

אבטחת מידע וסייבר – סיכום הרצאות 8-9

מסמך מסכם וברור ללמידה. צמצום בוצע רק במקומות שאינם פוגעים בהבנת החומר (איחוד חזרות והורדת פירוט מיותר/דוגמאות שוליות).
מגבלת אורך: הותאם כך שיישאר לכל היותר 70 עמודים (באמצעות עימוד צפוף ושמירה על תוכן מרכזי).

הרצאה 8 - אבטחת מידע וסייבר

הרצאה מס' 8

- מרצה: יניב מורדוב

נושאים כלליים

- סטגנוגרפיה Steganography
- הגדרה, טכניקות ואיתור
- הסוואה בקבצי מדיה
- Chaffing and Winnowing
- Null Cipher
- ניצול יתרות קובצי הרצה וספריות
- הסוואה בציוד היקפי
- סיכון מידע
- האפלט מידע

מהי סטגנוגרפיה?

- הגדרה:
- סטגנוגרפיה Steganography היא תחום באבטחת מידע שבו מידע מוסתר בתוך מדיה אחרת, כמו תמונות, קבצי אודיו, וידאו או טקסט. המטרה היא להסתיר את עצם קיומו של המידע כדי שמי שיתבונן במדיה לא יחשוד שמשהו מוסתר בתוכה.

הבדל בין סטגנוגרפיה לקריפטוגרפיה

- קריפטוגרפיה: מתמקדת בהצפנה של המידע כך שרק מי שיש לו מפתח מתאים יכול לקרוא אותו, אך המידע גלוי לעין (כמו טקסט מוצפן).
- סטגנוגרפיה: מסתירה את עצם קיומו של המידע כך שאף אחד לא יידע שהוא שם.

טכניקות עיקריות בסטגנוגרפיה

1. Least Significant Bit (LSB): מידע מוסתר על ידי שינוי הביטים הפחות משמעותיים (למשל, בתמונה). לדוגמה, בתמונה צבעונית, הערכים של רמות האדום, הירוק והכחול יכולים להשתנות קלות מבלי שהשינוי ייראה לעין.
2. סטגנוגרפיה בטקסט: שימוש ברווחים, שגיאות כתיב מכוונות, או שינויים זעירים בפונטים ובפורמט כדי להסתיר מידע.
3. סטגנוגרפיה בקבצי קול: החבאת מידע בתדרים שלא נשמעים לאוזן אנושית או בשינויים זעירים בגלי הקול.
4. סטגנוגרפיה בקבצי וידאו: המידע מוחבא במסגרת אחת או יותר מתוך רצף הווידאו, בדרך כלל באזורים שאינם קריטיים.
5. טכניקות מתקדמות: שימוש באלגוריתמים מתמטיים כמו Wavelet Transform או בהצפנה כפולה כדי להקשות על זיהוי.

הגדרה מורחבת של סטגנוגרפיה

- מטרות עיקריות של סטגנוגרפיה:
- 1. שמירה על דיסקרטיות: היכולת להחביא מידע רגיש כך שהוא לא יתגלה למתבונן לא מיומן.
- 2. שימוש בהקשר טבעי: המידע המוסתר משתלב בתוך המדיה כך שהיא נראית לגיטימית.
- 3. התגברות על סינון: סטגנוגרפיה מאפשרת לעקוף אמצעי אבטחה או צנזורה שמזהים תוכן בעייתי.
- שימושים נפוצים:

- 1. אבטחת מידע: העברת נתונים רגישים בערוצים לא בטוחים.
- 2. ריגול ותעשייה: העברת מסרים סודיים.
- 3. סימון דיגיטלי: Watermarking לזיהוי זכויות יוצרים.
- 4. תקשורת מחתרית: שימוש על ידי ארגונים כדי לתקשר ללא חשש מגילוי.

טכניקות מתקדמות בסטגנוגרפיה

- 1. סטגנוגרפיה מבוססת תכונות סטטיסטיות:
 - מנצלת תכונות סטטיסטיות של המדיה, כמו היסטוגרמות או פרמטרים סטטיסטיים (ממוצע, סטיית תקן) לצורך החבאת המידע.
 - דוגמה: שינוי קטן בערכים של קבוצת פיקסלים כדי לייצר קוד.
- 2. סטגנוגרפיה דינמית Adaptive Steganography
 - שיטה זו מתאימה את החבאת המידע לאזורים בעלי פחות בולטות במדיה.
 - למשל, באזורים בתמונה שיש בהם שינויים מהירים בצבע, קל יותר להחביא מידע מבלי שהוא יתגלה.
- 3. סטגנוגרפיה מבוססת טרנספורמציות:
 - מסתירה מידע במרחב התדרים במקום במרחב הישיר Spatial Domain
 - משתמשת בטרנספורמציות מתמטיות כמו:
 - Discrete Cosine Transform (DCT): שימוש במדיה דחוסה JPEG
 - Discrete Wavelet Transform (DWT): שיטה המאפשרת לשנות את הנתונים באופן מקומי ולאורך כל הקובץ.
- 4. חביאה באמצעות קומפרסיה:
 - מסתירים מידע בתוך מבנה הדחיסה של קבצים, לדוגמה:
 - במידע redundancy בקבצי MP3
 - בחללים שבין פריימים של וידאו.
- 5. סטגנוגרפיה מתקדמת עם למידה עמוקה:
 - שימוש ב-AI ובמודלים של למידת מכונה כדי לנתח את המדיה ולמצוא אזורים אופטימליים להחבאת מידע.
 - ניתן גם לזהות סטגנוגרפיה קיימת באמצעות רשתות עצביות שמחפשות תבניות בלתי טבעיות.

שיטות מתקדמות לאיתור סטגנוגרפיה

- 1. איתור לפי מרחב ישיר Spatial Domain Analysis
 - זיהוי שינויים זעירים בפיקסלים במדיה.
 - כלים נפוצים:
 - היסטוגרמות פיקסלים: לבדוק אם יש תבנית לא טבעית.
 - זיהוי שינויים בביטים הפחות משמעותיים LSB
- 2. איתור במרחב התדרים Frequency Domain Analysis
 - מדיה כמו קבצי JPEG משתמשים בדחיסה מבוססת תדרים, מה שמאפשר לאתר שינויים חריגים בתדר.
 - כלי עיקרי: בדיקת coefficients ב-DCT.
- 3. סטגנליזה מבוססת סטטיסטיקה:
 - ניתוח הבדלים סטטיסטיים בין מדיה רגילה למדיה שיש בה מידע מוסתר.
 - דוגמה: RS Analysis מזהה שינויים בתבניות חזרתיות של פיקסלים.
- 4. סטגנליזה בעזרת למידה עמוקה:
 - מודלים של AI משמשים לזיהוי תבניות שאינן נראות לעין אנושית.
 - לדוגמה, רשתות CNN שמנתחות את המדיה ומספקות הסתברות להימצאות סטגנוגרפיה.
- 5. שימוש בכלי סטגנליזה:
 - StegDetect: כלי לזיהוי סטגנוגרפיה בתמונות JPEG.
 - StegSecret: כלי לבדיקת קבצי אודיו ווידאו.
 - Forensic Tools: כלים לזיהוי ותיעוד שינויים דיגיטליים.

דגשים חשובים

- 1. איזון בין אבטחה לנראות: מידע מוסתר בצורה בולטת מדי עלול להתגלות בקלות.
- 2. שיקולי יעילות: ככל שהמדיה גדולה יותר, ניתן להחביא בה יותר מידע.
- 3. התמודדות עם דחיסה: דחיסה של קובץ עלולה להרוס את המידע המוסתר.

דוגמאות מעשיות

- דוגמה 1: שימוש ב-LSB בתמונה:
 - התהליך:
 - 1. קח תמונה בפורמט BMP שמאחסנת מידע גולמי.
 - 2. עבור על הביטים של כל פיקסל, ושנה את הביט הכי פחות משמעותי כך שיתאים לביט מתוך המידע שברצונך להסתיר.
 - 3. התוצאה היא תמונה כמעט זהה, אך המידע שלך מוסתר בתוכה.
- דוגמה:
 - אם לפיקסל בתמונה יש ערך RGB של (64, 128, 255):
 - בבינארי: 01000000 10000000 11111111
 - אם רוצים להסתיר את הביט 1 נחליף את הביט הכי פחות משמעותי:
 - חדש: 01000000 10000001 11111111

איתור סטגנוגרפיה

- איתור מידע מוסתר נעשה באמצעות כלים וטכניקות של סטגנליזה Steganalysis.
- 1. השוואת קבצים: השוואה בין הקובץ המקורי לבין הקובץ החשוד.
- 2. ניתוח סטטיסטי: זיהוי תבניות או שינויים חריגים בתמונה, בקול או בוידאו.
- 3. כלים אוטומטיים:
- StegExpose: כלי לזיהוי סטגנוגרפיה בתמונות.
- DeepSound: כלי לזיהוי וליצירת סטגנוגרפיה בקול.

מהי הסוואה בקבצי מדיה?

- הגדרה:
- הסוואה בקבצי מדיה היא תהליך שבו מוסתרים נתונים בתוך קבצים רגילים כמו תמונות, קבצי קול, או סרטוני וידאו. המטרה היא להעביר מידע סודי בלי לעורר חשד, תוך שמירה על מראה תקין של קובץ המדיה.

העקרונות המרכזיים

- 1. ביטוי מינימלי: ההסוואה משנה את קובץ המדיה באופן שמבחינה ויזואלית, קולית או שימושית לא יהיה הבדל מורגש.
- 2. התאמה לפורמט: טכניקות ההסוואה משתנות בהתאם לסוג הקובץ (תמונה, אודיו, וידאו).
- 3. עמידות לשינויים: לעיתים יש צורך שהמידע המוסתר ישרוד עיבודים, כמו דחיסה או עריכה בסיסית.

סוגי מדיה וטכניקות להסוואה

- א. הסוואה בתמונות:
 - טכניקות נפוצות:
 - 1. LSB (Least Significant Bit): שינוי הביטים הכי פחות משמעותיים של ערכי הצבע RGB של פיקסלים בתמונה.
 - לדוגמה: אם הביט האחרון של ערך 11001000 ישתנה מ-0 ל-1 ההבדל בצבע יהיה זניח.
 - 2. דחיסת DCT (Discrete Cosine Transform): בשימוש בתמונות JPEG המידע מוסתר בשינויים בתדרים (במקום ישירות בפיקסלים).
 - השיטה פועלת על מקדמי התדרים של התמונה, כך שהשינויים לא נראים לעין.

- 3. Watermarking: טכניקה שמשתמשת בתמונה כולה לשם סימון או הסוואה, לרוב בזכויות יוצרים.
- ב. הסוואה בקבצי אודיו:
- 1. LSB באודיו: כמו בתמונות, שינוי הביטים הפחות משמעותיים של ערכי האודיו.
- לדוגמה: בגלי קול בפורמט WAV
- 2. Echo Hiding: מוסיפים הד חוזר קל שלא מורגש באוזן אנושית, אך הוא מקודד מידע.
- 3. Phase Coding: משנה את הפאזה של גלי הקול כדי להחביא מידע.
- ג. הסוואה בקבצי וידאו:
- 1. שינוי צבעים במסגרות מסוימות: שילוב מידע בביטים של פיקסלים במסגרות ספציפיות.
- 2. שימוש ברווחים בין פריימים: מוסיפים מידע לשדות או מרווחים שאינם נראים בעין אנושית.
- 3. טרנספורמציות בתדרי פריימים: שינוי נתונים בתדרים הגבוהים של הפריימים.

עקרונות יסוד

- מה הופך קובץ מדיה למתאים להסוואה?
- נפח מידע גדול: קבצי תמונות, אודיו ווידאו מכילים כמויות עצומות של נתונים, מה שמקל על הסוואת מידע ללא פגיעה משמעותית באיכות.
- רגישות אנושית נמוכה: חוש הראייה והשמיעה האנושיים לא תמיד מבחינים בשינויים קטנים.
- חוסר אחידות בנתונים: פורמטים כמו WAV ו-BMP הם "גולמיים", כלומר אין בהם דחיסה שמוחקת מידע מיותר. זה מקל על שילוב מידע מוסתר.

סוגי טכניקות להסוואה מעבר ל-LSB

- טכניקות ישירות Spatial Domain:
- 1. Pixel Value Manipulation (PVM): שינוי ערכי פיקסלים ישירות.
- דוגמה: לשנות את רמת הבהירות של פיקסלים מסוימים בהתאם למידע המוסתר.
- 2. Bit-Plane Slicing: חלוקה של פיקסל לרמות שונות של ביטים.
- מידע מוסתר מוחדר לרמות הנמוכות ביותר LSB
- טכניקות עקיפות Transform Domain:
- 1. Discrete Cosine Transform (DCT): פירוק התמונה למרכיבי תדרים. המידע מוסתר בתדרים הגבוהים שאינם בולטים לעין. הדבר משמש במיוחד בתמונות דחוסות כמו JPEG
- 2. Discrete Wavelet Transform (DWT): טכניקה מתקדמת יותר שמפרקת את התמונה ל"תת-רמות" sub-bands, המידע מוסתר באזורים בעלי תדרים בינוניים, שהם פחות רגישים לשינויים.
- 3. Spread Spectrum: המידע המוסתר מפוזר על פני תדרים רבים, כך שהשפעתו על הקובץ קטנה אך עמידותו לדחיסה גבוהה.

מאפייני קבצים שונים וההתאמה שלהם להסוואה

- תמונות:
- פורמט BMP: אידיאלי להסוואה, כי הוא שומר על כל הנתונים ללא דחיסה.
- פורמט JPEG: מאתגר יותר, כי הוא מבצע דחיסת נתונים. נדרשות טכניקות במרחב התדרים DCT
- פורמט PNG: מתאים להסוואה בשיטות כמו LSB שכן הוא תומך בדחיסה ללא איבוד נתונים.
- אודיו:
- פורמט WAV: אידיאלי, כיוון שהוא אינו דחוס.
- פורמט MP3: קשה יותר להסוואה, כיוון שדחיסה מאבדת פרטים ש"לא נשמעים".
- דחיסת אודיו והסוואה: טכניקות כמו Phase Coding או Echo Hiding עוקפות את הבעיות של דחיסה.
- וידאו:
- שימוש בוידאו מאפשר הסוואת כמויות גדולות של מידע, כיוון שהנתונים מתפרסים על פני פריימים רבים.
- אתגרים: שינוי במסגרות בודדות עשוי להיות בלתי מורגש, אך דחיסה עלולה לפגוע במידע המוסתר.

יתרונות וחסרונות של טכניקות הסוואה

- יתרונות:
 1. חשאיות גבוהה: המידע המוסתר אינו גלוי לעין או נגיש ישירות.
 2. קושי לזיהוי: סטגנוגרפיה, בשונה מקריפטוגרפיה, אינה מבליטה את עצם קיומו של המידע המוסתר.
 3. קלות יישום: טכניקות כמו LSB פשוטות ליישום ומספקות רמה בסיסית של הסוואה.
- חסרונות:
 1. רגישות לשינויים: דחיסה או עריכה של הקובץ עלולה למחוק את המידע המוסתר.
 2. היקף מוגבל: בקבצים קטנים קשה להחביא מידע גדול.
 3. קלות זיהוי בטכניקות פשוטות: טכניקות כמו LSB קלות לזיהוי על ידי ניתוח סטטיסטי.

שימושים בסטגנוגרפיה בקבצי מדיה

- שימושים לגיטימיים:
 - סימני מים דיגיטליים Digital Watermarking הגנה על זכויות יוצרים.
 - אימות קבצים: שילוב נתוני אימות בקובץ כדי לוודא שלא שונה.
 - שימושים זדוניים:
 - גניבת נתונים: החבאת מידע מסווג והעברתו.
 - שימוש זדוני אפשרי קיים (למשל הסתרת מידע/פקודות בקבצי מדיה). בהגנה מתמקדים בזיהוי חריגות, סינון תכנים ובדיקות פורנזיות.

גילוי וניתוח סטגנוגרפיה

- טכניקות לניתוח:
 - ניתוח היסטוגרמות: שינויים לא טבעיים בפיזור ערכי פיקסלים או תדרים יכולים לחשוף מידע מוסתר.
 - ניתוח סטטיסטי: בדיקה אם יש תבניות חשודות, כמו פיזור ביטים לא אקראי בערכי RGB.
 - זיהוי דחיסת יתר: סטגנוגרפיה יכולה לשנות את מאפייני הדחיסה בקובץ.

שלבים בתהליך ההסוואה

- 1. קלט: בחירת קובץ המדיה המתאים (תמונה, אודיו, וידאו).
- 2. מידע מוסתר: המידע אותו נרצה להחביא (לדוגמה, מחרוזת טקסט, קובץ שלם).
- 3. שינוי הנתונים: הטמעה של המידע המוסתר בעזרת הטכניקות.
- 4. פלט: קובץ מדיה שנראה ומתפקד כרגיל, אבל מכיל בתוכו מידע סודי.

שיטות זיהוי ואיתור

- 1. היסטוגרמות של צבעים או ערכי אודיו: מזהים שינויים לא טבעיים בפיזור של ביטים במדיה.
- 2. ניתוח שאריות דחיסה: סטגנוגרפיה בתמונות JPEG יכולה להשאיר דפוסי דחיסה שונים.
- 3. כלי תוכנה לאיתור סטגנוגרפיה:
 - StegExpose: כלי אוטומטי שמנתח קבצי תמונות.
 - Forensic Tools: זיהוי תבניות חשודות בקבצים.

תשובה

- מרחב התדרים Frequency Domain הוא כלי חזק להסוואת מידע בקובץ JPEG שמתבסס על שינויים באזורים שקשה לזהות בעין אנושית. קבצי JPEG נעוברים תהליך דחיסה הכולל טרנספורמציה של התמונה למרחב התדרים באמצעות (DCT) במקום לעבוד ישירות על פיקסלים, דוחסים את התמונה על ידי פירוק למרכיבי תדרים:
 - תדרים נמוכים: מכילים את עיקר המידע של התמונה (כמו צבעים וצורות עיקריות).
 - תדרים בינוניים וגבוהים: מכילים פרטים עדינים של התמונה, ופחות בולטים לעין האנושית.
- תהליך ההסוואה:

- 1. פירוק התמונה למקטעי DCT:
- תמונה מחולקת לבלוקים של 8x8 פיקסלים.
- כל בלוק עובר טרנספורמציה ל-DCT שמייצרת מקדמים שמתארים את התדירים.
- 2. החבאת מידע בתדירים הגבוהים:
- במקום לשנות ישירות את הפיקסלים, מטמיעים ביטים של מידע במקדמי התדירים הגבוהים של כל בלוק.
- לדוגמה, אם יש לך מידע להחביא, תשנה מקדם בתדר גבוה בהתאם לערך הביט (0 או 1).
- 3. שחזור התמונה:
- לאחר הטמעת המידע, מבצעים אינברס DCT (IDCT) כדי להחזיר את התמונה למרחב הפיקסלים.
- 1. המרת הטקסט לנתונים בינאריים: כל אות בטקסט HELLO מתורגמת לקוד ASCII ואז לבינארי:
 - = H = 72 01001000
 - = E = 69 01000101
 - = L = 76 01001100
 - = O = 79 01001111
- כך, המידע הבינארי המלא הוא:
 - 01001111 01001100 01001100 01000101 01001000
- 2. שלבים להחבאה בסגמנט המשאבים:
- שלב 1: פתח את הקובץ באמצעות Hex Editor כמו HxD
- שלב 2: אתר את תחילת סגמנט המשאבים (rsrc.) לפי כותרת PE ניתן לזהות את הסגמנט בעזרת ה-Offset שמופיע בכותרת.
- שלב 3: מצא אזור ריק באורך של לפחות 40 ביטים (5 בתים) בתוך הסגמנט.
- שלב 4: כתוב את המידע הבינארי (לפי הטבלה למעלה) במקום הריק.
- 3. ווידוא שהקובץ נשאר שמיש:
- בדיקה 1: ודא שלא שינית נתונים קריטיים, כמו ערכים בכותרת או אזורים פעילים בקובץ.
- בדיקה 2: הרץ את קובץ ה-EXE ובדוק שהוא פועל כראוי.
- בדיקה 3: השתמש בכלי כמו PE Explorer כדי לוודא שהמבנה של הסגמנט נשאר תקין.
- 1. הטמעת המידע:
- המידע הסודי יומר לרצף בינארי (0 ו-1).
- לכל ביט (0 או 1) יותאם מיקום מסוים במסמך (למשל, שורות ותווים ספציפיים).
- המדפסת תוסיף נקודות צהובות זעירות במיקומים המתאימים, כך ש- "1" יסומן על ידי נקודה ו- "0" על ידי היעדר נקודה.
- 2. חילוץ המידע:
- העובד יקבל סורק מיוחד או אפליקציה המותאמת לקריאת הנקודות הבלתי נראות.
- הסורק יתרגם את מיקום הנקודות לרצף בינארי, שיעבור דה-קידוד למידע המקורי (למשל, טקסט או מספרים).
- 3. הבטחת סודיות:
- המידע יועבר מוצפן, כך שגם אם הנקודות הבלתי נראות יזוהו, הן לא יפוענחו ללא מפתח ההצפנה.
- המדפסת תדרוש הרשאה מיוחדת להדפסה בפורמט זה, למשל באמצעות סיסמה או כרטיס זיהוי של העובד המורשה.
- 1. תהליך הטמעת המידע:
- המידע יוטמע בשדות מטא-נתונים של המסמך, כגון:
- שדה Author (מחבר המסמך). [קובצי PDF מכילים שדות מטא-נתונים כמו Keywords, Title, Author, Subject ניתן לשלב את המידע בשדות אלה, במיוחד בשדות פחות בולטים כמו Keywords]
- שדה Keywords (מילות מפתח).
- ניתן להטמיע את המידע גם כשכבה בלתי נראית Invisible Layer במסמך PDF משום שמסמכי PDF תומכים בשכבות תוכן שניתן להציג או להסתיר, בנוסף ניתן להטמיע טקסט בלתי נראה על ידי שימוש בצבע לבן או בגופן בגודל 0.
- 2. שיטות למניעת גילוי:

- הצפנת המידע לפני הטמעתו: יש להצפין את המידע באמצעות אלגוריתם הצפנה כמו AES כך שגם אם יתגלה, לא יהיה קריא.
- שימוש במבנה רגיל של מטא-נתונים: המטא-נתונים יראו כחלק תקני של המסמך ולא יעוררו חשד.
- 3. תיאור טכני של התהליך:
- טוענים את קובץ ה-PDF באמצעות ספרייה מתאימה (למשל PyPDF2 בפיתון או iText בג'אווה).
- מוסיפים את המידע המוצפן לשדה מטא-נתונים כמו Keywords.
- שומרים את המסמך עם השינויים.
- לצורך קריאת המידע, משתמשים באותו מפתח הצפנה כדי לפענח את המידע המוסתר.

שאלה נוספת

- 2. אם הערך RGB של פיקסל הוא (255, 128, 64), כיצד היית משנה אותו כדי להחביא את הביטים 1, 0, 1?
- הטכניקה הנפוצה להסוואה בפיקסלים היא LSB (Least Significant Bit) שמשנה את הביט האחרון של כל ערך בערוץ הצבע (R, G, B) שלבים:
- 1. ייצוג RGB בבינארי:
- R (255) 11111111
- G (128) 10000000
- B (64) 01000000
- 2. מיפוי הביטים המוסתרים לערוצי RGB:
- הביטים להחביא הם 1, 0, 1
- כל ביט מותאם לערוץ צבע
- 1 לערוץ R
- 0 לערוץ G
- 1 לערוץ B
- 3. שינוי הביט האחרון בכל ערוץ:
- R אין צורך לשנות
- G אין צורך לשנות
- B משנים את הביט האחרון ל-1 ונקבל 01000001
- 4. הערכים החדשים:
- R- 255
- G- 128
- B- 65
- תוצאה סופית:
- הפסקול החדש של הפיקסל הוא (255, 128, 65).

מהו Chaffing and Winnowing?

- Chaffing and Winnowing היא טכניקה קריפטוגרפית שמיועדת להבטחת סודיות התקשורת, והיא פותחה על ידי רונלד ריבסט Ronald Rivest
- בניגוד לקריפטוגרפיה מסורתית, הטכניקה הזו אינה מצפינה את המידע אלא משתמשת באימות authentication כדי להפריד בין מידע אמיתי ל"מזויף".

כיצד השיטה מממשת אבטחת מידע?

- השיטה מבוססת על הרעיון של ניפוי רעש: זרם נתונים מורכב ממידע אמיתי ורעש Chaff המקבל משתמש במפתח הסודי כדי להבדיל בין נתונים אמיתיים למזויפים.
- שלבי הפעולה:
- 1. הפקת MAC (Message Authentication Code):
- ה-MAC מחושב עבור כל חבילת מידע בעזרת מפתח סודי.

- לדוגמה: חבילת מידע DATA1 תקבל MAC שמחושב לפי נוסחה:

$$MAC = H(DATA1, KEY)$$
- כאשר H היא פונקציית גיבוב כמו SHA-256.
- 2. הוספת רעש:
- חבילות מזויפות Chaff נוצרות ומתווספות לזרם המידע.
- לכל חבילה מזויפת מתווסף MAC ממזויף שאינו תואם לחבילה.
- 3. זיהוי במקבל:
- המקבל בודק את ה-MAC של כל חבילה.
- אם ה-MAC מתאים, החבילה מזוהה כאמיתית. אחרת, היא מזוהה כרעש וננפת.

