

סיכום אבטחת מידע

| | |
|----------|--|
| 4 | דרישות מערכת אבטחה - כלים ומטרות |
| 5 | סיכון ותקיפה – תקיפה סבילה ופעילה |
| 5 | криptoוגרפיה – טרמינולוגיה |
| 6 | הצפנה סימטרית |
| 7 | הצפנה אסימטרית |
| 8 | криptoוגרפיה וкриptoאנליזה |
| 8 | חיפוש בכוח גס (brute force) |
| 9 | התקפות קריptoאנליזות |
| 9 | ס"ווג שיטות הצפנה – unconditionally secure, computationally secure |
| 10 | צופן ק'יסר |
| 11 | צופן מונואלבטטי |
| 12 | צופן פלייפר |
| 13 | צופן ויג'ניר |
| 16 | צופן פורטו |
| 17 | צופן autokey |
| 18 | צופן ערבות עמודות |
| 19 | הצפנות כפולות |
| 19 | אניגמה |
| 19 | פרדוקס – הגדירה |
| 20 | אלגברה מודולרית |
| 22 | שקליות מודולו n |
| 22 | מספר הופci באלגברה מודולרית |
| 23 | מחלקים – Divisors |
| 23 | אלגוריתם אוקלידי: מציאת GCD |
| 23 | אלגוריתם אוקלידי המורחב |
| 25 | צופן هل |
| 26 | צפנים מודרניים |
| 26 | מדידות Diffusion |
| 28 | מדידות Confusion |
| 29 | צופן פיסטול |
| 30 | צופן DES – Data Encryption Standard |
| 32 | הצפנה כפולה - 2DES |
| 32 | הצפנה מושלשת - 3DES |
| 33 | Advanced Encryption Standard – AES |
| 36 | הפעלת צופן בлок על מסר ארוך |
| 36 | ECB |
| 36 | Cipher Block Chaining - CBC |
| 37 | Beast Attack against SSL/TLS |
| 39 | שיטת Counter |
| 39 | אלגוריתם RC4 |
| 40 | מספרים ראשוניים |
| 40 | משפט פרמה הקטן |
| 40 | חישוב חזקות מהיר |
| 41 | בדיקות ראשוניות |
| 41 | מבחן מילר-רבין |
| 42 | פונקציית אוילר (Φ) |
| 43 | שורשים פרימיטיביים |
| 44 | הצפנה סימטרית (מפורט) |
| 44 | הצפנה אסימטרית (מפורט) |
| 46 | RSA |

| | |
|----------|---|
| 48 | הפעת מפתחות |
| 48 | הפעת מפתח סימטרי בעזרת KDC |
| 50 | הפעת מפתחות ושימוש במפתחות ציבוריים |
| 52 | הפעת מפתח סימטרי בעזרת מפתח ציבורי ידוע |
| 52 | שיטת Diffie Helman Merkle |
| 53 | חתימה דיגיטלית |
| 54 | Hash Function – פונקציית גיבוב |
| 54 | עקרון שובר הIONS |
| 54 | תמצית ההודעה – message digest |
| 54 | פונקציית גיבוב קריפטוגרפית |
| 55 | פונקציית גיבוב לא קריפטוגרפית – TCP Checksum |
| 55 | SHAttered |
| 55 | SHA-512 |
| 56 | תקיפת פונקציות גיבוב – פרדוקס ים ההולדת |
| 57 | קוד אimoto מספרים – (MAC) Message Authentication Code |
| 58 | (Hash MAC) HMAC |
| 59 | Tables – Rainbow Tables |
| 60 | התנגשויות |
| 61 | אימות משתמשים (גיבוב סיסמות, salt) |
| 61 | התקפה מילונית |
| 62 | מתקפת קשת – Rainbow Attack |
| 63 | אכיפת בחרת סיסמות חזקות, בדיקת סיסמות פרואקטיבית |
| 63 | אנטropaיה של סיסמה |
| 64 | פרוטוקול מבוסס אתגר לסימת המשתמש – Challenge Response |
| 64 | הוכחת אפס מידע |
| 64 | (Steve and Carol) Secure Remote Password Protocol (SRP) |
| 68 | בטחת דואר אלקטרוני, שכבת הישום (אליס וbob) |
| 69 | MIME - SMIME |
| 70 | Domain Identification Mail |
| 70 | בטחת מסרים מיידית: WhatsApp |
| 72 | Web Security, Transportation Layer -SSL |
| 74 | IPsec – בטחה בשכבה הרשות |
| 74 | VPN |
| 75 | Authentication Header Protocol (AH) |
| 76 | Encapsulating Security Payload (ESP) |
| 77 | IPsec Security Association (SA) |
| 77 | SPD/SAD |
| 78 | חומות אש – Firewalls |
| 78 | Stateless Packet Filtering |
| 80 | Stateful Packet Filtering |
| 80 | Application Gateways |
| 81 | Bastion Hosts & DMZ |
| 82 | מערכות לזיהוי תקיפה - IDS |
| 82 | Honeypots |
| 83 | תקיפות רשות |
| 83 | Denial Of Service Attack – DOS |
| 83 | Flooding Attack – מתקפות הצפה |
| 84 | Source Address Spoofing – זיווף כתובת השולח |
| 84 | Distributed Denial of Service DDOS Attacks |
| 85 | SYN Spoofing |
| 86 | SYN Cookies |
| 87 | מתקפות על HTTP Flood & Slowloris - HTTP |

| | |
|----------|--|
| 87 | מתקפות Reflection |
| 88 | מתקפות DNS Amplification Attacks – Amplification |
| 88 | חולשות ב-IP Fragmentation |
| 88 | Tiny Fragment Attack |
| 89 | IPv6 Fragmentation |
| 89 | חbillות רעל - Ping of Death |
| 90 | DNS Cache Poisoning |
| 91 | איך מונעים מתקפות? סיכמן |
| 92 | אבטחת מידע במקום עבודה |

דרישות מערכת אבטחה

הגנה על 3 עקרונות:

Confidentiality 1. חישוב

- * סודיות המידע

* פרטיות

א. אמינות Integrity

- * אמינות המידע

*אמיות המערכת

3. זמינות Availability

מטרות

| הערות | אבטחת רשות | אבטחת המערכת |
|-----------------|-------------------------------------|-------------------|
| Authentication | אימות זהות השולח | אימות זהות המשתמש |
| Confidentiality | מניעת גניבת מידע | |
| | סיכון ניתוח תנוצה עיור | |
| Access Control | בקורת גישה | |
| Data Integrity | מניעת השתלה / מחיקה / שינוי של מידע | |
| Availability | מניעת חבלה בתפקוד | |
| | מניעת העמסת הרשות | מניכת העמסת המעבד |
| Non-repudiation | מניעת הכחשה | |

כליים

| הערות | אבטחת רשות | אבטחת המערכת |
|------------|--------------------------|------------------------|
| Encryption | הצפנה | הגנות במערכת הפעלה |
| | פרוטוקולי תקשורת מאובטחת | סיסמאות כניסה |
| | נהלי הפעלת הצפנה וכתיבת | אנטי וירוס וכלים דומים |
| | שיטות הפצת מפתחות | ניפוי תנועה |
| | | הנדסת תוכנה טובה |

סיכון ותקיפה

- **תקיפה (Attack)** היא פעולה של ירי בפגיעה באחד מהיבטי האבטחה של מערכת מידע.
 - **סיכון (Threat)** הוא כל דבר היוצר פוטנציאלי תקיפה – למשל באג בתוכנה/פרוטוקול תקשורת
 - ישנו 2 סוגים תקיפה:
 - * **תקיפה סבילה(Passive)** – התוקף יכול רק לקרוא מהערוץ
 - * ציותות, ניטור תקשורת
 - * קשה לזהות, דgas על מניעה
 - * **תקיפה פעילה (Active)** – התוקף יכול לכתוב/למחוק בערוץ
 - * התחזות, שינוי תכנים, denial of service
 - * דgas על גילוי והטאוששות מהתקיפה
 - * מטרת מערכת אבטחת מידע
 - * למנוע תקיפות, למנוע סיכונים
 - * **LAGOT TAKIFOT**, לטעוד אותן ולפתח כל הגנה נגדן

סוגי תקיפות סבילות:

1. במהלך שליחת הודעה באינטרנט, התוקף קורא פרטי הודעה הנשלחת.
 2. במהלך שליחת הודעה באינטרנט, התוקף רואה את התבנית של הודעה הנשלחת.

סוגי תקיפות פעילות:

1. במקום שההודעה תישלח מהשלוח לנשלח, ההודעה נשלחת ישירות מהתוקף לנשלח.
 2. התוקף תופס את ההודעה בין השולחים, ומאוחר יותר שלוח הודעה לנשלח בשם השלוח.
 3. התוקף משנה את ההודעה הנשלחת, כך שהנשלחה מקבל הודעה שונה מממה שהשלוח כתב.
 4. התוקף משבש את השירות שניתן ע"י השירות.

קRYPTוגרפיה

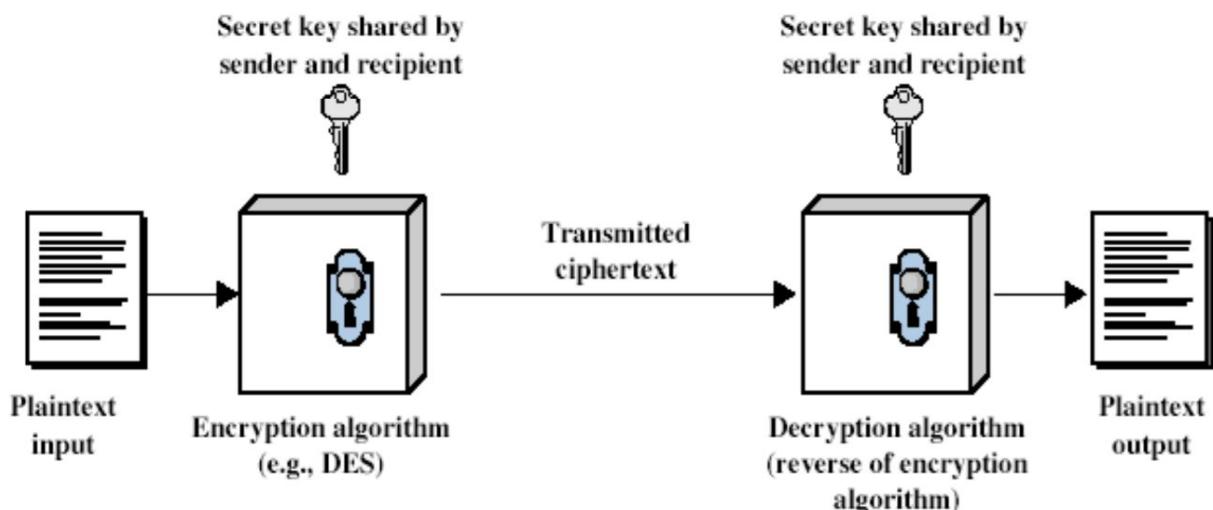
טרמינולוגיות

- | | |
|--------------------------------|---|
| P | טקסט גלי - Plaintext |
| C | טקסט מוצפן – Ciphertext |
| E() | אלגוריתם הצפנה – Cipher |
| $C = E_k(P)$ | הצפנה – Encryption : הפיכת טקסט גלי למוצפן |
| $P = D_k^*(C)$ או $P = D_k(C)$ | פונCTION – Decryption : הפיכת טקסט מוצפן לגלי |

הצפנה סימטרית

| חווןנות | יתרונות |
|--------------------------------------|---|
| יש צורך להעביר מפתחות בין שני הצדדים | מפתח יחיד להצפנה ופענוח |
| יש צורך לאחסן מפתחות | שני הצדדים נדרשים להחזיק באותו מפתח הצפנה |
| אין אפשרות לוודא מי משתמש ב מפתח | שיטת ייעלה ומהירה עם דרישות כוח מחשב נמוכות |

מודל הצפנה סימטרית



דרישות להצפנה סימטרית

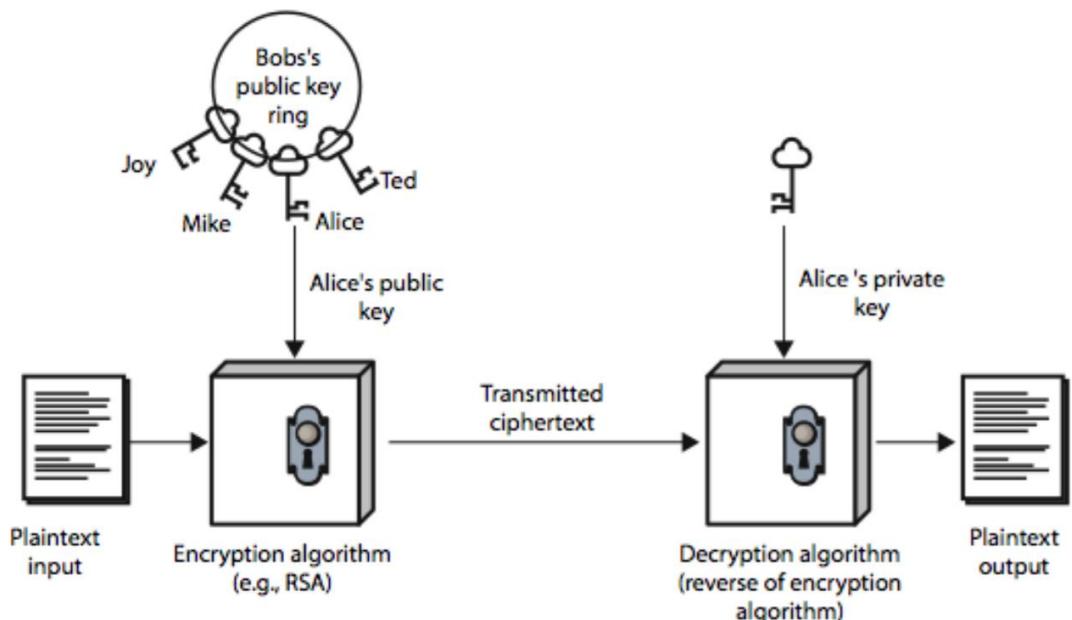
הצפנה ופענוח דרך אותו מפתח:
 $C = E_k(P)$
 $P = D_k(C)$

- אלגוריתם הצפנה חזק
- מפתח סודי שידוע לשני הצדדים
- האלגוריתם ידוע
- דרך בטוחה להעברת המפתחות

הצפנה אסימטרית

- לכל צד יש שני מפתחות: מפתח גלוי/פומבי/ציבורי ומספר מפתח פרטי.
- המפתח הציבורי ידוע, כל אחד יכול להשתמש בו כדי להצפין טקסט עבור בעל המפתח.
- פונCTION אפשרי רק עם המפתח הפרטי שידוע רק לבעל המפתח.
- חישוב האלגוריתם הרבה יותר מסובך מאשר שיטות סימטריות.

מודל הצפנה סימטרית



דרישות להצפנה אסימטרית

הצפנה ופענוח:
 $C = E_{K-PUBLIC}(P)$
 $P = D_{K-PRIVATE}(C)$

- אין אפשרות (פרקטיות) לחשב את המפתח הפרטי מהמפתח הפומבי.
- המפתח הפרטי ידוע רק לבעליו.
- אפשר להניח שהאלגוריתם ידוע.
- דרך לוודא שהשלוח ידוע את המפתח הציבורי של הנמען.

טרמינולוגיה

- **криптוגרפיה** – Cryptography : הענף המדעי שעוסק בחקר שיטות הצפנה.
- **криptoanalysis** – Cryptanalysis : חקר שיטות לשבירת צפנים.
- **криптולוגיה** – Cryptology : שדה שכולל את שני התחומים.

криптוגרפיה

- סוג פעולות הצפנה:
 - תחליף
 - פרמוטציה
 - שילוב שיטות
- מספר מפתחות:
 - מפתח בודד (שיטות סימטריות)
 - שני מפתחות (שיטות א-סימטריות)
- דרך עיבוד הטקסט המקורי
 - בלוקים (64 ביטים או יותר בכל בלוק)
 - Stream – כל אות בנפרד

криptoanalysis

- מטרות:
 - לקרוא את הטקסט
 - להציג את המפתח כדי לקרוא טקסטים נוספים בעתיד
- גישות:
 - אנליזה חכמה
 - שיטות מתמטיות / סטטיסטיות
 - כוח גס (Brute Force)
 - לנסות את כל המפתחות האפשריים
 - לבדוק כל תוצאה ולראות אם היא מהווה טקסט הגיוני

חיפוש בכוח גס (Brute Force)

- מנסים את כל המפתחות האפשריים
- הזמן הנדרש פרופורציונילי למספר המפתחות הנבדקים (נניח שקל לזהות טקסט קרייא)

התקפות קרייפטואנליטיות

ניתנות לשילוג לפי המידע שבידי התקוף

- **טקסט מוצפן בלבד**

- אלגוריתם ידוע

- המצב הכי קשה לקריפטואנליסט

- **טקסט גלוי** (או קל לניחוש) Known Plaintext

- אלגוריתם ידוע

- מшиיגים את הטקסט המוצפן המתאים

- **טקסט מוצפן נבחר** Chosen Cyphertext

- אלגוריתם ידוע

- מшиיגים את הטקסט הגלוי המתאים

- פחות משתמשים בשיטה הנ"ל

סילוג של שיטות הצפנה

- **Unconditionally Secure**

- שיטת הצפנה נקראת "בטוחה ללא סיג" אם אין שום אפשרות לפרוץ אותה,

גם בהינתן משאבים בלתי מוגבלים

- אפשרי בתיאוריה אך לא ישים

- **Computationally Secure**

- שיטת הצפנה נקראת "בטוחה מבחינה חישובית" אם כמות הזמן הנדרשת כדי לפענה

טקסט ארוכה יותר מאשר החיים של המידע המוצפן

- לווקח יותר מדי זמן לפענה את ההודעה, עד שהיא הופכת לחסרת ערך

שיטות הצפנה קלאסיות

צפוי החלפה קלאסיים - טבלאות

- מחליפים את אותיות הטקסט המקורי באותיות אחרות / מספרים / סימנים אחרים לפי חוקיות.
- אם مستقلים על הטקסט כסדרה של סיביות, אך צפוי החלפה מחליף רצפים של ביטים ברצפים של ביטים אחרים.

צפן קיסר

- צפן שבו כל אות מזוזת K מקומות נקרא צפן קיסר.
- הזרת האותיות היא מעגלית ($C \rightarrow B, Z \rightarrow Y, A \rightarrow X$)

איך פועל צפן קיסר?

• הצפנה:

- נמחק את כל הרוחחים (פרקטיקה מקובלת בהצפנה קלאסיות)
- המפתח הוא מספר הדילוגים K

• פענוח:

- נחליף כל אות בתחליף שלה לפי צפן קיסר

**meetmeafterthetogaparty
PHHWPHDIWHUWKHWRJDSDUWB**

- מקובל לסמן את הצפן ב UPPER CASE
- הטקסט המקורי יהיה lower case

חולשות צפן קיסר

- כולם יודעים שתהlixir ההצפנה יכול לפענה כל הودעה
- המפתח קטן מדי – לפי מספר האותיות בא"ב בשפה המוצפנת

יתרונות צפן קיסר

- אפשרות בשימוש במפתח רנדומלי, מספר האפשרויות: $!26^{26}$ לכל 26 האותיות בא"ב
- שבירת הצפן בלתי אפשרית ע"י שימוש בכוח גס.

צופן מונואלפבטי

במקרה הצעה פשוטה של הא"ב, נגיד ר החלפה כללית של אותיות הא"ב. כמובן, לכל אחת מה-26 אותיות נבחר אות שתחליף אותה. לפיכך, המפתח יהיה באורך 26 אותיות.

הצפנה

- שני הצדדים יסכימו על מילת מפתח (או משפט) וממנה ייצור פרמטרים של אותיות.
- מוחקם את כל המופיעים החזריים של אותיות במילת הקוד.
- מוסיף את כל האותיות החסרות לפי סדר הא"ב.

מפתח

לדוגמה, השתמש בטבלה (במפתח) הבאה:

| | |
|----------------|----------------------------|
| Plain: | abcdefghijklmnopqrstuvwxyz |
| Cipher: | DKVQFIBJWPESCXHTMYAUOLRGZN |

חזק של צופן מונואלפבטי

- מספר המפתחות השונים הוא $4 \times 10^{26} \sim !$
- שבירת הצופן בכוח גס אינה אפשרית

שימוש בкриptoאנליזה

אבחן: צופן מונואלפבטי אינו משפיע על שכיחות האותיות
על מנת לשבור צופן מוצפן:
 ○ נחשב שכיחות לכל אותיות
 ○ נשווה את השכיחויות זו לזו
 ○ נוכל לנחש חלק מהאותיות

שבירת צופן מונואלפבטי

- נספר מופיעים של אותיות ונחשב שכיחיות
- האותיות P ו-Z נפוצות, אך ננחש שהן e ו-t מכאן ש-ZWP מסמל the, געשה ניסוי וטעה.

יתירות בשפה

- שפות טבעיות מכילות מידע מיותר (יתירות)
- ניתן לשזר משפטים עם אותיות חסרות בקלות
- באותיות שונות נעשה שימוש בשכיחות שונה, האות הנפוצה ביותר היא: E.
- האותיות X, K, Q, J, Z, נדירות.
- קיימות טבלאות מוכנות לשכיחות של אותיות בודדות, זוגות אותיות, וכו', בשפות שונות.

צופן פלייפר

- המפתח בצופן פלייפר הוא מטריצה בגודל 5X5 המומלאת באותיות שונות. (בגלל שיש 26 אותיות, האות ז' תיצג גם את האות ל').

► **לדוגמה, מילת המפתח היא Y**

| | | | | |
|---|---|---|-----|---|
| M | O | N | A | R |
| C | H | Y | B | D |
| E | F | G | I/J | K |
| L | P | Q | S | T |
| U | V | W | X | Z |

Key

- ניתן לציר מטריצה בעזרת מילת מפתח:
- נקתוב את מילת המפתח במטריצה (ללא אותיות חוזרות)
- נמלא את המטריצה בשאר האותיות

הצפנה

- текסט מוצפן **שתי אותיות בכל פעם**
- שלב מקדים להצפנה, נחלק את הטקסט המקורי לזוגות אותיות
- אם מתקיים זוג אותיות זהות, נפריד בין ע"י האות X
- אם בסוף הטקסט נותרת אות בודדה, נוסיף X אחרת

an apple a day keeps the doctor away

an ap pl ea da yk ex ep st he do ct or aw ay

- אם זוג אותיות מופיעות באותה שורה בטבלה, כל אות תוחלף באאות **شمימין** לה (באופן מעגלי)
- אם זוג אותיות מופיעות באותנו טוֹר בטבלה, כל אות תוחלף באאות **שמתחת** לה (באופן מעגלי)
- בשאר המקרים**, כל אות מוחלפת באאות שנמצאת בשורה שלה בטבלה ובטור של האות השנייה

פענוח

- אם זוג אותיות מופיעות באותה שורה בטבלה, כל אות תוחלף באאות **משמאַל** לה(באופן מעגלי)
- אם זוג אותיות מופיעות באותנו טוֹר בטבלה, כל אות תוחלף באאות **שלמעלה** לה (באופן מעגלי)
- בשאר המקרים**, כל אות מוחלפת באאות שנמצאת בשורה שלה בטבלה ובטור של האות השנייה

חזק צופן פלייפר

- יותר מאובטח מצופן מונואלפבטי'
- עושה שימוש ב $600 = 25 \times 24$ זוגות, לעומת 26 אותיות בצופן מונואלפבטי'
- עדין ניתן לשבירה, בהינתן כמה מאות של אותיות, שומר על רוב התכונות של הטקסט המקורי

צופן וויג'ינר

צופן אלברטִי – דוגמא לצופן מסוג וויג'ינר

המכשור בניו מ-2 דיסקים מעגליים, שמחוברים אחד לשני, כך שאחד יכול להסתובב ביחד.

הdisk הגדל יותר היה קבוע, והdisk הקטן יותר יכול לנוע בתנועה מעגלית.

כל disk מחולק ל-24 תאים שוים.

הטבעת החיצונית מכילה אותיות ב UPPER CASEüber ה-*plaintext* ו-4 ספרדים,

והטבעת הפנימית מכילה אותיות מעורבות ב lower caseüber ה-*ciphertext*.



By default, every 4 characters (4 = period), the disk is rotated clockwise of 1 letter (1 = periodic increment), this changes the substituting alphabet.

צופן וויג'ינר

- משתמש בצפני קיסר
- המפתח הוא רצף L של אותיות $K_1, K_2 \dots K_n$
- אותן L אותיות מוצפנת בצופן קיסרי על מפתח K_i
- אחרי L אותיות חוזרים להשתמש במפתח מההתחלת
- הפענוח עושים שימוש באותו מפתח, ע"פ כללי הפענוח של צופן קיסר

הצפנה

- ▶ **Plaintext :** "ATTACKATDAWN"
- ▶ **Const Key:** LEMONLEMONLE
- ▶ **Ciphertext :** L XFOPV EF RN HR

- רשום את הטקסט באופן גלי.
- רשום מעליון, באופן מהזורי, את המפתח.
- כל אות מפתח משמש לצופן קיסר אחר.
- בעזרתו מוצפנים את האות בגלוייה
- שמתחתייה.

פונCTION – לשימר לב לצבעים בטבלה ובtekst בטלה הצלנה

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | |
| C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | | |
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | | | |
| E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | | | | |
| F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | | | | | |
| G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Z | | | | | | | |
| H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Z | | | | | | | | |
| I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Z | | | | | | | | | |
| J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Z | | | | | | | | | | |
| K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Z | | | | | | | | | | | |
| L | M | N | O | P | Q | R | S | T | U | V | W | X | Z | | | | | | | | | | | | |
| M | N | O | P | Q | R | S | T | U | V | W | X | Z | | | | | | | | | | | | | |
| N | O | P | Q | R | S | T | U | V | W | X | Z | | | | | | | | | | | | | | |
| O | P | Q | R | S | T | U | V | W | X | Z | | | | | | | | | | | | | | | |
| P | Q | R | S | T | U | V | W | X | Z | | | | | | | | | | | | | | | | |
| Q | R | S | T | U | V | W | X | Z | | | | | | | | | | | | | | | | | |
| R | S | T | U | V | W | X | Z | | | | | | | | | | | | | | | | | | |
| S | T | U | V | W | X | Z | | | | | | | | | | | | | | | | | | | |
| T | U | V | W | X | Z | | | | | | | | | | | | | | | | | | | | |
| U | V | W | X | Z | | | | | | | | | | | | | | | | | | | | | |
| V | W | X | Z | | | | | | | | | | | | | | | | | | | | | | |
| W | X | Z | | | | | | | | | | | | | | | | | | | | | | | |
| X | Z | | | | | | | | | | | | | | | | | | | | | | | | |
| Z | | | | | | | | | | | | | | | | | | | | | | | | | |

נבדוק בטבלה ויגנאר את ההצלבות של האותיות, נבעץ זאת بصورة הבא:

קודם כל, לזוג הראשון ניקח את האות הראשונה במשפט, נמצא את מיקומה בעמודה הראשונה של טבלת ויגנאר (מחוץ לטבלה עצמה).

באותה שורה של האות הראשונה במלת המשפט, נhapus את האותgalיה מהוצאה.

נhapus את האות הראשונה במלת המוצהנת.

לאחר הצלבה בין שתי האותיות, נראה איזו אות נקלט בשורה הראשונה (מחוץ לטבלה).

וזהו יהיה אותה האות הראשונה של המילה הטקסט שנרצה לגלות.

חזק צופן וויג'ינר

- כל אות בטקסט הגלוי מוצפנת במספר אופנים שונים
- שכיחות אותיות הצופן היא ערובה של מספר שכיחויות שונות

אסטרטגיות לתקיפה הצופן

- נמצא את אורך המפתח L
- חזרות בטקסט המוצפן רמזחות על אורך המפתח L
- חזרה בטקסט האלי ביחס לאותיות מפתח תיצור חזרה בטקסט המוצפן
- ברגע שאורך המפתח ידוע, מוכן לקבץ יחד אותיות צופן שהוצפנו בעזרת אות מפתח,
- נחשב שכיחויות אותיות אלו, ולפי הטבלה מוכן לנחש את אות המפתח.

| מפתח | | טבלת ויזינר | | | | | | | | | | | | | | | | | | | | | | | | | |
|------|---|-------------|---|---|---|---|---|---|---|---|---|---|---|---|------------|---|---|---|---|---|---|---|---|---|---|---|--|
| | | טקסט גלי | | | | | | | | | | | | | טקסט מוצפן | | | | | | | | | | | | |
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | | |
| A | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | |
| B | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | |
| C | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | |
| D | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | |
| E | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | |
| F | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | |
| G | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | |
| H | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | |
| I | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | |
| J | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | |
| K | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | |
| L | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | |
| M | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | |
| N | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | |
| O | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | |
| P | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | |
| Q | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | |
| R | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | |
| S | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | |
| T | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | |
| U | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | |
| V | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | |
| W | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | |
| X | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | |
| Y | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | |
| Z | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | |

שבירת אלגוריתם ויג'ינר ע"י שיטת קס'יסקי

1. חפשו חזרות באורך 3 לפחות בטקסט המוצפן
2. חשבו את המרחקים בין חזרות סמוכות
3. מצאו פירוק לגורמים ראשוניים של כל חזרה
4. מצאו אילו גורמים ראשוניים מופיעים יחד ברוב המקרים של המרחקים – נסמן B את מכפלת הגורמים הללו
5. נחשש ש: $L=N$, וננסה למצוא את מפתח ההצפנה ע"פ:
 - ברגע שאורך המפתח ידוע, מוכן לקבץ יחד אותיות צופן שהוצפנו בעזרת אות מפתח.
 - מוכן לנחש את אותיות מפתח.

לדוגמה:

נתון כתוב סתר שנוצר מצוף ויז'נر. בחיפוש אחרי רצף אותיות חוזרת, נתגלה:

רצף האותיות KMP הופיע 3 פעמים, כשהוא מתחילה במקומות 150, 528, 430

רצף האותיות EJ הופיע 4 פעמים, כשהוא מתחילה במקומות 54, 110, 236, 367

רצף האותיות QSSD הופיע פעמים, כשהוא מתחילה במקומות 301, 945.

**הערה: גם הרצפים QSS ו-SSD הופיעו פעמים (למה? איפה?)
עדיף שלא להתבסס עליהם (מדוע?)**

לפי שיטת Kasiski, מהו אורך המפתח הסביר ביותר?

$430-150=280$ divisors: (2, 2, 2, 5, 7) ▶

$528-430=98$ divisors: (2, 7, 7) ▶

$110-54=56$ divisors: (2, 2, 2, 7) ▶

$236-110=126$ divisors: (2, 3, 3, 7) ▶

$367-236=131$ divisors: (131) ▶

$945-301=644$ divisors: (2, 2, 7, 23) ▶

נתיחס למרחק 131 – כיווץ דופן (מיידי).

כל חמישה המרחקים האחרים כוללים 2 ו- 7 כמחוקקים ראשוניים => נניח $L=14$
מחוקקים אחרים (3, 5, 23) הם נדירים

2 מופיע פעמיים לפחות (בשלווה בין חמישה המרחקים, ללא המרחק 131), האם לקחת אותו פעמיים
ולניח $L=28$?

תשובה: לא נימוקים: א. 3 מתוך 5 אינם רוב מספיק ברור
ב. המרחקים 98 ו- 126 מתחלקים ב 14, אך לא ב 28

יתכן כי כל הגורמים 2 הינם יוצאי דופן מקרים, אז אין להזניח את $L=7$ אפשרויות לפתרון (אך $L=14$ יותר ריאלי)

Porto Code

Introduction

The Porta Cipher is a polyalphabetic substitution cipher invented by Giovanni Battista della Porta. Where the Vigenere cipher is a polyalphabetic cipher with 26 alphabets, the Porta is basically the same except it only uses 13 alphabets. The 13 cipher alphabets it uses are reciprocal, so enciphering is the same as deciphering.

The algorithm used here is the same as that used by the American Cryptogram Association. Another source is Helen Fouche Gaines book "Cryptanalysis".

The Algorithm

The 'key' for a porta cipher is a key word. e.g. 'FORTIFICATION'

The Porta Cipher uses the following tableau to encipher the plaintext:

| Keys | a b c d e f g h i j k l m n o p q r s t u v w x y z |
|------|---|
| A,B | n o p q r s t u v w x y z a b c d e f g h i j k l m |
| C,D | o p q r s t u v w x y z n m a b c d e f g h i j k l |
| E,F | p q r s t u v w x y z n o l m a b c d e f g h i j k |
| G,H | q r s t u v w x y z n o p k l m a b c d e f g h i j |
| I,J | r s t u v w x y z n o p q j k l m a b c d e f g h i |
| K,L | s t u v w x y z n o p q r i j k l m a b c d e f g h |
| M,N | t u v w x y z n o p q r s h i j k l m a b c d e f g |
| O,P | u v w x y z n o p q r s t g h i j k l m a b c d e f |
| Q,R | v w x y z n o p q r s t u f g h i j k l m a b c d e |
| S,T | w x y z n o p q r s t u v e f g h i j k l m a b c d |
| U,V | x y z n o p q r s t u v w d e f g h i j k l m a b c |
| W,X | y z n o p q r s t u v w x c d e f g h i j k l m a b |
| Y,Z | z n o p q r s t u v w x y b c d e f g h i j k l m a |

To encipher a message, repeat the keyword above the plaintext:

```
FORTIFICATIONFORTIFICATIONFO
DEFENDTHEEASTWALLOFTHECASTLE
```

Now we take the first key letter 'F', and find it on the first column (the key column containing two letters) on the tableau. Then, we move along the 'F' row of the tableau until we come to the column with the 'D' at the top (The 'D' is the first plaintext letter), the intersection is our ciphertext character, 'S'. The same process is repeated for all the characters.

So, the ciphertext for the above plaintext is:

```
FORTIFICATIONFORTIFICATIONFO
DEFENDTHEEASTWALLOFTHECASTLE
synnjscvrnnrlahutukucvryrlany
```

You may notice that it is possible for two different keywords to produce exactly the same enciphered message. The encryption and decryption process for this cipher is identical, so encrypting a piece of text twice with the same key will return the original text.

צופן Autokey

- מפתח הצפנה צריך להיות באורך הטקסט המוצפן.
- המפתח – מילת המפתח ולאחריה הטקסט המקורי:
- L האותיות הראשונות מוצפנות בעזרת מילת המפתח.
- האותיות הבאות מוצפנות בעזרת הטקסט שפענונו, מההתחלה לסוף עד לגודל המילה המוצפנת שנרצה למצוא.
- טעות בהצפנה באות אחת, תפגע בהמשך הפענוח, החל מאותה האות שנפגעה.
- צופן יותר חזק, היות שהמפתח כבר אינו מחזורי.

הצפנה: אותה דרך הצפנה, כמו קוד ויג'ניר

דוגמא:

א. מילת קוד: SECURITY
מספר: UEPN KWNAJ TUBL
סוג הצפנה: ויג'נאר, AutoKey,
א.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | E | C | U | R | I | T | Y | C | A | N | T | T |
| U | E | P | N | K | W | N | A | J | T | U | B | L |
| C | A | N | T | T | O | U | C | H | T | H | I | S |

המילה שקיבלת: Can't touch this

פענוח:

הדרך שקיבلت את המילה:

על מנת לפצח קוד ויג'נאר Autokey, علينا להגדיר קודם כל מפתח.
המפתח יהיה המפתח הנוכחי לנו בנוסף ל-L האותיות הראשונות שモוצפנות בעזרת מילת המפתח,
כך שבתרגיל זה מילת המפתח תהיה TTSECURITYCANTT. יש לציין כי גודל מילת המפתח תהיה שווה לגודל המילה
המווצפנת.
לאחר שהגדכנו את מילת המפתח באופן ברור, علينا לקחת כל אות במילת המפתח ולהשוו אותה לכל אות (בהתאם למיקום
שלה במילה) בקוד שנרצה לפענוח.

1. כך שבוגמא שלנו, זוג האותיות הראשונות שנרצה לשים ביחד יהיו S ו-U.
אחר מכן אנחנו נבדוק בטבלת ויג'נאר את ההסתובות של האותיות, נבצע זאת בצורה הבא:
קודם כל, לזוג הראשון ניקח את האות הראשונה במילת המפתח שיצרנו קודם, שהוא S.
נמצא את מקומה בעמודה הראשונה של טבלת ויג'נאר (מחוץ לטבלה עצמה).
באותה שורה של האות הראשונה במילת המפתח, נחפש את האות הראשונה במילה המוצפנת - שהוא U.
לאחר הצלבה בין שתי האותיות, נראה איזו אות נקלט בשורה הראשונה (מחוץ לטבלה),
וזהו יהיה הזוג האותים הראשוניים של המילה הטקסט שנרצה לגלות - שהוא C.

וכך הלאה לשאר אותיות המילה המוצפנת.

שימוש במפתח חד פומי – לחיזוק צופן Autokey

- אם נשמש במפתח אקריא לחלוטין, שאורכו באורך הטקסט, ההצפנה תהיה בטוחה ללא סיג, נקראת מפתח חד פומי.
- שימוש במפתח הנ"ל פעם אחת בלבד. ← בלתי שביר, כי אין קשר סטטיסטי בין הטקסטים.

הצפנה עם מפתח שבוני מרצף אקריא של ביתים

- פונCTION של המסר ע"י שימוש באוטו XOR אופרטור ששומש במהלך ההפניה של המסר.

| Msg | Key | = Msg XOR Key | Msg XOR Key | Key | (Msg XOR Key) XOR Key |
|-------|-------|---------------|-------------|-------|-----------------------|
| True | True | False | False | True | True |
| True | False | True | True | False | True |
| False | True | True | True | True | False |
| False | False | False | False | False | False |

צפוני שינוי סדר

- שיטות הצפנה המבוססות על שינוי סדר / פרמוטציה
- המסר מוצפן ע"י שינוי סדר האותיות, מבלי לשנות את האותיות עצמן
- צפניהם מסוג זה קלים לדיחוי

צופן ערבות עמודות

- המפתח הוא פרמוטציה של L, 1, 2, 3, ניתן ליצור פרמוטציה ממילת מפתח באורך זהה.
- 1. רשום את מילת המפתח בשורה
- 2. מספר את האותיות לפי סדר אלפבטי (האות הראשונה בא"ב קיבל את המספר 1 וכו'..)
- 3. רשום את המספרים מתחת לאותיות

דוגמיה:

cat nine ▶

213 3241 ▶

4. רשום את הטקסט האgliי מתחת למספרים

5. העתק את העמודות במלבן לשורה ארוכה, לפי סדר המספרים שמעל העמודות

Word key: M U S T A R D

Key:

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 7 | 5 | 6 | 1 | 4 | 2 |
|---|---|---|---|---|---|---|

Plaintext:

| | | | | | | |
|---|---|---|---|---|---|---|
| a | t | t | a | c | k | p |
| o | s | t | p | o | n | e |
| d | u | n | t | i | l | t |
| w | o | a | m | | | |

Ciphertext:

COIPE TAODW KNLTT NAAPT MTSUO

הצפנות כפולות

- הצפנה כפולה תיתן צופן חזק יותר
- שני תחליפים יוצרם צופן תחליפי חדש, ושינוי סדר יוצר צופן שניוי סדר חדש
- לא תמיד הצופן החדש יהיה חזק יותר, אבל התחליף שאחריו יוצר צופן חזק עוד יותר.

