

Project Book  
Storing passwords securely

# Password Manager

Password Manager provides an efficient solution  
for securely managing your passwords.  
Learn how to protect your privacy and stay  
safe in our digital world.



Tamar Avitan

Project Book

# Password Manager

Storing passwords securely

סמל מוסד: 711903

שם מכללה: דרכי חנה



לוגו המכללה

שם המנחה: המורה רחל טוג

שם הסטודנט: תמר אביטן

ת.ז. סטודנט: 327822664

תאריך ההגשה - 2025 יולי

## תוכן עניינים

|         |  |
|---------|--|
| 4.....  | הצעת פרויקט.....   |
| 16..... | מבוא.....  |
| 16..... | הרקע לפרויקט.....  |
| 16..... | תהליך המחקר.....   |
| 17..... | סקירת ספרות.....   |
| 17..... | אתגרים מרכזיים.....  |
| 17..... | הבעיה איתה התמודדתי.....   |
| 17..... | הסיבות לבחירת הנושא.....   |
| 18..... | מוטיבציה לעבודה.....   |
| 18..... | על איזה צורך הפרויקט עונה.....                                     |
| 18..... | פתרון: מערכת ניהול סיסמאות מאובטחת וחכמה.....                      |
| 18..... | הצגת פתרונות לבעיה.....  |
| 18..... | 1. פתרונות מבוססי ענן.....   |
| 18..... | 2. פתרונות מקומיים (Offline).....                                  |
| 19..... | מטרות / יעדים.....   |
| 19..... | מדדי הצלחה למערכת.....   |
| 19..... | אתגרים.....  |
| 20..... | רקע תיאורטי.....   |
| 26..... | תיאור מצב קיים.....  |
| 26..... | ניתוח חלופות מערכת.....  |
| 26..... | DES.....   |
| 28..... | הצפנת אל גמאל (ElGamal encryption).....                            |
| 29..... | אלגוריתם מבוסס תורת הגרפים וצופן בלוקים:.....                      |
| 29..... | שפת פיתוח וסביבת עבודה:.....                                       |
| 30..... | תיאור החלופה הנבחרת.....   |
| 31..... | שלבי יצירת מפתח משני Ki בזמן הצפנה:.....                           |
| 31..... | יצירת מפתחות מחדש בזמן פענוח:.....                                 |
| 32..... | שלב הקדם-הצפנה:.....   |
| 32..... | צופן הבלוקים:.....   |
| 33..... | ייחודיות האלגוריתם.....  |
| 33..... | אפיון המערכת.....  |
| 33..... | ניתוח דרישות המערכת.....   |
| 33..... | מודול המערכת.....  |
| 33..... | אפיון פונקציונלי.....  |
| 34..... | ביצועים עיקריים.....   |
| 34..... | אילוצים.....   |
| 34..... | תיאור הארכיטקטורה.....   |
| 34..... | הארכיטקטורה של הפתרון המוצע בפורמט של : Design level Down-Top..... |
| 35..... | תיאור הרכיבים בפתרון (שרתי DB, שרתי תקשורת, שרת יישום/ לקוח).....  |
| 36..... | MongoDB.....   |
| 39..... | תיאור פרטוקולי תקשורת.....   |
| 39..... | שרת לקוח.....  |

|    |                                   |
|----|-----------------------------------|
| 39 | תיאור תהליכי אבטחת מידע במערכת    |
| 40 | תיאור הצפנות                      |
| 40 | ניתוח ותרשים UML/ useCases למערכת |
| 40 | Use case Diagram                  |
| 40 | Use cases רשימת                   |
| 42 | יציאה מהמערכת:                    |
| 42 | מבנה נתונים                       |
| 42 | חישוב יעילות האלגוריתם            |
| 43 | הקשרים בין היחידות השונות         |
| 43 | עץ מודולים                        |
| 43 | תרשים UML                         |
| 44 | dal layer:                        |
| 45 | design class diagram              |
| 45 | תיאור המחלקות המוצעות             |
| 46 | קלט ופלט של היחידות               |
| 46 | רכיבי ממשק                        |
| 47 | תיכון המערכת                      |
| 47 | ארכיטקטורת המערכת                 |
| 47 | תיכון מפורט                       |
| 47 | חלופות לתיכון המערכת              |
| 47 | תיאור התוכנה                      |
| 47 | סביבת עבודה                       |
| 47 | שפות תכנות:                       |
| 48 | תיאור מסכים                       |
| 48 | תרשים מסכים ((diagram flow Screen |
| 48 | תיאור מסך הפתיחה-                 |
| 49 | מסכי האפליקציה:                   |
| 52 | ממשק משתמש                        |
| 52 | קוד התוכנית                       |
| 55 | תיאור מסד הנתונים                 |
| 56 | תרשימי ERD                        |
| 56 | מדריך למשתמש                      |
| 56 | בדיקות והערכה                     |
| 57 | ניתוח יעילות                      |
| 57 | אבטחת מידע                        |
| 58 | מסקנות                            |
| 58 | פיתוחים עתידיים                   |
| 59 | ביבליוגרפיה                       |

## הצעת פרויקט

### תיאור הפרויקט

אנשים רבים מתקשים לשמור על הסיסמאות שלהם בצורה בטוחה ושמורה מה שעלול להביא לגילוי הסיסמאות שלהם. אנשים גם בוחרים סיסמאות פשוטות שקל לנחש אותן וגם זה מסכן את פרטיות החשבון שלהם. בפרויקט זה אני שואפת להגיע לאחסון הסיסמאות בצורה שמורה ומאובטחת וכן להציע למשתמש סיסמא חזקה שתקשה על האקרים לגלות אותה. לפרויקט זה נקרא SecureVault.

בעידן הדיגיטלי המודרני, שבו כמעט לכל פעולה יומיומית יש היבט מקוון, המשתמשים נדרשים להיכנס למגוון אתרים ושירותים. כל אחד מהם מצריך יצירת שם משתמש וסיסמה על מנת להבטיח את הזיהוי והגישה הפרטית. בממוצע, אדם רשום לכ-70 אתרים ושירותים דיגיטליים, ורבים מהם דורשים סיסמאות שונות כדי לשמור על רמת אבטחה גבוהה. אלא שהמציאות היא שלא ניתן לזכור את כל הסיסמאות הללו, ולכן משתמשים רבים פונים לבחירת סיסמאות פשוטות וקלות כמו "1234", תאריך הלידה שלהם, או אפילו מספרי טלפון אישיים.

הבעיה המרכזית בכך היא שהסיסמאות הללו קלות מאוד לניחוש, והן מציבות סיכון ממשי לאבטחת המידע של המשתמש. כפי שמחקרים רבים הראו, סיסמאות חלשות, או כאלו המשתמשות במידע אישי שניתן לאסוף בקלות, פותחות פתח רחב עבור תוקפים לבצע התקפות כגון גניבת זהות, הונאות פיננסיות או חדירה לחשבונות אישיים, שיכולה להוביל לנזק כבד.

רשימת 10 הסיסמאות הכי משומשות בקרב האקרים



ההשוואה הזו מראה שבאופן כללי, הסיסמאות הכי פחות מאובטחות מכל מדינות ואוכלוסיות העולם הן "123456" ו-"12345678" - שתיים מהתבניות המספריות המובנות מאליהן והקלות ביותר לניחוש שמתאימות למגבלת אורך סיסמה של 6-8 תווים שרוב אתרי הרשת מחייבים בהתאם לכך, עולה הצורך מנהל סיסמאות, כלי שמרכז ומאחסן את כל הסיסמאות בצורה מאובטחת ומאפשר למשתמשים גישה מהירה ונוחה לחשבונותיהם מבלי להסתמך על זכירת כל סיסמה

איך עובד מנהל סיסמאות?

בפעם הראשונה שמשתמש מבקר באתר הדורש שם משתמש וסיסמה תוך שימוש במנהל סיסמאות, יכולות להתרחש תוצאות שונות. אם המשתמש לא יצר בעבר שם משתמש וסיסמה לאתר, מנהל הסיסמאות יכול לסייע ביצירת סיסמה אקראית וייחודית ביותר. כאשר המשתמש שם את הסמן בשדה הקלט עבור הסיסמה, מנהל הסיסמאות מבקש מהמשתמש ליצור סיסמה חדשה וחזקה. לאחר הזנת שם המשתמש והסיסמה החדשה, מנהל הסיסמאות בדרך כלל מבקש מהמשתמש לשמור את המידע. שם המשתמש והסיסמה מאוחסנים בצורה מאובטחת במנהל הסיסמאות. בפעם הבאה שהמשתמש מבקר באותו אתר, מנהל הסיסמאות פותח חלון הנחיה, בדרך כלל מעל המקום שבו נדרש קלט המשתמש, ושואל אם המשתמש רוצה להזין את המידע שנשמר קודם לכן. מצד שני, כאשר למשתמש כבר יש שם משתמש וסיסמה אך מבקר באתר בפעם הראשונה עם מנהל סיסמאות מותקן, הוא מבקש מהמשתמש לשמור פרטי חשבון לביקורים עתידיים..

### כיצד מנהל סיסמאות מזהה אם יש צורך בסיסמה?

אתרי אינטרנט רבים משתמשים בטופס סטנדרטי של HTML כדי לאסוף נתונים כמו שם משתמש וסיסמה. בטופס זה, השדות המיועדים להזנת שם המשתמש והסיסמה מקודדים בעזרת תגיות `<input type="text">` עבור שם המשתמש ו-`"password">` עבור הסיסמה. טכנולוגיות ניהול סיסמאות מזהות שדות אלו באופן אוטומטי, באמצעות אלגוריתמים המתמקדים באותם תגים ובמאפיינים כמו שם השדה (id, name) או אפילו מיקומו בדף. כאשר מנהל סיסמאות מזהה את השדות המתאימים, הוא בודק את כתובת האתר (URL) שבה המשתמש נמצא וממפה אותה לרשימה של אתרים מוכרים עם אישורים (credentials) שמאוחסנים בזיכרון. אם מנהל הסיסמאות מזהה את האתר כרשום במאגר שלו, הוא אוטומטית משווה את השם והסיסמה שמאוחסנים עבורו, ומזין אותם לשדות הרלוונטיים מבלי שהמשתמש יצטרך להזין את הסיסמה מחדש. במקרים שבהם האתר לא קיים ברשימה של מנהל הסיסמאות, או אם פרטי הגישה לא נמצאים במאגר, ידרוש המנהל מהמשתמש להזין סיסמה חדשה או להוסיף את האתר לרשימת האישורים. איך זה קורה בפועל?

שימוש באישורים (credentials) קורה בצורה של אוטומציה ושלב אחר שלב, במטרה לאמת את זהות המשתמש ולהבטיח גישה למערכת בצורה מאובטחת. להלן הצעדים הפשוטים שכוללים את השימוש באישורים בפועל:

#### **1. יצירת אישורים:**

כאשר משתמש נרשם לאתר או לשירות, הוא יזין את שם המשתמש והסיסמה שלו, שמאוחסנים במערכת או בשרת בצורה מוצפנת. לעיתים ייתכן שמערכת תדרוש מידע נוסף, כגון כתובת דוא"ל או תעודת זהות דיגיטלית.

#### **2. אימות משתמש:**

כאשר המשתמש מנסה להיכנס לאתר או לשירות, הוא יזין את שם המשתמש והסיסמה. המערכת משווה את האישורים שהוזנו לאלו ששמורים בבסיס הנתונים שלה. אם המידע תואם, המשתמש מזהה ומורשה לגשת לשירות. אם נדרש, המערכת יכולה להוסיף שלב נוסף של אימות דו-שלבי (2FA) בשלב זה, המשתמש יקבל קוד חד-פעמי לטלפון הנייד או לאימייל, שעליו להזין כדי לאמת את זהותו.

#### **3. העברת האישורים בצורה מאובטחת**

כאשר למשתמש יש אישורים, הם מועברים בין הלקוח (למשל דפדפן אינטרנט) לשרת באמצעות הצפנה. המידע נשלח על גבי פרוטוקול מאובטח כמו HTTPS כדי לוודא שאף אדם אחר לא יכול לגשת אליו או לשנות אותו במהלך הדרך. במקרה של מנהל סיסמאות, האישורים נשמרים בצורה מוצפנת במאגר מאובטח, ובזמן שהמשתמש מבצע את ההתחברות, המנהל מוציא את האישורים המתאימים (למשל את הסיסמה) וממלא את השדות בצורה אוטומטית.

#### **4. ניהול ואחסון אישורים**

אישורים יכולים להתעדכן במערכת אם המשתמש משנה את הסיסמה או הוסיפו אפשרויות אבטחה נוספות. המידע שמאוחסן בכל הנוגע לאישורים, כולל שמות משתמש, סיסמאות ותעודות דיגיטליות, נשמר בצורה מאובטחת ומוצפנת כדי למנוע גישה לא מורשית. לדוגמה, מנהלי סיסמאות כמו 1Password או LastPass מצפינים את הנתונים כך שגם הם לא יכולים לגשת אליהם ללא מפתח הצפנה (Master Password) של המשתמש. . זיהוי אתרים ודפים:

כשמשתמש פותח דף אינטרנט שבו קיימת בקשה להזין אישורים (למשל את שם המשתמש והסיסמה), מנהל הסיסמאות מזהה את השדות הרלוונטיים לפי התגים של HTML כמו `name="password"` ו `name="username"`, ומשווה את כתובת האתר שהמשתמש נמצא בו לרשימת האישורים המאוחסנים. אם המערכת מזהה שהמשתמש רשום באתר הזה ומאוחסן עבורו אישור (למשל סיסמה), היא תמלא אוטומטית את השדות עם האישורים המתאימים.

**דוגמה לשימוש בפועל:**

1. משתמש נכנס לאתר כלשהו
2. מנהל הסיסמאות מזהה את כתובת האתר ומשווה אותה לרשימת האישורים המאוחסנים.
3. הוא מוצא את שם המשתמש והסיסמה המתאימים, וממלא אוטומטית את השדות בטופס התחברות.

### איפה מותקן מנהל סיסמאות?

ניתן להתקין במכשיר האישי וניתן לשמור את הנתונים בענן. לכל אחת מהאפשרויות יש יתרונות וחסרונות.

#### -בענן:

מנהלי סיסמאות בענן הם הפופולריים ביותר כיום. עצם העובדה שהם מאחסנים את הסיסמאות בענן מאפשרת גישה אליהן מכל מכשיר בעולם. רוב המותגים הפופולריים מציעים אפליקציות גם לטלפון הנייד, כך שאם המשתמש מתעסק עם כמה מכשירים (כמו רובנו), השירותים מבוססי הענן יסנכרו את הסיסמאות בין כל המכשירים. חלק מהשירותים מציעים גם תוכנות למחשב ותוספים לדפדפנים, מה שמכסה את כל האפשרויות לשימוש נוח. עם זאת, חשוב לזכור שהנתונים המאוחסנים על שרתים מציבים סיכון. אם תוקפים מצליחים לפרוץ לשרת, הם עלולים להוריד את כל הנתונים המאוחסנים בו. בבת אחת, מה שעלול להעמיד את פרטי הגישה לחשבונות שלכם בסיכון גבוה.

