

17/3/20 - מתקשב

הכל יהיה תכנים מקצועיים. ספריות קוד.
נקבל ספריות צופן מוכנות
להבין סוגי צופן שונים

תהיה בחינה - הבדקת בקיאות.
לשלוט בסוגים השונים של הצפנות. ברמה מאוד מאוד מדויקת ומלאה.

יהיה בסוף שאלון אמריקאי. 40 שאלות מתוכן לבחור 35 שאלות.

קרברוס - אימות המשתמש - (שרת) x - 500active directory - תקן
כיצד מנהלים אובייקטים? יחידות אבטחה, יחידות ארגוניות, קבוצות הפצה(דיוור)
ניהול מבנה מטריציוני של ארגון.
האגפים השונים - הם יחידה ארגונית. כך מייצרים היררכיה ארגונית. יודעים איפה כל דבר

מה הקשר לקריפטוגרפיה ? - active directory מעבר לחלוקה לאגפים, הוא נותן מענה נוסף - הוא מנהל לנו את תהליך האימות של המשתמש. רץ פרוטוקול קרברוס מאחורי הקלעים (פותח ע"י MIT והוא חנימי), מיקרוסופט אימצו אותו אליהם. וכך כל השוק. מהות הפרוטוקול הוא יכולת לתת תשתית באמצעותה אנשים יוכלו לבצע לוגין למערכת על תווך שהוא חשוף (/מלוכלך). kdc - באמצעות טיקטים, דומה לטיסות לחו"ל - כרטיס טיסה שמציגים לנציג הרלוונטי והוא מאמתת שהגענו למקום הנכון וכך זה נעשה בפרוטוקול הזה.

tgss - מייצר טיקטים. תהליך של זיהוי האובייקט, בדיקת סיסמה (אוטנטיקציה) - השוואה מול המאגרים שהדומיין מחזיק - סוג של האשינג, ההזדהות נעשית פעם אחת, מה שמקשה על מישהו לרכוב לי על היוזר, וגם מוריד את כל העניינים של יוזרים ששוכחים סיסמאות.
ללא ההצפנה, קרברוס לא היה עובד.

הצפנת דיסק - full disk encryption
הצפנה על הדיסק. מצפין את כל הביטים בכונן, ולמי שאין המפתח לא יכול לגשת למידע.
כך שגם אם מאבדים את הכונן, לא נורא. המידע יחסית מאובטח - חשוב, זה עלול להגיע לפגיעה חמורה בארגון.

הסיפור של לאומי כארד - עובד שגנב מידע ע"י העתקה להארד דיסק נפרד (פרטישן נפרד), טס לתאילנד והחזיק את המידע בן ארובה.
ליקויים באבטחה שהתגלו:
גישה לכל המאגר? למה יש למישהו גישה כזו?
למה לא בוצע מעקב על שאילתות חריגות?
למה המשתמש יכול לפרק את הכונן?
איך לא היה מעקב על הדברים האלה?

פתרון אפשרי - הצפנת כל הכוננים הקשיחים. גם אם היה לוקח את כל הכונן, זה היה חסר משמעות כי המידע היה מוצפן.
מה הם בפועל עשו? - עבדו עם משטרת ישראל, התברר שהוא טס לתאילנד, מדינה עם הסכם הסגרה.. כך תפסו את הגאון.
הפתרון שלאוומי כארד בחרו בסוף:
עמדות קצה רזות - אין מחשב לוקלי, הכל מול שרת מרוחק. אין הארד דיסק. (thin client)

תצורות אחסון מידע:

On premises - במקום אצלי, פיסית

cloud - עולמות cloud

הצפנת מידע בענן - עדיין בעולם המחקר.

digital signatures - חתימה דיגיטלית של מידע

כיצד אפשר לדעת שהמידע הגיע מהיכן שאני חושב? איך אפשר לסמוך על המידע?

secure socket layer - לתת תווך מוצפן, הפרוטוקול המתקדם היום הוא tls

יכולת של הצפנה על פרוטוקולים שהוא לא ברמת החומרה, גם על פרוטוקולים שהם לא דווקא מוצפנים.

hashing - הצפנה חד כיוונית.

הצפנה בלי הצורך לחזור חזרה.

24/3/2020 - מתוקשב

קריפטוגרפיה - השימוש בשיטות וכלי הצפנה

cryptology - לימוד על הצפנה ופענוח צפנים

קריפטו-אנליסטים - אנשים המחפשים חולשות בהצפנה (בדרך כלל במפתח, שהוא הסוד)

הצפנה - תהליך השימוש באלגוריתם ומפתח ליצירת מידע מוצפן

שימוש במפתחות כדי לפענח את המידע המוצפן - Decryption

כדי ליישם את זה חייבים להשתמש בשני אלמנטים - אלגוריתם ומפתחות

אלגוריתם הוא מידע שהוא חשוף לציבור, נגיש לחלוטין (open source)

הסוד קיים במפתח (key) - באמצעותו מתבצעת האבטחה

כל האופציות האפשריות של מפתח, במסגרת מה שנתון לי - key space

לדוג', אם המפתח הוא של 32 ביט, האופציות הם לפי בסיס בינארי (של מחשב) - 2 בחזקת 32

תוקף שינסה לתקוף, ינסה קודם כל לשים את היד על המפתח בו משתמשים. אין היום יכולות מיחשוביות

"לנחש" את המפתח. זה מצריך המון משאבים. הופכים את זה ללא כלכלי עבור התוקף.

wep- wireless encryption protocol

בהתחלה סמכו על הפרוטוקול, אבל אז קריפטואנליסטים עלו על אופן הפענוח שלו ואפשר בדקות לפצח

אותו כבר. (ותוקפים ניצלו את זה)

לכן לא משתמשים בו כבר..

מרכיבי ההצפנה

input - plain text

אלגוריתם - נוסחה מתמטית (לא יודע להצפין לבד ללא המפתח(הסוד))

מפתח - סודי

output - מידע מוצפן

הצפנה גרועה היא גרועה מלהיות בלי הצפנה - כלל ברזל בהצפנה.

חייב להישאר מעודכן לגבי צפנים מעודכנים (חכמת ההמונים)

חייב להוסיף עוד מנגנוני הגנה על ההצפנה.

כדאי להשתמש באלגוריתמי הצפנה מוכרים - כי זה אומר שהרבה אנשים בחנו אותו ולכן יורד הסיכוי לחשיפה בהתאם.. (שוב, חכמת ההמונים)

2 שיטות הצפנה שמשתמשים בהם כמעט בכל מקום:
טרנספוזיציה - ביצוע שינוי במיקום האותיות בטקסט (permutation)
החלפה/שחלוף - החלפה של אותיות בתווים או אותיות אחרים

בדוג' האלגוריתם של הטרנספוזיציה (בשקופית 7 בהתחלה), האלגוריתם זה הא-ב האנגלי והמפתח - זה הטבלה והכתיבה בזיגזג
סוג הצפנה זה נקרא זיגזג - rail fence cipher

סוג נוסף של טרנספוזיציה:
צופן קולומוני (שימוש בעמודות להצפנת המידע)-
הצפנה באמצעות עימוד - כמו מערך דו מימדי:
מי שיודע את המפתח - כמה שורות ועמודות ליצור בטבלת הפענוח + סדר העימוד של העמודות עם האותיות יצליח לפענח את הטקסט המוצפן-
tete fwk mn iieyo hlhan gltbo tses hrrna esirt
(עמודות הטבלה - הסדר בו צריך לעמוד את טבלת הפענוח הוא הסוד והוא ממומש לפי סדר מספרים שמוסכם מראש - בטבלה בשורה השניה באדום)
אפשר כמובן לסבך את הקוד הזה עוד יותר אם אתה יוצר משפט ללא רווחים ואז צריך לסכם מראש כל כמה אותיות זו עמודה חדשה...

plain text - the fighters will strike the enemy base at noon

A	T	T	A	C	K	E	R
1	7	8	2	3	5	4	6
T	H	E	F	I	G	H	T
E	R	S	W	I	L	L	S
T	R	I	K	E	T	H	E
E	N	E	M	Y	B	A	S
E	A	T	N	O	O	N	

substitution - אלגוריתם החלפה-
החלפה(שחלוף), הזחה של האלף בית. ולפי זה לייצר מפתח ייחודי שרק אתה ומקבל המסר יודעים.
כמו 13rot, צופן קיסר, atbash(אות ראשונה מומרת בת', ב' ב-ש' וכך מתכנסים פנימה)

31/3/2020 - מתוקשב

שיטת צופן החלפה נוספת (substitution cipher). השיטה של הספרטנים.



שימוש במוט בעל מספר מקצועות (פאות) ייחודי על גביו מלפפים את המסר המוצפן שמתקבל ולפי זה מפענחים את המסר. (כך תקשרו עם לוחמים בשטח).

לוקחים רדיד נייר (ארוך וצר מאוד) ועליו כותבים את ההודעה (מופיע כ-plain text), לאחר מכן מגלגלים את רדיד הנייר על המוט וכך יוצרים מצב שכל אות מופיעה על פאה שונה במוט. (שיטת קולומונר) וכך כל פאה זה היה מילה חדשה. ומקבל המסר מלפף את הרדיד על גבי אותו מוט ורואה את ההודעה כלשונה.

2 מינוחים חשובים בעולם ההצפנה:

מונו-אלפבתי - אלף בית אחד בדיד שעליו אני עובד ועושה כל מיני עיבודים כדי לייצר הצפנות. כמו צופן קיסר
פולי-אלפבתי - שימוש במספר אלפבתיים שונים. שימוש באלפבתיים שונים באותו הצופן.

במונואלפבתי:

דוגמא להחלפה כאוטית (chaotic substitution): מתחילים במילה כלשהי שבחרים ושאר האלפבת ממשיך בצורה ישירה היינו לפי הסדר, לדוג': אם מילת הקוד שלי היא אנגלית, האלף בית שלי להצפנה יהיה:

א	ב	ג	ד	ה	ו	ז	ח	ט	י	כ	ל	מ	נ	ס	ע	פ	צ	ק	ר	ש
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

ז"א מילה וההמשך הוא פשוט לפי הסדר תוך דילוג על אותיות שהם כבר חלק מהמילה. בצופן מסוג זה אסור שהאותיות יחזרו על עצמם. היינו אי אפשר להשתמש במילה כמו אבא (אחרת לא היה ברור איך לפענח את הצופן הזה)

זה יוצר כמות שילובים מטורפת! - 26! (26 לפי כמות האותיות באלף בית בו אתה משתמש) - מספר המורכב מאיזה 27 ספרות.
מדובר על **key space** ענקי! - היינו המון אפשרויות לאלף בית להצפנה מה שאומר שלא ריאלי לאדם לבדוק את כל האפשרויות הללו.