אניגמה

- מבוסס על גליל המורכב מגלגלים, שכיל אחד מהם מייצג תחליף מונואלפטי.
- לשולשה גלגלים יש³ 26 מצבים הצפונה.
- אם ישנן 2 מכונות עם 2 סדרות זהות של 5 דיסקים בכל אחת,
- casar בכל דיסק 26 אותיות המסודרות בסדר אקריאי שונה.
- לכל מכונה יש 3 חריצים כאשר יש להכניס אליהם 3 דיסקים.

$$\frac{5!}{3! \cdot 2!} \cdot 3! \times 26^3 = 1054560 \quad \text{כך שמספר הרשומות מבוסד הנתונים הנ"ל הינו:}$$

▶ [Hans-Georg Schnell's role in French intelligence](#)

▶ Obtained information on its usage

- ▶ daily code book indicated rotors and orientation
- ▶ a different orientation key for each message

▶ Rejewski focused on the repetitions

- ▶ Message (3 letters) key encrypted twice in the message header
 - ▶ Formalized relationships between 1st-4th, 2nd-5th, and 3rd-6th letters
 - ▶ ABCDEFG H I J K LMNOPQRS TUVWXYZ - 1st letter
 - ▶ FQHPLWOGBMVRXUYCZ I TNJE A SDK - 4th letter
- ▶ Built chains
 - ▶ (AFW), (BQZKVELRI), (CHGOYDP), (JMXSTNU)
- ▶ Chains depend only on scrambler orientation, not on pairs swaps
 - ▶ Thus need to consider only $3! \times 26^3 = 105456$ configurations

| Msg | 6 | 5 | 4 | 3 | 2 | 1 |
|-----|---|---|---|---|---|---|
| 1 | M | G | R | K | O | L |
| 2 | E | Z | X | T | V | M |
| 3 | E | P | M | T | K | J |
| 4 | X | Z | P | Y | V | D |

פענוח

חרונות אניגמה

- i. The fact that any letter of the plaintext can not remain an identical letter in the corresponding position at the ciphertext.
- ii. A repeated identical message format in different messages (e.g., the date was written at the head of the message and salute to Hitler at the end of the message) enabled a known plaintext attack.
- iii. The fact that the session key was transmitted twice in a row at the beginning of the message. The key length was 3 letters and thus the 1st letter of the message was the same as the 4th letter, the 2nd same as the 5th and the 3rd same as the 6th. The Polish Marian Rejewski built letter chains of the above-mentioned letter correspondences and characterized any of the Enigma configurations by the lengths of these chains. When the Polish absorbed Enigma ciphertexts, letter chains were built and examined and the Enigma configuration (and master-key) was deduced according to the chain lengths.

פרדוקס – עקרון או טענה המנוגדים לדעה הרווחת, קביעה או דגש הנראים כמנוגדים לשכל היישר או סותרם אותו. דבר מה שהוא אבסורד לכואיה או בהגדלה. אך אפשר שהוא בעצם אמיתי.

אלגברה מודולרית

פעולות מודולו

$n = a \text{ mod } r$ אם ורק אם קיימים מספר q שמקיים:
 $r + n - q = a$ וגם $0 \leq r < n - 1$

במילים אחרות, r הוא השארית בחלוקת של a ב n

- בכלל, $n \text{ mod } (n \pm 1) = a \text{ mod } n$
- (-12) mod 7 = (-5) mod 7 = 2 •
- 23 mod 6 = 17 mod 6 = 11 mod 6 = 5 •

חשבון מודולרי

נגידיר קבוצה/חוג $\{0, 1, 2, \dots, n-1\}$ עם שתי פעולות, שנקראות חיבור וכפל ב- n : Z_n :

$$\begin{aligned} Z_n - a + b &= (a + b) \text{ mod } n & \circ \\ Z_n - b a * b &= (a * b) \text{ mod } n & \circ \end{aligned}$$

לוח החיבור ב- Z_8

| + | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 |
| 3 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 |
| 4 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 |
| 5 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 |
| 6 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 |
| 7 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

לוח הכפל ב- Z_9

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | 0 | 2 | 4 | 6 | 8 | 1 | 3 | 5 | 7 |
| 3 | 0 | 3 | 6 | 0 | 3 | 6 | 0 | 3 | 6 |
| 4 | 0 | 4 | 8 | 3 | 7 | 2 | 6 | 1 | 5 |
| 5 | 0 | 5 | 1 | 6 | 2 | 7 | 3 | 8 | 4 |
| 6 | 0 | 6 | 3 | 0 | 6 | 3 | 0 | 6 | 3 |
| 7 | 0 | 7 | 5 | 3 | 1 | 8 | 6 | 4 | 2 |
| 8 | 0 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

לוח החיבור ב- Z_{26}

| + | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 |
| 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 |
| 4 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 |
| 5 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 |
| 6 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 |
| 7 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 8 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 9 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 10 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 11 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 12 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 13 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 14 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 15 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 16 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 17 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 18 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 19 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 20 | 20 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 21 | 21 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 22 | 22 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 23 | 23 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 24 | 24 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 25 | 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |

- אם בטבלת ויג'ינר נחליף כל אות בתחום 0-25, גם בטקסט המקורי וגם באותיות המפתח, נראה כי טבלת ויג'ינר היא חיבור ב- Z_{26} .
- את תוצאות החיבור נתרגםഴירה לאותיות.
- תהליך הפענוח לפי טבלת ויג'ינר היא חיסור ב- Z_{26} .

טchniques לשוגג צפניהם ידניים שונים:

- הצפנה ופענוח של אות בודדה לפי הנוסחה **בצופן ויג'ינר** - המפתח הוא סדרה $K_0K_1\dots K_{L-1} \dots K_L$:

 - $C = P + K \text{ mod } 26$
 - $P = C - K \text{ mod } 26$

- **צופן קיסר** – המפתח הוא אות בודדה K :
 - $C_i = P_i + K \text{ mod } 26$
 - מפתח כמו בויג'ינר : $i < L \quad C_i = P_i + K_i \text{ mod } 26$
 - $i \geq L \quad C_i = P_i - P_{i-L} \text{ mod } 26$
- **מפתח חד פעמי** – המפתח הוא סדרה הארוכה ... $K_0K_1\dots K_L$:
 - $C_i = P_i + K_i \text{ mod } 26$

שקלות מודולו ח

סימון: $a \equiv b \pmod{n}$

הגדירה: $a \bmod n = b \bmod n$ אם ורק אם $a \equiv b \pmod{n}$

תכונות של שיקילות מודולו ח

אם $a \equiv b \pmod{n}$ אז $(a-b)$ מכפלה של n

$$(a \times b) \bmod n = ((a \bmod n) \times (b \bmod n)) \bmod n$$

אם $a \equiv b \pmod{n}$ ו $c \equiv d \pmod{n}$

עבור כל $a^k \equiv b^k \pmod{n}$

$$a + c \equiv b + d \pmod{n}$$

$$a \times c \equiv b \times d \pmod{n}$$

משמעותם בתוכנות אלו בעיקר למען חישוב מספרים גדולים

$$(17^3 \times 145)^4 + 2103 \text{ mod } 7 = 3^4 \times 5^4 + 2103 \text{ mod } 7 = 81 \times 625 + 2103 \text{ mod } 7 = 50625 + 2103 \text{ mod } 7 = 52728 \text{ mod } 7 = 0$$

$$(A)(B) \geq (17^3 \times 145) \bmod 7 \equiv 6 * 5 \bmod 7 \equiv 2$$

$$\Rightarrow (17^3 \times 145)^4 \bmod 7 \equiv 2^4 \bmod 7 \equiv 2 \quad (\text{C}) : 2103 \bmod 7 \equiv 3 \quad (\text{D})$$

$$(C)(D) \Rightarrow ((17^3 \times 145)^4 + 2103) \bmod 7 = (2+3) \bmod 7 = 5$$

הערה – במידה ונכפין באופן גלי 13

פעמים עם ויג'ינר, נקלט הצפנה פחות בטוחה,

מהירות ולאחר חישוב נקבע שנתיים בדיקת המפתח פעםיים בלבד, ככלומר יהיה קל לפיצוח.

ההופכי (inverse) של a ב- \mathbb{Z} הוא מספר b שמקיים

Z_n ב a · b = 1 ►

סימון: ►

תכונות של Z_n

- חוג שבו לכל מספר שונה מופיע יש הופכי, נקרא שדה. Z הוא שדה אם ורק אם סדרת רצוני. אם סדרת רצוני – למליהים שלן אין הופכי ב- Z .

דוגמאות ►

○ כדי שנוכל לפענח נctrר לוודא
שניתו לחשב הופci.

שודות ► Z₇, Z₃₁

o עברו 26=ח (או כל מספר פריק אחר)

לא תלמיד קיים הופci.

וועזר בפיטוח הצפנות חזקיות

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | 0 | 2 | 4 | 6 | 8 | 1 | 3 | 5 | 7 |
| 3 | 0 | 3 | 6 | 0 | 3 | 6 | 0 | 3 | 6 |
| 4 | 0 | 4 | 8 | 3 | 7 | 2 | 6 | 1 | 5 |
| 5 | 0 | 5 | 1 | 6 | 2 | 7 | 3 | 8 | 4 |
| 6 | 0 | 6 | 3 | 0 | 6 | 3 | 0 | 6 | 3 |
| 7 | 0 | 7 | 5 | 3 | 1 | 8 | 6 | 4 | 2 |
| 8 | 0 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

מחולקים Divisors

- מספר השונה מאפס, מחלק את a אם קיימים m, b (a=mb) ו- c' ש mb=c' (a, m, b מספרים שלמים)
 - טענה שקולה: $a \mid b$ אם ורק אם $b = a \cdot c$
 - נסמן זאת ע"י $a \mid b$, שפירושו:
 - b מחלק a
 - a כפולת של b
- לדוגמא 12, 1, 2, 3, 4, 6, 8, 12 מחולקים של 24

מחולק משותף הגדול ביותר ביחס ל- GCD

- הגדרה: $\gcd(a, b)$ הוא המספר הגדול ביותר שמסוגל לחלק את a ו- b
- a, b נקראים זרים (relatively prime) אם $\gcd(a, b) = 1$
- a נקרא ראשוני אם אין לו מחולקים פרט לעצמו ול-1

שימושי GCD ב- \mathbb{Z}_n

- איבר a הפיך ב- \mathbb{Z}_n אם ורק אם a ו- n זרים

▶ Euclid (a, b)

אלגוריתם האוקלידי : דרך יעליה למציאת gcd

1. $A = a; B = b$
2. If $B = 0$, return A
3. $R = A \bmod B$
4. $A = B$
5. $B = R$
6. Goto 2

אלגוריתם אוקלידי המורחב : מציאת הופכי ב- \mathbb{Z}_n

Goal: find integer coefficients x and y such that:

I.E.:

$$xa + yb = \gcd(a, b)$$

- Find the greatest common divisor of two positive integers a and b
- Write this greatest common divisor as an integer linear combination of a and b

Algorithm:

1. Set $c = \max(a, b)$, and set $d = \min(a, b)$.
2. Find the quotient (q) and the remainder (r) when c is divided by d . Write an expression for r in terms of a and b :

$$r = c - q \cdot d \rightarrow r = \max(a, b) - q \cdot \min(a, b)$$
3. If $r = 0$, then $\gcd(a, b) = d$. The expression for the previous value of r gives an expression for $\gcd(a, b)$ in terms of a and b . Stop.
4. Otherwise, use the current values of d and r as the new values of c and d , respectively, and go back to step 2.

ANOTHER WAY TO CALCULATE THE EXTENDED EUCLID ALGORITHM

- Calculate GCD(a, b)
- 1st line is: -, a, 1, 0
- 2nd line is:-, b, 0, 1
- 3rd line and thereafter:
calculated according to
the 2 lines above it:
 - $q_i = r_{i-2} / r_{i-1}$ (the quotient)
 - $r_i = r_{i-2} \text{ mod } r_{i-1} = r_{i-2} - q_i r_{i-1}$ (the remainder)
 - $j_i = j_{i-2} - q_i j_{i-1}$
 - $k_i = k_{i-2} - q_i k_{i-1}$
- The GCD is produced in the r column (one line before it zeroed)
- If $r_i=1$ than k_i is the inverse of b in Z_a $b^{-1} \equiv k_i \pmod{a}$

Example: GCD(1759, 550)

| q | r | j | k |
|----|------|------|------------|
| - | 1759 | 1 | 0 |
| - | 550 | 0 | 1 |
| 3 | 109 | 1 | -3 |
| 5 | 5 | -5 | 16 |
| 21 | 4 | 106 | -339 |
| 1 | 1 | -111 | 355 |
| 1 | 0 | | |

$$\text{GCD}(1759, 550) = 1$$

$$550^{-1} \equiv 355 \pmod{1759}$$

54

צופן היל

- צופן היל פועל על הא"ב האנגלית המכיל 26 אותיות, כאשר כל אות מיוצגת ע"י שלם מודולו. $b \equiv c \cdot A + 0, b = 1, c = 25$.
- המטריצה המשמשת להצפנה נקראת מפתח פענו.

הצפנה $\mathbf{u} \cdot \mathbf{A} \leftarrow \mathbf{v}$

- תחיליה מחלקים אותו לבלוקים באורך ח' אותיות.
- אז מצפינים את הבלוק ע"י הכפלת השם בוקטור (עמודה) במת' ריצה ההיפיכה מסדר NxN.

נקח את האלפבית העברי המכיל 22 אותיות (לא סופיות), נניח שהאותיות ממוספרות במספר הרגיל כמתואר בטבלה הבאה וכל החישובים הם מודולו 22.

| | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|
| ת | ש | ר | כ | ל | מ | נ | ס | ע | פ | צ | ז | ק | א | ב | ג | ד | ה | ו | ז | ח | ט | י |
| 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |

נניח ש- $\mathbf{u} = u$ ומפתח ההצפנה הוא המחרוזת "הקד הסודי" (תשע אותיות ללא הרווח), המפתח מותפרש משמאלי לימין במטריצה הבאה:

$$\mathbf{A} = \begin{pmatrix} 4 & 18 & 5 \\ 3 & 4 & 14 \\ 5 & 3 & 9 \end{pmatrix}$$

אפשר להמיר את מחרוזת האותיות של המפתח למטריצה בכמה דרכים. נניח שהמספר להצפנה הוא האותיות "אבג", לפי המספר שנקבע מתיקט הוקטור $(0, 1, 2)$ لكن ההצפנה היא הכפל הבא:

$$\begin{pmatrix} 4 & 18 & 5 \\ 3 & 4 & 14 \\ 5 & 3 & 9 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 28 \\ 32 \\ 21 \end{pmatrix} \equiv \begin{pmatrix} 6 \\ 10 \\ 21 \end{pmatrix} \pmod{22}.$$

פענוח $\mathbf{v} \cdot \mathbf{A}^{-1} \leftarrow \mathbf{u}$

- מכפילים כל בלוק במטריצה ההופכית.
- מטריצה תהיה הפיכה רק אם הדטרמיננטה גדולה מאפס.

כדי לפונח את הטקסט המוצפן תחיליה ממירmos אותו בחזרה לוקטור ואז מכפילים במטריצה ההופכית המודולרית. מטריצה הופכית היא מטריצה המקיים:

$$AA^{-1} = A^{-1}A = I_n \quad \text{כאשר } I_n \text{ היא מטריצת היחידה מסדר } n.$$

הדגמה לעיל המטריצה המודולרית ההופכית היא:

$$\mathbf{A}^{-1} = \begin{pmatrix} 8 & 9 & 6 \\ 5 & 11 & 7 \\ 11 & 6 & 14 \end{pmatrix}$$

והיא מתחילה למחרוזת "טיזהלהז". באלל שאמ מוחשבים את AA^{-1} מתיקט:

$$\begin{pmatrix} 4 & 18 & 5 \\ 3 & 4 & 14 \\ 5 & 3 & 9 \end{pmatrix} \cdot \begin{pmatrix} 8 & 9 & 6 \\ 5 & 11 & 7 \\ 11 & 6 & 14 \end{pmatrix} = \begin{pmatrix} 177 & 264 & 220 \\ 198 & 155 & 242 \\ 154 & 132 & 177 \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \pmod{22}.$$

לכן לפונח מוחשבים את:

$$\begin{pmatrix} 8 & 9 & 6 \\ 5 & 11 & 7 \\ 11 & 6 & 14 \end{pmatrix} \cdot \begin{pmatrix} 6 \\ 10 \\ 21 \end{pmatrix} = \begin{pmatrix} 264 \\ 287 \\ 420 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} \pmod{22}.$$

מתיקט הטקסט המקורי "אבג".

צפנים מודרניים

- צפנים מודרניים מבוססים ברובם על צפני בлок
- בכל שלב מצפינים קבוצת תווים ולאתו בודד
- מפתח ארוך – מגן מפני מתקפת כוח גס
- אפשר לפתח פונקציות הצפנה מורכבות
- ניתן להסביר צופן בлок לצופן זרימה
- תוכנות מסווגות לצפנים מודרניים
- מסתכלים על המסר המקורי סדרה של ביטים
- מבוססים על מחשב/מעבד

צפנים סימטריים מודרניים

- הצופן מבוסס על פעולה בסיסית של פעילים מספר פעמים, ובכל הפעלה נעשה שימוש במפתח שונה / בחלק שונה של המפתח
- הפעולה הבסיסית מורכבת ממספר שלבים שמשומשים באופן ייעיל בחומרה
- ערבות (Transportation) של כל הביטים בבלוק
- פועלות הצבה (Substitution)
- פונקציה מסווגת המופעלת על קבוצה קטנה של ביטים כל פעם
- פונקציה פחות מסווגת המוגדרת על כל הבלוק

התיאוריה של סודיות הצפנים

- **אנטropיה:** מספר השאלות כן/לא צריך על מנת לקבוע במידוייק את תוצאתו של אירוע מקרי.
- קיים קשר קרוב בין אנטropיה לייצוג מידע
- **אנטropיה והודעה:** המספר המינימלי של ביטים הנדרש כדי לבטא את כל ההודעות שאפשר.
- **בלבול (Confusion):** פונקציה קשה להיפוך וניתוח מתמטי, היחס בין המוצפן למפתח מסווג.
- **חלחול (Diffusion):** פיזור המידע על פני תחום רחוב
 - כל בית במסר המקורי משפיע על הרבה ביטים במסר המוצפן
 - כל בית במסר המוצפן משפיע מרובה ביטים במסר המקורי

מידדות Diffusion: צפנים מורכבים

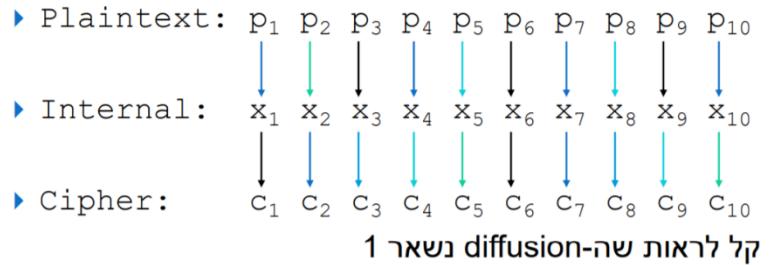
- לכל ריבב בצופן נחיש איך אותן בקלט (ה המקורי) משפיעה על אותיות בפלט (המוצפן), באותו אופן ניתן לבצע את הבדיקה הפוכה כאשר לכל אות בפלט נבדוק כמה אותיות בקלט משפיעות על הערך שלה.
- נתנו אופן לוגי איך בהצפנה כפולה, ההשפעה מחלוקת דרך צופן הביניים, לא מדובר בהכרח ככפל או חיבור של מספרים. החישוב תלוי בגורמים נוספים.
- אם הצופן מtabסס על פעולה לינארית, ה diffusion לא טוב. כמו ויג'נר, מפתח חד פעמי.
- אם הצופן מtabסס

דוגמה 1 : ויג'ינר כפול

בצופן ויג'ינר האות הגלואה במקומות ה- i קובעת לבדה את ערכה של האות המוצפנת באותה מקום ה- i .

- ככלומר, ה-diffusion הוא 1

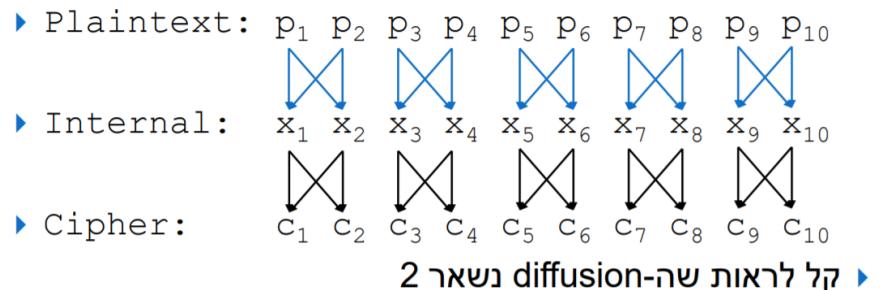
▸ נבחן הצפנה כפולת של ויג'ינר (מקירה כללי: מפתחות שונים
באורך שונה)

דוגמה 2 : פלייפר כפול

בצופן פלייפר, זוג אותיות סמוכות בגלוי קובעות יחס זוג אותיות במיקום זהה במסר המוצפן

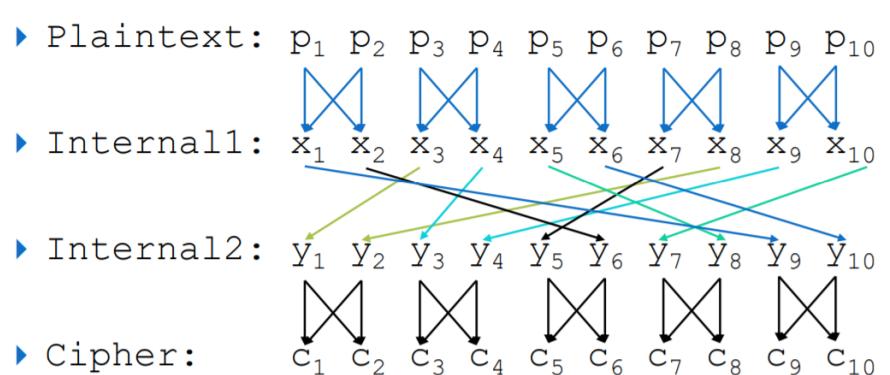
- ככלומר, ה-diffusion הוא 2

▸ נבחן הצפנה כפולת של פלייפר (מקירה כללי: טבלאות שונות,
נתעלם מאותיות סמוכות זהות)

דוגמה 3 : פלייפר + ערבות עמודות + פלייפר

- פלייפר: 2 = diffusion

- ערבות עמודות: 1 = diffusion , עם הערבות של האותיות

Diffusion: 3

מדידות Confusion בצדדים

- נוכל להעיר באופן השוואתי, לאיזה שני צפנים דומים יש יותר confusion.
- נניח שחלק מהמסר הגלי ידוע לנו (Known plaintext), מידת "הקלות" שבה יהיה לנו קל למצאו את המפתח ולפענו את יתר המסר מושפע גם מהconfusion, שילוב של מספר פעולות שונות, שחלקן לא לינאריות (כמו חיפוש בטבלה).
- אם אוטיות המפתח עצמן מיוצרת באופן אקראי, והצפן נראה כמו רצף אוטיות אקראי, אז ההconfusion מושלם.
- פענוח חלקו של הצפן ש כולל 26 אוטיות לדוגמא יאפשר לפענו את שאר הצפן, גורם לחסובים נמוך.
- ככל שיש פחות תאים בטבלה כך אופן השימוש בה מורכב יותר, והחסובים גדול יותר.

דוגמה 1: ויג'ינר

- אם ננחש נכון אורח המפתח L , מספיק שייגלו לנו L אוטיות רציפות כדי שנוכל לגלו את המפתח ולפענו את כל המסר.
- נוסחת הצפנה היא לינארית, ולכן ניתן לגלו את המפתח ע"י העברת אגפים פשוטה:
 - $C = K \rightarrow C - P \equiv K \pmod{26}$
 - המפתח הוא מחרורי, ולכן L אוטיות מפתח רציפות מגלוות לנו את כל היהת
- בגלל החסובים הנמוך, אם נגלה פחות מ- L אוטיות מהמסר הגלי, עדין נוכל לגלו חלק מאוטיות המפתח ולפענו את המסר באופן חלק.

דוגמה 2: מפתח חד פעמי

- הצפן מתבסס על חיבור אוטיות מפתח, שזו פעולה לינארית
- diffusion גראן
- אוטיות המפתח עצמן מיוצרות באקראי, ולכן הצפן נראה כמו רצף אוטיות אקראי
- תוכנה זו מתארת confusion מושלים
- בהינתן חלק מהמסר הגלי (Known plaintext), נוכל לגלו איזה אוטיות מפתח ייצרו אותו, אך לא לדעת שום דבר על חלקים אחרים של המפתח, מעבר לתוכנות הידעשות של השפה.

דוגמה 3: צופן מונואלפביתי

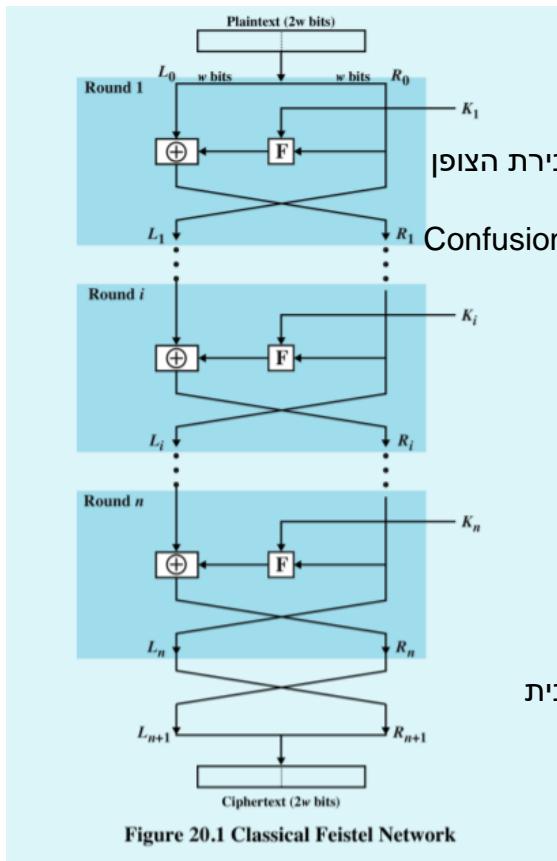
- הצפן אינו ניתן להציג כנוסחה לינארית ומבוסס על חיפוש בטבלה בגודל 26
- פענוח חלקו של הצפן שכולל את כל 26 האוטיות יאפשר לפענו את שאר הצפן
- שחזור הטבלה דורש אחת אחת לכל תא בטבלה
- מכאן שהחסובים נמוך, אבל מסיבות אחרות مثل ויג'ינר
- עקב החסובים הנמוך, פענוחים קצרים יותר מאפשרים לגלו את הטבלה באופן חלק, ולפענו חלקית.

דוגמה 4: פלייפר

- הצפן מתבסס על חיפוש אוטיות בטבלה, פעולה שלא ניתן למידול במשוואות לינאריות.
- הטבלה מכילה 25 תאים, אחד פחות מצופן מונואלפביתי, הטבלה מורכבת יותר והחסובים גדול.
- על כל זוג אוטיות סמוכות שנדיין לפענו:
 - נוכל לפענן זוגות נוספים זהים – עקב החסובים diffusion.
 - בעזרת מספיק אילוצים על הטבלה, נוכל לשחזר אותה ולפענו את שאר המסר.
 - החסובים תורם לתהילך מסודר באופן מדווג של שחזור הטבלה ע"י ניסוי וטעיה.

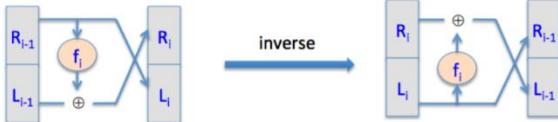
צופן Feistel

שיטה כללית לבניית צפוני בлок סימטריים



Claim: for all $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$
Feistel network $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ is invertible

Proof: construct inverse



- צופן פייסטל הוא תבנית כללית לבניית שיטות הצפנה.
- כל אלגוריתם הצפנה (P) EK חייב לפחות 2 תכונות:
 - הפונקציה Ek חייבת להיות חד חד ערכית
 - הפונקציה Ek חייבת להיות מסובכת, כדי למנוע את שבירת הצופן
- פייסטל מציע תבנית המפרידה בין הדרישות לבניית צופן ע"פ שיטה זאת צריכה לדאוג לפונקציה מסובכת – DES, Blowfish, CAST-128, Lucifer
- צפוני פייסטל מוכרים:

הסבר לתבנית ↗

- הצופן מטפל בבלוק P בן 2w ביטים
- הבלוק מחולק לשני חלקים בני w ביטים: $P = (L_0, R_0)$
- הסכימה מופעלת על (L_0, R_0) ליצור זוג חדש (L_1, R_1)
- לאחר מכן היא מופעלת על (L_1, R_1) ליצור זוג חדש (L_2, R_2)
- הזוג האחרון (L_n, R_n) הוא הבלוק המוצפן C
- מוציא צופן המבוסס על שיטת פייסטל, צריך:
 - להמציא פונקציה מסובכת $F(R, K)$ ולש滔ל אותה בתבנית
 - הפונקציה F מופעלת רק על החלק הימני של הבלוק
 - בכל שלב F מופעלת עם חלק אחר של המפתח K_i
 - הפונקציה אינה חייבת להיות חד-ע"ג ב-R

הפיוכות צופן פייסטל

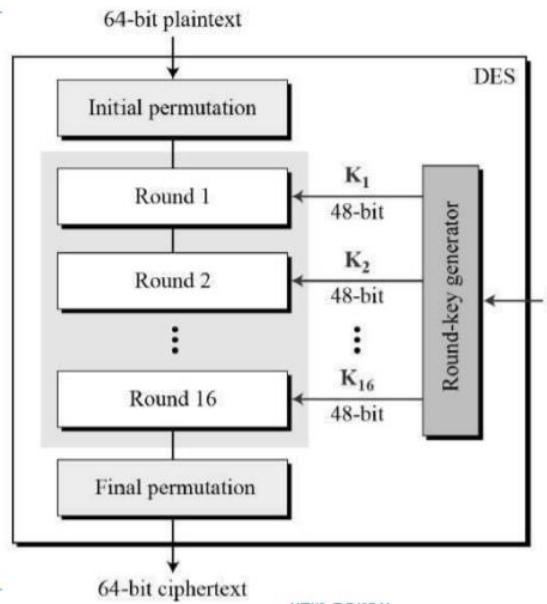
בכל שלב בצופן הפיר, גם אם F אינה הפיכה, لكن למתקנן הצופן יש חופש לבחור F בלי לדאוג להפיוכות, הפונקציה F מספקת את הסיבוכיות של השיטה.

הסבר כללי על הצופן

שיטת גנרטית לבניית פונקציה פנימית איטרטיבית של צופן בלבוקים. רשות פייסטל מחלקת את הטקסט הגלוי לשני חצאים, ימין ושמאל. בכל איטרציה מופעלת הפונקציה הפנימית על צד אחד כאשר פלט הפונקציה משמש להצפנת הצד השני באמצעות פעולה או-אקסקלוסיבי (XOR) ואילו החצי הראשון נותר ללא שינוי. באיטרציה הבאה מחליפים בין הצדדים כך שהפלט יהיה הקלט לצד השמאלי ולהפך.

הוכחת הפיות הצופן

- ▶ כל שלב בצופן בניין כך:
 - ▶ $(L_i, R_i) = (R_{i-1}, L_{i-1}) \oplus F(R_{i-1}, K_i)$
 - ▶ הוא XOR ביט-ביט
 - ▶ כדי להפוך את התהילה, נפרק לשתי נוסחאות:
 - ▶ $R_{i-1} = L_i$
 - ▶ $L_{i-1} \oplus F(R_{i-1}, K_i) = R_i$
 - ▶ נציב את הביטוי הראשון בשני:
 - ▶ $L_{i-1} \oplus F(L_i, K_i) = R_i$
 - ▶ נבצע XOR ל-F כדי להעביר אותו אגף:
 - ▶ $L_{i-1} = R_i \oplus F(L_i, K_i)$



צופן נתונים סטנדרטי – DES

- בניו ע"פ תבנית פיסטל
- כל בלוק גלי של 64 ביט הופך לבלוק צופן של 64 ביט
- המפתח מכיל 56 סיביות
- מפתח זה נגזרים 16 מפתחות בני 48 ביט

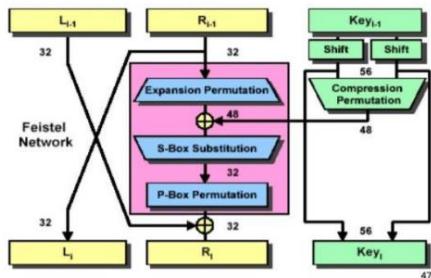
הסבר כללי על DES

תקן הצפנה מידע הוא צופן בלוקים סימטרי איטרטיבי. הוא מקבל בלוק טקסט קרייא שהוא מחרוזת של 64 סיביות (8 בתים) ומפיק בלוק מוצפן באורך זהה. הפונקציה הפנימית של הצופן מתבצעת בשש עשר חזרות בעזרת מפתח סודי באורך 64 סיביות (שמתוכנן רק 56 סיביות בשימוש). הפונקציה F של DES מכילה confusion & diffusion, בנוסף לפרMOVוטציות גלובליות על כל החלק הפעיל: החצי השני (השמאלי) רק מתחבר (XOR) לתוצאה זו.

DES Single Round

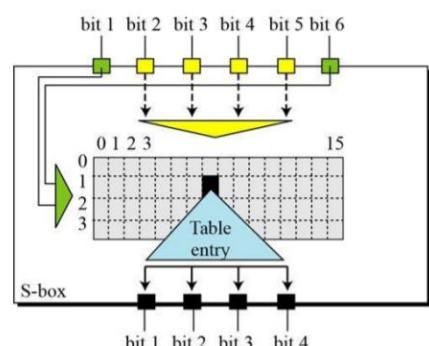
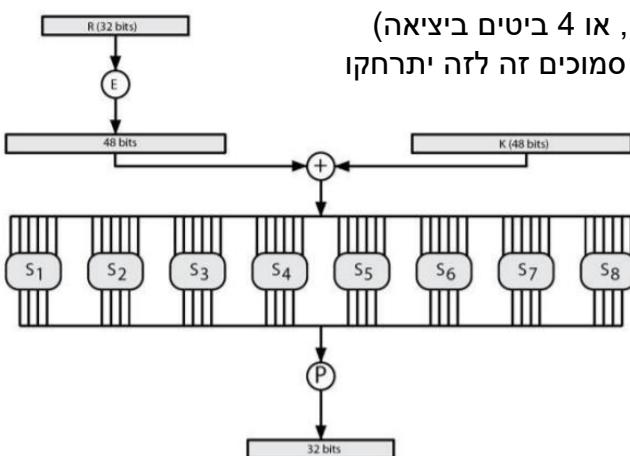
- הפונקציה F משתמשת בחצי הימני ובמפתח
- מבוצע XOR בין הפלט של F לבין החצי השמאלי, המועבר לيمין
- הפונקציה F מורכבת מארבע פעולות:
 - פרMOVוטציה + הרחבה
 - חלק מהביטים משוכפלים כדי להפוך 32 ביט ל-48 ביט
- XOR עם מפתח
- החלפה בעזרת טבלה, שמצומת חזרה ל-32 ביט
- פרMOVוטציה פשוטה
- בניית המפתח מבוססת על פרMOVוטציה עם דחיסה
- חלק מהביטים נזרקיים כדי לצמצם 56 ביט ל-48 ביט

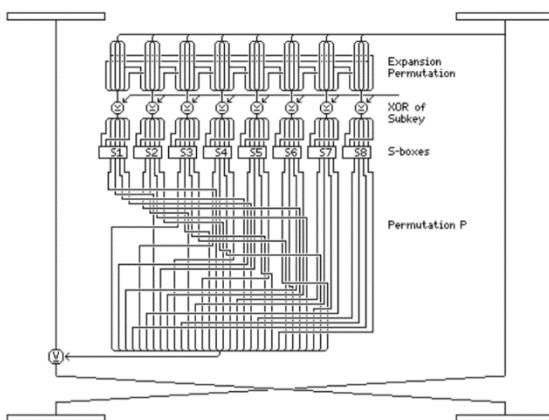
DES: Single Round



S-Boxes

- מבצעים את ה-confusion
- הטבלאות אינן לינאריות
- תרומות לחסוך diffusion באופן מקומי (לכל 6 ביטים בכניסה, או 4 ביטים ביציאה)
- לאחר הפרMOVוטציה של ה-P-Box (שלב 4), ביטים שהו סמוכים זה לזה יתרחקו
 - لكن באיטרציה הבאה יכנסו ל-S-Boxes שונים
 - מה שיגדיל את ההdiffusion diffusion לכל חצי הבלוק



שימוש ה-S-Boxes

- להלן טבלת ה-S-box הראשון:
- עבור הקלט 10001111 ניקח את עמודה 1 (0001) שורה 3 (11)
- הפלט הוא 1100 (12)
- דוגמה נוספת: קלט 01110110, פלט 110110

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 14 | 04 | 13 | 01 | 02 | 15 | 11 | 08 | 03 | 10 | 06 | 12 | 05 | 09 | 00 | 07 |
| 1 | 00 | 15 | 07 | 04 | 14 | 02 | 13 | 10 | 03 | 06 | 12 | 11 | 09 | 05 | 03 | 08 |
| 2 | 04 | 01 | 14 | 08 | 13 | 06 | 02 | 11 | 15 | 12 | 09 | 07 | 03 | 10 | 05 | 00 |
| 3 | 15 | 12 | 08 | 02 | 04 | 09 | 01 | 07 | 05 | 11 | 03 | 14 | 10 | 00 | 06 | 13 |

DES חזק

- הצפן נחשב חזק, קיימות חולשות מינוריות שאין ניתנות לניצול באופן פרקטני.
- DES יש חולשה מבנית למתקפת כוח גס, לאחר והמפתח הוא 56 ביט בלבד וכך המחשב עולה עם הזמן
- ישנם $\sim 2^{56} \times 7 \times 10^{16}$ מפתחות אפשריים

סוגי תקיפות לDESתקיפת Plaintext

- נתקח זוג ידוע (P, R) – מסר מקור P ובלוק צופן תואם C
- וננסה את כל המפתחות האפשריים
- ברגע שנמצא מפתח K עבורו $C = E_K(P)$, נטען שה K הוא המפתח, בסבירות גבוהה
- הסביר:
 - אם K אינו המפתח הנכון, אז הפענוח התקבל במקרה
 - הסיכוי שמפתח אקריאי יצפין את P ל C הוא 2^{-64} , כלומר כמעט בלתי אפשרי

תקיפת Ciphertext Only

- נתקח בלוק מצפן ונפענה אותו בכל המפתחות האפשריים
- מפתח K הנutan תוצאה משמעותית הוא בסבירות גבוהה המפתח הנכון
- נתקישה לעשות זאת אם מסר המקור אינו טקסט

מפתח חלש: מפתח שימושים בו בצווף ספציפי, גורם לצופן להתנהג בצורה שלא נרצה.

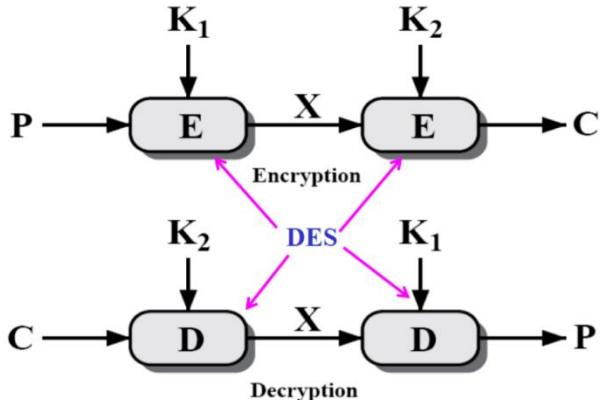
- דוגמה: מפתחות זהים לאוטו קוד במערכות, מפתחות אלטרנטיביים עם שינויים קלים, מפתחות מאוד קלים.