#### -אחסון מקומי:

מנהלי הסיסמאות המקומיים (שמותקנים במכשיר עצמו) דווקא הופכים אותם לבטוחים יותר. הקודים מאוחסנים על מכשיר ספציפי, ולכן לא תוכלו לגשת אליהם מכל אחד מהמכשירים שלכם; אך האקר יצטרך לתקוף ספציפית אתכם כדי להוציא את הקודים האלה, מה שהופך את המשימה הזאת לקשה יותר עבורם.

מצד שני, צריך לזכור שאם מאבדים את המכשיר, או אם הוא מתקלקל, המשתמש עשוי לאבד גישה לכל הסיסמאות שאוכסנו בו.

בפרייקט זה נעדיף להתקין את מנהל הסיסמאות על המכשיר עצמו.

כך שהסיסמאות יאוחסנו באופן מקומי ולא יועברו לאחסון בענן. פתרון זה מצמצם את החשיפה לפריצות פוטנציאליות דרך האינטרנט. כלומר, אין סיכון של גישה מרחוק לנתונים האישיים באמצעות תקיפות על שרתים מרוחקים או עננים שבהם מאוחסנות הסיסמאות, מה שמגביר את רמת האבטחה.

### הגדרת הבעיה האלגוריתמית

#### 1. חסינות מפני התקפות:

ההצפנה צריכה להיות חזקה דיה כדי להקשות על פיצוח, גם עבור סיסמאות נפוצות או חוזרות. חשוב להגן מפני התקפות כמו Rainbow Table, המנצלות טבלאות מוכנות מראש של ערכי Hash לסיסמאות



נפוצות, והתקפות Brute Force - Dictionary, המנסות לנחש סיסמאות על ידי בדיקה של אפשרויות רבות.

## 2. ניהול משאבים:

ההצפנה חייבת להישאר יעילה - כלומר, המערכת צריכה להיות מסוגלת לבצע הצפנה ושליפה של סיסמאות במהירות, גם כאשר מספר הסיסמאות גדול. האלגוריתמים צריכים להגדיל את רמת האבטחה מבלי להאט את ביצועי המערכת יתר על המידה, דבר שמורכב במיוחד כאשר מדובר במשתמשים רבים ובעומס גדול על המערכת.

## 3. שמירה על ביצועים מול עומסים:

יש למצוא איזון בין רמת אבטחה גבוהה (כמו הוספת Salt ו- Iterations) חישובים נוספים לכל סיסמה לבין עיבוד מהיר. ככל שמספר הפעולות או החישובים עולה, כך המערכת הופכת למאובטחת יותר אך עלולה להיות איטית יותר, ולכן יש צורך לקבוע את מספר החישובים כך שיהיה מספיק כדי לספק אבטחה טובה, אך לא יכביד יתר על המידה.

האלגוריתם של מנהל הסיסמאות המשודרג משתמש במבנה גרפי ומיישם הצפנה אסימטרית בנוסף לשיטות מבוססות תורת הגרפים, כדי ליצור מערכת הצפנה חזקה ומורכבת יותר.

## שלבי האלגוריתם

### 1. יצירת מפתחות אסימטריים

המערכת יוצרת זוג מפתחות - **מפתח פרטי ומפתח ציבורי** - במבנה של הצפנה אסימטרית.

- **מפתח ציבורי (Public Key):** נגיש לכולם ומשמש להצפנת הסיסמאות לפני שמירתן במנהל הסיסמאות.
- **מפתח פרטי (Private Key):** נשמר באופן סודי ומשמש לפענוח הסיסמאות.

### 2. הגדרת גרף סודי (מבנה גרפי בסיסי)

בנוסף לזוג המפתחות האסימטריים, עבור כל משתמש נבנה גרף סודי G המשמש כ"חתימה גרפית" ייחודית למשתמש. גרף זה משמש כבסיס להצפנה מבוססת תורת הגרפים, ויכלול מבנה קבוע של צמתים וקשתות עם מאפיינים ייחודיים.

### 3. יצירת גרף פומבי - מפתח ציבורי גרפי

לאחר הגדרת הגרף הסודי G, נבצע עליו טרנספורמציה איזומורפית PPP כדי לייצר גרף חדש G'G'G' שאינו נראה כמו הגרף המקורי אך שומר על הקשרים הלוגיים שלו. גרף זה יהיה המפתח הציבורי הגרפי, כך שניתן להצפין סיסמאות בעזרתו.

#### 4. הצפנת הסיסמאות (שילוב מבני גרפים והצפנה אסימטרית)

תהליך ההצפנה משלב את המפתח הציבורי הגרפי G עם המפתח הציבורי האסימטרי.

##### 1. שלב הצפנה ראשוני באמצעות הגרף:

- כל תו בסיסמה מיוצג במסלול או צבע של צומת בגרף G. ניתן למפות כל תו לסדרת צמתים וצבעים בגרף, כשהסיסמה המוצפנת היא רצף הצבעים והמעברים.

##### 2. שכבת הצפנה שנייה - הצפנה אסימטרית:

- לאחר הצפנת הסיסמה באמצעות מבנה הגרף, נשמור את הסיסמה המוצפנת כטקסט מוצפן נוסף, שעובר הצפנה נוספת באמצעות המפתח הציבורי האסימטרי.
- כתוצאה מכך, על מנת לפענח את הסיסמה נדרשים גם המפתח הפרטי הגרפי וגם המפתח הפרטי האסימטרי, מה שמוסיף שכבת אבטחה משמעותית.

#### 5. פענוח הסיסמה

כאשר המשתמש מבקש לשחזר את הסיסמה:

##### 1. שלב פענוח אסימטרי ראשוני: המערכת משתמשת במפתח הפרטי האסימטרי של המשתמש

כדי לפתוח את השכבה הראשונה של ההצפנה.

##### 2. פענוח גרפי:

- לאחר הפענוח האסימטרי, המערכת ניגשת למידע המוצפן ופותחת אותו באמצעות מפתח הגרף הסודי G ובטרנספורמציה ההפוכה  $P^{-1}P^{-1}$ .
- המערכת מאתרת את המסלולים והצבעים המתאימים בגרף וממירה אותם לתווים כדי לשחזר את הסיסמה המקורית.

#### 6. אבטחה ועמידות

כדי להבטיח רמת אבטחה גבוהה, האלגוריתם מתבסס על שילוב של הקשיים החישוביים של בעיות תורת הגרפים עם מורכבות ההצפנה האסימטרית:

- **הצפנה גרפית:** השימוש במבנה גרפי מקשה על תוקף לגלות את המפתח הגרפי או את המבנה הפנימי של הסיסמה.

- **הצפנה אסימטרית:** מספקת שכבה נוספת של הגנה, שמונעת גישה למידע גם במקרה של חשיפת גרף הפומבי 'G'G'.
- **עמידות נגד התקפות קוונטיות:** השימוש בשתי שיטות הצפנה שונות מגביר את העמידות גם במקרים של התקדמות במחשוב קוונטי.

## סיכום היתרונות

1. **חוזק אבטחה מוגבר:** שילוב תורת הגרפים עם הצפנה אסימטרית מספק אבטחה רב-שכבתית שקשה מאוד לפרוץ.
2. **פשטות שימוש:** המשתמשים יכולים לנהל סיסמאות באופן רגיל, כאשר המערכת מטפלת בכל תהליך ההצפנה והפענוח.
3. **גמישות והתאמה לעידן הקוונטי:** פתרונות המשלבים בעיות גרפיות עם הצפנה אסימטרית מספקים עמידות גם נגד התקפות ממחשבים קוונטיים.

## רקע תיאורטי בתחום הפרויקט

התקפות נפוצות:

| שם                                   | פירוט  |
|--------------------------------------|--|
| 1. התקפת כוח גס (Brute Force Attack) | התקפת כוח גס היא שיטה שבה האקר מנסה לנחש את הסיסמה הנכונה על ידי בדיקה של כל האפשרויות האפשריות, אחת אחרי השנייה, עד למציאת הסיסמה המתאימה. זו שיטה פשוטה למדי, אך אם הסיסמה קצרה או מורכבת מתווים פשוטים, האקר יכול למצוא אותה במהירות יחסית. ככל שהסיסמה מורכבת יותר (למשל, כוללת תווים מיוחדים, אותיות גדולות וקטנות, ומספרים), כך היא קשה יותר לפריצה בשיטה זו, ויידרש זמן רב יותר לניסיון ולשחזור של כל האפשרויות האפשריות. |
| התקפת Rainbow Table                  | התקפת Rainbow Table מתבצעת בעזרת טבלאות שנבנו מראש, המשלבות מחרוזות נפוצות ותוצאות של פונקציות Hash עבורן. הטבלה מכילה זוגות של טקסט קריא ותוצאת ה-Hash המתאימה לו, מה שמאפשר להאקר להשוות בקלות ערך Hash כלשהו למחרוזת מתאימה. אם הסיסמה נמצאת בטבלה, אפשר לפצח אותה במהירות מבלי לנסות לנחש את כל האפשרויות. הגנה בפני התקפות כאלה מתבצעת באמצעות הוספת Salt - ערך   |

|   |                  |
|---|------------------|
| אקראי שנוסף לסיסמה לפני יצירת ה-Hash - כדי לוודא שגם סיסמאות זהות יובילו לערכי Hash שונים.  |                  |
| התקפת Dictionary מבוססת על שימוש במילון מונחים ידוע מראש, הכולל רשימה של סיסמאות נפוצות, שמות, מילות מילון וביטויים שנמצאו בשימוש נפוץ. ההאקר מנסה להיכנס עם הסיסמאות מהרשימה עד למציאת סיסמה תואמת. התקפה זו יעילה אם הסיסמה נפוצה, כמו "123456" או "password" הגנה מפני התקפה זו יכולה לכלול בחירה בסיסמאות מורכבות יותר, כמו שילוב של תווים אקראיים, שמות שאינם נמצאים במילון ותחליפים יצירתיים. | התקפת Dictionary |

### סוגי הצפנות נפוצות עבור הצפנת סיסמאות:

#### Bcrypt.1

Bcrypt הוא אלגוריתם הצפנה שמיועד לאחסון סיסמאות בצורה מאובטחת. הוא משתמש במנגנון שנקרא (salt נתון אקראי) ומבצע גם Work Factor. Work Factor הוא קביעה של כמות העבודה או הזמן שנדרשים לצורך יצירת הצפנה של הסיסמה. ככל שכמות העבודה גבוהה יותר, כך יהיה יותר קשה לשבור את ההצפנה.

- יתרונות:

Salt : Salt הוא נתון אקראי שמתווסף לכל סיסמה, דבר שמונע התקפות כמו Rainbow Table (התקפה המנצלת טבלאות הצפנה מוכנות מראש כדי לפצח סיסמאות).

Work Factor - ככול שמספר החישובים שמבוצעים עבור כל סיסמה גבוה יותר, כך ההצפנה לוקחת יותר זמן ומקשה על האקר להריץ התקפות.

#### 2. Argon2

הוא אלגוריתם הצפנה חדש יותר, הוא מציע הגנה חזקה נגד התקפות, ומבצע שימוש בזיכרון בצורה חכמה. הוא גם מאפשר להגדיר את כמות הזמן והזיכרון הנדרשים לעיבוד הצפנה, דבר שמקשה על האקרים להריץ התקפות מבוססות חומרה.

יתרונות:

Memory-Hard - אלגוריתם זה דורש הרבה זיכרון בנוסף לזמן עיבוד, דבר שמקשה מאוד על ביצוע התקפות באמצעות GPU (מעבדים גרפיים) שהם מאוד מהירים ויכולים לבצע התקפות כמו Brute Force במהירות גבוהה.

Argon2 מאפשר להתאים את כמות הזמן והזיכרון הנדרש לפי הצרכים של המערכת. יכולת הסתגלות לעתיד: מאפשר להסתגל לשדרוגי חומרה עתידיים ולשיפור טכנולוגיות ההתקפה.

### 2.3 (PBKDF2 (Password-Based Key Derivation Function

PBKDF2 הוא אלגוריתם הצפנה שמבצע המרה של סיסמה למפתח הצפנה תוך שימוש בהחזרות רבות (iterations). כל החזרה היא חישוב נוסף שמקשה על פיצוח הסיסמה. האלגוריתם פועל כך שעם כל החזרה הוא הופך את הסיסמה למפתח חזק יותר.

יתרונות:

תמיכה רחבה: PBKDF2 נתמך כמעט בכל הפלטפורמות ויישומים מודרניים, כולל מערכות הפעלה כמו

Linux ו-Windows

- החזרות מרובות: מאפשר הגדלה של מספר החזרות (iterations) כדי להאט את ביצוע ההתקפות.

Salt : Salt הוא מבצע שימוש ב-Salt, כלומר מוסיף ערך אקראי לכל סיסמה כדי למנוע התקפות Rainbow

Table.

### 4. Script -

הוא אלגוריתם הצפנה שמבצע שימוש בזיכרון בנוסף לזמן החישוב. הוא נועד להקשות על התקפות באמצעות חומרה מתקדמת כמו GPU בדיוק כמו Argon2 הוא memory-hard דורש הרבה זיכרון, דבר שמקשה על התקפות כוח גס באמצעות חישובים מרובים במקביל.

יתרונות:

זיכרון וזמן: Script מצריך זיכרון נוסף לעיבוד, דבר שמקשה מאוד על התקפות עם חומרה מתקדמת.

מוכר ומאומת: נמצא בשימוש במערכות מסוימות כמו Bitcoin והיה חלק מההגנה על קריפטו-מטבעות.

לסיכום

במערכת לניהול סיסמאות, האתגר הגדול ביותר הוא להבטיח שהסיסמאות נשארות מאובטחות לאורך כל מחזור החיים שלהן, החל מאחסון ועד אחזור. הצפנה היא הדרך המובילה להבטיח שהמידע נשמר חסוי גם כאשר הוא נמצא בשרתים או במכשירים אחרים. הבעיה האמיתית היא כיצד לשמור על המידע מוצפן באופן שאינו מאפשר לחשוף את הסיסמאות בפני גורמים זרים, תוך שמירה על נגישות נוחה ובטוחה למשתמש המורשה.

הצפנה היא השלב הראשון בהגנה על הסיסמאות, עם זאת, הצפנה לבדה אינה מספיקה. כל אלגוריתם הצפנה דורש גם טיפול בטכנולוגיות נוספות כמו ניהול מפתחות הצפנה, קריפטוגרפיה של מפתחות, ותהליכי אימות על מנת למנוע גישה בלתי מורשית. בנוסף לכך, יש לקחת בחשבון את הצורך בהגנה מפני מתקפות כוח גס (Brute Force) מתקפות פנימיות במערכת, ומתקפות של פשינג שמטרתן לגנוב את מפתחות ההצפנה או את הסיסמאות עצמן.