עקרונות הפעולה

- 1. המונחים:
- Chaff: "פסולת" או "רעש" – נתונים מזויפים שמתווספים לזרם המידע כדי לבלבל את התוקף.
- Winnowing: "ניפוי" – תהליך שמאפשר למקבל המידע להפריד בין הנתונים האמיתיים לנתונים המזויפים.
- 2. תהליך הפעולה: המידע מחולק לחבילות packets
- כל חבילה מזוהה באמצעות קוד אימות Message Authentication Code - MAC שנוצר באמצעות מפתח סודי משותף.
- חבילות מזויפות Chaff מתווספות לזרם המידע, והן כוללות קודים לא תקינים או מזויפים.
- המקבל, שיש לו את המפתח הסודי, יכול לזהות את החבילות האמיתיות (באמצעות בדיקת ה-MAC) ולנפות את המזויפות.
- 3. מפתח הסודי: רק מי שיש לו את המפתח הסודי יכול להבדיל בין חבילות אמיתיות למזויפות.
- המידע המועבר הוא "גלוי", אך היכולת לזהות את המידע הרלוונטי מוגנת על ידי המפתח.

יתרונות

- 1. ללא הצפנה:
- בניגוד לשיטות הצפנה מסורתיות, המידע אינו מוצפן ואינו דורש כוח עיבוד רב לפענוח.
- אין צורך ברישיון לשימוש בטכנולוגיות הצפנה באזורים עם הגבלות חוקיות.
- 2. עמידות לתקיפה:
- קשה מאוד לתוקף להבחין בין חבילות אמיתיות למזויפות ללא המפתח הסודי.
- כמות ה"רעש" Chaff יכולה להיות גדולה כדי להקשות על ניסיון זיהוי.
- 3. שימושים אפשריים:
- מניעת גישה למידע על ידי יריבים.

חסרונות

- 1. תלות במפתח הסודי:
- אם המפתח נחשף, כל המידע יהיה חשוף.
- 2. בזבוז משאבים:
- הוספת כמויות גדולות של חבילות מזויפות יכולה להגדיל משמעותית את רוחב הפס הנדרש.
- 3. לא מונע יירוט מידע:
- המידע האמיתי והנתונים המזויפים מועברים "בבירור". לכן, המידע אינו מוגן מפני יירוט אלא רק מפני הבנה.

יישומים מעשיים מתקדמים

- א. אבטחת תקשורת ברשתות ציבוריות: ברשתות ציבוריות כמו Wi-Fi, Chaffing and Winnowing יכול להקשות על תוקפים ליירט מידע, משום שהם אינם יכולים לזהות אילו חבילות הן אמיתיות ואילו מזויפות.
- דוגמה: בתקשורת HTTP שאינה מוצפנת, ניתן להוסיף חבילות מזויפות כדי להקשות על תוקפים שמנסים לנתח תעבורה.

- ב. הגנה מפני מתקפות ניתוח תעבורה: מתקפות ניתוח תעבורה Traffic Analysis מנסות לחשוף מידע על בסיס מאפיינים כמו גודל חבילות, זמן שליחה ותדירות.
- הוספת Chaff גורמת לתעבורה להיראות אקראית יותר, מה שמקשה על מתקפות מסוג זה.
- ג. העברת מסרים במדינות עם פיקוח על הצפנה: במדינות שבהן יש מגבלות על שימוש בהצפנה, הטכניקה יכולה לשמש כחלופה להעברת מידע בצורה מאובטחת.

ניתוח אבטחה מתקדם

- התקפות אפשריות נגד השיטה:
 1. מתקפות ניחוש מפתח: תוקף עשוי לנסות לנחש את המפתח הסודי כדי לזהות את החבילות האמיתיות.
 2. פתרון: שימוש במפתחות חזקים ובפרוטוקולים לניהול מפתחות.
 3. ניתוח מבוסס תוכן: אם החבילות המזויפות נראות שונה מהחבילות האמיתיות (למשל, בגודל או במבנה), תוקף יכול לנסות לזהות אותן.
 4. פתרון: הקפדה על כך שהחבילות המזויפות ייראו זהות לחבילות האמיתיות מבחינת מבנה וגודל.
 5. מתקפת מניעת שירות DoS: תוקף יכול להציף את התקשורת בכמות גדולה של חבילות מזויפות כדי להפריע למקבל.
 6. פתרון: מנגנוני זיהוי וניפוי חבילות בזמן אמת.

תובנות מתמטיות על יצירת רעש

- א. פונקציות גיבוב Hash Functions: פונקציות גיבוב משמשות ליצירת קודי MAC ולוודא שהנתונים לא שונו.
- דוגמה: HMAC ו-SHA-256.
- ב. הדור אקראי של רעש: הוספת רעש Chaff דורשת יצירת נתונים אקראיים.
- חשוב שהנתונים האקראיים ייראו "אמיתיים" ככל האפשר כדי למנוע זיהוי.

קשר בין Chaffing and Winnowing לסטגנוגרפיה

- סטגנוגרפיה עוסקת בהסתרת מידע בתוך מדיה (תמונה, וידאו, אודיו), ואילו Chaffing and Winnowing עוסקת בהסוואת המידע בתוך זרם נתונים.
- השיטה של Chaffing יכולה להשתלב עם סטגנוגרפיה, למשל:
- מידע שמוסתר בתמונה מחולק לחבילות מידע, ולאחר מכן מתווסף רעש Chaff שמסתיר את החבילות האמיתיות.

מהו Null Cipher

- Null Cipher הוא טכניקת הסתרה פשוטה ועתיקה שמשתמשת בטקסט רגיל כדי להחביא מסר סודי.
- הגדרה:
- Null Cipher הוא סוג של סטגנוגרפיה טקסטואלית שבו המסר המוסתר נשמר בטקסט רגיל על ידי שימוש באותיות נבחרות ממילים, משפטים או פסקאות. המסר המוסתר מוסווה כחלק מהטקסט, ולכן מי שאינו יודע את הכלל לא יוכל לקרוא את המידע.

דוגמאות לשימוש בטכניקה

1. בחירת אותיות לפי מיקום מסוים לדוגמה: טקסט: "אני אוהב לטייל בים בשבתות ולצפות בשקיעה." כלל: קח את האות הראשונה של כל מילה. מסר: "אולטבבשו" (לא בהכרח קריא).
2. בחירת אותיות לפי תבנית ידועה מראש לדוגמה: טקסט: "ליל הסערה היה קר וגשום, ופתאום נשמעה דפיקה בדלת." כלל: קח את האות השלישית מכל מילה. מסר: "סההווהפעהקה".
3. שימוש במרווחים בין מילים או בפיסוק לדוגמה: הטקסט הרגיל מכיל מסר נסתר שמפוענח לפי מרחקים בין המילים או סימני הפיסוק.

יתרונות השיטה

- 1. פשטות: אינה דורשת טכנולוגיה מתקדמת או תוכנה מיוחדת.
- 2. התמזגות טבעית: הטקסט נראה רגיל לחלוטין ולא מעורר חשד.
- 3. עמידות לזיהוי אוטומטי: קשה לכלים אוטומטיים לחשוף את המסר אם הכלל אינו ידוע.

יתרונות טקטיים של Null Cipher

- 1. טבעיות ופשטות:
- המראה הטבעי של הטקסט מאפשר להעביר מסרים סודיים מבלי לעורר חשד.
- 2. קלות יישום:
- אין צורך בטכנולוגיה מיוחדת או במפתחות הצפנה.
- 3. עמידות בפני כלי ניתוח בסיסיים:
- Null Cipher יעיל במיוחד כאשר הכללים מורכבים או משתנים, כי קשה לנתח את המידע באופן אוטומטי.

חסרונות השיטה

- 1. מוגבלת בכמות המידע: כמות המידע שניתן להסתיר מוגבלת מאוד.
- 2. רגישות לחשיפת הכלל: אם התוקף מגלה את הכלל, כל המסר נחשף.
- 3. תלות בטקסט ברור: דורשת טקסט איכותי ומובן כדי לא לעורר חשד.

שימושים עכשוויים של Null Cipher

- מעבר מסרים סמויים: בתקופות מלחמה או בתנאים של מעקב אינטנסיבי, שיטה זו שימשה להעברת מסרים סודיים דרך טקסטים שנראו תמימים.
- תרגול באבטחת מידע: כיום, Null Cipher משמשת בעיקר כחלק מתרגולים בלימודי סטגנוגרפיה ואבטחת מידע.

אתגרים והגבלות

- 1. מגבלות מידע:
- קשה להעביר כמויות גדולות של מידע.
- אם רוצים להחביא מסר מורכב, יש צורך בטקסט ארוך מאוד.
- 2. חשיפת הכלל:
- אם המפתח (הכלל לבחירת האותיות) נחשף, המסר הסודי מאבד מהאבטחה שלו.
- 3. חוסר עמידות לעיבוד נתונים:
- אם הטקסט עובר עיבוד כלשהו (כמו דחיסה או עריכה), המידע המוסתר עלול להיהרס.

שילוב עם טכניקות אחרות

- 1. סטגנוגרפיה רב-שכבתית:
- Null Cipher יכול לשמש כשכבה ראשונה להסתרת מסר, ולאחר מכן אפשר לשלב את הטקסט המוסתר בתוך תמונה, אודיו או קובץ אחר.
- 2. קידוד נוסף:
- אפשר להסתיר את המסר עם Null Cipher ואז להצפין את האותיות שנבחרו עם צופן נוסף.

טכניקות נגדיות לזיהוי Null Cipher

- א. ניתוח סטטיסטי של הטקסט: כלים לניתוח טקסט יכולים לזהות דפוסים כמו תדירות אותיות, מילות מפתח, או דילוגים חוזרים.
- ב. שימוש באלגוריתמים לזיהוי דפוסים: אלגוריתמים ללמידת מכונה יכולים לנתח טקסטים ולגלות חריגות שמעידות על שימוש ב-Null Cipher.

- ג. ניתוח סמנטי: ניתוח משמעותי של הטקסט יכול לגלות אם הטקסט מנסה להעביר מסר נסתר דרך מילים מסוימות או מבנה משפטים.

דוגמאות לשימוש מתקדם

- א. שילוב של טכניקות סטגנוגרפיות: לדוגמה: טקסט רגיל שמכיל מסר ב-Null Cipher משולב עם תמונה שבה מוסתר מסר נוסף.
- ב. שימוש בטקסטים ציבוריים: ניתן להשתמש בטקסטים קיימים (למשל, נאומים, שירים או מאמרים) ולהתאים אותם כך שיכילו מסרים נסתרים.

תהליך מעשי ליצירת מסר באמצעות Null Cipher

1. הכנת הטקסט הרגיל: כתוב טקסט רגיל שנראה טבעי.
2. בחירת כלל הסתרה: בחר כלל לבחירת אותיות, לדוגמה כל אות שנייה.
3. פיזור המסר: השתמש בכלל כדי לשלב את האותיות בתוכן.
4. מסירת הטקסט: מסור את הטקסט הרגיל עם המסר המוסתר.

ניצול יתרות בקובצי הרצה וספריות בסטגנוגרפיה

- נושא זה עוסק בשימוש בנתונים בלתי מנוצלים או שדות יתירים בקובצי הרצה כגון קבצי EXE ובספריות (כגון DLL) להחבאת מידע סודי. מדובר באחת הטכניקות המתקדמות בסטגנוגרפיה דיגיטלית, והיא מנצלת מבנה פנימי של קובצי תוכנה.

מהן יתרות בקובצי הרצה וספריות?

- יתרות בקובצי הרצה EXE:
- בקובצי EXE קיימים שטחים שאינם בשימוש, כגון:
- שדות יתירים במבנה הכותרת Header.
- אזורים ריקים בין חלקי הקובץ, הנובעים מיישור נתונים בזיכרון Pad.
- אזורים שאינם מנוצלים על ידי התוכנה במהלך ההרצה ding
- יתרות בספריות דינמיות: DLL
- קבצי DLL משמשים כספריות קוד שניתן לטעון בזמן ריצה. גם בהם יש שדות ריקים או אזורים לא מנוצלים.

כיצד מנצלים יתרות אלו להחבאת מידע?

- א. החבאת מידע בכותרת הקובץ Header:
- קובצי EXE ו-DLL מכילים כותרות עם שדות שלא תמיד מנוצלים.
- ניתן לכתוב מידע סודי בשדות אלה בלי להשפיע על תפקוד הקובץ.
- ב. החבאת מידע בשדות Padding:
- במהלך יצירת קבצים, תהליך יישור הנתונים יוצר שטחים ריקים שמיועדים להתאמת נתונים לגדלים מסוימים (לדוגמה: מיושרים לגודל של 4 בתים).
- ניתן להשתמש בשטחים אלו להחבאת נתונים מבלי לפגוע בפעולת הקובץ.
- ג. החבאת מידע בנתונים שאינם קריטיים
- בקובצי תוכנה יש נתונים שאינם קריטיים להרצה (כגון נתוני גרפיקה, הערות או Metadata ניתן לשנות נתונים אלו ולהחביא בהם מסר סודי).
- ד. הוספת סגמנטים חדשים Segments
- בקובצי EXE ניתן להוסיף סגמנט חדש לתוכנית (למשל, קטע קוד או נתונים) שמכיל את המידע המוסתר.
- סגמנט זה נרשם בכותרת הקובץ, אך לא מבוצע בפועל במהלך הרצת התוכנית.

מבנה קובצי PE (Portable Executable)

- קובצי הרצה מבוססי Windows כגון EXE ו-DLL פועלים לפי פורמט PE הפורמט כולל מבנים ותתי-מבנים המספקים נקודות רבות להחבאת מידע. נסקור מבנה זה בקצרה:
- DOS Header: הכותרת הראשונית בפורמט DOS שכוללת שדות רבים שאינם מנוצלים במערכות מודרניות.
- PE Header: הכותרת המרכזית של הפורמט. היא מכילה שדות המצביעים למיקום נתונים וקוד.
- Sections: אזורים עיקריים של הקובץ:
- text: מכיל את הקוד הבינארי של התוכנית.
- data: מכיל נתונים גלובליים.
- rsrc: משאבים כמו אייקונים, תפריטים ותמונות.
- padding: שדות בסגמנטים או סגמנטים יתירים מהווים מיקום פוטנציאלי להסתרה.
- נקודות להחבאת מידע במבנה PE:
- 1. שדות שאינם בשימוש בכותרות: לדוגמה, שדות ששמורים לעתיד או שדות שאינם מנוצלים במערכות מודרניות.
- 2. שדות יתירים בין סגמנטים: יישור הנתונים בקובץ לגדלים מסוימים יוצר אזורים ריקים.
- 3. שדות בסגמנט המשאבים (.rsrc): ניתן לשנות או להוסיף מידע נסתר לתוך אובייקטים כמו אייקונים או טקסטים בקובץ.

שיטות חדשניות להחבאת מידע בקובצי הרצה וספריות

- א. הוספת מידע לסגמנט חדש Custom Section: אפשרות זו כוללת יצירת סגמנט חדש (לדוגמה, hidden) והוספת המידע המוסתר לשם.
- יתרון: לא משפיע על התפקוד של סגמנטים אחרים.
- חסרון: קל לזיהוי אם מישהו משווה את הקובץ המקורי לקובץ המעובד.
- ב. החבאת מידע באמצעות שגיאות מכוונות: שגיאות מכוונות (כגון שינוי checksum או הזנת ערכים לא תקינים) מאפשרות להחביא ביטים של מידע.
- יתרון: קשה לזיהוי אם השגיאה נראית "טבעית".
- חסרון: עלולה לגרום לתקלה בהרצת הקובץ.
- ג. החבאת מידע בתוך נתוני Debug: קבצי הרצה יכולים להכיל נתוני Debug שאינם קריטיים להרצה, ולכן ניתן להוסיף לשדות אלו מידע מוסתר.
- ד. שינוי תוכן שאינו בשימוש בפועל: לדוגמה: שינוי ערכים בקובץ הריצה שמשפיעים רק על תצוגת הודעות למשתמש, ולא על הפונקציונליות.

אתגרים ושיקולים טכניים

- א. שימור פונקציונליות הקובץ: כל שינוי חייב להיעשות בזהירות כדי שהקובץ ימשיך לפעול כראוי.
- לדוגמה, שינוי בסגמנט text עלול לשנות את התנהגות התוכנה ולגרום לתקלות.
- ב. זיהוי אוטומטי של תוכנות אבטחה: מערכות אנטי-וירוס וסורקי קבצים משתמשים בטכניקות מתקדמות כדי לזהות שינויים לא טבעיים בקבצי EXE.
- פתרון: להשתמש בטכניקות מתוחכמות שמחקות את מבנה הקובץ המקורי.
- ג. ניתוח סטטיסטי: כל שינוי חייב להיראות טבעי מבחינת גודל הקובץ ותוכנו, כדי להימנע מחשד.

כלים וטכניקות לזיהוי מידע מוסתר

- א. ניתוח מבנה הקובץ: כלים כמו PE Explorer ו-IDA Pro מאפשרים ניתוח מבנה קובץ וזיהוי שדות חשודים.
- ב. השוואת קבצים: השוואת הקובץ לקובץ המקורי יכולה לחשוף הבדלים חשודים.
- ג. ניתוח נתוני Metadata: בדיקת שדות Metadata לא טיפוסיים יכולה להצביע על שינויים שנעשו בקובץ.
- ד. ניתוח התנהגותי: הרצת הקובץ במכונה וירטואלית ובחינת השפעותיו מאפשרת לזהות קוד או נתונים מוסתרים.

יתרונות וחסרונות של ניצול יתרות בקובצי הרצה

- יתרונות:
- 1. שקיפות למשתמש: הקבצים נראים רגילים ופועלים כרגיל, מה שמקשה על זיהוי.
- 2. יכולת להחביא כמויות גדולות של מידע: במיוחד אם הקובץ מכיל הרבה שטחים ריקים.
- 3. עמידות לתוכנות אנטי-וירוס בסיסיות: רוב כלי האבטחה לא מזהים מסרים מוסתרים בשדות יתירים.
- חסרונות:
- 1. רגישות לשינויים בקובץ: עדכון או עריכה בקובץ עלולה למחוק את המידע המוסתר.
- 2. חשד מצד כלי ניתוח מתקדמים: כלים מתקדמים יכולים לזהות אזורים "מוזרים" בקובץ.
- 3. תלות במבנה הקובץ: המידע צריך להתאים למבנה הספציפי של הקובץ ולא להשפיע על תפקודו.

אתגרים בניצול יתרות בקובצי הרצה

- 1. ניתוח מבנה הקובץ: יש צורך להבין את מבנה הקובץ כגון PE Header בקובצי EXE.
- 2. מניעת פגיעה בהרצת הקובץ: חובה לוודא שהוספת המידע המוסתר לא תפגע בפונקציונליות של הקובץ.
- 3. גילוי על ידי כלים אוטומטיים: כיום קיימים כלים מתקדמים שמסוגלים לנתח את מבנה הקובץ ולגלות שינויים חשודים.

שאלה מעשית בכדי לשכך את האוזן

- נתון קובץ הרצה EXE פשוט הכולל את מבנה PE הסטנדרטי. בסגמנט המשאבים (rsrc) יש אזור ריק של 100 בתים שלא מנוצל.
- עליך להחביא את המידע הבא: "HELLO" באסקי
- 1. הסבר כיצד היית ממיר את הטקסט לנתונים בינאריים.
- 2. פרט את הצעדים שצריך לבצע כדי להוסיף את המידע לסגמנט המשאבים.
- 3. כיצד תוודא שהקובץ יישאר שמיש לאחר ההחבאה?

מהי הסוואה בציווד היקפי

- "הסוואה בציווד היקפי" היא שיטה בסטגנוגרפיה שבה מידע מוסתר בתוך התקנים פיזיים או נתונים שנוצרים על ידם. במקום להסתיר מידע בקבצים או בתקשורת דיגיטלית, הסוואה זו עושה שימוש בהתקנים חיצוניים כמו מדפסות, כונני USB מצלמות, מקלדות, או כל התקן אחר המתחבר למחשב.

מטרות השיטה

- 1. העברת מידע בסתר: ניצול ציווד היקפי להעברת נתונים מבלי לעורר חשד.
- 2. מניעת זיהוי ע"י מערכות אבטחה: רוב תוכנות האבטחה מתמקדות בקבצים ותעבורת רשת, ולא בציווד היקפי.
- 3. שימוש באמצעים פיזיים כערוץ להעברת נתונים: לדוגמה, הדפסת מידע בקודים בלתי נראים, או שימוש בויברציות של מכשירים להעברת נתונים.

שיטות להסוואה בציווד היקפי

- 1. הדפסה נסתרת Steganography in Printers: מדפסות יכולות לשמש להחבאת מידע בדרכים יצירתיות: שימוש בנקודות צהובות בלתי נראות: רבות מהמדפסות המודרניות מוסיפות באופן אוטומטי נקודות צהובות זעירות למסמכים שהן מדפיסות. ניתן לנצל שיטה זו להוספת נתונים מוסתרים בתוך ההדפסה.
- שימוש בקודים נסתרים בתמונות מודפסות: ניתן לשנות את מיקום הפיקסלים בתמונה או את הצבעים בצורה שלא תהיה גלויה לעין האנושית.
- 2. שימוש במקלדות או עכברים Keyboards and Mice: הסתרת נתונים בתקשורת HID (Human Interface Device): מכשירים אלה מתקשרים עם מערכת ההפעלה באמצעות פרוטוקולים סטנדרטיים. ניתן להטמיע נתונים מוסתרים בתוך אותות המועברים מהמקלדת או העכבר.

- שליחת מידע באמצעות לחיצות מקשים: למשל, מקלדת חכמה יכולה להעביר נתונים מוסתרים על ידי שינויים מזעריים בעיכוב בין לחיצות מקשים.
- 3. מצלמות ודיסק און קי USB Drives: שימוש במצלמות להעברת נתונים: מצלמות יכולות להקליט מידע מוסתר בתמונות או בסרטונים, למשל, באמצעות מיקוד או צבעים שאינם גלויים לעין האנושית (כגון תדרי אינפרה-אדום).
- החבאת נתונים בתוך Firmware של USB Drives: ניתן לשנות את הקוד הפנימי (Firmware) של כונני USB כדי להסתיר מידע בסקטורים בלתי נראים.
- 4. התקני קול ורמקולים: העברת נתונים באמצעות גלי קול: רמקולים יכולים להשמיע גלי קול בתדרים שאינם נשמעים לאוזן האנושית (אולטרסוניק), וכך להעביר נתונים באופן נסתר.
- הקלטה סמויה במיקרופונים: ניתן להחביא נתונים בתוך אותות מוקלטים, בדומה לסטגנוגרפיה בקבצי אודיו.

1. ניצול רכיבים אלקטרוניים להעברת מידע

- רכיבים אלקטרוניים נפוצים, כמו כרטיסי רשת, מצלמות, ומקלדות, יכולים להיות מנוצלים להחבאת מידע, במיוחד כשהם חלק ממערכות שנמצאות בשימוש שוטף. הנה כמה רעיונות:
- שינוי תדרי פעולה: חלק מהרכיבים פועלים בתדרים משתנים (למשל, שעון פנימי של רכיב). ניתן להשתמש בתדרים אלו להעברת מידע מוסתר, כמו רצפים בינאריים.
- תוכנות Firmware מותאמות אישית: עדכון ה-Firmware של התקנים פיזיים כך שיכילו מידע מוסתר הוא דרך מתוחכמת וקשה לזיהוי.

2. שימוש במיקרופונים ורמקולים להעברת מידע

- 2. העברת נתונים באמצעות גלי קול: ניתן לשדר נתונים בתדרי אולטרסאונד שאינם נשמעים לאוזן האנושית, ולקלוט אותם באמצעות מיקרופונים רגישים במיוחד.
- שימוש בקידוד כמו Amplitude Modulation (AM) או Frequency Modulation (FM) יאפשר העברת מידע יציבה ומוסתרת יותר.
- דוגמה מעשית: דמיון מצב שבו רמקול במחשב משדר "קול מוסתר" לחדר אחר, והמידע נקלט ע"י מיקרופון שמחובר למחשב אחר. על בסיס צלילים בתדרים שונים, ניתן להעביר ביטים בודדים.

3. שימוש במצלמות חכמות

- טשטוש מכון בתמונות: מצלמה יכולה לצלם תמונה עם "שגיאות מכוונות" בצבע או במיקום פיקסלים. השגיאות מכילות את המידע הסמוי.
- שימוש באינפרה-אדום: מצלמות המסוגלות ללכוד תדרי אינפרה-אדום יכולות לשדר נתונים מוסתרים למקלטים מותאמים, כמו חיישנים או מצלמות אחרות.
- דוגמה: תמונה תמימה יכולה לכלול מידע חבוי במקומות שלא מורגשים בעין, כמו זוויות חשוכות, פרטים מטושטשים, או "רעש" מכון.

4. הדפסה נסתרת במדפסות

- הדפסת נקודות בלתי נראות: כפי שהוזכר קודם, ניתן להחביא נתונים בתוך נקודות צהובות זעירות המודפסות על המסמך. הנקודות ניתנות לקריאה רק ע"י סורק מיוחד.
- קידוד פיקסלים: במדפסת תמונה, ניתן לשנות את ערכי הפיקסלים באופן שאינו פוגע בתמונה עצמה, אך כולל מידע סמוי.