שבירת צופן DES

- מפתחות חלשים וחצי חלשים:

- חלש: $E_K(P) = P$

- חצי חלש בזוגות: $E_{K1}(P) = D_{K2}(P)$



הצפנה כפולה 2DES

- הצפן עמיד בפני מתקפת כוח גס פשוטה
- יש לעבור על כל המפתחות האפשריים (K_1, K_2) 2^{112} אפשרויות במוצע עד להצלחה
- התקיפה Known Plaintext Meet in the middle מצורצת את התקיפה : Meet in the middle

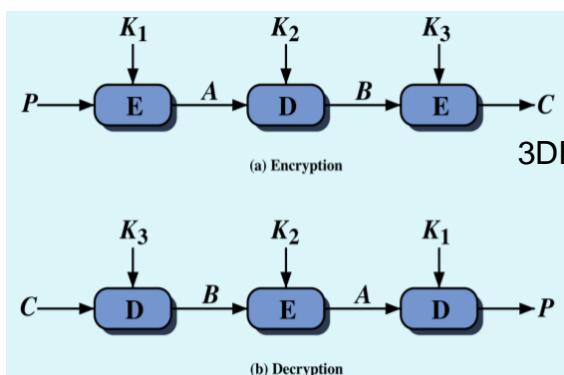
Meet In The Middle

ניקח שני זוגות של בלוקים ידועים (P, C) ו- (P^*, C^*) כך שמתקיים ($P, C) \neq (P^*, C^*$)
ע"י הפעלת D_{K2} על שני האגפים, באופן שקול מתקיים: $D_{K2}(C) = E_{K1}(P)$ ו- $D_{K2}(C^*) = E_{K1}(P^*)$

השיטה עצמה:

- נפעיל את כל המפתחות האפשריים K_1 על P
- נרשום טבלת תוצאות ($E_{K1}(P), K_1$), לפי האינדקס $E_{K1}(P)$
- נחשב $D_{K2}(C)$ לכל ערך K_2 , ונבדוק אם התוצאה מופיעה בטבלה
- אם התוצאה מופיעה בטבלה, אז נסמן K_1 שנמצא בטבלה
- אם הזוג (K_1, K_2) מופיע גם את (P^*, C^*) , סביר להניח שהוא המפתח המבוקש

הצפנה בשלושת 3DES

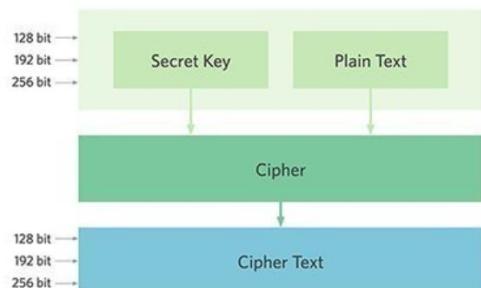


- עמיד בפני הפונקציה במקומות ההצפנה השנייה מקובל Meet in the Middle
- שימוש בפעולת הפונקציה במקום במקומות ההצפנה השנייה רגיל
- כאשר כל חלקו המפתח שווים מקבלים DES DES עם חומרת רגיל
- אפשר תאימות לאחר: הפעלת DES עם חומרת DES ניתן להפעיל עם 2 מפתחות במקום 3
- $K_1 = K_3$
- עדין עמיד בפני Meet in the Middle
- עדין בשימוש עד היום, אבל איטיות פי 3

AES – Advanced Encryption Standard

- עובד בבלוקים של 128 ביט
- מחליף את DES לאחר DES הבסיסי לא נחassoc לבטוח, וDES3 לוקח פי 3 מהזמן לחישוב
- אינו צופן פיסטול
- מבוסס על סבבים בני 4 שלבים
- הצפנה סימטרית – משתמש באותו מפתח להצפנה ופענוח

AES Design



אופן פעולה AES

- הצופן מסתכל על הבלוק של 128 ביט באופן הבא:
 - 4 עמודות, שבכל אחת יש מילה (word) של 32 ביט (4 בתים)
 - מטריצה 4×4 של בתים. כל בית מכיל מספר בתחום 0-255
- בכל סבב הפעולות מתבצעות על כל בלוק
 - נודא שהצפנה הינה הפיכה
 - עמיד בפני מתקפות ידועות
- מבוסס על פעולות ثنائية לחשב אותן ביעילות

אין AES פועל, גרסה בה המפתח בן 128 ביט ויש לו 10 סבבים

- המפתח מורחב למערך של 44 מיילים (מילה = 4 בתים)
- כל 4 מיילים הם 128 ביט שיישמשו לאחד הסבבים
- 4 המיילים הראשוניים משמשות לשלב ההתחלה
- בכל סבב, מבצעים על תוצר הסבב הקודם 4 פעולות:
 - **Byte substitution** : כל בית מוחלף בbite אחר לפי טבלת S-box
 - **Shift rows** : הזזה ציקלית של שורות המטריצה
 - **Mix columns** : כפל כל עמודה במטריצה במטריצה אחרת
 - **Add round key** : פעולה XOR עם המפתח
- הסבב הראשון והאחרון מעט שונים
 - הפענוח נעשה ע"י היפוך כל פעולה, מהוסף להתחלה

הסבר כללי על הצופן:

$AES(x)$

$s = x;$

$s = s \oplus K[1];$

For i=1 to i=Nr-1 do {

 s=SubBytes(s);

 s=ShiftRows(s);

 s=MixColumns(s);

 s = s $\oplus K[i+1];$ }

 s=SubBytes(s);

 s=ShiftRows(s);

 s = s $\oplus K[Nr+1];$

 y = s;

Output y;

צופן בבלוקים סימטריו, להצפנה נתונים מסוימים מסיבית, ישנו 10 סיבובים למפתח באורך 128 ביט.

רוב הפעולות בצופן הן פעולות אלגבריות על בתים.

מסיבות יולות נבחר ייצוג פולינומי רגיל.

בניסוח הרשמי: $A(x) = a_7x^7 + a_6x^6 + \dots + a_1x + a_0, a_i \in GF(2) = \{0, 1\}$

הצעד הראשון של הצופן הוא הכנסת המידע למערך.

הטרנספורמציה הראשונה היא החלפת המידע באמצעות substitution table.

הטרנספורמציה השנייה מזיזה מידע בשורות.

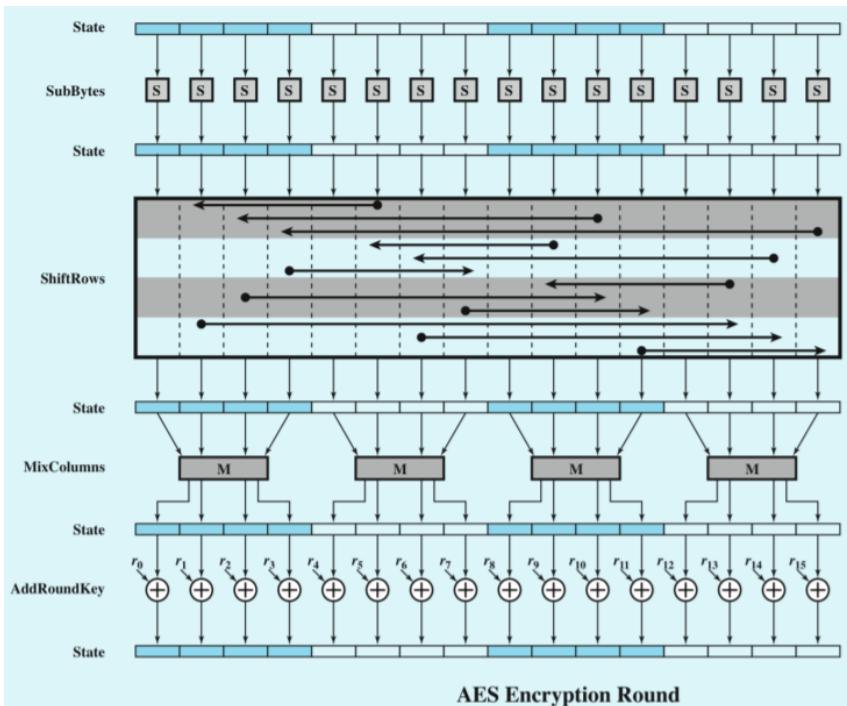
הטרנספורמציה השלישית מעלבבת עמודות.

הטרנספורמציה האחרונה מבוצעת על כל עמודה עם שימוש בחלק אחר של מפתח ההצפנה (פעולה XOR).

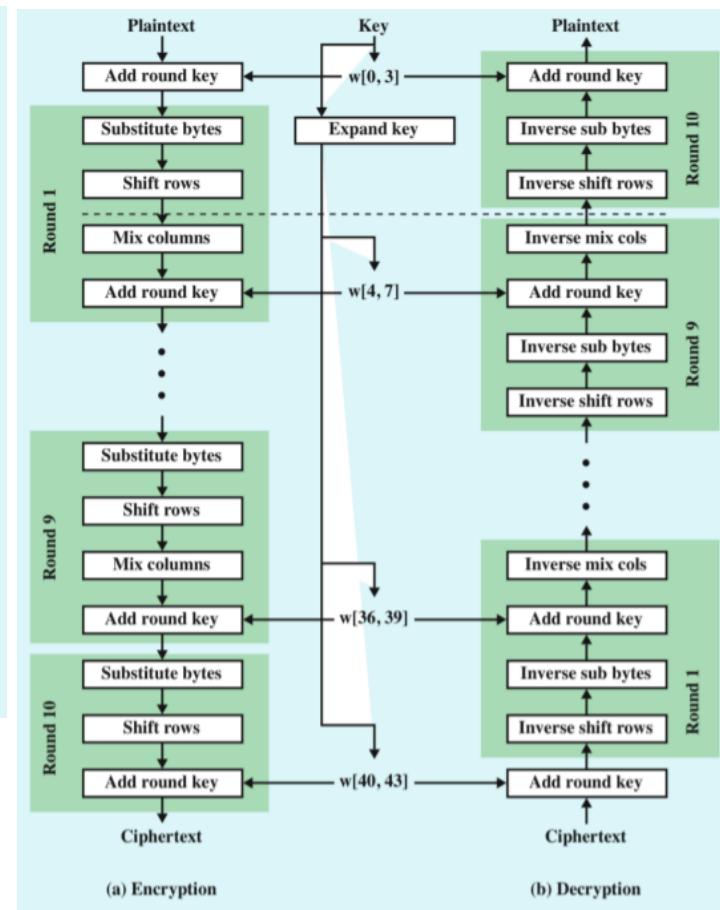
*מפתחות ארוכים יותר יצטרכו יותר סיבובים להשלמת ההצפנה.

מבנה סיבב

מבנה כללי הצפנה ופענוח



AES Encryption Round



(a) Encryption

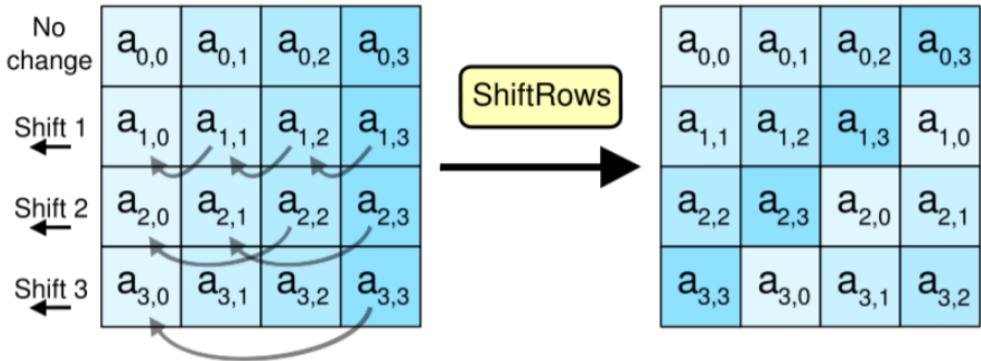
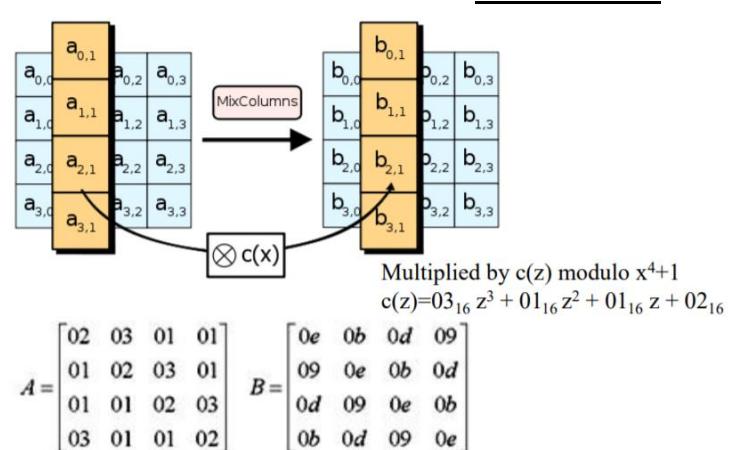
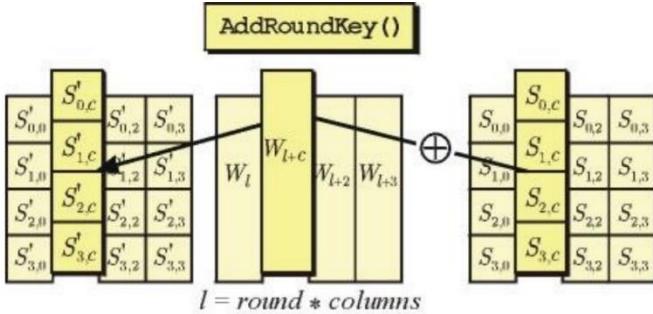
(b) Decryption

תבלת S-box עברות Byte Substitution

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|
| | | x | y | | | | | | | | | | | | | | |
| 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 01 | 67 | 2B | FE | D7 | AB | 76 | |
| 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 | |
| 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 | |
| 3 | 04 | C7 | 23 | C3 | 18 | 96 | 05 | 9A | 07 | 12 | 80 | E2 | EB | 27 | B2 | 75 | |
| 4 | 09 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 | |
| 5 | 53 | D1 | 00 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF | |
| 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 02 | 7F | 50 | 3C | 9F | A8 | |
| 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 | |
| 8 | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 | |
| 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB | |
| A | E0 | 32 | 3A | 0A | 49 | 06 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 | |
| B | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 08 | |
| C | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A | |
| D | 70 | 3E | B5 | 66 | 48 | 03 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E | |
| E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF | |
| F | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 | |

תבלת S-box

| | | y | | | | | | | | | | | | | | | |
|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| x | 0 | 52 | 09 | 6A | D5 | 30 | 36 | A5 | 38 | BF | 40 | A3 | 9E | 81 | F3 | D7 | FB |
| | 1 | 7C | E3 | 39 | 82 | 9B | 2F | FF | 87 | 34 | 8E | 43 | 44 | C4 | DE | E9 | CB |
| | 2 | 54 | 7B | 94 | 32 | A6 | C2 | 23 | 3D | EE | 4C | 95 | 0B | 42 | FA | C3 | 4E |
| | 3 | 08 | 2E | A1 | 66 | 28 | D9 | 24 | B2 | 76 | 5B | A2 | 49 | 6D | 8B | D1 | 25 |
| | 4 | 72 | F8 | F6 | 64 | 86 | 68 | 98 | 16 | D4 | A4 | 5C | CC | 5D | 65 | B6 | 92 |
| | 5 | 6C | 70 | 48 | 50 | FD | ED | B9 | DA | 5E | 15 | 46 | 57 | A7 | 8D | 9D | 84 |
| | 6 | 90 | D8 | AB | 00 | 8C | BC | D3 | 0A | F7 | E4 | 58 | 05 | B8 | B3 | 45 | 06 |
| | 7 | D0 | 2C | 1E | 8F | CA | 3F | 0F | 02 | C1 | AF | BD | 03 | 01 | 13 | 8A | 6B |
| | 8 | 3A | 91 | 11 | 41 | 4F | 67 | DC | EA | 97 | F2 | CF | CE | F0 | B4 | E6 | 73 |
| | 9 | 96 | AC | 74 | 22 | E7 | AD | 35 | 85 | E2 | F9 | 37 | E8 | 1C | 75 | DF | 6E |
| | A | 47 | F1 | 1A | 71 | 1D | 29 | C5 | 89 | 6F | B7 | 62 | 0E | AA | 18 | BE | 1B |
| | B | FC | 56 | 3E | 4B | C6 | D2 | 79 | 20 | 9A | DB | C0 | FE | 78 | CD | 5A | F4 |
| | C | 1F | DD | A8 | 33 | 88 | 07 | C7 | 31 | B1 | 12 | 10 | 59 | 27 | 80 | EC | 5F |
| | D | 60 | 51 | 7F | A9 | 19 | B5 | 4A | 0D | 2D | E5 | 7A | 9F | 93 | C9 | 9C | EF |
| | E | A0 | E0 | 3B | 4D | AE | 2A | F5 | B0 | C8 | EB | BB | 3C | 83 | 53 | 99 | 61 |
| | F | 17 | 2B | 04 | 7E | BA | 77 | D6 | 26 | E1 | 69 | 14 | 63 | 55 | 21 | 0C | 7D |

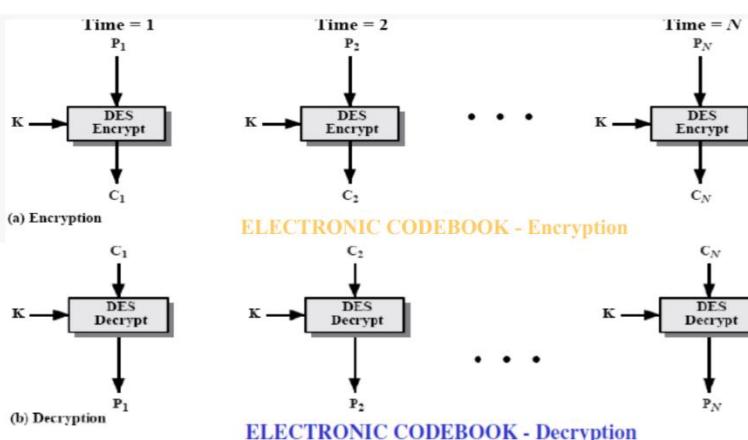
Shift RowsAdd Round

באיורים הבאים ניתן להחליף את DES בכל צופן בлок אחר

הפעלת צופן בлок על מסר ארוך

- מסר ארוך מגודל בлок יש לחלקם בגודל בлок, אז להפעיל את הצופן מספר פעמים
- שיטות של הצפנה בлок:

- **הצפנה כל בлок בנפרד – (ECB)**
- **הצפנה בлок בזיקה לקודם – (CBC)**
- **Counter (CTR)**
- **קידום המפתח מבלוק לבלוק –**
- **שיטות של הצפנה זרימה (Stream):**
 - Cipher Feedback (CFB)
 - Output Block Feedback (OFB)
 - Rivest Cipher 4 (RC4)



הפעלה DES בשיטת ECB

העקרון:

ECB מצפין כל בлок של מסר-מקור בנפרד כלומר, אותו בлок גלי יוצפן תמיד לאותו בлок מוצפן. כמו ספּר קוד שבו לכל מילה מותאמת מילת קוד ייחודית משלה.

חולשה – הצפנה מסר ארוך:

1. בלוקים זהים בכתב הצופן **זהים** גם במקור ← הזרמתו ריגול
2. התוקף יכול לחזור על אותו בлок למען **התקפת Replay**
3. התוקף יכול **להחליף בлок** בבלוק אחר מתוך כתב הצופן
4. הצפנה בלוקים רבים בעלי קשר תוכן באותה שיטה הצפנה מקלה על **תקיפה קריפטואנליזטית**

שיטת Cipher Block Chaining ,CBC

$$\text{ההצפנה: } C_0 = \text{IV}, \quad C_i = E_K(C_{i-1} \oplus P_i)$$

- לפני הצפנה בлок P_i מוסיפים XOR לבלוק המוצפן הקודם
- לשם אתחול מגדרים ווקטור התחלתי $\text{IV} = C_0$ הנשלח גם הוא ליעד
- יש לשים לב כי שילוב מידע משלב קודם מזכיר את Autokey, אבל שם השתמשנו בו.

$$\text{הפענה: } C_0 = \text{IV} \quad C_i = E_K(C_{i-1} \oplus P_i) \quad (\text{ידוע ביעד})$$

יתרונות: הצפנה כל בлок תלויה בבלוקים הקודמים, ולכן:

1. לא רואים בכתב הצופן איזה בלוקים זהים במסר המקורי
2. העתקת בлок צופן במקומות אחרים לא תיתן תוצאה ידועה
- פותר את חולשות-1-3 בהצפנה ECB, ובמידה רבה גם את חסרון 4.

פיתוח השיטה להצפנה בблокים

- כיצד משפייע שימוש בצופן על תהליכי הפענוח?
- לעיתים יש הבדל בין שימוש בזמן חישוב הצופן לבין שימוש בשליחת הצופן.
- אם יש שימוש קטן נרצה שעדין יהיה ניתן לפענוח את רוב המסר.
- האם התוקף יכול ליצור שימוש מכוון בצופן מוביל להתגלות?
- שכפול בלוקים, שינוי מיקום של בלוקים, שינוי ביטים בודדים במסר וכו' .
- במידה והתקוף מסוגל ליצור שימוש מכוון מוביל להתגלות, מדובר בסיכון ל Integrity .
- כיום יש שיטות אינומת שמאפשרות לזהות שימושים אלו.
- האם ניתן לבצע הצפנה/פענוח באופן מקביל?
- ניתן להשיג זאת עם חיבור ב-XOR כמו בפנקס חד פעמי.
- האם ניתן לבצע את מרבית החישובים הנדרשים להצפנה/פענוח מראש?
- עיבוד מוקדים, עוד לפני שקיבלו את הטקסט המקורי/מצפן.

CBC

- שימוש בלוק צופן אחד בקשרו של משבש את הפענוח בהמשך
- התקוף יכול לשנות את P_i ביעד כרצונו
- אם $i = IV$ (ווקטור התחלה) נשלח בגלוי, ניתן לשנות את P_1 כרצונו ע"י שינוי IV
- אם IV קבוע, ניתן לדעת אם המסר חוזר על עצמו ומתי משתנה

על מנת לפתר בעיות אלו: ← פוטר את החיסרון השלישי והרביעי

1. בוחרים תמיד IV אקריא'
2. מצפינים אותו בעזרת ECB לפני צירופו לכתב הצופן

פירוט תקיפת CBC:

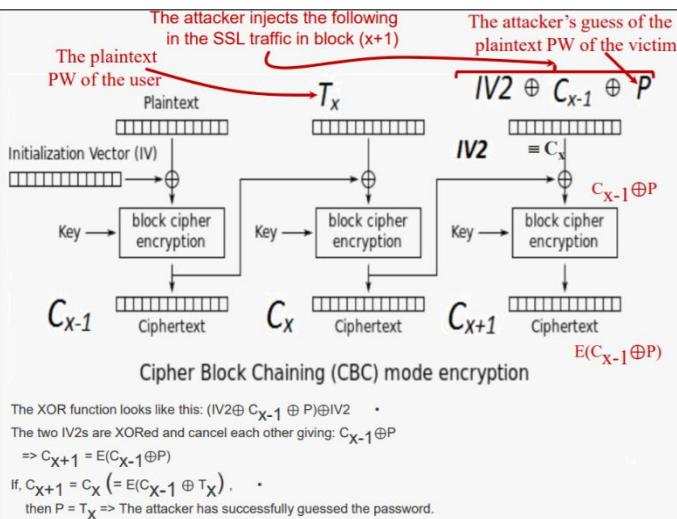
- פונchio בלוק i הוא ע"פ הנוסחה : $P_i = C_{i-1} \oplus D_K(C_i)$ (ידוע ביעד)
- נניח שאנו יודעים את בלוק המקורי P_i ורוצים לשנותו לבלוק Q_i .
- לשם כך נחשב את הפרש XOR של שני הבלוקים: $G_i = Q_i \oplus P_i$
- עכשו נרצה את C_{i-1} לערך $*C_{i-1}$ ע"י: $C_{i-1}^* = G_i \oplus C_{i-1}$
- אז כאשר היעד ירצה לקבל את P_i הוא יקבל במקומו: $P_i = Q_i \oplus D_K(C_i) = G_i \oplus P_i = Q_i \oplus C_{i-1}^*$
- מחיר התקיפה: במקום $i-1$ יתקבל ביעד זבל (אך שאר המסר יהיה נכון)

שימוש בתקיפה: ניתן לשנות ספירה במשכורת של חשבון שימוש שעיה העקרות שלא יתגלה

Beast – Beast Attack

Browser Exploit Against SSL/TLS

- פרוטוקול 3.0 SSL ופרוטוקול 1.0 TLS שהחליף אותו, מספקים שכבת אבטחה מעיל TCP.
- אשר נעשו בה שימוש בין היתר ב-HTTPS, הגרסה המאובטחת של HTTP.
- התקפה "התקפת גלו-נבחר" בשיטת אדם באמצעות לנוחש מפתח השיחה.
- השיטה מנצלת חולשה ב-SSL/TLS-גרסת 1.0 שבה וקטור האתחול של חבילות מידע מוצפנות היה תוצאה הצפנה של חבילה קודמת. החוקרים הדגימו באמצעות תוכנה וקוד JavaScript איך התקוף מחלץ וקטור אתחול מהעוגיות, מזריק מידע כלשהו לבקשות מוצפנות מנסה לנחש טקסטים גלוים כלשהם ומחפש התאמה לטקסט המוצפן בית אחר בית.

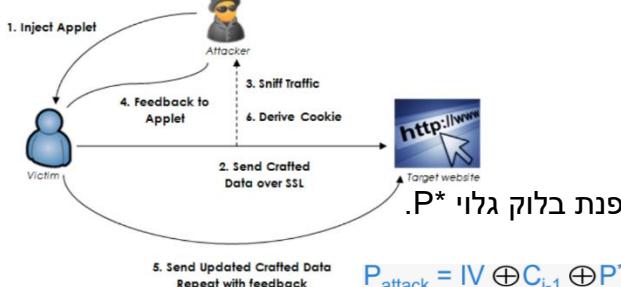


התקפה מוסתרת על ידי מיזוג ההודעות הזוגיות עם הודעות הקורבן.

מטרים אבטחה, העוגיות נגמרות על המחשב במצב שנקרא HTTP only שבו סקירה פיטים שרצים על הדפדפן לא יכולים לקרוא בעצמם את תוכן העוגיות. תוך כדי שעה של האזנה לחיבור מוצפן עם כמות מסוימת של טקסטים מוצפניהם שמקורם ידוע, התקוף יכול לפרק עוגיות מוצפנות, לדעת את המקור שלהם ואת הקורבן, בקשوت מוצפנות, לדעת את המקור שלהם ואת הקורבן, ובעצם להשתלט על השיחה.

התקפה מבוססת על חולשה נוספת של CBC, היא נחשבת לשיכון תיאורתי בלבד.

אך הحل מגרסה 1.1 של TLS היא אינה קיימת עוד.



$$C_{\text{attack}} = E_k(IV \oplus P_{\text{attack}}) = E_k(IV \oplus IV \oplus C_{i-1} \oplus P^*) = E_k(C_{i-1} \oplus P^*)$$

$$P_{\text{attack}} = IV \oplus C_{i-1} \oplus P^*$$

SSL/TLS ב-CBC חולשה

- מתקפה מסוג Chosen Plaintext
- מאפשרת לתוקף לבדוק האם בלוק מוצפן C_i התקבל מהצפנה בבלוק גלי P^*
- בבלוק i נוצר ע"י: $C_i = E_k(C_{i-1} \oplus P)$
- התקוף יבקש להצפן טקסט גלי המתחיל בבלוק הבא:

- IV : וקטור האתחול של ההודעה החדשה
- C_{i-1} : בלוק הצופן הקודם לבlok ההודעה המקורי אותה מנסים לתקוף.
- P^* : הניחוש של התקוף לבlok הטקסט הגלי P .

- התקוף יקבל את הבלוק המוצפן הבא: $(P^*)^* = C_{i-1}$
- אם התקוף צודק בನיחוש, כולם $P = P^*$ יתקבל

TLS CBC IV attack

- Force Alice to send her password T_X
- Eavesdrop and get $C_X = E(\text{Key}, C_{X-1} \oplus T_X)$
- Let P be a blind guess of T_X
- Force Alice to send plaintext $C_X \oplus C_{X-1} \oplus P$
- Alice sends $C_j = E(\text{Key}, C_X \oplus C_X \oplus C_{X-1} \oplus P)$
- $C_{X+1} = E(\text{Key}, C_{X-1} \oplus P)$
- If $C_{X+1} == C_X$ ($= E(\text{Key}, C_{X-1} \oplus T_X)$)
- then $P = T_X$

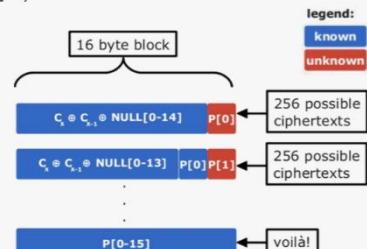
This requires a lot of guessing and it is not very handy

BEAST (Browser Exploit Against SSL/TLS)

- Force Alice to send NULL[0-14] $T_X[0]$
- Eavesdrop and get $C_X = E(\text{Key}, C_{X-1} \oplus \text{NULL}[0-14] T_X[0])$
- Let P be a blind guess of $T_X[0]$
- Force Alice to send plaintext $C_X \oplus C_{X-1} \oplus \text{NULL}[0-14] P$
- Alice sends $C_{X+1} = E(\text{Key}, C_X \oplus C_X \oplus C_{X-1} \oplus \text{NULL}[0-14] P)$
- $C_{X+1} = E(\text{Key}, C_{X-1} \oplus \text{NULL}[0-14] P)$
- If $C_X == C_{X-1}$
- then $P = T_X[0]$

This requires up to $2^8 = 256$ guesses.

We can do this!

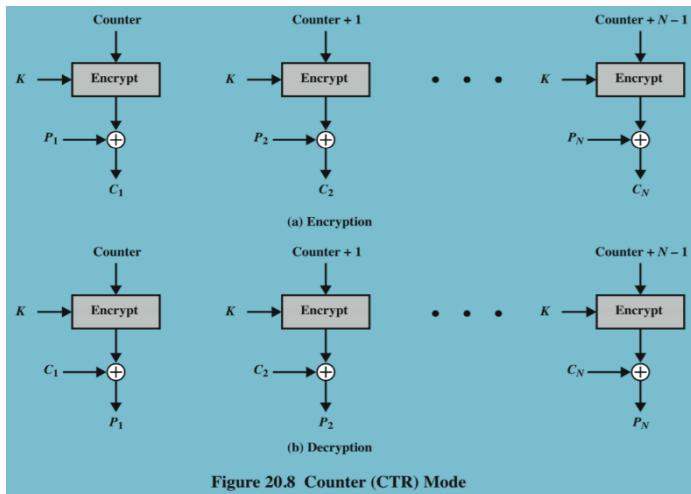


- כיצד יודע התקוף מה יהיה IV בהודעה הבאה שתישלח?
- בפרוטוקולים SSL 3.0 ו-TLS 1.0, בלוק הצפן האחרון מגיע מההודעה הקודם ששימוש IV לבאה
- כיצד מגיע התקוף למצב שבו הוא יכול לבצע מתקפת Chosen Plaintext?
- התקוף מאמין לתשדורות היוצאות של הקורבן ומצליח לפנות את הדפדפן של הקורבן להרץ סקירה זדוני

שיטת Counter

יתרונות

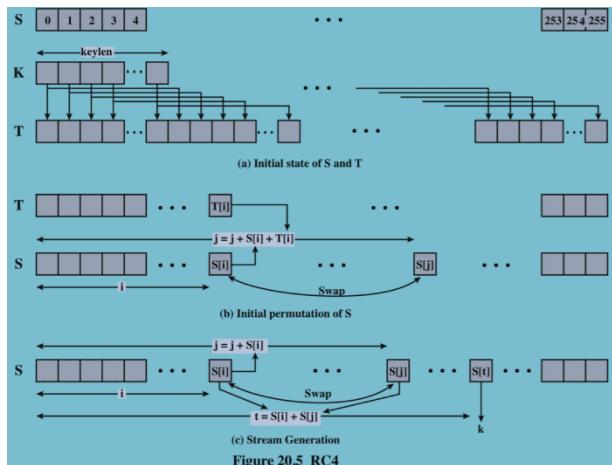
- **יעיבוד מקביל:** ניתן להצפין/לפענח בлокים במקביל
- **יעיבוד מוקדים:** ניתן לחשב מראש את רוב אלגוריתם ההצפנה ולבצע XOR מול הגלוי/צופן
- **גישה אקראית:** ניתן לפענח בлок בודד באמצעות הצופן,
- **בטחה:** השיטה בטוחה מפני פיצוח הצופן, לא מפני שיבושים מכוניים.
- **פשטות:** פונקציית ההצפנה זהה לפונקציית הפענוח



| Cipher | Key Length | Speed (Mbps) |
|--------|------------|--------------|
| DES | 56 | 21 |
| 3DES | 168 | 10 |
| AES | 128 | 61 |
| RC4 | Variable | 113 |

אלגוריתם RC4

- צופן זרימה "אמיתי" ולא הסבה של צופן בлокים
- מפתח בגודל משתנה
- מבוסס על פרמוטציה אקראית ופעולות פשוטות על בתים:
- מאפשר הצפנה ופענוח מהירים יותר מאשר צופן בлок, קל למימוש



מבנה האלגוריתם RC4

- האלגוריתם משתמש בשני וקטורים של 256 בתים
- הווקטור S מאותחל במספרים 0 עד 255, לפי הסדר
- הווקטור T מאותחל בחזרות של המפתח
- הווקטור T משמש ליצירת פרמוטציה מהווקטור S
 - $j := 0$
 - for ($i=0; i < 256; i++$) {
 - $j := (j + S[i] + T[i]) \text{ mod } 256;$
 - swap($S[i], S[j]$);
 - }

תיאור האלגוריתם – ההצפנה בפועל

ההצפנה עצמה משתמש רק בווקטור S, ומשנה את הפרמוטציה אחרי כל אות מפתח

```

- j := 0; i := 0;
do {
    i := (i + 1) mod 256;
    j := (j + S[i]) mod 256;
    swap(S[i], S[j]);
    K := S[(S[i] + S[j]) mod 256];
    output K; // use K to encrypt/decrypt one byte
} while(required);

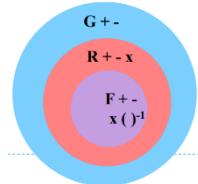
```

מספרים ראשוניים

- המספר k הוא ראשוני אם ורק אם אין לו מחלקים פרט לעצמו ול 1 . המספר הראשוני הקטן ביותר הוא 2 .
 - רשימה של המספרים הראשוניים הקטנים מ- 200 :
- 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127
131 137 139 149 151 157 163 167 173 179 181 191 193 197 199

פירוק לגורמים ראשוניים

- כתיבה של מספר כמכפלה של מספרים קטנים היא פירוק לגורמים $c \times b \times a = n$
 - פירוק לגורמים ראשוניים של מספר a כתיבתו כמכפלה של מספרים ראשוניים
- תזכורת – 2 מספרים שלמים הם זרים אם ורק אם אין להם מחלקים משותפים פרט ל- 1 ,
ואם ורק אם $\gcd(a, b) = 1$



3 מבנים אלגבריים בסיסיים

- קבוצה (Group): סט של אלמנטים עם אופרטורים $+$, $-$
 - טבעת (Ring): סט של אלמנטים עם אופרטורים $+$, $-$, \times
 - שדה (Field): סט של אלמנטים עם אופרטורים $+$, $-$, \times , \times^{-1}
- תזכורת – ההופכי הכפלי ב \mathbb{Z} : חוג, הופכי ושדה.

משפט פרמה הקטן

(Fermat's little Theorem)

יהי p מספר ראשוני, אז:

זהו משפט מרכזי בתורת המספרים עם חשיבות
בהצפנה א-סימטרית ובודיקת ראשוניות

- אבחנות:
 - טענה b הינה מקורה פרטני של טענה c
 - טענה c נובעת מטענה a
- (a) אם $1 \equiv a^{p-1} \pmod{p}$, אז $\gcd(a, p) = 1$
- (b) לכל a , מתקיים $a^p \equiv a \pmod{p}$
- (c) לכל k , מתקיים $a^{k(p-1)+1} \equiv a \pmod{p}$

19 = (10011)₂ , $M^{19} \bmod 115$ הנוסחה: כל בלוק M יוצפן ע"פ

(bit=1): $20 \bmod 115 = 20$ (1)
(bit=2): $(20^2) \bmod 115 = 55$ (2)
(bit=3): $(55^2) \bmod 115 = 35$ (3)
(bit=4): $20 \cdot (35^2) \bmod 115 = 5$ (4)
(bit=5): $20 \cdot (5^2) \bmod 115 = 40$ (5)

$$40 \div 22 = 1 \text{ Rem } 18$$

$$\Rightarrow 40 = 1 \cdot 22^1 + 18 \cdot 22^0$$

למען חישוב חזקות מודולריות על מספרי ענק שלא ניתן לאחסן בזיכרון

חישוב חזקות מהיר

- נחשב את החזקה בסדרה של מכפלות
- אחרי כל מכפלה, נבצע \bmod על התוצאה
- בשום שלב לא נאלץ ליזג מספרים גדולים מ- 2^n

חישוב חזקות מהיר: דוגמה

אלגוריתם איטרטיבי

חישוב $a \bmod n$, ישום מקובל בתוכנה
היצוג הבינארי של n הוא $b_k b_{k-1} \dots b_1 b_0$

```

f = 1
for i = k down to 0
    do
        f = (f * f) mod n
    if b_i == 1 then
        f = (f * a) mod n
return f

```

40

| | |
|--|--|
| $a^{25} \bmod n$ $25 = (11001)_2$ Start: 1 R 1 (bit=1): a R 2 (bit=1): a(a ²) R 3 (bit=0): [a(a ²)] ² R 4 (bit=0): [(a(a ²)] ²) ² R 5 (bit=1): a[[a(a ²)] ²] ² יש לבצע \bmod בכל שורש סה"כ 6 כפלים והעלות בריבוע | $7^{25} \bmod n$ $25 = (11001)_2$ Start: 1 R 1 (bit=1): 7 R 2 (bit=1): 3 R 3 (bit=0): 9 R 4 (bit=0): 13 R 5 (bit=1): 10 |
|--|--|

השלב הראשון לא נספר כי פעולות על המספר 1 טרייאליות

```

f = 1
for i = k down to 0
    do
        f = (f * f) mod n
    if b_i == 1 then
        f = (f * a) mod n
return f

```

בדיקות ראשוניות

- קיימת שיטה דטרמיניסטית המבוססת על שיטת הנפה:
- על מנת לבדוק אם n הוא ראשוני, יש לחלקו בכל הראשוניים הקטנים מ(n) \sqrt{n}
- שיטה זו אינה יסימה עבור מספרים גדולים
- נתבוסס על מבחן סטטיסטי – השיטה מהירה יותר מאשר דטרמיניסטיות, **מבחן מילר-רבין**
- נמצא תמונה Q כך ש:
 - מספרים ראשוניים תמיד מקיימים את התמונה Q
 - מספרים פריקים מקיימים את התמונה Q בהסתברות קטנה

תכונות המבדילות בין מספרים ראשוניים למספרים פריקים

- מספרים פריקים ניתן לפרק לגורמים, אך לא נוכל להסתמך על שיטה זו
 - מספרים ראשוניים מקיימים את משפט פרמה הקטן:
 - לכל $a < n$ מתקיים $a^{n-1} \equiv 1 \pmod{n}$
 - האם המשפט מתקיים עבור ח' פריק (במקום ק' ראשוני)?
 - אם a אינו זר ל- n , המשפט אינו מתקיים?
 - האם אפשר למצוא a (שאינו בהכרח זר ל- n פריק) עבורו המשפט אינו מתקיים?
 - לא תמיד. למעשה, יש מספרים פריטיים הנקראים **מספרים קרמייקל**, עבורם לכל a שזר ל- n מתקיים ($a \pmod{n}$) $a^{n-1} \equiv 1$
- מספרים קרמייקל ראשוניים: 1729, 1105, 561

מבחן מילר-רבין , מבוסס על משפט פרמה הקטן

ח' הוא המספר שרצים לבדוק אם הוא ראשוני
a הוא מספר אקראי שיעזר בבדיקה

Miller-Rabin(n):

- ▶ בחרו d מספרים אקראיים בלתי תלויים $1 < a_1, a_2, \dots, a_d < n-1$
- ▶ עבור $j=1, 2, \dots, d$ בצעו את $\text{Test}(n, a_j)$
- ▶ אם מבחן אחד (לפחות) החזיר "ח' פריק"
או ח' הוא פריק
- ▶ אם כל המבחנים החזירו "יתכן ש-ה' ראשוני"

או בהסתברות גבוהה זו הוא ראשוני

TEST(n, a):

- ▶ מצאו מספרים $0 < k < q$ אי זוגי, עבורם $q^k = n-1$
- ▶ אם $1 = n \pmod{a^k}$ אז החזר: **"יתכן ש-ה' ראשוני"**

לכל j מ-0 ועד $1-k$, חשב:
אם $(a^{2^j})^q \pmod{n} = 1$

או החזר: **"יתכן ש-ה' ראשוני"**

הchezr: **"ח' פריק"**

$n=13$ (Prime), $n-1 = 12 = 2^2 \times 3$; $q=3$, $k=2$

▶ נבחר ב"אקרים" $a=2$

$$a^q = 2^3 = 8 \not\equiv 1 \pmod{13}$$

ולכן נמשיך לולאה
($j=0$) $a^{2^q} = 2^6 = 64 \equiv 12 = n-1 \pmod{13}$

$$(j=1) \quad a^{2^q} = 2^6 = 64 \equiv 12 = n-1 \pmod{13}$$

מחזיר "יתכן ש-13 ראשוני"

▶ נוסחה $3 = 27 \equiv 1 \pmod{13}$, קיבל: Test(13,2)

▶ נוסחה $3 = 27 \equiv 1 \pmod{13}$, קיבל: Test(13,3)

מספר המבחנים d באלגוריתם מילר-רבין

משפט: בהינתן מספרים טבעיים n ו- $a < n$

אם n ראשוני, $\text{Test}(n, a)$ \rightarrow 1

אם n פריק ו- a נבחר באקראי, ה הסתברות שהמבחן יחזיר

"יתכן ש- n ראשוני" קטנה מ- $\frac{1}{4}$

מסקנה: אם נבחר ערך מסוים גדול של d ,

אם עבור ערך כלשהו של a קיבלנו $"1"$ הוא פריק" או בודאות,

הו פריק

אם ב- d מבחנים בלתי תלויים, קיבלנו בכלל "יתכן ש- n

ראשוני", אז כמעט בוודאות a ראשוני

אם a פריק, הסיכוי שנחליט שהוא ראשוני $> \frac{3}{4}$

פונקציית אוילר (Φ)

- קבוצת השאריות מודולו n היא $\{0, 1, \dots, n-1\}$
- קבוצת השאריות המצוימות מודולו n היא השאריות שזרות לו, כלומר הפיקות ב- \mathbb{Z}_n //מספרים ראשוניים
- לדוגמה, עבור $n=10$ קבוצת השאריות: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ והמצוימות: $\{1, 3, 7, 9\}$
- פונקציית אוילר מונה את **מספר האיברים בקבוצת השאריות המצוימות (הראשוניים)**, מודולו n

чисוב פונקציית אוילר

- יותר נוח לחשב את הפונקציה Φ "מציאת השאריות שאין בקבוצה = מספר המספרים שאינם זרים לו"
- ישנה נוסחה לאוילר, אך היא דורשת לפרק לגורמים ראשוניים את n

$$\begin{aligned} \Phi(p) &= p-1 & \text{אם } p \text{ ראשוני} \\ \Phi(pq) &= (p-1)(q-1) & \text{אם } p \neq q \text{ ראשוניים} \end{aligned}$$

לדוגמא:

$$\Phi(37) = 36$$

$$\Phi(21) = \Phi(3 \times 7) = (3-1) \times (7-1) = 2 \times 6 = 12$$

נוסחה כללית לפונקציית אוילר

יהי $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdots p_k^{\alpha_k}$ כאשר p_i הם גורמים ראשוניים של n , וה- α_i הן החזקות שלהם, בהתאם

$$\Phi(n) = (p_1-1) p_1^{\alpha_1-1} \cdot (p_2-1) p_2^{\alpha_2-1} \cdots (p_k-1) p_k^{\alpha_k-1}$$

דוגמא :

$$72 = 8 \cdot 9 = 2^3 \cdot 3^2 \quad \blacktriangleright$$

$$\Phi(72) = (2-1) \cdot 2^{3-1} \cdot (3-1) \cdot 3^{2-1} = 1 \cdot 2^2 \cdot 2 \cdot 3^1 = 2^3 \cdot 3 = 24 \quad \blacktriangleright$$

משפט אוילר

משפט (Euler)

יהי $a < n$ מספר טבעי. מתקיים :

$$a^{\Phi(n)} \equiv 1 \pmod{n}, \text{ איז } \gcd(a, n) = 1 \quad \text{(a)}$$

$$a^{k\Phi(n)+1} \equiv a \pmod{n} \quad \text{לכל } k, a, \text{ מתקיים (b)}$$

דוגמא לחלק (a) :

$$a=3, n=10; \Phi(10)=(2-1)*(5-1)=4; \\ \text{לכן, } a^{\Phi(n)} = 3^4 = 81 \equiv 1 \pmod{10}$$

$$a=2, n=15; \Phi(15)=(3-1)*(5-1)=8;$$

$$\text{לכן, } a^{\Phi(n)} = 2^8 = 256 \equiv 1 \pmod{15}$$

דוגמא לחלק (b) :

$$a=2, k=3, n=10; (\text{not relatively prime}): \Phi(10)=4; \\ a^{k\Phi(n)+1} = 2^{3 \times 4 + 1} = 2^{13} = 8192 \equiv 2 \pmod{10} \quad \text{לכן (c)}$$

- משפט אוילר הוא הרחבה של משפט פרמה
- משפט פרמה הוא עבור n ראשוני בלבד
- משפט אוילר הוא לכל n
- אם a הוא ראשוני, אז משפט אוילר מתכנס למשפט פרמה

-
-
-
-

שורשים פרימיטיביים

- יהי k מספר ראשוני. לפיכך \mathbb{Z}_k הוא שדה.
- המספרים $\{1, 2, \dots, k-1\}$ הם קבוצה סגורה לכפל, הנקראת החבורה הכפלית מודולו k .
- מספר $k > 1$ נקרא שורש פרימיטיבי של k אם $a^m \equiv 1 \pmod{k}$ והמספר הקטן ביותר m הוא $k-1$.
- ע"פ משפט פרמה הקטן, $(k-1) \equiv a^{k-1} \pmod{k}$ אם $a \not\equiv 0 \pmod{k}$.
- אם a שורש פרימיטיבי של k ואם r חזקota של a יוצרות את החבורה הכפלית מודולו k , כלומר: $\{a^r, a^{2r}, a^{3r}, \dots, a^{(k-1)r}\} = \{1, 2, \dots, k-1\}$.
- מה זה שורש פרימיטיבי? מספר שאם נסתכל על כל חזקוטיו, מייצר את כל המספרים בחז \mathbb{Z}_k פרט ל-0.