לאחר הצפנת הסיסמאות, הצורך להחזיר את המידע באופן מאובטח הופך להיות אתגר נוסף. על מנת למנוע חשיפה של הסיסמאות למשתמשים בלתי מורשים, המערכת חייבת להשתמש בשיטות אימות חזקות, כגון אימות דו-שלבי (2FA) או אימות ביומטרי (כגון טביעת אצבע או זיהוי פנים), במטרה לוודא שהגישה לאחסון הסיסמאות היא אך ורק של המשתמש המורשה.

מורכבות החיפוש והאחזור:

כאשר מדובר בניהול סיסמאות, כמות הסיסמאות שברשות המשתמש יכולה להיות גבוהה מאוד, במיוחד כאשר ישנן סיסמאות עבור אתרים, אפליקציות, רשתות חברתיות, מערכות בנקאיות ועוד. המערכת חייבת להיות מסוגלת לספק מנגנון חיפוש יעיל ומהיר כדי לאפשר למשתמשים לאתר במהירות את הסיסמה שהם מחפשים, מבלי לבלות זמן מיותר בגלישה או חיפושים מייגעים. החיפוש לא אמור להחמיר את הבעיה של חשיפת המידע, ולכן נדרשים פתרונות אלגוריתמיים שיאפשרו חיפוש מהיר, תוך שמירה על רמת האבטחה.

כדי להשיג חיפוש מהיר, ניתן להשתמש במבני נתונים מתקדמים המאפשרים חיפושים בסיבוכיות נמוכה, כמו Hash Tables שמאפשרים חיפוש בגבול זמן קבוע,  $O(1)$ , או עצים בינאריים מאוזנים (כמו AVL או Red-Black Trees), המאפשרים חיפוש, הוספה ומחיקה בזמן לוגריתמי  $O(\log n)$ . כל סיסמה במערכת תהיה מקוטלגת לפי קריטריונים שונים כמו תיוג, קטגוריות (כגון חשבונות בנק, רשתות חברתיות), או שמות אתרים. מבנה זה מאפשר למשתמש לחפש סיסמאות לפי מאפיינים מסוימים במקום לחפש אותן לפי המילה המדויקת.

לדוגמא, אם המשתמש רוצה לאתר סיסמה עבור חשבון פייסבוק, המערכת תוכל לחפש תחילה בקטגוריית "רשתות חברתיות" ולאחר מכן לאתר את הסיסמה המתאימה, וזאת ללא צורך בשאילתת חיפוש מסורבלת או בהשוואות חיפוש אינדיבידואליות לכל סיסמה. בנוסף, המערכת תוכל לכלול מנגנוני חיפוש חכמים נוספים, כמו חיפוש במילים חלקיות (autocomplete), סינון לפי זמן עדכון אחרון או דירוג סיסמאות לפי תדירות השימוש שלהן.

אך יש לקחת בחשבון שהצורך בחיפוש מהיר ומורכב לא תמיד מתיישב עם האבטחה. ככל שיותר נתונים ישמרו במערכת, כך יש להקפיד יותר על שמירה על אבטחת המידע במהלך החיפושים. למשל, על מנת להבטיח שסיסמאות לא ייחשפו בעת החיפוש, ניתן להחביא את המידע עצמו ולהציג רק תמצית מידע (כגון תיוג, שם אתר או שם משתמש), וזאת לאחר בדיקה מדוקדקת של האישורים המתקבלים בעת הבקשה.

הפרויקט מתמקד בפיתוח מנהל סיסמאות מתקדם, שנועד להבטיח אבטחה ונוחות בשמירה וניהול סיסמאות למגוון שירותים דיגיטליים. מנהל הסיסמאות SecureVault יאפשר למשתמשים לשמור את סיסמאותיהם במקום מאובטח, ייצר סיסמאות חזקות וימנע את הצורך לזכור סיסמאות פשוטות וחלשות. הפרויקט יכלול ממשק מאובטח ונגיש לשמירה וניהול הסיסמאות. הליכים עיקריים בפתרון הבעיה:

| שלב                           | תיאור   |
|-------------------------------|---|
| 1. הצפנה ואחסון סיסמאות       | בעת יצירת סיסמה חדשה, מנהל הסיסמאות מצפין אותה באמצעות אלגוריתם הצפנה חזק, כמו bcrypt. סיסמה מוצפנת זו נשמרת במאגר המידע המאובטח של המנהל.              |
| 2. שימוש ב-Master Password    | משתמשים נדרשים ליצור סיסמת-מאסטר חזקה שתאפשר גישה למנהל הסיסמאות כולו. סיסמת-מאסטר זו אינה מאוחסנת במערכת, אלא מעובדת ומאומתת בכל פעם לפי מפתחות הצפנה. |
| 3. אימות דו-שלבי              | פתרונות אבטחה רבים כוללים אימות דו-שלבי, שבו המשתמש נדרש לאמת את זהותו דרך אמצעי נוסף, כמו קוד חד-פעמי או הודעה לטלפון.                                 |
| 4. זיהוי אוטומטי ומילוי טפסים | מנהל הסיסמאות מזהה אתרים שבהם יש צורך באימות, מאמת את כתובת האתר ומשתמש באישורים המאוחסנים כדי למלא את שדות ההתחברות באופן מאובטח                       |

### פרוטוקולי תקשורת אבטחתיים בניהול סיסמאות:

#### 1:HTTPS.

זהו הפרוטוקול הנפוץ ביותר לתקשורת מאובטחת בין הלקוח לשרת, והוא מבוסס על SSL/TLS כדי להבטיח שהמידע המועבר מוגן. HTTPS מונע מתקפות "אדם בתווך" (MITM), בכך שהוא מצפין את המידע המועבר.

#### 1:TLS.

פרוטוקול אבטחה מתקדם המספק פרטיות ושלמות נתונים, TLS מהווה גרסה מתקדמת של SSL ומשמש כסטנדרט בתקשורת מאובטחת. שהנתונים המועברים מוצפנים היטב.

1SRP (Secure Remote Password Protocol): פרוטוקול זה מאפשר אימות מאובטח בין הלקוח לשרת מבלי לשדר את הסיסמה עצמה. כך, גם אם התקשורת נפרצה, הסיסמה עצמה נשמרת מוגנת.

2OAuth 2.0:

פרוטוקול הרשאות שמאפשר למשתמשים לגשת לשירותים שונים באינטרנט מבלי לחשוף את הסיסמה שלהם, תוך שימוש באישורים זמניים או בהרשאות שניתן לנהל ולשלט בהן. שילוב של הליכים אלו ופרוטוקולים מתקדמים מבטיח שמירה מיטבית על הסיסמאות והנתונים האישיים, תוך אפשרות לשימוש נוח ונגיש.

### הרחבות נוספות/פיתוחים עתידיים לפרויקט:

#### 1. אינדקס חיפוש חכם למנהל הסיסמאות:

על מנת לייעל את החיפושים, ניתן ליצור אינדקס חיפוש שמפרט את שמות השירותים (כגון פייסבוק, ג'מייל) עם החיבורים לסיסמאות. האינדקס יאפשר למשתמש לאחזר סיסמה על פי שם השירות בצורה מהירה בלי לעבור על כל רשימת הסיסמאות.

#### 2. מצב חירום - גישה מוגבלת לשעת חירום:

המערכת תכלול אפשרות חירום, שתאפשר לקרוב משפחה או לאדם אחר שקיבל הרשאה מראש גישה מוגבלת לחשבונות במקרה של מצבי חירום. פונקציה זו תוגדר מראש על ידי המשתמש ותחייב את אישורו.

#### 3. מערכת תזכורות ועדכונים:

המערכת תוכל לשלוח למשתמש תזכורות לעדכון סיסמאות ישנות, ולשלוח התראות על שימוש בלתי מורשה במידה ויאותרו ניסיונות כניסה חריגים. המשתמש יוכל להגדיר באילו אתרים ואפליקציות הוא מעוניין לבצע החלפה תקופתית של הסיסמאות.

### מסדי נתונים

בפרויקט זה נשתמש ב- DB מונגו על מנת לנהל בצורה מסודרת ויעילה את המידע הרגיש של המשתמשים, כמו סיסמאות מוצפנות, פרטי גישה נוספים והגדרות אישיות. מסד הנתונים יאפשר אחסון מאובטח של סיסמאות בצורה שהן חסיונות בפני התקפות כמו כוח גס והתקפות Rainbow Table. בנוסף, השימוש במסד נתונים יאפשר ניהול גמיש של הנתונים, תוך שמירה על רמת אבטחה גבוהה. תשתית זו תאפשר לעמוד בדרישות אבטחה מחמירות ולספק למשתמש חווית שימוש חלקה ונוחה.





## מבוא

### הרקע לפרויקט

הפרויקט עוסק בפיתוח מנהל סיסמאות מאובטח שישמור סיסמאות בצורה מאובטחת. הפרויקט עוסק בעיקר בהצפנת הסיסמאות ושמירתן בצורה מאובטחת וכן הצפנת מפתח הכניסה של המשתמש שדורש הצפנה משמעותית כדי למנוע גישה לכל סיסמאות המשתמש.

### תהליך המחקר

במהלך המחקר התחלתי בלמידה כללית על עולם מנהלי הסיסמאות - אלו כלים קיימים, מה ההבדלים ביניהם, ואילו טכנולוגיות עומדות מאחוריהם. זיהיתי שקיימים סוגים שונים של מנהלי סיסמאות: כאלה שמובנים כתוסף בדפדפן (כגון Chrome Password Manager), תוכנות עצמאיות שניתן להתקין על המחשב או על המכשיר הנייד (כגון KeePass), שירותים מבוססי ענן המאפשרים סנכרון בין מכשירים שונים (כגון LastPass, Bitwarden).

בדקתי את היתרונות והחסרונות של כל שיטה - בעיקר בהיבטי אבטחה, נוחות שימוש, ונגישות. גיליתי שלמרות הנוחות הרבה שבשירותי ענן, קיימים סיכונים של פריצות או גניבת נתונים אם לא קיימת מערכת הגנה חזקה.

בהמשך העמקתי בנושא של אבטחת מידע והצפנת סיסמאות. למדתי כיצד מאובטחים המידע והסיסמאות במנהלי הסיסמאות באמצעות אלגוריתמים מתקדמים של הצפנה. כאן התחלתי בלמידה עצמית של מושגים שלא הכרתי, תוך שימוש במאמרים מקצועיים, סרטוני הסבר, ואתרים טכנולוגיים אמינים. מצד שני, חזרתי גם על נושאים שכבר הכרתי מהלימודים - כמו מבנה בסיסי של אלגוריתם הצפנה, ואופן השימוש במפתחות.

האלגוריתמים שנסקרו:

AES (סימטרי)

DES (סימטרי)

RSA (אסימטרי)

Diffie-Hellman (להחלפת מפתחות)

(Hashing (SHA-256 - לצורך אימות סיסמאות ולא הצפנה

למדתי גם על חשיבותו של **אימות דו-שלבי (2FA)** בהגנה על חשבונות, ובייחוד כשמדובר בגישה למנהל סיסמאות. זהו רכיב חשוב במניעת גניבת זהות גם אם הסיסמה עצמה דלפה.

## סקירת ספרות

**internet israel** - הוספת מלח לסיסמה:

הוספת מלח (Salt) לסיסמה היא טכניקה לאבטחת סיסמאות המאוחסנות במסד נתונים. המלח הוא מחרוזת אקראית שנוספת לסיסמה לפני ההצפנה, כך שגם אם שתי סיסמאות זהות, הפלט שלהן יהיה שונה.

**fortinet** - מהי הצפנה, סוגי הצפנות ועוד.

הצפנה היא שיטה להגנה על מידע על ידי הפיכתו לבלתי קריא עבור גורמים לא מורשים. ישנם סוגים שונים של הצפנות, למשל הצפנה סימטרית שבה אותו מפתח משמש להצפנה ולפענוח, והצפנה אסימטרית שמשמשת בזוג מפתחות – ציבורי ופרטי. הצפנה משמשת באבטחת תקשורת, הגנה על נתוני משתמשים, ותעבורת מידע באינטרנט.

**eset** - מהו מנהל סיסמאות, מה צריך להיות בו-

מנהל סיסמאות הוא כלי שמאחסן ומנהל סיסמאות בצורה מאובטחת. הוא שומר סיסמאות מוצפנות ומאפשר גישה נוחה בעזרת סיסמה ראשית אחת. חשוב שיהיה בו קידוד חזק, אימות דו-שלבי, ושירות גיבוי מאובטח.

**RSA** - geeks for geeks

אלגוריתם RSA הוא אלגוריתם קריפטוגרפיה אסימטרי או מבוסס מפתח ציבורי, כלומר הוא פועל על שני מפתחות שונים: מפתח ציבורי ומפתח פרטי. המפתח הציבורי משמש להצפנה והוא ידוע לכולם, בעוד שהמפתח הפרטי משמש לפענוח וחייב להישמר בסוד על ידי המקבל.

**Mathematics Library** מעגלים המילטונים

מסלול המילטון הוא מעגל שחייב לעבור דרך כל קודקוד בגרף פעם אחת ורק פעם אחת מלבד לעומת מעגל המילטון שבו עוברים בקודקוד ההתחלה פעמיים - בהתחלה ובסיום.

**כתב עת בינלאומי למדע מחשבים מתקדמים ויישומים (IJACSA)**

המאמר מפרט על הצפנה סימטרית המבוססת על תורת הגרפים, האלגוריתם מממש מודל מתקדם של תורת הגרפים על ידי ייצוג כל בלוק של 78 תווים כגרף שלם מסדר 13, המכיל 6 מעגלים המילטוניים זרים. בתהליך ההצפנה, כל אחד מ-6 תת-בלוקים (13 תווים כל אחד) ממופה למעגל המילטוני ייחודי, כאשר ערכי ASCII של התווים משמשים כמשקלי הקשתות בגרף. המעגלים הזרים מבטיחים שכל קשת בגרף משתייכת למעגל יחיד בלבד, ובכך מונעים דליפת מידע בין תת-בלוקים שונים.

## אתגרים מרכזיים

### הבעיה איתה התמודדתי

1. **אבטחה** - מציאת אלגוריתם מאובטח מספיק להצפנת סיסמאות מהווה אתגר עיקרי. על האלגוריתם להיות עמיד בפני מתקפות כגון כוח גס (brute force), ניחוש חוזר (dictionary attacks) ועוד.
2. יעילות - על האלגוריתם להיות יעיל מבחינת משאבי חישוב וזמן, אך גם להקשות באופן משמעותי על תוקפים לשחזר את הסיסמה המקורית.
3. **אימות זהות המשתמש בצורה בטוחה** - היה צורך לבחון כיצד אפשר לוודא שרק המשתמש האמיתי יוכל לגשת לסיסמאות שלו, תוך שילוב שיטות כמו אימות דו-שלבי, מבלי לפגוע בשימושיות.