קריפטו-אנליסטים

המטרה של תוקפים היא לפענח את מפתח ההודעה ולא תוכן הודעה ספציפית כלשהו. לפעמים תוקפים יצליחו לזהות טקסט מסוים ששלחתי, אבל זה לא אומר שהם ידעו מהו המפתח שלי. ובזה יכול להתנחם המותקף. לכן בכלל לא את זה מחפשים התוקפים, הם מחפשים את שיטת ההצפנה.

שיטות פענוח מפתח

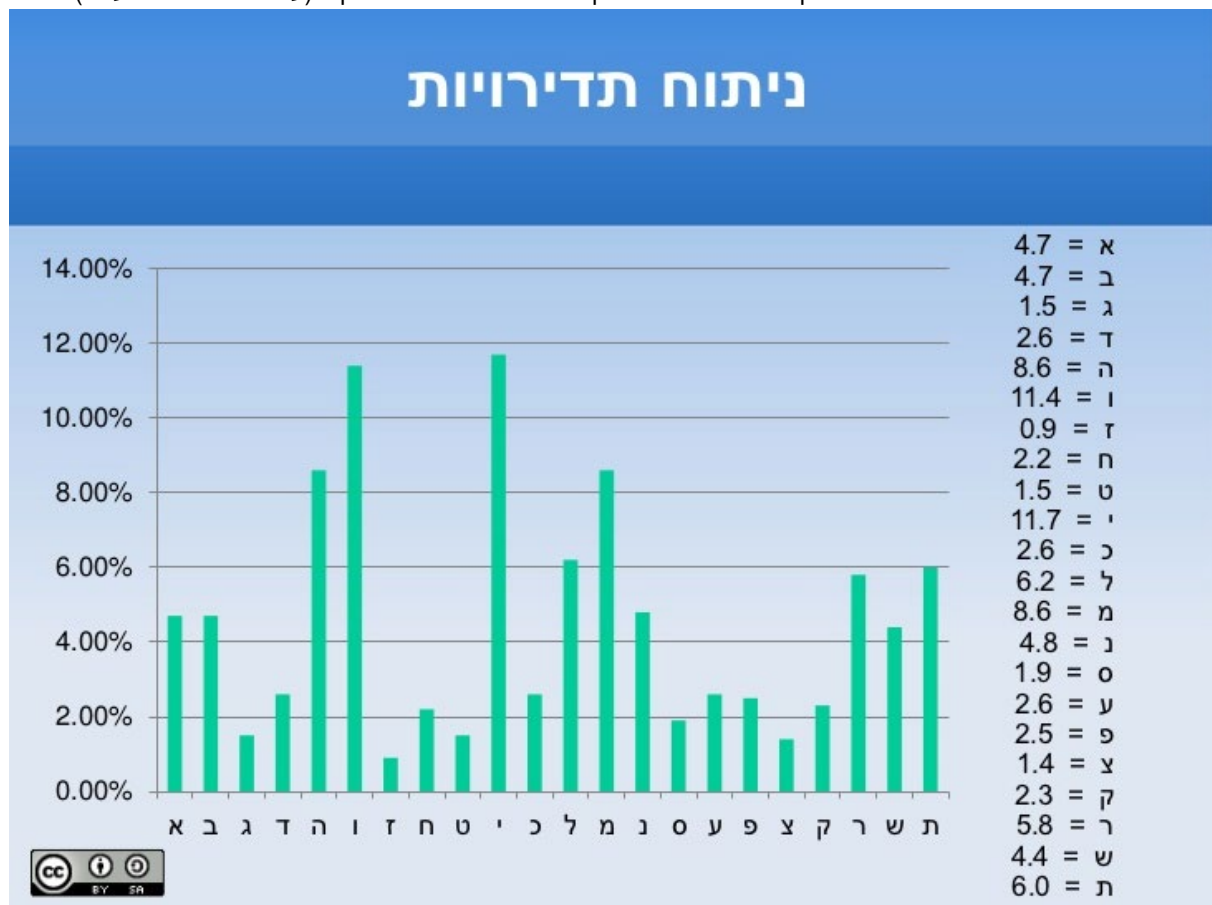
1. **Brute force** - אין לתוקף מושג כלשהו על המפתח מראש, התוקף פשוט מנסה לממש כל אופציה אפשרית. צורת ההתקפה הכי פשוטה והכי פחות אפקטיבית. בעצם לולאה שרצה על קומבינציות אפשריות.

dictionary** - תת מחלקה של *brute force*. בעצם תוקף שיושב על כל המדיות החברתיות של מי שמנסה לתקשר בצורה מוצפנת ואוסף מידע עליהם מהאינטרנט. ובעצם יוצר לולאה שרצה על כל הקומבינציות האפשריות מהמידע שאסף. יותר ממוקד מ*brute force* פשוט

2. *security breach* - חורים באופן מימוש ההצפנה. האם אני מממש את האלגוריתם נכון, האם הגדרתי את זה כך שקטעי קוד שקשורים לאלגוריתם ההצפנה שלי חשופים למשתמש לדוג'..

3. מתקפות לשוניות וסטטיסטיות - ניצול מגבלות השפה - ניתוח תדירויות של אותיות בשפת המקור של ההודעה. במידה ועובדים עם אלף-בית מונו אלפבתי ברגע שאני אנתח את התדירות של כל האותיות בטקסט המוצפן, אני אוכל להשוות בין התדירויות שלהם לתדירות של אות מסוימת בשפה. אם למשל באנגלית האותיות הכי נפוצות הן: e,a,o,t - האות הכי נפוצה בטקסט המוצפן כנראה מייצגת את אחת מאותיות אלה, וכך אמשך בניתוח התדירויות עד שאגיע לחלקי טקסט ומיזה אדע לפענח את השאר.

בגלל ניתוח תדירויות - שדרים הם מסורתית קצרים. כי הרבה יותר קשה להוציא מהם סטטיסטיקות (עד בלתי אפשרי למעשה)



הצופן הפולי-אלפבתי

עקב הנושא של ניתוח תדירויות ניסו למצוא פתרון יותר מלא ומקיף לנושא ההצפנה.

החולשות של הצופן המונו-אלפבתי הביאו לכך שאני בתור מצפין צריך לדאוג לכך ש:

א. ההודעות שלי יהיו קצרות וכך לא פתוחות לניתוח תדירויות

ב. כדי שהצד השני לא יאסוף מידע ויתחמש לקראת ניתוח סטטיסטי אני גם בתור מצפין, חייב לדאוג

כמה שאפשר שמסרים מוצפנים שלי לא יגיעו לידי האויב.

חולשות אלה הביאו לפיתוח הצופן הפולי-אלפבתי. כעת מי שניסה לקרוא את המסר המוצפן שלי יתקשה הרבה יותר.

בפולאלפבתי יש לי המרה מ-אלף בית אחד לכמה אלפבטים מוצפנים שאני רק רוצה. בכל מופע של אלפבת שונה אליו אני מצפינה יהיה ייצוג שונה של אותה האות:

באלף בית הבא, אלף מיוצגת ע"ת, ש ו-ר. בכל אלף בית אות אחרת. וכך אם חלק מהצופן שלך הוא סידרתיות בפניה לאלפבטים השונים בעת ההצפנה, זה אומר שאי אפשר לייצר סטטיסטיקות אמיתיות. כל פעם א' נראית אחרת. אי אפשר לגלות את ערכה האמיתי. או ערך אמיתי של כל אות בטקסט המוצפן.

אלף-בית מקור:

א	ב	ג	ד	ה	ו	ז	ח	ט	י	כ	ל	מ	נ	ס	ע	פ	צ	ק	ר	ש	ת
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

אלף בית 1 של הצפנה

ש	ר	ק	צ	פ	ע	ס	נ	מ	ל	כ	י	ט	ח	ז	ו	ה	ד	ג	ב	א	ת
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

אלף בית 2 של הצפנה

ר	ק	צ	פ	ע	ס	נ	מ	ל	כ	י	ט	ח	ז	ו	ה	ד	ג	ב	א	ת	ש
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

אלף בית 3 של הצפנה

ק	צ	פ	ע	ס	נ	מ	ל	כ	י	ט	ח	ז	ו	ה	ד	ג	ב	א	ת	ש	ר
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

בעצם פה מי שתוקף את הצופן, לא יודע כמה אלף בית מדובר. לא יודע איזה אלף בית בא קודם. יש פה הרבה נעלמים עבור התוקף.

דוגמה לצופן ויז'נר (פוליאלפתי):

[illegible]

כל שורה וכל עמודה מיוצגים ע"י האלף בית הרגיל, ובעצם נוהגים בטבלה כמו בלוח הכפל, א*א זה א, א*ב זה ב וכן על זו הדרך. והמפתח שאתה יוצר הוא לפי כמה שורות לדלג בין כל אות שאתה מפענח

קפיצה בין שורות שונות של מונו אלף בית רבים.

ואין לדבר סוף.. (אפשר היה לשים אינסוף כמות אלף ביתים שונים בטבלה זו כדי לייצג כמות בלתי נגמרת של אפשרויות לאלף בית)

21/4/2020 - מתקשב

one time pad

הצפנה בלתי ניתנת לפענוח. (כמו צופן וירנם - דוג' ל-one time pad)

ההתניות הנדרשות כדי לדאוג לכך שצופן יהיה בלתי ניתן לפענוח:

- בתיאוריה הצופן של וירנם הוא בלתי ניתן לפענוח
אין קשר סטטיסטי בין הטקסט המקורי לטקסט המוצפן
אם יש קשר סטטיסטי, לאורך זמן אפשר לפענח את הצופן (ע"י ניתוח הסתברויות)
- מפתח הוא רנדומלי ולעולם לא חוזר על עצמו (חד פעמי)
בלי רנדומליות והימנעות מחזרות, אפשר שמישהו המשתמש בשירות מסוים (כמו לקוח של בנק הפועלים) יכול להצטרף לבנק, רק בשביל לקחת את הטקסט המוצפן שיוצא מה-plain text שלו עצמו כאשר הוא מעביר שדר (באמצעות סניפר) מתוך זה להבין מה שיטת ההצפנה של בנק, ולאחר מכן לנצל את אותו התהליך כדי לפענח שדר שמישהו אחר (המותקף) מעביר לבנק.
ובעצם ככה לפגוע בהצפנה. - לכן חייב להציג אלמנט של רנדומליות. בעצם מוריד את הביטחון של התוקף שאפשר לפענח את הצופן.
אם הצופן היה חוזר על עצמו בכל מימוש אפשר היה להבין שיש אפשרות לתקוף. זה לא רנדומלי ולכן פתוח למתקפה.
(דוג' לצורך ברנדומליות - במהלך המלחמה הקרה, הרוסים השתמשו בקוד מאוד חזק וטוב, אבל כיוון שהיו חזרות על אותו המפתח כמה פעמים (הם לא עמדו בכללי one time pad) האמריקאים הצליחו לפענח את הקוד של הרוסים והקשיבו לכל הסודות הכמוסים ביותר שלהם ללא ידיעת הרוסים)
חייבים להשתמש ברנדומליות כדי שלא תהיה מחזוריות. אם זה לא יעשה באופן רנדומלי בהכרח אנחנו נהיה רפטיטיביים ועכביים.
- אין רנדומליות אמיתית - אנו שואפים לרנדומליות. (מוגבל מכוח עיבוד וגודל key space וכו')
המפתח באורך טקסט המקור שנכנס להצפנה (plain text)
מאחר ואנחנו מדברים על עידן של שנים של התממשקות בין מחשבים/מאגרים בעידן הזה. כדוג'
התממשקות בין משתמש לחשבון בנק שלו, או בין חולה לקופ"ח שלו, כמות הפעמים המידע מוצפן בכל ששן הוא גדול מאחר כמות ההודעות היא גדולה, אם אשתמש ב-key space קטן מדי (כמו 100 ביט שמאפשר רק 2 בחזקת 100 שילובים) זה בעצם עלול להגיע למצב שאני אחזור על אותו מפתח יותר מפעם אחת באותו הסשן וכך נוצרת פרצת אבטחה שמישהו יוכל לנצל. לכן מה שעושים הוא יוצרים מפתח באורך הטקסט, מה שאומר שאף לא נחזור על אותו מפתח פעמיים. כל פעם הצופן יוחלף (כמה פעמים שצריך בתוך אותו סשן). ככה נאבטח עצמנו בצורה הטובה ביותר
טקסט תמיד יוצפן לפי אורך הטקסט - אסור שתוויים יחזרו על עצמם - כדי ההצפנה תהיה חד - חד ערכית

עקרונות ההצפנה החזקה של קלוד

1. מערכת מוצפנת צריכה להיות גם אם תיאורטית שבירה, בלתי שבירה מבחינתה מציאותית (נצטרך יותר מדי זמן כדי לפצח את הקוד)
2. עיצוב מערכת קוד לא אמור לדרוש סודיות, וגילוי האלגוריתם לא צריך להיות פחד שיהיה למשתמשים בקוד. (היינו גם אם מגלים את האלגוריתם, אין לזה משמעות כי המפתח סודי לחלוטין ובלתי ניתן לפענוח)
3. המפתח צריך להיות זכיר בקלות וניתן לשינוי בקלות
4. אופן שליחת המידע המוצפן צריך להיות קל (דרך טלגרף פעם, או אימייל או טלפון בימינו..)
5. את מערכת ההצפנה ניתן יהיה להעביר ולהיזיז בקלות ע"י אדם בודד
6. השימוש במערכת צריך להיות קל ונגיש לכל אדם (לא צריך ידע מיוחד כדי לתפעל)

5/5/2020 - מתקשב

הצופן האולטימטיבי:

1. הצופן צריך להיות חד פעמי. כי אם עושים שימוש חוזר, ברגע שהמפתח נחשף אז המידע המועבר בשיטה הזו פגיע - וחשוף.
מאוד קשה להגיע למצב כזה של חד פעמיות - כי ברגע שמקציבים שטח זיכרון אז מין הסתם בשלב מסוים נגיע למחזוריות - זה המשמעות של keyspace גדול.
היום ההצפנות הם בחזקה של 4000, 2048, 512 ומשהו - זה הרמות הצפנה עליהם אנו מדברים היום.
גופי התקינה הוציאו את 5md משימוש (לא המליצו להשתמש בו) - כי הוא 2 בחזקת 128. עוד 10 שנים הצופן הזה יהיה לא רלוונטי. ככל שכח העיבוד מתגבר הצפנים החלשים נהיים יותר ויותר פגיעים. (כח עיבוד >---< כח המפתח - ככל שכח עיבוד גדול יותר, מגדילים את Keyspace של המפתחות - כדי להמשיך ולשמור על המידע שלנו.)
2. אורך המפתח כגודל המידע שאני מצפין. אם אני אשתמש במפתח קצר מאורך הטקסט שלי, יתחילו להיות לי חזרות במפתח ההצפנה, מה שיהפוך את המידע לחשוף יותר.

בשיטת הצפנה של AES-256 - יש חלוקה של קובץ המקור לחלקים - לבלוקים קטנים ואז מצפינים את החלקים. 2 בחזקת 256. מפתח סימטרי.
אחרי החלוקה לבלוקים, זה כל בלוק בנפרד. ויכול להיווצר מצב ששני בלוקים יוצפנו עם אותו הקוד בדיוק, כיוון שקובץ המקור למשל היה אחד טרה, והמפתח המוגבל שלנו של 2 בחזקת 256 הגיע למצב שהוא חוזר על עצמו בין הבלוקים השונים.

start initialization vector - (כמו סיסמה להצפנה) כדאי שיהיה כמה שיותר ייחודי, אחרת באמצעות מילון עליי יפרצו את הסיסמה שלי, וכך גם את ההצפנה שלי.

המפתח חייב להיות רנדומלי - אחרת תהיה חזרה בהמשך. מדובר בpseudo random - אין אפשרות לרנדומליות אמיתית, לכן שואפים לרנדומליות על מנת שלא תהיה חזרה

3. למנוע קשר סטטיסטי בין הcipher text שלי לplaintext.

קריפטוגרפיה מודרנית

★ שיטות הצפנה היום מתבססות על תצורה בינארית. (שפת מחשב)
כיוון ש0 ו1 זו השפה הנפוצה ביותר - לכל המחשבים (זהה בכל העולם) - כך כל המחשבים מתקשרים

★ אלגוריתם הצפנה בימינו מיושם בצורה מתמטית. דוג' לשיטת הצפנה הקיימת בימינו היא XOR (exclusive or):

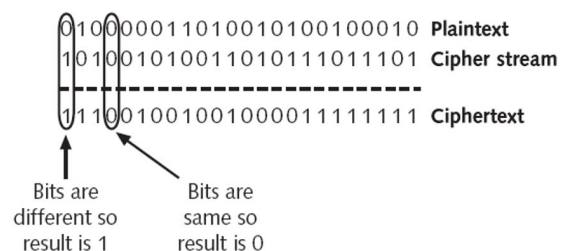


Figure 11-10 Creating ciphertext with XOR

רק אם יש הבדל בין המקור למפתח הקידוד התוצאה היא 1 - אחרת התוצאה היא 0 (false)
למעשה כל שינוי שעד עכשיו עשינו על אותיות, נעשה כעת ברמת הביטים על 0 ו1. (substitution transposition)

מינוחים של שיטות הצפנה בעולם ההצפנה המודרנית:

- ★ p-box - permutation - כמו XOR שינוי כלשהו, כמו
- ★ s-box - substitution - הזחה של מספרים לפי מפתח מוסכם - מראש

אופן יישום המידע - האם ההצפנה שלי היא התצורה של בלוקים, או בתצורה של מספרים. בלוקים (block cipher) מתייחס למצב בו לוקחים מידע ומגדירים חלוקה של אותו מידע לבלוקים בגודל מוגדר מראש. (כמו מיליון לחלק ל128). כל בלוק מוצפן בפני עצמו. מקבל המידע, יצטרף לפענח את הקוד, ולחבר את אותם בחזרה ביחד לקובץ מידע אחיד. בבלוקים, אם אחד מהם נופל אז כל המידע corrupted - אי אפשר כבר לפענח חזרה את המידע.

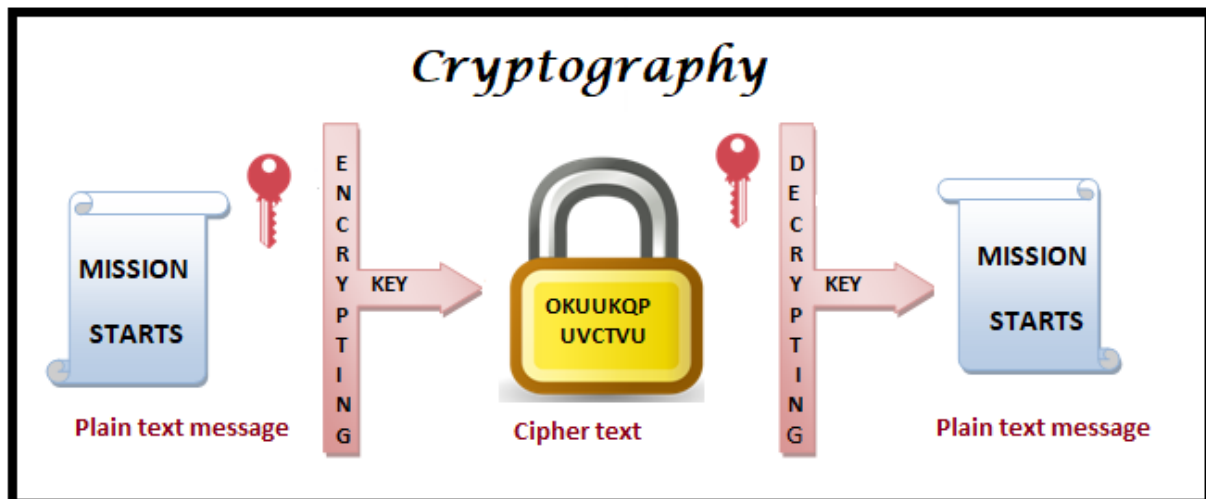
◀ סטרים (stream cipher) יודע לעבוד ביט ביט בלבד - במקום חלוקה לצ'אנקים. כשחשובה לנו המהירות דווקא, נעבוד בתצורה זו.

12/5/2020 - מתקשב

חבילות של מידע (בלוקים) - יחידת המידה שאנחנו עושים שימוש בבלוק ובקי היא אותה יחידת מידה - יחידת המידה היא ביט. רק הגדלים משתנים. גודל הבלוק הוא אחיד לכל בלוק ובלוק המרכיב את המידע שלנו - נשאר באותו גודל לכל אורכו. היכן שיצא פאקטת מידע קטנה מידה (שארית) האלגוריתם יוסיף ביטים שהם null, כאשר תישאר שארית שלא תגיע לחלוקה שלמה.

סטרים - מעביר ביט ביט לעבר היעד. הצפנה ברמת הביטים.

בהצפנה מודרנית אנו עושים שילוב בין s-box לp-box - כדי לשבור את הקשר הסטטיסטי בין ה plain text לcipher text וכדי להקשות על התוקפים, משלבים בין שיטות הצפנה אלה ולא מסתפקים באחד.



בבלוקים ייתכן שינוטר חלק מהמידע אם יצליחו להשיג בלוק אחד, בסטרים זה בלתי אפשרי - אין מה לעשות עם ביט בודד.

