יסודות הקריפטוגרפיה שיעור #3

Block Ciphers

<u>פונקציה חד כיוונית</u>: קל להצפין אך קשה לפענח אם לא יודעים את המפתח. כלומר, $x \to f(x)$ קל, אך $f(x) \to x$ קשה.

הצפנת בלוק קלט לבלוק פלט, כאשר גודלם לרוב זהה (לכל הפחות; ה-*ciphertext* לא יהיה קצר מה-*plaintext*).

השיטה הפשוטה ביותר לשימוש בהצפנת בלוקים:

<u>Modes of operation</u>:

ECD: *electronic code*: $P_i \to E_k \to C_i$ - כל בלוק עובר דרך פונקצית ההצפנה עם המפתח k. הבעיה בשיטה זו היא שחזרות בלוקים, למשל הודעות במלחמה, יכולות לתת מידע ליריב. בלוקים של *plaintext* זהים עוברים לבלוקי *ciphertext* זהים, ויתכן שהיריב הבין את ה-*plaintext* הראשון ומכאן יוכל להקיש על הופעת אותו *plaintext* בעתיד. <u>הערה</u>: זו לא חולשה של E_k אלא של הצורה בה משתמשים ב-E_k.

<u>הערה</u>: E_k טובה מערבלת היטב הודעות, אפילו אם C_1, C_2 נבדלים בביט בודד, P_1, P_2 רחוקים זה מזה.

שתי הגישות האפשריות הן להוסיף *padding* ל-*plaintext* כדי להוסיף אקראיות לטקסט, או להוסיף *states*.

CBC mode: *cipher block chaining*: s_0 מילה קבועה; משתמשים בה להצפין את P_1 עם *xor* לפני הכנסה לפונקצית ההצפנה. לאחר מכן משתמשים בפלט (C_i) לאותו מנגנון להודעות הבאות. מערכת זו היא אסינכרונית, כלומר ברגע שיודעים *ciphertext* של בלוק מסוים ניתן לפענח את השאר (בהינתן מפתח כמובן).

OFD: *output feedback mode*: מתחילים מ-s_0, מצפינים אותו באמצעות E_k ל-s_1 וכן הלאה, וכל הודעה m_i מוצפנת ל-c_i ע"י *xor* עם s_i:

שיטה זו לא סינכרונית, שיטה זו עמידה ל-*chosen plaintext attack*, כמו ה-*CBC*.

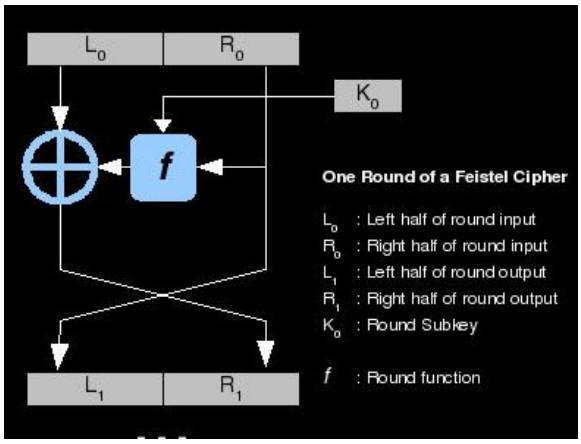
עקרונות תכנון ל-Block ciphers:

נתאר את E_k כמכיל f_{k_i} פונקציות כאשר k_i נגזרים מהמפתח k. כל f_{k_i} "מסתירה קצת", וכשמשרשרים אותן מקבלים משהו קשה – E_k. אם k נתון אז חישוב E_k (הצפנה) ו-E_k^{-1} (פענוח) הוא קל (מהיר). אם k לא נתון אז E_k אמורה להיראות כמו תמורה אקראית על מרחב ההודעות. $P \to E_k \to C$:

1. מותר לצפות בזוגות P_i, C_i.
2. מותר לבחור P_i ולבקש C_i (*chosen plaintext attack*).

כל זה במספר סביר. גם אחרי זאת, לא ניתן לקבל אינפורמציה על k ואפילו (לגבי זוג שלא ראינו P_{new}, c_{new}) לדעת האם הזוג עומד ביחס $P_{new} \to E_k$. $E_k \to C_{new}$ היא תמורה פסאודו-אקראית; היא פונקציית הצפנה, דהיינו חח"ע (לכן תמורה); פסאודו אקראית – אם ניקח 2^{128} מפתחות, נקבל את הפונקציה (עבור k מפתח בן 128 ביטים). פירוט עקרונות במצגת, עמוד 9.

DES: Data Encryption Standard:



שלב אחד לבד כמובן לא מוצלח כי R_i עובר ל-L_{i+1} איך שהוא. רוצים חח"ע; f יכולה להיות קבוע, ו-k_i (חלק מהמפתח) ושתנה משלב לשלב.

ניתן להשתמש באותו מנגנון גם להצפנה וגם לפענוח. אם f היא פונקציה פסאודו אקראית אז 4 הרכבות *Feistel* שואפת לתמורה פסאודו אקראית (הוכח).

שחזור:

- R_i יהיה פשוט L_{i+1}
- L_i יהיה $f(k_i, L_{i+1}) \oplus R_{i+1}$

(לכן הפונ' היא חח"ע).

***AES*: *Advanced Encryption Standard*:**

- *Symmetric block cipher*.
- אורך מפתח: 128, 192 ו-256 ביטים.
- עמיד בפני כל ההתקפות הידועות.