## הסיבות לבחירת הנושא

בחירת נושא לפיתוח מנהל סיסמאות, שילבתי שיקולים מקצועיים עם מניעים אישיים. מבחינה מקצועית, ניהול סיסמאות הוא תחום חיוני באבטחת מידע, במיוחד בעידן שבו מתקפות סייבר הופכות לנפוצות ומתוחכמות יותר. האתגר ביצירת מערכת שמאזנת בין אבטחה גבוהה לנוחות שימוש, תוך יישום טכנולוגיות כמו הצפנה מתקדמת ואימות רב-גורמי, מהווה הזדמנות ללמידה מעמיקה ולהתמודדות עם בעיות אמיתיות בתחום התכנות.

מהבחינה האישית, תחום ההצפנות מאוד מעניין אותי. אני רואה בכך הזדמנות להעמיק בתחום קריטי באבטחת מידע, להבין כיצד אלגוריתמים מורכבים מגנים על נתונים ולפתח פתרונות שמספקים הגנה גבוהה מפני מתקפות שונות.

שילוב זה של עניין מקצועי בצדדים הטכניים והאבטחתיים של המערכת, יחד עם סקרנות אישית לעולמות ההצפנה והאבטחה, הפך את פיתוח מנהל הסיסמאות לנושא משמעותי ומרתק עבורי.

## מוטיבציה לעבודה

הפרויקט מספק מערכת מאובטחת שתאחסן את כל הסיסמאות בצורה בטוחה ותבטיח למשתמש שהסיסמאות שלו בטוחות.

## על איזה צורך הפרויקט עונה

בעידן המודרני, אדם ממוצע מנהל עשרות חשבונות מקוונים, כאשר כל אחד מהם דורש סיסמה ייחודית, מורכבת ומעודכנת תדיר. מציאות זו יוצרת פרדוקס אבטחתי: מחד, שימוש בסיסמאות חלשות או חוזרות מגדיל את פגיעות המשתמש לפריצות; מאידך, שימוש בסיסמאות מורכבות וייחודיות מקשה על זכירתן, מה שמוביל לשימוש בשיטות לא מאובטחות לאחסון.

## פתרון: מערכת ניהול סיסמאות מאובטחת וחכמה

הפרויקט מציע מנהל סיסמאות מתקדם המשלב:

- אחסון מאובטח של כל הסיסמאות תחת מפתח ראשי יחיד
- מנגנון יצירת סיסמאות חזקות המותאמות לדרישות הספציפיות של כל פלטפורמה
- זיהוי אוטומטי של צורכי הסיסמה בזמן אמת
- שכבות הגנה מרובות להבטחת פרטיות המידע האישי
- ממשק משתמש אינטואיטיבי המאזן בין אבטחה לשימושיות

### הצגת פתרונות לבעיה

#### 1. פתרונות מבוססי ענן

נבחנו פלטפורמות לניהול סיסמאות המאחסנות את הנתונים בשרתים מרוחקים. יתרונות: גישה מכל מכשיר, סנכרון אוטומטי וגיבוי מתמיד. חסרונות: תלות בחיבור אינטרנט, אבטחה תלויה בספק השירות, וסיכון פוטנציאלי במקרה של פריצה לשרתים.

#### 2. פתרונות מקומיים (Offline)

נחקרו אפליקציות לאחסון סיסמאות באופן מקומי על המכשיר. יתרונות: פרטיות מוגברת, אי-תלות בחיבור אינטרנט. חסרונות: חוסר סנכרון אוטומטי בין מכשירים, סיכון לאובדן נתונים במקרה של תקלה במכשיר.

### מטרות / יעדים

המטרה העיקרית בפיתוח מנהל סיסמאות מאובטח היא הצפנת הסיסמאות באמצעות אלגוריתם הצפנה מתקדם לשמירה על פרטיות המשתמשים. בנוסף, המערכת שואפת לספק ממשק פשוט ואינטואיטיבי לניהול סיסמאות, אימות רב-שלבי

### מדדי הצלחה למערכת

בבדיקת מדדי הצלחה של מערכת ניהול סיסמאות נתמקד במספר היבטים קריטיים: אבטחת המידע, שמירת אמינות הנתונים, וחווית משתמש. נבחן את איכות ההצפנה וחוזק אלגוריתם, לוודא שלא קיימת דליפת מידע בעת העברת סיסמאות והזדהות, ולבדוק עמידות המערכת בפני התקפות. במקביל, חשוב למדוד את נוחות השימוש - מהירות אחזור סיסמאות טפשטות היצירה של סיסמאות חזקות. מערכת מצליחה תאזן בין רמת אבטחה גבוהה לבין נגישות ושימושיות ללא פגיעה באחד מהם.

### אתגרים

- הגנה על המפתח הראשי KEK מפני חשיפה או אובדן

- הבנה מעמיקה של מעגלים המילטוניים זרים ויישום נכון במטריצות סמיכות.

## רקע תיאורטי

### מהו מנהל סיסמאות

מנהל סיסמאות הוא כלי דיגיטלי המשמש לאחסון, ניהול ואבטחת סיסמאות המשתמש בצורה מוצפנת. הוא מאפשר למשתמשים ליצור סיסמאות מורכבות וייחודיות לכל חשבון מקוון, לאחסן אותן במקום מאובטח, ולגשת אליהן באמצעות סיסמה ראשית אחת בלבד. מנהלי סיסמאות נועדו להתמודד עם האתגר של זכירת מספר רב של סיסמאות שונות, ובכך לשפר את אבטחת המידע האישי באינטרנט ולהקטין את הסיכון לפריצות הנובעות משימוש חוזר בסיסמאות פשוטות או זהות.

אחת לתקופה מגיעה טכנולוגיה חדשה שמבטיחה לשים סוף לסיסמאות אחת ולתמיד, אך נראה שכרגע אף טכנולוגיה לא הצליחה להחליף את הסיסמה באופן מלא.

סיסמאות טומנות בחובן סיכוני אבטחה אפשריים, והאמירה הזאת הופכת למסוכנת כפליים כשנזכרים על מה הן אמורות להגן - על הכל: החל באפליקציות המסרים והרשתות החברתיות ועד לחשבונות הסטרימינג והזמנת מוניות. הוסיפו לכך את העובדה שרבים מהאנשים אינם משתמשים באימות דו-שלבי, גם בחשבונות הכי חשובים שלהם.

כתוצאה מכל אלה, אם האקרים מצליחים לשים את ידיהם על סיסמה כלשהי, הם יכולים לקבל גישה לאינספור נתונים אישיים ואמצעי תשלום שאוחסנו באמצעים שונים. למעשה, נוצר שוק שחור די גדול שעוסק בסחר בסיסמאות לחשבונות מקוונים.

החדשות הטובות הן שמנהלי סיסמאות מציעים אופן פעולה שמתגבר על רבים מהחסרונות של סיסמאות סטטיות ועל האופן הלא-בטוח שבו רבים מאיתנו משתמשים בהן. אך לא כל מנהלי הסיסמאות זהים אחד לשני.

### למה סיסמאות כ"כ חשובות

מדוע שימוש בסיסמאות כולל סיכון אבטחה? זה מכיוון שניתן להשיג אותן באופן לא-מורשה במספר דרכים, ביניהן:

- גניבה שלהן מחברות שאתם בהתקשרות איתן, כחלק מדליפת מידע בקנה מידה גדול.
- השגה שלהן באמצעות מתקפת פישג
- ניחוש שלהן באמצעות מתקפת Brute Force אוטומטית שמנסה שילובים שונים של סיסמאות נפוצות.

גם היום, אחת הסיסמאות הנפוצות היא password, כשהסיסמה הבאה אחריה היא 123456. מבין 10 הסיסמאות הנפוצות ביותר, הרוב המכריע יכול להיפרץ בתוך שנייה בודדת.

לאחר שהסיסמאות נגנבו הן נמכרות ברשת האפלה, בדרך כלל כחלק ממאגר גדול שכולל גם שמות משתמש שמתאימים לסיסמאות האלה. דוח משנת 2022 חשף 24 מיליארד צמדים כאלה (שמות משתמש וסיסמאות) הנסחרים בשווקים שחורים של עולם פשיעת הסייבר - עלייה של 65% לעומת כמות הצמדים שנסחרו בשנת 2020. ברוב המקרים, האקרים יזינו את הסיסמאות הגנובות האלה בכלים ל"העמסת סיסמאות" (שימוש באותו שם המשתמש והסיסמה באתרים שונים) כדי לראות אם אותה הסיסמה הוגדרה גם באפליקציות או באתרים אחרים. אם זה אכן נכון, הם יוכלו לפרוץ גם את החשבונות האלה.

כל אלה הופכים את השימוש בסיסמאות ייחודיות וחזקות לחשוב יותר מתמיד, ויש לעשות זאת בכל האתרים, האפליקציות והחשבונות המקוונים. מנהל סיסמאות זו דרך מצוינת לעשות זאת.

מה צריך לחפש במנהל סיסמאות

מנהלי סיסמאות הם אפליקציות שנועדו לאחסן את כל הסיסמאות במקום בטוח. הרעיון המרכזי הוא שהתוכנה הזאת תבקש ממכם להזין סיסמת מאסטר בודדת - זה כל מה שהמשתמש צריך לזכור. כל יתר הדברים ינוהלו באופן אוטומטי ע"י האפליקציה - כולל יצירת סיסמאות ארוכות וייחודיות לכל אתר בנפרד ומילוי אוטומטי שלהן בזמן כניסה לחשבון. עם זאת, ישנן מבחר אפשרויות שונות בשוק.

בין התכונות שצריכות להיות במנהל הסיסמאות:

1. **הסיסמאות מוגנות באמצעות הצפנה חזקה.** זה אומר שגם אם ספק שירותי ניהול הסיסמאות חווה פריצה, עברייני הסייבר לא יוכלו לגשת לסיסמאות שהלקוחות אחסנו בו.
2. **מחולל סיסמאות חזקות** שנועד להציע מחרוזות ארוכות, מורכבות ואקראיות של מספרים, אותיות וסימנים מיוחדים לכל סיסמה וסיסמה. כך למעשה כמעט ואין סיכוי שהאקר יוכל לנחש את הסיסמה באמצעות תוכנות פריצה בכוח.

הגדרת הצפנה

הצפנה היא סוג של אבטחת מידע שבה מידע מומר לטקסט מוצפן. רק אנשים מורשים בעלי המפתח יכולים לפענח את הקוד ולגשת למידע המקורי בטקסט רגיל.

במילים פשוטות עוד יותר, הצפנה היא דרך להפוך נתונים לבלתי קריאים עבור גורם לא מורשה. זה משמש לסכל פושעי סייבר, שאולי השתמשו באמצעים מתוחכמים למדי כדי לקבל גישה לרשת ארגונית - רק כדי לגלות שהנתונים אינם קריאים ולכן חסרי תועלת.

הצפנה לא רק מבטיחה את סודיות הנתונים או ההודעות, אלא גם מספקת אימות ושלמות, ומוכיחה שהנתונים או ההודעות הבסיסיים לא שונו בשום צורה ממצבם המקורי.

כיצד הצפנה פועלת

מידע מקורי, או טקסט רגיל, יכול להיות משהו פשוט כמו "שלום עולם!". כטקסט מוצפן, זה עשוי להיראות כמשהו מבלבל כמו gvU2x+#0\*7 - משהו שנראה אקראי או לא קשור לטקסט המקורי.

הצפנה, לעומת זאת, היא תהליך הגיוני, לפיו הצד המקבל את הנתונים המוצפנים - אך גם מחזיק במפתח - יכול פשוט לפענח את הנתונים ולהפוך אותם בחזרה לטקסט רגיל.

במשך עשרות שנים, תוקפים ניסו בכוח גס - למעשה, על ידי ניסיון שוב ושוב - להבין מפתחות כאלה. לפושעי סייבר יש גישה גוברת לכוח מחשוב חזק יותר, כך שלפעמים, כאשר קיימות פגיעויות, הם מסוגלים לקבל גישה.

יש להצפין נתונים כאשר הם נמצאים בשני מצבים שונים: "במנוחה", כאשר הם מאוחסנים, כמו במסד נתונים; או "במעבר", בזמן שהם נגישים או מועברים בין צדדים.

אלגוריתם הצפנה הוא נוסחה מתמטית המשמשת להמרת טקסט רגיל (נתונים) לטקסט מוצפן. אלגוריתם ישתמש במפתח כדי לשנות את הנתונים בצורה צפויה. למרות שהנתונים המוצפנים נראים אקראיים, ניתן למעשה להפוך אותם בחזרה לטקסט רגיל על ידי שימוש חוזר במפתח. כמה אלגוריתמי הצפנה נפוצים כוללים את, Blowfish, Advanced Encryption Standard (AES), Rivest Cipher 4 (RC4), RC5, RC6, Data Encryption Standard (DES) ו-Twofish.

ההצפנה התפתחה עם הזמן, מפרוטוקול ששימש רק ממשלות לפעולות סודיות ביותר, ועד לחובה יומיומית עבור ארגונים כדי להבטיח את אבטחת ופרטיות הנתונים שלהם.

#### סוגי הצפנות

ישנם סוגים רבים ושונים של הצפנה, לכל אחד יתרונות ושימוש משלו.

#### הצפנה סימטרית

בשיטת הצפנה פשוטה זו, משתמשים רק במפתח סודי אחד להצפנה ולפענוח מידע. למרות שזוהי טכניקת ההצפנה העתיקה והמוכרת ביותר, החיסרון העיקרי הוא ששני הצדדים צריכים שיהיה להם את המפתח המשמש להצפנת הנתונים לפני שיוכלו לפענח אותם. אלגוריתמי הצפנה סימטריים כוללים את AES-128, AES-192 ו-AES-256. מכיוון שהיא פחות מורכבת ומבצעת מהר יותר, הצפנה סימטרית היא השיטה המועדפת להעברת נתונים בכמויות גדולות.



## הצפנה אסימטרית

הצפנה אסימטרית, הידועה גם כקריפטוגרפיה של מפתח ציבורי, היא שיטה חדשה יחסית המשתמשת בשני מפתחות שונים אך קשורים להצפנה ולפענוח נתונים. מפתח אחד הוא פרטי ומפתח אחד הוא ציבורי. המפתח הציבורי משמש להצפנת נתונים, והמפתח הפרטי משמש לפענוח (ולהיפך). אבטחת המפתח הציבורי אינה נחוצה מכיוון שהוא זמין לציבור וניתן לשתף אותו באינטרנט.

הצפנה אסימטרית מציגה אפשרות חזקה להבטחת אבטחת המידע המועבר באינטרנט. אתרי אינטרנט מאובטחים באמצעות תעודות (Secure Socket Layer (SSL או Transport Layer Security (TLS). שאלתה לשרת אינטרנט שולחת עותק של התעודה הדיגיטלית, וניתן לחלץ מפתח ציבורי מתעודה זו, בעוד שהמפתח הפרטי נשאר פרטי.