Another equivalent definition of a primitive root mod n is (from Wikipedia),

a number g is a primitive root modulo n if every number coprime to n is congruent to a power of g modulo n

For example, 3 is a primitive root modulo 7, but not modulo 11, because

Modulo 7,

$$3^0 \equiv 1, 3^1 \equiv 3, 3^2 \equiv 2, 3^3 \equiv 6, 3^4 \equiv 4, 3^5 \equiv 5, 3^6 \equiv 1 \pmod{7}$$

And you got all the possible results: 1, 3, 2, 6, 4, 5. You can't get a 0, but 0 is not coprime to 7, so it's not a problem. Hence 3 is a primitive root modulo 7.

Whereas, modulo 11,

$$3^0 \equiv 1, 3^1 \equiv 3, 3^2 \equiv 9, 3^3 \equiv 5, 3^4 \equiv 4, 3^5 \equiv 1 \pmod{11}$$

And modulo 11, you only got the possible values 1, 3, 9, 5, 4 and the sequence starts repeating after 3^5 , so you will never get a $3^k \equiv 2$, for example. Hence 3 is *not* a primitive root modulo 11.

אם a הוא שורש פרימיטיבי של p , אז השאריות

$\{a^1, a^2, a^3, \dots, a^{p-1}\}$ שוונות זו זו זו (מודולו p)

אחרת, אחת מהחזקות תתן שרירות 1, ומכאן הסדרה תחזר על עצמה

דוגמא: מצאו את השורשים הפרימיטיביים של 7

אין צורך בחישוב חזקוטות מהיר. אנו מחשבים את כל החזקוטות ולכ

נכפול את a בעצמו שוב ושוב (ונבצע מודולו אחרி כל כפל)

$$7 \pmod{1} \\ 7 \pmod{2} \\ 7 \pmod{4} \\ 7 \pmod{8} \\ 7 \pmod{16} \\ 7 \pmod{32} \\ 7 \pmod{64} \\ 7 \pmod{128} \\ 7 \pmod{256} \\ 7 \pmod{512} \\ 7 \pmod{1024} \\ 7 \pmod{2048} \\ 7 \pmod{4096} \\ 7 \pmod{8192} \\ 7 \pmod{16384} \\ 7 \pmod{32768} \\ 7 \pmod{65536} \\ 7 \pmod{131072} \\ 7 \pmod{262144} \\ 7 \pmod{524288} \\ 7 \pmod{1048576} \\ 7 \pmod{2097152} \\ 7 \pmod{4194304} \\ 7 \pmod{8388608} \\ 7 \pmod{16777216} \\ 7 \pmod{33554432} \\ 7 \pmod{67108864} \\ 7 \pmod{134217728} \\ 7 \pmod{268435456} \\ 7 \pmod{536870912} \\ 7 \pmod{1073741824} \\ 7 \pmod{2147483648} \\ 7 \pmod{4294967296} \\ 7 \pmod{8589934592} \\ 7 \pmod{17179869184} \\ 7 \pmod{34359738368} \\ 7 \pmod{68719476736} \\ 7 \pmod{137438953472} \\ 7 \pmod{274877906944} \\ 7 \pmod{549755813888} \\ 7 \pmod{1099511627776} \\ 7 \pmod{2199023255552} \\ 7 \pmod{4398046511104} \\ 7 \pmod{8796093022208} \\ 7 \pmod{17592186044416} \\ 7 \pmod{35184372088832} \\ 7 \pmod{70368744177664} \\ 7 \pmod{140737488355328} \\ 7 \pmod{281474976710656} \\ 7 \pmod{562949953421312} \\ 7 \pmod{1125899906842624} \\ 7 \pmod{2251799813685248} \\ 7 \pmod{4503599627370496} \\ 7 \pmod{9007199254740992} \\ 7 \pmod{18014398509481984} \\ 7 \pmod{36028797018963968} \\ 7 \pmod{72057594037927936} \\ 7 \pmod{144115188075855872} \\ 7 \pmod{288230376151711744} \\ 7 \pmod{576460752303423488} \\ 7 \pmod{1152921504606846976} \\ 7 \pmod{2305843009213693952} \\ 7 \pmod{4611686018427387904} \\ 7 \pmod{9223372036854775808} \\ 7 \pmod{18446744073709551616} \\ 7 \pmod{36893488147419103232} \\ 7 \pmod{73786976294838206464} \\ 7 \pmod{147573952589676412928} \\ 7 \pmod{295147905179352825856} \\ 7 \pmod{590295810358705651712} \\ 7 \pmod{1180591620717411303424} \\ 7 \pmod{2361183241434822606848} \\ 7 \pmod{4722366482869645213696} \\ 7 \pmod{9444732965739290427392} \\ 7 \pmod{18889465931478580854784} \\ 7 \pmod{37778931862957161659568} \\ 7 \pmod{75557863725914323219136} \\ 7 \pmod{151115727458228646438272} \\ 7 \pmod{302231454916457292876544} \\ 7 \pmod{604462909832914585753088} \\ 7 \pmod{1208925819665829171506176} \\ 7 \pmod{2417851639331658343012352} \\ 7 \pmod{4835703278663316686024704} \\ 7 \pmod{9671406557326633372049408} \\ 7 \pmod{19342813114653266744098816} \\ 7 \pmod{38685626229306533488197632} \\ 7 \pmod{77371252458613066976395264} \\ 7 \pmod{154742504917226133952785328} \\ 7 \pmod{309485009834452267905570656} \\ 7 \pmod{618970019668904535811141312} \\ 7 \pmod{1237940039337808571622282624} \\ 7 \pmod{2475880078675617143244565248} \\ 7 \pmod{4951760157351234286489130496} \\ 7 \pmod{9903520314702468572978260992} \\ 7 \pmod{19807040629404937145956521984} \\ 7 \pmod{39614081258809874291913043968} \\ 7 \pmod{79228162517619748583826087936} \\ 7 \pmod{158456325035239497167652178872} \\ 7 \pmod{316912650070478994335304357744} \\ 7 \pmod{633825300140957988670608715488} \\ 7 \pmod{1267650600281915977341217430976} \\ 7 \pmod{2535301200563831954682434861952} \\ 7 \pmod{507060240112766390936486972384} \\ 7 \pmod{1014120480225332781872939944768} \\ 7 \pmod{2028240960450665563745879889536} \\ 7 \pmod{4056481920901331127491759779072} \\ 7 \pmod{8112963841802662254983519558144} \\ 7 \pmod{16225927683605324509967039116288} \\ 7 \pmod{32451855367210649019934078232576} \\ 7 \pmod{64903710734421298039868156465152} \\ 7 \pmod{129807421468842596079736312930304} \\ 7 \pmod{259614842937685192159472625860608} \\ 7 \pmod{519229685875370384318945251721216} \\ 7 \pmod{103845937175074076863789050344232} \\ 7 \pmod{207691874350148153727578100688464} \\ 7 \pmod{415383748700296307455156201376928} \\ 7 \pmod{830767497400592614910312402753856} \\ 7 \pmod{1661534994801185229820624805507712} \\ 7 \pmod{3323069989602370459641249611015424} \\ 7 \pmod{6646139979204740919282499222030848} \\ 7 \pmod{13292279958409481838564998444061696} \\ 7 \pmod{26584559916818963677129996888123392} \\ 7 \pmod{53169119833637927354259993776246784} \\ 7 \pmod{106338239667279454708519987552493568} \\ 7 \pmod{212676479334558909417039975104987136} \\ 7 \pmod{425352958669117818834079950209974272} \\ 7 \pmod{850705917338235637668159900419948544} \\ 7 \pmod{1701411834676471273336318900839891888} \\ 7 \pmod{3402823669352942546672637801679783776} \\ 7 \pmod{6805647338705885093345275603359567552} \\ 7 \pmod{1361129467741177018669055120671913504} \\ 7 \pmod{2722258935482354037338110241343826708} \\ 7 \pmod{5444517870964708074676220482687653416} \\ 7 \pmod{1088903574192941614935244096535316632} \\ 7 \pmod{2177807148385883229870488193070633264} \\ 7 \pmod{4355614296771766459740976386141266528} \\ 7 \pmod{8711228593543532919481952772282533056} \\ 7 \pmod{17422457187087065838963905544565066112} \\ 7 \pmod{34844914374174131677927811089130132224} \\ 7 \pmod{69689828748348263355855622178260264448} \\ 7 \pmod{13937965749669652671171144435652052896} \\ 7 \pmod{27875931499339305342342288871304105792} \\ 7 \pmod{55751862998678610684684577742608211584} \\ 7 \pmod{111503725997357221369369155485216423688} \\ 7 \pmod{223007451994714442738738310970432847376} \\ 7 \pmod{446014903989428885477476621940865694752} \\ 7 \pmod{89202980797885777095495324388173138904} \\ 7 \pmod{178405961595771554190986648776346277808} \\ 7 \pmod{356811923191543058381973297552692555616} \\ 7 \pmod{713623846383086116763946595105385111232} \\ 7 \pmod{1427247692766172233527893190210770222464} \\ 7 \pmod{2854495385532344467055786380421540444928} \\ 7 \pmod{570899077106468893411157276084308089856} \\ 7 \pmod{1141798154212937786822314532168616179712} \\ 7 \pmod{228359630842587557364462906433723235944} \\ 7 \pmod{456719261685175114728925812867446471888} \\ 7 \pmod{913438523370350229457851625734892943776} \\ 7 \pmod{1826877046740700458915703251469785887552} \\ 7 \pmod{365375409348140091783140650293957177504} \\ 7 \pmod{730750818696280183566281300587914355088} \\ 7 \pmod{1461501637392560367132562601178286710176} \\ 7 \pmod{2923003274785120734265125202256573420352} \\ 7 \pmod{5846006549570241468530254404513148440704} \\ 7 \pmod{1169201309914048293706050880902629688144} \\ 7 \pmod{2338402619828096587412101761805259376288} \\ 7 \pmod{4676805239656193174824203523610518753576} \\ 7 \pmod{9353610479312386349648407047221037507152} \\ 7 \pmod{18707220958624772699296814094442075014304} \\ 7 \pmod{37414441917249545398593628188884150028608} \\ 7 \pmod{74828883834499090797187256377768300057216} \\ 7 \pmod{149657767668998181594374512755536600114432} \\ 7 \pmod{299315535337996363188749025511073200228864} \\ 7 \pmod{598631070675992726377490501022146400457728} \\ 7 \pmod{1197262141351985452754981002044292800915456} \\ 7 \pmod{2394524282703970905509962004088585601830912} \\ 7 \pmod{4789048565407941811019924008177171203661824} \\ 7 \pmod{9578097130815883622039848016354342407323648} \\ 7 \pmod{19156194261631767244079696032678684814647296} \\ 7 \pmod{38312388523263534488159392065357369629294592} \\ 7 \pmod{76624777046527068976318784130714739258589184} \\ 7 \pmod{153249554093054137952637568261429478571778368} \\ 7 \pmod{306499108186108275905275136522858957443556736} \\ 7 \pmod{612998216372216551810550273045717914887113472} \\ 7 \pmod{1225996432744433103621100546091435829742226944} \\ 7 \pmod{2451992865488866207242201092182871659484453888} \\ 7 \pmod{4903985730977732414484402184365743318968907776} \\ 7 \pmod{9807971461955464828968804368731486637937815552} \\ 7 \pmod{1961594292391092965793760873746293327967563104} \\ 7 \pmod{3923188584782185931587521747492586655935126208} \\ 7 \pmod{7846377169564371863175043494985173311870252116} \\ 7 \pmod{15692754339128743726350086989970346623740504332} \\ 7 \pmod{31385508678257487452700173979940693247481008664} \\ 7 \pmod{62771017356514974905400347959881386494962017328} \\ 7 \pmod{125542034713029949810800699199762772989924034656} \\ 7 \pmod{251084069426059899621600198399525545979848069312} \\ 7 \pmod{502168138852119799243200396799051091959696138624} \\ 7 \pmod{100433627770423959848640079359810208391939227728} \\ 7 \pmod{200867255540847919697280018719620416783878455456} \\ 7 \pmod{40173451108169583939456003743924083356775691092} \\ 7 \pmod{80346902216339167878912007487848166713551382184} \\ 7 \pmod{160693804432678335757824014975696333427052764368} \\ 7 \pmod{321387608865356671515648029951392666854055528736} \\ 7 \pmod{642775217730713343031296059852785333708111057532} \\ 7 \pmod{1285550435461426686062592119705570667416222115064} \\ 7 \pmod{2571100870922853372125884239411141334832444230128} \\ 7 \pmod{5142201741845706744251768478822282669664888460256} \\ 7 \pmod{10284403483691413488503536957644565339329777920512} \\ 7 \pmod{20568806967382826977007073915289130796659555841024} \\ 7 \pmod{41137613934765653954014147830578261593319111682048} \\ 7 \pmod{82275227869531307908028295661156523186638223364096} \\ 7 \pmod{16455045573906261581605659132231304637327644672192} \\ 7 \pmod{32910091147812523163211318264462609274655289344384} \\ 7 \pmod{65820182295625046326422636528925218549310576688768} \\ 7 \pmod{13164036459125089265284527305785043709862115337534} \\ 7 \pmod{26328072918250178530569054611570087419724230675072} \\ 7 \pmod{52656145836500357061138109223140174839448461350144} \\ 7 \pmod{105312291673000714122276218446280349678896922700288} \\ 7 \pmod{210624583346001428244552436892560699357793845400576} \\ 7 \pmod{421249166692002856489104873785121398715587690801152} \\ 7 \pmod{842498333384005712978209747570242797431175381602304} \\ 7 \pmod{168499666676801142595641949514048559466235076320464} \\ 7 \pmod{336999333353602285191283899028096118932470152640928} \\ 7 \pmod{673998666707204570382567798056192237864803035218556} \\ 7 \pmod{1347997333414409140765135596112384475729606070437112} \\ 7 \pmod{2695994666828818281530271192224768951458921414854224} \\ 7 \pmod{5391989333657636563060542384449537902917842829708448} \\ 7 \pmod{10783978667155273126121084768899075805835685659416896} \\ 7 \pmod{21567957334305546252242169537798151611671371318833792} \\ 7 \pmod{43135914668610592504484339075596303223342742637667584} \\ 7 \pmod{86271829337221185008968678151192606446685485275335168} \\ 7 \pmod{17254365867444237001793356103038521289337091055067032} \\ 7 \pmod{34508731734888474003586712206077042578674182110134064} \\ 7 \pmod{69017463469776948007173424412154085157348364220268128} \\ 7 \pmod{13803492693955389601434844882430817034969672844053624} \\ 7 \pmod{27606985387910779202869689764861634069939345688107248} \\ 7 \pmod{55213970775821558405739379529723266139878691376214496} \\ 7 \pmod{11042794155164311681148755905944653227975738275242892} \\ 7 \pmod{22085588310328623362297511811889306455951476550485784} \\ 7 \pmod{44171176620657246724595023623778601271902953100971568} \\ 7 \pmod{88342353241314493449190047247557202543805906201943136} \\ 7 \pmod{17668470648262898689838009449511440508761181240386672} \\ 7 \pmod{35336941296525797379676004899022880101522362480773344} \\ 7 \p$$

הצפנה סימטרית

הצפנה במפתח יחיד – השולח והנמען באותו מפתח, אם המפתח מתגלה פרטיות התקשרות נפגעת

מגבליות הצפנה במפתח יחיד

- B יכול לקבל הודעה מ-A ולהתחשך לכך ש-A הוא מקור הודעה
בשני המקרים לא ניתן להוכיח מי מבינם משקר
- A יכול לשЛОח הודעה ל-B ולהוכיח שיש לה אותה
- במסמך פיזי (hard copy) A יכול לחותם על המסמר,
- במקרה שהחתימה יכולה להעיד מי מבינם משקר. שירות זה נקרא **עקרון אי ההכחשה** (non repudiation)
- הצפנה סימטרית אינה מאפשרת גרסה דיגיטלית לחתימה, מכאן שהוא לא תומכת בעקרון אי ההכחשה

הצפנה אסימטרית

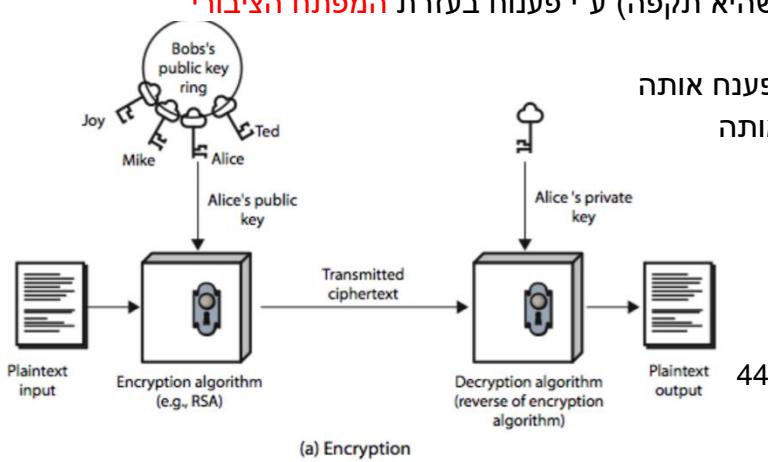
- נקראת **הצנת מפתח ציבורי**, ההצנה עשויה שימוש בשני מפתחות
 - מפתח ציבורי / **פומבי** (public key)
 - מפתח פרטי (private key)
- ההצנה אסימטרית כי הצדדים אינה שווים
 - הם מוחזקים מפתחות שונים
 - אחד הצדדים יכול לבצע פעולות שהשני אינו יכול לבצע
- שירות **משלים** להצנה אסימטרית, ולא תחולף שלו
- השימוש בהצנה אסימטרית נעשו במקרים מסוים לו יתרון

מדוע משתמשים בהצנה אסימטרית

- **הפצת מפתחות** – כיצד לנגן תקשורת בטוחה מבלי להסתמך על גוף שלישי (KDC)
- **חתימה דיגיטלית** – על מנת להוכיח שההודעה נכתבת ע"י מחבר מסוים

אופן פעולה

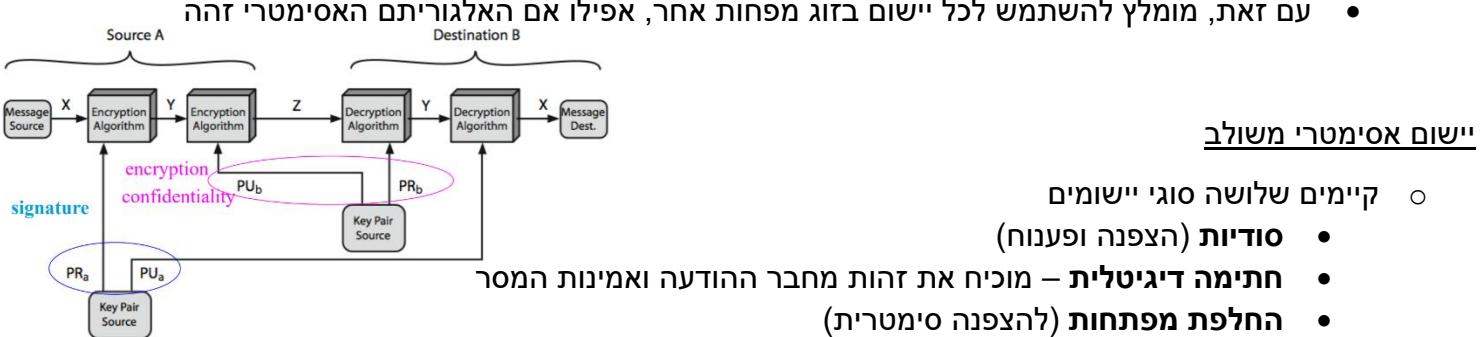
- הצנה אסימטרית עשויה שימוש בשני מפתחות, אחד להצנה והשני לפענוח
 - מפתח פומבי/ציבורי הידוע לכל
 - מפתח פרטי, הידוע רק לבעליו
- **ישום חיסין (Confidentiality)**
 - כולם מצפים הודעות עבור A, בעזרתו **המפתח הציבורי** של A
 - רק A יוכל לפענוח את ההודעות, כיוון שיש צורך **במפתח פרטי**
- **ישום חתימה (non Repudiation)**
 - A חותם על המסמר ע"י הצפנתו **במפתח פרטי** שלו
 - כל אחד יכול לתקוף את החתימה (לודא שהיא תקפה) ע"י פענוח בעזרתו **המפתח הציבורי**
- **ישומים אלו הם אסימטריים** מכיוון ש:
 - המפתח שהצפן את הודעה אין יכול לפענוח אותה
 - המפתח שייצר את החתימה אין מתקף אותה



מאפייני הצפנה אסימטרית

אלגוריתמי הצפנה אסימטרית עושים שימוש שני מפתחות, עבורם מתקיים:

- בלתי אפשרי מבחינה חשובית למצוא את המפתח הפרט, אם יודעים רק את האלגוריתם ואת המפתח הציבורי
- באופן יחסית הרבה יותר קל להצפין ולפענה כאשר המפתח הנכון (ציבורי או פרטי) ידוע
- כל אחד משני המפתחות יכול לשמש להצפנה, ובן זוגו לפענה
- תכונה זו מאפשרת להשתמש באותו אלגוריתם גם עבור יישום החיסון וגם עבור יישום החתימה
- בין היתר, RSA מקיים תכונה זו
- עם זאת, מומלץ להשתמש לכל יישום בזוג מפתחות אחר, אפילו אם האלגוריתם האסימטרי זהה



- ישנים אלגוריתמים להצפנה אסימטרית המתאימים לכל השימושים, ויש שמתאים רק לחלקם
- הצפנה אסימטרית הרבה יותר מהירה - הצפנה DES מהירה פי 1000 ~ 100 ~ מ RSA
- لكن נשתמש במפתח אסימטרי כדי לשולח מפתח סימטרי, ונשתמש בו להצפנה מידע עצמו

אבטחה בהצפנה אסימטרית

- בדומה להצפנה סימטרית, תמיד קיימת אפשרות לחיפוש המפתח **בכח גס**, אבל המפתחות ארוכים מדי
- רמת הבטחון של הצופן תלוי בפער בין הפעולות השונות:
 - פעולות ההצפנה והפענוח הן באופן יחסית **קלות לחישוב**
 - קרייפטואנליזה של הצופן משמעותית יותר **קשה** ← החישוב **לא מעשי**
- אלגוריתמים להצפנה סימטרית עושים שימוש במספרים גדולים מאוד ופעולות כפל רבות, **לכן הם הרבה יותר איטיים** מאשר ההצפנה סימטרית

חרונות ההצפנה האסימטרית

העובדת שיש לשמור את מפתח ההצפנה בסוד ואין לחשוף אותו בשום שלב לאף אחד מלבד המשתתפים היגיטיים, עובדה שיוצרת בעיה מהותית הנקראת בעית הפצת מפתחות.

הסבר כללי של הצפנה אסימטרית

ההצפנה האסימטרית היא שיטת הצפנה שבה מקבל מכין לעצמו שני מפתחות, אחד הנקרא מפתח פרטי שנשמר בסוד ומשמש לפענוח ואילו השני נקרא מפתח ציבורי המפורסם לכל דורש ומועד להצפנה בלבד.

אף על פי שהקיים קשר הדוק בין שני המפתחות למתבונן מהצד אמרו להיות קשה מאוד לנחש מהו מפתח אחד בהינתן השני ולהפוך.

בדרך זו כל אחד יכול להצפין מידע עם המפתח הציבורי של מקבל כי הוא ידוע לכל, אך רק מקבל לבדו מסוגל לפענוח עם מפתח הפענוח המתאים שנשמר בסוד.

השיטה האסימטרית מתגלה כשיתמושת במיוחד כאשר צד אחד רוצה להעביר לצד השני מפתח שייחה לצורך הצפנה/פענוח של מידע שהוא מעוניין לחלוק עמו.

הוא עושה זאת על ידי הצפנה באמצעות המפתח הציבורי של מקבל והקבל מצד אחד יכול לפענוח את המסר ולקבל את מפתח השיחה מבלי שיצטרך לשתף במידע כלשהו מראש עם השולח.

עם מפתח זה שני הצדדים יכולים לבצע התקשרות מוצפנת בטחנה, באמצעות צופן סימטרי מהיר

RSA

- שיטת ההצפנה האסימטרית הידועה והנפוצה ביותר
- מتبוססת על חישוב חזקה מודולו מספר גדול – הعلاה (מודולרית) בחזקת ח קלה לחישוב עשויה שימוש במספרים גדולים, החל במקור עם 1024 ביט
- חזקתו של האלגוריתם מבוסס על הקושי לפרק לגורמים מספרים גדולים

מה זה RSA?

- מפתח ההצפנה אינם סודי והוא שונה ממפתח הפענוח שנשמר בסוד, על כן היא נקראת אסימטרית.
- האסימטריה ב-RSA נובעת מהקשיש המעש שבסירוק לגורמים של מספר פריק שהוא כפולת של ראשוניים גדולים, שהוא עביה פתוחה בתורת המספרים.
- ב-RSA-השלוח משתמש במפתח ההצפנה הציבורי של הנמען כדי להצפין עבورو מסר כך שרק הנמען מסוגל לפענוו באמצעות המפתח הפרטי המתאים שברשותו.

יצירת מפתחות RSA

המשתמש מייצר זוג מפתחות, ציבורי וprtvi, באופן הבא:

1. בוחר שני מספרים ראשוניים גדולים באקראי, p ו- q , למשל **בעזרה אלגוריתם Miller-Rabin**.
2. אסור שהוא קל לנחש את הראשוניים p ו- q מtower ח, צריכים להיות גדולים אך לא קרובים מידי.
3. מחשב את בסיס המודולו, $q^e \equiv 1 \pmod{\Phi(n)}$
4. בוחר **אינדקס ציבורי אקראי** e , עבורו מתקיים: $(e, \Phi(n)) = 1$ וגם $e < \Phi(n)$
5. מוצא את **האינדקס הprtvi** d , ע"י פתרת המשוואה: $d \cdot e \equiv 1 \pmod{\Phi(n)}$
6. מפרסם את **המפתח הציבורי** שלו: $PU = \{n, e\}$
7. שומר בסוד את **המפתח prtvi** שלו: $PR = \{n, d\}$

↙ הטקסט הגלי הוא מספר בייצוג בינארי M , המקיים: $n < M < 2^n$

אם הטקסט M אינו מקיים את התנאי, נחלק אותו לבלוקים M_1, M_2, \dots, M_k כך שכל בלוק M_i מקיים את התנאי

For example you need to get d in the next:

$$3 \cdot d \equiv 1 \pmod{9167368}$$

הצפנה ב RSA

this is equally:

$$3 \cdot d = 1 + k \cdot 9167368, \text{ where } k = 1, 2, 3, \dots$$

rewrite it:

$$d = (1 + k \cdot 9167368)/3$$

Your d must be the integer with the lowest k .

Let's write the formula:

$$d = (1 + k \cdot f)/e$$

- השולח מшиיג את המפתח הציבורי של הנמען $\{n, e\} = PU$
- מחשב ושולח את n

- הנמען צריך להשתמש במפתח prtvi שלו, $\{n, d\} = PR$
- וליחסב את $n \cdot d \pmod{C} = C^d$

פענוח ב RSA

דוגמא: הצפנה ופענוח
נשתמש במפתחות מהדוגמה הקודמת:

נתונה הودעה: $88 = M$ (שים לב: $187 < 88$)

הצפנה:

$$C = 88^e \pmod{187} = 11$$

פענוח:

$$M = 11^d \pmod{187} = 88$$

את חישובי החזקיות נבצע באופן ייעיל

על ידי האלגוריתם שמוופיע בהרצאה הקודמת

או בשיטה דומות בעלות זמן ריצה דומה

1. נבחר ראשוניים: $p=17$ & $q=11$
- 2.-Calculates: $n = p \cdot q = 17 \cdot 11 = 187$
3. Calculating: $\Phi(n) = (p-1)(q-1) = 160$
4. נבחר e עבורה $\gcd(e, \Phi(n)) = 1$, למשל $e=7$
5. נקבע את d : $d \cdot e \equiv 1 \pmod{\Phi(n)}$, $d < 160$.
23x7=161=1x160+1 ש: $d=23$
6. ערכו הוא $C=23$ ציוויל ש: $C^d \pmod{n}$
7. שאלה: איך נמצא את d מבל' לנחש? $d = e^{-1} \pmod{\Phi(n)}$
8. מפרסם מפתח פומבי: $PU = \{7, 187\}$
9. נשמר בסוד על המפתח prtvi: $PR = \{23, 187\}$

רמת אבטחה ב RSA

- קיימות גישות שונות לתקיפת RSA:

- חיפוש המפתח הפרטי בעזרת כוח גס, אין ישים עבור מספרים גדולים
- **תקיפות על בסיס אנגליזה מתמטית:**
 - פירוק לגורמים של המספר n לשתי הגישות רמת קושי דומה
 - חישוב ישר של (ch)
- **חישוב קוונטי** – אין ישים
 - עושה שימוש בubit, בניגוד לביט רגיל שערך 0 או 1, ערך של qubit הוא שילוב של השניים
 - עוזר לחישוב אלגוריתמים מהירים יותר
- **תקיפות תזמן** – תקיפה מסווג כתקיפה על פער זمان ביחסים שונים, למשל כפל במספר קטן, אם
 - על בסיס הזמן ריצה, המתפרק מנסה להטיק תוכנות של הקלט
 - **אמצעי הגנה לתקיפה:** חישובים בזמן קבוע, והוספת השהיות אקראיות
- **מתකפת Chosen Ciphertext** אחראית על תוכנות החישוב של RSA
 - התוקף בוחר מסר מוצפן ונוטן למחשב המותקף לפענה אותו
 - המסר המוצפן נבחר על בסיס אנגליזה מתמטית על מנת לתקוף תוכנות של פונקציית הצפנה אמצעי הגנה אפשרי: הוספה "מייסור" אקרייא לתהليل הפענוח, והוספה ביתית ביקורת למסר.
 - עוזר בכך שהתקוף לא יודע לייצר מסר מוצפן המתפענה באופן חוקי, וכך מתגלה.
- מקובל להשתמש במפתח בן 2048 ביט (= 112 ביט סימטרי)
 - ק-פ צרכים להיות באורך דומה

מציאת ערכו של $a^{-1} \pmod{n}$ בעזרת מתකפת תזמן

- ▶ נבחר מסר מוצפן Z עבורו $Z < n^3$
 - ▶ אם $a=0 \pmod{n}$, אין פעולות מודולו באיטרציה השנייה
 - ▶ אם $a=1 \pmod{n}$, גם כן אין פעולות מודולו באיטרציה השנייה
 - ▶ נבחר מסר מוצפן Z עבורו $Z < n < n^2$
 - ▶ אם $a=0 \pmod{n}$, אין פעולות מודולו באיטרציה השנייה
 - ▶ אם $a=1 \pmod{n}$, יש פעולה מודולו אחת
 - ▶ הנסה לפענה את שניהם, וنمודד את הזמן ש עבר עד לקבלת הפענוח
 - ▶ אם $a=0 \pmod{n}$, זמני הפענוח יהיו דומים
 - ▶ אם $a=1 \pmod{n}$, הפענוח של Z יהיה איטי יותר
 - ▶ היות שהאטרציות הבאות בחישוב מושיפות "רעש" למדידה, נחזיר על הניסוי עם מספר קליטים ונשווה את המוצע
 - ▶ לאחר שגילינו את $a^{-1} \pmod{n}$, ניתן לגלו את הביטים הבאים בטכניות דומות
 - ▶ ביט אחד בכל פעם
 - ▶ במתකפה שערוכה שעתיים ננד שרת מרוחק שעשה שימוש ב-SSL, Open SSL, שוחרר מפתח בן 1024 ביט

תקיפות נוספות

- **מתකפת אקוסטית**
 - מנתח רעים המופקים מהמעבד בזמן הפענוח כדי לגלות את המפתח הפרטי
- **PITA – Portable Instrument for Trace Acquisition**
 - חשיפת מפתח RSA בעזרת עיבוד הקירינה האלקטרומגנטית שנפלטה בזמן הפענוח

דוגמה למתකפת תזמן על RSA

תזכורות: חישוב חזקות מהיר של $a^b \pmod{n}$

היצוג הבינארי של a הוא $b_k b_{k-1} \dots b_1 b_0$

```
f = 1
for i = k down to 0
    f = (f * f) mod n
    if b_i == 1 then
        f = (f * a) mod n
return f
```

האטרציה הראשונה היא טרייאלית

$b_k = 1, f = a$

האטרציה השנייה מבצעת אחד מהחישובים הבאים:

$a^2 \pmod{n}$, if $b_{k-1} = 0$

$(a^2 \pmod{n}) * a \pmod{n}$, if $b_{k-1} = 1$

הפעולה היקרה ביותר (מבחינת זמן חישוב) היא $a \pmod{n}$

הגנה מפני מתකפת תזמן באמצעות מייסון

הרעין: החזקה תחשוב על מספר אקרייא, השונה מהקלט המקורי למספר האكريיא אין בהכרח את תוכנות הקלט שהתקוף בחר

עדין צריך לחשב את התוצאה הרצויה

זמן החישוב יהיה רק מעט יותר ארוך

תיאור ההגנה

התקוף מבקש לפענה טקסט נבחר C

המחשב המתגונן מבצע את הפעולות הבאות:

בוחר, מבعد מועד, מספר אكريיא r , ומחשב את ההפכי $r^{-1} \pmod{n}$

מציף, מבعد מועד, את $r^e \pmod{n} = E = r^e \pmod{n}$

מחשב (זמן אמת): $X = r^{eC} \pmod{n}$

מפענח את X: $a = X^d \pmod{n} = r^{edC} \pmod{n} = r^{Cd} \pmod{n}$

כפול $b^{-1} \cdot r^d$ כדי לקבל את התוצאה אותה יש להחזיר: M

המספר C מופיע נבל שיחסנו במפורש את $a \pmod{n}$

כל החישובים מתבצעים על מספרים אكريיאים, שאינם ידועים לתקוף

הפצת מפתחות

הגדרת הבעה

- שיטות הצפנה סימטריות דורשות משני הצדדים להסכים על **מפתח סודי משותף**
- **הבעיה:** כיצד להעביר את המפתח באופן מאובטח באופן מואבטח
- לעיתים קרובות כשלים במערכות מאובטחות נגרמים בשל פרצה במנגנון הפצת המפתחות

פתרונות להפצת מפתחות

- A יכול לבחור מפתח ולהעביר אותו באופן פיזי ל-B
- צד שלישי יכול לבחור מפתח ולהעביר אותו ל-A ול-B
- **KDC – Key Distribution Center** – מרכז להפצת מפתחות
- A יכול להשתמש במקודם כדי להציג מחדש ושלוחו אותו ל-B.
- אם יש ערך מאובטח בין צד שלישי C לבין A ו-B, אז C יכול לתמוך ולהעביר מפתח ש-A בחרה ל-B

היררכיית המפתחות

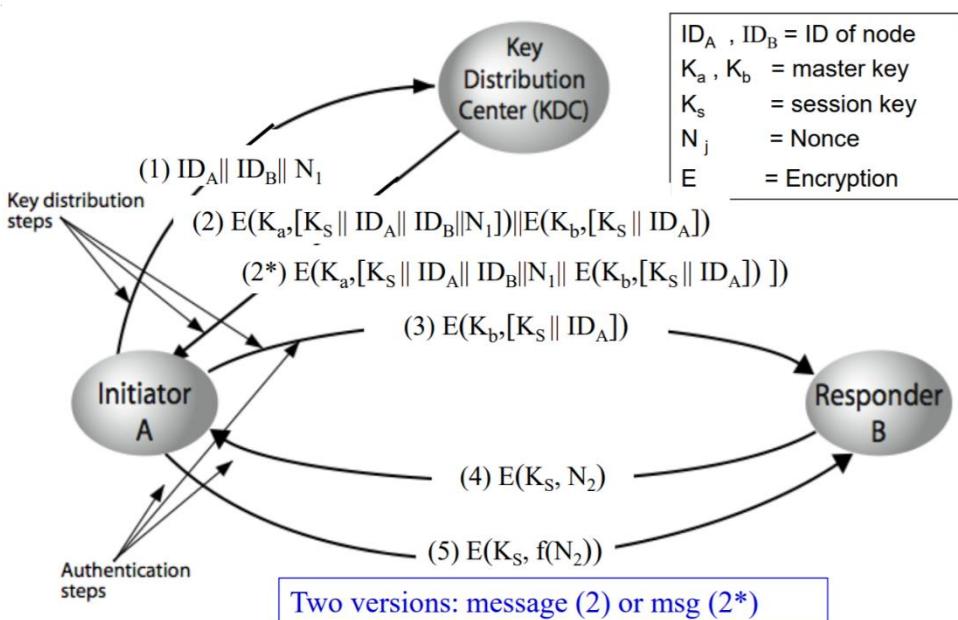
- ישנו מספר סוגים של מפתחות
- **מפתח שיחה – session key**
 - זהו מפתח לפרק זמן קצר – נעשה בו שימוש לשיחת אחת, ולאחר מכן הוא מושמד
 - משמש להצנת המידע שעבר בין המשתמשים
 - מיוצר ומופץ ע"י KDC
- **מפתח מאסטר – master key**
 - משמש להצנת מפתחות שיחה
 - לכל משתמש יש מפתח מאסטר שידוע גם ל-KDC, והם מוחלפים לעתים רוחקות יותר

← ניתן להשתמש במפתח המאסטר לטוח אורך על מנת לשמור יותר מידע (ביטים)

הסבר לתרשים

- A רוצה מפתח חדש לשיחה עם B
מקרה לתרשים:
KDC הוא מפתח המאסטר של A, הידוע רק ל-A ו-L-KDC-KDC-R
KDC הוא מפתח המאסטר של B, הידוע רק ל-B ו-L-KDC-KDC-R
ID_A, ID_B הם מזהים של A ו-B, בהתאם, ידועים לכל K
E(K, M) הוא המסר M מוצפן באמצעות המפתח K
שלבים עיקריים:
(1) A מבקש מפתח שיחה K_S מה-KDC
(2) KDC שולח את מפתח השיחה ל-A
(3) A מעביר את מפתח השיחה ל-B
את הדיוון בשלבים (4) ו-(5) נדחה להמשך

הפצת מפתח סימטרי בעזרת KDC



הסבר לאיור

- 1.** **בשלב הראשון** שלוח את המזהה שלו , את המזהה של B משורשר, $N1 = \text{nonse} 1$ למרכז ההפצת המפתחות, כדי לשמור הפצת המפתחות בין על מה מדובר, A ו- B
- 2.** **בשלב השני** מועבר מידע מוצפן (session key, Id's, Encrypted session key with B), יש פונקציה שעושה הצפנה והتوزאה היא סדרה של ביטים שמציגים את הדבר המוצפן, משורשר עם עוד שהוא מוצפן, בוצע ע"י מרכז להפצת המפתחות.
- 3.** מבצע את ההצפנה הראשונה בעזרת המפתח של A, K_A , ואת ההצפנה השנייה הוא עושה בעזרת המפתח של B, K_B .
- 4.** בעקבו המרכז להפצת המפתחות המציא מפתח session לא ו-B דרך הזאות של A ו-B שקיבל בשלב 1, וכל זה הגיע לא.
- 5.** A יודע שהודעה 2 חוברה ע"י KDC כי הוא מקבל את key session ואת הזאות S ו-B שהוא שלח בהודעה. מפתח המאסטר ידוע רק לא ולKCDC لكن הוא יודע שהה ChiB לבוא ממנו.
- 6.** **בשלב השלישי*** כל המסרורים נשלחו ביחד, הצפנה בתוך הצפנה. "מעטפה בתוך מעטפה" שרק B יודע לפתוח אותה. B יודע לפתוח רק את המעטפה הגדולה ואת הקטנה הוא פשוט שולח חזרה (לא יודע לפתוח אותה). השיטה הזאת יותר טובה כי אם מישהו היה מקליט את השיחה, בגישה הראשונה הוא יוכל ללקח את כל ההודעה כמו שהוא, אבל בגישה השנייה הוא לא יוכל כי B לא יוכל לפתוח את המעטפה הפנימית.
- 7.** התוצאה הייתה שהיא בא A מראה שהוא הוזן באופן חד ערכי.
- 8.** **בשלב השלישי** A מקבל 2 תוצאות של הצפנה, הוא לוקח את התוצאה השנייה כמו שהוא עם המאסטר של B ושולח אותה ל-B responder. הבעייה היא ש-A לא יכול לפענן את זה, כי זה מוצפן עם המפתח מאסטר של B. A שלוח את זה כמו שהוא ל-B מבלי לפתח את הצפנה הזאת. B כן יכול לפתוח את הצפנה.
- 9.** B יודע שהודעה 3 חוברה ע"י KDC כי הוא מפענן עם מפתח שידוע רק לו ולKCDC, והוא את הזאות של A שהוא מכיר אותה (יותר חשוב מהמפתח שלו ושל KCDC).
- 10.** B יכול להיות בטוח ש-A שלח את הודעה 3 כי יש שם את ID של A. זה כן יכול להיות מתחזה.
- 11.** **בשלב הרביעי** B מוצפן Nonce2 עם מפתח החסия session ושולח אותה לא.
- 12.** **בשלב החמישי** A שלוח פונקציה מוצפנת עם המפתח session פונקציה עם N.