5. שימוש בכונני USB

- החבאה ב-Firmware של הכונן: ניתן לשנות את ה-Firmware (תוכנה פנימית) של כונן ה-USB כדי להוסיף חלקי קוד שמסתירים נתונים.

- שימוש בסקטורים בלתי נראים:
- כל כונן USB מכיל סקטורים ריקים Unallocated Space ניתן להטמיע מידע באזורים אלו מבלי שזה יראה במערכת ההפעלה.

יתרונות וחסרונות של הסוואה בציווד היקפי

- יתרונות:
 1. קושי בזיהוי: רוב מערכות האבטחה אינן מנתחות את המידע שמועבר בציווד היקפי.
 2. אפשרויות יצירתיות: ניתן לנצל מגוון רחב של התקנים להעברת נתונים.
 3. נגישות גבוהה: ציווד היקפי זמין ונפוץ, ולכן קל להשתמש בו להסתרת נתונים.
- חסרונות:
 1. מגבלות טכניות: חלק מהשיטות דורשות ידע מתקדם בתכנות ובחומרה.
 2. חשש מזיהוי ידני: אם מישהו בודק את ההתקנים באופן פיזי, ניתן לגלות את המידע.
 3. שיעור העברת נתונים מוגבל: ברוב המקרים, קצב העברת הנתונים נמוך יחסית.

אתגרים

- 1. איתור המידע המוסתר:
- מידע שמוסתר בציווד היקפי קשה מאוד לגילוי, במיוחד כשהוא נמצא בחומרה ולא בתוכנה.
- 2. שמירה על תפקוד תקין של ההתקן:
- כל שינוי ב-Firmware או במבנה הקובץ עלול לגרום לתקלות בתפקוד הציווד.
- 3. אבטחה מוגברת בציווד מודרני:
- חלק מההתקנים כוללים מערכות שמונעות שינויים לא מורשים, כמו חתימות דיגיטליות ב-Firmware.

פתרונות

- 1. הצפנה של המידע המוסתר:
- גם אם המידע מזוהה, הצפנה תמנע את קריאתו על ידי גורמים זרים.
- 2. שימוש בשיטות סטגנוגרפיות עדינות:
- טכניקות שמפחיתות את הסיכון לשינויים בולטים במכשירים.
- 3. תכנון מבוסס תרחישים:
- שימוש בציווד שנבדק בקפידה כדי להבטיח שתפקודו לא ייפגע.

שאלה מעשית

- אתה עובד כמתכנן אבטחת מידע במכללת כנפי רוח אצל יוסי לביא. יש לך מדפסת משרדית שתומכת בהדפסת נקודות בלתי נראות Invisible Dots המנהל שלך ביקש ממך לפתח שיטה שתאפשר להעביר מסר סודי לעובד אחר באמצעות המדפסת, תוך שמירה על סודיות המסר והסוואתו מפני גורמים שאינם מורשים.
- משימות:
 1. תאר כיצד תטמיע את המידע הסודי בתוך הדפסת מסמך באמצעות נקודות בלתי נראות.
 2. כיצד העובד יוכל לחלץ את המידע מהמסמך המודפס?
 3. מה תעשה כדי להבטיח שאף גורם בלתי מורשה לא יוכל לקרוא את המידע החבוי?
- נניח שאתה מתכנת מערכת לניהול מסמכים, וברצונך להוסיף תכונה שמאפשרת הטמעת מידע סודי בקובץ PDF מבלי לשנות את הטקסט או העיצוב הגלוי במסמך. המידע הסודי צריך להיות מוסתר בצורה כזו שרק מי שמכיר את המפתח לקריאת המידע יוכל לגשת אליו.
- 1. כיצד תבצע את הפעולה הזו מבחינה טכנית?
- 2. אילו טכניקות היית משתמש כדי לוודא שהמידע לא יתגלה בבדיקה רגילה של המסמך?
- 3. כתוב תיאור טכני של התהליך שבו תשתמש, כולל אילו שדות או מבנים בקובץ PDF יישמשו להסתרת המידע.

סיכוך מידע בסטגנוגרפיה

- מה זה סיכוך מידע?
- סיכוך מידע הוא תהליך שבו מידע רגיש או סודי מוגן על ידי עטיפה (או "סיכוך") של המידע בתוך קונטקסט שאינו נראה או נגיש ישירות. בניגוד להחבאת מידע Steganography שבה מסתירים את המידע בתוך מדיה (תמונה, אודיו, וידאו), סיכוך מידע מתמקד בהגנה על המידע על ידי הפרדתו, הצפנתו, או הסטתו למקום שבו לא ייחשף בקלות.

השימושים המרכזיים בסיכוך מידע

1. הסתרת מיקום המידע הרגיש: מידע מאוחסן במקומות שאינם נגישים למשתמש הממוצע, למשל, אזורים לא מנוצלים בזיכרון או בקבצים
2. יצירת שכבת הגנה נוספת: סיכוך משמש כשכבת הגנה נוספת מעל הצפנה או טכניקות אבטחה אחרות, כך שגם אם הצפנה נפרצת, הסיכוך מונע את זיהוי המידע.
3. הונאה ואבטחה מבוססת דיסאינפורמציה: סיכוך יכול להסתיר את המידע האמיתי באמצעות יצירת מידע "מיותר" או מטעה.

שיטות מרכזיות לסיכוך מידע

1. שימוש במבני נתונים בלתי נראים
- ניצול שטח ריק בקבצים Slack Space : שטח ריק שנשאר לאחר כתיבת קובץ לדיסק עשוי להיות מנוצל להחבאת מידע סודי.
- אזורים לא מנוצלים בזיכרון: תוכנות מסוימות שומרות חלקים לא מנוצלים בזיכרון העבודה RAM שאליהם ניתן להכניס מידע סודי.
2. ריבוי שכבות הגנה
- שימוש בשכבות אבטחה מרובות (למשל, שילוב בין הצפנה וסטגנוגרפיה) כדי להבטיח שלא ניתן יהיה לזהות את המידע.
- הטמעת המידע בתוך מידע "מיותר" או "רעש".
3. הסוואה באמצעות מבנים תקינים
- מידע סודי מוטמן בתבניות קיימות כמו טבלאות נתונים או לוגים, כך שהמבנה אינו נראה חשוד.
4. שימוש במטא-נתונים
- מידע מוסתר במטא-נתונים של קבצים (כגון תמונות, מסמכים וקבצי וידאו). לדוגמה: הוספת שדות נסתרים בקובץ PDF

1. שילוב סיכוך עם הצפנה

- שימוש בהצפנה לפני הסיכוך: לפני שמסווים את המידע, ניתן להצפין אותו. גם אם הסיכוך נחשף, המידע עצמו יישאר בלתי קריא ללא מפתח ההצפנה.
- יתרון: שכבת הגנה כפולה.
- אתגר: מורכבות ניהול המפתחות.
- שימוש בקוד הטעיה Decoy Code: ניתן לשלב קוד הטעיה בתוך המידע הסמוי כדי להסוות מידע אמיתי. לדוגמה, קובץ שמכיל שתי שכבות: שכבה חיצונית עם נתונים תמימים ושכבה פנימית מוצפנת.

2. סיכוך במבני קבצים מורכבים

- סיכוך במבני קבצים מורכבים
- קבצי PDF: בפורמט זה ניתן להחביא מידע באמצעות שדות נסתרים, אובייקטים לא מוצגים Hidden Objects או ריווח שולי המסמך.
- דוגמה: ניתן להוסיף שכבות בלתי נראות הכוללות טקסט מוסתר.
- קבצי ZIP או RAR: בפורמטים אלה ניתן לסכך מידע על ידי הוספת קבצים תמימים לצד הקובץ הרגיש, כך שהוא ייראה כחלק מהחבילה התקנית.

- קבצי וידאו ואודיו: פורמטים אלה מציעים אפשרויות רבות לסיכון, כמו הסתרת מידע באזורים שאינם משפיעים על איכות התצוגה או השמע (לדוגמה, אזורים עם רעש רקע).

3. סיכון מבוסס תעבורה

- טכניקות להחבאת מידע בתעבורת רשת:
- שימוש בכותרות פרוטוקול Protocol Headers: ניתן לשלב מידע מוסתר בכותרות של פרוטוקולים, כמו HTTP או TCP. לדוגמה: הכנסת מידע שולי לכותרת User-Agent
- הטמעה בתוך נתונים רגילים: ניתן להוסיף מידע מוסתר למידע שמועבר בתעבורה שגרתית, כמו תעבורת דוא"ל, צ'אט, או נתוני וידאו זורמים.
- שימוש במידע מיותר Padding: חלק מהפרוטוקולים מוסיפים מידע מיותר לנתונים Padding ניתן לנצל את האזורים הללו להטמעת מידע סודי.

4. דינמיקה של אבטחה באמצעות סיכון מידע

- הגנת עומק Defense in Depth: סיכון מידע משמש כחלק ממערך אבטחה רחב יותר, שבו מספר שכבות הגנה מגנות על המידע. לדוגמה: גם אם פורצים את שכבת ההצפנה, המידע עדיין מוגן על ידי הסיכון.
- טכניקות להטמעה אקטיבית: יצירת מערכות שמכילות מידע מטעה המכוון את התוקפים למידע שאינו רלוונטי.
- דוגמה: הטמעת קבצים שנראים רגישים אך למעשה מכילים "פיתיונות" Decoys

דוגמאות מעשיות לסיכון מידע

1. קובץ תמונה:
- הקובץ מכיל מטא-נתונים עם פרטים כמו מיקום צילום, אבל בפועל, המידע הסודי מוטמן במטא-נתונים במקום בפיקסלים עצמם.
2. מסמך טקסט:
- שימוש ברווחים מיותרים או תווים נסתרים כדי לאחסן מידע.
3. תעבורת רשת:
- סיכון המידע באמצעות תעבורת רקע בלתי מזיקה, למשל על ידי שילוב נתונים מוסתרים בתוך בקשות HTTP תמימות.

האתגרים בסיכון מידע

1. זיהוי אוטומטי:
- תוכנות אבטחה מתקדמות יכולות לזהות פעילות חשודה באזורים לא מנוצלים.
2. נפח מידע מוגבל:
- הטמעת מידע באזורים קטנים, כמו מטא-נתונים או Slack Space מגבילה את כמות המידע שאפשר לסכך.
3. סיכון לגילוי עקיף:
- אם הסיכון אינו מושלם, ניתן לגלות את המידע בעזרת ניתוח מעמיק של המערכת.

האפלט מידע בסטגנוגרפיה

- האפלט מידע היא טכניקה שבה אנו מסווים את המשמעות האמיתית של המידע או משנים את המבנה שלו כך שהוא ייראה כמידע לא רלוונטי, חסר ערך, או חסר משמעות, ובכך מקטינים את הסיכוי שמישהו יזהה את המידע או יבין אותו. זהו כלי נפוץ בעולם הסטגנוגרפיה והאבטחת מידע, והוא מאפשר להחביא מידע רגיש בצורה יצירתית.

עקרונות ההאפלה Obfuscation

1. שינוי מבנה המידע: המידע הרגיש מקודד או מוצפן כך שהוא נראה חסר משמעות לעין בלתי מזוינת.
2. הסוואה במידע זמין: הטמעת המידע בתוך תוכן קיים, כגון קבצים, נתונים סטטיסטיים, או טקסטים נפוצים.
3. שימוש באלגוריתמים מתקדמים: לדוגמה, הצפנה או שינוי סדרי ביטים בצורה שמסתירה את המידע.
- מטרות ההאפלה:

- 1. הסוואת קיום המידע: כך שהגורם התוקף לא יחשוד בקיומו של מידע בעל ערך.
- 2. מניעת הבנה: גם אם המידע מתגלה, הוא לא יהיה קריא או שמיש.
- 3. הגנה על פרטיות: מניעת גישה למידע רגיש מבלי לסכן את שלמותו.

כיצד ניתן להאפיל מידע?

- דוגמאות לטכניקות אפלה:
- 1. הסוואה בקובצי טקסט: כתיבת מידע בצורה של טקסט נורמלי, אך הסתרת המסר האמיתי באמצעות דילוגים, קודים, או החלפות.
- לדוגמה: המידע יכול להיות בקוד ASCII שבו המסר מופיע רק באותיות ראשונות של כל שורה.
- 2. שימוש בפורמטים דיגיטליים כגון PDF או EXE: הטמעת המידע בשדות מטא-נתונים, תגים מוסתרים, או חלקי קוד שאינם בשימוש.
- 3. שינוי סדרי ביטים LSB:
- שינוי הביטים הפחות משמעותיים LSB של קובצי מדיה (תמונות, וידאו, או אודיו) כדי להחביא את המידע.
- לדוגמה: המרה של פיקסל RGB כך שיכלול את המידע.
- 4. הטמעה בקבצים בינאריים: ניצול אזורים "לא פעילים" או אזורי Padding בקובץ בינארי להטמעת מידע.

סוגים מרכזיים של האפלה

- א. האפלת תוכן Content Obfuscation:
- שינוי המידע עצמו כך שהוא ייראה חסר משמעות:
- דוגמה: הצפנה בסיסית, שבה כל אות מוחלפת בערך מספרי, או בשיטות כמו Caesar Cipher או Base64.
- שימושים נפוצים: הטמעת מסרים בשדות נתונים או שדות מטא-נתונים.
- ב. האפלת מבנה Structural Obfuscation
- שינוי המבנה הפיזי או הלוגי שבו המידע מאורגן, מבלי לשנות את תוכנו:
- דוגמה: הטמעת המידע במיקומים בלתי צפויים בקובץ, כמו אזורי Padding בקובץ בינארי או בתגים מוסתרים בקובצי HTML או XML
- שימושים נפוצים: הסתרת מידע באזורים לא גלויים בקבצי PDF או ביצוע שינויים בקוד התכנות של תוכנה.
- ג. האפלת תפקוד Functional Obfuscation:
- שינוי התנהגות המערכת כך שמידע מוסתר יפעל רק בתנאים מסוימים.
- שימושים נפוצים: קובצי EXE שנטענים בתנאים ספציפיים בלבד.

טכניקות מתקדמות להאפלת מידע

- א. שיטות אפלה פשוטות:
- 1. שימוש בשיטות הצפנה קלות: הצפנה בעזרת Base64, XOR, או שינוי ערכי ASCII.
- 2. שינוי רווח Whitespace: הוספת רווחים, טאבים או שורות ריקות עם משמעות נסתרת.
- 3. החלפת תווים Character Substitution: החלפת תווים בטקסט באותיות או סמלים דומים (כמו "I" → "0", "O" → "1")
- ב. שיטות אפלה מורכבות:
- 1. הטמעה במדיה דיגיטלית:
- בקובצי תמונה: שינוי פיקסלים פחות משמעותיים LSB
- בקובצי אודיו: הוספת מידע בתדרים שאינם נשמעים לאוזן האנושית.
- בקובצי וידאו: שימוש בשינויים עדינים בפריימים בודדים.
- 2. שימוש בשכבות מידע מוסתרות:
- בקובצי PDF: הוספת מידע בשכבות בלתי נראות או בתגים מוסתרים.
- בקובצי Word: שימוש בגרסאות קודמות של המסמך Revision History
- 3. קידוד נתונים בצורה דינמית:
- המידע משתנה בהתאם למפתח סודי או למצב דינמי כלשהו.
- ג. טכניקות התקדמות בהנדסה לאחור:

- שינוי סדר פעולות בקוד: הסתרת לוגיקה מרכזית של קוד על ידי שינוי סדר הפעולות.
- השמטת חלקי קוד פעילים: החבאת קוד בקטעים שאינם נגישים ישירות.

יתרונות וחסרונות של האפלה

- יתרונות:
 1. הגנה על המידע: מונעת הבנה או גישה למידע רגיש.
 2. הסתרה פשוטה: מתבצעת לעיתים ללא צורך בשינויים גדולים בקובץ.
 3. עמידות בפני ניתוחים פשוטים: קשה לזהות מידע מוסתר ללא כלים מתקדמים.
- חסרונות:
 1. עמידות נמוכה מול פורניקה מתקדמת: כלים מתקדמים יכולים לחשוף את ההסתרה.
 2. רגישות לשינויים בקובץ: שינוי קל בקובץ (כמו דחיסה או עריכה) עלול לפגוע במידע המוסתר.
 3. בולטות בהצפנות לא סטנדרטיות: שימוש בשיטות לא מוכרות יכול לעורר חשד.

דוגמאות מעשיות להאפלה בקובצי PDF

- שימוש במטא-נתונים:
 - מידע מוסתר בשדות כמו Title , Author , /Keywords/ במטא-נתונים:
 - דוגמה: הטמעת ההודעה "12345" במטא-נתונים:


```
"metadata[/Keywords/] = "12345"
```
 - שימוש בשכבות בלתי נראות: הוספת טקסט בצבע לבן על רקע לבן, כך שהוא לא נראה לעין.
 - הטמעה בתוכן לא קריא: שימוש בקידוד Base64 לטקסט מוסתר והטמעתו בתוך התוכן.

תרגול עצמי (תמצית)

1. מה ההבדל בין סטגנוגרפיה לקריפטוגרפיה?
 2. מהם היתרונות והחסרונות של שימוש בסטגנוגרפיה?
 3. ציין לפחות שתי שיטות סטגנוגרפיה בקול.
1. קריפטוגרפיה מצפינה מידע אך אינה מסתירה את עצם קיומו; סטגנוגרפיה מסתירה את המידע כך שלא יתגלה שהוא קיים.
 2. יתרונות: שומר על דיסקרטיות, קשה לגילוי. חסרונות: מגבלות בכמות המידע שניתן להסתיר, רגישות לכלים לזיהוי.
 3. שיטות בקול: החבאת מידע בתדרים גבוהים, החבאה באמצעות שינוי אמפליטודה.
1. אם לפיקסל יש ערך RGB של (100, 150, 200) מה יהיה הערך החדש אם רוצים להסתיר את הביט 0?
 2. הסבר כיצד היית בודק אם תמונה מכילה מידע מוסתר.
1. (100, 150, 200) בבינארי: 01100100 10010110 11001000
 2. ניתוח סטטיסטי של התמונה ובדיקת שינויים חריגים בערכים, לדוגמא דחיסות מידע, אם יש קובץ מקורי ניתן לנהשוות ולגלות, ישנם כלים שבודקים זאת, במקרים פשוטים ניתן לקרוא את הביטים המשמעותיים.
1. מה היתרון של טכניקת LSB בתמונות
 2. כיצד ניתן להחביא מידע בתוך קובץ קול?
 3. למה קובצי BMP מתאימים במיוחד להסוואה?
1. כיצד ניתן להשתמש במרחב התדרים Frequency Domain להסוואה בקובץ JPEG?
 1. מהם המונחים Chaffing ו-Winnowing בטכניקה זו?
 2. מה ההבדל בין הטכניקה הזו לבין הצפנה?
 3. לשם מה נדרש המפתח הסודי?
 4. כיצד תוקף יכול לנסות לתקוף את המערכת, ומה מקשה עליו להצליח?

הרצאה 9 - אבטחת מידע וסייבר

הרצאה מס' 9

- מרצה: יניב מורדוב

נושאים כלליים

- קריפטוגרפיה
- הצפנה סימטרית
- צופן בלוקים
- צופן זרם
- AES
- DES
- תיבת תמורה
- תיבת החלפה
- הצפנה א-סימטרית
- שיתוף מפתחות
- PKI
- RSA
- צופן אל-גמאל
- NTRU
- הצפנת רבין
- צופן קריימר-שופ
- הצפנת מאקליס
- הצפנת התרמיל
- פרוטוקול דיפי הלמן
- עקרון קרקהופס
- פונקצית גיבוב
- מחולל פסבדו-רנדומלי קריפטוגרפי

מהי הצפנה סימטרית?

- הצפנה סימטרית היא שיטה להצפנה שבה נעשה שימוש במפתח אחד בלבד לצורך הצפנה וגם לצורך פענוח של המידע. כלומר, אותו מפתח משמש גם לנעול את המידע (הפיכתו לבלתי קריא) וגם לפתוח אותו (להחזירו לפורמט המקורי).

שלב 1: הצפנה

1. מהו טקסט קריא Plaintext?
 - זהו המידע המקורי שאנו רוצים להגן עליו. למשל:
 - הודעה טקסטואלית: "שלום, איך אתה?"
 - מסמך רגיש.
 - פרטי משתמשים, כמו סיסמאות או מספרי אשראי.
2. מפתח ההצפנה Key:
 - המפתח הוא רצף של ביטים (מספרים בינאריים) והוא זה שמאפשר את הפיכת הטקסט הקריא לטקסט מוצפן.
 - אורך המפתח (למשל, 128, 192 או 256 ביטים ב-AES) משפיע על רמת האבטחה: ככל שהמפתח ארוך יותר, כך קשה יותר לפרוץ את ההצפנה.
3. אלגוריתם ההצפנה:

- האלגוריתם הוא המתמטיקה שמאחורי ההצפנה. הוא מבצע סדרה מורכבת של פעולות על הטקסט הקריא והמפתח כדי ליצור טקסט מוצפן.
- דוגמה לפעולות:
 - חלוקת הטקסט לחלקים קטנים (נקראים בלוקים).
 - ערבוב של הנתונים.
 - החלפה של ביטים לפי כללים מסוימים.
 - שילוב המידע עם המפתח באמצעות פעולות מתמטיות.
- 4. טקסט מוצפן Ciphertext:
 - זהו הפלט של ההצפנה: גרסה בלתי קריאה של המידע המקורי.
 - לדוגמה, אם הטקסט המקורי היה "שלום, איך אתה?", לאחר הצפנה זה עשוי להיראות כמו: &7Fg5\$4n@Q3pZ!9
 - ללא המפתח והאלגוריתם המתאימים, אי אפשר לשחזר את המידע המקורי.

שלב 2: פענוח

- 1. שימוש באותו מפתח:
 - בתהליך הפענוח, הצד המקבל חייב להחזיק באותו מפתח ששימש להצפנה.
 - חשוב מאוד: המפתח חייב להישמר בסוד! אם מישהו מצליח להשיג אותו, הוא יכול לפענח את המידע.
- 2. אלגוריתם הפענוח:
 - זהו תהליך מתמטי הפוך שמבצע את הפעולות ההפוכות לאלו שבוצעו בהצפנה:
 - פירוק הטקסט המוצפן לבלוקים.
 - ביצוע הפעולות ההפוכות של ערבוב, החלפה ושילוב.
 - שחזור המידע המקורי.
- 3. התוצאה:
 - אם המפתח נכון ותהליך הפענוח בוצע בצורה מדויקת, נקבל את הטקסט המקורי בדיוק כפי שהוא היה לפני ההצפנה.

דוגמה פשוטה לתהליך הצפנה ופענוח

- נניח:
 - טקסט קריא Plaintext: "שלום"
 - מפתח: "12345"
 - אלגוריתם: XOR פשוט (חישוב מתמטי שמשווה ביטים).
 - הצפנה:
 - 1. הופכים את הטקסט הקריא לקוד בינארי: "שלום" = 10101111 10100110 10101011 10100011
 - 2. משלבים את המפתח (גם הוא בבינארי) עם הטקסט בעזרת XOR:
 - לדוגמה, אם המפתח הוא 11110000, נוצרת תוצאה חדשה של ביטים.
 - פענוח:
 - לוקחים את הטקסט המוצפן ומבצעים עליו XOR עם אותו מפתח בדיוק.
 - הפעולה ההפוכה משחזרת את הטקסט המקורי.

מפתח חשוב: ביטחון המפתח

- שמירת המפתח בסוד: אסור לשתף אותו בערוצים לא מאובטחים.
- חילופי מפתחות מאובטחים: כשמשתמשים בהצפנה סימטרית, יש צורך בשיטות בטוחות כדי להעביר את המפתח (למשל, באמצעות הצפנה אסימטרית כמו RSA)

מהו צופן בלוקים?

- צופן בלוקים הוא סוג של אלגוריתם להצפנה סימטרית שבו הטקסט הקריא `plaintext` מחולק ליחידות בגודל קבוע שנקראות בלוקים. כל בלוק מוצפן בנפרד באמצעות אותו אלגוריתם ומפתח. למשל, אם הטקסט הוא באורך 200 ביטים, ואלגוריתם ההצפנה עובד עם בלוקים בגודל 64 ביטים, הטקסט יחולק ל-3 בלוקים של 64 ביטים, כאשר היתרה תושלם באמצעות ריפוד `padding`.
- העבודה בבלוקים מאפשרת לשפר את האבטחה ולהקשות על זיהוי דפוסים בטקסט המקורי, מה שחשוב מאוד במניעת פריצות.

מושג הבלוק Block

- צופן בלוקים עובד על נתונים בגודל קבוע מראש, שנקרא בלוק.
 - לדוגמה: ב-AES הבלוק תמיד בגודל 128 ביטים.
- אם גודל הקלט (הטקסט הקריא) אינו מתחלק באופן מושלם בגודל הבלוק, נעשה שימוש בטכניקת ריפוד `Padding`.
- דוגמת ריפוד:
 - אם הבלוק הוא בגודל 16 בתים (128 ביטים), והטקסט הוא "שלום עולם" (10 בתים), נדרשים עוד 6 בתים כדי להשלים בלוק מלא. משתמשים בתווים מיוחדים או סכמה מוגדרת למילוי החסר.