### תקן הצפנת נתונים (DES)

DES היא שיטת מפתח סימטרי להצפנת נתונים מיושנת. DES פועלת באמצעות אותו מפתח להצפנה ופענוח הודעה, כך שגם השולח וגם המקבל חייבים להיות בעלי גישה לאותו מפתח פרטי. DES הוחלף על ידי אלגוריתם AES המאובטח יותר. הוא אומץ על ידי ממשלת ארה"ב כתקן רשמי בשנת 1977 להצפנת נתוני מחשב ממשלתיים. ניתן לומר ש-DES היה הדחף לתעשיית הקריפטוגרפיה וההצפנה המודרנית.

### תקן הצפנת נתונים משולשת (3DES)

תקן הצפנת הנתונים המשולשת כלל הפעלת אלגוריתם DES שלוש פעמים, עם שלושה מפתחות נפרדים. 3DES נתפס בעיקר כאמצעי זמני, מכיוון שאלגוריתם DES היחיד נתפס יותר ויותר כחלש מדי מכדי לעמוד בפני התקפות כוח ברוט וה-AES החזק יותר עדיין היה בהערכה.

### RSA

RSA (Rivest-Shamir-Adleman) הוא אלגוריתם והבסיס למערכת קריפטוגרפית - חבילה של אלגוריתמים קריפטוגרפיים המשמשים לשירותי אבטחה או מטרות ספציפיות. זה מאפשר הצפנת מפתח ציבורי ומשמש לעתים קרובות על ידי דפדפנים כדי להתחבר לאתרי אינטרנט ועל ידי רשתות פרטיות וירטואליות (VPN). RSA הוא אסימטרי, שבו שני מפתחות שונים משמשים להצפנה: אחד ציבורי ואחד פרטי. אם הפענוח מתבצע באמצעות המפתח הציבורי, ההצפנה מתבצעת באמצעות המפתח הפרטי, או להיפך.

### תקן הצפנה מתקדם (AES)

תקן ההצפנה המתקדם, שפותח בשנת 1997 על ידי המכון הלאומי לתקנים וטכנולוגיה (NIST) כחלופה לתקן הצפנת הנתונים, הוא צופן שנבחר על ידי ממשלת ארה"ב כדי להגן על מידע רגיש. ל-AES שלושה אורכי מפתח שונים להצפנה ולפענוח של בלוק של הודעות: 128 סיביות, 192 סיביות ו-256 סיביות. AES נמצא בשימוש נרחב להגנה על נתונים במנוחה ביישומים כגון מסדי נתונים ודיסקים קשיחים.

## הצפנה בענן

הצפנת ענן היא שירות המוצע על ידי ספקי אחסון ענן, שבו נתונים מוצפנים תחילה באמצעות אלגוריתמים לפני שהם נדחפים לענן אחסון. לקוחות ספק אחסון ענן חייבים להיות מודעים לרמת העומק של המדיניות והנהלים של הספק להצפנה וניהול מפתחות הצפנה ולהרגיש בנוח עם זה.

מכיוון שהצפנה צורכת רוחב פס רב יותר, ספקי ענן רבים מציעים הצפנה בסיסית רק על מספר שדות של מסד נתונים, כגון סיסמאות ומספרי חשבון. לעתים קרובות זה לא מספיק עבור ארגונים מסוימים. לכן הם מסתמכים על מודל "הבא את ההצפנה שלך" (BYOE) שבו הם משתמשים בתוכנת ההצפנה שלהם ומנהלים את מפתחות ההצפנה שלהם כדי להבטיח רמת אבטחה של מחשוב ענן שהם מרגישים בנוח איתה.

כגישה הפוכה, הצפנה כשירות (EaaS) התפתחה כשירות פשוט בתשלום לפי שימוש שלקוחות יכולים לרכוש מספק ענן, ולנהל את ההצפנה בעצמם בסביבה מרובת דיירים.

## הצפנה מקצה לקצה

הצפנה מקצה לקצה (E2EE) מבטיחה שרק שני המשתמשים המתקשרים זה עם זה יוכלו לקרוא את ההודעות. אפילו מתווך, כמו ספק שירותי התקשורת או האינטרנט, אינו יכול לפענח את ההודעות. E2EE נתפס בדרך כלל כדרך הבטוחה ביותר לתקשר באופן פרטי ומאובטח באינטרנט. דוגמאות ל-E2EE הנמצא בשימוש כוללות את שירות המסרים WhatsApp, אשר טוען כי הודעות המשתמשים מאובטחות באמצעות "מנעולים".

## יתרונות ההצפנה

### פרטיות ואבטחה

הצפנה יכולה למנוע פרצות נתונים. גם אם תוקף יקבל גישה זדונית לרשת, אם מכשיר מוצפן, המכשיר עדיין יהיה מאובטח, מה שהופך את ניסיונות התוקף לצרוך את הנתונים לחסרי תועלת. הצפנה מבטיחה שאף אחד לא יוכל לקרוא תקשורת או נתונים מלבד הנמען המיועד או בעל הנתונים. זה מונע מתוקפים ליירט ולגשת לנתונים רגישים.

### תקנות

הצפנת נתונים מאפשרת לארגונים להגן על נתונים ולשמור על פרטיות בהתאם לתקנות התעשייה ולמדיניות הממשלה. תעשיות רבות, במיוחד אלו בשירותים פיננסיים ובריאות, מקימות כללים מפורשים בנושא הגנת נתונים. לדוגמה, חוק Gramm-Leach-Bliley מחייב מוסדות פיננסיים ליידע את הלקוחות כיצד הנתונים שלהם משותפים וגם כיצד הנתונים שלהם נשארים מוגנים. הצפנה מסייעת למוסדות פיננסיים לעמוד בחוק זה.

שמירה על מידע רגיש

הצפנה תמשיך להיות מאפיין אבטחה מרכזי בכל דבר, החל מצ'אטי וידאו ועד מסחר אלקטרוני ומדיה חברתית. בעיקרון, אם ניתן לשתף או לאחסן מידע, הוא יוצפן. גם ארגונים וגם משתמשים פרטיים ייהנו ממעקב אחר סטנדרטים של הצפנה כדי להבטיח שהנתונים האישיים והמקצועיים שלהם מוגנים מפני שימוש לרעה או פגיעה.

אתגרי הצפנה

תוקפים ימשיכו לתקוף גם כשהם יודעים שנתונים או מכשירים מוצפנים. הם סבורים שעם קצת מאמץ, הם עשויים לעבור. במשך שנים רבות, סיסמאות חלשות שימשו כגורם מניע לתוקפים להמשיך לנסות, שכן תוכנות מתוחכמות מסוימות יכלו במוקדם או במאוחר לגלות סיסמאות.

מתקפות כוח גס כאלה הפכו מתוחכמות יותר, שכן תוקפים מקווים שעל ידי ביצוע אלפי או אפילו מיליוני ניחושים, הם יגלו את המפתח לפענוח. עם זאת, רוב שיטות ההצפנה המודרניות, בשילוב עם אימות רב-שלבי (MFA), עוזרות לארגונים להיות עמידים יותר בפני מתקפות כוח גס

תורת הגרפים

תורת הגרפים היא חקר מבני נתוני גרפים, אשר ממדלים קשרים בין אובייקטים באמצעות קודקודים (צמתים) וקצוות. היא מספקת כלי מועיל לכמת ולפשט את החלקים הנעים של מערכת דינמית, ומאפשרת לחוקרים לקחת קבוצה של צמתים וחיבורים שיכולים להפשט כל דבר, החל מתסריטים של ערים ועד נתוני מחשב וניתוח מסלולים אופטימליים. תורת הגרפים וגרפים משמשים בקשרים של רשתות חברתיות, דירוג היפר-קישורים במנועי חיפוש, מפות GPS כדי למצוא את המסלול הקצר ביותר הביתה ועוד.

מעגל המילטוני:

**מסלול** המילטוני הוא מסלול בגרף בלתי-מכוון העובר בכל צומת בדיוק פעם אחת. **מעגל** המילטוני הוא מסלול בגרף העובר בכל צומת פעם אחת פרט לצומת שממנו יצא (ואז הוא עובר בו בדיוק פעמיים - בהתחלה ובסוף)

**מטריצת סמיכויות**

מטריצת שכנות היא שיטה לייצוג גרף מכוון בעל  $n$  צמתים בעזרת מטריצה ריבועית בגודל  $n \times n$ , שערכי תאיה הם 0 או 1. תא  $(i,j)$  בגרף מתאר את קיומה (או העדרה) של הקשת המכוונת מקודקוד  $i$  לקודקוד  $j$  בגרף. אם אין קשת כזו, הערך בתא במטריצה יהיה 0. אם יש קודקוד כזה, אזי הערך יהיה 1. פעמים רבות דנים בתורת הגרפים בגרפים פשוטים;

מטריצת שכנויות של גרף לא מכוון היא סימטרית. זאת משום שבגרף לא מכוון אין כיוון לקשתות ולכן, אם קיימת קשת מ- $i$  ל- $j$  אזי קיימת קשת מ- $j$  ל- $i$ . גרפים שאינם מכוונים נהוג לשמור רק את המשולש העליון של מטריצת השכנויות.

עבור גרף שבו אין לולאות עצמיות (כלומר, קשת מקודקוד כלשהו לעצמו), האלכסון הראשי של מטריצת השכנויות המתאימה יהיה 0.

מטריצת השכנויות של גרף שלם תהיה המטריצה בה כל התאים בעלי הערך 1, פרט לאלכסון הראשי. מטריצת השכנויות של גרף חסר קשתות תהיה מטריצת האפס.

בגרפים שאינם פשוטים, ייתכנו מספר קשתות מקבילות בין אותו זוג צמתים. במקרה זה, ניתן לבנות מטריצת שכנויות שערך התא במטריצה יהיה מספר הקשתות מ-i ל-j.

מחולל מספרים פסאודו אקראי

**מחולל מספרים פסאודו-אקראיים קריפטוגרפיים** (Cryptographically secure pseudorandom number generator) או בקיצור CSPRNG הוא מחולל פסאודו אקראי עם תכונות המתאימות במיוחד לשימוש בקריפטוגרפיה. מחולל פסאודו אקראי מהווה מרכיב חיוני ובלתי נפרד במערכות אבטחת מידע.

דוגמא ל מחולל מבוסס אריתמטיקה מודולרית

Blum-Blum-Shub בקיצור BBS הוא מחולל PRNG חזק, מבוסס אריתמטיקה מודולרית שפותח על ידי מנואל בלום, לינור בלום ומייקל שוב. בוחרים שלם  $n$  שהוא כפולה של שני מספרים ראשוניים גדולים וסודיים באופן שקשה לפרק את  $n$  לגורמים וגרעין התחלתי כלשהו. בכל פעם שנדרשת סיבית אקראית הפלט יהיה הסיבית הכי פחות חשובה של תוצאת העלאה בריבוע מודולו  $n$  של הערך הפנימי. האלגוריתם הוכח כבטוח, כלומר ניחוש הסיבית הבאה קשה לפחות כפירוק לגורמים של  $n$ .

## תיאור מצב קיים

### 1Password

מנהל סיסמאות שקיים כבר מ-2006 ונחשב בפי רבים בתחום כמוצלח ביותר. ב-1Password מתגאים בכך שהמידע שתחליטו לאחסן במערכת שלהם מוצפן מקצה לקצה בתקן AES 256bit. בנוסף, המערכת של 1Password יוצרת גם "מפתח סודי" שיחד עם הסיסמה הראשית ייתנו את הגישה לכל המידע.

היתרון, מדובר בדרך מאוד מאבטחת לשמור על המידע - כש-1Password מציעים להדפיס ולשמור את ה"מפתח הסודי" במקום נוח. החיסרון, במקרה שבו תאבד הגישה לאותו צירוף של אותיות ומילים, החברה לא מחדשת אותו - והמידע נשאר נעול לעד. ל-1Password אפליקציה נוחה מאוד שמתממשת גם עם נעילה ביומטריה ומאפשרת לחסוך את הזנת הסיסמה בכל פעם שרוצים לגשת אליה. היא מתממשת עם ממשק המחשב של התוכנה ואחד הפיצ'רים שלה הוא היכולת לשמש כאפליקציית אימות דו-שלבי כמו Google Authenticator - הכל במקום אחד.

### NordPass

NordPass שם דגש על אבטחת מידע, קלות שימוש, הצפנה מתקדמת ברמת XChaCha20, ומדיניות של הגנה על פרטיות המשתמשים. מילוי אוטומטי של טפסים וסיסמאות, תמיכה ברוב מערכות ההפעלה, חוויית משתמש פשוטה וקלה לתפעול

### Dashlane

Dashlane הוא מנהל סיסמאות המציע אבטחה מתקדמת עם הצפנה ברמה גבוהה, וממשק משתמש נוח המאפשר ניהול סיסמאות, מילוי טפסים אוטומטי, ושירות VPN מובנה. השירות תומך בכל הפלטפורמות ומספק חוויית שימוש מהירה ובטוחה, הצפנה מתקדמת שמעניקה אבטחה ברמה גבוהה, ניטור קבוע של רמת האבטחה בחשבונות, כלים ייחודיים לחיזוק רמת האבטחה.

## ניתוח חלופות מערכת

אלגוריתמים ידועים המשמשים להצפנה:

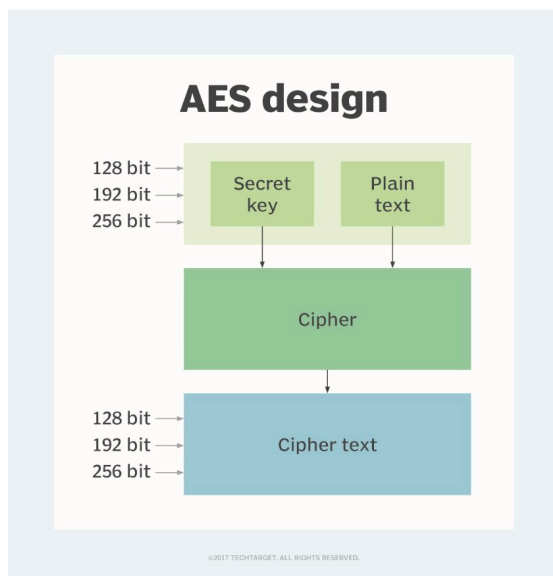
### DES

ממשלת ארה"ב פיתחה אלגוריתמים של DES לפני יותר מ-40 שנה כדי להבטיח שכל המערכות הממשלתיות ישתמשו באותו תקן מאובטח כדי להקל על קישוריות.

DES שימש כעמוד התווך של הקריפטוגרפיה הממשלתית במשך שנים עד 1999, כאשר חוקרים שברו את מפתח ה-56 סיביות של האלגוריתם באמצעות מערכת מחשב מבוצרת.

DES הוא אלגוריתם הצפנה הנמצא בשימוש נרחב בתעשיית הפיננסים. הוא משפר את צופן הבלוקים הקודם ומביא יתרונות שונים, כגון קלות המימוש שלו בחומרה ובתוכנה והתמיכה הנרחבת שלו בקרב ספריות ופרוטוקולים קריפטוגרפיים.