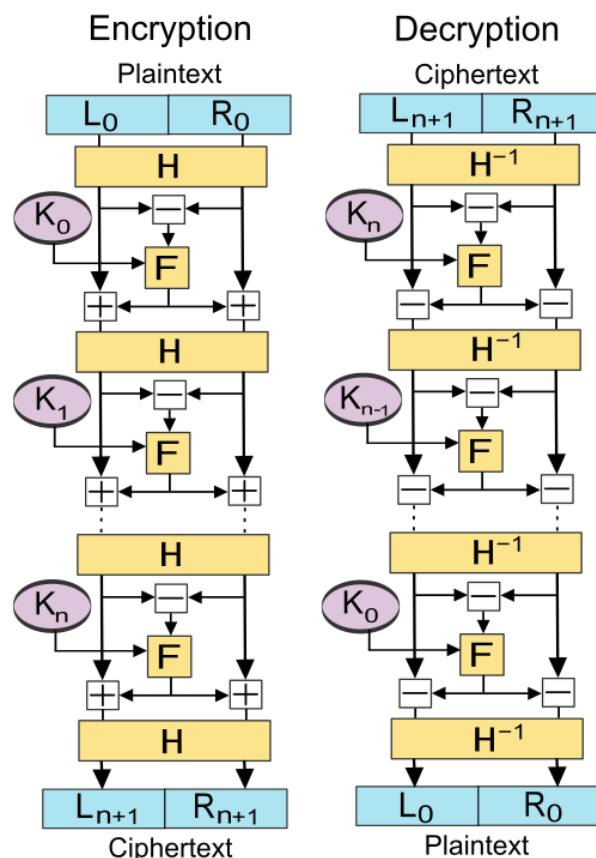
tcp/udp - transport layer - שכבה בה מועברים הנתונים הנתונים
tcp מאפשר העברה של מידע במוד יותר בטוח - תמיד יועבר כל המידע - כל התוואי ידוע מראש (יש מספור לכל פאקטה) - הוא יודע לזהות כל פאקטה שהולכת לאיבוד - משתמשים בזה היכן שאנחנו חייבים בקרת איכות - כמו מערכות בנקאיות

udp - עובד בשיטה של ווקי טוקי. צד א לא יודע אם צד ב קיבל את המידע. אין בדיקה ובקרת איכות. דוג' לשימוש בudp - צפיה ביוטיוב, נטפליקס וכו'.. (העברת מידע בrealtime).
פרוטוקול זה הוא הרבה יותר מהיר

הסיבות לבחור בסטרים או בלוקים:

1. סוג המידע - אם עובדים עם tcp - בוחרים בבלוקים, אם עובד עם udp עובדים בסטרים.
2. הסיבה לבחור בudp היא אם רוצים להעביר מידע במהירות בלי להבטיח שכולו מועבר (כאשר זה לא קריטי), בוחרים בtcp - כאשר יש חשיבות רבה לשמירה על המידע ומעקב אחריו שהכל עובר בצורה מלאה

round - כמה פעמים אני עושה סיבובים עם בלוק המידע. כדי להוסיף סיבוכיות
fiestal cipher structure - לוקחים מידע ומתחילים לשחק איתו - שימוש בrounds ובמספר מפתחות
בתהליך ההצפנה



+ - סימון של XOR

- מתואר בצירוף הצפנה של כל בלוק ובלוק
- בלוק ימין גם עובר נקי להמשך התהליך, וגם עובר תהליך של הצפנה, משתמשים בתוצאה של ההצפנה על מנת לבצע XOR על בלוק שמאל. בלוק שמאל שעובר הלאה להמשך תהליך ההצפנה מכיל גם את בלוק שמאל וגם את בלוק ימין המוצפן.
- ואז מבצעים הפיכה (round) בין הכיווני שני הבלוקים (הצרה), ועכשיו בלוק ימין יעבור XOR תוך שימוש בהצפנה של בלוק שמאל.
- בכל שלב בהצפנה משתמשים במפתח אחר מלבד ההיפוך בין הבלוקים שעוברים הצפנה

בכל אלגוריתם הצפנה כמות הפעמים שהתהליך הזה (פייסטל - עם הראונד) יקרה משתנה, אבל השאיפה היא לכמה שיותר פעמים.

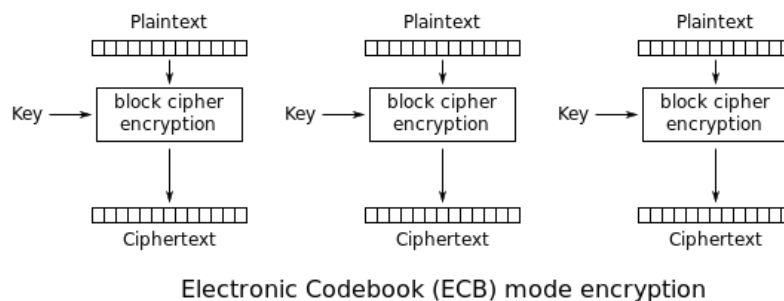
כדי לפענח את מה שקרה בתהליך ההצפנה - אנו נבצע את כל התהליך מההתחלה רק בצורה הפוכה (lifo - מבחינת מפתחות ההצפנה שלנו)

19/5/2020 - מתוקשב

שיטות הצפנה שונות ב**mode** של הבלוק (רוב המערכות היום עובדות בשיטה זו(בלוקים), אבל הבחירה בשיטת עבודה משתנה לפי צרכי כל מצב ומערכת ספציפית) סוגיה קריטית בנושא היא נושא של Initialization vector. המטרה של האלמנט הזה בתהליך ההצפנה הוא למנוע כל קשר סטטיסטי בין המפתח לכל השאר המרכיבים של ההצפנה. שההצפנה תהיה כמה שיותר רנדומלית.

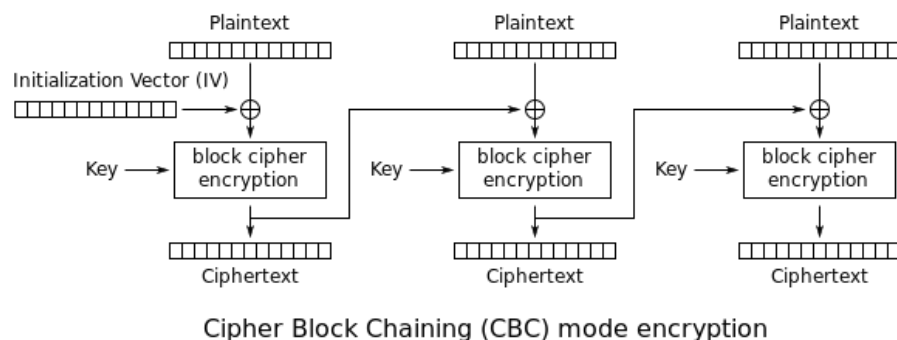
האם נכון לבצע הצפנה של כל קוד בנפרד? -- כמובן שלא. כיוון שאם כל בלוק מוצפן בעצמו, יתכן שתוקף שידע לפענח בלוק אחד ידע לפענח גם את שאר הבלוקים.

שיטות הצפנה בבלוק
ElectronicCodeBook - משתמש בשיטה כזו. הוא מצפין כל בלוק בנפרד:



בשיטת ECB ניתן יהיה לתוקף לראות חלק מהתמונה אם הוא יפענח חלק מהבלוקים. מצב לא תקין. יכול לתת לו אינדיקציה משמעותית על המידע שאנחנו מעבירים.

תצורה נוספת -
CipherBlockChaining - בשימוש הנרחב ביותר:

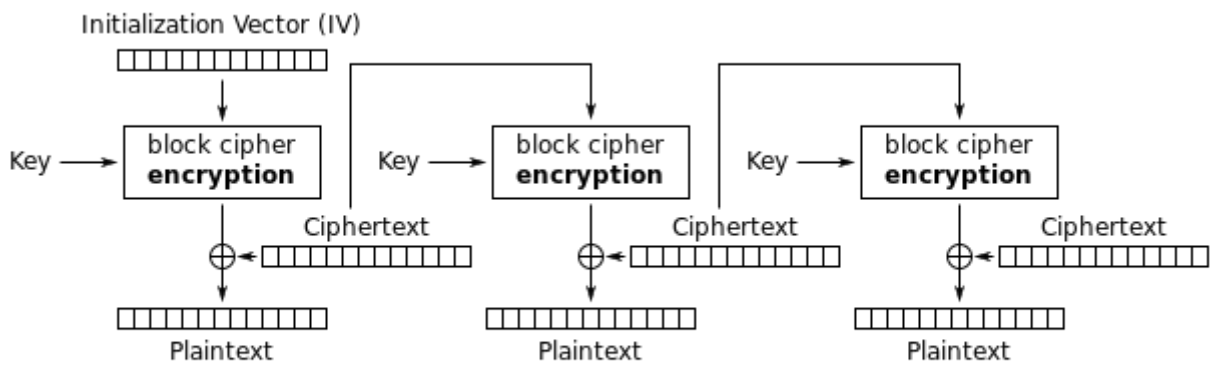


CBC בא לסבך יותר את התהליך ביחס למה שקורה בECB - שרשור של הבלוקים המוצפנים ה-initialization vector הוא רנדומלי לחלוטין (כדי שתוקף לא יהיה דרך לגלות מהו אותו וקטור התחלה)

התהליך הוא שמבצעים XOR (או פעולה אחרת) על ה-plaintext שלנו באמצעות ה- initialization vector ואז הבלוק עובר הצפנה עם מפתח ואז נוצר לנו ה-ciphertext, אנחנו משתמשים ב-ciphertext בבלוק הבא כ-initialization vector למעשה כדי להמשיך את התהליך באותו האופן עד אחרון הבלוקים. צורת ההצפנה הזו הרבה יותר בטוחה מההצפנה הקודמת מאחר ואם התוקף מצליח לגנוב בלוקים מסוימים מהמידע שלנו, ואת המפתח אפילו, עדיין יחסר לו ה-initialization vector מאחר והוא תלוי בבלוקים הקודמים.

תצורה 3 - CipherFeedBack mode

הקריפטו-אנליסטים שבדקו את צופן CBC ראו בו חולשה. ברגע שמשיגים את הבלוק הראשון, את המפתח, ואת ה-initialization vector כבר הכל פרוץ. ויש בעיה נוספת שבצורה הזו, יש קשר בין ה-plaintext שזה ה-ciphertext ל-input שהוא ה-output - למי שיהיה גדולה מהבלוקים המוצפנים באופן הזה ייתכן שיוכל להסיק מסקנות על מהו המפתח וה-initialization vector שלנו וכך להשיג גישה לכל המידע.

