permutations*.

ידוע כי קשה למצוא z כך ש- $a = g^z$. נניח בשלילה כי f_0, f_1 אינן *claw-free*, אזי קל למצוא $x, y \in \mathbb{Z}_p^*$ כך ש- $g^x \equiv a \cdot g^y \pmod{p}$. כיוון ש- $g^x \equiv a \cdot g^y \pmod{p}$ נכפיל את האגפים ב- $g^{-(p-1)-x}$ ונקבל:
 $g^x \equiv g^{y+z} \pmod{p}$ ולכן $g^x \equiv g^z \cdot g^y \pmod{p}$. אזי $a = g^z \bmod p$ כך ש- $a = g^z \bmod p$. כיוון ש- $1 \equiv g^{(p-1)-x+y+z} \pmod{p}$ נניח בשלילה כי $a = g^z \bmod p$ אז אפשר למצוא z ע"י פתרון המשוואה $g^x \equiv a \cdot g^y \pmod{p}$ כלומר $(p-1) - x + y + z = p - 1$.
 $z = x - y$. אם מספר זה יוצא שלילי, ניתן להוסיף לו $p - 1$ כיוון ש- $g^l \equiv g^{l \pmod{p-1}}$.
 $g^{p-1+l} = g^{p-1} \cdot g^l \equiv g^l \pmod{p}$ כיוון ש- $g^l \equiv g^l \pmod{p}$ (מספיק להוסיף כפולה אחת של $p - 1$ אם יוצא z שלילי, כי $x, y \in \mathbb{Z}_p^*$ ולכן $\min\{z\} = 1 - (p - 1) = 2 - p$ ואז $2 - p + (p - 1) = 1$ שכבר אינו שלילי).

כיוון שקשה לחשב את z כך ש- $a = g^z$ הן f_0, f_1 הן *claw-free permutations*.

(b) יהיו f_0, f_1 *claw-free permutations* וידוע כי קשה למצוא z כך ש- $f_0(z) = IV \in D$ או $f_1(z) = IV \in D$. נניח בשלילה כי H אינה עמידה בפני התנגשויות, אזי נוכל למצוא בקלות $m_1 \neq m_2$ כך ש- $H(m_1) = H(m_2)$. נראה כי ניתן בקלות למצוא זוג $x, y \in D$ כך ש- $f_0(x) = f_1(y)$ (כלומר f_0, f_1 אינן *claw-free*) או $z \in D$ כך ש- $f_0(z) = IV \in D$ או $f_1(z) = IV \in D$. בסתירה לנתונים.

יהיו m_1, m_2 כך ש- $m_1 \neq m_2$ ו- $m_1 = b_1 b_2 \dots b_n$ ו- $m_2 = c_1 c_2 \dots c_k$ ובה"כ $k \leq n$. עבור $1 \leq i \leq k$ נבצע את התהליך הבא:

- כל עוד $b_i = c_i$ נחשב את $H(b_{i+1} \dots b_n)$ ואת $H(c_{i+1} \dots c_k)$, שהן כמובן שוות.
- אם הגענו ל- i כך ש- $b_i \neq c_i$ בה"כ $b_i = 0, c_i = 1$, אזי נחשב את $x := H(b_{i+1} \dots b_n)$ ו- $y := H(c_{i+1} \dots c_k)$ ולמעשה מתקיים $f_0(x) = f_1(y)$ בסתירה ל-*claw-freeness* של f_0, f_1 .
- אם לא נתקלנו ב- i עבורו $b_i \neq c_i$ אז למעשה "הפשטנו" את m_2 עד למחרוזת $IV \in D$, ואז מתקיים ש- $H(b_{k+1} \dots b_n) = IV$, כלומר מצאנו $f_{k+1}(H(b_{k+2} \dots b_n)) = IV$ וזה בסתירה לקושי במציאת z כזה.

(5) חלקי בלבד

 (a) הוכחה כי $\gcd(a, m) = 1 \Leftrightarrow \text{ord}_m(a) < \infty$:יהי $m = p_1^{k_1} \cdot \dots \cdot p_l^{k_l}$. נניח תחילה כי $\gcd(a, m) = 1$ אז קיימים x, y כך ש- $ax + my = 1$ ו- x הוא ההופכי הכפלי של a ב- \mathbb{Z}_m^* , כלומר $a \in \mathbb{Z}_m^*$.ידוע כי לכל $x \in \mathbb{Z}_m^*$ מתקיים $\text{ord}_m(x) \mid \text{ord}(\mathbb{Z}_m^*) = \phi(m) = \prod_{i=1}^l (p_i^{k_i} - p_i^{k_i-1}) \in \mathbb{N}$ כאשר $\text{ord}_m(a) < \infty$ מכאן ש-נניח כעת כי $\text{ord}_m(a) = k \in \mathbb{N}$ עבור k כלשהו.

...

- (b)