### AES



AES הוא אלגוריתם הצפנה סימטרי מסוג צופן בלוקים. המשמעות היא שהוא משתמש באותו מפתח להצפנה ולפענוח של הנתונים – גם השולח וגם המקבל חייבים להכיר את אותו מפתח סודי ולהשתמש בו. בכך הוא שונה מאלגוריתמים אסימטריים, שבהם נעשה שימוש בשני מפתחות שונים – אחד להצפנה ואחר לפענוח.

כצופן בלוקים, AES מחלק את ההודעה לבלוקים קטנים, ומצפין כל בלוק בנפרד. תהליך זה ממיר את הודעת הטקסט הרגיל לטקסט מוצפן – כלומר, צורה שאינה קריאה או מובנת ללא מפתח הפענוח המתאים.

AES משתמש במספר מפתחות קריפטוגרפיים, שכל אחד מהם עובר מספר סבבי הצפנה כדי להגן טוב יותר על הנתונים ולהבטיח את סודיותם ושלמותם. ניתן להשתמש בכל אורכי המפתחות כדי להגן על מידע ברמה סודית וסודית. באופן כללי, AES-128 מספק אבטחה והגנה נאותים מפני התקפות כוח גס עבור רוב יישומי הצרכנים. מידע

המסווג כסודי ביותר - למשל, מידע ממשלתי או צבאי - דורש אבטחה חזקה יותר המסופקת על ידי אורכי מפתח של 192 או 256 סיביות, שגם הם דורשים יותר כוח עיבוד ויכולים לקחת זמן רב יותר לביצוע.

כיצד פועלת הצפנת AES

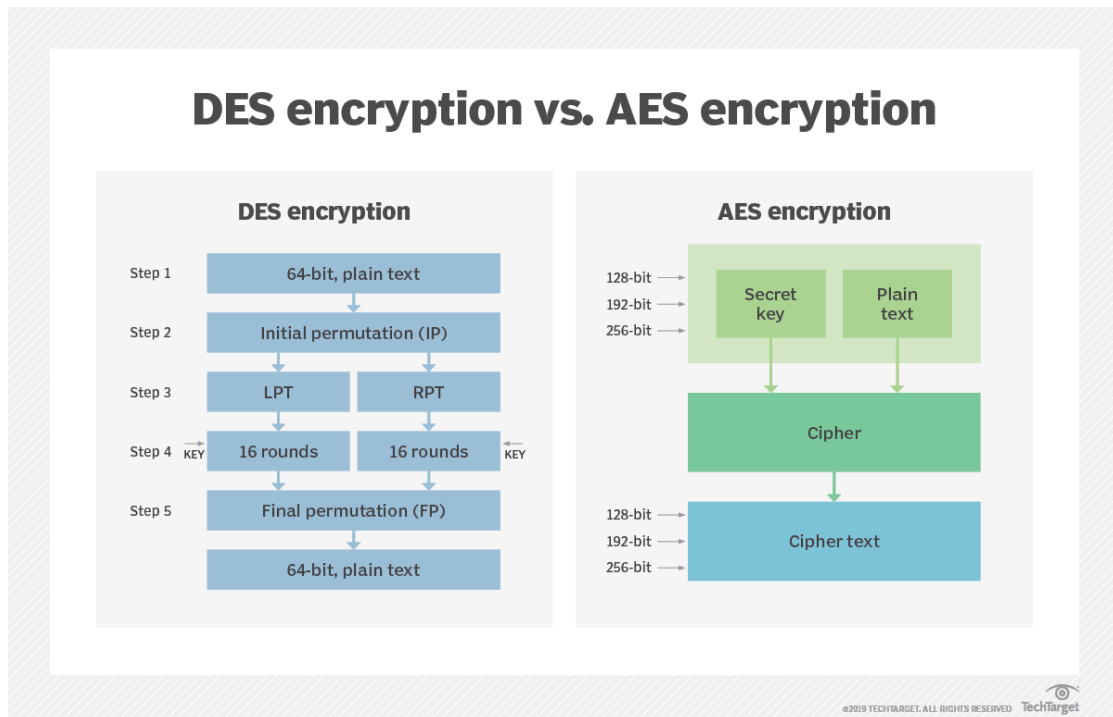
AES כולל שלושה צופני בלוקים או מפתחות קריפטוגרפיים:

1. AES-128 משתמש באורך מפתח של 128 סיביות כדי להצפין ולפענח בלוקים של הודעות.
2. AES-192 משתמש באורך מפתח של 192 סיביות כדי להצפין ולפענח בלוקים של הודעות.
3. AES-256 משתמש באורך מפתח של 256 סיביות כדי להצפין ולפענח בלוקים של הודעות.

כל צופן מצפין ומפענח נתונים בבלוקים של 128 סיביות באמצעות מפתחות קריפטוגרפיים של 128, 192 ו-256 סיביות, בהתאמה. מפתחות ה-128, 192 ו-256 סיביות עוברים 10, 12 ו-14 סבבי הצפנה, בהתאמה. סבב מורכב ממספר שלבי עיבוד כולל החלפה, טרנספוזיציה וערבוב של קלט הטקסט הרגיל כדי להפוך אותו לפלט הטקסט המוצפן הסופי. ככל שיש יותר סיבובים, כך קשה יותר לפצח את ההצפנה, והמידע המקורי בטוח יותר.

ב-AES, טרנספורמציות רבות מבוצעות על נתונים. ראשית, הנתונים מוכנסים למערך, ולאחר מכן טרנספורמציות הצופן חוזרות על עצמן על פני מספר סבבי הצפנה. הטרנספורמציה הראשונה היא החלפת נתונים באמצעות טבלת החלפה וצופן מוגדר מראש. בהמרה השנייה, כל שורות הנתונים מוזזות בשורה אחת מלבד השורה הראשונה. הטרנספורמציה השלישית מערבבת עמודות באמצעות צופן Hill. הטרנספורמציה האחרונה מתבצעת בכל עמודה, או בלוק נתונים, באמצעות חלק אחר או חלק קטן ממפתח ההצפנה. מפתחות ארוכים יותר זקוקים לסיבובים נוספים כדי להשלים.

במהלך הפענוח, נמען ההודעה משתמש בעותק של הצופן כדי להסיר את שכבות ההצפנה השונות ולהמיר את הטקסט המוצפן בחזרה לטקסט רגיל. לאחר החזרה בתשובה, הם יכולים לקרוא את ההודעה, בידיעה שהיא לא יורטה או נקראה על ידי אף אחד אחר.



## הצפנת אל גמאל (ElGamal encryption)

הצפנת אל גמאל (ElGamal encryption) היא שיטת הצפנה אסימטרית אקראית שהומצאה ב-1984 על ידי טאהר אל-גמאל, קריפטוגרף אמריקאי ממוצא מצרי. בדומה לפרוטוקול דיפי-הלמן, ביטחונה מסתמך על הקושי המשוער שבבעיית הלוגריתם הבדיד ובעיית דיפי-הלמן. והיא יכולה לשמש הן להצפנה והן לחתימה דיגיטלית. צופן אלגמאל היה בשימוש בתוכנות חופשיות GPG וכן PGP ומערכות אחרות. אלגוריתם חתימה דיגיטלית DSA הוא וריאציה של חתימה דיגיטלית של אל-גמאל.

האלגוריתם מספק אקראיות בהצפנה, כך שהצפנה של אותו מסר תיתן תוצאה שונה בכל פעם. הוא משמש לעיתים קרובות להצפנת מפתחות סימטריים בשל יעילותו.

## אלגוריתם מבוסס תורת הגרפים וצופן בלוקים:

אלגוריתם זה מבוסס על צופן בלוקים סימטרי המשלב את העקרונות של תורת הגרפים לייצוג הודעות טקסט. בשלב קדם-הצפנה, הטקסט המקורי מחולק לבלוקים, מומר לערכי ASCII, ותתי-בלוקים אלו מיוצגים באמצעות מעגלים המילטוניים זרים בגרף, כאשר ערכי ה-ASCII הופכים למשקלי הקשתות. גרף זה מומר למטריצת שכנויות, המשמשת כקלט לשלב ההצפנה העיקרי.

ההצפנה מתבצעת בשיטת צופן בלוקים בשרשור (CBC), וכוללת שילוב (באמצעות XOR) של מטריצת השכנות המייצגת את הבלוק הנוכחי עם בלוק הצופן הקודם (או מטריצת אתחול עבור הבלוק הראשון), ולאחר מכן שילוב נוסף עם מפתח-משנה ייעודי שנוצר לכל בלוק באמצעות מחולל ייחודי המבוסס על מפתח ראשי ומחולל

פסאודו-אקראי (BBS). תהליך זה שואף להשיג רמה גבוהה של אבטחה תוך שמירה על ביצועי עיבוד נתונים טובים, תהליך הפענוח הינו היפוך של תהליך ההצפנה.

שפת פיתוח וסביבת עבודה:

שפת #C -

**יתרונות:** אבטחה מובנית גבוהה (Managed Code)

תמיכה רחבה בקריפטוגרפיה (כגון Hashing, Symmetric/Asymmetric encryption)

**חסרונות:** פחות גמישה לעבודה על מערכות מבוזרות קלות (כמו Node.js)

תלוייה לעיתים במערכת ההפעלה Windows (אם לא עובדים עם .NET Core)

Python -

**יתרונות:** פופולרית מאוד בלמידת מכונה, פיתוח מהיר, תיעוד עשיר.

**חסרונות:** ביצועים איטיים יחסית, פחות טובה למערכות שדורשות ממשק גרפי עשיר, פחות טיפוסיות

סביבת עבודה:

1. Visual Studio

IDE חזק מאוד מבית Microsoft, עם תמיכה מלאה ב-#C ובפלטפורמת .NET.

**יתרונות:**

- כלי דיבוג (Debugging) מהמתקדמים בעולם
- תמיכה ב-Windows Forms / WPF לפיתוח ממשק גרפי
- ספריות קריפטוגרפיה מוכנות לשימוש
- ניהול פרויקטים בצורה נוחה מאוד
- Intellisense חכם, תבניות קוד, בדיקות יחידה ועוד

**חסרונות:**

- יחסית כבד (דורש משאבים)
- מתאים בעיקר לפיתוח על Windows

2. Visual Studio Code



עורך קליל, מבוסס הרחבות, תומך במגוון שפות.

**יתרונות:** מהיר, פתוח וחינמי

**חסרונות:** פחות עוצמתי מ-VS המלא, לא מותאם לעבודה מלאה ב-C# ללא הרחבות

PyCharm / Eclipse / IntelliJ .3

סביבות עבודה טובות לשפות אחרות (Python, Java וכו'), אך לא מותאמות לפיתוח ב-C#.

## תיאור החלופה הנבחרת

האלגוריתם מיישם מודל הצפנה סימטרי חדשני המבוסס על תורת הגרפים לאבטחת סיסמאות במנהל סיסמאות מתקדם. הגישה החדשה מנצלת את התכונות המתמטיות של מעגלים המילטוניים זרים בגרף שלם ליצירת מטריצת הצפנה דו-ממדית ששומרת את המידע בצורה אופטימלית.

**עיבוד נתונים ראשוני:** הסיסמה המקורית (25-8 תווים) עוברת תהליך הכולל הוספת קידומת אורך לשחזור מדויק, ומילוי במלח קריפטוגרפי אקראי להגעה לבלוק אחיד של 78 תווים. תהליך זה מבטיח אחידות בעיבוד ומונע ניתוח דפוסים על בסיס אורך הסיסמה המקורית.

**המודל הגרפי:** הבלוק המעובד מתחלק ל-6 תתי-בלוקים של 13 תווים כל אחד, כאשר כל תת-בלוק ממופה למעגל המילטוני ייחודי בגרף שלם מסדר 13. המעגלים הזרים (edge-disjoint) מקיימים את התכונה המתמטית שכל קשת בגרף משויכת למעגל יחיד בלבד, ובכך מבטיחים בידוד מושלם בין תת-בלוקים ומונעים דליפת מידע צולבת. מספר המעגלים הזרים נקבע על פי הנוסחה  $(n-1)/2$  עבור גרף שלם עם  $n$  קודקודים אי-זוגי, ומניב 6 מעגלים עבור  $n=13$ .

**יצירת מטריצת הסמיכות:** ערכי ASCII של התווים בכל תת-בלוק משמשים כמשקלי הקשתות במעגל המילטוני המתאים, ויוצרים מטריצת סמיכות  $13 \times 13$  המכילה את כל המידע המוצפן. המטריצה מייצגת גרף שבו המידע הרגיש מוטמע במשקלי הקשתות ולא במבנה הגרף עצמו, מה שמאפשר שילוב של יעילות חישובית עם ביטחון קריפטוגרפי.

**שכבות ההצפנה:** המטריצה הגרפית עוברת הצפנה דו-שכבתית במודל Cipher Block Chaining (CBC)). השכבה הראשונה כוללת פעולת XOR עם מטריצת אתחול לשבירת דפוסים, והשכבה השנייה מבצעת XOR עם תת-מפתח ייחודי הנוצר באמצעות מחולל (Blum-Blum-Shub (BBS. התת-מפתח מתבסס על תו שנבחר אקראית מהבלוק ועל מפתח ראשי (KEK), ומבטיח ייחודיות קריפטוגרפית לכל סיסמה.

**פענוח ושחזור:** תהליך הפענוח מבצע את השלבים בסדר הפוך, תוך שימוש בוקטור מיקומים (VP) השמור במסד הנתונים. המערכת משחזרת את התת-מפתחות, מבטלת את שכבות ה-XOR, מחלצת את התווים המקוריים מהמעגלים ההמילטוניים, ומסירה את המלח הקריפטוגרפי לקבלת הסיסמה המקורית.

#### שפה:

הבחירה ב-**C#** כבסיס לפיתוח נבעה מהשילוב שבין שפה מודרנית, יציבה ובטוחה, לבין יכולות מתקדמות בתחום ההצפנה ואבטחת המידע.

#### סביבת העבודה:

Visual Studio מספקת סט כלים מקצועי שמאפשר פיתוח נוח, מהיר ומסודר - במיוחד למערכת מורכבת כמו מנהל סיסמאות. הבחירה בה מהווה יתרון משמעותי ביציבות, אבטחה ויעילות פיתוח.

### תהליך האלגוריתם:

האלגוריתם מורכב משלושה חלקים עיקריים:

1. יצירת מפתחות
2. תהליך ההצפנה
3. תהליך פענוח

#### 1. יצירת מפתחות

שלבי יצירת מפתח משני  $K_i$  בזמן הצפנה:

1. קבלת בלוק - סיסמה להצפנה.
2. בחירת תו אקראי Char מהבלוק הנוכחי  $Block_i$ .
3. שימוש בערך ה-ASCII של התו :  
 - נכניס את הערך ה-ASCII של התו לוקטור המיקומים VP (שישמש מאוחר יותר לפענוח).  
 - משתמשים בערך זה כאינדקס כדי לשלוף ערך N ממפתח האב KEK. (בגודל 256).
4. יצירת וקטור **Si** : משתמשים בערך N כזרע למחולל המספרים הפסאודו-אקראיים כדי ליצור וקטור Si בגודל 13.
5. יצירת מפתח משני **Ki** : משתמשים בוקטור Si כדי ליצור מפתח משני  $K_i$  שהוא מטריצה ריבועית מסדר 13.