***למה נשלחו פונקציות בין הצדדים?** כי מישהו יכול להאזין לשיחה ולפענן את הצופן המוצפן

Nonce – כל פעם שימושים בהזזה מקבלים מספר אקראי אחר ונעשה בו שימוש חד פעמי. זהו מנגן שעובד לנו למנוע replay attack.
 – "ניגון חדש" של מפתח שהתווך נמצא בעבר.
Replay Attack – Man in the middle – מאזין לשיחה.

סוגיות בהפצת מפתחות

- במערכות גדולות יש מספר KDC עם יחס היררכיה ביןם
- השימוש במפתח לשיחה צריך להיות בזמן מוגבל
- המשתמשים חייבים לתת אמון ב-KDC
- הגנה על מפתחות המאסטר מבני גניבה או הדלפה היא סוגיה חשובה באבטחת המערכת

הפצת מפתחות ושימוש במפתחות ציבוריים

- התלות של המשתמשים ב KDC היא רבה, הוא בוחר עברו המשתמשים את מפתח ההצפנה ונדרש לתחזק מפתח סימטרי הידוע רק למשתמש הקצה ול KDC
- ננסה לצמצם תלות זו בצד שלישי ע"י שימוש במפתחות ציבוריים

שימוש במפתח ציבורי להפצת מפתח סימטרי

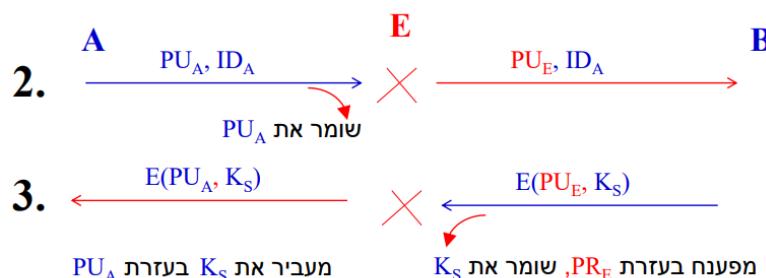
- אלגוריתמי הצפנה אסימטריים הם איטיים, לכן לא מקובל להשתמש בהם להצנת מידע רב, אבל כן נצפים בעזרת אלגוריתמי הצפנה אסימטרית מפתחות על עיקר המידע נגן באמצעות הצפנה סימטרית נשתמש במפתח סימטרי לכל שיחה (session key) נשתמש במפתח ציבורי בצד להעברת המפתח הסימטרי

הפצת מפתח סימטרי – ניסיון ראשון

ננסה לוותר על השימוש ב KDC

1. משתמש A מייצר זוג מפתחות חד פ уни (PU_A, PR_A)
2. A שולח ל B את PU_A ואת המזהה שלו ID_A
3. B בוחר מפתח שיחה K_S ושולח ל-A את המפתח של השיחה מוצפן עם המפתח הציבורי באמצעות המפתח הציבורי של A, K_S(PU_A, K_S)
4. רק A יוכל לפענח את ההודעה, שכן K_S ידוע רק ל-A ול-B

מתתקפה Man in the Middle על הפרוטוקול



- עכשו, ה頓וף E יכול:
- I. **לקרוא** את כל ההודעות בין A ל-B
 - II. **לשנות** חלק מההודעות

מאפייני המתתקפה:

- מתתקפה אקטיבית – אפילו אם מטרתה רק להאזין לשיחה דורשת התחזות בו זמןית לשני הצדדים
 - הרצת פרוטוקול כלשהו מול שני הצדדים
- המתתקפה אפשרית בגין האNONymity ברשות – העובדה שימושו סיפק A או אינה אומרת שהוא משתמש A על מנת למנוע אותה, נדרוש מ A לספק הודעה ל'זהות שלו
 - לבצע פעולה שרק A יכול לבצע
- מקור ההוכחה ידרוש אמון בגוף שהנפיק אותה – ויכולת להבדיל בין הוכחות שמקורן בגוף האמין לדויופים

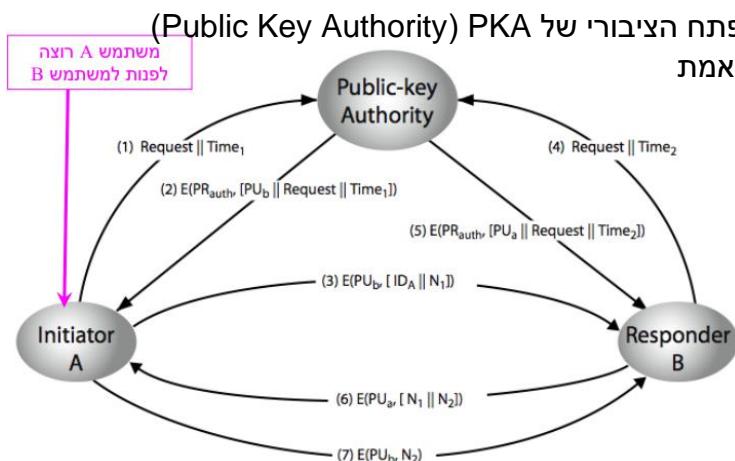
הפצת מפתחות ציבוריים

נשקל את החלופות הבאות:

- **הצירה פומבית**
 - משתמשים יכולים להפיץ את המפתחות הפומביים שלהם ע"י פרסום ברבים, למשל בדוא"ל
 - הקשי העיקרי הוא הקלות לגניבת זהות, כל אחד יכול לטעון שהמפתח של מישחו אחר שיר לו עד שהדבר יתגלה, פרסום המפתח מתחזה למישחו אחר
- **מאגר מפתחות ציבורי**
 - ניתן לשפר את רמת האבטחה אם נדרש שהמפתח ירשם במאגר ציבורי כלשהו
 - המאגר צריך להיות אמין, עם התכונות הבאות:
 - מכיל רשומות של {שם, מפתח ציבורי}
 - משתמשים יכולים להתחבר באופן מאובטח כדי להחליף את המפתחות שלהם במאגר
 - מעט לעת המאגר מפורסם ברבים
 - המאגר זמין באופן מקוון
 - המאגר אינו מכיל מפתחות פרטיים
 - הפתרון אינו מושלם כי המאגר רגיש לחבלות וזריפים, ניתן להתחזות לשרת המכיל את המאגר ולפרסם נתונים מסוימים

- **רשות למפתחות ציבוריים (public key authority)**

- מבוסס על תוכנות המאגר
- תהליכי הפצת המפתחות מכיל אבטחה נוספת
- מניח שככל המשתמשים מכירים את המפתח הציבורי של PKA
- חסרון: דרוש פניה למאגר בזמןאמת



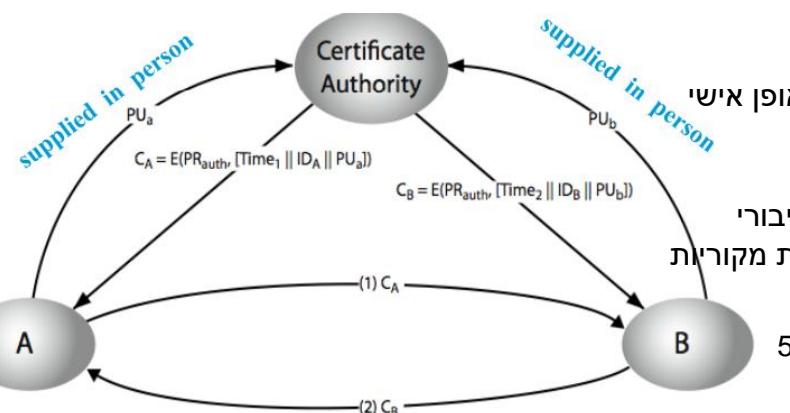
הסכם המקשר מפתחות ציבוריים עם הגוף.

למה שלוחים את הזמן? כדי למנוע replay –
A יודע שהודעה 2 חוברה ע"י הרשות –

הוא חתום
בפרוטוקול זה אין צורך בפונקציה

- **תעודות (סertyifikטים) למפתחות ציבוריים (public key certificates)**

- התעודה מאפשרת הפצת מפתח פומבי מוביל להפנות בזמןאמת לרשות המרכזית כמו ה-PKA
- התעודה מוכיחה קשר בין המפתח הציבורי של משתמש לבין הזהות שלו
- יחד עם מידע נוסף כמו שיטת הצפנה, תאריך תוקף המפתח ועוד
- תוכן התעודה חתום בעזרת המפתח הפומבי של רשות מוסמכת (CA- Certificate Authority)
- כל מי שיעדעת את המפתח הציבורי של ה-CA יוכל לוודא שהחתימה תקפה



המפתח הציבורי של ה-CA ידוע לכל

כל משתמש מעביר את המפתח הציבורי שלו ל-CA באופן אישי

או בתקשרות מאובטחת לאחר אימות הזהות שלו

תוכן התעודה ידועות, רק חתומות – לא מוצפנות

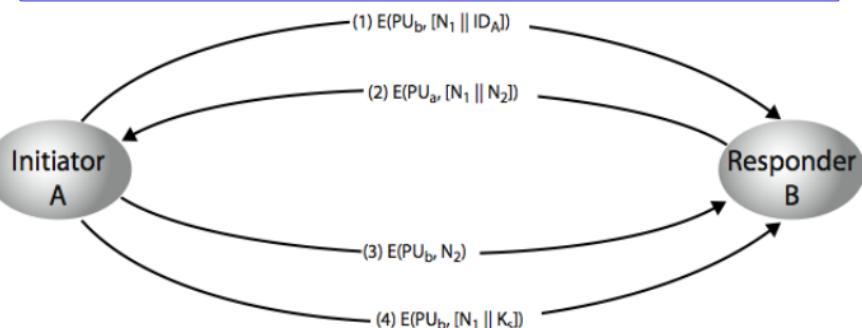
בתוך התעודה יש גם את הפרטים וגם את המפתח הציבורי

קשר ביןם לאותה תעודה, וכך הם יודעיםשאלות מקוריות

המטרה של הזמן היא למנוע replay

הפצת מפתח סימטרי בעזרת מפתח ציבורי ידוע

נקודות המוצאו: המפתחות הציבוריים הופצו באופן מאובטח כלשהו



1. בשלב 1 נשלח משוחה מוצפן עם המפתח הציבורי של B ו-ID_A
2. בשלב 2 שלוחה חוזרת את E(PB) שלו מוצפן
3. בשלב 3 A שלוח ל-B את המפתח הציבורי של B מוצפן עם N2
4. שלוח חוזרת את המפתח session key N2+N1 מייצרים את האישור שלו, כמו קודם

שיטת Diffie-Helman Merkle

שיטת הצפנה שהכרנו עד כה, ובכללם שיטת One Time Pad חייבו העברת פיזית של מפתחות. למעשה, לפני שאليس שולחת הודעה מוצפנת לבוב, עליה לשולח לבוב את המפתח הסודי.

במילים אחרות, לפני שני אנשים יכולים להחליף ביניהם סוד (ההודעה מוצפנת), הם חיבים כבר לחלוק סוד (המפתח). פיתחו דיפי וולמן שיטה המתגברת על בעיה זו של הפצת המפתחות.

כעת, **אליס יכולה** לשולח הודעה מוצפנת לבוב בלבד להביר לו מפתח וודין לבוב ורק לבוב יוכל לעננה את ההודעה.

השיטה מבוססת על אלגוריתם לצירת מפתח שהיה מוכר רק לשולח ההודעה ולמקבל ההודעה מבלי שייפגשו זה את זה, וב毫无疑ים שיעזרו בשילוח שיביר ביןם את המפתחות (כמו KDC), האלגוריתם מבוסס על פונקציות חד סטריות.

השיטה מבוססת על החלפת מפתח סימטרי בין צד'A' לצד'B' – הסכמה על מפתח מבלי לחושף אותו ומוביל להצפן אותו ע"י מפתח בשימוש רב פעמי. – יש לציין כי החלפת המפתח תגן רק על תקשורת עתידית. כמובן – בחירת מספר מחושב המשותף לשנייהם, שבאמצעותו יוחשב המפתח.

אופן השיטה

想起ת לשורשים פרימיטיביים:
מספר שאם נסתכל על כל חזקוטי,
מייצר את כל המספרים בZ פרט ל-0

a ו-k מוסכמים מראש
וניתנים לפרטום



1. נבחר k ראשוני מספיק גדול

2. נבחר p שורש פרימיטיבי מודולו p

3. A בוחרת מפתח פרטוי סודי k_A

4. A מחשבת מפתח פרטי ציבורי k mod p $y_A = g^{x_A} \mod p$

5. בائفון דומה, B בוחר x_B ומחשב בעזרתו k mod p $y_B = g^{x_B} \mod p$ ושולח אותו לא- A

6. משתמשים A ו-B מחשבים מפתח שיחה חד פעמי K_{AB}

$K_{AB} = g^{x_A \cdot x_B} \mod p$

$= y_A^{x_B} \mod p$

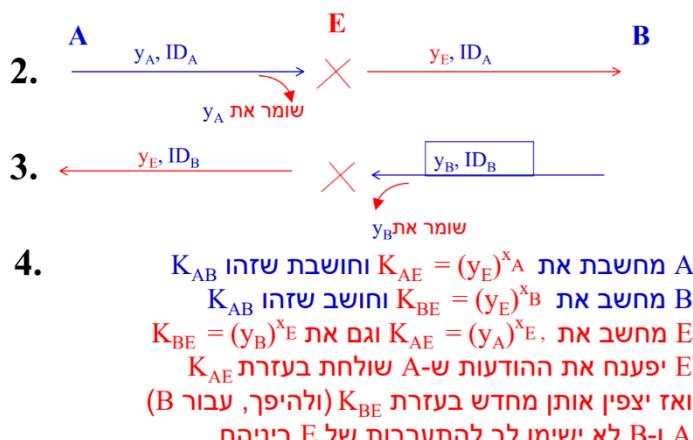
$= y_B^{x_A} \mod p$

○ על מנת להחליף את מפתח השיחה הסימטרי,

○ יש צורך לחשב מפתחות ציבוריים חדשים

○ תוקף שרוצה לגנות את המפתח צריך למצאו את x_A (או את x_B), ע"י חישוב לוגריתם דיסקרטי : $x_A = \log_p y_A$

מתפקיד Man in the Middle על DHM



52

דוגמה מספקית

אליס ובוב רוצים לבחור מפתח סימטרי
הם בוחרים $k=353$ ו- $p=3$

הם בוחרים מפתחות פרטיים סודיים:
 $x_A=97$, בוב בוחר $x_B=233$

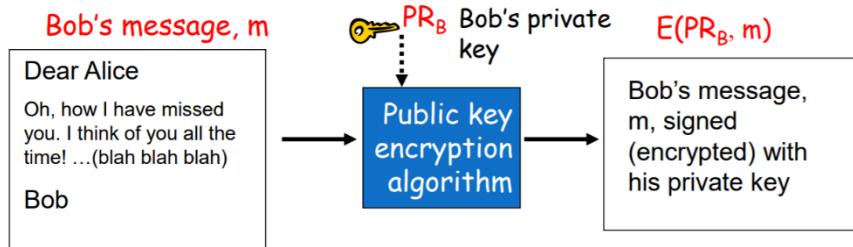
הם מחשבים מפתחות ציבוריים:
 $y_A = 3^{97} \mod 353 = 40$ (אליס)
 $y_B = 3^{233} \mod 353 = 248$ (בוב)

ומחשבים את מפתח השיחה הסימטרי:
 $K_{AB} = y_B^{x_A} \mod 353 = 248^{97} \mod 353 = 160$ (אליס)
 $K_{AB} = y_A^{x_B} \mod 353 = 40^{233} \mod 353 = 160$ (בוב)

חתימה דיגיטלית

חתימה דיגיטלית פשוטה על הודעה m

- בוב חותם על m בכך שיצפין אותה עם המפתח הפרטי שלו PR_B , ובכך י יצא הודעה חתומה $E(PR_B, m)$



- חתימה אינה מספקת חיסיון (Confidentiality)
 - כל אחד יכול לקרוא את תוכן הודעה (וأتזקף החתימה)
 - אם נרצה חיסיון, יש להשתמש בהצפנה רגילה בנוסף.
- אליס מתקפת (verifies) את ההודעה m חתומה ע"י בוב, בכך שתפעיל את המפתח הציבורי PU_B על $E(PU_B, m)$ וAz תבדוק שהתקבל הודעה הגיונית ← חותמים עם המפתח הציבורי אז יכולים כלו' לדעת אותו – لكن חותמים עם המפתח הציבורי, אם הינו חותמים עם המפתח הציבורי אז יכולים כלו' לדעת אותו – לכן חותמים עם המפתח הציבורי, אם כן, מי שחתם על הודעה יודע את המפתח הפרטי של בוב
- תזקוף החתימה מוכיח את הדברים הבאים:
 - בוב חתום על m (אימות זהות – Authentication)
 - בוב חתום על m ולא על ' m' (אימות תוכן – Message integrity)
 - התזקוף אינו מוכיח שבוב שלח את m עכשווי. יתכן שצד שלישי שולח הודעה ישנה (replay)
 - עקרון אי הכחשה (Non Repudiation):
 - אליס יכולה לחתם את m ואת החתימה $E(PR_B, m)$ ולהוכיח לכל אחד שבוב חתום עליה
 - גם בוב יכול להוכיח זאת, אם מישחו יפרק בכך

- חתימה נעשית באמצעות הצפנה אסימטרית, כך שיש לנו זוג מפתחות – מפתח פרטי וציבורי
- החתימה נעשית ע"י המפתח פרטי
- מספקת אימות זהות של מי שלח והמקור של המסר הנחתם
- מספק את שלמות הודעה / אימות תוכן – אם החתימה שקיבלונו מתאימה לחתימה שהיחסנו אז ניתן להיות בטוחים בrama גבוהה שזאת הודעה המקורית ללא שינוי
- מניעת הכחשה

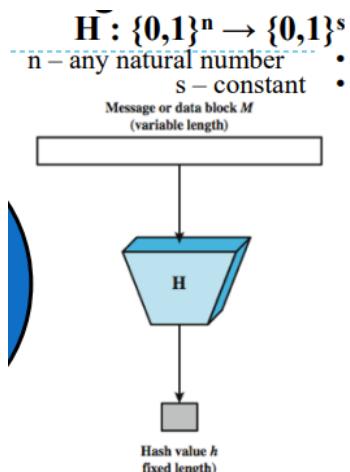


Digital Signature using RSA

$$\text{Sig}(M) = S(M, (d, n)) = M^d \pmod{n}$$

$$V(M, \text{Sig}(M), (e, n)) = \text{true} \quad \text{iff} \quad M = [\text{Sig}(M)]^e \pmod{n}$$

Hash Function



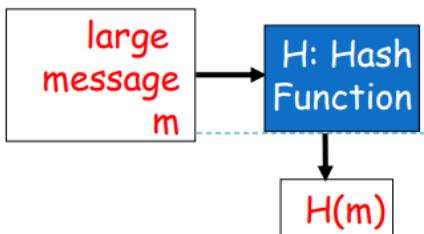
- פונקציית HASH ממפה קלט בגודל כלשהו לקלט שהוא אורך מסוים ←
- קלט באורך כלשהו, מכנים אותו לפונקציית HASH ומקבלים אורך S מדויק
- כמובן, כל פונקציית HASH מוציאה פלט באורך קבוע
- לערכים בטעות קוראים **message digest**, לקלט קוראים **message**
- פונקציית HASH היא לא חד"ע - מוגדים התאים בשובך קבוע
- יוניים – input, שובר – output
- תמיד תהיה התנגדות – לפחות 2 יוניים יצטרכו לחלוק אותו תא
- ▶ $H(x)=H(x')$ for some inputs $x \neq x'$
- ▶ Collisions must happen
 - ▶ Length of x unlimited, length of $H(x)$ is s -bits !
 -▶ Even if length of x is bounded, $|x| = b$, and $b \geq s$, there will be collisions..
 - ▶ “Pigeonhole principle”



Digest – תמצית ההודעה

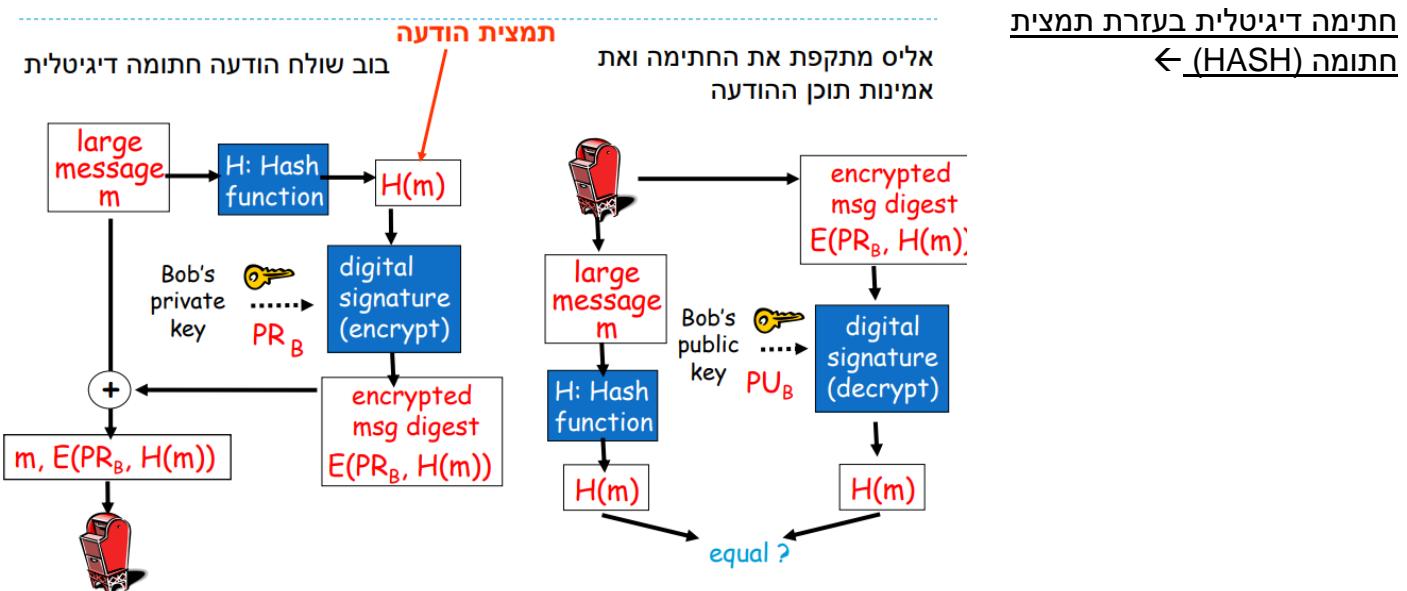
פעולות הרצפנה האסימטרית אינה יכולה מבחינה חישובית להודעות ארוכות

- הרעיון: תמצית (טביעה אצבע) באופן רביע, קצר וקלה לחישוב
- **מבצע פונקציית גיבוב** (HASH Function) H על m ונקבל **תמצית ההודעה** ($H(m)$)
- חישוב H מהיר, גם להודעה ארוכה
- נשלח לצד שני את תמצית ההודעה – // נשלח לצד השני את ($PR_B, H(m)$) E
- נציג רק את ($H(m)$)



תכונות פונקציית גיבוב קרייפטוגרפית

- לא נדרש מפתח
- תמצית פלט קצר באורך קבוע, **כלומר לא חד"ע**
- בהינתן תמצית x , **קשה מאוד** למצוא הودעה m עבורה $x = H(m)$
- בהינתן הודעה m , **קשה מאוד** למצוא הודעה m' עבורה $x = H(m')$



פונקציית גיבוב לא קריפטוגרפית: TCP Checksum

לטבTCP יש תכונות של פונקציית גיבוב

- תמצית באורך קבוע (16 ביט של סיכון הקלט במנוחת קבועות)
- אינה הפיכה (לא חח"ע)
- מיועד לבדיקה אם אין שגיאות בדרך בחבילות של TCP

אבל בהיעדר הودעה בכלת ערך גיבוב כלשהו, קל למצוא הודעה אחרת בעלת ערך גיבוב זהה

| <u>message</u> | <u>ASCII format</u> | <u>message</u> | <u>ASCII format</u> |
|----------------|---------------------|----------------|---------------------|
| I O S 1 | 49 4F 55 31 | I O S 9 | 49 4F 55 39 |
| 0 0 . 9 | 30 30 2E 39 | 0 0 . 1 | 30 30 2E 31 |
| 9 B O B | 39 42 D2 42 | 9 B O B | 39 42 D2 42 |
| | <hr/> | | <hr/> |
| B2 C1 D2 AC | | B2 C1 D2 AC | |

הודעות שונות **checksum** **זהה**

אם קיבלנו **checksum** שונה אך מבנים שכנראה הייתהפה שגיאה,
אם שווים אך אחד לא נגע בהודעה

דוגמאות לפונקציות HASH קריפטוגרפית : SHA ,MD5 -128bit

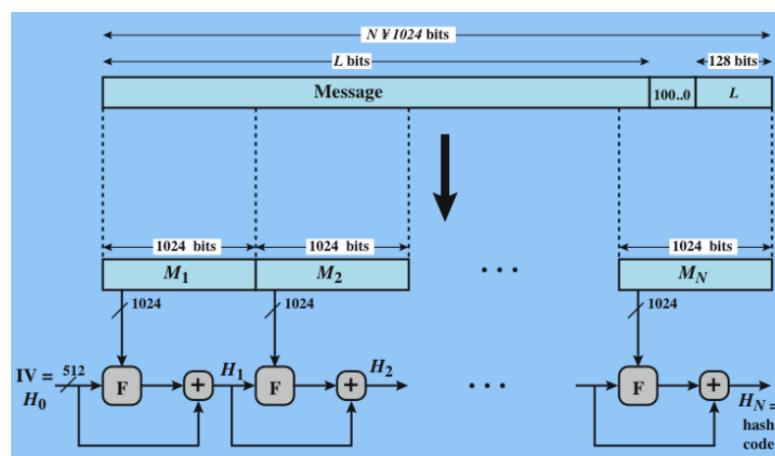
התקן נועד לשימוש כחלק ממנגנון אימות והבטחת שלמות מידע וחלוקת מתקן חתימה דיגיטלית **SHAttered**

- ב-2017 חוקים מגול הדגימו מתקפה מסווג Collision attack על **SHA-1**
- המתקפה חישבה את פונקציית **SHA-1** 2^{63} פעמים
- אורך digest message (תמצית ההודעה) הוא 160 ביט
- פורסמה ב-2001 **SHA-2**
- אורך digest message (תמצית ההודעה) הוא 512 ביט
- יש לו מאפיינים דומים **SHA-1**
- מתקפות ידועות על **SHA-1** אין שימוש ב-**SHA-2**
- פורסמה ב-2007 **SHA-3**
- מוכחת כסטנדרט

אם אין מספיק ביטים כדי להשלים ל-1024

- "rifod" של ההודעה המקורי ותוספת אורך בסוף
- חלוקה למקטעים של 1024 ביט
- מתחילהים ממערך שרירותי וקבוע של 512 ביט
- על כל מקטע מבוצעת סדרה של חישובים של 80
- שלבים, יחד עם הפלט שחושב על בלוקים קודמים
- התוצאה הסופית היא פלא פונקציית הגיבוב
- תבנית זו של rifod, חלוקה למקטעים וסדרת חישובים נקראת **Markle Damgard Construction**
- הוכחו שהתבנית שומרת על תכונות אבטחה של הפונקציה שמופעלת על כל מקטע בנפרד

מבנה פונקציית SHA-512 (HASH קריפטוגרפית)



- הסבר
- מתחילה עם 1024 ביטים
 - הקלט לכל סבב
 - 512 ביט מהසבב הקודם (H_{i-1})
 - 64 ביט מהמקטע (M_i)
 - קבוע של 64 ביט
 - בכל סבב מבצעים
 - סדרה של הזרות ופעולות בינהיות
 - את פלט הסבב ה-80 מחברים לקלט המקורי H_{i-1}
 - כדי לקבל את הפולט הסופי H_i
 - פעולות נורא מהירות

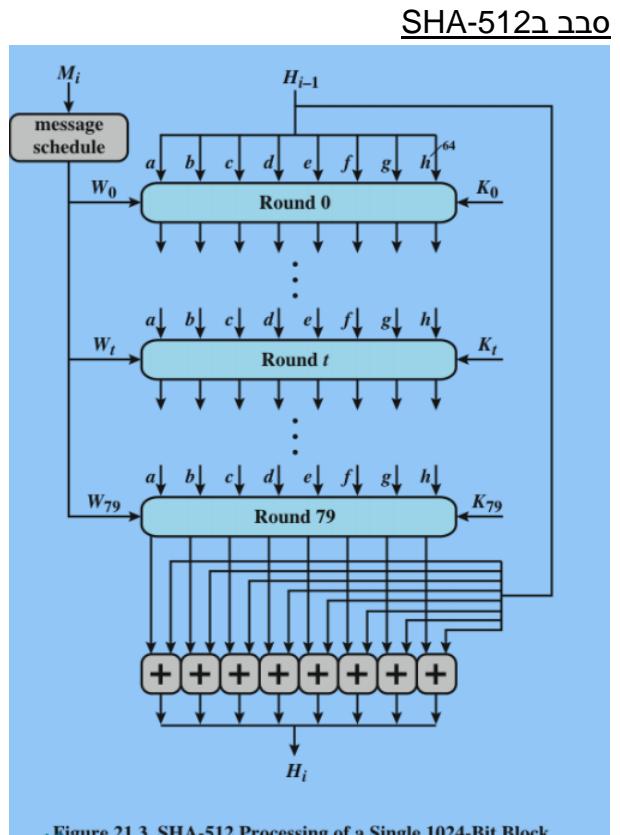


Figure 21.3 SHA-512 Processing of a Single 1024-Bit Block.

תקיפת פונקציות גיבוב

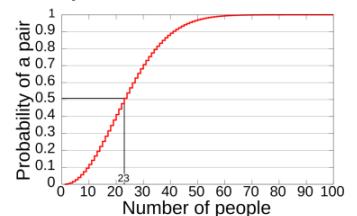
פרדוקס יום ההולדת

Q: What is the num. of people that should be in the room so that the probability that there is someone in the room which was born on the same day and month as me, is > 0.5?

A: 183

Q: What is the num. of people that should be in the room so that the probability that there are 2 people in the room which were born on the same day and month, is > 0.5?

A: 23



Intuitively:

In a group of 23 people there are $\frac{23 \times 22}{2} = 253$ couples

The probability of at least 1 couple to have the same birthday > 0.5

The accurate computation:

$$1 - 365/365 \times 364/365 \times 363/365 \times 362/365 \times \dots \times 343/365 \approx 0.507297$$

$$k = 2^{\frac{n+1}{2}} \sqrt{\ln \frac{1}{1-\gamma}}$$

γ ≡ prob for at least 1 collision

E.g., = 0.5

n ≡ # of output bits

E.g., # of bits of # of days in a year (365) = $\log_2 365 \approx 8.5$

k ≡ # of inputs

E.g., # of people in the room

$$k = 2^{\frac{8.5+1}{2}} \sqrt{\ln \frac{1}{1-0.5}} \approx 22.4 \approx 2^{4.52}$$

We need at least 23 students in the room so that the probability that there are 2 students in the room which were born on the same day and month, is > 0.5

ובאופן כללי,
אם ניציר באופן רנדומלי K הودעת,
ההסתברות להתנגשות תלויה ב n
Consider a hash function $h: \{0,1\}^* \rightarrow \{0,1\}^n$

על מנת להתנגד להתקפת יום ההולדת,
נצרך לבחור ח גודל מספיק כדי שיוכל להתמודד עם
k גדול

איזה פונקציית HASH נדרש לבחור לפרודוקס יום ההולדת?

Birthday paradox

Example:

► Probability for collision - $p \geq 0.5$

► with k as small as 23

$$(n = \log_2 365 \approx 8.5)$$

$$\downarrow k = 2^{\frac{n+1}{2}} \sqrt{\ln \frac{1}{1-\gamma}}$$



- To resist birthday attack,
we choose n to be sufficiently large so that
it will take an infeasibly large k to
Have a non-negligible probability of collision

עמידות בפני התנגשויות

- **Weak Collision Resistance** – עמידות חלשה בפני התנגשויות
 - בהינתן הודעה m, קשה מאוד למצאו הודעה m^* כך ש: $(m)H = (m^*)H$
 - לכל פונקציית גיבוב עם תמצית של B ביטים יש מתקפה בכוח גס שדורשת בתוחלת של 2^B חישובים
- **Strong Collision Resistance** – עמידות חזקה בפני התנגשויות
 - קשה מאוד למצאו שתי הודעות $m \neq m^*$ כך ש: $(m)H = (m^*)H$
 - לכל פונקציית גיבוב עם תמצית של B ביטים יש מתקפה גס הולמת שדורשת בתוחלת $2^{B/2}$ חישובים
- חולשות ידועות של פונקציות גיבוב יכולות לאפשר מתקפות חזקות יותר, הדורשות פחות חישובי תמצית

(MAC) Message Authentication Code – קוד אימות מסרים

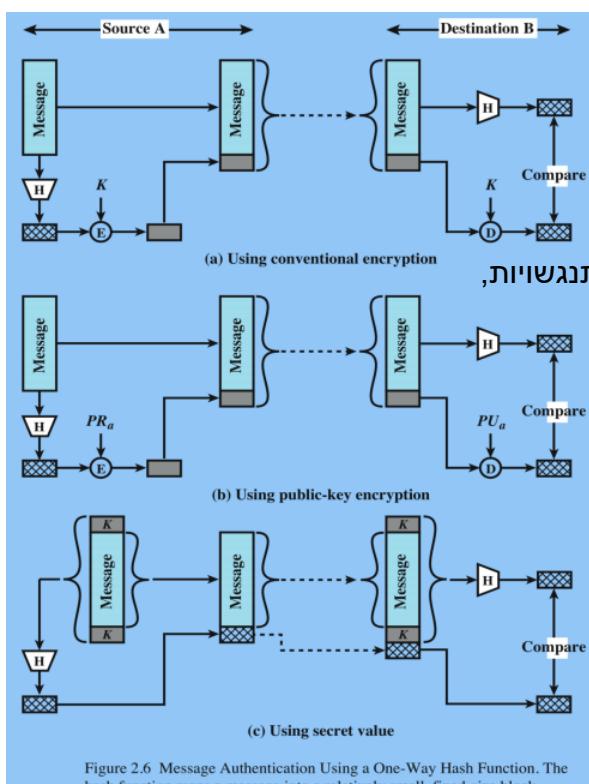


Figure 2.6 Message Authentication Using a One-Way Hash Function. The hash function maps a message into a relatively small, fixed-size block.