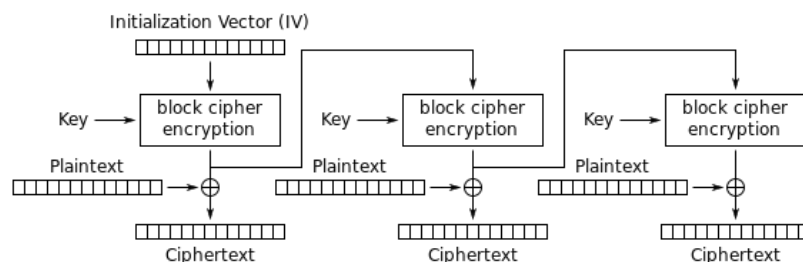


Cipher Feedback (CFB) mode decryption

ב-cfb - נעשה ניסיון לנתק עוד יותר את הקשר בין ה-plaintext ל-ciphertext - מה שעושים הוא תהליך דומה ל-CBC רק שמצפינים את ה-initialization vector באמצעות ה-key וכך יוצרים keystream שאיתו עושים לדוג' XOR על ה-Plaintext וכך עוד יותר מנתקים את הקשר בין מה שנכנס למה שיוצא מהצופן. כמובן התהליך לא מסתיים בפעם אחת של הצפנה של בלוק אלא, כמו ב-CBC המידע המוצפן מבלוק א' משמש Initialization vector לבלוק הבא בתור וכן הלאה..

תצורה 4 - OutputFeedBack Mode

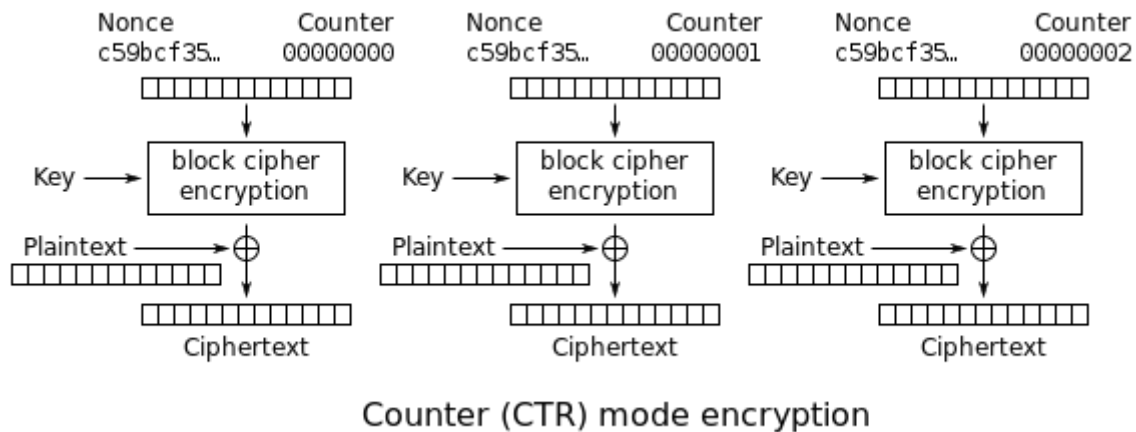
בתצורת הצפנה זו, מנתקים עוד יותר את הקשר בין ה-plaintext ל-ciphertext מאחר ובניגוד ל-CFB אנו לא משתמשים במידע המוצפן מבלוק A כ-initialization vector של הבלוק הבא, אלא משתמשים ב-keystream של בלוק א' כ-initialization vector של בלוק ב' וכן על זו הדרך עד הבלוק האחרון.



Output Feedback (OFB) mode encryption

תצורה 5 - Counter Mode

counter mode עוד יותר מסבך את העניין. עכשיו אין שימוש אפילו בתוצאת סטרים מבלוק א', אלא משתמשים ב initialization vector רנדומלי הגדל או קטן מבלוק לבלוק לפי חוקיות מסוימת שאפשר להגדיר מראש. כך עוד יותר מקטינים את הקשר בין ct ל pt - או יותר נכון מנתקים את הקשר. כעת צריך לדעת, על מנת לפצח את הצופן, גם את ה initialization vector גם את המפתח וגם את חוקיות בו ה initialization vector (למעשה ה-counter שלנו) משנה את ערכו.



26/5/2020 - מתוקשב

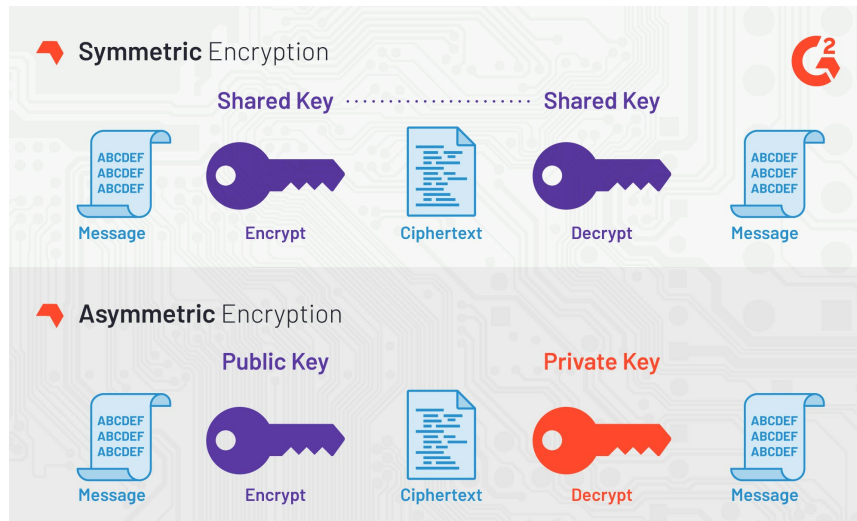
דוגמאות לסוגי הצפנה בבלוק

Data Encryption Standard (2des, 3des), Blowfish, Twofish, IDEA,
Advanced Encryption Standard (symmetrical)

- **check sum - CRC** - בדיקת תקינות. לבדוק הקובץ לא corrupted או שלא קיבלנו פקטה שגויה. למעשה בקרת איכות של העברת המידע. אין דרך לעשות את הבדיקה הזו בלי מנועים שהם שווים ערך למנועים שמצפינים, כי צריך להריץ אלגוריתם מסוים שיבדוק שכל המידע הגיע בצורה תקינה. ברמה הזו זה לא קשור בכלל לאבטחת מידע (ברמה פונקציונלית) - הרבה פעמים סוגיות של איכות העברת מידע יושפעו וישפיעו על נושאים של אבטחה. כי לדוג' אם חסר מידע כאשר מועברים נתונים הרבה פעמים ניתן להסיק שזה עניין של אבטחת מידע (זה יכול להיות בעיה תפעולית אבל זה גם יכול להיות מתקפה ועניין של אבטחה), לדוג' אולי אובדן מידע מצביע על מקרה של **ManInTheMiddle** הגונב לנו פקטות של מידע בזמן העברת הנתונים.. כל הבדיקות האלה נעשות באמצעות אלגוריתמי הצפנה.

מתוך סוגי ההצפנה בבלוק שהצגנו כאן למעלה (DES וכו'..) יש שתי קבוצות -

1. קבוצה של אלגוריתמיקה - נוסחה מתמטית שמבצעת את ההצפנה (אלגוריתמי הצפנה בד"כ האות האחרונה היא האות A - שמייצגת Algorithm)
 2. קבוצה של סטנדרט - מתייחס ל framework - מסגרת העבודה שלנו. האלגוריתם מוכל בתוך הסטנדרט. בעצם סטנדרט אומר מהם האילוצים שאנו נדרשים לעמוד בהם (כח עיבוד, גודל זכרון ועוד..) וכתוצאה מיזה נגזרת שיטת ההצפנה הרלוונטית עבורנו. האלגוריתם הוא נישא בתוך הסטנדרט (בסטנדרטים בד"כ האות האחרונה היא S, ומייצגת את המילה standard)
- עד היום חילקנו את הנושאים של הצפנה בין הצפנה בבלוק להצפנה בסטרים - וזה מתייחס לאופן הטיפול במידע (האם הוא מועבר כבלוקים או כסטרים רציף של מידע), בנוסף לאבחנה זו יש גם אבחנה ביחס לשיטת ההצפנה, החלוקה היא ל2:



אובייקטים בהצפנה הם השולח והמקבל - בד"כ יהיו לפחות 2 אובייקטים בכל תהליך של שילוח מידע מוצפן (וגם לא מוצפן כמובן..)

1. הצפנה סימטרית מול א-סימטרית

ב-קריפטוגרפיה, הצפנה סימטרית (symmetric encryption) או צופן סימטרי הוא **אלגוריתם** הצפנה שבו משתמשים במפתח **הצפנה** יחיד הן להצפנה של הטקסט הקריא והן לפענוח של הטקסט המוצפן. בפועל המפתח הוא בדרך כלל סוד משותף לשנים או יותר משתתפים ובדרך כלל מתאים לכמות מוגבלת של נתונים. הסיבה שהצופן נקרא סימטרי היא כי נדרש ידע שווה של חומר סודי (מפתח) משני הצדדים.
(נקרא session key).
מפתח 1

2. **הצפנה אסימטרית** (Asymmetric encryption), שבו מפתח ההצפנה שונה ממפתח הפענוח. לכל אובייקט בתהליך יש 2 מפתחות (מפתח פרטי ומפתח ציבורי) כך שבכל תהליך של הצפנה אסימטרית מעורבים בפועל 4 מפתחות. 2 לכל אובייקט 2 מפתחות (**public key and private key**) כפול 2 (או כפול כמות האנשים המקבלים את המידע ושולחים אותו)

הצפנה יכולה להיות בסטרים וא-סימטרית, או ההיפך או כל שילוב אחר. - כל השילובים אפשריים בעיקרון

הסבר נוסף על סימטרי מול א-סימטרי מסמסטר א:

הצפנה סימטרית- המפתח משמש גם להצפנה וגם לפענוח.

האתגר של הצפנה סימטרית היא - **key exchange**

ניתן לגנוב את המפתח ואז להצפנה אין משמעות

חוזקה - חוזק מול כח העיבוד שהיא צורכת. הצפנה מאוד חזקה וצורכת יחסית מעט משאבים

הצפנה א-סימטרית- 2 מפתחות.

לכל אובייקט(כל דבר שמתקשר - מחשבים, בנ"א וכו') יש מפתח ציבורי ומפתח פרטי.

אי אפשר להצפין ולפענח באותו מפתח. אלא במפתחות מנוגדים.

מפתח אישי - רק שלי, מפתח ציבורי - של כולם.

מפתח ציבורי משותף עם כולם (עם כל מי שאני מעוניין) - (אסור לשלח את המפתח הפרטי לעולם!!)

עם איזה מפתח להצפין? - עם המפתח הציבורי.

ועם מפתח של מי להצפין? - מפתח ציבורי של האובייקט אליו שולחים.