יצירת מפתחות מחדש בזמן פענוח:

1. **שחזור מפתח** : משתמשים בוקטור המיקומים VP כדי ליצור מפתח מפתח בגודל  $13^2 \times$  מתוך מפתח האב KEK.

2. **חלוקת המפתח** : מחלקים את המפתח לתתי מפתחות  $S_i$  בגודל 13.
3. **יצירת מפתחות משניים** : משתמשים בכל וקטור  $S_i$  כדי ליצור מפתח משני  $K_i$  שהוא מטריצה ריבועית מסדר 13.

### 3. תהליך ההצפנה

#### שלב הקדם-הצפנה:

1. **המרת הטקסט הגלוי לערכי ASCII** : הטקסט הגלוי מומר לערכי ASCII.
2. **חלוקה לבלוקים** : הטקסט מחולק לבלוקים בגודל 13 תווים ( $Block_i$ ). משתמשים בנוסחה:  $n = 13k + r$ , כאשר  $n$  הוא גודל הטקסט הגלוי,  $r$  השארית, ו- $k$  מספר הבלוקים.
3. **המרה למעגלי המילטון** : כל בלוק מומר למעגל המילטון, כאשר המשקלים של הקשתות בגרף  $G_i$  מיוצגים ע"י ערכי ה-ASCII של התווים.
4. **המרה למטריצת סמיכות** : הגרף הנוצר מומר למטריצת סמיכות  $M_i$ .

צופן הבלוקים:

1. שימוש ב-Cipher Block Chaining (CBC) - שרשור בלוקים:  
האלגוריתם משתמש בשיטת CBC, כך פענוח כל בלוק תלוי בבלוק המוצפן הקודם.
2. פעולת XOR ראשונה :  
  - עבור הבלוק הראשון:  $M'_0 = IM \oplus M_0$  (כאשר  $IM$  היא מטריצת אתחול אקראית)
  - עבור שאר הבלוקים:  $M'_i = C_{i-1} \oplus M_i$  (כאשר  $C_{i-1}$  הוא הבלוק המוצפן הקודם)
3. פעולת XOR שנייה :  $C_i = M'_i \oplus K_i$  (כאשר  $K_i$  הוא המפתח המשני)
4. **יצירת וקטור מהמטריצה המוצפנת** : שורות המטריצה  $C_i$  מחוברות ליצירת וקטור  $eBlock_i$  בגודל  $13^2$ .
5. **יצירת ההודעה המוצפנת הסופית** : כל הווקטורים  $eBlock_i$  מחוברים ליצירת וקטור אחד  $EM$  בגודל  $k \times 13^2$ .

### 4. תהליך פענוח

תהליך הפענוח הוא תהליך ההצפנה בסדר הפוך:

1. **פירוק ההודעה המוצפנת** : ההודעה המוצפנת  $EM$  מומרת לתווי אסקי,
2. המרת ההודעה למעגל המילטוני המיוצג בגרף
3. המרת הגרף למטריצת סמיכות

4. **שחזור המפתחות המשניים** : המפתחות המשניים  $K_i$  משוחזרים באמצעות מפתח האב KEK והווקטור VP.
5. **פעולת XOR ראשונה** :  $M'_i = C_i \oplus K_i$
6. **פעולת XOR שנייה** :
  - עבור הבלוק הראשון:  $M_0 = IM \oplus M'_0$
  - עבור שאר הבלוקים:  $M_i = C_{\{i-1\}} \oplus M'_i$
7. **שחזור הבלוק המקורי** : הגרף  $G_i$  מומר בחזרה לבלוק  $Block_i$ .
8. **שחזור ההודעה המקורית** : שרשור הבלוק  $Block_i$  המפוענח.

## ייחודיות האלגוריתם

האלגוריתם מבוסס על יישום מלא של כל המעגלים ההמילטוניים הזרים בגרף שלם מסדר 13. מעגלים זרים הם מעגלים שאינם חולקים קשתות משותפות, כאשר כל קשת בגרף שייכת למעגל יחיד בלבד. מספר המעגלים הזרים המקסימלי בגרף שלם עם  $n$  קודקודים אי-זוגי הוא  $(n-1)/2$ , ולכן עבור גרף בן 13 קודקודים ניתן לבנות 6 מעגלים זרים.

האלגוריתם משלב גנרטור מספרים פסאודו-אקראיים מסוג Blum Blum Shub (BBS) ליצירת תת-מפתחות מאובטחים, כאשר הביטחון מבוסס על מורכבות פירוק המספר  $n=pq$  למכפלת שני מספרים ראשוניים גדולים. הגנרטור מקבל זרע התחלתי מתוך הטקסט עצמו דרך בחירה אקראית של תו מכל בלוק, מה שיוצר תלות דינמית בין המפתח לבין התוכן המוצפן.

## אפיון המערכת

ניתוח דרישות המערכת

סביבת פיתוח: visual studio

חומרה: i7

עמדת פיתוח: njac bhhs

מערכת הפעלה: Windows 11

מסד נתונים: MongoDB

חיבור לרשת: wifi

תוכנות: Visual Studio 2022 , Google Chrome, MongoDB Compass , VScode.

## מודול המערכת

המערכת עוסקת בניהול מאובטח של סיסמאות אישיות. היא כוללת שמירה והצפנה של סיסמאות, ממשק משתמש להזנה וצפייה בסיסמאות, ומנגנון כניסה מאובטח. המערכת אינה כוללת ממשק שיתוף סיסמאות, סנכרון בין מכשירים שונים, או ניתוח סיכוני אבטחה של הסיסמאות.

## אפיון פונקציונלי

---

- הרשמה
- התחברות
- הצפנה
- פענוח

## ביצועים עיקריים

---

האלגוריתם המרכזי במערכת הוא אלגוריתם הצפנה סימטרית מבוסס צופן בלוקים ותורת הגרפים. פעולת ההצפנה והפענוח אורכת בממוצע פחות מ-100 מילישניות עבור מחרוזת באורך ממוצע של 16 תווים. האלגוריתם שומר על ביצועים יציבים גם בנפחי נתונים גדולים, ומאפשר עבודה בזמן אמת עם חוויית משתמש תקינה.

## אילוצים

---

אבטחה חזקה - חובה שכל הסיסמאות תהיינה מוצפנות  
פענוח- פענוח מדויק של הסיסמא המוצפנת  
ממשק פשוט ונוח גם למשתמשים לא טכניים  
שמירה על ביצועים טובים גם במחשבים חלשים

## תיאור הארכיטקטורה

---

הארכיטקטורה של הפתרון המוצע בפורמט של Design level Down-Top :

המערכת בנויה בארכיטקטורה רב-שכבתית (Multi-layered Architecture).

### MongoDB - בסיס הנתונים

- משמש לאחסון מסמכים בפורמט JSON.
- כל פריט במערכת נשמר כ-Document בקולקציה.

### (DAL) Data Access Layer

- אחראי לגישה פיזית לבסיס הנתונים.
- מממש את הממשקים 'IDAL', הכוללים קריאה, כתיבה, עדכון ומחיקה של ישויות.
- משתמש ב-Drivers של MongoDB.

### (DTO) Data Transfer Objects

- אובייקטים פשוטים שמעבירים מידע בין שכבות.
- מכילים רק שדות נתונים - ללא לוגיקה.
- משמשים למניעת תלות בין מבני הנתונים של המערכת לממשק ה-API.

### שכבת לוגיקה עסקית - (BL) Business Logic Layer

- מרכז הלוגיקה העסקית של האפליקציה.
- מממש את `IBL`.
- אחראי לאימותים, חישובים, ולוגיקה עסקית (למשל, בדיקות או עיבוד נתונים).
- מקבל ומחזיר DTO בלבד.

### API) .NET Core

- שכבת קצה לשרת.
- כוללת Controllers שמטפלים בבקשות HTTP.
- מתרגמת בקשות מ-React ומחזירה תשובות.
- משתמשת ב-BL לביצוע פעולות.

### React - ממשק המשתמש

- אפליקציית Single Page Application (SPA).
- שולחת בקשות ל-API ומציגה את המידע למשתמש.
- בנויה מ-Components ומנהלת State באמצעות useState/useEffect.

הארכיטקטורה שומרת על עקרונות של הפרדה ברורה, תחזוקה קלה, והרחבה פשוטה בעתיד. כל שכבה תלויה רק בשכבה שמתחתיה ונבדקת באופן מבודד (Testable & Decoupled).

## תיאור הרכיבים בפתרון (שרתי DB, שרתי תקשורת, שרת יישום/ לקוח)

---

react

ריאקט היא ספריית ג'אווה סקריפט חינומית בקוד פתוח, אשר פותחה על ידי חברת פייסבוק, ומאפשרת למתכנתים ליצור אתרי אינטרנט ואפליקציות דינאמיות, היא ספריה המאפשרת למתכנתים לבנות ממשקי תצוגה אינטואיטיביים בצורה קלה ומהירה. ריאקט הפופולרית מאפשרת למפתחים ליצור יישומי אינטרנט גדולים שיכולים לבצע שינוי נתונים מבלי צורך בטעינה מחודשת של העמוד שמטרתה העיקרית היא מהירות.

ריאקט מקלה באופן משמעותי על בנייה ויצירה של יישומים דינאמיים מהסיבה שהיא דורשת פחות קידוד ומציעה יותר פונקציונליות בניגוד ל JavaScript בה הקידוד עלול להיות מורכב יותר.

react עושה שימוש ב virtual dom וככה יוצרת את היישומים מהר יותר. virtual dom משווה את המצבים הקודמים של הרכיבים ומעדכנת רק את הפריטים ב virtual dom שוונו במקום לעדכן שוב את כל הרכיבים כפי שעושים ביישומי אינטרנט רגילים.

הרכיבים הם הבסיס לכל יישום react ויישומים מורכבים לרוב ממספר רכיבים. לרכיבים אלה יש את ההגיון שלהם, וניתן לעשות בהם שימוש חוזר בכל יישום מה שמקטין באופן דרמטי את זמן פיתוח היישום.

לסיום ריאקט היא אחת הספריות הפופולריות ביותר בזכות היתרונות הרבים שהיא מציעה וגם בזכות הקהילה הגדולה סביבה.

## MongoDB

MongoDB היא חברה שמפתחת מוצרים בתחום ה-Database, כאשר המוצר המרכזי שלה הוא מסד נתונים מסוג NoSQL. השם MongoDB מורכב מהמילה Humongous, מה שמשדר שמדובר במערכת עצומה שנועדה לטפל בכמויות גדולות של נתונים ולעבוד איתם ביעילות רבה.

החברה הוקמה בשנת 2007 ומאז הפכה לאחת המובילות בתחום מסדי הנתונים החדשניים, עם מאות אלפי התקנות ברחבי העולם ושימוש נרחב בקרב חברות הטכנולוגיה הגדולות והחדשניות. החברה גם מציעה שירותי ענן מתקדמים דרך MongoDB Atlas, המאפשר הפעלה וניהול של מסדי נתונים בענן בצורה אוטומטית ובטוחה.

### היתרונות של MongoDB

מדוע MongoDB כל כך פופולרית? MongoDB מציעה מספר יתרונות מרכזיים שהופכים אותה למועדפת בקרב מפתחים וחברות טכנולוגיה. הנה חלק מהיתרונות:

#### תמיכה ב-Embedded Documents

מסד הנתונים MongoDB מאפשר שימוש במסמכים מוטבעים (Embedded Documents), שבאמצעותם ניתן לבנות מבני נתונים מורכבים בצורה יעילה וברורה. טכניקה זו מסייעת לארגן את המידע באופן הגיוני, במיוחד במודל נתונים של יחיד ליחיד (One-to-One), כאילו היו אובייקטים בשפת תכנות.

היכולת להשתמש ב-Embedded Documents מעלה משמעותית את מהירות הביצועים בזמן שליפת נתונים וכתובתם למסד הנתונים, מה שמהווה יתרון בולט בפיתוח אתרים ואפליקציות.

בנוסף, MongoDB תומך גם בשימוש ב-Reference Documents, שמאפשרים ליצור קישורים (references) בין נתונים במסמכים שונים. שימוש זה חשוב במיוחד למבני נתונים שבהם יש צורך לשמור על קשרי תלות בין אלמנטים שונים, ולכן מוסיף גמישות רבה לעיצוב וניהול הנתונים.

### תמיכה ב-BSON Format

מסד הנתונים MongoDB משתמש בפורמט BSON לשמירת ולעיבוד הנתונים. BSON הוא גרסה בינארית של JSON, כלומר (Binary JSON), שמספקת יתרונות רבים. פורמט זה מאפשר אחסון יעיל יותר של סוגים מורכבים של נתונים כגון תאריכים ומערכים, ומסייע בביצוע גישה מהירה ויעילה לנתונים.

הבינאריות של BSON מאפשרת גם ל-MongoDB לבצע שאילתות וסינון של הנתונים בצורה מהירה ובכך לשפר את ביצועי המערכת במיוחד במערכות עם עומסי עבודה גבוהים ודרישות לטיפול מהיר בכמויות נתונים גדולות.

### תמיכה ב-GeoSpatial Indexing

מסד הנתונים MongoDB מציע תמיכה מובנית לעיבוד ואחסון של נתונים גיאוגרפיים, מה שמאפשר למפתחים לבצע שאילתות הקשורות למיקומים דיי בקלות. לדוגמה, אפשר למצוא את כל המסעדות הנמצאות במרחק של 5 קילומטרים ממקום מסוים. זה מאוד שימושי לאפליקציות כמו תכנון נסיעות או שירותי מפות, שבהן צריך לנהל מידע שקשור למיקומים ולהציע תוצאות מהירות ומדויקות למשתמש.

### MongoDB Atlas

MongoDB Atlas הוא אחד ממוצרי הדגל של חברת MongoDB. זהו שירות מנוהל בענן שמספק למפתחים פתרון יעיל לניהול מסד נתונים בלי צורך בהתעסקות טכנית מורכבת. שירות זה כולל אוטומציה למשימות ניהוליות כמו גיבויים ושחזור, מה שמאפשר למפתחים להתמקד בפיתוח ולא בתחזוקה.

המוצר Atlas מצויד בכלים מתקדמים לאבטחת נתונים וניהול הרשאות, ומספק בטחון גבוה למידע הארגוני. כמו כן, השירות מותאם לסקלבליות, כך שהוא יכול להתרחב ככל שהדרישות גדלות, מה שמבטיח שמסד הנתונים ימשיך לעבוד באופן אופטימלי בכל סביבת עבודה. במילים פשוטות, MongoDB Atlas פועל כדשבורד לניהול שוטף של מסד הנתונים.

### סקלבליות (Scalability)

מסד הנתונים MongoDB מאופיין בגמישות בהתאמה לכמות הנתונים שניתן להכיל, והוא מתאים הן לפרויקטים קטנים והן לאפליקציות גדולות. תכונה זו מאפשרת למסד הנתונים להתרחב ולצמוח, ולהתאים את עצמו לצרכים השונים של המשתמשים ושל העסק.