איך זה עובד?

- צופן בלוקים עובד בשלושה שלבים עיקריים:
 1. חלוקה לבלוקים: הטקסט הקריא מחולק ליחידות בגודל קבוע. לדוגמה, אם הבלוק הוא בגודל 128 ביטים והטקסט הוא "HelloWorld" האלגוריתם יתאים אותו כך שיכיל בלוקים בגודל מתאים.
 2. הצפנה של כל בלוק: כל בלוק מוצפן בנפרד באמצעות המפתח, לפי סדרה של פעולות מתמטיות כמו XOR החלפות ושינויים בסדר הביטים.
 3. חיבור הבלוקים המוצפנים: כל הבלוקים מוצפנים בנפרד, והתוצאה היא רצף של בלוקים מוצפנים שמתחברים ליצירת הטקסט המוצפן `ciphertext`.

אלגוריתם צופן בלוקים

- אלגוריתם פופולרי מאוד שמיישם צופן בלוקים הוא AES (Advanced Encryption Standard).
- ב-AES הבלוק הוא בגודל קבוע של 128 ביטים, והמפתח יכול להיות באורך 128, 192 או 256 ביטים. האלגוריתם מבצע מספר סבבים `rounds` של פעולות כדי להצפין כל בלוק. ככל שהמפתח ארוך יותר, כך מספר הסבבים גדול יותר:
 - מפתח 128 ביטים → 10 סבבים.
 - מפתח 192 ביטים → 12 סבבים.
 - מפתח 256 ביטים → 14 סבבים.

מה קורה בכל סבב של צופן בלוקים?

- בכל סבב, האלגוריתם מבצע את הפעולות הבאות:
 1. ערבוב `Substitution`: כל חלק מהטקסט מוחלף לפי טבלה קבועה מראש שנקראת `S-box` קיצור של `Substitution Box`.
 2. שחלוף: `Permutation` שינויים במיקום של הביטים כדי לפזר את המידע ולהקשות על ניתוח.
 3. שילוב עם מפתח `Key Mixing`: כל בלוק מחובר עם המפתח באמצעות פעולת XOR.
 4. חוזר על הפעולות: הפעולות הללו חוזרות מספר פעמים בהתאם לאורך המפתח.
- לדוגמה:
 - הטקסט המקורי `plaintext` הוא HELLO
 - המפתח הוא KEY123
 - התוצאה אחרי מספר סבבים היא טקסט מוצפן שנראה כמו D12F3A

ריפוד Padding

- אם גודל הטקסט אינו מתחלק בגודל הבלוק, מוסיפים תווים מיותרים כדי להשלים את הבלוק האחרון. לדוגמה, אם גודל הבלוק הוא 128 ביטים, אבל הטקסט הוא באורך 200 ביטים:
- 128 ביטים → בלוק ראשון.
- 72 ביטים → בלוק שני → הוספת 56 ביטים של ריפוד כדי להשלים ל-128 ביטים.

מצבי פעולה של צופן בלוקים

- כשמשתמשים בצופן בלוקים, יש דרכים שונות לטפל בבלוקים כדי להבטיח אבטחה גבוהה יותר. הנה המצבים הנפוצים:
- 1. ECB (Electronic Codebook):
 - כל בלוק מוצפן בנפרד.
 - חסרון: אם יש בלוקים זהים בטקסט המקורי, הם יישארו זהים גם בטקסט המוצפן, מה שמאפשר לזהות דפוסים.
- 2. CBC (Cipher Block Chaining):
 - כל בלוק מוצפן תוך שימוש בבלוק הקודם, כך שאותו בלוק בטקסט המקורי יפיק בלוק מוצפן שונה.
 - מצריך וקטור יזום Initialization Vector - IV כדי להתחיל את התהליך.
- 3. CTR (Counter Mode): הופך את צופן הבלוקים לצופן זרם, מהיר וגמיש לשימוש.
- 4. GCM (Galois/Counter Mode):
 - משלב הצפנה עם אימות נתונים Authentication
 - נפוץ מאוד בפרוטוקולים מאובטחים כמו TLS

מבנה פנימי של צופן בלוקים

- צופן בלוקים מבוסס לרוב על אחד משני מבנים עיקריים:
- 1. רשת פייסטל Feistel Network:
 - מחלק את הבלוק לשני חלקים Left ו-Right
 - מבצע הצפנה רק על חלק אחד בכל סבב, ומשלב את התוצאה בחלק השני.
 - קל יחסית למימוש, ולכן נפוץ באלגוריתמים ישנים יותר, כמו DES
- 2. רשת החלפה-שחלוף Substitution-Permutation Network:
 - מבצע סדרת החלפות ושחלופים על כל הבלוק.
 - נפוץ באלגוריתמים מודרניים, כמו AES

חוזק אבטחה של צופן בלוקים

- פרמטרים שמשפיעים על רמת האבטחה:
- 1. אורך המפתח:
 - ככל שאורך המפתח ארוך יותר, כך קשה יותר לפרוץ את ההצפנה.
 - לדוגמה:
 - מפתח באורך 128 ביטים → דורש 2^{128} ניסיונות במתקפת כוח גס.
 - מפתח באורך 256 ביטים → דורש 2^{256} ניסיונות.
- 2. עמידות למתקפות:
 - מתקפת כוח גס Brute Force: ניסיון לבדוק את כל המפתחות האפשריים.
 - ניתוח דפוסים: Pattern Analysis ניסיון לזהות דפוסים בטקסט המוצפן.
- 3. מניעת זליגת מידע:
 - השימוש במצבי פעולה מתקדמים כמו CBC או GCM מגן על המידע ומונע דליפת דפוסים.

דוגמה מוחשית

- נניח שאתה רוצה להצפין את המשפט הבא: "שלום עולם"

- שלב 1: המרת הטקסט לייצוג בינארי (לדוגמה, UTF-8)
- שלב 2: חלוקה לבלוקים בגודל 128 ביטים.
- שלב 3: ביצוע פעולות מתמטיות על כל בלוק בנפרד לפי האלגוריתם (AES לדוגמה).
- תוצאה: הטקסט המוצפן יהיה מחרוזת של תווים או מספרים שנראים אקראיים, כמו 7d1e9a4bcf

יתרונות וחסרונות של צופן בלוקים

- יתרונות:
 1. אבטחה חזקה: מתאים להצפנת נתונים רגישים.
 2. עמידות גבוהה למתקפות: בזכות סבבים מרובים ומבנים מורכבים.
 3. תמיכה במגוון מצבי פעולה: מאפשר גמישות בהתאמה ליישומים שונים.
- חסרונות:
 1. תלות בגודל הבלוק: אם גודל הבלוק לא מתאים, נדרשת הוספת ריפוד.
 2. רגישות למצב פעולה לא נכון: בחירה שגויה של מצב פעולה עלולה לחשוף מידע.

שימושים נפוצים בצופן בלוקים

- פרוטוקולי רשת: TLS, SSH, Ipsec
- הצפנת מסדי נתונים: אבטחת מידע רגיש.
- הצפנת דיסקים: BitLocker, FileVault
- מערכות אחסון בענן: הגנה על נתוני משתמשים.

חשוב לזכור למבחן הגמר שלכם ב16/3

- צופן בלוקים הוא שיטה להצפנה סימטרית שבה עובדים עם יחידות בגודל קבוע.
- הוא משתמש במבנים מתמטיים מורכבים כדי להבטיח אבטחה.
- מצבי פעולה כמו CBC ו-GCM מונעים דליפת מידע.
- אלגוריתמים מודרניים כמו AES הם סטנדרטיים ונפוצים במגוון מערכות.

מבוא לצופן זרם

- צופן זרם הוא סוג של הצפנה סימטרית שבו המידע מוצפן בזרימה רציפה של ביטים או בתים (ולא בבלוקים קבועים כמו בצופן בלוקים). במקום לחלק את הנתונים לבלוקים, צופן זרם משתמש במחולל מפתח Keystream Generator שמייצר זרם ביטים אקראי Keystream ואז משלב אותו עם הנתונים בטכניקת XOR.

דוגמה פשוטה לצופן זרם

- נניח שיש לנו את הטקסט הקריא plaintext:
- HELLO
- והמחולל מייצר את זרם המפתח keystream:
- XMCKL
- אם נבצע XOR בין כל אות בטקסט לבין האות המתאימה בזרם המפתח, נקבל את הטקסט המוצפן. למשל:
 - $H \oplus X =$ טקסט מוצפן
 - $E \oplus M =$ טקסט מוצפן
 - $L \oplus C =$ טקסט מוצפן
 - $L \oplus K =$ טקסט מוצפן
 - $O \oplus L =$ טקסט מוצפן
- הפענוח מתבצע באותה דרך, על ידי חיבור הזרם המוצפן שוב עם אותו keystream

איך פועל צופן זרם?

- 1. יצירת זרם המפתח Keystream Generation

- מחולל זרם המפתח מייצר רצף ביטים פסאודו-אקראי באותו אורך של הטקסט המקורי.
- כל ביט בטקסט הקריא מוצפן באופן נפרד, מה שמקנה גמישות בצופן.
- זרם המפתח לעולם לא חוזר על עצמו, אחרת תוקף יוכל לשחזר את הנתונים.
- 2. שימוש ב-XOR להצפנה ופענוח
- הצפנה:
- $Ciphertext = Plaintext \oplus Keystream$
- פענוח (היפוך ההצפנה):
- $Plaintext = Ciphertext \oplus Keystream$
- מכיוון ש-XOR היא פעולה הפיכה, ניתן לקבל חזרה את המידע המקורי אם זרם המפתח ידוע.

צופן זרם אסינכרוני / תלוי מצב

- כל ביט מוצפן מושפע מהביטים שהוצפנו לפניו.
- הטקסט המוצפן נשמר במצב פנימי של האלגוריתם, ולכן ניתן לשחזר את הסנכרון במקרה של שיבוש.
- דוגמה: OFB (Output Feedback Mode), CFB (Cipher Feedback Mode)
- יתרון:
- אם חלק מהטקסט המוצפן משתבש, ההצפנה תחזור לפעול נכון אחרי כמה ביטים.
- חיסרון:
- רמת האבטחה תלויה ברגישות המצב הפנימי של המחולל.

מחוללי מספרים פסאודו-אקראיים PRNG בצופן זרם

- מחוללי מספרים פסאודו-אקראיים PRNG בצופן זרם
- PRNG (Pseudorandom Number Generator) הוא רכיב קריטי בצופן זרם, המשמש להפקת זרם מפתח חזק. אם המחולל אינו אקראי מספיק, ניתן לפרוץ את ההצפנה.
- ♦ דוגמאות למחוללי מספרים פסאודו-אקראיים נפוצים:
- 1. LFSR (Linear Feedback Shift Register): נפוץ בצפנים ישנים, אך ניתן לחיזוי בקלות.
- 2. RC4 PRNG: היה בשימוש ב-WEP אך יש בו חולשות קריפטוגרפיות.
- 3. ChaCha PRNG: בטוח ומהיר, נמצא בשימוש ב-TLS 1.3 וב-SSH.
- בעיות של PRNG חלש: אם PRNG יוצר keystream שחוזר על עצמו (מחזוריות), התוקף יכול לנחש את המפתח ולפענח את ההצפנה!

מתקפות על צופן זרם

- מתקפת שימוש חוזר של keystream
- אם שני מסרים שונים מוצפנים עם אותו keystream ניתן לבצע XOR ביניהם ולקבל:
- $C1 \oplus C2 = P1 \oplus P2$ התוצאה היא XOR בין שני הטקסטים הקריאים, מה שמאפשר ניתוח סטטיסטי ושחזור המידע.
- דוגמה היסטורית: במהלך מלחמת העולם השנייה, הסובייטים השתמשו ב-One-Time Pad (OTP) בצורה לא נכונה (שימוש חוזר ב-keystream ואנשי הקריפטוגרפיה האמריקאים הצליחו לפענח הודעות סודיות).

מה הבעיה עם שימוש חוזר ב-keystream בצופן זרם?

- נזכור שצופן זרם מצפין כל ביט בנפרד על ידי XOR עם זרם מפתח keystream
- אם משתמשים באותו זרם מפתח כדי להצפין שני מסרים שונים, אפשר לגלות מידע חשוב עליהם!
- איך זה עובד? בוא נפרק את זה שלב אחרי שלב:
- 1. יש לנו שני מסרים שונים:
- מסר 1: P1
- מסר 2: P2
- 2. הצפנה בצופן זרם מתבצעת כך:

- מצפינים את המסר הראשון: $C1 = P1 \oplus \text{Keystream}$
- מצפינים את המסר השני עם אותו keystream וזו הטעות הקריטית! $C2 = P2 \oplus \text{Keystream}$
- 3. עכשיו, אם תוקף מצליח להשיג C1-C2 הוא יכול לחשב: $C1 \oplus C2 = (P1 \oplus \text{Keystream}) \oplus (P2 \oplus \text{Keystream})$ עם אותו ערך פעמיים מבטל אותו, נשאר לנו: $P1 \oplus P2$ כלומר, התוקף קיבל את ה-XOR בין שני הטקסטים הקריאים P1 ו-P2
- 4. מכאן, אם המסרים מספיק צפויים (למשל, הודעות טקסט באנגלית או קודים בינאריים), אפשר לנחש חלקים מהמסרים, להשלים מילים נפוצות, ולהצליח לשחזר את המסרים המקוריים.

השוואה בין צופן זרם לצופן בלוקים

- אם צריך אבטחה גבוהה יותר – משתמשים בצופן בלוקים. אם צריך מהירות גבוהה – צופן זרם עדיף.

יתרונות וחסרונות של צופן זרם

- יתרונות:

- 1. מהירות גבוהה – ההצפנה מתבצעת ביט אחר ביט, ולכן היא מהירה מאוד.
- 2. מתאים להצפנת תקשורת בזמן אמת – משמש בפרוטוקולים כמו Bluetooth, Wi-Fi (WEP), ו-TLS
- 3. חסכוני בזיכרון – אין צורך לאחסן בלוקים של נתונים, מה שהופך אותו ליעיל במכשירים קטנים כמו IoT

- חסרונות:

- 1. רגישות לשימוש חוזר של keystream – אם משתמשים באותו זרם מפתח לשני מסרים, קל לחשוף את המידע.
- 2. פחות עמיד בפני מתקפות מסוימות – דורש ייצור זרם אקראי מאוד חזק.
- 3. אם זרם המפתח אינו באמת אקראי, אפשר לפצח את ההצפנה.

דוגמאות מעשיות

- 1. דוגמה עם RC4 פשוטה
- נניח שיש לנו טקסט: HELLO
- ונבחר מפתח "KEY = SECRET" המחולל של RC4 ייצור זרם מפתח (Keystream) שנשתמש בו כדי לבצע XOR מול הטקסט המקורי.
- Plaintext: HELLO
- Keystream: XMCKL
- Ciphertext: ? ? ? ? ?
- (התוצאה תהיה טקסט מוצפן, ולא ניתן לקריאה בלי המפתח).
- 2. דוגמה לשבירת צופן זרם עם שימוש חוזר בזרם המפתח
- נניח ששני מסרים שונים מוצפנים עם אותו זרם מפתח:
- $C1 = P1 \oplus \text{Keystream}$
- $C2 = P2 \oplus \text{Keystream}$
- אם תוקף יודע C1 ו-C2 הוא יכול לבצע:
- $C1 \oplus C2 = (P1 \oplus \text{Keystream}) \oplus (P2 \oplus \text{Keystream})$
- $P1 \oplus P2 =$
- כלומר, ה-keystream מתבטל, והתוקף יכול לזהות דפוסים בין שני המסרים! זו הסיבה שאסור להשתמש באותו keystream פעמיים!

דוגמה פשוטה: הצפנת שתי מילים עם אותו keystream

- נניח שאנחנו מצפינים את שתי המילים HELLO ו-WORLD עם אותו keystream
- 1. יצירת ההצפנה:
- $P1 (\text{HELLO}) = 01001000 \ 01000101 \ 01001100 \ 01001100 \ 01001111$
- $P2 (\text{WORLD}) = 01010111 \ 01001111 \ 01010010 \ 01001100 \ 01000100$
- $\text{Keystream} = 10111001 \ 11001011 \ 10110100 \ 01110110 \ 11101100$

- נחשב את ה-C1 וה-C2:
- $C1 = P1 \oplus \text{Keystream} = 11110001\ 10001110\ 11111000\ 00111010\ 10100011$
- $C2 = P2 \oplus \text{Keystream} = 11101110\ 10000100\ 11100110\ 00111010\ 00001000$
- 2. מתקפה: חישוב XOR בין C1 ו-C2:
- $C1 \oplus C2 = (P1 \oplus \text{Keystream}) \oplus (P2 \oplus \text{Keystream})$
- $P1 \oplus P2 =$
- וזה ייתן לנו:
- $P1 \oplus P2 = 00011111\ 00001010\ 00011110\ 00000000\ 10101011$

הצפנת AES – Advanced Encryption Standard

- הצפנת AES ראשי תיבות של Advanced Encryption Standard היא אחת משיטות ההצפנה החשובות והנפוצות ביותר בעולם. היא משמשת לאבטחת מידע רגיש, כולל נתונים ממשלתיים, עסקיים ופרטיים.
- רקע היסטורי
- AES נבחר כסטנדרט ההצפנה הרשמי של ממשלת ארה"ב בשנת 2001 על ידי ה-NIST המכון הלאומי לתקנים וטכנולוגיה). הוא החליף את תקן DES שהיה כבר לא בטוח בגלל אורכו הקצר של המפתח (56 ביטים). AES מבוסס על האלגוריתם Rijndael שפותח על ידי שני קריפטוגרפים בלגיים: Joan Daemen ו-Vincent Rijmen.

מאפייני AES

- צופן בלוקים: Block Cipher מצפין מידע בבלוקים בגודל 128 ביטים. מפתח משתנה: תומך באורכי מפתח של 128, 192, ו-256 ביטים. מספר סבבים: Rounds מספר הסבבים תלוי באורך המפתח:
- 10 – AES-128 סבבים
- 12 – AES-192 סבבים
- 14 – AES-256 סבבים
- בטיחות גבוהה – נחשב בטוח לשימוש ומאוד עמיד בפני התקפות קריפטוגרפיות.

איך AES עובד? - שלבי ההצפנה

- AES עובד לפי עקרון של תחליפים וטרנספוזיציות Substitution-Permutation Network ההצפנה מתבצעת בסבבים Rounds ובכל סבב יש ארבעה שלבים עיקריים:
- 1. שלב ראשון – SubBytes תיבות תחליף:
- כל בית Byte במטריצה מוחלף בערך אחר לפי טבלה שנקראת S-Box. זה יוצר בלבול Confusion במידע, כך שאי אפשר לנחש את הקלט מהפלט.
- 2. שלב שני – ShiftRows הזזת שורות:
- כל שורה במטריצת הבלוק מוזזת מספר צעדים שמאלה. השורה הראשונה לא זזה, השנייה מוזזת מקום אחד, השלישית שניים, והרביעית שלושה מקומות. זה יוצר פיזור Diffusion של הנתונים בתוך הבלוק.
- 3. שלב שלישי – MixColumns ערבוב עמודות:
- כל עמודה במטריצה מעורבבת עם עצמה באמצעות חישובים מתמטיים על מספרים בגוף גלואה Galois Field השלב הזה מוסיף פיזור נוסף במידע, כך שאם שינוי קטן בנתונים יגרום לשינוי משמעותי בפלט. שלב זה לא מתבצע בסבב האחרון, כדי לאפשר תהליך פענוח תקין.
- 4. שלב רביעי – AddRoundKey הוספת מפתח סבב:
- מבצעים XOR בין הבלוק הנוכחי לבין מפתח הסבב שהופק מאלגוריתם הרחבת המפתח Key Expansion זה מוסיף ביטחון, מכיוון שכל סבב משתמש במפתח אחר.
- כל ארבעת השלבים חוזרים על עצמם עד לסבב האחרון, שבו לא מתבצע MixColumns.

דוגמה מספרית לתהליך AES

- ניקח בלוק טקסט לדוגמה:
- "HELLO123WORLD456"

- נתרגם אותו לערכים בינאריים ונצפין בעזרת AES-128 עם מפתח הצפנה. בכל סבב, התוכן של הבלוק ישתנה בעקבות ארבעת השלבים שהסברנו. לבסוף, נקבל טקסט מוצפן בלתי קריא.
- כדי לפענח את ההודעה, מבצעים את אותם השלבים בסדר הפוך, עם אותם המפתחות.

הרחבה על שלבי AES

1. SubBytes תחליף בתים S-Box
 - בכל סבב, כל בית Byte מוחלף בערך חדש מתוך טבלה קבועה שנקראת S-Box תיבת תחליף. מטרת השלב היא למנוע קשר ישיר בין הקלט לפלט, כדי למנוע ניתוח סטטיסטי של ההצפנה. ה-S-Box נבנה כך שיהיה עמיד בפני התקפות מתמטיות.
 - דוגמה לתיבת S-Box חלקית:
 - 2. ShiftRows הזזת שורות:
 - מטרת שלב זה היא לפזר מידע על פני הבלוק.
 - השורה הראשונה נשארת במקום, השנייה מוזזת שמאלה במקום אחד, השלישית בשני מקומות, והרביעית בשלושה מקומות.
 - פעולה זו מונעת קשר ישיר בין בתים סמוכים בתוך הבלוק.
 - דוגמה להזזת שורות
- | | | | |
|---|---|---|---|
| A | B | C | D |
| E | F | G | H |
| I | J | K | L |
| M | N | O | P |
- לפני ההזזה:
- | | | | |
|---|---|---|---|
| A | B | C | D |
| E | F | G | H |
| I | J | K | L |
| M | N | O | P |
- אחרי ההזזה:
- | | | | |
|---|---|---|---|
| F | G | H | E |
| K | L | I | J |
| P | M | N | O |
3. MixColumns ערבוב עמודות:
 - בשלב זה, כל עמודה של המטריצה מוכפלת במטריצה קבועה מעל שדה גלואה 2^8 Galois Field.
 - מטרת הפעולה היא להגביר את הפיזור של המידע, כך ששינוי קטן בקלט ישפיע על כל הביטים של הפלט.
 - לא מבוצע בסבב האחרון, כדי להקל על תהליך הפענוח.
 - חישוב דוגמתי של MixColumns
 - אם ניקח עמודה במטריצה:
- | | | | |
|---|---|---|---|
| A | B | C | D |
|---|---|---|---|
- נכפיל אותה במטריצה קבועה:
- | | | | |
|---|---|---|---|
| 1 | 1 | 3 | 2 |
| 1 | 3 | 2 | 1 |
| 3 | 2 | 1 | 1 |
| 2 | 1 | 1 | 3 |
- ונקבל עמודה חדשה מוצפנת.
4. AddRoundKey הוספת מפתח סבב:
 - בשלב זה, מבצעים XOR בין הבלוק הנוכחי לבין מפתח הסבב שהופק מהתהליך של הרחבת מפתחות.
 - הפעולה הזאת קושרת את כל שלבי ההצפנה למפתח ומבטיחה שכל שינוי במפתח ישפיע על כל הביטים של ההודעה.
 - מפתח חדש נוצר לכל סבב, בעזרת פונקציית הרחבת המפתחות Key Expansion.

הרחבת מפתחות Key Expansion

- כדי שכל סבב ישתמש במפתח אחר AES יוצר סדרת מפתחות מתוך המפתח המקורי. התהליך הזה מבוצע ע"י:
- חלוקת המפתח הראשוני למילים Words של 32 ביטים.

- שימוש בפונקציה RotWord שמסובבת בתים במפתח.
- שימוש בפונקציה SubWord שמחליפה כל בית בעזרת S-Box.
- שילוב ערכים קבועים הנקראים RCON (Round Constants).
- חיבור XOR עם המפתחות הקודמים.
- תהליך זה מונע קשר ישיר בין סבבי ההצפנה. משפר את עמידות האלגוריתם בפני התקפות קריפטוגרפיות.

התקפות אפשריות על AES

- למרות שאי אפשר לשבור את AES עם Brute-Force ישנן התקפות מתקדמות שמנסות לנצל חולשות אלגוריתמיות או טעויות ביישום.
- התקפות עיקריות על AES
- התקפת ערוץ צדדי Side-Channel Attack:
- במקום לנסות לפענח ישירות את AES תוקפים מודדים צריכת חשמל, זמן ריצה, ופליטת קרינה כדי להסיק מידע על המפתח.
- התקפת טבלת חיפוש Lookup Table Attack:
- אם AES מיושם בצורה לא בטוחה (למשל באמצעות טבלאות חיפוש בזיכרון), תוקף יכול לנחש חלקים מהמפתח בעזרת גישה לזיכרון המטמון.
- התקפת מפתחות חלשים Related-Key Attack:
- תוקפים מנסים למצוא קשרים מתמטיים בין מפתחות שונים כדי להפחית את המורכבות של שבירת ההצפנה.

למה AES נחשב בטוח?