- גישה נוספת לאימות וחתיימה
 - דומה לפונקציית הגיבוב (HASH), אבל:
 - מקבל שני קלטים – ההודעה M ומפתח סימטרי K
 - לא מספק את עקרון אי ההכחשה (non repudiation)
 - כי המפתח סימטרי, אז יותר מממשחו אחד יודע עליו, ואם הוא דלף לצד שלישי לא יודעים ממי הוא דלף – בוב או אליס.
 - לעיתים מבוסס על פונקציית גיבוב בעלת עמידות חלשה להתנגשויות,
 - לעיתים יכול להפוך פונקציית גיבוב בעלת עמידות חלשה להתנגשויות, לבעלת עמידות חזקה להן
- **שיטת חתימה שונה בעזרת פונקציית גיבוב** ←
- **a. פונקציית גיבוב + הצפנה סימטרית (אותו מפתח)**
 - רק מקבל המסר יכול לתקוף אותו
 - אין עקרון אי ההכחשה
- **b. פונקציית גיבוב + הצפנה אסימטרית (מפתחות שונים)**
 - אימות מספרים בעזרת מפתח ציבורי
 - כל אחד יכול לוודא את תוכן המסר
- **c. פונקציית גיבוב על המסר המשורשר למפתח K**
 - תוכנות דומות ל-a, ובונוסף: אין צורך להצפין/לפענה,
 - ומוגן יותר מפני מתקפות התנגשויות וזריפת

מה היתרונות והחסרונות של חתימה על ערך תמצית:

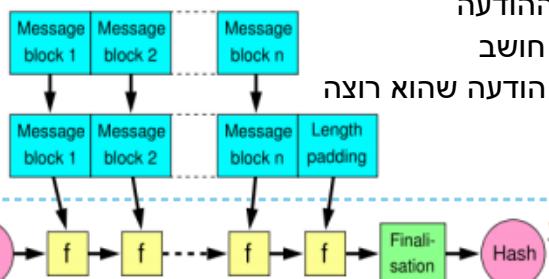
- יתרון – תהילך הרבה יותר מהיר של הatzפנה עקב פונקציית HASH
- חסרון – בכל מקום שיש בו מפתח סימטרי (a-c) לא קיים עקרון אי ההכחשה

שרשור מפתח סודי להודעה

- שיטות MAC שונות מבוססות על הסבר פונקציית גיבוב - כך שתתקבל קלט נוסף את המפתח K
- לפני זה הוצגה הפונקציה $C(M,K) = H(K||M||K)$
- שרשור המפתח רק לאחר מכן מקצועות ההודעה איננו בטוח עבור פונקציות גיבוב מסוימות
- פונקציות אלו מבוססות על תבנית דפי הלמן מרכז
 - חלוקת ההודעות למקטעים
 - כל מקטע מעובד יחד עם תוצאה מצטברת של כל המקטעים לפניו

חולשות בשרשור מפתח להודעה

- **שרשור המפתח לסוף ההודעה:** $C(M,K) = H(M||H(K||M||K))$
- נניח ש-M הוא מספר שלם עם מקטעים, ו-K נמצא לבדו במקטע האחרון (עם ריפוד)
- אם התמצית מחושבת במקטעים, אז עבור הודעות * M, M^* , $H(M) = H(M^*)$
- אז גם $(H(M^*)||K) = H(H(M)||K)$, כלומר קיבל message digest שווה – כי זה קורה בסוף
- **שרשור המפתח לתחלת ההודעה:** $C(M,K) = H(K||H(M||M||M))$ - **הופך את HASH לבעל חולשה**

פתרון ל-Length Extension Attack

פתרונות הפתרון HMAC: עד עכשוו ייצרו את HMAC כhash Function מcryptographic hash function ועכשוו אנחנו ניצר אותו מ

: HMAC יתרונות ל-

- יתרונות HMAC מתיבצע יותר מהר בתוכנה, מאשר אלגוריתם של הצפנה cryptographic hash function
- אין צורך לעשות היפוך של הצפנה לעוניה
- בזמןנו היו הגבלות על אורכי המפתח, וגם זה הוא תרם
- זהו פרוטוקול ההצפנה של IP, SSL, TLS ← HTTPS

מה זה HMAC?

- X-HMAC מצין את HMAC המסויים המשמשים בו בפרוטוקול HTTP, למשל HMAC-256 ← 256 ביט
- החזק הקרייפטוגרافي תלוי ב:
 - החזק הקרייפטוגרافي של פונקציית HASH
 - הגודל של הפלט של HASH
 - הגודל והaicות של המפתח
- המפתח הנutan מייצר 2 מפתחות – המפתח החיצוני והמפתח הפנימי
- על מנת לספור חסינות טוביה יותר נגד מתקפות – הוא משתמש בשני מעברים של HASH .

Hash Based Message Authentication Code

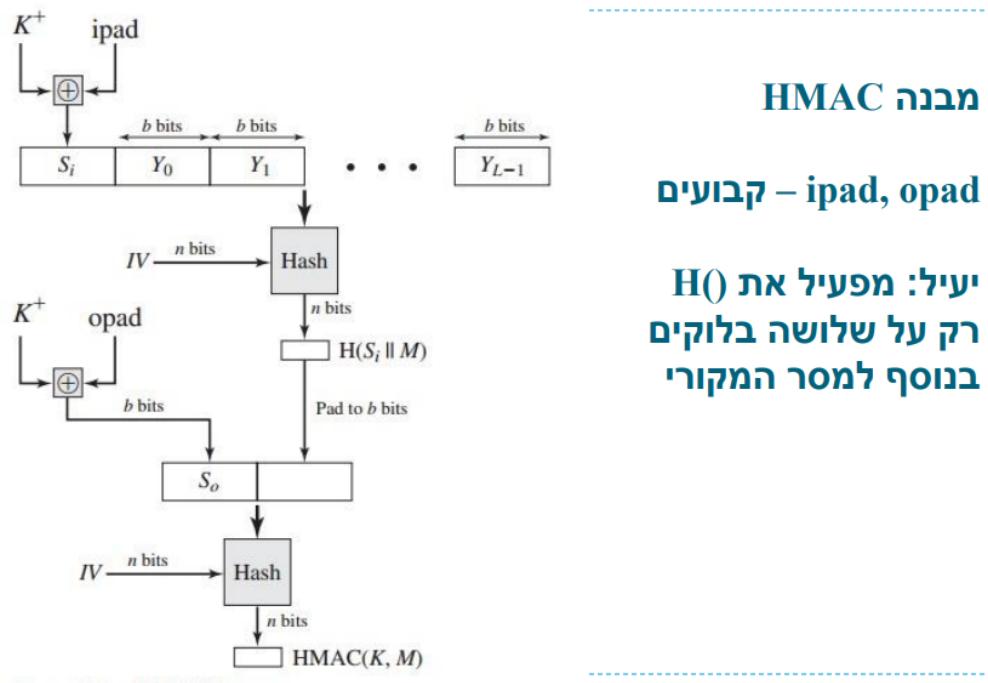


Figure 12.5 HMAC Structure

לפונקציית HASH יש 4 תכונות:

1. כמו כל פונקציית HASH ניתן פלט באופן קבוע
2. אם יש הودעה נתונה קשה למצאו הודעה אחרת שיש להם אותו message digest
3. קשה למצוא שתי הודעות כלשהן שיש להן אותו message digest
4. One way – כיוון אחד אפשרי, אבל שיש לנו את message digest קשה עד בלתי אפשרי לחזור חזרה, לחזור חזרה כלומר להציג את המידע המקורי, לפני ההפנה בפונקציית HASH

HASH – Rainbow Tables

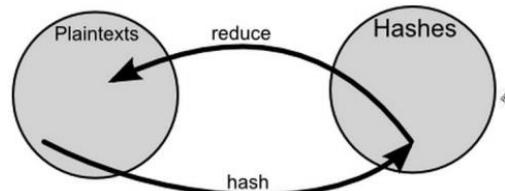
שיטות להשגת plaintext בשביל hash מסוים:

- לנסוט מתקפת כוח גס של כל האפשרויות \leftarrow לא יעיל, לוקח הזמן זמן
- שומרים את כל האפשרויות בטבלת HASH ענקית, וברגע שמקבלים message digest נחפש אותו בטבלה ונשיג אותו $(\leftarrow O)$ גורם לביעית זיכרון

טבלאות קשות צרכות להפוך את המידע הקRIPTוגרافي מפונקציות HASH בהתחשבות בזמן ובמקום (זיכרון)

Chains – פשרה בין הזמן למקום שנדרש

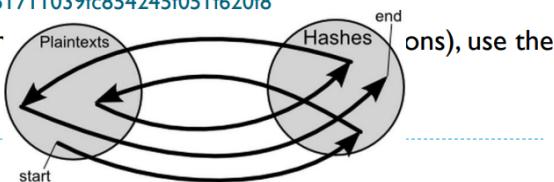
פונקציית ההפחתה Reduction function



- לא מחייב אחרה את פונקציית HASH,
- לא תיקח אותנו מהערך שמננו יצאנו
- במקרה הוא משיג plaintext חדש מה-hash,
- ערך שונה מהערך המקורי שלנו שרצינו להציג

Example of Reduction Function

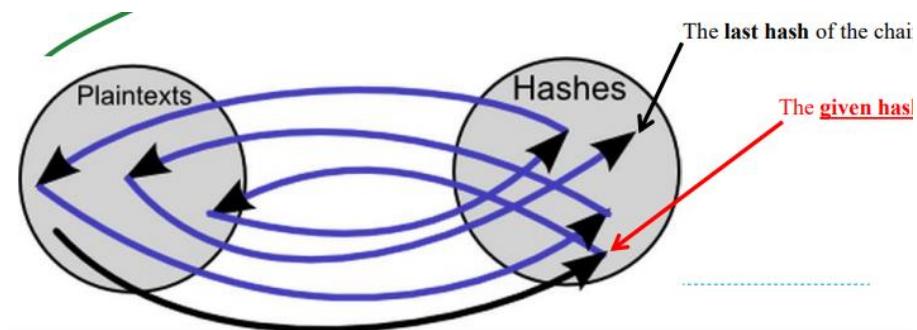
- ▶ Set of plaintexts made up of 7 numbers [0-9]
- ▶ **Hash function:** MD5
 - ▶ E.g., MD5(1234567) -> fcea920f7412b5da7be0cf42b8c93759
- ▶ **Reduction function:** E.g., plaintext \equiv take the 1st 7 numbers from the hash
- ▶ Chain:
 - ▶ MD5(1234567) -> fcea920f7412b5da7be0cf42b8c93759
 - ▶ Reduction(fcea920f7412b5da7be0cf42b8c93759) -> 9207412
 - ▶ MD5(9207412) -> 137fe751711039fc854245f051f620f8
 - ▶ Reduction(137fe751711039fc854245f051f620f8) -> 1377517
- ▶ Stores only the 1st plaintext and the last hash of the chain
 - ▶ 1234567; 137751711039fc854245f051f620f8
- ▶ Once enough chair final rainbow table



אבל זהו מצב שבעייתי לזכרון, אז נשמר רק את הערך שמננו התחלנו והערך הסופי שלו הגענו.
אחרי שיש לנו מספיק שרשרות כאלה, נצטרך להתחילה לשבור אותן.

1. נבדוק אם hash קיים בummuda האחרונה של אחת מהשרשרות האלה:
ניקח את הערך הסופי ונפחית אותו בעזרת פונקציית הפחתה, ולאחר שהוא הולך לplaintext נוכל לעשות לו hash. אחרי שנגיעה לערך hash האחרון של השרשרת, נבין שהערך שאנו חפשנו ממחפשים נמצא איפשהו בשרשראת.
2. נבדוק אם hash קיים בummuda השנייה המוסף של כל השרשרות.
מבצע reduce & hash פעמיים, ואז נבדוק את hash שקיבלנו.
3. נבדוק אם hash קיים בummuda השלישית המוסף של כל השרשרות
מבצע reduce & hash שלוש פעמיים, ואז נבדוק את hash שקיבלנו.

... וכן נמשיך לבדוק אחריה עד שנמצא את hash המקורי שלחנו והוא רוץים למצוא.



התנשויות

- דבר רע לאלגוריתמי HASH ויציר לופם, אבל במקרה של טבלאות קשת, אלגוריתם hash שיוצר יותר התנשויות יהיה יותר בטוח
- סיבות:
 - שרשרות שונות שמתאחות לשרשרת אחת
 - כשמבצעים את פעולה hash עד לנקודה מסוימת עלולות להיווצר לולאות.

אימות משתמשים

אימות בעזרת ס'סמא



- אמצעי הגנה נפוץ נגד כניסה לא מושית
 - המשתמש מספק שם חשבון וסיסמה
 - המערכת משווה את הסיסמה לנזונים שהמורים במערכת עבור המשתמש
 - שם החשבון של המשתמש:
 - קובע שהמשתמש מורה לבצע פעולות במערכת
 - קובע את הרשאות המשתמש

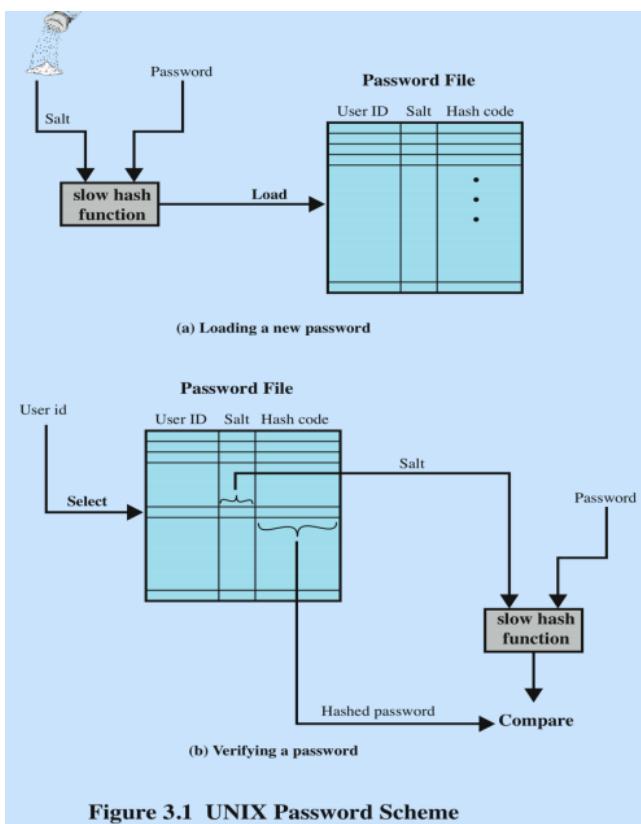


Figure 3.1 UNIX Password Scheme

חולשות השימוש בסיסמא

- התקפה מילונית – יהיה קל לחשב
 - nichush sisama motam moshem
 - nitzul shimush chodor besisamaot
 - nitvor elektronni
 - haftalutot ul emdat uboda
 - nitzul tevuyot moshem

פתרונות – טבלאות גיבוב לסייע

- o התוקף יראה משלו שקשה לו להציג מידע את הסיסמה
 - o Salt – מוסיפים לכל סיסמא שקיימת, מוסיפים משלו רנדומלי, נשרר לסיסמא ונשמר את ה = hash code message digest
 - o אם 2 משתמשים בחרו אותה סיסמא, אז שרשרת של ה salt יהיה לפי הגדרה, ואז הסיסמאות שלהם יהיו שונות זו מזו.
 - o נגד התקפתميلון
 - o ניתן להציג באופן גלוי את ערך המלח כי התוקף עדין לא ידע מה לבדוק הסיסמא האמיתית ומה ערך המלח
 - o חשוב לבחור בפונקציית גיבוב איטית מפני זהה מעכב את

תקפה מילונית

- מתתקפה לא מקוונת (offline attack) o
 - מניחה שהתוקף השיג את קובץ גיבוב הסיסמאות - בהינתן הרשות מתאימות / עבודה מבפנים o
 - התוקף יודע את פונקציית הגיבוב o
 - ניהול הקובץ הוא חלק ממבנה מערכת הפעלה o
 - ערכי המלח של כל משתמש מופיעים באופן גלוי o
 - התוקף מכין מאגר סיסמאות שהוא רוצה לבדוק (מילון) o
 - התוקף בודק כל סיסמה במיילון מול קובץ o
 - הוא ציריך גם להשתמש בערך המלח שנבחר לכל משתמש o

התקפה מילונית – תרגיל

בקובץ סיסמאות יש טבלה עם 2000 רשומות, עריך המלח הוא באורך 12 ביט.
لتוקף יש מיליון עם 10,000 סיסמאות, והtokf יכול לחשב 1000 פעם בשנייה את פונקציית הגיבוב.
* מניחים שקובץ הסיסמאות נפרץ וערכי המלח ידועים.

- כמה זמן נדרש מתקפה מילונית?
- איזה נתון לא יהיה בשימוש? מודיע?

תשובה

$\text{hours} = 6 \sim 3600 / 20,000 \text{ seconds} = 1000 / (10,000 * 2000) = 1000 / 20,000 = 0.05 \text{ hours}$
לא נשתמש באורך של עריך המלח, אם עריך המלח הוא מאד גדול הוא ישפיע על המהירות חישוב (כמה פעמים בשנייה ניתן לתקן) של פונקציית הגיבוב, יש לו חשיבות משנה. אבל כרגע לחישוב הנ"ל אין צורך בו.

Rainbow Attack – קשת

- המטרה: היפוך יעיל של פונקציית הגיבוב, אפילו כשהפונקציה לא הפיכה להיפוך אותה
- נתוני הפעילה דומים למתקפה מילונית
 - התקוף השיג את הקובץ עם טבלת הגיבוב של הסיסמאות
 - לתקן יש מיליון של סיסמאות שהוא רוצה לבדוק
 - התקוף עובד באופן לא מקוון (offline)
- המתקפה מבוססת על חישוב מוקדים (Precomputation)
 - מחשבים לכל סיסמה שנרצה לבדוק עריך גיבוב
 - אם יש שימוש בערך מלח, נרצה להתחשב בכל הצירופים האפשריים
 - מתקבלת טבלה ענקית המתאימה לערכי הגיבוב (hash) לסיסמאות
- בעת במתקפה נחפש כל עריך מלח בטבלה הענקית שהכנו מראש
- חיפוש בטבלה הענקית הוא יעיל כי אינו דורש חישוב פונקציית הגיבוב האיתית
- מניחים שננסה לבצע מספר מתקפות בעזרת אותה טבלה שהוכנה פעם אחת
- שימוש בערך מלח אরוך הופך את המתקפה לבליי אפשרית

מתקפת קשת – תרגיל

בקובץ סיסמאות יש טבלה עם 2000 רשומות, עריך המלח הוא באורך 12 ביט.
لتוקף יש מיליון עם 10,000 סיסמאות, והtokf יכול לחשב 1000 פעם בשנייה את פונקציית הגיבוב.
* מניחים שקובץ הסיסמאות נפרץ וערכי המלח ידועים.

- כמה זמן נדרש מתקפה מילונית?
- שלב הכנת הטבלה
- המתקפה בפועל: נניח שבכל שנייה אפשר לחפש 1,000,000,000,000 עריכי גיבוב בטבלה

תשובה

- ישנו 2^{12} עריכי מלח אפשריים
- עבור הכנת הטבלה צריך לגבב כל סיסמה עם כל אחד מערכי המלח האפשריים
- $2^{12} * 40,960,000 = 10,000 * 40,960,000,000,000$ חישובים של פונקציית הגיבוב = 40,000 שניות (מחלקים ב24*60)

אכיפת בחירת סיסמות חזקות

בדיקות סיסמות פרואקטיבית

- ▶ אכיפת כלים
- ▶ חוקים מפורטים שהסיסמה צריכה לעמוד עליו:
 - ▶ אורך מינימלי
 - ▶ איסור שימוש בפרטיטים אשיים בסיסמה
 - ▶ שילוב אותיות, ספרות, סימנים אחרים
 - ▶ מודד אקראיות: מרוחקים בין אותיות באלף בית ועל פני המקלדת
 - ▶ "אנטロפיה" – מודד כמהי לחזק סיסמה
 - ▶ **הוֹמָגָע** שמציג גישה ביקורתית שונה
 - ▶ הצלבה מול רשימה שחורה
 - ▶ עיבוד קבוצ סיסמות אסורה לשימוש לבניה נתונים בעל גישה מהירה
 - ▶ קיימות שיטות "לנחש" האם פורץ סיסמות היה מייצר את הסיסמה שספק המשתמש, מבלי להריץ את פורץ הסיסמות בזמן אמיתי

- ▶ חינוך המשתמשים
- ▶ הסבר על מאפייני סיסמה חזקה, מתן הנחיות לבחירת סיסמה
- ▶ מחולל סיסמות ממוחשב
- ▶ המשתמשים מתקשים לזכור אותן
- ▶ בדיקת סיסמות ריאקטיבית (תגובהית)
- ▶ מדי פעם המ מערכת מರיצה כל אוטומטי לפיצוח סיסמות על מגיר המשמשים שלה
- ▶ פורץ סיסמות – **password cracker**, כמו John The Ripper, Hashcat
- ▶ בדיקת סיסמות פרואקטיבית
- ▶ המשתמשים בחור סיסמה כרצונו, אך המערכת בודקת אותה בזמן אמיתי ועשיה לדוחות אותה
- ▶ אפשר לבחור סיסמה שקללה לזכירה (למשתמש) תוך כדי שלילת סיסמות קלות לניחוש (لتוקף)

אנטרופיה של סיסמא

- מודד כמהי שמנסה להעיר את חזק הסיסמא
- ככל שהערך גדול יותר, הסיסמא חזקה יותר
- מודד "ביטים של מידע". למשל אנטרופיה 30 פירושה שהתוקף ינסה 2^{30} סיסמות כדי לנחש נכון
- הגנוסחה מניפה שלתוקף יש מידע כליל על יצירת סיסמא
- מעריכים כמה סיסמות קיימות שנוצרו בשיטה דומה
- \log_2 על ערך זה יתן את האנטרופיה
- נוסחה בסיסית: $R = \log_2(R) \times L$
- R – מספר האפשרויות לכל תוו
- L – אורך הסיסמא
- אנטרופיה נדרשת:
 - 29 בית – מוגן מפני תוקף שמנסה סיסמות באופן מקוון
 - 64 בית – המתפרק החזקה ביותר על סיסמות עד היום
 - 96 בית – מוגן בביטחון מפני תוקף בעל כוח מחשוב רב

דוגמאות

דוגמה: sword989

- ▶ על פניו, 9 תווים שלכל אחד מהם 36 אפשרויות (אות קטנה או ספירה)
- ▶ $9 \approx \log_2(36)$
- ▶ בעצם זהו שילוב של מילה נפוצה ומספר בן 4 ספרות
- ▶ יש מספר רשימות של מילים נפוצות. המילה sword מופיעה ברשימת מפורסמת (diceware) המכילה 7776 מילים

$$\log_2(7776) + 4 \log_2(10) \approx 26.2$$

▶

כל תוו ASCII הדרושים במקלדת למעט הרווח (~94), אורך 8

הסיסמה: 78341

10 ספרות בלבד, אורך 5

$5 \log_2(10) \approx 16.6$

הסיסמה: KmpLQrjU

אותיות קטנות (26) וגדלות (עד 26), אורך 8

$8 \log_2(52) \approx 45.6$

הסיסמה: #6qjE!#

כל תוו ASCII הדרושים במקלדת למעט הרווח (~94), אורך 8

$8 \log_2(94) \approx 52.4$

פתרון שגוי: שידור הסיסמא באופן מגובב, שגוי כי עכשו התוקף יוכל לקחת את הסיסמא מחוץ לערך התמציאות

פתרון מקובל: יצירת ערוץ מאובטח (עליו מסתמכים) בעזרת מפתח שיחה ואיומות המשתמש על גבי הערוץ המוצפן

| Client | Transmission | Host |
|---------------------------------------|------------------------------|---|
| U , user | $U \rightarrow$ | |
| | $\leftarrow \{r, h(), f()\}$ | random number $h()$, $f()$, functions |
| P' password r' , return of r | $f(r', h(P')) \rightarrow$ | |
| | \leftarrow yes/no | if $f(r', h(P')) = f(r, h(P(U)))$ then yes else no |

פרוטוקול מבוסס אתגר לשיסמת המשתמש

- המשתמש שולח את המזהה ((p)) שלו לשרת מרוחק
- השרת מייצרnonce (מספר אקראי), ושולח למשתמש
- השרת אינו שומר את הסיסמה עצמה אלא גיבוב שלה ((p))
- חלק מהפרוטוקול, המשתמש מקבל את הפונקציה (H) ופונקציה קשה נוספת להיפוך (f)
- השימוש בnonce מיקשה על תקיפת הפרוטוקול ע"י שידור מחדש של תדרות קודמות

דוגמה לפרוטוקול challenge-response

ביקורת על פרוטוקול מבוסס אתגר

- הפרוטוקול שהציג מבוסס על חישוב קשה להיפוך שהמשתמש מבצע על הסיסמה המגובבת
- אבל השרת מסוגל לבצע אותו חישוב בדיק
- בעיה: הפרוטוקול מוכיח שהמשתמש מכיר את הערך המגובב, ולא את הסיסמה המקורי
- אולי ערך זה דלף מהטבלה השמורה בשרת
- בעיה: תוקף פאטי יכול לבצע התקפה מיליוןית
- הוא רואה את האתגר ואת הפתרון שלו, ומהפץ סיסמה מתאימה
- מצד שני, יש סיכון בכך שהסיסמה תשלח בהצפנה וגילתה (חישוב הפיר) לשרת, או תישמר על גבי השרת
- נרצה פרוטוקול שמאפשר למשתמש להוכיח לשרת שהוא יודע את הסיסמה (הלא מגובבת) מבלי לגלות אותה לשרת

הוכחת אפס מידע

- אנו מנסים לפתח פרוטוקול שבו אחד הצדדים מוכיח שהוא יודע משהו
- הוא יודע את הסיסמה שלו
- במהלך ההוכחה הצד השני לא לומד שום דבר חדש מעבר לעצם הידע
- השרת משתמש שהמשתמש יודע את הסיסמה שלו, אך לא לומד על הסיסמה שום דבר מעבר למה שידע קודם לכן
- סוג זה של הוכחות נקרא אפס מידע (Zero Knowledge Proofs)
- מזכיר את השימוש במפתח הציבורי
- הצד שיצא את המפתח יכול להוכיח שהוא יודע את המפתח הפרטי, מבלי לגלות אותו, בכך שיצפין או יפענה chose שנבחר ע"י הצד השני
- אנו מנסים לפתור בעיה בה המידע הפרטי הוא סיסמה אנושית

Secure Remote Password Protocol (SRP)

- Password Authenticated Key Agreement (PAKE) – שיטות לאיניות זהות או ייצור מפתח סימטרי על בסיס הסיסמה - פרוטוקול איניות סיסמה מרוחק
- Secure Remote Password (SRP) – מבוסס על Augmented PAKE עם יתרונות רבים
- מיושם במגוון ספריות קוד ופלטפורמות (OpenSSL)
- אנו מפר פטנטים קיימים
- אנו דורש צד שלישי
- אנו דורש מצד השרת להכיר את הסיסמה
- התקשרות בין שני הצדדים מוגנת מפני תוקף רשות, אפילו אם הסיסמה אינה חזקה
- תוקף שמנסה להתחזות לצד הלוקה יכול לנחש רק 2-1 סיסמות בכל ניסיון (מתתקפה מקוונת מול השרת ולן היא איטית מאוד)

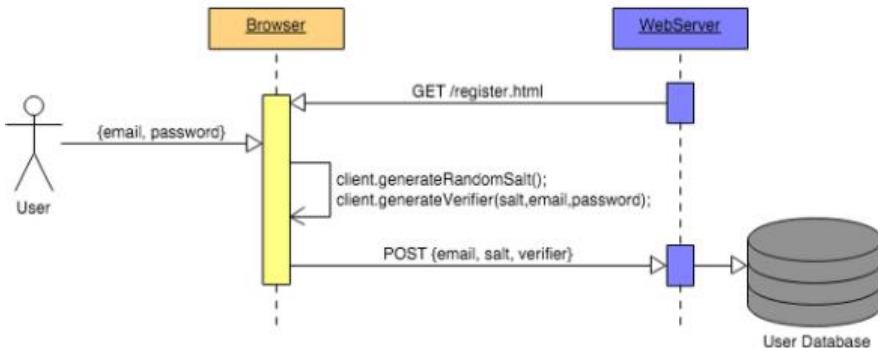
מידע כללי על SRP

- זה פרוטוקול מסווג הוכח לאפס מידע, שבו השירות לא צריך לאחסן סיסמאות (בגרסת גיבוב), והלקוח יכול לאמת את השירות בצורה בטוחה.
- מתקפת Man in the Middle לא יכולה להשיג מידע שימושי על מנת לבצע מתקפה.
- יתרונות ל-SRP:
 - יכול לאמת ללא הצורך בשילוח סיסמה לרשות Man in the middle, brute force & dictionary, כמו איסביות וاكتיביות, מוכיח את הזהות של השירות והלקוח אחד לשני על מנת לוודא שאין מתחזה שמעורב.
 - בנוסף מציע חיבור מאובטח עם מפתח הצפנה, כדי שהשירות והלקוח יכולים להעביר מידע מוצפן אחד לשני ובכך לספק רמה גבוהה יותר של אבטחה.

איך SRP פועל?

```
Client(username, verifier, salt) → Server stores (verifier, salt) for user
```

- ייצור של מלח אקראי
- הלקוח מעביר את השם משתמש, סיסמה ומלה לפונקציה מתמטית KDF ומוחזיר X לאחר מכן זה מייצר verifier, בשימוש עם x ו-SRP Group (=מספר ראשי גודל וגנרטור, אפשר לבחור בין 1024 ביט, 2048 ביט וכו')
- שלוח את ה verifier, salt and the SRP Group לשרת, והשרת יאחסן את המלח של המשתמש



שלב הרישום – SRP

- כל הפעולות החישוביות נעשות באזן כאשר
 - N הוא מספר ראשי גדול, ידוע לכלם, שמקיים מספר תכונות נוספות
 - כל החישובים מתבצעים עם N mod, מבלי שהדבר נכתב במפורש
- ערכים נוספים
 - P – סיסמת המשתמש
 - ID – מזהה המשתמש (username)
 - s – עריך המלח (МОТОР למשתמש לבחור)
 - H - פונקציית גיבוב(hash) קרייפטוגראפית ידועה לכלם
 - g - שורש פרימיטיבי באזן, ידוע לכלם
- שלב 1 – הלקוח מחשב ($S \parallel ID \parallel H = x$)
 - מותר להשתמש בפונקציה מסובכת יותר מ- H
 - המטריה היא לעובד עם הערך x במקומ P, שנראה יותר אקראי וקשה לניחוש – one way function
- שלב 2 – הלקוח מעביר לשרת את הערך $x^g = v$ ואת s-ID
 - נקרא verifier
 - יש להעביר את v באופן מאובטח כך שלא ידלוף
 - השירות שומר תחת האינדקס ID את s-v

- הלקוח שוכח את v ויכול לשוכח גם את x ואת S

סיום הפרטוקול - SRP

- ניתן לבדוק ש $S_{client} = S_{server}$
- הלקוח יכול להוכיח את זהותו שלו לשרת באופן הבא:
 - הלקוח מחשב ערך כלשהו על S , למשל $(S||B||A)H$ ושולח לשרת
 - השרת חוזרת על החישוב עם ה- S שהוא חישב ומשווה בין התוצאות
 - אם התוצאות זהות, לשניהם אותו S
 - בacr, הלקוח הוכיח שידע את x , אותו אפשר לחשב רק מtower P
 - באופן דומה, גם השרת יכול להוכיח שהוא יודע את v
 - עם זאת, הפרטוקול דורש שהלקוח יוכל את זהות הראשן
 - בנוסף, שני הצדדים יכולים לחשב מפתח סימטרי מtower S , למשל $(S)H$

הוכחת אפס מידע

דוגמה מספקית – SRP

- השרת מחשב את הביטוי $(Av^b)^u$
- מכיוון $S = b$ סודי, העובדה שהלקוח הצליח לחשב ביטוי שקול אומרת שנעשה שימוש ב $b = v + g^b$, שהוא הביטוי היחיד שמכיל b שידוע ללקוח.
- רק מי שידעו מראש את x יכול להיעזר בביטוי $(Av^x)(B - g^x)$ שחוקל $\rightarrow B - v$ לחישוב המסקנה היא שאהלקוח יודע את x
- החישוב היחיד שمبוסס על x , ממנו אפשר אולי ללמוד מהו על x , זהה לערך שהשרת כבר חישב בעצמו.
- מכאן שהשרת לא יכול ללמידה מהו חדש על x מtower הערך הזה – הוכחת אפס מידע

אבטחה - התוקף מתחזה לשרת עם הנוסחה של

- התוקף לא יודע את x ואת S
- אבל עם הסיסמה P קלה לניחוש אז הוא יכול לנסוט לחשב אותם מtower ניחושים של הסיסמה
- אם הוא ינסה להתחזות לאחד מהצדדים, הוא יוכל לנסוט מתקפה מלונית מקוונת, אבל יוכל לבדוק רק סיסמה אחת לכל הפעלה של הפרטוקול
- לכן, המתקפה תהיה איטית במילוי ולא ישימה – כמו שציינתי לפני כן
- תוקף פסיבי יכול ללמידה את A ו- B , אך לא להשלים את החישוב
- הוא צריך את v או את b כדי לבדוק אם הניחוש שלו של P נכון
- או לדעת לחשב לוגיריתם דיסקרטי ביעילות
- התוקף יכשל גם במתקפת Man in the Middle
- אם ינסה, אבל לא ינחש את הסיסמה הנכונה, הוא יחשב ערך שגוי של S
- ובשלב אימות הזהות הוא יחשוף והמתקפה תיכשל

הסבר לערך v

- על פניו לא ברור תפקידו של הערך v
- הוא משודר בגלוי
- נתונים שאסור להשתמש קבוע כמו $v = 1$ במקום בערך אקראי כפי שדורש הפרטוקול
- נניח שתוקף מנסה להתחזות לצד הלקוח
- נניח שבמקום v אקראי יש ערך קבוע יודיעו מראש
- נניח שהערך של v התגלתה לתוקף, אך מתקפה מלונית לא הצליחה לגלוות את P או את x
- במקומות $A = P$ התוקף ישלח לשרת את $v = g^x$
- מתקבל $v = B = g^{ab}$ אותו ניתן לחשב מבלי לדעת את הסיסמה
- כאשר חישוב B תלוי ב- v , התוקף לא יצליח את הפרטוקול
- ניתן לבצע מניפולציה כזו:

הסבר לחישוב הערך של v

נשים לב ש- B מחושב כך: $v = g^x + v$

היחסוב לא סימטרי לחישוב של $v = g^x$

מיליא הלקוח מחסר מ- B את $v = g^x$ כשהוא משתמש בו

נדגים שאסור לפשט את הפרטוקול ולהגדיר $v = B'$

↳ נניח שתוקף מנסה להתחזות לצד השרת

↳ נניח שתאת ערך המחל S הוא גילה בעבר

↳ הוא יירץ את הפרטוקול עד שהלקוח ישלח לו הוכחה (ערך שחושב מtower S)

↳ בעזרת מתקפה מלונית הוא ינסה למצאו סיסמה מתוכה אפשר

↳ לחשב ערך v שיתן לו אותו S

↳ כאשר חישוב B תלו依 ב- v , התוקף לא יצליח את הפרטוקול

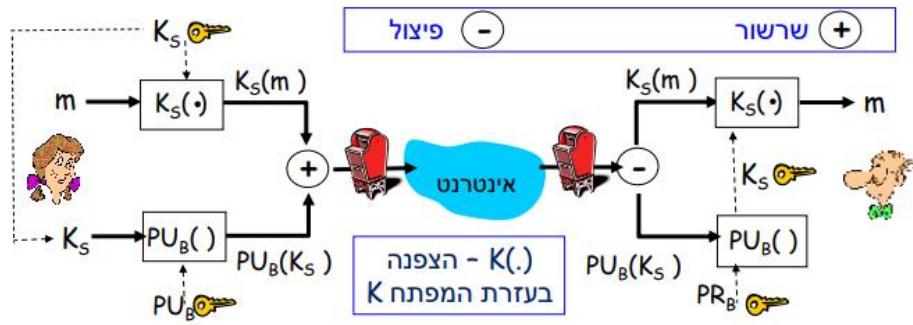
↳ או שייאלץ להמציא ערך של B , אבל לא יהיה לו v

| שלבים | שרות ← לkeys | שרות ← מגריל ערכאים ושולח | שרות ← לkeys חישוב |
|--|--------------------------|---------------------------------------|--------------------|
| מה כל צד יודע בהתחלה | keys | keys | keys |
| P | $v=g^x$ | ID | |
| ID | g | $H()$ | |
| $x=H(P \parallel ID \parallel S)$ שוכח את S ו P | h (מספר ראשוני גדול) | s | |
| a | | $A=g^a$ | |
| $B = v + g^b$ | | b | |
| u | | u | |
| $S_{client} = (B - g^x)^{a+ux} = g^{b(a+ux)}$ | | $S_{server} = (Av^u)^b = g^{b(a+ux)}$ | |

אבטחת דואר אלקטרוני: שכבת היישום (Application)

דואל מאובטח

- אליס רוצה לשלוח דוא"ל חסוי לבוב



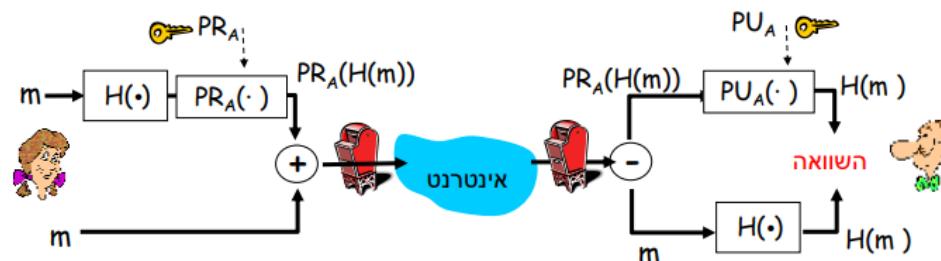
אליס:

- מיצרת מפתח סימטרי אקראי K_s
- מצפינה את ההודעה בעזרת K_s
- מצפינה את K_s בעזרת המפתח הציבורי של בוב
- שלוחת לבוב את $K_s(m)$ וגם את $PU_B(K_s)$

בוב:

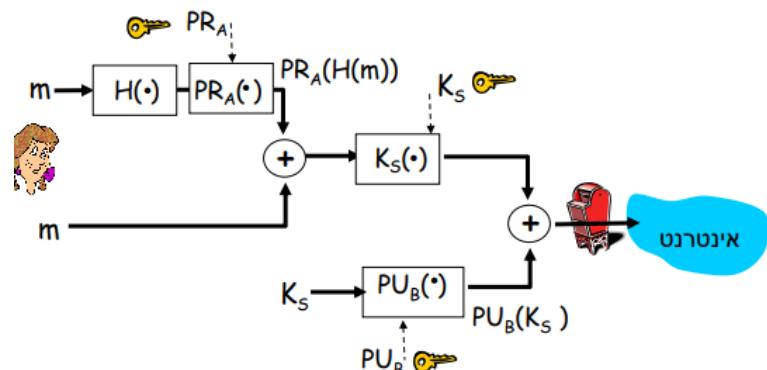
- משתמש במפתח הפרטי שלו כדי לפענה ולהזיר את K_s
- פענה בעזרת K_s את $K_s(m)$ ומאחר את זו

- אליס רוצה לספק אמינות (Integrity) לתוך ההודעה ולזהות השולח ; H – פונקציית HASH, חסר סודיות



- אליס חותמת דיגיטלית על ההודעה
- אליס שלוחת גם את ההודעה וגם את החתימה

- אליס רוצה חיסין (sender authentication), אימות השולח (message confidentiality), ואמינות ההודעה (message integrity)



- אליס משתמש ב-3 מפתחות:
- המפתח הפרטי שלה, המפתח הציבורי של בוב
- מפתח סימטרי חד פעמי

סיכום: רוב הпрוטוקול שראינו איננו שונה מקודמי, המרכיב החדש היחיד הוא הצפנה המפתח הסימטרי וצירוף להודעה- אליס אינה מתקשרת עם בוב ישירות, אליס שלוחת את הדוא"ל לשרת, בוב מושך את הדוא"ל משרת, ולכן אליס אינה מסכמת עם בוב על מפתח אלא מחלטה בלבד, וזה מיידעת (בגוף ההודעה) את בוב על מה שהיא מחליטה

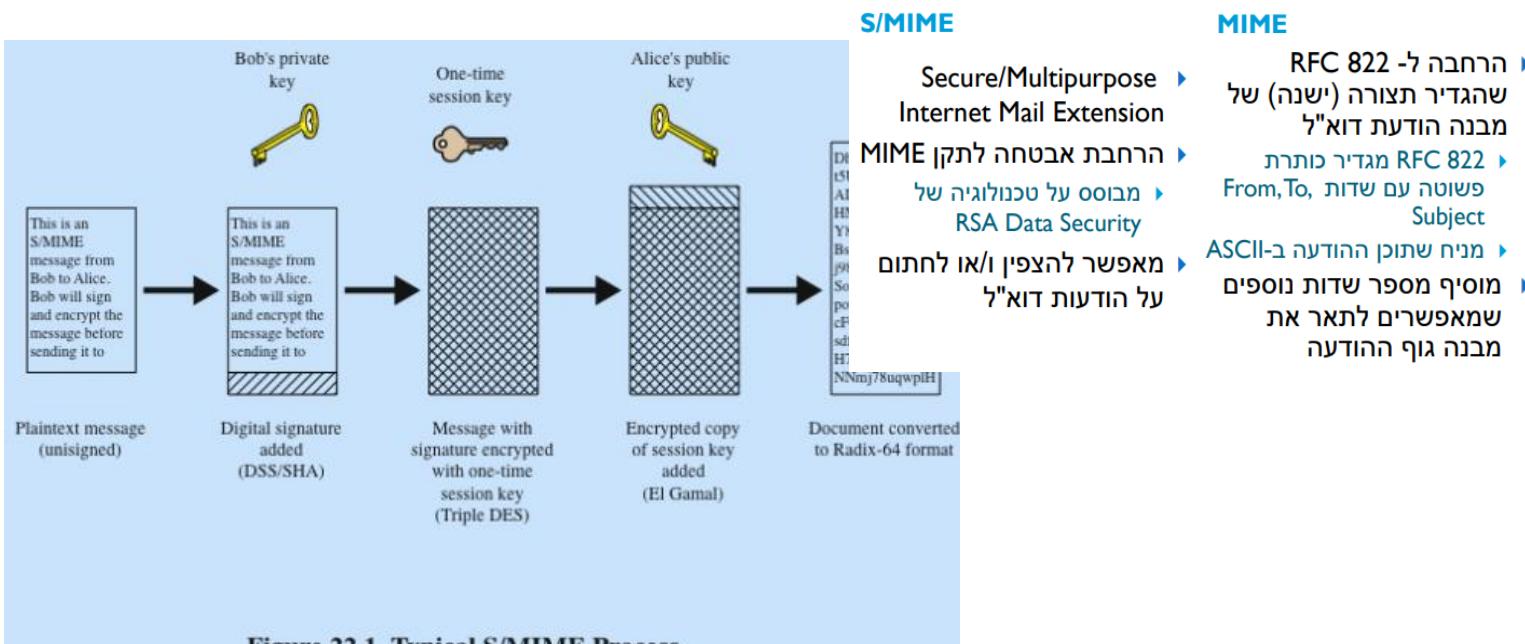
שליחת דוא"ל – ללא אבטחה

- אליס צריכה לבצע באופן חד פעמי מספר דברים כדי לשלוח דוא"ל (לא מאובטח)
 - להציג לעצמה כתובות דוא"ל על שרת כלשהו
 - להתקין יישום שיתקשר עם השירות
 - כדי לשלוח הודעת דוא"ל לבוב, אליס צריכה:
 - להציג את כתובות הדוא"ל לבוב
 - לכתוב את ההודעה בתוכנת עריכה כלשהי
 - לשלוח את ההודעה לשרת המיל שלו
 - לבוב יש פעולה מקובלות כדי שיוכל לקבל הודעות דוא"ל ולקראן אותן:
 - לבוב יש כתובות דוא"ל, שהיא לו הסכם עם שירות מיל, יישום כדי שהוא יוכל למשוך מהשרת את המיל.