ואז רק האובייקט שמקבל יכול לפענח את ההצפנה עם המפתח הפרטי שלו.

חולשה - כדי להגיע להצפנה חזקה צריך המון כח עיבוד. ולכן עלות תועלת יוצאת נמוכה.

הגיעו לרעיון לעשות שילוב: בין סימטרי לא-סימטרי. (לקחת יתרונות של שניהם) שולחים את המידע עצמו בהצפנה סימטרית. ואת המפתח שהוא קטן יחסית, מצפינים עם הצפנה א-סימטרית.

כך נהנים מ2 העולמות.

באמצעות ההצפנות האלה שמרנו על הסודיות.

אבל איך יודעים שמי ששלח את הקובץ הוא באמת מי שאנחנו חושבים?

מצפינים בסימטרי את המידע

מצפינים בא-סימטרי את המפתח (עם הפאבליק של האובייקט שמקבל)

ואת זה עוטפים בהצפנה עם המפתח האישי שלי. האובייקט שמקבל יעשה פיענוח עם הפאבליק שלי.

מאחר ורק לי יש את המפתח הפרטי שלי, אז רק אני יכלתי להצפין את זה כך שהמפתח הציבורי שלי יוכל לפתוח אותו.

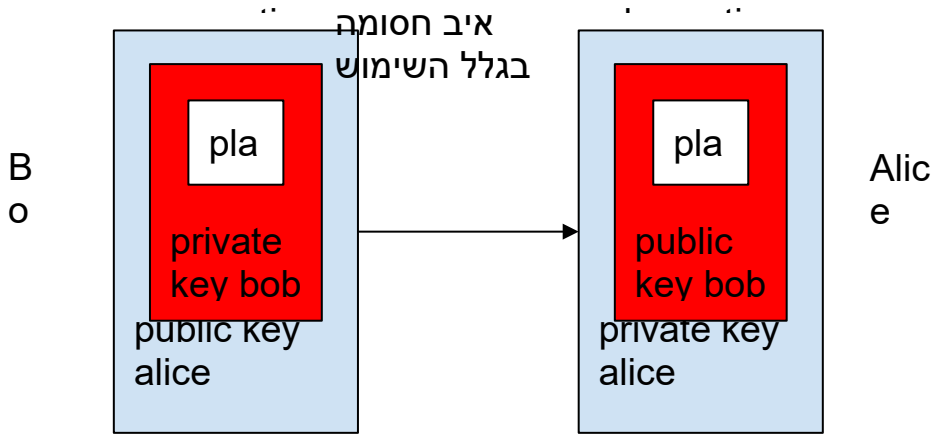
איך אפשר להבטיח שהאובייקט המקבל לא יוכל להכחיש שקיבל אותה?

משתמשים בבקורות מפצות (דברים שמעלים את הסיכוי שלא יוכל להכחיש) - כמו פרוטוקולי תקשורת מאובטחים. עוד דבר - ברגע הפענוח של המידע, זה יוצר אסמכתא (מקשרים לעוד תהליכים כדי להבטיח שזה קרה)

איך אני מבטיח את מהימנות המידע?

האשינג - וגם את ההאשינג צריך להצפין כדי שאפשר יהיה להאמין למידע שמתקבל

תהליך הצפנה א-סימטרי:



אליס של בוב, בעצם מאפשר ל private השימוש ב לדעת בודאות שבוב הוא זה ששלח את המידע כיוון שרק לו יש את המפתח הפרטי שלו (שלמעשה נפתח

תכונה	הצפנה סימטרית	הצפנה א-סימטרית
מפתחות	מפתח אחד. נוסחה לחישוב כמות המפתחות הנדרשים לכל שיחה:	2 מפתחות לכל אובייקט המשתתף בתהליך. $2 * n$

	$n(n-1)/2$	
חילופי מפתחות	זה הקושי הגדול שיש בהצפנה סימטרית - אופי העברת המפתח: באמצעים מאובטחים חיצוניים לתהליך העברת המידע עצמו	המפתח הציבורי פתוח לכולם והמפתח הפרטי נמצא רק בידי האובייקט שהוא הבעלים של אותו מפתח
מהירות	אלגוריתם פחות מסובך ומהיר יותר בצורה משמעותית. עם כח עיבוד קטן יותר אני יכולה להגיע להצפנה חזקה יותר	אלגוריתם מסובך ואיטי יותר - צורך יותר כוח עיבוד
שימוש	תשמש להצפנת מידע (קבצים) ותקשורת	תשמש להצפנה של מפתח ההצפנה הסימטרי וחתימות דיגיטליות
לטובת הצפנה היברידית [^]		
אבטחה המסופקת במסגרת השימוש באופן ההצפנה	להבטיח את סודיות המידע (כי זה משמש להצפנת המידע)	אותנטיקציה (authentication) - הזדהות (בגלל שיש התאמה בין הפרטי לציבורי) מניעת התכחשות (non repudiation) - אי אפשר להגיד שמידע לא נשלח או לא התקבל

n - מציין אובייקט אחד, או משתתף אחד בתהליך

2/6/2020 - מתוקשב

בשימוש במפתח בלבד, יש בעיה. יחסית קל לפרוץ את ההצפנה, ברגע שתוקף מזהה מהו המפתח כל תעבורת הנתונים שלנו פתוחה בפניו. לכן הכניסו סיבוכיות נוספת של אלמנט רנדומלי: initialization vector - מטרתו להכניס סיבוכיות לתהליך - לייצר מצב שלא יהיה לנו דפוס קבוע כלשהו שממנו התוקף יוכל לפענח את הצופן שלנו. מפתח אפשר לנחש בצורה מושכלת - זה משהו פגיע. את וקטור האתחול הזה אי אפשר לנחש, מאחר והוא בלתי תלוי במשתתפים בתהליך.

צורת מתקפה שמשמשת בפגיעות של שיטה המשתמשת במפתח בלבד: chosen plaintext attack - אם התוקף יודע להתחבר למערכת והוא יודע לדחוף לתוכה מידע גלוי ומהצד השני הוא יכול לתפוס את המידע המוצפן (ע"י סניפר). ולפי זה להתחיל לדעת בעצם מהו מפתח ההצפנה, וכך לפענח מידע אחר שנשלח באותו האופן (ע"י שימוש ומעקב אחר דפוסים שחוזרים על עצמם) - מסיבה הזו הוסיפו סיבוכיות נוספת לתהליך.

נעשה שימוש ב-2 פונקציות כדי לשבור את הקשר בין ה-input ל-output:

1. **diffusion** - מטרתו לשבור ולנטרל כל קשר סטטיסטי בין ה-input ל-output - ע"י הזזה של תווים לדוג' - כל פעם שמוכנס תו מסוים, מתקבל תו אחר - שזה לא יהיה עקבי
2. **confusion** - מטרתו לשבור ולנטרל כל קשר סטטיסטי בין ה-key ל-output (אין שימוש במפתח שחוזר על עצמו, גם אם יש סיטואציה שמפתחות משתנים, שלא יהיה לתוקף דרך לגלות איזשהו דפוס) - מכניסים אלמנט של רנדומליות למפתח (למשל שימוש ב-counter mode, initialization vector, keystream)

אם יש שימוש ב confusion בד"כ יהיה גם diffusion - (רק במקרי קיצון זה לא יהיה כך)
לעומת זאת יש מקרים של diffusion ללא confusion

סטנדרטים נפוצים להצפנה

- **DES** בתצורה של בלוק. Data Encryption Standard - גודל הבלוק הוא קבוע. 64 ביט.
השימושים בו הם מאוד כלליים.

משתמשים בו בראוטר הביתי, בסוויצ'ים, בראוטרם תעשייתיים, ברכיבי IOT גדולים יחסית (כי זה דורש כח עיבוד משמעותי יחסי)

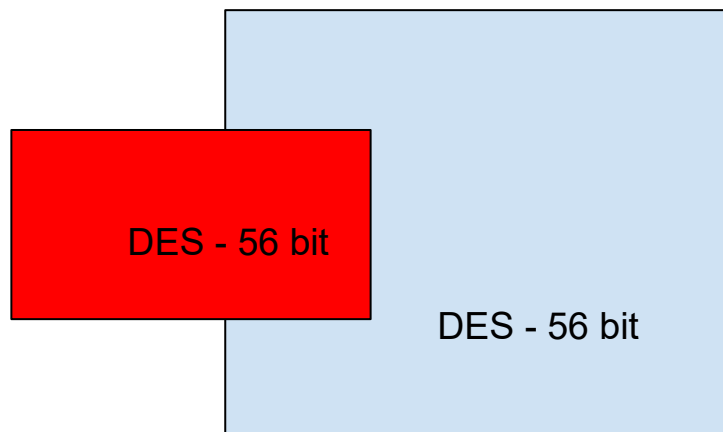
- **3DES** מסוף שנות ה-90 DES נהייתה לא אפקטיבית, כי היא מתבססת על גודל בלוק של 64 ביט שאינו משתנה ואורך המפתח בהתאם הוא 64 ביט - הבעיה נובעת מיזה שההצפנה בפועל היא לא 64 ביט אלא 56 ביט. 8 ביט הנותרים, הם פריטי (pariti). פריטי אלה (8 ביט) משמשים לניהול אינדקס של מפתח ההצפנה - הצד המקבל צריך את זה כדי לפענח את הצופן. DES היה מאוד פופולרי בשנות ה-90 מאחר וזה התאים לכח העיבוד שהיה אז.

~ אפקטיביות של הצפנה היא תמיד פונקציה של כח העיבוד שיש באותה התקופה ~

האיום המתקרב אלינו בכל מה שקשור *confidentiality* זה המחשב הקוונטי

היום 64 ביט זה הצפנה מאוד חלשה. ניתן לפרוץ את זה יחסית בקלות.

מאחר שהבינו את זה הקריפטו-אנליסטים, הם הציעו, בואו נעשה הצפנה של DES2 - דס שעטפו אותו בעוד DES, לכאורה מיזה קיבלתי הצפנה של 112 ביט (כביכול חזק). אך לא באמת, מאחר ומדובר בשיטת בצל של דס על גבי דס, DES אחד זה 56 ביט שאנחנו יודעים שהוא קל מאוד לפריצה, אז זה פשוט אמר שצריך פעמיים לפרוץ למנגנון של 56 ביט, קצת יותר ארוך אמנם, אך לא מגבלה משמעותית עם כח העיבוד שיש בימינו...