- אין שום התקפה יעילה ששוברת את AES-128, AES-192, או AES-256 בזמן מעשי.
- אפילו עם מחשבים קוונטיים, AES-256 יישאר מאובטח מאוד בגלל אורך המפתח הארוך.
- הוא סטנדרט גלובלי ונמצא בשימוש בכל התעשיות – בנקים, ממשלות, VPNs, WiFi ועוד.
- יוצא מכל האמור הלכה למעשה – נקודות לבחינת גמר:
- הוא מהיר ויעיל לעיבוד מידע גדול.
- הוא מאובטח מאוד נגד התקפות קריפטוגרפיות.
- הוא נמצא כמעט בכל מערכת אבטחת מידע מודרנית.
- הוא ניתן למימוש בחומרה ובתוכנה בקלות.
- AES ***** הוא עמוד השדרה של אבטחת מידע בעולם הדיגיטלי.

מה זה DES ולמה הוא פותח?

- DES פותח בשנות ה-70 על ידי IBM ואומץ על ידי הממשל האמריקאי בשנת 1977 כתקן להצפנה של מידע רגיש.
- הוא צופן בלוקים שפועל על 64 ביטים בכל בלוק
- הוא סימטרי – אותו מפתח משמש להצפנה ולפענוח
- הוא משתמש במפתח של 56 ביטים (המקור הוא 64 ביטים, אך 8 ביטים משמשים לבדיקה)
- למה DES כבר לא מספיק מאובטח?
- אורך המפתח קצר מדי – ניתן לשבור אותו בכוח גס Brute-Force תוך מספר שעות.
- התקפות קריפטוגרפיות חדשות (למשל התקפת "התקפה דיפרנציאלית") מצאו חולשות במבנה שלו.
- מה החליף אותו? AES: הצפנה מודרנית עם מפתחות של 128, 192 או 256 ביטים. Triple DES (3DES) גרסה מאובטחת יותר המשתמשת ב-DES שלוש פעמים ברצף.

איך עובד DES?

- DES מבוסס על מבנה פייסטל Feistel Network שבו כל בלוק מידע עובר 16 סבבים של הצפנה. המבנה מאפשר להצפין ולפענח באותו אלגוריתם – פשוט מריצים את הסבבים הפוך.
- שלבי ההצפנה ב-DES:
- Permutation ראשונית IP : ערבוב הביטים הראשוני של הבלוק.
- 16 סבבי הצפנה – בכל סבב:

- מחלקים את הבלוק ל-2 חצאים L ו-R
- מחילים פונקציה קריפטוגרפית F על החצי הימני ומבצעים XOR עם החצי השמאלי.
- מחליפים בין החצאים.
- Permutation סופית FP : ערבוב נוסף לקבלת התוצאה הסופית.

הרחבה על שלבי ההצפנה

1. שלב ה-Initial Permutation (IP)
 - הבלוק של 64 ביטים מעורבב בעזרת טבלה קבועה IP Table
 - אין כאן הצפנה אמיתית – רק סידור מחדש של הביטים כדי להפחית קשר ישיר בין הקלט לפלט.
2. פיצול ל-2 חצאים L ו-R
 - מחלקים את הבלוק ל-32 ביטים שמאליים L ו-32 ביטים ימניים R בכל סבב:
 - החלק הימני R עובר עיבוד בפונקציה F עם מפתח משנה Subkey
 - מבצעים XOR עם החלק השמאלי L
 - מחליפים בין החלקים לפני המעבר לסבב הבא.
3. פונקציית ההצפנה F
 - פונקציה זו היא לב האלגוריתם ומבצעת את עיקר ההצפנה.
 - הפונקציה מקבלת:
 - 32 ביטים מהחצי הימני R
 - 48 ביטים של מפתח משנה Subkey שלבי החישוב:
 - הרחבת R ל-48 ביטים Expansion Function E
 - ביצוע XOR עם מפתח המשנה
 - מעבר דרך S-Boxes תיבות החלפה - Substitution Boxes
 - Permutation נוספת לפיזור הביטים
 - 5. S-Boxes התיבות הקריפטוגרפיות החשובות:
 - ישנן 8 תיבות S-Box שכל אחת ממירה 6 ביטים ל-4 ביטים לפי טבלה קבועה.
 - מטרת ה-S-Boxes :
 - לגרום לכך שהפלט יהיה בלתי ניתן לחיזוי אפילו אם יש שינוי קטן בקלט.
 - להקשות על התקפות סטטיסטיות (למשל התקפה לינארית).
6. סיום ההצפנה – Final Permutation (FP)
 - לאחר 16 סבבי הצפנה, הביטים מעורבבים שוב לפי טבלת FP ההפוכה ל-IP
 - מתקבל 64 ביטים של טקסט מוצפן.

פענוח ב-DES

- אותו תהליך כמו בהצפנה – רק בסדר הפוך!
- פשוט מפעילים את 16 הסבבים בסדר ההפוך עם אותם מפתחות משנה.

דוגמה מעשית לאופן פעולתו של DES

- כדי להמחיש את פעולתו של DES בצורה מעשית אך ללא קוד, ניקח דוגמה של טקסט קריא Plaintext קצר ונדגים איך הוא עובר את שלבי ההצפנה.
- נתונים להתחלה
- טקסט קריא Plaintext: HELLO123
- ייצוג בינארי של הטקסט ASCII:
- H -> 01001000
- E -> 01000101
- L -> 01001100
- O -> 01001111

- 1 -> 00110001
- 2 -> 00110010
- 3 -> 00110011
- סה"כ בלוק שלם של DES
- מפתח הצפנה 56 – Key ביטים:
- 10010111 10101010 11010010 01001100 10101011 11001010 01100101
- שלבי ההצפנה ב-DES
- שלב 1: Permutation ראשונית IP:
- הביטים של ה-plaintext מעורבים מחדש לפי סדר מוגדר מראש IP Table
- לדוגמה, אם סדר הביטים במקור היה:
- 00101101 11001100 01100110 10011001 11001100 01010101 00110010 10101011
- אז אחרי ה-IP:
- 10011001 11001100 01010101 00101101 00110010 01100110 10101011 11001100
- שלב 2: פיצול לשני חצאים
- לאחר ה-IP מחלקים את הבלוק לשניים:
- L 32 ביטים ראשונים
- R 32 ביטים אחרונים
- לפני הפיצול:
- אחרי הפיצול:
- L0 = 11001100 10101011 01100110 00110010
- R0 = 00101101 01010101 11001100 10011001
- שלב 3: 16 סבבי הצפנה
- כל אחד מ-16 הסבבים מבצע את הפעולות הבאות:
- שימוש במפתח משנה Subkey שהופק מהמפתח הראשי.
- הרחבת R ל-48 ביטים Expansion Function
- ביצוע XOR עם מפתח המשנה.
- מעבר דרך תיבות S-Box החלפה לא לינארית של ביטים.
- Permutation נוספת (פיזור הביטים).
- XOR עם L הקודם כדי ליצור את R החדש.
- החלפת ערכים R הופך להיות L הבא, והתוצאה של השלב הופכת להיות R החדש.
- דוגמה לסבב ראשון:
- L1 = R0
- R1 = L0 \oplus F(R0, Subkey1)
- במהלך הסבבים הביטים משתנים לחלוטין בזכות הפעולות הלא לינאריות.
- שלב 4: החזרת שני החצאים יחד
- לאחר 16 סבבי הצפנה, מחברים שוב את שני החלקים בסדר הפוך:
- Ciphertext = R16 + L16
- שלב 5: Permutation סופית FP
- לבסוף, מבצעים ערבוב ביטים נוסף FP כדי לקבל את הטקסט המוצפן הסופי.
- דוגמה לתוצאה הסופית Ciphertext:
- אם נקבל למשל את הבלוק הבא אחרי ההצפנה:
- 10100101 01011001 00101111 10011100 01010101 11001110 01101001 10110011
- זהו Ciphertext – טקסט מוצפן לחלוטין שלא ניתן להבין ללא המפתח.
- פענוח DES עובד באותו אופן, רק בסדר הפוך!
- יוצא לנו מספר נקודות שצריך לזכור למבחן, והם:
- כל בלוק של 64 ביטים מוצפן בנפרד.
- השינויים הלא לינאריים Permutations, XOR, S-Boxes מבטיחים שהתוצאה תהיה בלתי ניתנת לניחוש.

- הצפנה עם אותו מפתח תמיד תוביל לאותה תוצאה אם אין IV.

התקפות ידועות על DES

- Brute Force Attack: בשל אורך המפתח הקצר (56 ביטים), ניתן לנסות את כל האפשרויות.
- Cryptanalytic Attacks: התקפות מתמטיות כמו התקפה דיפרנציאלית או התקפה לינארית מנצלות חולשות במבנה ההצפנה.
- Meet-in-the-Middle Attack על DES 2 – התקפה שמקטינה את הזמן הנדרש לשבירת ההצפנה.

Triple DES (3DES) גרסה בטוחה יותר

- כדי להתגבר על הבעיות של DES פותח DES 3 שמשמש בשלוש ריצות של DES ברצף:
 1. הצפנה עם מפתח ראשון
 2. פענוח עם מפתח שני
 3. הצפנה עם מפתח שלישי
- אורך המפתח הכולל ב- 3DES: 112 ביטים (אם שני מפתחות שונים)
- 168 ביטים (אם שלושה מפתחות שונים)
- 3DES בטוח יותר מ-DES אך עדיין איטי יותר מ-AES ולכן כיום כמעט לא משתמשים בו.

מהו תיבת תמורה

- בקריפטוגרפיה, תיבת תמורה P-Box היא מנגנון שבו מסדרים מחדש את הביטים של המידע המוצפן לפי סדר קבוע מראש. המטרה של התיבה היא לערבב את המידע כדי להגביר את הDiffusion – כלומר, לפזר את ההשפעה של כל ביט של המידע המקורי על כמה שיותר ביטים של הבלוק הסופי.
- תיבות תמורה נמצאות בצופן בלוקים, למשל באלגוריתמים כמו DES ו-AES.

סוגים של תיבות תמורה

1. תיבת תמורה רגילה Straight P-Box מחליפה את מיקומי הביטים ללא שינוי בכמותם. דוגמה: אם הקלט הוא 10110011 והתמורה היא [3, 1, 4, 2, 5, 7, 8, 6], אז הפלט יהיה 11001011.
2. תיבת תמורה מצמצמת Compression P-Box לוקחת מספר ביטים מהקלט ומפחיתה את כמותם בפלט. שימוש: צמצום גודל בלוקים מסוימים בהצפנה כדי לחזק ביטחון.
3. תיבת תמורה מרחיבה Expansion P-Box מגדילה את מספר הביטים ביצירת עותקים שלהם, כך שהפלט גדול מהקלט. שימוש: הרחבת גודל בלוקי ביניים בהצפנה כדי להגביר ביטחון. לדוגמה: ב-DES הבלוק של 32 ביטים מורחב ל-48 ביטים לפני השימוש ב-S-Box.

איך תיבת תמורה פועלת?

- תיבת תמורה לוקחת קלט של מספר ביטים, מערבבת אותם לפי כלל ידוע, ומוציאה פלט מסודר מחדש. דוגמה פשוטה:
 - נניח שיש לנו בלוק של 8 ביטים בקלט: 0 0 1 0 1 1 0 1 (קלט)
 - כעת נשתמש בתיבת תמורה עם הסדר הבא: [3, 8, 2, 5, 1, 6, 4, 7]
 - כלומר, המיקום החדש של הביטים יהיה לפי סדר זה:
 - הביט השלישי יעבור למקום הראשון,
 - הביט השמיני יעבור למקום השני,
 - הביט השני יעבור למקום השלישי, וכך הלאה...
 - פלט לאחר תמורה: 0 1 1 1 0 0 0 1 (פלט)
- עכשיו הביטים ערבבו מקומות, מה שמקשה על תוקף להבין את המידע המקורי.

תיבת תמורה ב-DES

- באלגוריתם DES ישנן שתי תיבות תמורה עיקריות:
 1. תמורה ראשונית IP – Initial Permutation:
 - מתבצעת על ה-plaintext לפני תחילת סבבי ההצפנה.
 - חשובה כדי לערבב את הביטים ולמנוע זליגת מידע.
 2. תמורה סופית FP – Final Permutation:
 - משמשת לשחזור הסדר לאחר 16 הסבבים.
 - משחזרת סדר קבוע לפני הוצאת ה-ciphertext.
- בנוסף, בתוך כל סבב ב-DES יש תיבת תמורה נוספת שמסדרת מחדש את הביטים לאחר שלב S-Boxes כדי לפזר את האפקט של הביטים המעובדים.

תיבות תמורה בצופן AES

- בניגוד ל-DES ב-AES אין תיבות תמורה קלאסיות P-Box אלא שלב בשם MixColumns. MixColumns מבצע טרנספורמציה מתמטית שמערבבת עמודות של מטריצה במקום להחליף ביטים ספציפיים.
- מטרת MixColumns דומה למטרת P-Box ב-DES – להפיץ את ההשפעה של כל ביט על כל הבלוק כולו.

השפעה של סוג תיבת התמורה על קריפטואנליזה (ניתוח הצפנה)

- P-Box חשוב במיוחד נגד מתקפות מבוססות ניתוח סטטיסטי. ללא P-Box חזק, מתקפת גלוי נבחרת Chosen Plaintext Attack - CPA יכולה לחשוף דפוסים קבועים במידע המוצפן ולנחש חלקים מהמפתח.
- לדוגמה: אם היינו מצפינים מסרים באורך קבוע עם P-Box חלשה או ללא P-Box תוקף היה יכול לראות שחלקים מסוימים במסרים שונים נותנים תמיד את אותה תוצאה אחרי הצפנה.

חשיבות תיבת תמורה באבטחה

- מונעת זיהוי תבניות – ללא P-Box תוקף יכול לזהות דפוסים חוזרים בנתונים המוצפנים.
- מגבירה Diffusion – שינוי קטן בקלט משפיע על כל הפלט, מה שמקשה על ניתוח סטטיסטי.
- משפרת עמידות למתקפות – הופכת את פענוח ההצפנה למאתגר יותר עבור תוקפים.

שאלה בנושא תיבת תמורה

- שאלה:
- נתונה תיבת תמורה שמבצעת את הסידור הבא:
[5, 2, 6, 1, 3, 7, 4, 8]
- אם הקלט הוא:
0 1 0 0 1 1 0 1
- מה יהיה הפלט לאחר הפעלת התיבה?

תשובה

- קלט:
0 1 0 0 1 1 0 1
- תיבת התמורה P-Box:
[5, 2, 6, 1, 3, 7, 4, 8]
- כלומר, הביט במקום ה-5 יעבור להיות במקום ה-1, הביט במקום ה-2 יעבור למקום ה-2, וכך הלאה. שלבי הפתרון
- נרשום את הקלט עם מיקומי הביטים:
- עכשיו נמקם את הביטים לפי הסדר החדש של תיבת התמורה:
- תוצאה סופית של הפלט לאחר תיבת התמורה:

0 1 1 1 1 0 0 0 •

- יוצא לנו הלכה למעשה:
- לקחנו את הקלט השתמשנו בתיבת התמורה כדי לסדר מחדש את הביטים קיבלנו פלט חדש בהתאם לכללים שנקבעו מראש

מהי תיבת החלפה S-Box - Substitution Box ?

- תיבת החלפה S-Box היא רכיב קריפטוגרפי המשמש באלגוריתמים של הצפנה סימטרית, כגון DES ו-AES מטרתה היא לשנות את הערכים של ביטים או קבוצות ביטים לפי טבלה מוגדרת מראש.
- תיבת החלפה יוצרת ערפול Confusion היא הופכת את הקשר בין הטקסט הקריא plaintext לבין הטקסט המוצפן ciphertext למסובך יותר, כך שיהיה קשה לתוקף להסיק מידע על המפתח.

כיצד פועלת תיבת החלפה?

1. הקלט ל-S-Box הוא קבוצת ביטים לדוגמה, 6 ביטים ב-DES או 8 ביטים ב-AES.
2. ה-S-Box מחליף את קבוצת הביטים בערך חדש לפי טבלת המרה קבועה מראש.
3. הפלט הוא קבוצת ביטים שונה בגודל זהה או קטן יותר מהקלט.
- דוגמה פשוטה לתיבת החלפה: נניח שיש לנו 4 ביטים כקלט, ואנו משתמשים בטבלה הבאה להחלפה: לדוגמה, אם הקלט הוא 0011, הפלט לאחר ה-S-Box יהיה 0001.

תיבות החלפה באלגוריתמים ידועים

1. DES (Data Encryption Standard):
 - משתמש ב-8 תיבות החלפה שונות, כאשר כל אחת מקבלת 6 ביטים כקלט ומחזירה 4 ביטים כפלט.
 - התיבות נבחרו כך שהן עמידות נגד התקפות קריפטואנליזה.
2. AES (Advanced Encryption Standard)
 - משתמש בתיבת החלפה אחת בלבד, שנקראת S-Box של AES.
 - תיבת החלפה של AES מבוססת על פעולות מתמטיות בשדה Galois Field כדי להבטיח ביטחון גבוה.
 - כל בתיבים Bytes במטריצת ה-State של AES מוחלפים על פי ה-S-Box.

סוגי S-Box באלגוריתמים שונים

- S-Box קבוע Fixed S-Box: טבלה קבועה שנבחרת מראש ומשמשת תמיד באותו אופן, כמו ב-DES. יתרונות: קל ליישום, מהיר. חסרונות: אם התוקף מגלה חולשה, כל המערכת נחשפת.
- S-Box ממחושב Dynamic S-Box: מחושב דינמית על בסיס המפתח הסודי, כמו בחלק מהגרסאות המתקדמות של AES. יתרונות: מקשה על מתקפות קריפטואנליזה. חסרונות: דורש יותר משאבי חישוב.

כיצד פועל S-Box באלגוריתם DES?

- ב-DES יש 8 תיבות S-Box שונות, שכל אחת מקבלת 6 ביטים כקלט ומחזירה 4 ביטים כפלט.
- אופן הפעולה: 1. הקלט (6 ביטים) מחולק ל-2 חלקים:
 - הביטים הראשון והאחרון קובעים את השורה בטבלה.
 - ארבעת הביטים האמצעיים קובעים את העמודה בטבלה.
- 2. הערך המתאים מוחלף ב-4 ביטים חדשים שהם הפלט.

דוגמה מעשית S-Box ב-DES

- ניקח דוגמה עם אחת מתיבות החלפה של DES
- קלט: 6 ביטים $101011 \rightarrow$
- 1. הביט הראשון והאחרון (10) קובעים את השורה בטבלה.
- 2. ארבעת הביטים הפנימיים (0101) קובעים את העמודה בטבלה.
- 3. הערך שנמצא באותו מקום בטבלה מוחלף כפלט (4 ביטים).

כיצד פועל S-Box באלגוריתם AES?

- ב-AES קיימת תיבת החלפה אחת בלבד, המכונה AES S-Box והיא פועלת על בתים שלמים (8 ביטים) ולא על קבוצות קטנות יותר כמו ב-DES.
- פעולת ההחלפה ב-AES מורכבת משני שלבים עיקריים: 1. אינוורסיה בשדה גלואה: Galois Field Inversion חישבו הופכי מתמטי עבור כל בייט ב- $GF(2^8)$. 2. טרנספורמציה ליניארית: Affine Transformation. הכפלה מטריציונית כדי להוסיף ערבוב נוסף.
- יתרון משמעותי של ה-S-Box של AES הוא שהוא עמיד יותר למתקפות קריפטואנליזה מאשר ה-S-Boxes של DES בזכות תכנון מתמטי חזק יותר.

מתקפות על S-Box ואופן ההתמודדות איתן

- מתקפה ליניארית: Linear Cryptanalysis: אם תיבת ההחלפה ניתנת לייצוג באמצעות משוואות ליניאריות פשוטות, ניתן לחשוף חלקים מהמפתח הסודי. פתרון: שימוש ב-S-Box לא ליניארי עם תכונות מתמטיות שמונעות ייצוג ליניארי פשוט.
- מתקפה דיפרנציאלית: Differential Cryptanalysis: אם שינוי קטן בקלט מוביל לשינוי קטן וצפוי בפלט, ניתן לנצל את הדפוסים כדי לשבור את ההצפנה. פתרון: תכנון S-Box כך ששינוי קטן בקלט יגרום לשינוי גדול ובלתי צפוי בפלט.
- מתקפת תזמון: Timing Attack: אם זמן החישוב של S-Box משתנה בהתאם לקלט, ניתן להסיק מידע על הנתונים והפעולות הפנימיות. פתרון: שימוש בטבלאות קבועות או אלגוריתמים עם זמן ביצוע אחיד.

מדוע S-Box חשוב בקריפטוגרפיה?

1. יוצר ערפול: Confusion על תוקף להסיק מידע על המפתח.
2. מונע מתקפות ליניאריות ודיפרנציאליות: הופך את הקשרים בין קלט לפלט לפחות צפויים.
3. משנה את הנתונים באופן לא ליניארי: מה שמונע מתקפות מתמטיות ישירות.

מהי הצפנה א-סימטרית?

- הצפנה א-סימטרית היא שיטה בקריפטוגרפיה שבה נעשה שימוש בשני מפתחות שונים:
- מפתח ציבורי: Public Key משמש להצפנה. ניתן להפיץ אותו לכולם.
- מפתח פרטי: Private Key משמש לפענוח. הוא סודי ונמצא רק אצל הבעלים.
- איך זה עובד?
- כאשר מישהו רוצה לשלוח מידע סודי, הוא מצפין אותו בעזרת המפתח הציבורי של הנמען. רק הנמען, שמחזיק את המפתח הפרטי המתאים, יכול לפענח את המידע.

דוגמה מעשית

- נניח שאליס רוצה לשלוח הודעה לבוב באופן מאובטח.
- 1. בוב מפרסם את המפתח הציבורי שלו לכל העולם.
- 2. אליס משתמשת במפתח הציבורי של בוב כדי להצפין את ההודעה.
- 3. אליס שולחת את ההודעה המוצפנת לבוב.
- 4. בוב משתמש במפתח הפרטי שלו כדי לפענח את ההודעה.

דוגמאות לאלגוריתמים נפוצים בהצפנה א-סימטרית

1. RSA: אחד האלגוריתמים הנפוצים ביותר, מבוסס על פירוק מספרים גדולים לגורמים.
2. ECC (Elliptic Curve Cryptography): שיטה מתקדמת יותר שמבוססת על עקומים אליפטיים.
3. Diffie-Hellman: משמש בעיקר להחלפת מפתחות בצורה בטוחה.

שימושים נפוצים של הצפנה א-סימטרית

- 1. תקשורת מאובטחת SSL/TLS: הצפנה א-סימטרית משמשת בפרוטוקול HTTPS שבו הדפדפן והשרת מחליפים מפתחות בצורה בטוחה.
- 2. חתימות דיגיטליות: משמשות לאימות מקורות מסמכים והודעות. לדוגמה, אם בנק שולח לך הודעה חתומה דיגיטלית, תוכל לוודא שההודעה אכן נשלחה מהבנק.
- 3. אימות משתמשים: מערכות כמו SSH ו-GPG משתמשות בהצפנה א-סימטרית כדי לאמת זהות של משתמשים בלי צורך בסיסמה.
- 4. הצפנת אימיילים: PGP (Pretty Good Privacy) פרוטוקול מאפשר הצפנה בטוחה של הודעות דואר אלקטרוני באמצעות מפתחות ציבוריים ופרטיים.

אילו בעיות קיימות בהצפנה א-סימטרית?

- למרות היתרונות הרבים, קיימות גם מגבלות:
- 1. ביצועים איטיים – הצפנה א-סימטרית דורשת יותר כוח חישובי מהצפנה סימטרית, ולכן בדרך כלל משתמשים בה רק להחלפת מפתחות, ולא להצפנת כמויות גדולות של מידע.
- 2. הגנה על המפתח הפרטי – אם מישהו מצליח לגנוב את המפתח הפרטי, הוא יכול לזייף חתימות ולפענח מידע סודי.
- 3. פגיעות להתקפות קוונטיות – אלגוריתמים א-סימטריים כמו RSA עשויים להיות שבירים מול מחשבים קוונטיים (עקב אלגוריתם שור).

מהו שיתוף מפתחות?

- שיתוף מפתחות Key Exchange הוא תהליך שבו שני צדדים שרוצים לתקשר באופן מאובטח מחליפים ביניהם מפתחות קריפטוגרפיים, כך שהאקרים או גורמים שלישיים לא יוכלו להאזין או ליירט את המפתחות.
- למה צריך שיתוף מפתחות?
- בקריפטוגרפיה, הצפנה סימטרית כמו AES דורשת מפתח משותף לשני הצדדים. אבל איך משתפים את המפתח הזה בצורה בטוחה? כאן נכנסת ההצפנה הא-סימטרית – היא מאפשרת החלפת מפתחות מאובטחת, כך שגם אם מישהו מאזין לתקשורת, הוא לא יוכל לדעת מהו המפתח הסודי.

שיתוף מפתחות בהצפנה סימטרית וא-סימטרית

- 1. בהצפנה סימטרית:
- דורשת מפתח משותף. שיתוף מפתחות חייב להיות מאובטח, אחרת כל התקשורת בסכנה. נפתר ע"י פרוטוקולי החלפת מפתחות למשל Diffie-Hellman
- 2. בהצפנה א-סימטרית:
- אין צורך לשתף מפתח סימטרי ישירות. מפתח ציבורי משמש להצפנה, והמפתח הפרטי לפענוח. מאפשר שימוש ב-RSA לשיתוף מפתחות בביטחון.