שליחת דוא"ל מאובטחת, הצד של אליס

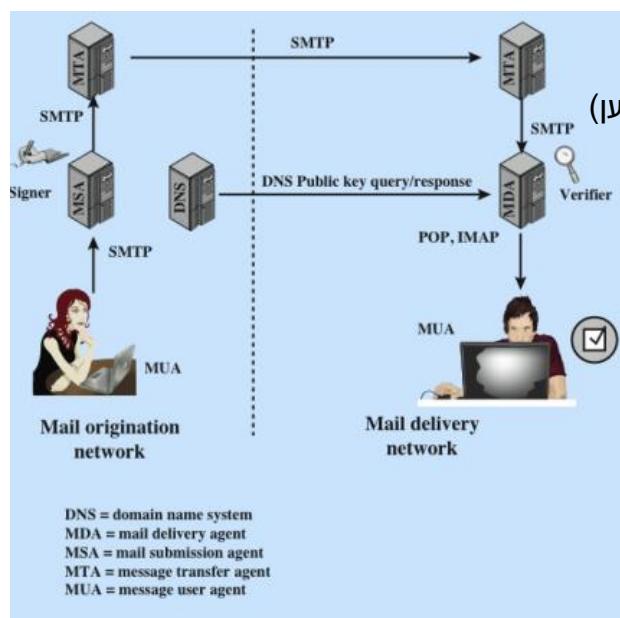
- עברו דוא"ל מאובטח, אליס צריכה לבצע שלבים נוספים:
 - להציג לעצמה מפתח ציבורי ופרטי
 - לדאוג להפצת המפתח הציבורי שלו
 - להתקין יישום להצפנה ופענוח המפתח הציבורי
 - להתקין יישום להצפנה סימטרית
- כדי לשלוח את ההודעה שככבה לבוב, אליס צריכה גם:
 - להציג את המפתח הציבורי שלו, ולאמת את התוקף שלו
 - אם בוב אינו משתמש באותה הצפנה ציבורית, להתקין תוכנה נוספת
 - להגביר מפתח סימטרי
 - להצפן (בחצינה הציבורית של בוב) את המפתח
 - לחתום על ההודעה בעזרת התוכנה שברשותה
 - ולקווות של בוב יש את יכולת לאמת חתימות מסווג זה
 - להצפן (באופן סימטרי) את ההודעה שככבה, עם החתימה
 - ולקווות של בוב יש גם תוכנת פענוח דומה
 - לצרף את המפתח הסימטרי המוצפן ציבורית להודעה המוצפנת
 - לקודד את פלט ההצפנה בתווים דפיסים (printable) בלבד
 - לקוות שבב יבין מה לעשות עם מה שאליס שולחה לו,
 - יהיה לו כוח לבצע את כל השלבים הנדרשים לקריאת הדוא"ל

S/MIME - ו- MIME



DomainKey Identified Mail

- תקן לחתימה דיגיטלית על הודעות דוא"ל המאפשר ל-chain לטען במדויק על הודעות דוא"ל (RFC 8301, 6376, 8463)
- לכן, לא מדובר על אבטחה end-to-end אלא ברמת ספק הדוא"ל (לא משתמש למשתמש)



אבטחת מסרים מיידיים: WhatsApp

תכונות עיקריות

- תוכנית העברת מסרים העוברים דרך שרת
- כך ניתן להעביר מסרים גם אם הצד השני אינו זמין כרגע
- אבטחת end-to-end
 - השרת אינו יכול לקרוא את ההודעות העוברות דרכו
 - תיקוף שיגנוב מידע מהשרת לא יוכל להשתמש בו כדי לקרוא הודעות
 - מבוסס על אמון בשרת:

השרת מתחייב לא לשמור אצליו מידע שעשו פגוע באבטחה, לאחר שאינו נדרש לו יותר Forward Security
- במידה ו מפתחות הצפנה יתגלו, לא יהיה ניתן לעונח הודעות ישנות יותר, רק הודעות עתידיות Group Messaging
- שליחת הודעה לקבוצת אנשים תוך שמירה על אבטחה מקצועית

מפתחות ציבוריים

- WhatsApp משתמש במפתחות ציבוריים לייצרת קשר ראשון בין שני משתמשים
- المפתחות הציבוריים מבוססים על אלגוריתם Elliptic Curve RSA ולא DHM.
- المפתחות הציבוריים משמשים לשיכום סוד "סוד" סימטרי בין שני הצדדים, באlgorigthm דומה לDHM.
 - לכל צד מפתח פרטי ומפתח ציבורי
 - כל צד מעביר את המפתח הציבורי שלו לצד השני
 - כל צד מחשב ערך סוד מ�ור המפתח הפרטי שלו והציבורי של הצד השני
 - הчисוב נעשה באופן שmbטיח של שני הצדדים יהיה אותו ערך סוד
 - לא ניתן לגלו את הסוד ע"י פסיבית למפתחות הציבוריים
 - נדרשת חתימה על המפתחות כדי לסכל מתתקפת MITM

רישום משתמש חדש

- הרישום מתבצע בכל פעם שהרישום מותקן מחדש וכל הchlפת מכשיר הנרדם מייצר מספר זוגות מפתחות ומעבר לשרת את החלק הציבורי של כל מפתח Identity Key – נוצר בעת התקנת הרישום
- – חתום ע"י ה-*Identity*, מוחלף מיידי פעמיים One Time Pre-Key – מאגר מפתחות לשימור חד פעמי, מיידי פעמיים מפתחות למאגר

פניה ראשונית למשתמש

- יוזם התקשרות מבקש מהשרת את המפתחות הציבוריים של הנמען
- One Time Pre-Key, Signed Pre-Key, Identity One Time Pre-Key, Signed Pre-Key, Identity Key – שלושה מפתחות –
 - השרת מוחק את המפתח One Time Pre-Key שלוש פעמים
 - היוזם משתמש בשני מפתחות פרטיים בלבד
 - מפתחה Identity שלו
 - מפתח חד פעמי (Ephemeral) שנוצר הרגעים
- היוזם מחשב מספר סודות מזgoות של מפתח פרטי וציבורי
- ומשרשר אותם יחד לייצור "סוד על" (master secret)

שליחת הודעות

- היוזם משתמש ב-*master secret* כדי לחשב מפתחות סימטריים:
- 256 ביט להצפנה AES בשיטת CBC
- 256 ביט לאימות מסרים בשיטת HMAC-SHA256-Sha256 digest שאורך הוא 256 ביט
- 128 ביט עבור VII
- כל הודעה מוצפנת בעזרת מפתחות חדשים
- המפתחות החדשים מחושבים בעזרת פונקציות גיבוב חד-כיווניות
- הקלט לפונקציות הוא המפתחות הישנים, ומידע נוסף שיחסב מתוך *master secret*
- כל עוד הנמען לא ענה, השולח מצירף את המפתחות הציבוריים שלו להודעה כשהנמען מושך לראשונה הודעה מהשולח (יוזם) והוא משתמש במפתחות פרטיים שלו ובציבוריים המצורפים להודעה כדי לחשב את ה-*master secret*.
- לאחר מכן הוא יכול למחוק את ה- One Time Pre-Key
- באמצעות *master secret*, הנמען יכול ליצור סדרת מפתחות לפענוח ואימות ההודעות להגברת האבטחה, הצדדים ממשיכים ליצור מפתחות פומביים חד-פעמיים ולמצרף אותם להודעות,
- כדי לבצע שינויים נוספים במפתחות הצפנה

הודעות קבוצתיות

- כאשר שולחים הודעה לנ-*N* משתמשים, ישנו 2 ארכיטקטורות אפשריות :
- Client Side Fan Out : הישום השולח מוציא *N* הודעות שונות, אחת לכל נמען
- Server Side Fan Out : הישום השולח מוציא הודעה אחת לשרת, שማיץ אותה לנ-*N* משתמשים
- ב WhatsApp יש ארכיטקטורת Server Side Fan Out
- אך זה דורש שהמפתחות הסימטריים יהיו זמינים לכל חברי הקבוצה
- הפרטוקול שראינו מייצר מפתח שונה שונה לכל זוג משתמשים
- הפתרון :
- ההודעה ה-1 של משתמש לקבוצה נשלחת ב-*Client Side*, ותוכן ההודעה הוא מפתחות סימטריים התחלתיים.
- ההודעות הבאות נשלחות ב-*Server Side* כמספרות ההצפנה והאימות מחושבים במפתחות ההתחלתיים.

אבטחה ב- WhatsApp סיכום

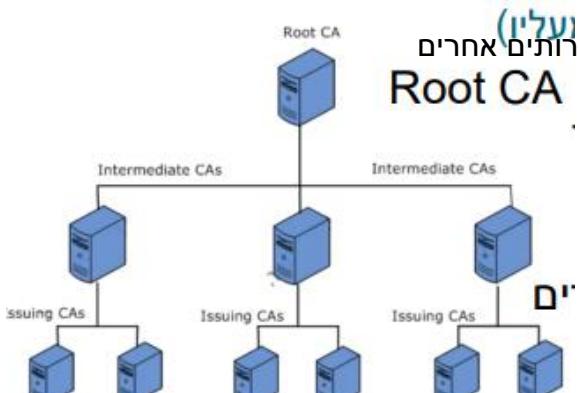
- הצפנה הודעת באמצעות AES
 - פתרונות נוספים להצנת מדיה ושיחות HMAC-SHA256
 - אימות מספרים באמצעות SHA256
 - השרת רק מתוך התקשרות בזמן שהצדדים אינם זמינים
 - ואינו שומר מידע קריפטוגרפי לאחר שהעביר אותו הלאה
 - החלפת מפתחות לכל הודעה
 - לא ניתן "לחשב לאחור" מפתח קודם מתוך מפתח הנוכחי
 - שימוש בפרוטוקול DHM לקביעת המפתחות הראשונים
 - הלהפה נוספת של המפתחות בכל פעם שהנמען עונה חזרה
 - הגנה מפני MITM
 - חתימה על המפתחות הציבוריים של המשתמש
 - החותם הוא המשתמש עצמו ולא צד שלישי אמין
 - בתיאוריה, MITM אפשרי אם התקיפה מתחילה כבר בהפעלה הראשונה של היישום
 - בתפריט היישום: הצגת איש קשר → הצפנה כדי להשוו מפתחות סימטריים

SSL – Web Security, Transportation Layer

- אבטחת תעבורת כל יישום המבוסס TXP
 - למשל בעת גישה לדף אינטרנט רגישים, כמו סחר אלקטרוני
 - באמצעות HTTPS, שהוא HTTP על גבי SSL/TLS
 - שירות אבטחה
 - אימות זהות השירות
 - הצפנת מידע
 - אימות זהות הלקוח (אופציונלי)
 - SSL הוא בסיס ל TLS (Transport Layer Security)
 - ניתן לשימוש גם בישומים שאין גישה (web) כמו SMTP, FTP, IMAP
 - היות שהוא עובר על שכבה 4 (תעבורה)

Public Key Infrastructure (PKI)

- סקורייפטים דיגיטליים שנוצרו ע"ח PKI CA מואמותים באמצעות שרשרת אמון (chain of trust) – העליון בהיררכיה של CAs
 - בד"כ עצמאי ופועל ב offline
 - כל CA מתחת ל CA Root נקרא Subordinate CA
 - צריך להיות מאומת ע"י CA האב (শ্মুলাই)
 - CA הCPF ל Root CA או Intermediate CA אחר
 - בד"כ עצמאי ופועל ב offline



שייחת SSL – עקרונות מרכזיםo **אימות השרת (ע"י הדפדפן)**

- לדף יש גישה למספר מפתחות ציבוריים של CA (Certificate authorities)
- הדף מבקש תעודה (סרטיפיקט) מהשרת, חתומה ע"י CA מוכר
- הדף משתמש בפתח הציבורי של CA כדי להלץ ולאמת את מפתח השרת
- דרך הגדרות האבטחה של הדף ניתן למצוא רשימה של CA מוכרים

o **החלפת מפתחות**

- הדף מייצג מפתח שיחה סימטרי Ks, מצין אותו בעזרת המפתח הציבורי של השרת ושולח לו את מפתח השיחה המוצפן.
- בעזרת המפתח הפרטי שלו, השרת מפענה את Ks.
- עכשו שניהם חולקים את אותו מפתח שיחה

o **העברה מידע**

- על מידע הנשלח ל TCP Socket (ע"י השרת או הלוקה) מוצפן בעזרת מפתח שיחה.
- מידע שמאגייל socket מפענה לפני שימושו.

SSL – פרטימ טכנייםo **לחיצת ID:**

- הלוקה עשוי לשלוח גם את הסרטיפיקט שלו
- שני הצדדים מסכימים על:
 - אלגוריתם להצפנה המידע – AES או 3DES
 - אמצעי להחלפת מפתח סודי – הצפנה RSA או DHM חתום
 - בשיטה זו הצדדים מסכימים על מפתח מסטר לשיחה
- כל צד מייצר מתוך מפתח המאסטר:
 - מפתח הצפנה סימטרי
 - מפתח סימטרי עבור MAC

o **העברה מידע:**

- כל צד מחלק את המידע **לרשומות** (records) קטנות
- כל רשומה מוצפנת בנפרד ומוחשב לה MAC
- ניתן לאמת כל רשומה בנפרד ולא להמתין לכל המידע
- לכל רשומה יש **מספר סידורי**
 - תוקף אינו יכול להעלים או להוסיף רשום 4 ת
- בכל רשומה משולב **nonce** שנקבע עבור השיחה
 - תוקף אינו יכול לבצע replay לשיחה

TLS/SSL הוא פרוטוקול וורטואלי שמטרתו אבטחת **שייחת** שרת/локו בשיטות קרייפטוגרפיות חזקות והוא אמור למנוע ציתות, זיהוי, או חבלה (שינוי דזוני) של המידע העובר בין השרת והлокו. אפשר לחבר אוניברסיטאי, **אימות** שרת (חד-צדדי) או אימות דו-צדדי, תוך שמירה על דיסקרטיות ושלמות המסרים. שלוש נקודות עיקריות שהפרוטוקול אמור לתת להן מענה הן:

- פרטיות** - המושגת באמצעות **הצפנה סימטרית**.
- אימות** - המושגת באמצעות **תעודת מפתח ציבורי**.
- אותנטיות** - המושגת באמצעות **קוד אימות מסרים**.

בפרוטוקול תקשורת שתומך במצב SSL אופצייתו, על הלוקה לידע את השרת על רצונו לעבור לתקשרות מאובטחת, דרך אחת היא להשתמש בפורט ייעודי (מקובל לנוסיף את האות s) שהוקצה על ידי ICANN כמו שער 443 של HTTPS או שערים 989/990 של **FTPS**. דרך אחרת היא לנצל מגנון תליי פרוטוקול ספציפי (כמו בקשת STARTTLS בדואר אלקטרוני) כדי לשלוח לשרת בקשה לעבור למצב של SSL/TLS.

| כל ארבעת הצלופים קיימים: | |
|--------------------------|----------------------|
| Host mode with AH | Host mode with ESP |
| Tunnel mode with AH | Tunnel mode with ESP |

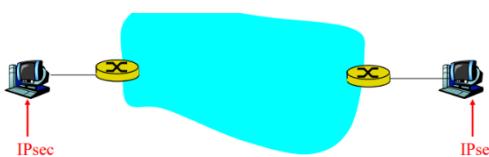
השילוב הנפוץ ביותר
וההשוב ביותר

IPsec - אבטחה בשכבה הרשת

- סודיות בשכבה ה-IP:
 - המאחר השולח מצפין את המידע בחבילת ה-IP
 - לדוגמה, מקטעי TCP ו-UDP: הודעות ICMP
- אמונות ברמת הרשת:
 - המאחר ביעד יכול לאמת את כתובות ה-IP של השולח, ואת אמינות המידע
- ל-IPsec שני פרוטוקולים עיקריים:
 - Authentication header (AH) – אמינות בלבד, ללא חיסון המידע
 - Encapsulation security payload (ESP) – מספק חיסון למידע, ובאופן אופציוני גם אמינות
- שתי צורות הפעלה (לכל פרוטוקול):
 - Transport mode: מגן על המידע בرمות 3 ומעלה. הכוורתה בשכבה ה-IP נותרת ללא שינוי.
 - לא מצפינים את הכוורתה כי הנתב הבא צריך לדעת מי היעד, ואם היעד מזיפן אי אפשר אותו.
 - Tunneling mode: מגן על כל המידע כולל שכבה 3. לאחר מכן מוסיף כוורתה חדשה. הכוורתה כן מוצפנת, כי לוקחים אותה כל החבילה, שמים בתוך חבילה אחרת ואז בתוכה יש את הכוורתה והיעד שלה.



IPsec Tunneling mode



IPsec Transport (Host) mode

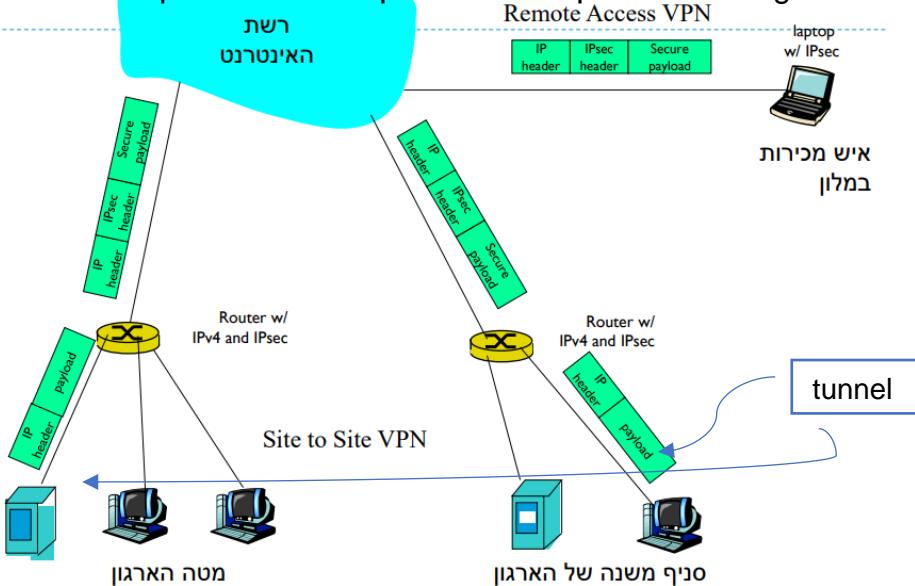
- הנתבים בקצוות הרשת מרים IPsec, המארחים אינם חיבאים בפנים.
- אפשר להצפין את הכוורתה (header) המקורי
- כתובת IP חדשה מצורפת לתחילת הודעה
- End - to - End



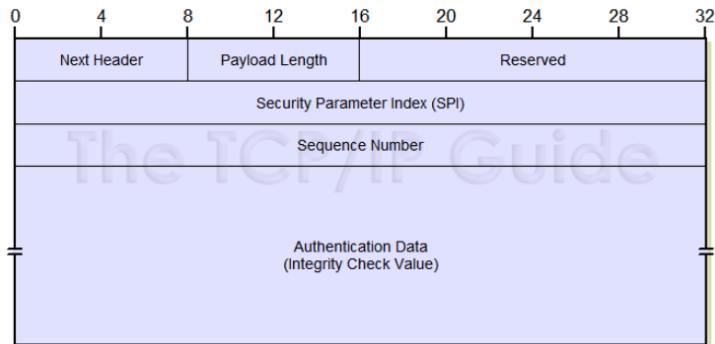
- מנות המידע בפרוטוקול IPsec נשלחות ומתקבלות ע"י תחנות קצה.
- מגן על הפרוטוקולים בשכבות הגבהות יותר (תעבורה, יישום)
- כוורתה ה-IP לא מוצפנת

Virtual Private Networks (VPNs)

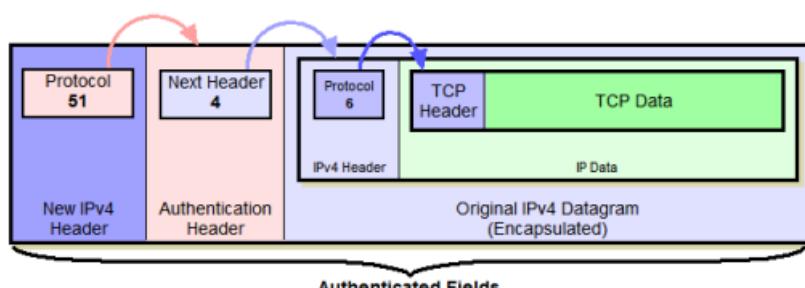
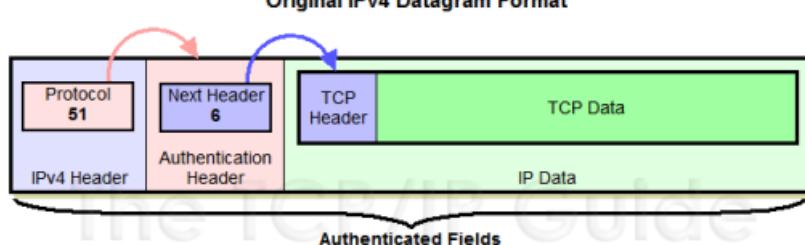
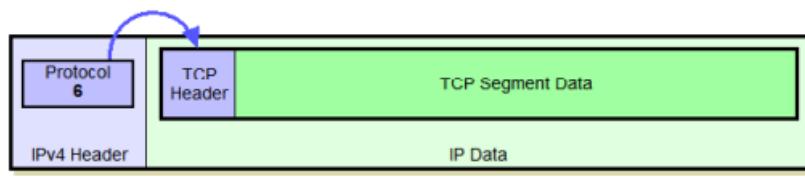
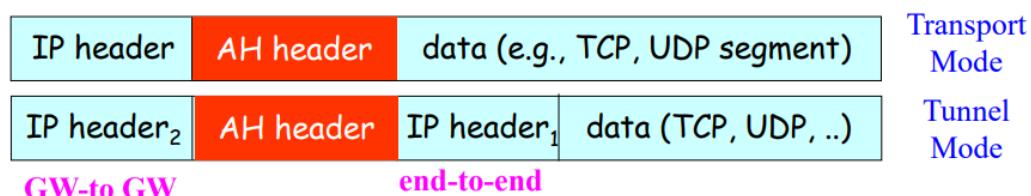
- ארגוני רבים מעוניינים ברשת פרטית לצרכי אבטחת מידע (במקום שרתים במקומות גיאוגרפיים שונים)
- בעזרת VPN אפשר לשולח מידע בין מרכזיים שונים של הארגון על גבי רשת האינטרנט
 - אבל יש להצפין את המידע לפני שהוא יוצא אל האינטרנט למען שמירה על סודיות
- נתבי VPN משתמשים ב-IPsec ו- Tunneling Mode מכיוון שאפשר להצפין את הכוורתה המקורית



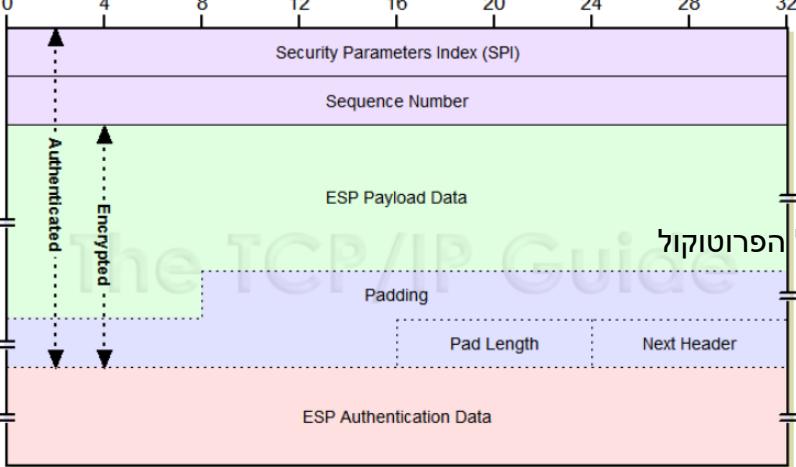
Authentication Header (AH) Protocol



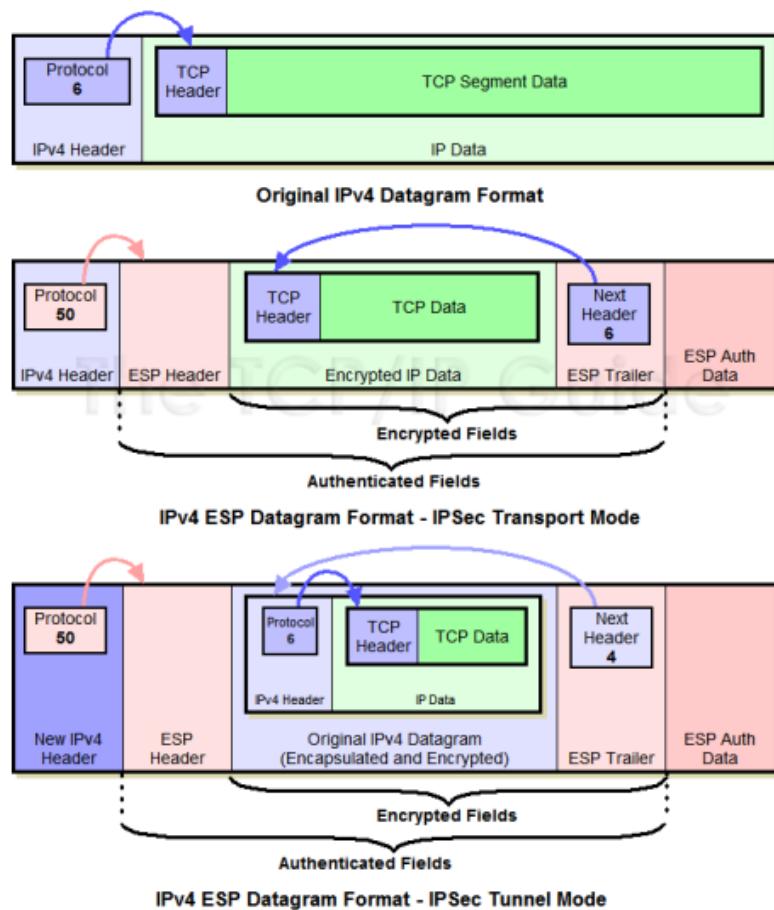
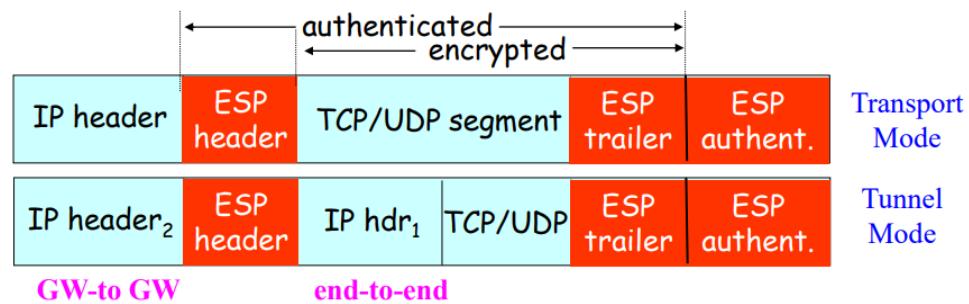
- מספק אימות השולח, אמינות המידע, ללא חיסון
- כותרת ה-AH נכנסת בין כותרת ה-IP למידע
- שדה הפרטוקול ב-IP: 51
- הנתבים הפנימיים בראש מטפלים בחבילות כרגע
- **כותרת ה-AH כוללת:**
 - מזהה (ID) החיבור
 - מידע לאימות
 - חתום בידי השולח
 - תמצית כל החבילות, למעט שדות מסוימים בכותרת (TTL וכו')
- שדה next header מתאר את תוכן החבילה (TCP, UDP וכו') – התוכן המקורי, כדי שהميدע לא יירא לאיבוד



ESP Protocol



- Encapsulating Security Payload (ESP)
- מספָּגָן:** ח'יסיון, אימות השולח, אמינות המידע
- החלק המוצפן כולל את המידע ואת ה-trailer של ה-protocol
- שדה ה-next header נמצא ב-ESP דומים לאלו שב-AH
- שדות האימות ב-ESP דומים לאלו שב-AH
- שדה ה-protocol ב-IP: 50



► **הערה:** יכול להתקיים רק פרוטוקול AH או ESP, לא שניהם בו זמן.

IPsec Security Association (SA)

- גם ב-AH וגם ב-ESP יש צורך בקשר לוגי ברמת הרשת
 - נקרא SA- Security Association
 - נועד לאפשר אילו חבילות יש להצפין/לפענוח/לחתום/לאמת וכייד (הסכם בין שני הצדדים)
 - שמתפקידים, שמסכימים בין הצדדים שיטה הם יעשו את ההצפנה, פענוחים וכו')
- כל SA מתאר ערך מידע וירטואלי לכיוון אחד
 - לכל SA המאפיינים הבאים המזהים אותו:
 - פרוטוקול האבטחה (AH או ESP)
 - כתובת ה-IP בצד השולח
 - מזהה בן 32 ביט (מופיע בשדה ה-SPI)
- בעזרת-SA אפשר לדעת:
 - מהו אלגוריתם ההצפנה (למשל CBC/AES)
 - סוג האימיות (HMAC)
- על מנת ליצור-SA חדש, מתרחש התהליך הבא:
 - פותחים SA מיוחד שנועד להחלפת מפתחות
 - מתבצעת החלפת מפתחות על גבי-SA המווחד

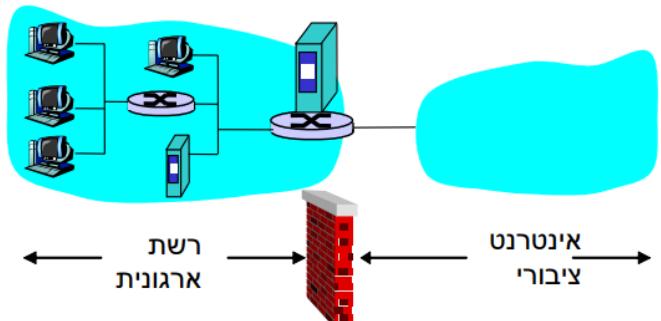
שימוש ב-SA: מסדי הנתונים SAD/SPD

- לכל מארח שMRI IPsec יש – מסד הנתונים של מדיניות האבטחה (SPD) Security Policy Database
- SPD קובע לכל חבילה شنשלחת
 - האם מותר לשימוש ב-IPsec
 - במידה וכן, מהו-SA המתאים (*)
- כל רשומה ב-SPD מזהה חבילות לפי השדות הבאים:
 - כתובות ה-IP של השולח והנמען
 - ה-port של השולח והנמען
 - הפרוטוקול בשכבה 4 (TCP/UDP)
- לכל מארח יש גם מסדי הנתונים המכיל רשומה לכל-SA (SAD) Security Association Database
 - על מנת לשנות חבילה על גבי-SA
 - נמצא את הרשומה המתאימה ל-SA ב-SAD (*)
 - הרשומה תספק את המידע: באיזה פרוטוקול לשימוש, באיזו mode, אלגוריתמים ומפתחות
- כאשר מתקובלת חבילה ב-IPsec
 - נמצא את הרשומה המתאימה ב-SAD
 - הרשומה תספר את המידע הדרושים לפענוח ואיומות

חומות אש – Firewalls

Firewall – מבודדת את הרשות הפנימית של החברה מפני יתר האינטרנט, או מפרידה בין רשותות פנימיות בעלות דרישות אבטחה שונות. מאפשרת לחלק מהחבילות לעבור, וחוסמת את היתר, לפי חוקים קיימים.

מיקום – ממוקמת בקצה הרשות (נקודות כניסה/יציאה), כל התקשרות הנכנסת והיצאה עוברת דרך חומת האש.



מטרות חומת אש

- **מניעת מתקפות שירות (Denial of service)**
- לדוגמא Flood SYN – התוקף שולח הרבה הרבה מאוד חבילות SYN TCP וסוטם את Mbps ה:right>
- **מניעת גישה למשאים או שניינים ע"י גורם לא מורשה**
- למשל, התוקף מחליף את עמוד הבית (homepage) בעמוד אחר
- **מניעת גישה לרשות הפנימית בפנים מי שלא אושר לכך**
- רק מארחים/משתמשים שסומנו מראש יכולים לגשת לרשות הפנימית
- **שלושה סוגים עיקריים:**
 - Stateless packet filters – לא שומר מצב
 - Stateful packet filters – שומר מצב
 - הסינון נעשה ברמת היישום Application gateways

Stateless packet filtering

- הרשות הפנימית מחוברת לאינטרנט דרך נתב firewall
- הנתב מסנן כל חבילה בנפרד, ומחליט אם להעביר או לחסום לפיה:
 - לסן לפי כתובות IP של השולח והנמען
 - לסן לפי פרוטוקול בשכבות התעבורה (TCP, UDP)
 - **מעבר TCP ו-UDP:** לסן לפי ה포רטים (ports) של השולח והנמען
 - **מעבר ICMP:** לסן לפי סוג ההודעה
 - **מעבר TCP:** לסן לפי דגלי ACK ו-SYN

טבלת סינון החבילות

- סינון החבילות מנוהל בעזרת טבלת חוקים
- כל שורה בטבלה מאפיינת קטגוריה של חבילות
- ע"פ השדות בקטגוריות החבילה שצוינו מעלה ^
- כל שורה מגדרה פעולה Accept/Deny
- כאשר חבילה מגיעה ל-wall: firewall
- הנתב מתאים אותה לשורה בטבלה – ע"פ סדר השורות
- השורה הראשונה שמתאימה לחבילה קובעת מה הפעולה שיש לבצע

דוגמאות ל-packet filtering

- חסום את כל החבילות הנכונות והויצאות עבורן 17 = UDP (IP Protocol)
- המידניות:** חסום את כל התעבורה מסוג UDP, לשני היכיוונים
- חסום את כל החבילות הנכונות והויצאות עבורן source port או destination port הוא 23
- המידניות:** חסום את כל התעבורה של פרוטוקול telnet, לשני היכיוונים
- חסום מקטעי TCP נכנים עבורם ACK=0
- המידניות:** מנע מארחים מחוץ לארגון לפתח חיבור TCP מול מארחים בתוך הארגון, מבלי למנוע זאת בכיוון ההפוך

| המידניות | חוקי הפירול |
|--|---|
| לא ניתן לגלוש (web) אל מחוץ לארגון | חסום את כל החבילות עם dest port = 80 |
| מחוץ לארגון לא ניתן לגלוש (web) לשרתים בתוך הארגון, למעט לשרתת ה- web הרשמי של הארגון | חסום את כל חבילות TCP הנכונות לכל IP עם dest port = 80 למעט IP 130.207.244.203, |
| מנع מטרוי streaming להשתלט על רוחב הפס של הארגון | חסום את כל תעבורות UDP הנכונות למעט פרוטוקול DNS |
| מנע שימוש ברשת הארגונית כמקור ל- DOS Smurf Attack | חסום את כל חבילות ICMP הנכונות שה- dest IP שלו הוא כתובת הפצה 130.207.255.255, (broadcast), כמו |
| מנع שימוש ב-traceroute אל תוך הרשת | חסום את כל חבילות ICMP היוצאות מסוג TTL-expired |

Access Control Lists

טבלת חוקים אוטם יש לחשב, מלמעלה למטה, לכל חבילה **:ACL □**

| action | Flow direction | source address | dest address | protocol | source port | dest port | flag bit |
|--------|----------------|----------------------|----------------------|----------|-------------|-----------|----------|
| allow | Out | 222.22/16 | outside of 222.22/16 | TCP | > 1023 | 80 | any |
| allow | In | outside of 222.22/16 | 222.22/16 | TCP | 80 | > 1023 | ACK |
| allow | Out | 222.22/16 | outside of 222.22/16 | UDP | > 1023 | 53 | --- |
| allow | In | outside of 222.22/16 | 222.22/16 | UDP | 53 | > 1023 | ---- |
| deny | In | outside of 222.22/16 | any | any | any | any | any |
| deny | | all | all | all | all | all | all |

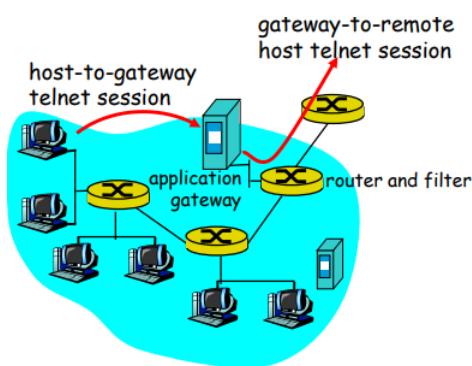
Stateful packet filtering

- סינון לפי קriterיונים מוגבלים
 - ACK = 1-ה destination port לא הגינויות, למשל: חבילה כניסה עם 80 למרות שאין חיבור TCP פתוח.

| action | source address | dest address | protocol | source port | dest port | flag bit |
|--------|----------------------|--------------|----------|-------------|-----------|----------|
| allow | outside of 222.22/16 | 222.22/16 | TCP | 80 | > 1023 | ACK |

- עוקב אחרי חיבור TCP פתוחים Stateful packet filter
 - עוקב אחרי פתיחת קישור (SYN) וסגירותו (FIN) ולכן יכול להחליט אם חבילה שmag'ua שיכת ל קישור פעיל, או לא.
 - יכול מגנון timeout לקישורים לא פעילים, שלאחריו ידחו החבילות.
 - מסוגל גם לעקב אחרי "שיחות" UDP : אם נשלחת חבילה בכיוון אחד, בתוך זמן סביר (עד ה-out timeout) יכולה להציג חבילה תואמת בכיוון הפוך.
 - ה-ACL יכול גם יכול לבדוק את מצב הקישור לפני קבלת החלטה על החבילה.

| action | source address | dest address | proto-col | source port | dest port | flag bit | check connection |
|--------|----------------------|----------------------|-----------|-------------|-----------|----------|------------------|
| allow | 222.22/16 | outside of 222.22/16 | TCP | > 1023 | 80 | any | |
| allow | outside of 222.22/16 | 222.22/16 | TCP | 80 | > 1023 | ACK | ✗ |
| allow | 222.22/16 | outside of 222.22/16 | UDP | > 1023 | 53 | --- | |
| allow | outside of 222.22/16 | 222.22/16 | UDP | 53 | > 1023 | ---- | ✗ |
| deny | all | all | all | all | all | all | |



Application gateways

- סינון חבילות ע"פ מידע ברמת היישום, בנוסף לשודות של UDP/TCP/IP
- לדוגמה: לאפשר למשתמשים מסוימים (users) להשתמש ב-telnet, מתוך הארגון החוצה
- פותר בעיות שנגרמו משימוש ב-NAT
- המשך הדוגמא:

 - כל משתמש ה-telnet יפעילו את היישום מול ה-gateway
 - למשתמשים מורשים בלבד, ה-gateway יפתח קישור telnet מול השירות החיצוני האמיתי,
 - יתווך (relay) בין הצדדים
 - ה-gateway בנתב החיצוני חוסם את כל החבילות ששויות ל-telnet ואין מגיעות מה-gateway

– הסבר Telnet

פרוטוקול **Telnet** משמש בעיקר משתמשים המעורנים להתחבר באמצעות שורת הפקודה בין מחשבים ברשת. מושתמש ב프וטוקול Telnet רק בחיבור לפורט 23. כאשר משתמשים בו לחיבור לפורט אחר הוא יוצר חיבור TCP פשוט, המאפשר להשתמש בו כדי לעובד עם פרוטוקולי תקשורת טקסטואליים כגון HTTP ו-SMTP.