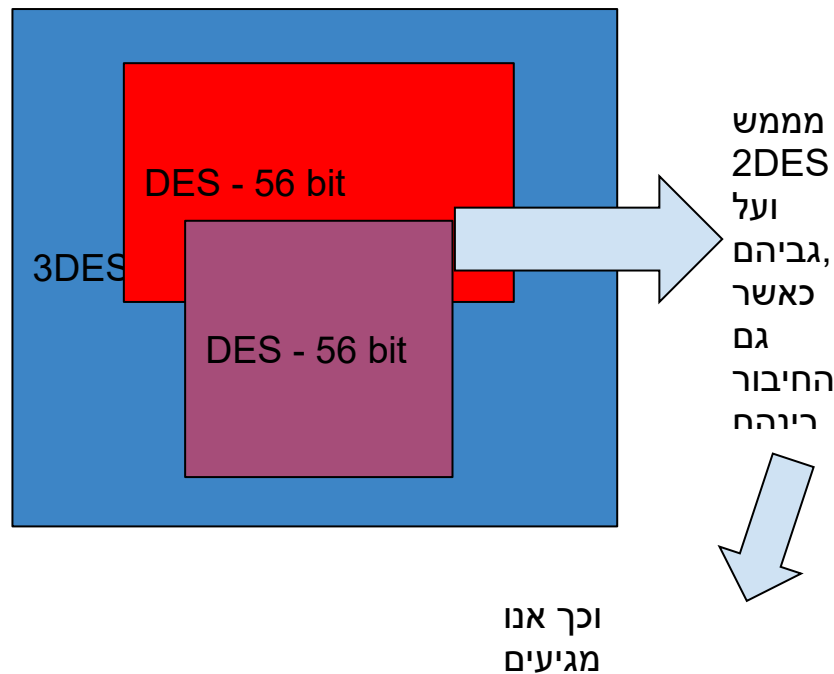


2DES Illustration

$$\text{Des in Des} - 2^{56} + 2^{56}$$

המתקפה שהרגה את DES2 (כבר לא משתמשים בזה היום) היא - **meet in the middle**
מתקפה שמנצלת את זה שבעצם זה פעמיים 2 בחזקת 56 (ניצול של הפריטי שמוריד את גודל ה key space שנותר לשימוש בהצפנה במקום 64, 56) - חלק מהפיצוח של שהבלוק הפנימי יותר ב-DES2 הוא שלאחר שמפצחים את הבלוק החיצוני, מקבלים את חלק מהמפתח שמאפשר לפצח את החלק הפנימי יותר

ואז החליטו לעשות את הצפנת DES3. למה DES3 שונה מ-DES2? כי היא מצליחה להגיע להצפנה של 112 ביט האמת ובימינו הצפנה של 100 ומשהו ביט היא הצפנה אפקטיבית (בהתאם לכח המחשוב). בעצם ב-DES3 מתרחשת עוד הצפנה העוטפת את השתיים הקודמות וכך היא גם מצפינה את קטע החיבור בין שתי ההצפנות הקודמות מה שיוצר את ההצפנה בחזקת 112 ביט encryption.



ה-DES השלישי בעצם מונע את הגישה לנקודת החיבור בין שני חלקי ה-DES2 על כל גווניו מממש את אלגוריתם ההצפנה - **International Data Encryption Algorithm** - **American Encryption Standard** סטנדרט ההצפנה הרווחת היום בעולם. גם בגלישה באינטרנט, גם בZIP. מאוד נפוצה ומאוד סטנדרטית. הצפנה בבלוק.

AES מממש אלגוריתם ההצפנה **Rijndael**. גוף התקינה האמריקאי בשילוב הDOD יצא בתחרות למתמטיקאים בעולם, למצוא אלגוריתם ההצפנה. עלו 3 "לגמר" - rijndael, twofish, blowfish - בסופו של דבר ניצח rijndael (שני מתמטיקאים - rijn, dael) שיטת rijndael עובדת בשיטה הבאה, לפי רכיבי הליבה הבאים:

- key - 128 או 192 או 256
- block - 128 ביט
- round - 10 או 12 או 14

המפתח שנבחר משפיע על הround בהתאמה.

Rivest Cipher 4 שיטת ההצפנה של סטרים. כל ביט שעובר הוא מוצפן. SSL, ו-WEP עושים שימוש בשיטת ההצפנה זו.

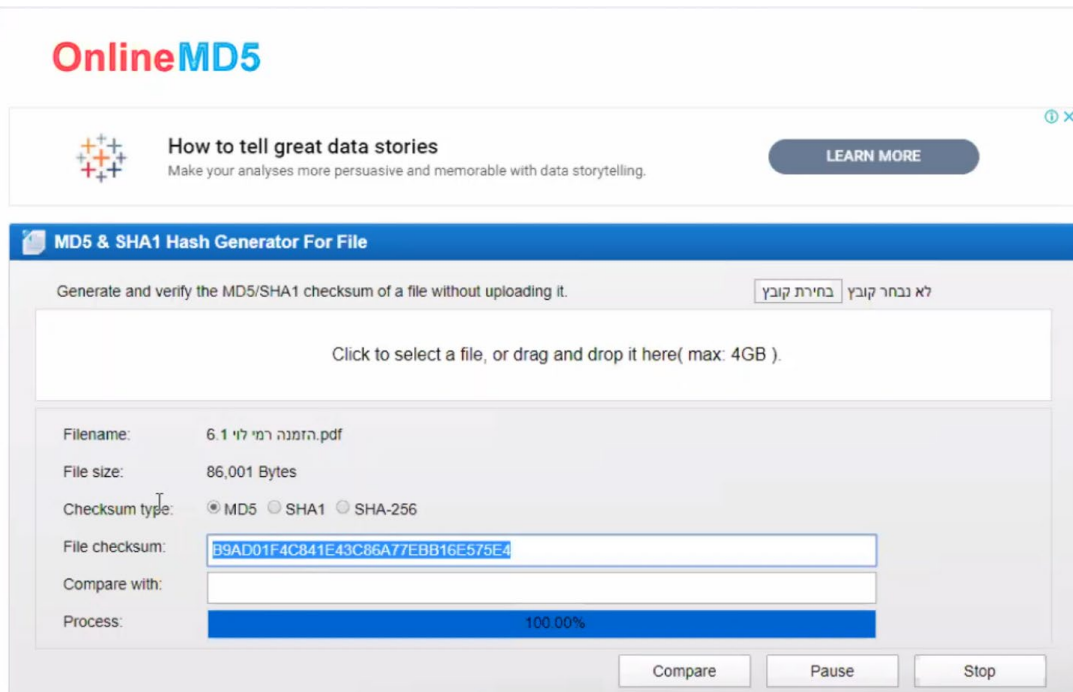
Wireless Encryption Protocol - משתמשים לטובת גלישה - רצוי שלא להשתמש - מדוע? ביישום של הפרוטוקול הזה, לא נתנו את הדעת על diffusion ו-confusion - הבעיה לא באלגוריתם אלא, באופן המימוש. אפשר להגיע עם לפטופ ואנטנה חזקה ולתפוס פקטות, ולזהות את כל הACCESS POINT שיש באיזור. לאחר הקלטת הפקטות (לוקח 10 דק' - רבע שעה). במידה ורוצים להגיע לרמת דיוק של 95% בתקיפה, צריך 85,000 פקטות (לדוג' סרטון שאנו רואים של דקה זה כבר 85,000 פקטות). לאחר מכן, מריצים סקריפט שמנתח את הסטטיסטיקות של הפקטות שנתפסו ומאחר ואין diffusion ו-confusion אין בעיה לפענח את הנתונים וכך לגלות את מפתח ההצפנה, וכעת הבנאדם חשוף בפני התוקף.

הצפנות **hash** - האש הוא אלגוריתם ההצפנה חד כיווני. הבטחת מהימנות של המידע - integrity. (אקסה בייט = טרה ** הערת אגב) האשים עובדים עם גודל קבוע של פלט. גם אם אתה מזין לאלגוריתם קובץ של טרה בייט או אם תכניס קובץ של מגה בייט, תמיד תקבל גודל פלט קבוע.

מעבר לכך, אותו קובץ תמיד יפיק את אותו הפלט, לכן זה אמצעי בדיקה טוב לוודא שקובץ שהגיע לידיך זהה למה שנשלח אליך.

הבעיה עם האשים הוא בקונספט שיש לכל קובץ ת.ז. ייחודית, ערך ייחודי שיצא עבור אותו קובץ - ניתן לוודא שהערכים ייחודיים לכל קובץ רק ככל שאתה מגדיל את גודל key space שלך.. כי ככל שה key space יותר גדול הסיכוי שמישהו יוכל לשחזר את אותו ערך עם קובץ אחר קטן משמעותית. לכן ההמלצה המינימלית היום להאשים היא של 128 ביט.

hash - **MessageDigest5** - פלט 128 ביט תמיד → ההמלצה היא להשתמש ב128 ביט ומעלה מבחינת אלגוריתמי האש



birthday attack - מה הסיכוי שניקח קובץ אחד לגיטימי וקובץ אחד זדוני, ונקבל עבור כל אחד מהם את אותו האש? עם 128 ביט זה המון שילובים, אבל עדיין יכול להיות מצב בו לשני קבצים יש את אותה הת.ז. ואז אפשר כך להעביר קובץ X במקום קובץ Y והמקבל מעולם לא ידע את ההבדל, או אם הוא ידע זה יהיה מאוחר מדי... (יש דרכים להגדיל את הסיכוי "לפגוע" באותה ת.ז.)
באו ובדקו את זה קריפטו אנליסיס השונים - מה עושים במצב כזה?
מה מגדיר את הסיכויים של התקפה מסוג זה להצליח?

1. כמות פריסת מחשבים בעולם

2. כח עיבוד

בואו נגדיל את כמות הפלטים שאלגוריתם ההאש מסוגל לייצר. פעם היה 4MD עד שהוא deprecated, עברו ל5MD וגם הוא לאט לאט מתחיל לאבד מהיוקרה שלו.. ואז נכנסו:

hash - **SecureHashAlgorithm-1** - פלט 160 ביט

SHA-256 - עוד יותר גדול עם key space של 256 ביט

SHA-512 - פלט 512 ביט

ככל שגודל הפלט גדל, הסיכוי של מתקפת birthday להצליח קטן.

א-סימטרי

האלגוריתמים הבאים הם א-סימטריים, שמטרתם, להבטיח יכולת של העברת מידע במקרה בו היעד והמקור נמצאים לא בסמיכות אחד לשני.

בהצפנה הסימטרית הוטיקה, הבעיה היתה בהעברת המפתח. אם המפתח היה מועבר עם ההודעה, ברגע שהיו עושים intercept להודעה, היה נתפס גם המפתח ואז כל התכתובת היתה שקופה עבור התוקף. מצב שחייבים להימנע ממנו בכל מחיר. (תמיד התוקף קודם כל רוצה את המפתח) מקרה לדוג' שממחיש את זה הוא התקפות כופר ישנות - הם לא לקחו בחשבון את אופן מימוש ההצפנה. ההצפנה היתה הצפנה סימטרית, לאחר שהתוקף מצפין את הקובץ הוא מקבל בחזרה את המפתח בצורה לא מוצפנת מהמחשב המוצפן (הם לא הבינו את הבעיה בשליחת מפתח ההצפנה באותו תדר תקשורת של העברת המידע עצמו). מה שהיה קורה כיוון שהפיירוול של החברה היה מאבטח גם את התנועה היוצאת, אז המפתח שהיה אמור להיות סודי עבור התוקף, נתפס בחתכו של הפיירוול (המקליט את התשדורות) ואז בעצם המפתח הגיע לידי המותקף, והוא כמובן לא הסכים לשלם את הכופר. ההתקפה לא היתה אפקטיבית.