בעיות ואתגרים בשיתוף מפתחות

- התקפת "אדם בתווך" (Man-in-the-Middle) – MITM
- מה הבעיה? אם אין מנגנון לאימות זהות, תוקף יכול להתחזות לכל אחד מהצדדים וליירט את תהליך החלפת המפתח.
- דוגמה:
- אליס חושבת שהיא שולחת מידע לבוב, אך בפועל הוא עובר דרך תוקף.
- התוקף מבצע חילופי מפתחות נפרדים עם כל צד.
- התוקף מפענח ומצפין מחדש את כל ההודעות, מה שמאפשר לו לקרוא את התקשורת.
- פתרון: אימות זהות באמצעות חתימות דיגיטליות או תעודות דיגיטליות SSL/TLS
- שימוש בפרוטוקולים כמו DH עם חתימות או RSA עם אישורים דיגיטליים.

הרחבות

- בעיית חישוביות – האם שיתוף מפתחות בטוח בטווח הארוך?
- כיום, רוב הפרוטוקולים מבוססים על בעיות מתמטיות קשות (למשל, בעיית הדיסקרטית של לוגריתמים ב-DH עם זאת, מחשבים קוונטיים עתידיים יוכלו לשבור פרוטוקולים כמו RSA ו-DH).
- לכן, מחפשים אלגוריתמים פוסט-קוונטיים כמו Lattice-Based Cryptography.
- הצפנה היברידית – הפתרון המודרני:
- רוב המערכות כיום משלבות הצפנה א-סימטרית וסימטרית:
- משתמשים ב-RSA או ECDH לשיתוף מפתח סימטרי.
- לאחר מכן משתמשים ב-AES להצפנה מהירה של הנתונים עצמם. כך מקבלים ביצועים גבוהים + אבטחה חזקה.

מה זה PKI?

- PKI תשתית מפתח ציבורי הוא מכלול של חוקים, נהלים, ותשתיות טכנולוגיות המאפשרות הצפנה א-סימטרית, חתימות דיגיטליות, ואימות זהויות בצורה בטוחה.

רקע על הצפנה א-סימטרית

- בניגוד להצפנה סימטרית שבה משתמשים במפתח אחד להצפנה ולפענוח, הצפנה א-סימטרית משתמשת בשני מפתחות:
- 1. מפתח ציבורי Public Key : משמש להצפנה, ניתן להפצה חופשית.
- 2. מפתח פרטי Private Key : משמש לפענוח, חייב להישאר סודי.
- שימושים של PKI:
- הצפנת מידע: למשל, שליחת הודעות מוצפנות.
- חתימה דיגיטלית: אימות מקוריות ושלמות המידע.
- אימות זהות: לדוגמה, בכניסה לאתרים עם תעודות SSL/TLS.

איך PKI עובד?

- PKI מבוסס על רשות מאשרת CA - Certificate Authority שמנפיקה תעודות דיגיטליות המאשרות את הזהות של גורם מסוים.
- תהליך אימות זהות בעזרת PKI:
- 1. משתמש/שרת מבקש תעודה דיגיטלית מ-CA.
- 2. ה-CA מאמת את זהותו ומנפיק לו תעודה דיגיטלית Digital Certificate.
- 3. המשתמש משתמש בתעודה כדי להצפין מידע או לחתום עליו דיגיטלית.
- 4. צד שלישי יכול לבדוק את התעודה בעזרת המפתח הציבורי של ה-CA ולוודא שהיא אמינה.

מרכיבי PKI עיקריים

- א. רשות מאשרת CA - Certificate Authority
- גוף שמנפיק תעודות דיגיטליות המוכיחות שהמפתח הציבורי שייך לגורם מסוים. CA חותמת על התעודות שלה עם המפתח הפרטי שלה, וכך ניתן לאמת שהן אמינות. ישנן רשויות מאשרות גלובליות כמו Let's, GlobalSign, DigiCert.
- Encrypt וכן פנימיות (לשימוש בארגון).
- ב. רשות רישום RA - Registration Authority
- משמשת לתהליך אימות זהות לפני הנפקת תעודה ע"י ה-CA. ה-RA מוודאת שהבקשה לגיטימית לפני אישורה.
- ג. מאגר תעודות Certificate Repository
- מסד נתונים שמאחסן את התעודות הדיגיטליות שהונפקו, כך שהן יהיו זמינות לבדיקת אמינות.
- ד. רשימת תעודות מבוטלות Certificate Revocation List - CRL
- רשימה של תעודות שכבר אינן תקפות בגלל גניבה, פג תוקף, או ביטול יזום. ניתן לבדוק את הסטטוס של תעודה באמצעות פרוטוקול OCSP (Online Certificate Status Protocol).

מבנה של תעודה דיגיטלית X.509 Certificate

- תעודה דיגיטלית היא מסמך מבוסס תקן X.509 המכיל:
- 1. שם הישות (שם משתמש/שרת/ארגון) 2. המפתח הציבורי של הישות 3. שם הרשות שהנפיקה את התעודה CA
- 4. תאריך תוקף התעודה 5. חתימה דיגיטלית של ה-CA
- התעודה מוצגת בפורמט PEM או DER וניתנת לקריאה על ידי דפדפנים ותוכנות אבטחה.

סוגי תעודות דיגיטליות

- 1. SSL/TLS Certificates משמשות להצפנת תעבורת אינטרנט HTTPS
- 2. Code Signing Certificates מאשרות שהקוד נחתם ע"י מפתח ידוע ולא שונה ע"י צד שלישי.
- 3. Email Certificates (S/MIME) משמשות לאימות זהות ושמירת פרטיות באימיילים.
- 4. Client Certificates משמשות לאימות משתמשים ושירותים.

ניהול מחזור חיי תעודה דיגיטלית

- 1. יצירה – המשתמש מייצר בקשה לתעודה CSR - Certificate Signing Request
- 2. אימות – ה-RA בודק את זהות המבקש
- 3. הנפקה – ה-CA מנפיק את התעודה ומחתיים אותה דיגיטלית.
- 4. שימוש – התעודה משמשת להצפנה או חתימה דיגיטלית.
- 5. חידוש – כשהתעודה עומדת לפוג, יש לחדש אותה כדי לשמור על אבטחה.
- 6. ביטול – אם המפתח הפרטי נפרץ או הזהות השתנתה, התעודה מבוטלת ונכנסת ל-CRL

דוגמה מעשית - אתר אינטרנט עם HTTPS

- כאשר אתה נכנס לאתר מאובטח כמו Gmail הדפדפן שלך מבצע את השלבים הבאים:
- 1. האתר מציג את התעודה הדיגיטלית שלו, שמונפקת על ידי CA מהימן.
- 2. הדפדפן בודק שהתעודה תקפה (לא פג תוקף, נחתמה על ידי CA מוכר, וכו').
- 3. אם התעודה אמינה, הדפדפן יוצר מפתח הצפנה סימטרי ומצפין אותו עם המפתח הציבורי של האתר.
- 4. רק האתר יכול לפענח את ההצפנה בעזרת המפתח הפרטי שלו – וכך התקשורת ביניכם הופכת מאובטחת.

יתרונות וחסרונות של PKI

- יתרונות:
- 1. אבטחה גבוהה - הצפנה א-סימטרית מגנה מפני פריצות וזיופים
- 2. ניתן לאימות - תעודות דיגיטליות מאפשרות לאמת זהויות בביטחון
- 3. גמישות - מאפשר הצפנה, חתימה דיגיטלית ואימות זהות
- חסרונות:
- 1. מורכב לניהול - דורש מנגנוני ניהול תעודות, רשת CA ועוד
- 2. ביצועים נמוכים יחסית - הצפנה א-סימטרית איטית יותר מהצפנה סימטרית
- 3. בעיית ביטול תעודות - אם מפתח פרטי נפרץ, צריך לבטל את כל התעודות הקשורות

מה זה RSA ולמה הוא חשוב?

- תאר לך שיש לך יומן סודי, ואתה רוצה שמישהו יוכל לשלוח לך הודעות אבל רק אתה יכול לקרוא את זה. איך עושים את זה?
- אחת אפשרות היא הצפנה סימטרית – כמו AES אבל אז תצטרך למצוא דרך להעביר את המפתח הסודי לשולח... וזה מסוכן!
- כאן נכנסת לתמונה הצפנה א-סימטרית ב-RSA אין צורך לשתף מפתח סודי! כל אחד יכול להצפין עבורך הודעה, אבל רק אתה יכול לפענח.
- RSA תקשורת משתמשת ל: הגנה על סיסמאות ונתונים רגישים וחתימות דיגיטליות

מהי הצפנת RSA?

- RSA היא שיטת הצפנה אסימטרית, כלומר היא משתמשת בשני מפתחות שונים:
- מפתח ציבורי: Public Key משמש להצפנת הודעות. אפשר לפרסם אותו בפומבי, וכל אחד יכול להשתמש בו כדי להצפין הודעה עבורך.
- מפתח פרטי: Private Key משמש לפענוח הודעות. אותו אתה שומר בסוד, ורק אתה יכול לפענח הודעות שהוצפנו עבורך באמצעות המפתח הציבורי שלך.
- כיצד זה עובד?
- הצפנת RSA מבוססת על בעיה מתמטית שנקראת "בעיית פירוק לגורמים ראשוניים". קשה מאוד למצוא את הגורמים הראשוניים של מספר גדול, וזה מה שמבטיח את אבטחת ההצפנה.

בעיית פירוק לגורמים ראשוניים

- הבסיס המתמטי של RSA הוא בעיה שנקראת "בעיית פירוק לגורמים ראשוניים". בעיה זו עוסקת במציאת הגורמים הראשוניים של מספר גדול. לדוגמה, קל למצוא את הגורמים הראשוניים של המספר 15 (3 ו-5), אך קשה מאוד למצוא את הגורמים הראשוניים של מספר בן מאות ספרות.
- RSA מסתמכת על הקושי של בעיה זו. המספר חשוא חלק מהמפתח הציבורי והפרטי הוא מכפלה של שני מספרים ראשוניים גדולים. אם מישהו יצליח למצוא את הגורמים הראשוניים של n הוא יוכל לחשב את המפתח הפרטי ולפענח את ההודעות.

חשבון מודולרי

- RSA משתמשת בחשבון מודולרי, שהוא סוג של חשבון שבו המספרים "מתגלגלים" לאחר מספר מסוים. לדוגמה, חשבון מודולרי 12 משמש בשעון. כאשר השעה היא 10 בבוקר, ואנו מוסיפים 5 שעות, השעה תהיה 3 אחר הצהריים ($10 + 5 = 15$, ו-15 מודולו 12 הוא 3).
- ב-RSA כל החישובים מתבצעים מודולו n . זה מבטיח שהתוצאות יישארו במסגרת המספרים המוגדרת על ידי n .

פונקציית אוילר

- פונקציית אוילר $\phi(n)$ מחשבת את מספר המספרים הטבעיים הקטנים מ- n וזרים לו. מספרים זרים הם מספרים שאין להם גורם משותף פרט ל-1.
- פונקציית אוילר משחקת תפקיד חשוב בחישוב המפתח הפרטי d .

משפט אוילר

- משפט אוילר קובע שאם a ו- n הם מספרים זרים, אז $a^{\phi(n)} \equiv 1 \pmod{n}$ (הסימן שדומה לשווה הכוונה "שקילות") משפט זה הוא הבסיס המתמטי להוכחת נכונותה של RSA.
- * סימן השקילות אומר ששני האגפים משאירים אותה שארית כאשר מחלקים אותם במספר מסוים

אלגוריתם אוקלידס המורחב

- אלגוריתם אוקלידס המורחב משמש למציאת המספרים d ו- e שמקיימים את הקונגרואנציה $d * e \equiv 1 \pmod{(p-1)(q-1)}$. RSA.

תהליך הצפנה ופענוח

- תהליך ההצפנה והפענוח כולל את השלבים הבאים:
- 1. יצירת מפתחות:
- בחרים שני מספרים ראשוניים גדולים ושונים זה מזה, p ו- q
- בחירת שני מספרים ראשוניים גדולים p ו- q היא קריטית לאבטחת RSA ככל שהמספרים גדולים יותר, כך קשה יותר למצוא אותם, ולכן ההצפנה בטוחה יותר.
- בעולם האמיתי, משתמשים במספרים ראשוניים בני מאות ספרות ויותר.

- חשוב לבחור מספרים ראשוניים שונים זה מזה, כדי למנוע חולשות אפשריות.
- 2. חישוב n:
- מכפילים אותם זה בזה: $n = p * q$
- n יהיה חלק מהמפתח הציבורי והמפתח הפרטי.
- 3. חיפוש e:
- מחפשים מספר e שזר ל- $(q-1) * (p-1)$ וגם גדול מ-1.
- מספרים זרים הם מספרים שאין להם גורם משותף פרט ל-1.
- e משמש להצפנה, ולכן חשוב לבחור אותו כך שיהיה קל לחישוב.
- 4. חיפוש d:
- מחפשים מספר d כך ש- $d * e \equiv 1 \pmod{(p-1) * (q-1)}$
- משמעות הדבר היא ש- $d * e$ חלקי $(p-1) * (q-1)$ נותן שארית 1.
- d משמש לפענוח, והוא חייב להיות סודי.
- המפתח הציבורי הוא (n, e) והמפתח הפרטי הוא (n, d).
- d הוא מספר שלם כך ש- $d * e \equiv 1 \pmod{(p-1) * (q-1)}$
- מה זה אומר?
- d הוא מספר שלם: זה אומר ש- d יכול להיות כל מספר שלם (חיובי, שלילי).
- $d * e \equiv 1 \pmod{(p-1) * (q-1)}$ זהו סימון מתמטי שנקרא "קונגרואנציה מודולרית". המשמעות שלו היא:
 - כאשר מחלקים את המכפלה של d ו- e במספר $(p-1) * (q-1)$ השארית המתקבלת היא 1.
 - דוגמה מספרית
 - נחזור לדוגמה שלנו:
 - $p = 5, q = 11$
 - $4 * 10 = 40 = (q-1) * (p-1)$
 - $e = 7$
 - $d = 23$
 - בואו נבדוק את הקונגרואנציה:
 - $161 = 7 * 23$
 - $4 = 40 / 161$ עם שארית 1
 - אכן, $1 \equiv 7 * 23 \pmod{40}$

דוגמא

- בואו נראה דוגמה עם מספרים גדולים יותר:
- יצירת מפתחות:
 - $p = 61, q = 53$
 - $n = 61 * 53 = 3233$
 - $60 * 52 = 3120 = (q-1) * (p-1)$
 - $e = 17$ (כי הוא זר ל- 3120)
 - $d = 2753$ כי $17 * 2753 \equiv 1 \pmod{3120}$
 - [בישילכם: כלומר, אנחנו מחפשים מספר d שכאשר נכפיל אותו ב- e (שהוא 17 בדוגמה שלך), התוצאה שתתקבל תשאיר שארית 1 בחילוק ב- 3121 (שהוא הפוקנציה של n שחושב בשלב 3)
 - המפתח הציבורי הוא (17, 3233), והמפתח הפרטי הוא (2753, 3233).
 - 2. הצפנה:
 - ההודעה $m = 1234$
 - $c = 1234^{17} \pmod{3233} = 855$
 - כלומר: נניח שההודעה שרוצים לשלוח היא המספר 1234, לכן אנו (אושרי) מעלים את ההודעה בחזקת המפתח הציבורי $e=17$ ומבצעים מודולו n, וזה בעצם החישוב שאנו רואים למעלה.
 - 3. פענוח:
 - $m = 855^{2753} \pmod{3233} = 1234$

- כלומר: לוקחים את הטקסט המוצפן (855) ומעלה אותו בחזקת המפתח הפרטי שלך $d=2753$ ומבצעים מודולו n , וזה בעצם החישוב שאנו רואים למעלה, היינו קבלנו את ההודעה 1234.
- 1. פרמטרים:
 - $N = 7$
 - $p = 3$
 - $q = 32$
- 2. יצירת מפתחות:
 - בחירת g : $g = x^2 - 1$
 - $g = 2x + 1$
- חישוב h :

$$h = (p * f * g) / q = (3 * (x^2 - 1) * (2x + 1)) / 32 = (6x^3 + 3x^2 - 6x - 3) / 32$$
 - לאחר חישוב מודולו 32, נקבל: $h = 10x^3 + 15x^2 + 26x + 29$
 - הסבר שקופית הבאה..
- במשוואה שלנו, $h = (6x^3 + 3x^2 - 6x - 3) / 32$ עלינו למצוא את השארית של כל מקדם לאחר חילוק ב-32.
 - 6: $6 \bmod 32 = 6$ קטן מ-32, ולכן השארית היא 6 עצמה.
 - 3: $3 \bmod 32 = 3$ קטן מ-32, ולכן השארית היא 3 עצמה.
 - 6: $-6 \bmod 32 = 26$ כאן אנו נתקלים במספר שלילי. כדי למצוא את השארית, אנו מוסיפים 32 למספר השלילי עד שנקבל מספר חיובי. במקרה זה, $-6 = 32 + -6$.
 - 3: $-3 \bmod 32 = 29$ כמו קודם, אנו מוסיפים 32 למספר השלילי: $-3 = 32 + -3$.
- הצפנה:
 1. בחירת r : $r = x + 1$
 2. חישוב c :

$$c = (p * r * h + m) \bmod q = (3 * (x + 1) * (10x^3 + 15x^2 + 26x + 29) + x^2) \bmod 32$$
 - לאחר חישוב מודולו 32, נקבל: $c = 2x^3 + 17x^2 + 15x + 2$
 - בואו נפשט את זה:
- 1. נפתח את הסוגריים:

$$(x + 1) * (10x^3 + 15x^2 + 26x + 29) = 10x^4 + 25x^3 + 41x^2 + 55x + 29$$
- 2. נכפיל ב-3:

$$(10x^4 + 25x^3 + 41x^2 + 55x + 29) * 3 = 30x^4 + 75x^3 + 123x^2 + 165x + 87$$
- 3. נוסיף x^2 :

$$30x^4 + 75x^3 + 123x^2 + 165x + 87 + x^2 = 30x^4 + 75x^3 + 124x^2 + 165x + 87$$
- ניקח את השארית מודולו 32:
 - לכל מקדם, נמצא את השארית לאחר חילוק ב-32:
 - $30 \bmod 32 = 30$
 - $75 \bmod 32 = 11$
 - $124 \bmod 32 = 28$
 - $165 \bmod 32 = 5$
 - $87 \bmod 32 = 23$
- דמיינו שיש לנו שעון שמראה רק 32 שעות.
 - אם השעה היא 30, אז השעה "האמיתית" היא 30, כי היא קטנה מ-32.
 - אם השעה היא 75, אז השעה "האמיתית" היא 11, כי $75 - 32 - 32 = 11$.
 - אם השעה היא 124, אז השעה "האמיתית" היא 28, כי $124 - 32 - 32 - 32 = 28$.
 - אם השעה היא 165, אז השעה "האמיתית" היא 5, כי $165 - 32 - 32 - 32 - 32 = 5$.
 - אם השעה היא 87, אז השעה "האמיתית" היא 23, כי $87 - 32 - 32 - 32 = 23$.

- $c = 30x^4 + 11x^3 + 28x^2 + 5x + 23$ וזאת התוצאה
- פענוח:
- חישוב a:
- $a = (f * c) \bmod q = ((x^2 - 1) * (2x^3 + 17x^2 + 15x + 2)) \bmod 32$
- לאחר חישוב מודולו 32, נקבל: $a = 2x^3 + 15x^2 + 2x + 30$
- חישוב m:
- $m = (a * g) \bmod p = ((2x^3 + 15x^2 + 2x + 30) * (2x + 1)) \bmod 3$
- לאחר חישוב מודולו 3, נקבל: $m = x^2$
- תוצאה:
- הצלחנו לפענח את ההודעה המקורית $m = x^2$.
- א. יצירת מפתחות:
- 1. בחירת מספרים ראשוניים:
- $p = 5$
- $q = 11$
- 2. חישוב n
- $n = p * q = 5 * 11 = 55$
- המפתח הציבורי: $n = 55$ המפתח הפרטי: $(p, q) = (5, 11)$
- ב. הצפנה:
- 1. בחירת הודעה:
- $m = 7$
- $c = m^2 \bmod n = 7^2 \bmod 55 = 49 \bmod 55 = 49$
- הטקסט המוצפן: $c = 49$
- ג. פענוח:
- 1. חישוב שורשים ריבועיים:
- יש למצוא אכך ש- $x^2 \bmod 55 = 49$
- בעזרת המפתח הפרטי $(5, 11)$, ניתן למצוא את ארבעת השורשים הריבועיים הבאים:
- $x_1 = 7$
- $x_2 = 48$
- $x_3 = 18$
- $x_4 = 37$
- 2. בחירת ההודעה המקורית:
- מתוך ארבעת השורשים הריבועיים, רק $x_1 = 7$ הוא ההודעה המקורית m
- בחירת פרמטרים:
- $p = 23$ (מספר ראשוני גדול)
- $g = 5$ (יוצר של החבורה הכפלית של $\mathbb{Z}/23\mathbb{Z}$)
- בחירת מפתח פרטי:
- $x_1 = 7$
- $x_2 = 11$
- חישוב מפתח ציבורי:
- $y_1 = g^{x_1} \bmod p = 5^7 \bmod 23 = 19$
- $y_2 = g^{x_2} \bmod p = 5^{11} \bmod 23 = 2$
- המפתח הציבורי: $(2, 19, 5, 23)$
- המפתח הפרטי: $(11, 7)$
- פרמטרים:
- אורך הקוד: $n = 7$
- מימד ההודעה: $k = 4$
- יכולת תיקון שגיאות: $t = 1$

- מטריצה יוצרת G:
- $|G| = |1000||0100||0010||0001||1101||0110||1011|$.
- מטריצת בדיקה H:
- $|H| = |101||111||010||101||100||010||001|$.
- המפתח הציבורי: G המפתח הפרטי: מידע נוסף שמאפשר פענוח יעיל (לא נכלל בדוגמה זו)
- 1. יצירת מפתחות:
- מפתח פרטי (סופר-גדלים):
- $w = (2, 3, 7, 14, 30)$.
- כופל:
- $a = 17$.
- מודולו:
- $m = 105$.
- מפתח ציבורי:
- $b = (a * w) \bmod m = (34, 51, 119, 238, 510) \bmod 105 = (34, 51, 14, 23, 80)$.
- נניח ש:
- $p = 11$.
- $g = 2$.
- $a = 3$.
- $b = 5$.
- אז:
- $A = 2^3 \bmod 11 = 8$.
- $B = 2^5 \bmod 11 = 10$.
- $K = 10^3 \bmod 11 = 1000 \bmod 11 = 10$.

הסבר על שורת ההשוואה

- נתונה לנו השורה: $d = 2753$ (כי $d \equiv 17 * 2753 \pmod{3120}$):
- $d = 2753$: זה אומר ש- d הוא מספר שלם שערכו 2753. d הוא חלק מהמפתח הפרטי שלנו, והוא משמש לפענוח הודעות.
- (כי $17 * 2753 \equiv 1 \pmod{3120}$): זו הסיבה מדוע, $d=2753$ הסימן \equiv נקרא "קונגרואנציה מודולרית". בוא נפשט אותו עוד יותר:
- $17 * 2753$: זו פשוט מכפלה של שני מספרים. 2753 כפול 17 שווה 46801.
- $1 \pmod{3120}$: זה החלק המעניין. זה אומר שאנחנו מתעניינים בשארית שתתקבל כאשר נחלק את התוצאה (46801) ב-3120.
- במילים אחרות: אנחנו מחפשים את השארית של 46801 כשמחלקים אותו ב-3120.
- בואו נחשב את זה:
- $15 = 3120 / 46801$ עם שארית 1.
- אז, 46801 אכן שקול ל-1 מודולו 3120. לכן, הקונגרואנציה מתקיימת.

מהו צופן אל-גמאל?

- צופן אל-גמאל הוא שיטת הצפנה אסימטרית, כלומר היא משתמשת בשני מפתחות שונים:
- מפתח ציבורי: Public Key משמש להצפנת הודעות. כל אחד יכול להשתמש בו כדי להצפין הודעה עבורך.
- מפתח פרטי: Private Key משמש לפענוח הודעות. רק אתה יכול להשתמש בו כדי לפענח הודעות שהוצפנו עבורך.
- כיצד זה עובד?
- צופן אל-גמאל מבוסס על בעיה מתמטית שנקראת "בעיית הלוגריתם הברידי". קשה מאוד למצוא את הלוגריתם הברידי של מספר גדול, וזה מה שמבטיח את אבטחת ההצפנה.
- בנוסף, צופן אל-גמאל מבוסס על בעיית הלוגריתם הדיסקרטי, שדומה במורכבותה לבעיית פירוק לגורמים עליה מבוסס RSA במילים פשוטות, זוהי בעיה מתמטית שקשה לפתור כאשר המספרים גדולים.

שימושים עיקריים

- 1. הצפנת מידע: צופן אל גמאל משמש להצפנה של מידע רגיש כמו סיסמאות, מספרי כרטיסי אשראי ומידע אישי אחר.
- 2. חתימה דיגיטלית: צופן אל גמאל יכול לשמש ליצירת חתימות דיגיטליות, שמאפשרות לוודא את מקוריות המידע ולוודא שהוא לא השתנה.