אבלחה היא הסיבה המשמעותית ביותר להימנענות משימוש בטלנט במערכות שרתים או אינטרנט מודרניות לצורכי שליטה מרוחק על השרת. ישנן מספר סוגיות אבלחה ובהן:

- בתקשורת Telnet הנתונים מעוברים כמו שהם (plain-text) ולא מוצפניהם - ובכך עלולות להיחשף סיסמות ומידיע סודי אחר לכל ישوت בעלת יכולות הסנפה ברשת.
- שיטת ההזדהות של פרוטוקול Telnetمامמת רק את זהותו של שולח הבקשה, וכן התקשורת פגעה להתקפת מידע מסווג התקפת אדם באמצעות".

רמת האבטחה הירודה של Telnet נובעת מرمמת האבטחה הנמוכה שנדרשה בעת שפותח הprotokol.

Bastion Hosts-DMZ

Bastion Host : חוקים מגבלים בכינסה לרשות – "מצודות"

- שירות ארגוני המשרת (גמ) משתמשים חיצוניים – לא ניתן לתקשורת הצעת להיכנס אליו לארגון
- נגיד מתוך החברה, וגם מבוחר
- חייב להיות מאובטח מאוד – מערכת הפעלה "זהה": רק תוכנות הכרחיות מותקנות עליו
- ללא קבצים רגילים, ולרוב בהרשאות קרייה בלבד

אזור מפורץ : Demilitarized zone (DMZ)

- החלק ברשת הארגונית שאינו מוגן ע"י ה-firewall הפנימי
- לעיתים מוגן ע"י firewall עם פחות חוקים מגבלים
- נגיש למשתמשים חיצוניים ומשרת אותם
- מכיל אתbastion hosts

magblot h – Gateway firewall

- spoofing דה: לא ניתן לדעת אם המידע באמת הגע מהכתובת המוצחרת – ביטים אפשר לשנות, لكن ברגע שכתובת IP יצא מהמחשב היא תהיה חשופה לכולם, כולל מתקפות (התוקף עלול לשנות את השדות)
- כל application gateway מטפל ביישום אחד
- התוכנה בצד הלוקו צריכה לדעת לתקשר עם gateway
- למשל, ניתן להגדיר בדףן את כתובות gateway- web proxy
- כללי הסיכון הם לרוב "הכל או כלום" עבור UDP
- קשה לנתק תעבורות UDP ולהבדיל בין חבילה "טובה" ל"רעיה"
- Tradeoff: פשרה
 - קלות הת לחברות אל מוחוץ לארגון
 - אמת אבטחה ברשת הארגון
- אין פתרון מושלם: גם אתרים מאובטחים היטב סובלים מתקיפות רשת

מערכות לזיהוי תקיפה – IDS

IDS- Intrusion Detection System

בסיסו על 3 רכיבים לפחות:

- פריסת סנסורים לאיסוף מידע

• אנליזה של המידע לזיהוי תקיפה

• ממשק משתמש

Host based IDS : מחפש התנהוגות חשודה של מארח בודד
Network Based IDS : ניטור התקשרות ברשת

זיהוי מבנים: בדומה לטבלת החקקים של Firewall, מידולהתנהוגות הנורמלית

▪ עשוי להתבצע על עותק של התקשרות, אחרי סינון כלשהו של חבילות

▪ עשוי לכלול state : ספירת חבילות, התנייניתchor בחבילה כלשהי וכו'

▪ **זיהוי אנומליות:** זיהוי מאפיינים חריגיים(רשימת חשודים) של שימוש ברשת,

ע"פ כלליים/למידת מכונה, מידול ההתקפה ← יותר קל לישם את הגישה הנ"ל

Intrusion Prevention System : IPS

▪ מסוגל לקבל החלטות אופרטיביות: מחיקת קבצים, איפוא חיבורים, חסימת פורטים וכו'

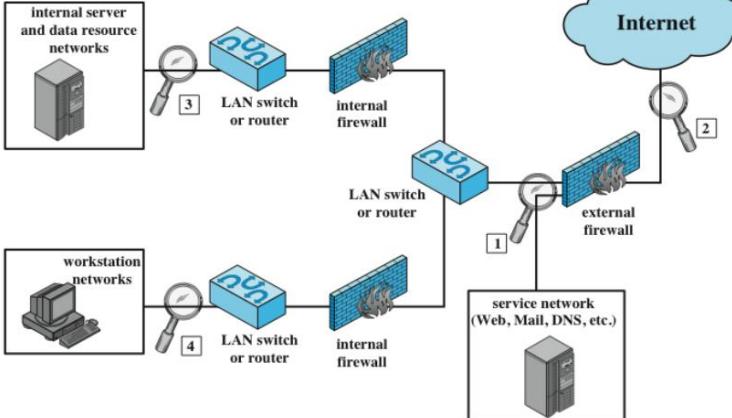


Figure 8.5 Example of NIDS Sensor Deployment

מלכודות דבש – Honeypots

▪ מערכות דמה שמטרתן:

• למשוך את התקוף הרחק מהמערכות הקritisיות

• לאסוף מידע על פעילות התקוף

• לעודד את התקוף להישאר במערכת מספיק זמן כדי לאפשר למנהל הרשות לzechות אותו להתחמדות

מכילות מידע פיקטיבי שמשתמש לגיטימי לא ייגש אליו

המערכת אינה קשורה לכיצות המבצעית של הארגון

• התקשרות הנכנסת אליה מקורה כנראה בסריקות רשות או בתקיפה

• תקשורת שיזכט מהמערכת מבצעיה על אפשרות חדירה אליה

ברגע שהתוכפים נמצאים במערכת, מנהלי הרשות יכולים ללמוד את ההתנהוגות שלהם ולהתגונן מפניהם =

המטרה

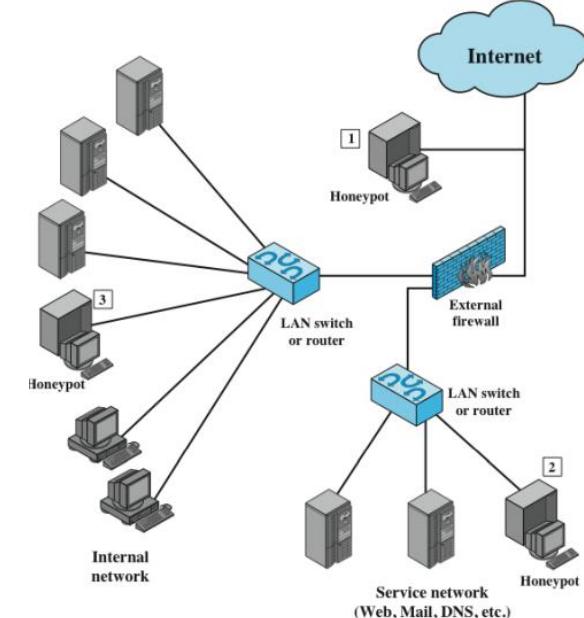


Figure 8.8 Example of Honeypot Deployment

תקיפות רשת

Denial Of Service (DOS) Attack

- מתקפת מניעת שירות
- פעולה המונעת או פוגעת בשימוש המורשה של רשות, מערכות או "ישומים ע"י התשתת המשאים כגון יחידות עיבוד מרכזיות (CPU), זיכרון, רוחב פס ושטח דיסק

הסיבות לתקיפה

- אידיאולוגיה (אנונימיום לדוגמא)
- רוח כספי: תוכנות כופר, פגעה במתחרה עסקי
- שלב ראשון במתקפה משולבת
- מתקפת DOS על ה-Firewall תגרום לו לקרו
- כאשר חומות ההגנה "למטה", אפשר לפזר לארגון ולגנוב/לשנות מידע

מניעת שירות – מה זה

- מתקפת על הזמינות (Availability) של משאב כלשהו
- ישנים משאים שונים שניתן לתקוף דרך הרשות, למשל:



– מתקפות הצפה – Flooding Attacks



- לכל פרוטוקול רשות מתקפות משלו
- המטריה היא להציג את כל רוחב הפס של
העורך שモבייל לשרת כלשהו
- ניתן להשתמש בכל סוג של חבילות
תקשורת

- ב-UDP אין אבטחה טובה מספיק
מאחר ואין את תהליך לחיצת
הידיים בהקמת התקשרות
- ב-TCP יש סכנה בשלב הראשוני של לחיצת הידיים, חשוף לתקיפות

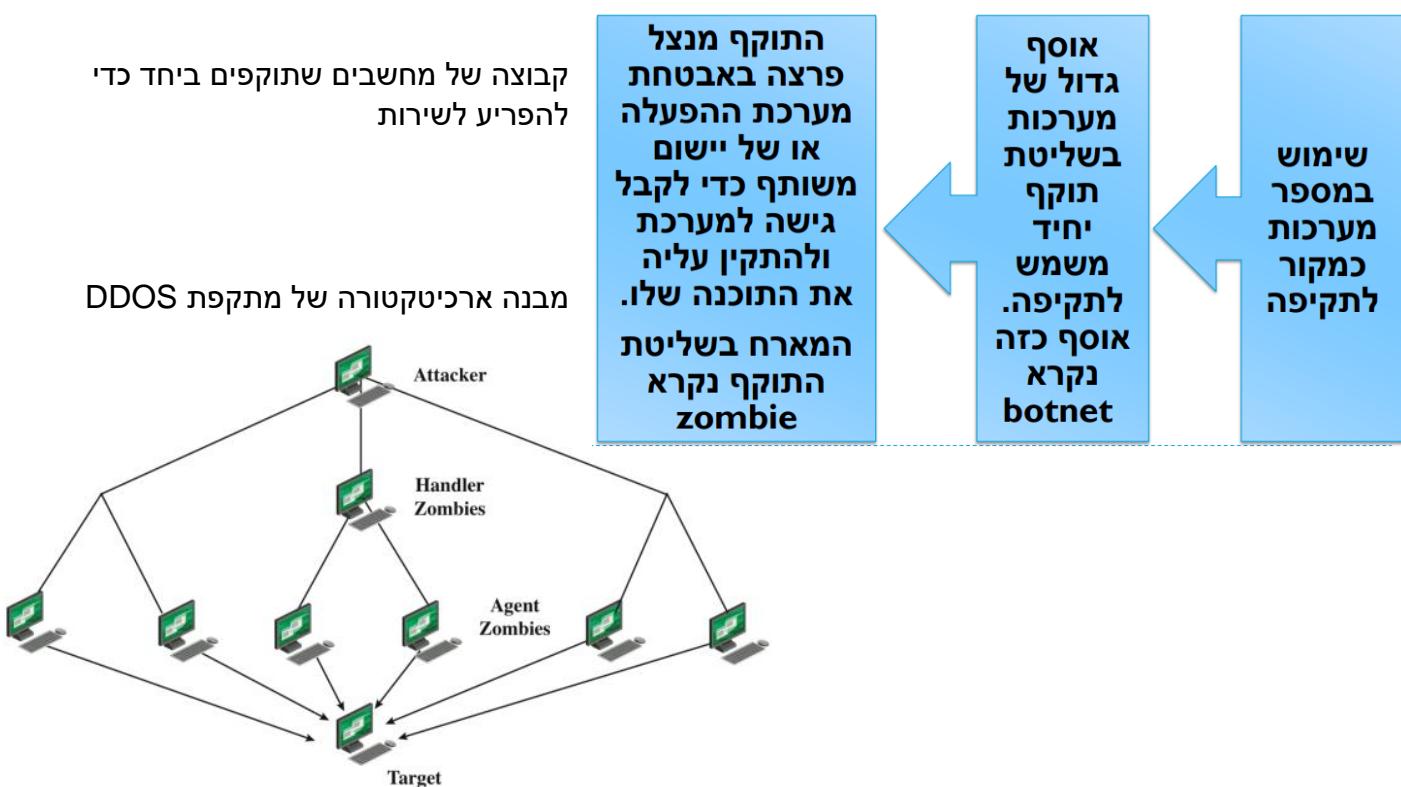
מתקפות DOS "קלאסיות"

- דוגמא: flooding ping command
- מטרת המתקפה היא לנצל את כל רוחב הפס המחבר את הארגון לאינטרנט
- ב"מרכז האינטרנט" יש ערוצים בעלי רוחב פס גבוה שיכולה להתמודד עם ההצפה.
- צוואר הבקבוק הוא סמור לאגרון
- ניתן לגלוות את מקור התקיפה, אלא אם כתובות השולח מזויף (spoofed)
- התקיפה פוגעת באופן מORGASH ומהותי בביטול הרשות

זיהוף כתובות השולח: Source Address Spoofing

- שימוש בכתובות IP מזויפות לשולח
- לרוב ע"י עקיפת שכבת התUberה (raw socket)
- מנסה על זיהוי המערכת התקפה
- התקוף מייצר כמות גדולה של חבילות המכונות כולן אל המערכת המותקפת
- נוצר עומס בנתב האחורי שלפני המערכת המותקפת, שם לרוב הצוואר הבקבוק של רוחב הפס
- זיהוי מקור התקיפה דורש שייתופי פעולה על מנת לשחרר לאחר מכן את המסלול שעברו החבילות
- ספקי אינטרנט מתבקשים למנוע העברת חבילות עם כתובות מזויפות
- פעולה זו צריכה להתבצע קרוב ככל האפשר למקור (האמתית) של התקיפה
- Backscatter traffic
- אמצעי מחקר/מניעתי, מפרטים מסלולים לכתובות IP שאינן בשימוש,
- ואז מתמקדים בחבילות שנשלחות לשם, שהם ככל הנראה חלק מתקפה רשת,
- לכן אם מקבלים חבילות מכתובות IP זאת: אז נראה שזאת תקיפת הצפה וצריך לזרוק את החבילה

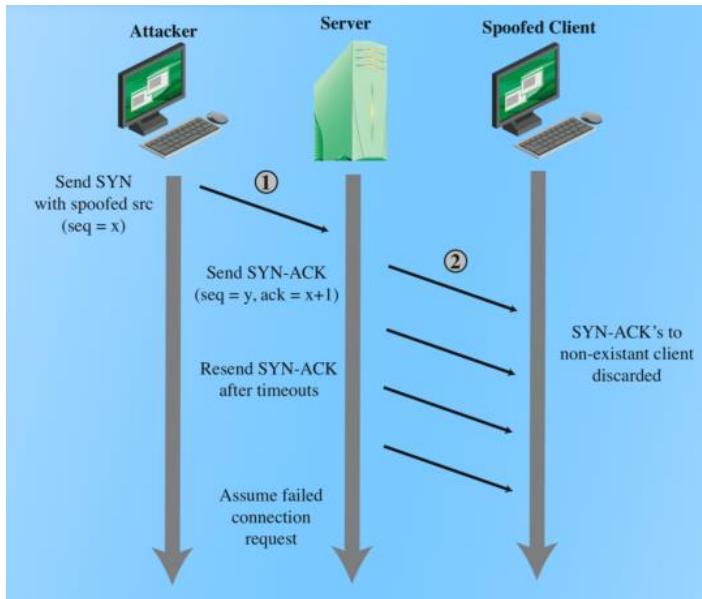
Distributed Denial of Service DDOS Attacks



SYN Spoofing

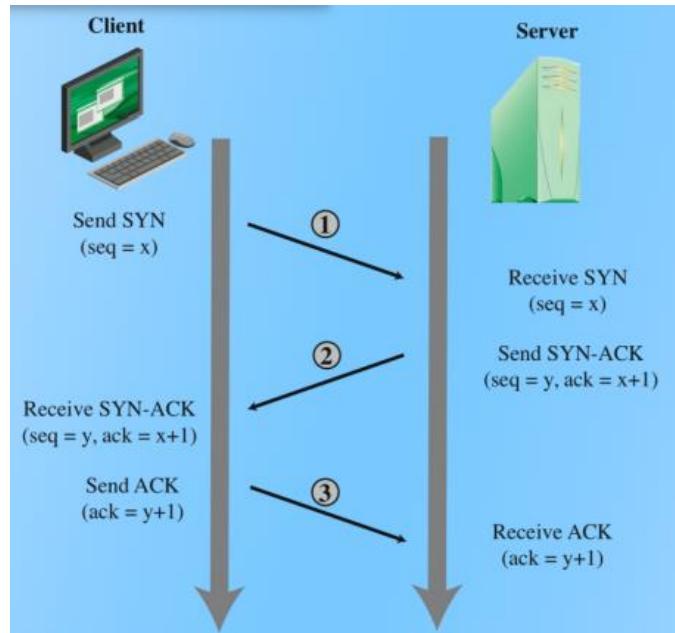
- מתקפת מניעת שירות נפוצה
- פוגעת ביכולת של השרת לטפל בבקשתות עתידיות לפתיחת קישורי TCP ע"י הצפת טבלאות שמטפלות בקישורים אלו
- דרוש הרבה פעות משאבי רשת מתקפות הצפה באופן זה, נמנעת ממשתמשים אחרים גישה לשרת זהי מתקפה על משאבי מערכת.
- בפרט, מתקפה על הקוד במערכת הפעלה שמטפל בשרת

TCP SYN Spoofing Attack



המתקפה מתקיימת בשלב הראשון של חיבור הTCP

TCP 3 way Connection Handshake



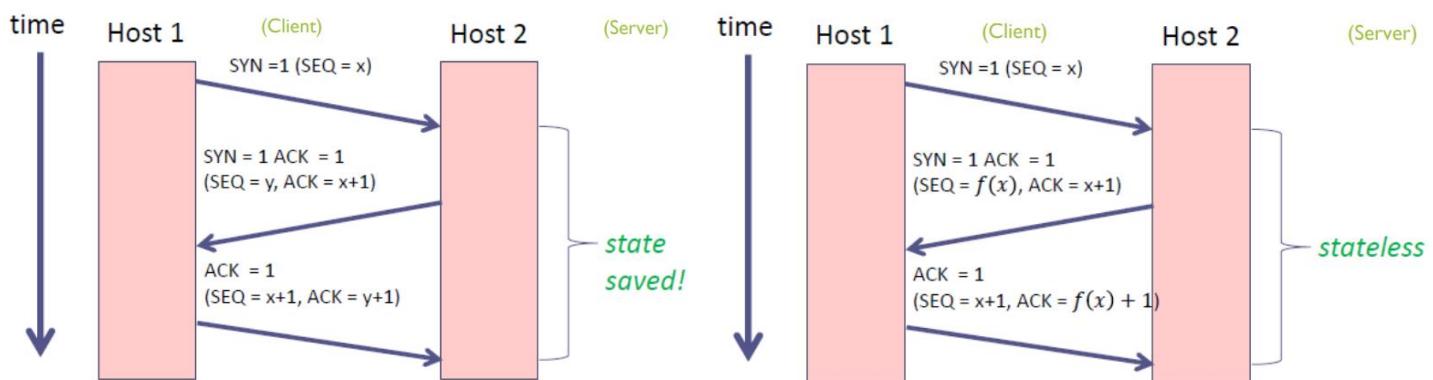
דוגמה - TCP SYN Spoofing

- ▶ **נתוני התקיפה:**
- ▶ שרת יכול לטפל בפתיחת עד 200 קישורי TCP, בו זמנית
- ▶ קישור "חצי פתוח" נסגר אחרי 40 שניות
- ▶ גודלה של חבילת SYN הוא 40 בתים (לא סופרים את כוורתת ה-Ethernet)
- ▶ השרת מחובר לערוצ שרוחב הפס שלו 1Gb/sec
- ▶ **чисובים עבור התקיפה:**
- ▶ $40 \text{ bytes} = 320 \text{ bits}$
- ▶ $3,125,000 \text{ SYN packets} / 1 \text{ Gbit/sec} = 3.125 \text{ seconds}$
- ▶ **תקיפת הצפה קלאסית:**
- ▶ $200 \text{ packets} = 64,000 \text{ bits}$
- ▶ $5 \text{ packets/sec} = 1600 \text{ bits/sec}$

פתרונות לתקיפה: SYN-Cookies

- השרת שולח ערך מייחד ע"ב תור ה-SN (Sequence Number) ההתחלה שלו $(s \parallel H(t \parallel IP_{Client} \parallel IP_{Server} \parallel Port_{Client} \parallel Port_{Server})) = Y$
 - כאשר:
 - T הוא חותמת זמן המתקדים באיטיות (כל 64 שניות)
 - נשים לב ש-t מופיע גם בגלוי וגם בתוך פונקציית הגיבוב
 - X הוא ה-SN ההתחלה של השולח
 - H הוא פונקציה גיבוב קרייפטוגרפית
 - s הוא ערך סודי שידוע רק לשרת
 - S מונע מהתוקף לחשב עצמו את H (ואת y), **replay** מונע מתוקפות t
- תכלס:** השרת דוחה את הקצת המשאים שלו עד השלב השלישי, מקבל בקשה SYN, לא מקצה את המשאים, שולח עוגייה, הצד השני שולח עוגיה חזרה, ורק אחרי שהצד הראשון מקבל את העוגה שהוא שלח, הוא מבין שהוא מדובר עם מישחו שעונה לו ולא מישחו שמאציג אותו, ורק אז הצד הראשון מקצה את המשאים.

Normal TCP Three Way Handshake SYN-Cookie 3-Way Handshake



אופן הפעולה של SYN-Cookies

- **אימות חיבור לגיטימי**
 - תתקבל מהלך חבילת ACK שאינה שייכת לשום חיבור פעיל
 - מחשבים 1 – (client's ACK number) = 'y'
 - מחשבים 1 – (client's SN) = 'x'
 - לוקחים את החלק הראשון של 'y', שמייצג חתימת זמן – ומודאים שהוא עדכנית
 - לוקחים את ערכיו-IP וה포רטים מתוך החבילה שהגיעה
 - לוקחים את הערך הסודי S
 - מחשבים את פונקציית הגיבוב על כל הנתונים, ומודאים שהוא תואמת לחלק השני של 'y'
 - המכיל את הערך המגובב

דיהוי התקיפה:

- התוקף אינו מסוגל לנחש את הערך הנכון של y, כי אין לו את S

יתרונות:

- אין צורך לשמור מידע (state) על-ה-SYN עבור חיבור חצי פתוח עד להשלמת לחיצת היד
- התוקף אינו יכול לזייף את ה-ACK שישלים את לחיצת היד

מתקפות על HTTP - נראות כמו תעבורת HTTP רגילה אז firewall לא ייחסם אותו, אבל השרת יופצץ בהמון הודעות ועולה לגורם לקריסתו אויבוד המשאים שלו

HTTP Flood

- תקיפה נגד שרת HTTP ע"י הפיצתו בבקשת – כי כביכול כל בקשה היא לגיטימית, אז השרת לא יתעלם
- שואב משאים מסווגים שונים
- :Spidering
- הבוט מבקש URL, מוצא את הקישורים שמופיעים בו ומבקש אותם גם.. וכו'

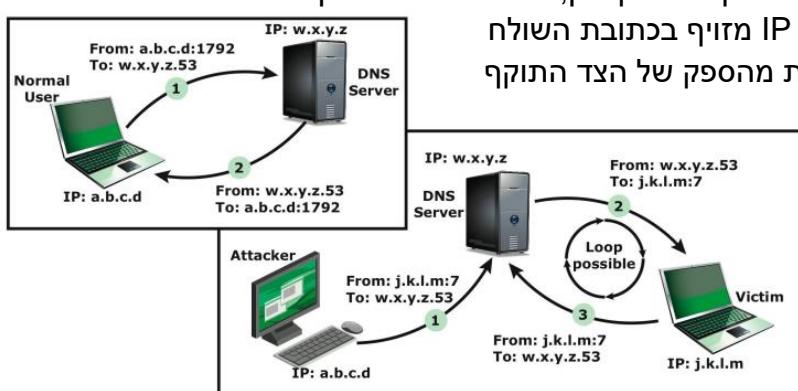
Slowloris

- השתלטות על כל משאבי השרת ע"י שליחת בקשות HTTP שאין מסתיימות לעולם
- מנצל את העובדה שהשרת יכול לטפל במספר מוגבל של בקשות בו בזמןית
- נראה כמו תעבורת HTTP לגיטימית
- אמצעי ניטור ובקרה (כמו IDS) יתקשו לזהות את התעבורה זו

ישם רכיבי רשת שיודעים לעכב חבילות אלו עד שיגיע החלק האחרון של בקשה ה-HTTP

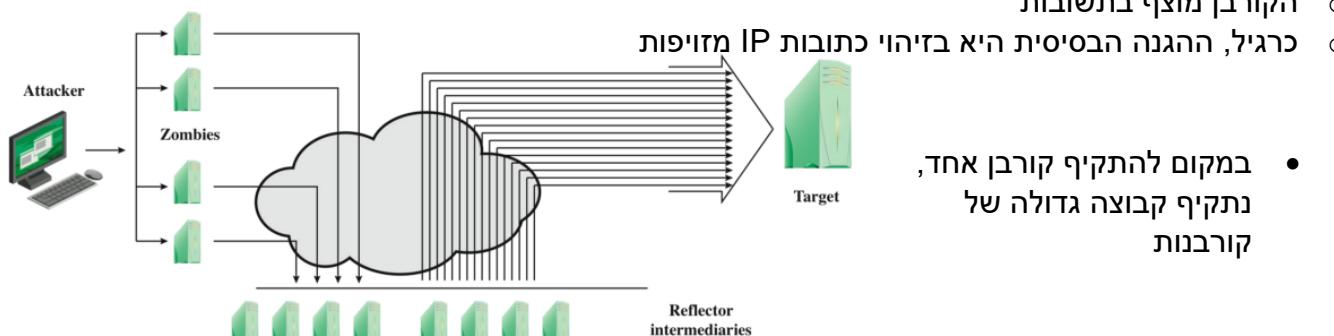
מתקפות Reflection

- התקוף שלוח חבילות לשירות רשות מוכר למארח צד ג' (מתווך), ומציף בכתובת השולח את כתובת-h-IP של המארח הקורבן.
- ברגע ששולחים לDNS בקשה צאת, אבל בכתובת המקור כתובים שזאת בקשה שהגיעה מהקורבן.
- התשובה של הבקשות האלו DNS הן נורא ארוכות, ואז הקורבן מוצף בבקשת שהוא לא ביקש.
- כאשר הצד המתווך עונה, המענה נשלח לקורבן
- המטרה היא ליצר מספיק תעבורה כדי להציג את הקורבן, מבליל שהצד המתווך יחשוד
- ההגנה הבסיסית היא לסנן חבילות עם IP מציף בכתובת השולח
- לא פשוט ליישום כי הפעולה נדרשת מהספק של הצד התקוף



DNS Amplification Attacks - Amplification

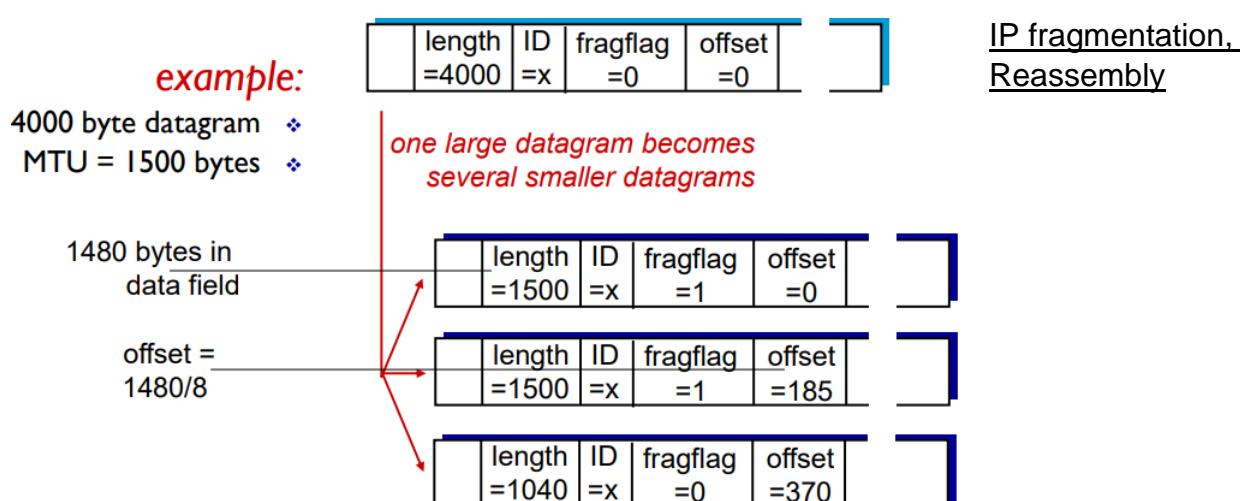
- היכולות נשלחות לשרת DNS שמשמש בטור הצד המתווך
- התקוף שלוח סדרת בקשות-L-DNS עם כתובות ה-IP של הקורבן בטור השולח
- התקיפה מנצלת תוכנה של פרוטוקול DNS שמאפשרת ליצור שאילתת קטנה שהמענה עליה הוא הרבה יותר גדול – ניתן להגיד לייחס של 1:1000



- במקום להתקיף קורבן אחד, נתקיף קבוצה גדולה של קורבנות

חולשות ב-IP Fragmentation

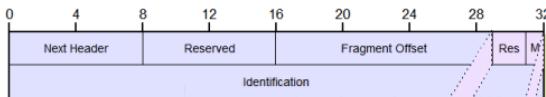
- IPv4 מחלק חבילות גדולות לקטעים (fragments), ה-ID הוא היחיד שנשאר מכל החבילות שודות שונים בכותרת ה-IP מאפשרים למארכ החרון להרכיב חוזרת את החבילה המקורי
- בחבילה שחולקה, כותרת ה-TCP/UDP/IP יופיע רק בקטע הראשון
- מרבית החוקים ב-firewall ניתנים לחישוב רק על הקטע הראשון
- עבור קטיעים אחרים, ה-firewall חייב לתת להם לעبور על פניו, זה לא נראה, כי אם ה-firewall יחליט לסנן את הקטע הראשון,
- שאר הקטעים יזרקו במארכ היעד לאחר שלא יצא להרכיב את החבילה המקורי



Tiny Fragment Attack

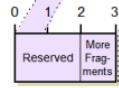
- מתקפה שמטרתה לגרום לתעבורה זדונית לעبور את כל ה-firewall
- התקוף מחלק את חבילת ה-IP כך שהקטע הראשון אינו מכיל את כל כותרת ה-TCP (או ה-UDP)
- כמובן, חלק מהשדות יופיעו רק בקטע השני
- firewall אין דרך להחליט אם לסנן את הקטעים או לה弃 them - אך על הקטעים יגיעו לעד התקיפה
- ניתן להגן בפשטות. בשימוש לגיטימי, הקטע הראשון אינו אמור להיות קטן.
- ובנוסף לזה נסיף חוק ל-firewall שיזהה קטיעים אלו ויסנן את כולם.

IPv6 FRAGMENTATION



הפרגמנטציה נעשית
בתחילת המסלול,
ולכן ההתקפות ברובן
נענות

- ▶ Fragmentation requires the use of the fragmentation extension header.



- ▶ **Next Header** (8 bits): protocol number of the next header
- ▶ **Reserved** (8 bits): not used; set to zeroes
- ▶ **Fragment Offset** (13 bits):
 - Position (in units of 8 bytes) in the msg where the data in this fragment goes
- ▶ **Res** (2 bits): not used; set to zeroes
- ▶ **M** (1 bit): more fragments flag: 0=the last fragment in a msg
1=more fragments are yet to come
- ▶ **Identification** (32 bit): specific value that is common to each of the fragments belonging to a particular msg

- ▶ Only one fragment will contain the layer 4 header

- ▶ Typically the 1st packet
- ▶ Although there are scenarios where it could appear in the 2nd fragment

- ▶ Intermediate nodes cannot fragment IPv6 fragments

=> Path MTU Discovery (PMTUD) is crucial in IPv6

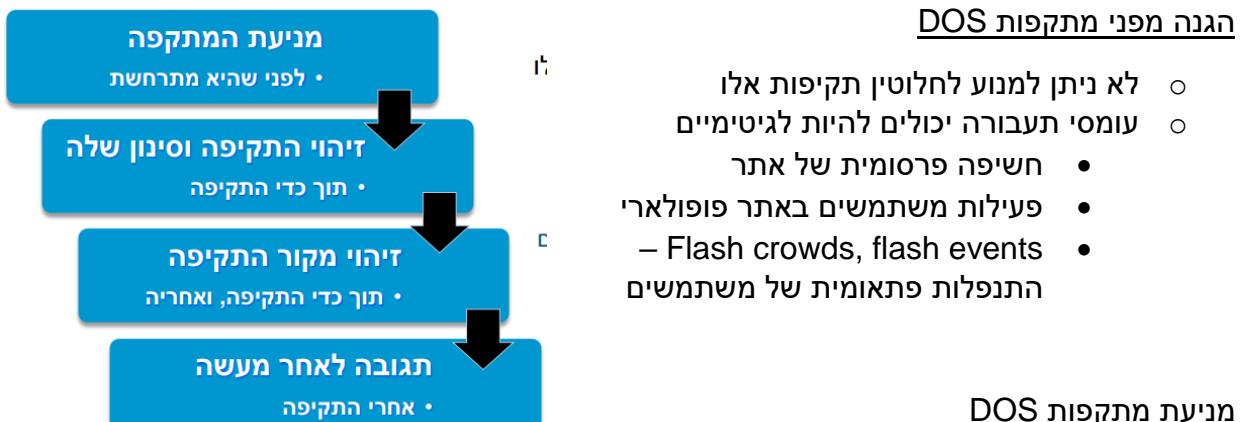
חbillת רעל – Ping of Death

- חbillת רעל (poison packet) – ניצול של שגיאה (באג) במערכת כדי לגרום לקריסתה בעזרת מספר קטן של חbillות, לעיתים אפילו חbillת בודדת
- Ping of Death
 - שדה ה-length בכותרת chbillת IP הוא 16 ביט ← גודל chbillת מקסימלי 65535 בטים
 - התוקף שלוח קטע (fragment) (payload) עם ערך גדול בשדה ה-offset (65528, למשל),
 - ועם תוכן גדול בתוך החbillת (payload)
 - בעבר, מערכות רבות הניחו מראש שגודל החbillת אינו יכול לעבור את 65535 בטים
 - ומקצתן זיכרין בהתאם לכך להרכיב קטעים
 - המערכת קורשת ברגע שתנסה לטפל ב-fragment המזוייף בגלל גלישת זיכרין
 - הערה – תקיפה זו עוברת עם כל סוג של chbillת IP, ואינה קשורה בהכרח לחbillות ping
 - ליצירת ה-fragment ישגרם לגליישת זיכרין נדרשת יכולת טכנית מסוימת מהתוקף
 - TZCOROT: בקשת ping היא סוג של chbillת ICMP
 - 20 בטים של כותרת IP
 - 9 בטים של כותרת ICMP
 - תוכן החbillת בגודל משתנה וניתן לשינוי
 - ה-ping המקסימלי הוא לפיך: $65507 = 8 - 20 - 65535$ בטים
 - מערכות הפעלה מסוימות (כמו Windows 95) אפשרו שליחה של הודעות ping גדולות יותר ping target.example - 65527
 - CRASH!
 - לסייע – ברגע שהתקפה פורסמה, יכולים רצוי לנסתות אותה
 - המתקפה חוסלה בעזרת תיקוני אבטחה למערכות הפעלה
 - כמו כן, ניתן לזרות אותה ב-firewall ולסנן אותה

DNS Cache Poisoning

- המתקפה מכינה מידע שקרי למטרון של LNS (Local Name Server)
- שאלות DNS עתידיות יקבלו את המידע השקרי
- אפשר לשלוח משתמשים לשרת שהקים הtopicן
- דוגמא: נרעיל את המטרון עבור רשותת NS של target.example
 - רשומת ה-NS מפנה ל-DNS הייעודי של domain כלשהו subdomain.attacker.example IN A
 - נשלח שאלת DNS עם: NS target.example IN A
 - שרת ה-LNS ישלח שאלת המשך (כי אין ידוע את התשובה)
 - נשלח לו באופן מיידי תשובה מזויפת: attacker.example IN NS ns.target.example
 - ns.target.example IN A w.x.y.z
 -
- אמצעי הגנה:
 - DNSSEC (Secure DNS) – אימות החתימה על הודעות DNS
 - שימוש ב-nonces קשים לניחס בשדות שונים (message ID, פורטים, ID)
 - בדיקה האם המענה שהתקבל רלוונטי לשאלתה המקורי

4 שכבות הגנה נגד DoS



איך מונעים התקפות? סיכום

מניעה התקפה

- מבצעים אוטנטיקציה של הishiות עליהם אנחנו סומכים
- מודדים את השלמות של בקשות
- מודדים שהמשתמשים/סוכנים שלנו הם מורשים - בקרת גישה (מטרת אבטחה)
- מערכות הפעלה שמספקות אבטחה ע"י הפרדה בין משתמשים "חזקם" עם פרווקיליגיות למשתמשים פשוטים

גילוי התקפה

- לפי תוכנות של ההתקפות
- המידול ההתנהגות הנורמלית, אם רואים שהוא חריג אז מדוחים עליו
- אפשר לנתח ב- offline ובי- Realtime נוכל לבצע ניתוחים במידה וקורה שהוא
- חיפוש ראיות שיוכלו לעזור לנו בתביעה פלילית של התקופים
- עלויות

תגובה להתקפה

- **Block List** – Signature based ← Need to build the signature offline
- **Allow List** – Anomaly Based ← Need to define or build a model of the system

כלי להגנה

- **הגנה היינפית:**
 - אנטי וירוס
 - monitors a process execution to identify potentially malicious behavior – HID
- **Bump in the wire**
 - Routers
 - Firewalls
 - Packet filtering
 - Stateful firewall
 - Application gateway firewall
 - DMZ
 - Guard
 - Control (at the application level) the info exchange between entities of different security levels
 - Even under attack and failure condition
 - NIDS – Passive
 - Monitors traffic entering and leaving the network
 - IPS
 - Honeypot (**מלכודות דבש**)
 - Pure honeypots – full fledged production systems
 - High interaction honeypots – initiate activities of the real systems
 - Low interaction honeypots – simulate only the service frequently requested by attackers
 - Honey farms – centralized collection of honeypots and analysis tools

Cybersecurity at Workplace

- ▶ Security solutions shall be tied to business processes

- ▶ **Training, awareness**

- ▶ **Business processes that enhance security**

- ▶ E.g., proper authorization, strong auth, “need to know”, segregation of duty

- ▶ **“Treat security as an important part of doing business.**

- It is not less important than features and performance” (Bill Gates)

- ▶ **Corporate governance:**

- ▶ **New executives:** Chief Info Security Officer (CISO), Chief Compliance

- ▶ Business managers in all ranks are asked to assume security **responsibility**

* יש לציין כי אבטחה היא בעיה מרכזית, יש קשר לקריפטו, לסייעות, חומות אש ...