ואז הם שידרגו את המתקפה שלהם ועברו למתקפה עם הצפנה א-סימטרית. הם סיפקו את **Private** שלהם למותקף רק לאחר קבלת התשלום

אין הצפנה של מידע עם הצפנה א-סימטרית, אלא הצפנה של המפתח הסימטרי עם הצפנה א-סימטרית:
Diffie-Hillman - מתעסק בסוגיות העברת המפתח
RSA
PGP

כיוון שהצפנה א-סימטרית היא איטית וכבדה - קשה מאוד להצפין איתה בלוקים של מידע כיוון שזה מאוד יקשה ויאט על העברת המידע.

RC2 - עוד יישום של הצפנה סימטרית
Rijndael - אלגוריתם הצפנה שרץ בסטנדרט שנקרא AES
RSA - הצפנה א-סימטרית, שמתעסקת עם הצפנה תוך שימוש במספרים ראשוניים
DSA - הצפנה א-סימטרית

DES - אורך 56 ביט. הומצא ע"י IBM אומץ ע"י NSA עד שנות ה-90. כבר פחות עושים בו שימוש
3DES - עושים שימוש בהצפנה של 112 ביט (ההצפנה האפקטיבית). לכאורה אורך המפתח הוא 3*56 - היינו 168, אבל אנחנו יודעים שבפועל זה לא 168, כי אופי ההצפנה של הDES2 הפנימי הוא של 56 ביט + 56 ביט, כאשר הפתרון של אחד מהDESים מונע מהפתרון של הDES הקודם. היינו ההצפנה הפנימית פשוטה יותר וניתן לנצל אותה למתקפה שנקראת *meet in the middle*
rijndael - ההצפנה הנפוצה ביותר בעולם היום. נעשה שימוש באלגוריתם זה בתוך סטנדרט הצפנה AES

היתה תחרות של משרד ההגנה האמריקאי לאלגוריתמי הצפנה, היו שלושה שהגיעו לשיקול הסופי, אחד מהם היה rijndaelblowfish היה זה שזכה לבסוף.
על כל אחת מההצפנות יש לנו:

1. round

2. אורך מפתח

3. גודל הבלוק

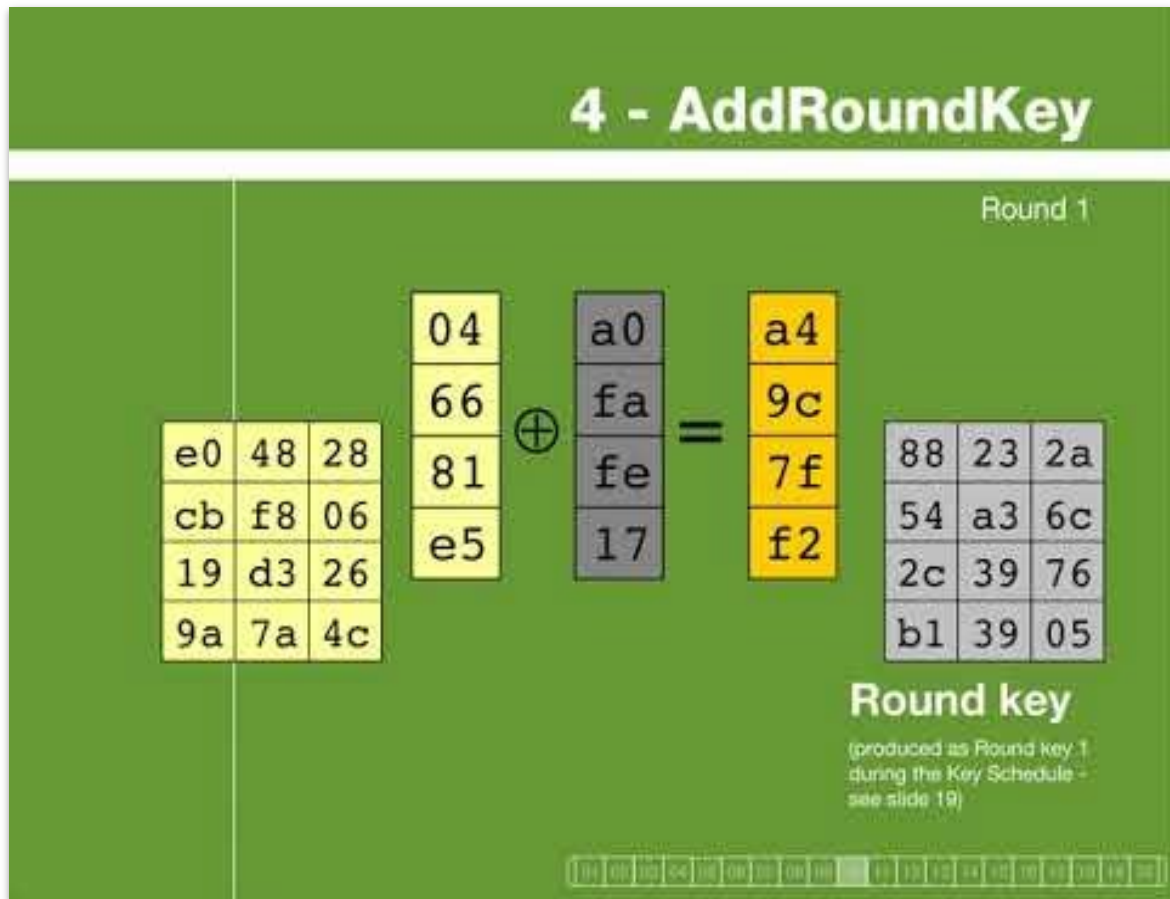
גודל הבלוק בrijndael הוא 128 ביט. גודל המפתח משתנה בין 128, ל192, ל256. הראונדים (כל ראונד הוא עוד מורכבות בתהליך) במקרה של rijndael הם כנגזרת של המפתח:

1. מפתח הצפנה של 128, יבוצעו 10 ראונדים

2. מפתח הצפנה של 192 יבוצעו 12 ראונדים

3. מפתח הצפנה של 256, יבוצעו 14 ראונדים

להנפשה על צופן *rijndael* יש ללחוץ על התמונה הבאה:



- כמות הראונדים הנספרים מתחילה מ0. כמו במערך. אז על מפתח של 128 ביט עבורו מבוצעים, כאמור, 10 ראונדים, כאשר הם נספרים בין 0 ל - 9.
- blowfish** - מתייחס ל128 ביט הצפנה. (זה אלגוריתם, לא סטנדרט). גודל הבלוק הוא קבוע ולא ניתן לשינוי - 64 ביט. אורך המפתח בין 32 ל448 ביט.
- idea**-הצפנה של 128 ביט. מגובה בפטנט (לא open source).
- צופן נחשב כאמצעי לחימה. מבחינת ההתנהלות שקשורה אליו. צריך אישורים למשל כדי למכור צפנים לגורמים זרים. כדי שלא יהיה אפשרי להשתמש בזה נגדנו..
- יש אפילו במדינה, חוק שנקרא חוק הצופן.

אין הצפנה בלתי שבירה, יש הצפנה שלזמנה היא טובה. ז"א שצריך לבחור שיטת הצפנה שהיא תהיה טובה לאורך זמן.

Factors	DES	3DES	AES	E-DES
Key length (bits)	56	112, 168	128, 192 or 256	1024
Cipher Type	Sym.	Sym.	Sym.	Sym.
Block (bits)	64	64	128	128
Rounds	16	48	10,12,14	16
Developed	1975	1978	1998	2013
Security	Not good	Passing	Secure	Secure
Possible Key	2^{56}	2^{112}	2^{128}	2^{1024}
Time for Brute Force key attack (10^{13} keys/sec)	<1 day	1.6×10^{14} years	10^{19} years	10^{152} years
Avalanche effect	Resists	Resists	Resists	Resists
Encryption Software	Fast	Slow	Medium	Very fast
Time to encrypt 48KB	7s	21s	13s	2s

הסטנדרט המומלץ היום הוא AES.

על הצפנה חדשה לוקח זמן עד שסומכים עליה. עורכים על זה מחקרים. זה לוקח זמן. רק לאחר הבדיקות האלה יש רמת מהימנות טובה לאותו צופן.

יש איזון בין כח המחשוב ובין רמת המהימנות של הצופן האולטימטיבי, כביכול.

מדינות יפתחו הצפנות משלהם. לא ישתמשו בהצפנות ציבוריות, או שישלבו הצפנות ציבוריות עם הצפנות נוספות שלהם. אותם מדינות לא יפרסמו את הצפנים שלהם אף פעם.

חברות עסקיות ישתמשו בסטנדרטים פתוחים לעולם - או בתשלום או קוד פתוח.

מטרה של הצפנה היא להתיש את הצד התוקף. להפוך את זה למשהו לא כלכלי עבור התוקף.

נושאים בהם הצפנה מטפלת:

1. פרטיות (נגזרת של סודיות) - רק מי שרשאי נחשף למידע.
2. authenticity - זהות והזדהות. אין יכולת לזהות את הבנאדם אחרת באינטרנט
3. integrity - שאנחנו נדע שהמידע ששלחתי או קיבלתי הוא בדיוק מה שהיתה הכוונה במקור
4. non-repudiation - אף אחד לא יכול לטעון שלא קיבל תשלום לדוג' אם אכן הועבר אליו התשלום

רנדומליות

גם אם יש לתוקפים מידע מוקדם, זה לא יספיק להם כדי לפרוץ את הקוד שלנו כי אנחנו משתמשים בערכים רנדומליות. כמו initialization vector
 salt ו seed - דומים ל initialization vector. השוני הוא שהם תמיד לא יהיו חשופים, בIV הערך יכול להיות חשוף, ויותר מיזה יכול להיות שאתה תייצר את הערך ההתחלתי הזה.
 סולט(בהצפנות כבדות, הצפנה ברמת פונקציה ייחודית שמג'נרט כל הזמן ערך רנדומלי), סיד (דברים כמו הצפנה של השלט של המפתח לרכב, של התשדורת בין הרכב לשלט, איזשהו ערך מוצפן שהשלט שולח באופן קבוע לאוטו והוא יודע לפענח אותו. רק כך השלט יעבוד והרכב יתניע)