יתרונות וחסרונות

- יתרונות:
 - 1. ביטחון גבוה: צופן אל גמאל נחשב לאלגוריתם בטוח מאוד, כל עוד משתמשים במפתחות חזקים ובאקראיות נכונה.
 - 2. גמישות: צופן אל גמאל יכול לשמש גם להצפנה וגם לחתימה דיגיטלית.
- חסרונות:
 - 1. יעילות: צופן אל גמאל אינו יעיל כמו אלגוריתמים סימטריים, ולכן הוא לא מתאים להצפנה של כמויות גדולות של מידע.
 - 2. גודל מפתח: גודל המפתחות בצופן אל גמאל גדול יחסית, מה שיכול להשפיע על הביצועים.
 - ביטחון מוכח מתמטית – אם ניתן לשבור את הצפנת רבין, ניתן גם לפרק לגורמים את n מה שנחשב בעיה קשה. פשוט ומהיר יותר מ-RSA – הפעולות העיקריות הן העלאת חזקות מודולריות.
 - פענוח לא חד-חד-ערכי – מתקבלים 4 פתרונות אפשריים, ולכן יש צורך במנגנון נוסף כדי לזהות את ההודעה הנכונה. לא בשימוש רחב – בניגוד ל-RSA ו-ECC צופן רבין פחות פופולרי בקריפטוגרפיה מעשית. רגישות להתקפות מסוימות – אם התוקף יודע תכונות מסוימות על q ניתן לשפר התקפות.
 - 1. ביטחון גבוה: עמידות מוכחת בפני התקפות שונות.
 - 2. גמישות: מתאים למגוון יישומים, כולל הצפנה וחתימה דיגיטלית.
 - 1. מורכבות: האלגוריתם מורכב יחסית להבנה וליישום.
 - 2. יעילות: פעולות ההצפנה והפענוח עשויות להיות איטיות יחסית.
- שימושים:
 - צופן קריימר-שופ משמש בעיקר ביישומים שבהם נדרשת רמת אבטחה גבוהה במיוחד, כמו למשל:
 - הצפנת מידע רגיש.
 - חתימה דיגיטלית מאובטחת.
 - פרוטוקולים קריפטוגרפיים מתקדמים.
 - ביטחון:
 - ביטחון צופן מקאליס מבוסס על הקושי שבפענוח קוד ליניארי כללי. בעיה זו נחשבת קשה מאוד לפתרון, ולכן צופן מקאליס נחשב לבטוח.
 - 1. עמידות בפני התקפות קוונטיות: צופן מקאליס נחשב לעמיד בפני התקפות של מחשבים קוונטיים עתידיים.
 - 2. יעילות: פעולות ההצפנה והפענוח יחסית מהירות.
 - 1. גודל מפתח גדול: המפתח הציבורי בצופן מקאליס גדול מאוד, מהווה חיסרון מבחינת אחסון ושידור.
 - 2. מורכבות: האלגוריתם מורכב יחסית להבנה וליישום.
 - צופן מקאליס משמש בעיקר ביישומים שבהם נדרשת עמידות בפני התקפות קוונטיות, כמו למשל:
 - פרוטוקולים קריפטוגרפיים עמידים בפני קוונטים.
 - 1. יעילות יחסית
 - 2. פוטנציאל לעמידות בפני התקפות קוונטיות
 - 1. מורכבות יחסית להבנה
 - 2. גודל מפתח גדול
- לסיכום:
 - הצפנת תרמיל היא שיטת הצפנה אסימטרית ייחודית ומעניינת, המציעה רמת אבטחה גבוהה. עם זאת, חשוב להבין את המורכבות שלה ואת מגבלותיה.
 - 1. מאפשר יצירת מפתח סימטרי משותף בלי להעביר אותו ישירות.

- 2. עמיד בפני האזנות פסיביות (אף אחד שמאזין לא יכול לגלות את המפתח).
- 3. מהווה בסיס לפרוטוקולים מאובטחים רבים (VPN, TLS וכו').
- 1. חשוף למתקפת "אדם בתווך": MITM תוקף יכול ליירט את ההתקשרות וליצור מפתחות נפרדים עם כל צד.
- 2. אינו מספק אימות זהות – לא ניתן לדעת אם הצד השני הוא אכן מי שהוא טוען להיות.
- 3. דורש מספרים גדולים מאוד כדי להיות בטוח נגד התקפות כוח גס.

תהליך ההצפנה

- תהליך ההצפנה והפענוח כולל את השלבים הבאים:
- יצירת מפתחות:
 - בוחרים מספר ראשוני גדול p .
 - בוחרים מספר שלם g שהוא שורש פרימיטיבי (חכו להמשך) מודולו p .
 - בוחרים מספר שלם פרטי x .
 - מחשבים את $y = g^x \pmod{p}$.
 - המפתח הציבורי הוא (p, g, y) , והמפתח הפרטי הוא x .
- הצפנה:
 - כדי להצפין הודעה m בוחרים מספר שלם אקראי k .
 - מחשבים את $a = g^k \pmod{p}$.
 - מחשבים את $b = m * y^k \pmod{p}$.
 - ההודעה המוצפנת היא (a, b) .
- פענוח:
 - כדי לפענח את ההודעה המוצפנת, (a, b) מחשבים את $m = b * a^{-x} \pmod{p}$.

דוגמא הלכה למעשה

- בואו ניקח דוגמה פשוטה עם מספרים קטנים כדי להמחיש את התהליך:
- יצירת מפתחות:
 - $p = 11$
 - $g = 2$
 - $x = 3$
 - $y = 2^3 \pmod{11} = 8$
 - 2^3 : זה פשוט 2 בחזקת 3, כלומר $2 * 2 * 2 = 8$.
 - $(\pmod{11})$: זה החלק המעניין. זה אומר שאנחנו מתעניינים בשארית שתתקבל כאשר נחלק את התוצאה (8) ב-11.
- הצפנה:
 - 8: זה אומר שהתוצאה הסופית של הביטוי היא 8.
 - המפתח הציבורי הוא $(8, 2, 11)$, והמפתח הפרטי הוא 3.
- הצפנה:
 - ההודעה $m = 7$
 - $k = 5$
 - $a = 2^5 \pmod{11} = 10$
 - $b = 7 * 8^5 \pmod{11} = 7 * 32768 \pmod{11} = 7 * 10 \pmod{11} = 70 \pmod{11} = 4$
 - כלומר: $32,768 \% 11 = 10$, ועכשיו נחשב $7 * 10 = 70$.
 - ההודעה המוצפנת היא $(4, 10)$.
- פענוח:
 - $m = 4 * 10^{-3} \pmod{11} = 4 * 10^8 \pmod{11} = 4 * 100000000 \pmod{11} = 4 * 10 \pmod{11} = 40 \pmod{11} = 7$
 - $m = 4 * 10^{-3} \pmod{11}$:
 - m מייצג את ההודעה המקורית שאנו מנסים לפענח.
 - 4 הוא חלק מההודעה המוצפנת.

- 10 הוא בסיס החזקה.
- 3 היא החזקה השלילית.
- $(\text{mod } 11)$ אומר שאנו עובדים בחשבון מודולו 11, כלומר, אנו מתעניינים בשארית לאחר חילוק ב-11.
- חזקה שלילית פירושה 1 חלקי הבסיס בחזקה חיובית. לכן, 10 בחזקת 3- שווה ל-1 חלקי 10 בחזקת 3: $10^{-3} = 1 / 10^3$
- $1000000000 * 4 = (\text{mod } 11) 3$.
- פשוט חישוב את 10 בחזקת 8 וקיבלו 100,000,000.
- $10 * 4 = (\text{mod } 11) 4$.
- כאן חישוב את השארית של $100,000,000 (\text{mod } 11)$.
- $9090909.0909 = 11 / 100,000,000$
- השארית היא 10.
- $40 = (\text{mod } 11) 5$.
- כפלו את 4 ב-10 וקיבלו 40.
- $7 = 6$.
- חישוב את השארית של $40 (\text{mod } 11)$.
- $3.6363 = 11 / 40 \dots$
- השארית היא 7.

דוגמא הלכה למעשה - דרך שניה

- הסבר על השורה: $b = 7 * 8^5 (\text{mod } 11) = 7 * 32768 (\text{mod } 11) = 7 * 10 (\text{mod } 11) = 70 (\text{mod } 11) = 4$
- 1: $b = 7 * 8^5 (\text{mod } 11)$ זה אומר שאנחנו מחפשים את הערך של b הביטוי בצד ימין של המשוואה הוא הביטוי שנחשב כדי למצוא את b.
- $5 * 8^7$: זה פשוט אומר 7 כפול 8 בחזקת 5. 8 בחזקת 5 זה $8 * 8 * 8 * 8 * 8$, ששווה ל-32768. אז יש לנו 7 כפול 32768.
- $(\text{mod } 11) 3$: זה אומר שאנחנו מתעניינים בשארית שתתקבל כאשר נחלק את התוצאה $(32768 * 7)$ ב-11.
- $(\text{mod } 11) 32768 * 7$: 4: עכשיו נחשב את זה. 7 כפול 32768 שווה ל-229376.
- $(\text{mod } 11) 229376$: 5: כאן אנחנו מחפשים את השארית של 229376 כשמחלקים ב-11. אפשר לעשות חילוק ארוך או להשתמש במחשבון. התוצאה היא 20852 ושארית 4.

מהי הצפנת NTRU?

- NTRU (NTRUEncrypt) היא שיטת הצפנה מבוססת סריגים (Lattice-Based Cryptography) היא נחשבת בטוחה גם מול מחשבים קוונטיים, ולכן מהווה אלטרנטיבה להצפנות כמו RSA ו-ECC.
- למה NTRU שונה מהצפנות קלאסיות כמו RSA?
- מהירות – אלגוריתם NTRU מהיר יותר מ-RSA גם בהצפנה וגם בפענוח.
- עמידות בפני מחשוב קוונטי – בשונה מ-RSA ו-ECC הצפנה זו בטוחה מפני התקפות קוונטיות.
- שימוש בפולינומים – במקום מספרים גדולים NTRU משתמש בפולינומים עם חישובים מודולריים.

שלבי ההצפנה ב-NTRU

- ההצפנה ב-NTRU מבוססת על חישובים במערכת פולינומית מודולרית. נחלק את התהליך ל-3 שלבים:
 1. יצירת מפתחות
 2. הצפנת הודעה
 3. פענוח הודעה
- 1. יצירת מפתחות:
 - בוחרים מספרים שלמים גדולים p, q ו- N .
 - יוצרים שני פולינומים f ו- g עם מקדמים קטנים.
 - מחשבים את הפולינום $h = (p * f * g) / q$.
 - המפתח הציבורי הוא (N, p, q, h) .

- המפתח הפרטי הוא (f, g) .
- 2. הצפנה:
 - כדי להצפין הודעה, מבחרים פולינום אקראי r עם מקדמים קטנים.
 - מחשבים את הפולינום $c = (p * r * h + m) \bmod q$.
 - הטקסט המוצפן הוא c .
- 3. פענוח:
 - כדי לפענח את הטקסט המוצפן c , מחשבים:
 - $a = (f * c) \bmod q$
 - $m = (a * g) \bmod p$

מבוא

- הצפנת רבין היא מערכת הצפנה אסימטרית שהומצאה על ידי מיכאל רבין בשנת 1979. המערכת מבוססת על הקושי של בעיית שארית הריבוע המודולרית, מה שהופך אותה לאחת השיטות הקריפטוגרפיות הבטוחות מבחינה תאורטית.
- שלא כמו RSA שהביטחון שלו תלוי בבעיית פירוק לגורמים (שהיא קשה אך לא בהכרח בלתי אפשרית לפתרון), הביטחון של הצפנת רבין מוכח מתמטית כלומר, אם ניתן לפענח את ההצפנה ביעילות, ניתן גם לפרק לגורמים מספרים שלמים גדולים, מה שנחשב לבעיה קשה מבחינה חישובית.
- צופן מקאליס הוא מערכת הצפנה אסימטרית שפותחה על ידי רוברט מקאליס בשנת 1978. היא נחשבת לאחת ממערכות ההצפנה הפוסט-קוונטיות המבטיחות ביותר, שכן היא עמידה בפני התקפות של מחשבים קוונטיים עתידיים.
- הצפנת התרמיל היא שיטת הצפנה אסימטרית מוקדמת שהוצעה בשנת 1978 על ידי רלף מרקל ומרטין הלמן. הרעיון המרכזי מבוסס על בעיית התרמיל Knapsack Problem שהיא בעיה קשה חישובית. עם זאת, האלגוריתם עצמו התגלה כפריץ והוא אינו בשימוש כיום בשל התקפה מוצלחת של אדלמן, שניר וטר'אן (1982).
- הרעיון המרכזי:
 - דמיינו שיש לכם תרמיל עם חפצים שונים, שלכל אחד משקל שונה. המפתח הציבורי הוא רשימה של משקלים, והמפתח הפרטי הוא "סוד" שמאפשר לגלות אילו חפצים נבחרו כדי להגיע למשקל מסוים.
- פרוטוקול דיפי-הלמן Diffie-Hellman Key Exchange הוא שיטה לחילופי מפתחות קריפטוגרפיים בצורה מאובטחת על גבי ערוץ תקשורת שאינו בטוח. הפרוטוקול פותח בשנת 1976 על ידי ויטפילד דיפי ומרטין הלמן, והוא נחשב לאבן יסוד בקריפטוגרפיה מודרנית.
- המטרה של הפרוטוקול
- המטרה היא ששני צדדים (למשל, אליס ובוב) יוכלו לייצר מפתח סודי משותף ללא צורך בהעברת המפתח עצמו על גבי הרשת. המפתח הסודי הזה ישמש להצפנה סימטרית כגון AES לתקשורת מאובטחת.

עקרונות מרכזיים

- הצפנה אסימטרית: צופן רבין משתמש בשני מפתחות שונים - מפתח ציבורי ומפתח פרטי. המפתח הציבורי משמש להצפנה, והמפתח הפרטי משמש לפענוח.
- בעיית פירוק לגורמים: ביטחון הצופן מבוסס על הקושי שבפירוק מספרים גדולים לגורמים ראשוניים.
- שורשים ריבועיים: תהליך הפענוח בצופן רבין כולל מציאת שורשים ריבועיים מודולו מספר פריק.
- הצפנה אסימטרית: בדומה ל-RSA גם קריימר-שופ משתמש בשני מפתחות שונים: מפתח ציבורי (משמש להצפנה) ומפתח פרטי (משמש לפענוח).
- בעיית הלוגריתם הדיסקרטי: ביטחון הצופן מבוסס על הקושי שבפתרון בעיית הלוגריתם הדיסקרטי, בדומה לצופן אל גמאל.
- הוכחת אפס ידע: קריימר-שופ משלב מנגנון של הוכחת אפס ידע, המאפשר לצדדים להוכיח שהם מחזיקים במידע מסוים מבלי לחשוף אותו.
- 1. הצפנה אסימטרית: צופן מקאליס משתמש בשני מפתחות שונים - מפתח ציבורי ומפתח פרטי. המפתח הציבורי משמש להצפנה, והמפתח הפרטי משמש לפענוח.

- 2. קוד תיקון שגיאות: צופן מקאליס מבוסס על קוד תיקון שגיאות, שהוא קוד מתמטי שמאפשר זיהוי ותיקון שגיאות בהעברת מידע.
- 3. בעיית פענוח קוד ליניארי: ביטחון הצופן מבוסס על הקושי שבפענוח קוד ליניארי כללי, בעיה מתמטית קשה שדורשת כוח חישוב עצום.

התהליך שעברנו

- יצירת מפתחות:
- בחרים שני מספרים ראשוניים גדולים ואקראיים p ו- q
- מחשבים את $n = p * q$
- המפתח הציבורי הוא n .
- המפתח הפרטי הוא (p, q)
- הצפנה:
- כדי להצפין הודעה m ממחשבים את $c = m^2 \bmod n$
- הטקסט המוצפן הוא c
- פענוח:
- כדי לפענח את הטקסט המוצפן c יש למצוא את השורשים הריבועיים של c מודולו n
- בעזרת המפתח הפרטי (p, q) ניתן למצוא ארבעה שורשים ריבועיים אפשריים.
- יש לבחור את השורש המתאים, שהוא ההודעה המקורית m

יוצא לנו למעשה

- הערות:
- בדוגמה זו, השתמשנו במספרים קטנים מאוד כדי לפשט את ההבנה.
- במערכות הצפנה אמיתיות, משתמשים במספרים ראשוניים גדולים מאוד, בעלי מאות ספרות.
- תהליך מציאת השורשים הריבועיים עשוי להיות מורכב יותר כאשר מדובר במספרים גדולים.
- לסיכום:
- הצפנו את ההודעה $m = 7$ וקיבלנו את הטקסט המוצפן $c = 49$ לאחר מכן, פענחנו את הטקסט המוצפן וקיבלנו את ההודעה המקורית $m = 7$

יצירת מפתחות

- 1. בחירת פרמטרים: בחרים מספר ראשוני גדול p יוצר ראשוני q ומספר שלם g שהוא יוצר של החבורה הכפלית של \mathbb{Z}/p
- 2. בחירת מפתח פרטי: בחרים שני מספרים שלמים אקראיים x_1 ו- x_2 .
- 3. חישוב מפתח ציבורי: מחשבים את $y_1 = g^{x_1} \bmod p$ ו- $y_2 = g^{x_2} \bmod p$
- 4. המפתח הציבורי: (p, g, y_1, y_2)
- 5. המפתח הפרטי: (x_1, x_2)
- 1. בחירת פרמטרים: בחרים פרמטרים מתאימים, כמו אורך הקוד ויכולת תיקון השגיאות.
- 2. יצירת מטריצות: יוצרים מטריצה יוצרת G ומטריצת בדיקה H
- 3. המפתח הציבורי: המטריצה G
- 4. המפתח הפרטי: מידע נוסף שמאפשר פענוח יעיל, כמו מטריצת הפענוח או מבנה הקוד.

הצפנה

- 1. בחירת הודעה: ההודעה m צריכה להיות מספר שלם בין 0 ל- $p-1$
- 2. בחירת מספר אקראי: בחרים מספר שלם אקראי r .
- 3. חישוב הטקסט המוצפן:
 - $u_1 = g^r \bmod p$
 - $u_2 = y_1^r \bmod p$
 - $u_3 = y_2^r \bmod p$

$$v = m * (y1^r)^{x2} \bmod p$$

• הטקסט המוצפן: $(u1, u2, u3, v)$.

1. הכנת ההודעה: ההודעה m מיוצגת כווקטור.

2. הוספת שגיאות: מוסיפים וקטור שגיאות אקראי z .

3. חישוב הטקסט המוצפן: $c = mG + z$.

פענוח

• חישוב ההודעה המקורית:

$$m = v * (u2^{(x1)} * u3^{(-1)}) \bmod p$$

• ביטחון:

• צופן קריימר-שופ נחשב לאחד מצפני ההצפנה האסימטריות הבטוחים ביותר כיום. הוא מספק רמת אבטחה גבוהה מאוד, ועמיד בפני התקפות ידועות, כולל התקפות של מחשבים קוונטיים.

1. תיקון שגיאות: משתמשים במפתח הפרטי כדי לתקן את השגיאות בטקסט המוצפן c .

2. חילוץ ההודעה: לאחר תיקון השגיאות, מקבלים את ההודעה המקורית m .

דוגמא - המשך

• הצפנה:

• בחירת הודעה:

$$m = 15$$

• בחירת מספר אקראי:

$$r = 3$$

• חישוב הטקסט המוצפן:

$$u1 = g^r \bmod p = 5^3 \bmod 23 = 125 \bmod 23 = 10$$

$$u2 = y1^r \bmod p = 19^3 \bmod 23 = 6859 \bmod 23 = 16$$

$$u3 = y2^r \bmod p = 2^3 \bmod 23 = 8$$

$$v = m * (y1^r)^{x2} \bmod p = 15 * (19^3)^{11} \bmod 23 = 15 * 6859^{11} \bmod 23 = 15 * 16^{11} \bmod 23 = 15 * 2048 \bmod 23 = 15 * 2 \bmod 23 = 30 \bmod 23 = 7$$

• הטקסט המוצפן: $(7, 8, 16, 10)$

• הסבר על: $11^{6859} * 15 \bmod 23$

• נתחיל עם החזקה: 6859 בחזקת 11 . זהו מספר עצום, אך לנו מעניין רק השארית שלו מחלוקה ב- 23 .

• בגלל שאנו משתמשים בחשבון מודולרי לכן, במקום לחשב 11^{6859} ואז לחלק ב- 23 , נמצא את השארית של 6859 מחלוקה ב- 23 , ואז נעלה את השארית הזאת בחזקת 11 .

• לכן, נקבל $11^{16} \bmod 23$

• כעת נעלה את 16 בחזקת 11 . גם כאן נקפיד למצוא את השארית מחלוקה ב- 23 בכל שלב, כדי לא לעבוד עם מספרים עצומים.

• אחרי כמה פעולות של העלאה בריבוע ומציאת שארית, נקבל $16^{11} \bmod 23 = 2048 \bmod 23$

$$15 * 2048 \bmod 23$$

• נכפיל את 15 ב- 2048 . שוב, מעניין אותנו רק השארית מחלוקה ב- 23 .

$$2 * 2048 \bmod 23$$

$$15 * 2 \bmod 23$$

$$15 * 2 \bmod 23 = 30 \bmod 23$$

• נכפיל את 15 ב- 2 ונקבל 30 .

$$30 \bmod 23 = 7$$

• השארית של 30 מחלוקה ב- 23 היא 7 .

• פענוח:

• חישוב ההודעה המקורית:

$$m = v * (u2^{(x1)} * u3^{(-1)}) \bmod p = 7 * (16^7 * 8^{(-1)}) \bmod 23$$

$$(3 \bmod 23 = 1 * 8 \text{ כי } 3 \bmod 23 = 3 \text{ } (-1)^8 \circ$$

$$m = 7 * (16^7 * 3) \bmod 23 = 7 * (279936 * 3) \bmod 23 = 7 * 839808 \bmod 23 = 7 * 15 \bmod 23 = 105 \bmod 23 = 15$$

• ההודעה המקורית: 15

• לסיכום:

• הצפנו את ההודעה $m = 15$ וקיבלנו את הטקסט המוצפן (7, 8, 16, 10). לאחר מכן, פענחנו את הטקסט המוצפן

וקיבלנו את ההודעה המקורית $m = 15$

1. תיקון שגיאות:

• בעזרת המפתח הפרטי (לא נכלל בדוגמה זו), ניתן לזהות ולתקן את השגיאה בטקסט המוצפן.

2. חילוץ ההודעה:

• לאחר תיקון השגיאות, מתקבלת ההודעה המקורית $m = (1 \ 0 \ 1 \ 1)$.

2. הצפנה:

• הודעה:

"message = "cat"

• ייצוג בינארי:

message_binary = (01100011, 01100001, 01110100) ◦

$$\text{ciphertext} = (0 * 34 + 1 * 51 + 1 * 14 + 0 * 23 + 0 * 80, 0 * 34 + 1 * 51 + 1 * 14 + 0 * 23 + 0 * 80, 0 * 34 + 1 * 51 + 1 * 14 + 1 * 23 + 0 * 80) = (65, 65, 88)$$

3. פענוח:

• חישוב ההופכי הכפלי של a מודולו m

$$(a_{\text{inv}} = 62 \text{ כי } 62 * 17 \bmod 105 = 1) \text{ מתי יהיה שווה לשארית } 1$$

• חישוב t :

$$t = (\text{ciphertext} * a_{\text{inv}}) \bmod m = (65 * 62, 65 * 62, 88 * 62) \bmod 105 = (4030, 4030, 5456) \bmod 105 = (2, 3, 7)$$

$$(a_{\text{inv}} = 62 \text{ כי } 62 * 17 \bmod 105 = 1) \text{ מתי יהיה שווה לשארית } 1$$

• הגדרה:

• ההופכי הכפלי של מספר a מודולו m הוא מספר b כך ש:

$$a * b \equiv 1 \pmod{m}$$

• במילים פשוטות, זהו מספר שאם נכפיל אותו במספר המקורי a נקבל שארית 1 לאחר חילוק ב- m .

• הסבר על הפענול של T :

$$\text{ciphertext} = (65, 65, 88)$$

$$a_{\text{inv}} = 62$$

$$m = 105$$

$$t = (65 * 62 \bmod 105, 65 * 62 \bmod 105, 88 * 62 \bmod 105) = (2, 3, 7)$$

1. המרת t לייצוג בינארי:

• המטרה: לקבל רצף של 0 ו-1 שנוכל לפענח.

• השיטה: ממירים כל מספר ב- t לייצוג בינארי שלו.

• דוגמה:

$$010 = 2$$

$$011 = 3$$

$$111 = 7$$

$$t = (2, 3, 7) = (010, 011, 111)$$

2. פענוח ההודעה הבינארית:

• המטרה: לקבל את ההודעה המקורית.

$$\text{message_binary} = (01100011, 01100001, 01110100)$$

• הקשר: כל מספר ב- t מתאים לרצף בינארי שמרכיב אות.