בכל שלב נכנס בלוק (128 ביטים), יחד עם תת מפתח (שהוא גם 128 ביטים) והפלט נכנס שוב עם עוד תת מפתח וכן הלאה כמה פעמים עד הפלט.

גודל בלוק קלט ופלט: 128 ביטים; מצב הוא 128 ביטים מסודרים במטריצה 4 על 4 (של בתים), כל בית נראה כאיבר של $GF(2^8)$. ב-128 ביטים משתמשים ב-10 סיבובים (לא סיבובי *Feistel* כמו ב-*DES*). מה קורה בכל סיבוב:

1. *Substitution*: יהיו איברי המטריצה S_{ij}, מחליפים אותו ב-S_{ij}^{-1}, ההופכי הכפלי שלו בשדה. 0 נשאר 0. פעולה זו חח"ע ולא לינארית.
2. הזזת שורות: בשורה ה-i עושים הזזה ציקלית ב-i מקומות.
3. ערבוב עמודות: פעולות אריתמטיות כלשהן.
4. *XOR round key*: לוקחים את התת מפתח לאותו סיבוב ועושים לו*XOR* עם ה-*state* כדי לקבל *state* חדש.

אלגוריתם *GCD* של אאוקליד:

סיבוכיות (מס' פעולות mod שדורש):

נניח נתונים r_0, r_1 טבעיים, $r_0 > r_1 \geq 0$. האלגוריתם: מגדירים סדרת פעולות $r_{i+2} = r_i \ mod \ r_{i+1} \ldots r_2 = r_0 \ mod \ r_1$. עוצרים כש- $r_i = 0$ ואז $gcd = r_{i-1}$. כדי לנתח מספר פעולות, נניח $r_0 < 2^n$ (כלומר n ביטים לכל היותר).

ידוע: $r_2 = r_0 \ mod \ r_1$, לכן קיים $c \geq 1$ טבעי כך ש- $r_0 = c \cdot r_1 + r_2$, $r_2 < r_1$. לכן $r_0 = c \cdot r_1 + r_2 \geq r_1 + r_2 \geq 2r_2$, ומכאן $r_2 \leq \frac{r_0}{2}$, $r_3 \leq \frac{r_1}{2}$, $r_4 \leq \frac{r_0}{4}$, וכן הלאה – קטן בפקטור 2 כל 2 סיבובים. לכן מספר הסיבובים הוא לכל היותר$2n$.

אלגוריתם אאוקליד המוכלל:

טענו כי אם r_o ו-r_1 טבעיים כך ש-$\gcd(r_0, r_1) = g$ אז יש x, y שלמים כך ש-$r_0 x + r_1 y = g$. נוכיח באינדוקציה על שלבי האלגוריתם של אאוקליד כי בכל שלב יש שלמים A_i, B_i כך ש-$r_i = A_i r_0 + B_i r_1$.

עבור $i = 0$: $A_0 = 1, B_0 = 0$.

עבור $i = 1$: $A_1 = 0, B_1 = 1$.

נעשה עבור $i = 2$ (ההמשך זהה ונפסח עליו):

$$r_0 = c \cdot r_1 + r_2$$

$$r_2 = r_0 - c \cdot r_1 = A_0 r_0 - c \cdot r_1 \Rightarrow A_2 = A_0, B_2 = -c$$

תזכורת: הענין שלנו באאוקליד המורחב נובע מהופכיים כפליים מודולוm. נניח כי a ו-m זרים, אז יש x ו-y כך ש-$ax + my = 1$ (כאשר 1 הוא ה-gcd). ולכן x הוא ההופכי הכפלי של a מודולו m. חישוב הופכי כפלי מתבצע דרך חישוב באוקליד מורחב.

<u>שאלה</u>: נניח m טבעי. כמה מספרים טבעיים בתחום $[1, \ldots, m]$ זרים להם?

<u>תשובה</u>: פונקציה זו מסומנת ע"י $\varphi(m)$ ונקראת פונקציית *totient* של אויילר. למשל: $\varphi(3) = 2, \varphi(7) = 6, \varphi(21) = 12$. למשל על 21: כל המספרים שהם לא 3 או כפולותיו או 7 או כפולותיו. מתברר כי אם l, k זרים אז $\varphi(l \cdot k) = \varphi(l) \cdot \varphi(k)$. ההוכחה קומבינטורית ולא תיעשה. אם p ראשוני אז $\varphi(p) = p - 1$; אם $m = p^l$, אז מספר הלא זרים הוא p^{l-1} (על כל p יש אחד זר), ולכן מספר הזרים הוא $\varphi(p^l) = p^l - p^{l-1} = p^l \left(1 - \frac{1}{p}\right)$.

נניח $m = p_1^{\alpha} \cdot \ldots \cdot p_k^{\alpha_k}$, אז $\alpha_i \geq 1$. בהינתן הפירוק, קל לחשב את $\varphi(m) = \prod_{i=1}^{k} \left(p_i^{\alpha_i} - p_i^{\alpha_i - 1}\right)$. אם הפירוק לא נתון, מתברר שחישוב $\varphi(m)$ קשה כמו פירוק m (לכן קשה חישובית). נהיה מעוניינים ב-$\varphi(m)$ כי זה מספר האיברים בחוג הכפלי $\mathbb{Z}_m^*$ - אוסף כל האיברים שיש להם הופכי כפלי מודולו m. סדר חבורה זו הוא $\varphi(m)$. למשל ב-$\mathbb{Z}_{21}^*$ יש 12 איברים, 8 אחד מהם, ולכן $8^{12} = 1 \ mod \ 21$ (12 היא לא חזקה מינימלית בהכרח, אך מקיימת זאת).