• ההמרה:

01100011 <- 2 °

01100001 <- 3 °

01110100 <- 7 °

"c" = 01100011 °

"a" = 01100001 °

"t" = 01110100 °

דוגמא - המשך

• הצפנה:

• הודעה: $m = (1\ 0\ 1\ 1)$

• וקטור שגיאות: $z = (0\ 0\ 1\ 0\ 0\ 0\ 0)$

• חישוב הטקסט המוצפן:

$c = mG + z = (1\ 0\ 1\ 1) * G + (0\ 0\ 1\ 0\ 0\ 0\ 0) = (1\ 1\ 0\ 1\ 1\ 0\ 1) + (0\ 0\ 1\ 0\ 0\ 0\ 0) = (1\ 1\ 1\ 1\ 1\ 0\ 1)$ °

• הטקסט המוצפן: $c = (1\ 1\ 1\ 1\ 1\ 0\ 1)$

כיצד זה עובד?

1. יצירת מפתחות:

• המפתח הפרטי מורכב מרצף של מספרים שנקראים "סופר-גדלים" Superincreasing Sequence כל מספר גדול מסכום כל המספרים הקודמים לו.

• המפתח הציבורי נוצר על ידי "ערבוב" המספרים בסופר-גדלים בעזרת שני מספרים נוספים: כופל ומודולו.

2. הצפנה:

• כדי להצפין הודעה, ממירים אותה לסדרת מספרים בינאריים (0 ו-1).

• כל מספר בינארי מוכפל במספר המתאים לו במפתח הציבורי.

• סכום התוצאות הוא הטקסט המוצפן.

3. פענוח:

• בעזרת המפתח הפרטי, ניתן "לפתור" את בעיית התרמיל ולקבל את ההודעה המקורית.

סוגים ושימושים

• למה זה בטוח?

• בעיית "פתרון תרמיל" באופן כללי היא בעיה קשה מאוד לפתרון, במיוחד כאשר מדובר במספרים גדולים.

• סוגי הצפנת תרמיל:

• קיימים סוגים שונים של הצפנת תרמיל, כאשר המפורסמת ביותר היא הצפנת מרקל-הלמן.

• שימושים:

• הצפנת תרמיל יכולה לשמש למגוון מטרות, כגון:

• הצפנת הודעות דואר אלקטרוני

• חתימה דיגיטלית

• העברת מידע רגיש

הרקע לקריפטוגרפיה אסימטרית ולצורך בדיפי-הלמן

• לפני המצאת דיפי-הלמן (1976), תקשורת מאובטחת בין שני צדדים דרשה חלוקה מראש של מפתח סודי משותף

– בעיה לוגיסטית משמעותית. אם שני צדדים רצו לתקשר, הם היו צריכים למצוא דרך בטוחה להחליף ביניהם את

המפתח הסודי מראש, מה שהגביל את היכולת לקיים תקשורת מאובטחת על פני ערוצים ציבוריים.

• הפתרון הגיע עם פיתוח קריפטוגרפיה אסימטרית, שבה ניתן ליצור תקשורת מאובטחת גם בין צדדים שמעולם לא

נפגשו קודם לכן. דיפי-הלמן היה האלגוריתם הראשון שהציע דרך בטוחה להחלפת מפתחות על גבי רשת פומבית.

איך הוא בעצם עובד

- בואו נדמיין שני אנשים, אליס ובוב, שרוצים להחליף ביניהם מידע מוצפן.
- שלב 1: הסכמה על פרמטרים
- אליס ובוב מסכימים על שני מספרים:
 - p : מספר ראשוני גדול.
 - g : מספר שלם קטן יותר, שנקרא "יוצר".
- שלב 2: בחירת מספרים סודיים
- אליס בוחרת מספר סודי אקראי, a
- בוב בוחר מספר סודי אקראי, b
- שלב 3: שליחת מידע
- אליס מחשבת את הערך $A = g^a \mod p$ ושולחת אותו לבוב.
- בוב מחשב את הערך $B = g^b \mod p$ ושולח אותו לאליס.
- שלב 4: חישוב המפתח המשותף
- אליס מחשבת את המפתח המשותף: $K = B^a \mod p$
- בוב מחשב את המפתח המשותף: $K = A^b \mod p$
- שימו לב:
- גם אליס וגם בוב הגיעו לאותו מפתח משותף, K
- אדם זר שמצותת לערוץ התקשורת לא יכול לחשב את המפתח המשותף, מכיוון שהוא לא יודע את המספרים הסודיים a ו- b

ועוד יותר טוב

- דוגמה
- נניח כי $p = 23$ ו- $g = 5$
- אליס בוחרת $a = 6$ ומשתמשת בנוסחה $A = 5^6 \mod 23 = 8$
- בוב בוחר $b = 15$ ומושתמש בנוסחה $B = 5^{15} \mod 23 = 19$
- אליס שולחת לבוב את המספר 8 ובוב שולח לאליס את המספר 19.
- אליס מחשבת את המפתח המשותף על ידי $\mod 23 = 2 \cdot 6^{19}$
- בוב מחשב את המפתח המשותף על ידי $\mod 23 = 2 \cdot 15^8$
- שני הצדדים הגיעו לאותו מפתח משותף, למרות העובדה שהם לא שיתפו ביניהם אף מידע פרט למספרים 8 ו-19.

שיפור אבטחת דיפי-הלמן

- כדי למנוע מתקפת MITM אפשר לשלב את הפרוטוקול עם חתימות דיגיטליות כגון RSA או ECC כדי לוודא שהצדדים הם אותנטיים. פרוטוקולים משופרים כוללים:
- DHE (Diffie-Hellman Ephemeral): גרסה שבה מחליפים מפתחות לכל סשן, מגבירה את האבטחה.
- ECDH (Elliptic Curve Diffie-Hellman): גרסה המשתמשת בעקומות אליפטיות ECC להקטנת גודל המפתחות תוך שמירה על בטיחות.

הגדרה

- עקרון קרקהופס Kerckhoffs's Principle הוא עיקרון יסודי בקריפטוגרפיה, הקובע:
- "מערכת הצפנה צריכה להיות בטוחה גם אם כל פרטי האלגוריתם שלה ידועים לציבור, למעט המפתח הסודי."
- במילים אחרות, הביטחון של מערכת הצפנה לא צריך להיות תלוי בסודיות של האלגוריתם עצמו, אלא רק במפתח ההצפנה.

למה זה חשוב?

- 1. עמידות בפני הדלפות מידע

- אם מערכת הצפנה מבוססת על סודיות האלגוריתם, דליפה אחת עלולה לגרום לקריסתה.
- אם רק המפתח צריך להיות סודי, ניתן להחליף אותו במקרה של חשיפה בלי להחליף את כל המערכת.
- 2. נוחות בשימוש ובקרה
 - אם אלגוריתם ההצפנה פומבי, ניתן לנתח ולבדוק את רמת האבטחה שלו.
 - מומחי אבטחה יכולים לזהות חולשות ולשפר את האלגוריתם.
- 3. אבטחה מבוססת מתמטיקה ולא על "ערפול"
 - אם האלגוריתם סודי, קשה להעריך עד כמה הוא באמת מאובטח.
 - אם האלגוריתם גלוי, ניתן להוכיח מתמטית את רמת הבטיחות שלו.

עקרון קרקהופס מול "ערפול"

- "ערפול": Obscurity הוא מצב שבו מערכת מסתמכת על כך שהאויב לא מכיר את האלגוריתם שלה כדי להישאר בטוחה.
- הבעיה בערפול:
 - אם התוקף מגלה את האלגוריתם – כל ההגנה קורסת.
 - קשה לבדוק את רמת האבטחה של מערכת סודית.
 - לרוב, ערפול מספק אשליית אבטחה, ולא אבטחה אמיתית.
 - קרקהופס אומר את ההפך:
 - אין להסתמך על סודיות האלגוריתם – רק המפתח צריך להיות סודי.
 - מערכות קריפטוגרפיות צריכות להיות פומביות ונבדקות על ידי מומחים כדי להבטיח שהן חזקות.
 - יש להסתמך על בעיות מתמטיות קשות ולא על הסתרת קוד.

דוגמאות ליישום עקרון קרקהופס

- הקריפטוגרפיה המודרנית מבוססת על עקרון קרקהופס. כל האלגוריתמים הקריפטוגרפיים הבטוחים בעולם הם כאלה שהאלגוריתם שלהם גלוי לכולם, והביטחון נשען אך ורק על סודיות המפתח.
- אלגוריתמים העומדים בעיקרון קרקהופס:
 - AES (Advanced Encryption Standard)
 - RSA (Rivest-Shamir-Adleman)
 - ECC (Elliptic Curve Cryptography)
 - ElGamal
 - NTRU
 - Diffie-Hellman
 - SHA-256 (פונקציית גיבוב קריפטוגרפית)
- אלגוריתמים שלא עומדים בעיקרון קרקהופס (ולכן לא מומלצים):
 - הצפנות קנייניות שלא עברו סקירה ציבורית
 - מערכות המסתמכות על "ערפול" במקום על בעיות מתמטיות קשות
 - מערכות שבהן "אם הקוד דולף – כל ההגנה קורסת"

דוגמאות מעשיות לעקרון קרקהופס

- דוגמא 1: פרוטוקול TLS הצפנת HTTPS
- TLS לשעבר SSL הוא תקן הצפנה לאינטרנט שבו כל המנגנונים הציבוריים גלויים וידועים, ורק המפתחות משתנים בין המשתמשים. אם TLS לא היה מבוסס על עקרון קרקהופס, כל עדכון של דפדפן היה עלול לסכן את כל האינטרנט.
- דוגמא 2: מערכת PGP הצפנת אימיילים
- האלגוריתמים שבהם משתמשים ב-PGP פומביים וידועים (RSA, AES וכו').
- למרות שהקוד פתוח, אם התוקף לא יודע את המפתח הפרטי – אין לו דרך לפענח הודעה מוצפנת.

למה קריפטוגרפיה מאמינים בעקרון קרקוהופס?

- אלגוריתמים שעוברים בדיקה ציבורית נוטים להיות בטוחים יותר. אם מערכת ההצפנה תלויה רק בערפול, היא מועדת לכישלון אם האלגוריתם ידלף. תקנים פתוחים מאפשרים שימוש בטוח וחופשי בהצפנה, ללא תלות ביצרן יחיד.
- דוגמה מציאותית:
- TLS הפרוטוקול שמאבטח את האינטרנט מבוסס על הצפנות שנבדקו היטב.
- מערכות כמו Wi-Fi WEP שהסתמכו על ערפול – נפרצו תוך שנים ספורות בלבד!

מתי כן משתמשים בערפול בהצפנה?

- יש מקרים שבהם ערפול יכול לעזור, אבל לא כתחליף לקריפטוגרפיה חזקה.
- לדוגמה: הסתרת מפתחות הצפנה בזיכרון של תוכנות – כדי להקשות על גניבה דרך תקיפות צד. ערפול קוד של תוכנות מסחריות – כדי להקשות על הנדסה לאחור. הגנה זמנית על מידע מסווג – בצבא או במערכות רגילות.
- אבל: ערפול לבדו אינו דרך בטוחה להגן על נתונים!

מהי פונקציית גיבוב Hash Function

- פונקציית גיבוב היא פונקציה קריפטוגרפית שממירה קלט (באורך משתנה) לפלט בגודל קבוע.
- הפלט מכונה "ערך גיבוב" או "Digest"
- פונקציות גיבוב קריפטוגרפיות משמשות לאימות נתונים, חתימות דיגיטליות, אחסון סיסמאות ועוד.
- דוגמה:
- אם נפעיל את פונקציית הגיבוב SHA-256 על המילה hello נקבל:
- 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
- גם אם נשנה אות אחת בקלט, נקבל תוצאה שונה לחלוטין!

מאפיינים של פונקציית גיבוב קריפטוגרפית

1. קביעה: Deterministic אותו קלט תמיד ייתן את אותו פלט.
2. מהירות חישוב: פונקציית גיבוב צריכה להיות מהירה לחישוב
3. עמידות להתנגשויות: Collision Resistance קשה למצוא שני קלטים שונים עם אותו פלט.
4. עמידות לחישוב קדם-תמונה: Preimage Resistance אי אפשר לשחזר את הקלט מהפלט.
5. עמידות לתקיפות יום הולדת: Birthday Attack Resistance הסיכוי למצוא שני קלטים עם אותו ערך גיבוב נמוך מאוד

דוגמאות לפונקציות גיבוב נפוצות

- MD5: ישן, אך לא בטוח (אפשר לפרוץ אותו).
- HA-1: בעבר היה בשימוש, אך גם נחשב לא בטוח.
- SHA-256 / SHA-3 : מאובטח ונמצא בשימוש כיום.
- BLAKE2 / BLAKE3: מהיר יותר מ-SHA ומשמש במערכות מתקדמות.

דוגמאות מעשיות לפונקציות גיבוב

- דוגמה 1 – שמירת סיסמאות בבסיס נתונים
- כאשר משתמש יוצר חשבון, במקום לשמור את הסיסמה שלו, נשמור את הגיבוב שלה. למשל, אם הסיסמה היא password123 נחשב את:
- SHA-256("password123") =
- ef92b778ba93f49a4e4ddae84a2c807789618ca2734696ff97a8a6bcae1b3bc6
- בזמן התחברות, המערכת משווה את ערך הגיבוב שהוזן לזה ששמור בבסיס הנתונים.

- למה לא מומלץ להשתמש בגיבוב בלבד לסיסמאות? כי האקרים יכולים לבצע תקיפת טבלאות קשת Rainbow Tables הפתרון: הוספת salt ערך אקראי לכל סיסמה לפני הגיבוב.
- דוגמה 2 – שימוש בגיבוב באימות מסרים HMAC:
- מערכת תקשורת רוצה לוודא שההודעה לא שונתה בדרך
- שולח מחשב SHA-256 של ההודעה ושולח גם את הגיבוב יחד עם ההודעה
- המקבל מחשב את ה-Hash מחדש ומשווה לערך שהתקבל. אם הם זהים – ההודעה לא שונתה.

פונקציות גיבוב קריפטוגרפיות מול פונקציות גיבוב כלליות

- ישנן פונקציות גיבוב כלליות שמשמשות לבדיקת תקינות נתונים כגון CRC32 אך פונקציות גיבוב קריפטוגרפיות נדרשות להיות עמידות בפני תקיפות קריפטוגרפיות ולספק אבטחה גבוהה יותר.

תקיפות על פונקציות גיבוב

1. תקיפת קדם-תמונה Preimage Attack: המטרה: למצוא קלט x כך שהגיבוב שלו יהיה $h = H(x)$ נתון. מסובך אם הפונקציה חזקה מספיק (דורש כוח חישוב עצום). פונקציות כמו SHA-256 ו-SHA-3 נחשבות עמידות לתקיפה זו.
2. תקיפת קדם-תמונה שנייה Second Preimage Attack: המטרה: נתון קלט x_1 למצוא $x_2 \neq x_1$ כך ש- $H(x_1) = H(x_2)$ קשה כמו תקיפת קדם-תמונה, אך מתקפות מתקדמות עשויות להפחית את המאמץ החישובי.
3. תקיפת התנגשות Collision Attack: המטרה: למצוא שני קלטים שונים $x_1 \neq x_2$ כך ש- $H(x_1) = H(x_2)$ לפי עקרון יום ההולדת, הסיכוי להתנגשות עולה עם מספר הקלטים. פונקציות כמו MD5 ו-SHA-1 כבר הוכח שהן חלשות נגד תקיפה זו!

פונקציות גיבוב וסיסמאות - מדוע אסור להשתמש בגיבוב בלבד?

- שמירת סיסמאות כערך גיבוב בלבד עלולה להיות לא בטוחה! האקרים יכולים להשתמש בטכניקות כגון: טבלאות קשת Rainbow Tables מסדי נתונים מוכנים של גיבובים וסיסמאות תואמות. Brute-force בדיקה שיטתית של כל האפשרויות.
- הפתרון: Salt: הוספת מחרוזת אקראית לפני הגיבוב, כך ששתי סיסמאות זהות לא יניבו אותו ערך גיבוב.
- פונקציות גיבוב ייעודיות לסיסמאות:
 - PBKDF2
 - bcrypt
 - Argon2 (הטוב ביותר כיום!)

מהו מחולל פסבדו-רנדומלי קריפטוגרפי CSPRNG

- מחולל פסבדו-רנדומלי קריפטוגרפי – Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) (הוא אלגוריתם שמייצר מספרים אקראיים באופן שנראה אקראי לחלוטין, אך מבוסס על חישובים דטרמיניסטיים. ההבדל בין CSPRNG לבין מחולל פסבדו-רנדומלי רגיל PRNG הוא בכך ש-CSPRNG חייב להיות בטוח לשימוש בקריפטוגרפיה, כלומר:
 - בלתי ניתן לחיזוי – אי אפשר לנחש את המספר הבא מתוך המספרים הקודמים.
 - עמיד למתקפות – תוקף שמקבל גישה לכמה פלטים של המחולל לא יכול לשחזר את מצב המחולל או לחזות ערכים עתידיים.
 - מבוסס על אנטרופיה גבוהה – משתמש במקור אנטרופיה חזק (למשל רעש תרמי, תנועות עכבר וכו').

איך עובד CSPRNG?

1. מקור אנטרופיה: Seed Source נבחר ערך ראשוני אקראי ממקור פיזי בלתי צפוי, כגון רעש תרמי במעבד או זמני תגובה ברשת.
2. פונקציה חד-כיוונית: הערך הראשוני עובר דרך פונקציה מתמטית, כמו פונקציית גיבוב או הצפנה אסימטרית, כדי להפוך אותו לבלתי צפוי.

- 3. הרחבת האקראיות: אלגוריתם מרחיב את הביטים האקראיים לכמות גדולה יותר של ביטים באופן פסבדו-אקראי.
- 4. Output: מספרים שנראים אקראיים אך הם קריפטוגרפית בטוחים.

תכונות חשובות של CSPRNG

- כדי שמחולל פסבדו-רנדומלי ייחשב קריפטוגרפי, עליו לעמוד בדרישות הבאות:
 - א. עמידות קדימה Forward Security: אם תוקף נחשף לחלק מהפלט של המחולל, אסור לו להיות מסוגל להסיק על ערכים קודמים. כלומר, גם אם מישהו גילה את אחד המספרים האקראיים שיצאו, הוא לא יכול לשחזר את המספרים שנוצרו לפניו.
 - ב. עמידות אחורה Backward Security: אם מחולל נפרץ ותוקף נחשף למצב הפנימי הנוכחי שלו, הוא לא אמור להיות מסוגל לחשב את כל הפלטים העתידיים. הדרך להתמודד עם זה היא Reseeding הזרקת אנטרופיה חדשה למחולל.
 - ג. חסינות לניבוי Prediction Resistance: אפילו אם תוקף יודע את כל האלגוריתם של המחולל, אסור לו להיות מסוגל לנבא את הפלט הבא ללא ידיעת ה-Seed.

איך בודקים אם CSPRNG בטוח?

- כדי לוודא שמחולל עומד בדרישות קריפטוגרפיות, משתמשים במבחנים סטטיסטיים:
 - מבחן האנטרופיה Entropy Test: מודד כמה מידע חדש (אקראי) קיים בזרם הפלט.
 - מבחן ההתפלגות Chi-Square Test: בודק אם הפלט מתפלג בצורה אחידה Uniform Distribution.
 - מבחן הרצת אפסים ואחדים Runs Test: בודק אם מופעים רצופים של 0 או 1 מופיעים בתדירות צפויה.
 - מבחן NIST SP800-22: סדרת מבחנים סטטיסטיים שמספקים אישור לקריפטוגרפיה.

חסרונות של מחוללים פסבדו-רנדומליים קריפטוגרפיים

- למרות שהם נחשבים בטוחים, CSPRNG לא חף מחסרונות:
 1. תלות במקור אנטרופיה: אם המחולל מקבל Seed חלש (למשל, מספרים שנוצרו ע"י שעון מערכת), הוא לא בטוח.
 2. צורך בניהול Seed נכון: אם המערכת מאחסנת את ה-Seed באופן גלוי, תוקף יכול לשחזר את כל הפלטים.
 3. ביצועים: מחוללים קריפטוגרפיים לעיתים איטיים יותר ממחוללים רגילים כי הם משתמשים בפונקציות קריפטוגרפיות כבדות.

התקפות אפשריות על CSPRNG

- א. התקפת שחזור מצב פנימי State Recovery Attack:
 - אם מחולל משתמש באלגוריתם חלש כמו Mersenne Twister יכול להסיק מהמצב הנוכחי של המחולל מה יהיו הערכים הבאים.
 - הגנה: שימוש ב-CSPRNG שעומד בתקנים כמו AES - CTR DRBG.
- ב. התקפת זריעת Seed חלשה Weak Seeding Attack:
 - אם ה-Seed מגיע משעון מערכת או מספרי PID של תוכנות, ניתן לחזות את המספרים העתידיים.
 - הגנה: שימוש במקור אנטרופיה חזק כמו תנועות עכבר, רעש תרמי, או קלט ממשתמשים.
- ג. התקפת קורלציה Correlation Attack:
 - אם מחולל מבוסס על פונקציה ליניארית, אפשר לזהות דפוסים ולחזות את המספרים הבאים.
 - הגנה: שימוש בפונקציות חד-כיווניות כמו SHA-256 להגדלת הבלתי-צפויות של הפלט.

מה קורה אם מחולל קריפטוגרפי נשבר?

- דוגמאות לכשלי מחוללים בהיסטוריה:
 1. Windows XP Random Generator השתמש ב-Seed חלש, מה שאיפשר ניבוי מפתחות של משתמשים.

- 2: OpenSSL Debian Bug (2008) באג הפחית את האנטרופיה של מחולל המספרים האקראיים, מה שאיפשר פריצה למפתחות הצפנה.
- 3: Dual_EC_DRBG: מחולל רשמי של NIST שהתברר כחשוד בכך שהכיל דלת אחורית אפשרית עבור ה-NSA.

דוגמאות לקוד של מחולל פסבדו-רנדומלי קריפטוגרפי

- יצירת מספרים אקראיים בצורה בטוחה בפיתון:
- `import secrets`
- יצירת מספר אקראי בין 0 ל-100
- `random_number = secrets.randbelow(101)`
- `print(random_number)`
- יצירת מחרוזת אקראית של 16 בתים (כגון טוקן אימות)
- `token = secrets.token_hex(16)`
- `print(token)`
- יצירת מפתח AES אקראי:
- `import os`
- יצירת מפתח אקראי של 256 ביט (32 בתים)
- `key = os.urandom(32)`
- `print(key.hex())`

תרגול עצמי (תמצית)

1. מהו צופן בלוקים, ומה ההבדל בינו לבין צופן זרם?
 2. מהו גודל הבלוק הסטנדרטי של AES?
 3. מהי המטרה של ריפוד Padding בצופן בלוקים?
 4. מה ההבדל בין טקסט קריא Plaintext לטקסט מוצפן Ciphertext?
 5. איזה מצב פעולה Mode of Operation בצופן בלוקים לא מומלץ בגלל שהוא חושף דפוסים בטקסט המוצפן?
 6. מהו וקטור ייזום IV ובאיזה מצב פעולה הוא נדרש?
1. צופן בלוקים הוא שיטת הצפנה שבה הנתונים מחולקים לבלוקים בגודל קבוע (למשל 128 ביטים) וכל בלוק מוצפן בנפרד. לעומת זאת, צופן זרם Stream Cipher מצפין את הנתונים בזרימה ביט אחר ביט או בתים בודדים, ללא חלוקה לבלוקים קבועים.
 2. ב-AES הבלוק תמיד בגודל 128 ביטים (16 בתים).
 3. אם הטקסט הקריא plaintext אינו מתחלק בצורה מושלמת בגודל הבלוק, יש להוסיף ריפוד כדי להשלים את הבלוק האחרון לגודל הנדרש. למשל, אם גודל הבלוק הוא 128 ביטים והטקסט הוא 90 ביטים, נוסיף 38 ביטים של ריפוד.
 4. Plaintext: המידע המקורי לפני ההצפנה, ניתן לקריאה.
 5. מצב הפעולה ECB (Electronic Codebook) אינו מומלץ, כי הוא מצפין כל בלוק בנפרד, כך שבלוקים זהים בטקסט הקריא יישארו זהים בטקסט המוצפן, מה שחושף דפוסים מסוכנים.
 6. וקטור ייזום IV - Initialization Vector הוא ערך אקראי שמתווסף לבלוק הראשון כדי למנוע חזרתיות ולשפר אבטחה. הוא נדרש במצבי פעולה כמו CFB, OFB, GCM-ICBC.
1. מהו צופן זרם, ובמה הוא שונה מצופן בלוקים?
 2. כיצד מתבצע תהליך ההצפנה בצופן זרם?
 3. מהי הפעולה המתמטית העיקרית שמשמשת בצופן זרם?
 4. למה חשוב לא להשתמש באותו זרם מפתח Keystream פעמיים?
 5. מהם היתרונות של צופן זרם?
1. צופן זרם מצפין את המידע ביט אחר ביט או בית אחר בית באופן רציף, בעוד צופן בלוקים מצפין קבוצות קבועות של ביטים (בלוקים), למשל 128 ביטים ב-AES. בנוסף צופן זרם מהיר יותר ומתאים לתקשורת בזמן אמת, בעוד צופן בלוקים מתאים להצפנת קבצים ונתונים גדולים.