**.NET**

EF Core (Entity Framework Core) הוא מסגרת עבודה (Object-Relational - ORM) (Mapper) מודרנית וחזקה מבית מיקרוסופט, המיועדת לפיתוח אפליקציות בפלטפורמת .NET. EF Core מאפשר למפתחים לבצע אינטראקציה עם בסיסי נתונים בצורה מונחית עצמים (Object-Oriented), מה שמפשט משמעותית את תהליך העבודה עם הנתונים. במקום לכתוב שאילתות SQL מורכבות, המפתחים עובדים עם מודלים בשפת #C שמייצגים את הטבלאות והישויות בבסיס הנתונים. EF Core תומך במגוון רחב של מסדי נתונים, כגון SQL Server, SQLite, PostgreSQL, MySQL ועוד, ומאפשר גמישות בבחירת הפלטפורמה המתאימה לפרויקט.

אחד היתרונות המרכזיים של EF Core הוא התמיכה במנגנון Migrations, המאפשר לנהל בצורה מסודרת את השינויים במבנה בסיס הנתונים לאורך זמן - כגון הוספה או שינוי של טבלאות ועמודות - בצורה מבוקרת, תוך שמירה על שלמות הנתונים. בנוסף, EF Core מאפשר למפתחים לכתוב שאילתות באמצעות LINQ (Language Integrated Query), המאפשרת שימוש בסינטקס של שפת #C ליצירת שאילתות בצורה בטוחה ונוחה, אשר מתורגמת אוטומטית לשפת SQL.

EF Core תוכנן להיות קל משקל, מודולרי ומהיר, ומספק ביצועים גבוהים גם באפליקציות בקנה מידה גדול. השימוש בו משפר את הפרודוקטיביות של צוות הפיתוח, מפחית את כמות הקוד הנדרש לניהול הנתונים, ומאפשר התמקדות בלוגיקה העסקית של האפליקציה. בנוסף, EF Core תומך בעקרונות של Dependency Injection ומאפשר אינטגרציה חלקה עם ארכיטקטורות מודרניות כמו Clean Architecture ושכבות שירותים (Layered Architecture).

לסיכום, EF Core מהווה כלי מרכזי ויעיל בפיתוח מערכות דוטנט, המאפשר עבודה נוחה, מאובטחת וגמישה עם בסיסי נתונים תוך חיסכון בזמן ומשאבים.

**visual studio**

Visual Studio היא סביבת פיתוח משולבת (IDE) שפותחה על ידי מיקרוסופט לפיתוח יישומי שולחן עבודה, ממשק משתמש גרפי (GUI), קונסולות, יישומי אינטרנט, יישומים ניידים, ענן ושירותי אינטרנט ועוד. בעזרת IDE זה, ניתן ליצור קוד מנוהל וכן קוד מקורי. הוא משתמש בפלטפורמות שונות של תוכנות פיתוח תוכנה של מיקרוסופט, כמו חנות Windows, Microsoft Silverlight ו-Windows API ועוד. זה לא IDE ספציפי לשפה, שכן ניתן להשתמש בו כדי לכתוב קוד ב-C#, JavaScript, Python, VB (Visual Basic), C++ ושפות רבות נוספות. הוא מספק תמיכה ב-36 שפות תכנות שונות. Windows וגם עבור macOS.

## יתרונות השימוש ב- Visual Studio IDE

Visual Studio IDE, פלטפורמת תכנות מלאה עבור מספר מערכות הפעלה, האינטרנט והענן, זמינה. משתמשים יכולים לגלוש בקלות בממשק המשתמש כדי שיוכלו לכתוב את הקוד שלהם במהירות ובדייקנות.

כדי לעזור למפתחים לזהות במהירות שגיאות פוטנציאליות בקוד, Visual Studio מציע כלי ניפוי שגיאות חזק. מפתחים יכולים לארח את האפליקציה שלהם בשרת בביטחון מכיוון שהם ביטלו כל דבר שעלול להוביל לבעיות ביצועים. לא משנה באיזו שפת תכנות משתמשים מפתחי Visual Studio, משתמשי Visual Studio יכולים לקבל תמיכה בקידוד בזמן אמת. לפיתוח מהיר יותר, הפלטפורמה מציעה אפשרות השלמה אוטומטית. המערכת החכמה המובנית מציעה תיאורים וטיפים עבור ממשקי API.

באמצעות Visual Studio IDE ניתן לשתף פעולה בקלות עם חברי צוות באותו פרויקט. IDE עוזר למפתחים לשתף, לדחוף ולמשוך את הקוד שלהם עם חברי הצוות שלהם. לכל משתמש של Visual Studio יש את היכולת להתאים אותו אישית. יש להם אפשרות להוסיף תכונות בהתאם לצרכים שלהם. לדוגמה, הם יכולים להוריד תוספים ולהתקין הרחבות ב-IDE שלהם. אפילו מתכנתים יכולים להגיש הרחבות משלהם.

## תיאור פרוטוקולי תקשורת

**HTTPS** - זהו הפרוטוקול הנפוץ ביותר לתקשורת מאובטחת בין הלקוח לשרת, והוא מבוסס על SSL/TLS כדי להבטיח שהמידע המועבר מוגן. HTTPS מונע מתקפות "אדם בתווך" (MITM), בכך שהוא מצפין את המידע המועבר.

## שרת לקוח

המערכת פועלת במבנה שרת-לקוח. צד הלקוח נכתב בטכנולוגיית React ופועל בדפדפן, ואילו צד השרת נכתב בשפת #C באמצעות ASP.NET Core.

התקשורת בין השרת ללקוח מתבצעת באמצעות פרוטוקול **HTTPS** המאובטח, תוך שימוש בקריאות API. הנתונים מועברים בין הלקוח לשרת, באופן אסינכרוני בעזרת קריאות fetch או axios מצד הלקוח. השרת מעבד את הבקשות, מבצע פעולות מול בסיס הנתונים (MongoDB), ומחזיר תשובות ללקוח בהתאם.

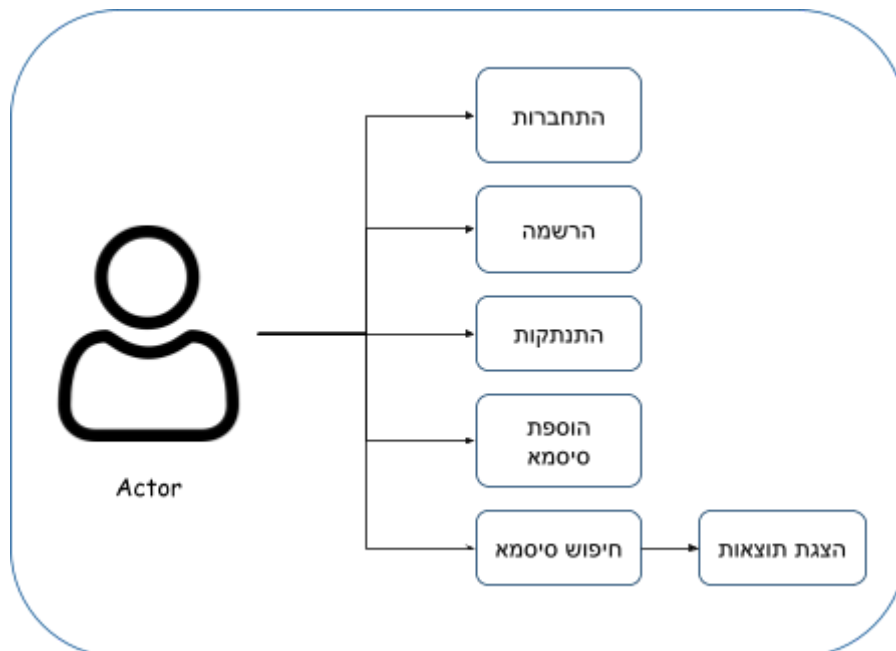
## תיאור תהליכי אבטחת מידע במערכת

## תיאור הצפנות

בפרויקט זה השתמשתי באלגוריתם הצפנה מתקדם שכולל שימוש בתורת הגרפים, צופן בלוקים, מעגלים המילטוניים, מחולל מספרים פסאודו אקראיים להצפנת מפתח המאסטר (- כניסה למערכת) השתמשתי בהצפנה אסימטרית - RSA

## ניתוח ותרשים UML/ useCases למערכת

### Use case Diagram



### Use cases רשימת

**הרשמה** - משתמש חדש נרשם למערכת

**התחברות** - משתמש מזוהה נכנס

**הוספת סיסמה חדשה** - המשתמש שומר סיסמה לשירות כלשהו

**הצגת הסיסמאות** - המערכת מציגה את הסיסמאות

**יציאה מהמערכת** - המשתמש מתנתק מהמערכת

הרשמה:

| שם מקרה שימוש | הרשמה   |
|---------------|---|
| תיאור         | המשתמש מכניס את פרטיו האישיים, לוחץ על הרשמה והמערכת מוסיפה אותו לרשימת המשתמשים. |
| שחקנים        | משתמש   |
| תנאי מקדים    | הכנסת פרטים   |
| הזנק          | הרצון להירשם למערכת   |
| מסלול עיקרי   | המשתמש מכניס את פרטיו למערכת  |
| תנאי סיום     | המערכת הוסיפה את המשתמש   |
| תדירות        | פעם אחת   |

התחברות:

| שם מקרה שימוש | התחברות   |
|---------------|---|
| תיאור         | המשתמש מכניס את מייל וסיסמא, לוחץ על התחברות והמערכת מחפשת את המשתמש, בודקת שהסיסמא תואמת לשם משתמש ומעבירה את המשתמש לדף הבית במידה והפרטים נכונים |
| שחקנים        | משתמש   |
| תנאי מקדים    | הכנסת פרטים   |
| הזנק          | הרצון להתחבר למערכת   |
| מסלול עיקרי   | המשתמש מכניס את פרטיו למערכת  |
| תנאי סיום     | המערכת מצאה את המשתמש   |
| תדירות        | הרבה פעמים  |

הוספת סיסמא:

| שם מקרה שימוש | הוספת סיסמא  |
|---------------|--|
| תיאור         | המשתמש מכניס פרטי סיסמא חדשה, לוחץ על הוספה והמערכת מוסיפה את הסיסמא |
| שחקנים        | משתמש  |
| תנאי מקדים    | הכנסת פרטים  |
| הזנק          | הרצן לשמירת סיסמא חדשה במערכת  |
| מסלול עיקרי   | המשתמש מכניס פרטי סיסמא  |
| תנאי סיום     | המערכת הוסיפה את הסיסמא  |
| תדירות        | הרבה פעמים   |

יציאה מהמערכת:

| שם מקרה שימוש | יציאה מהמערכת              |
|---------------|----------------------------|
| תיאור         | המשתמש לוחץ על התנתקות     |
| שחקנים        | משתמש/מערכת                |
| תנאי מקדים    | לחיצה על כפתור התנתקות     |
| הזנק          | הרצון לצאת מהמערכת         |
| מסלול עיקרי   | המשתמש לוחץ על כפתור יציאה |
| תנאי סיום     | -                          |
| תדירות        | הרבה פעמים                 |

## מבנה נתונים

### גרפים:

מעגל המילטוני המיוצג באמצעות גרף - משמש בתהליך ההצפנה והפענוח. יש לציין שתורת הגרפים נחשבת לתחום מבטיח מאוד בקריפטוגרפיה, וכן רבות מבעיות ה-NP-קשות נגזרות מתורת הגרפים, מה שהוביל ליישום רעיונותיה בקריפטוגרפיה בקנה מידה גדול.

בנוסף במהלך ההצפנה והפענוח אני משתמשת במערכים, מטריצות, ורשימות.

## חישוב יעילות האלגוריתם

האלגוריתם מציג יעילות גבוהה בשל היותו מותאם במיוחד להצפנת סיסמאות. בניגוד לאלגוריתמי הצפנה כלליים המיועדים לטקסטים ארוכים, האלגוריתם עובד עם בלוק יחיד באורך קבוע של 78 תווים, מה שמביא למורכבות זמן קבועה  $O(1)$  עבור כל פעולת הצפנה ופענוח, ללא תלות באורך הסיסמה המקורית.

מבחינת צריכת זיכרון, האלגוריתם דורש שטח קבוע של כ-1KB בלבד (מטריצה  $13 \times 13$  ובלוק נתונים), מה שהופך אותו למתאים במיוחד לסביבות מוגבלות משאבים כמו אפליקציות נייד.

מבחינת efficiency שטח אחסון, כל סיסמה מוצפנת דורשת בסביבות 320-470 בתיים במסד הנתונים (כולל וקטור המיקומים), מה שמאפשר אחסון יעיל של מיליוני סיסמאות.

בסה"כ, האלגוריתם משיג יחס אופטימלי בין עוצמה קריפטוגרפית לבין יעילות מעשית, מה שהופך אותו למתאים לשימוש בסביבות ייצור עם דרישות גבוהות לביצועים ואמינות.

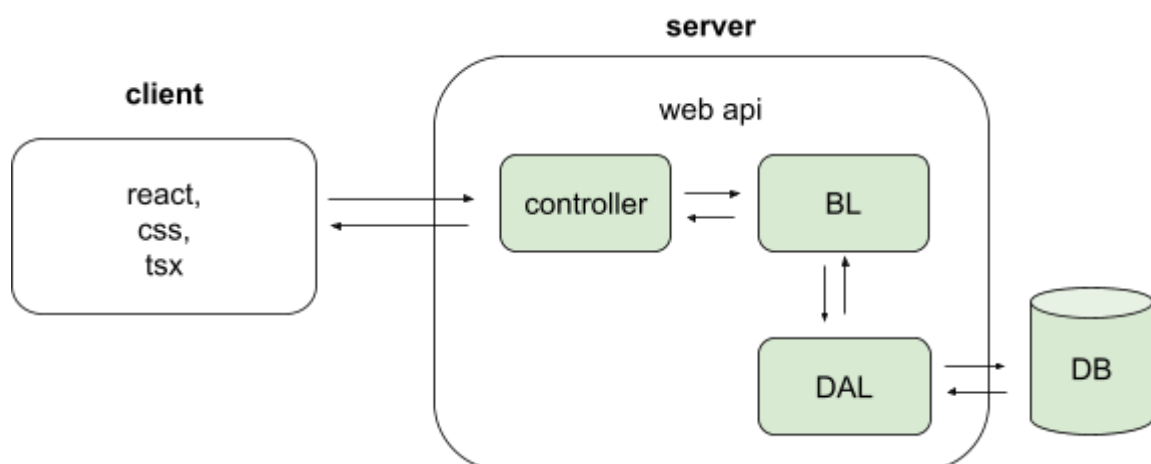
## הקשרים בין היחידות השונות

בפרויקט זה נעשה שימוש בארכיטקטורת שכבות מופרדת (Layered Architecture) הכוללת צד שרת בטכנולוגיית .NET Core, צד לקוח React, ובסיס נתונים MongoDB.

UI → API → BL → DAL → MongoDB

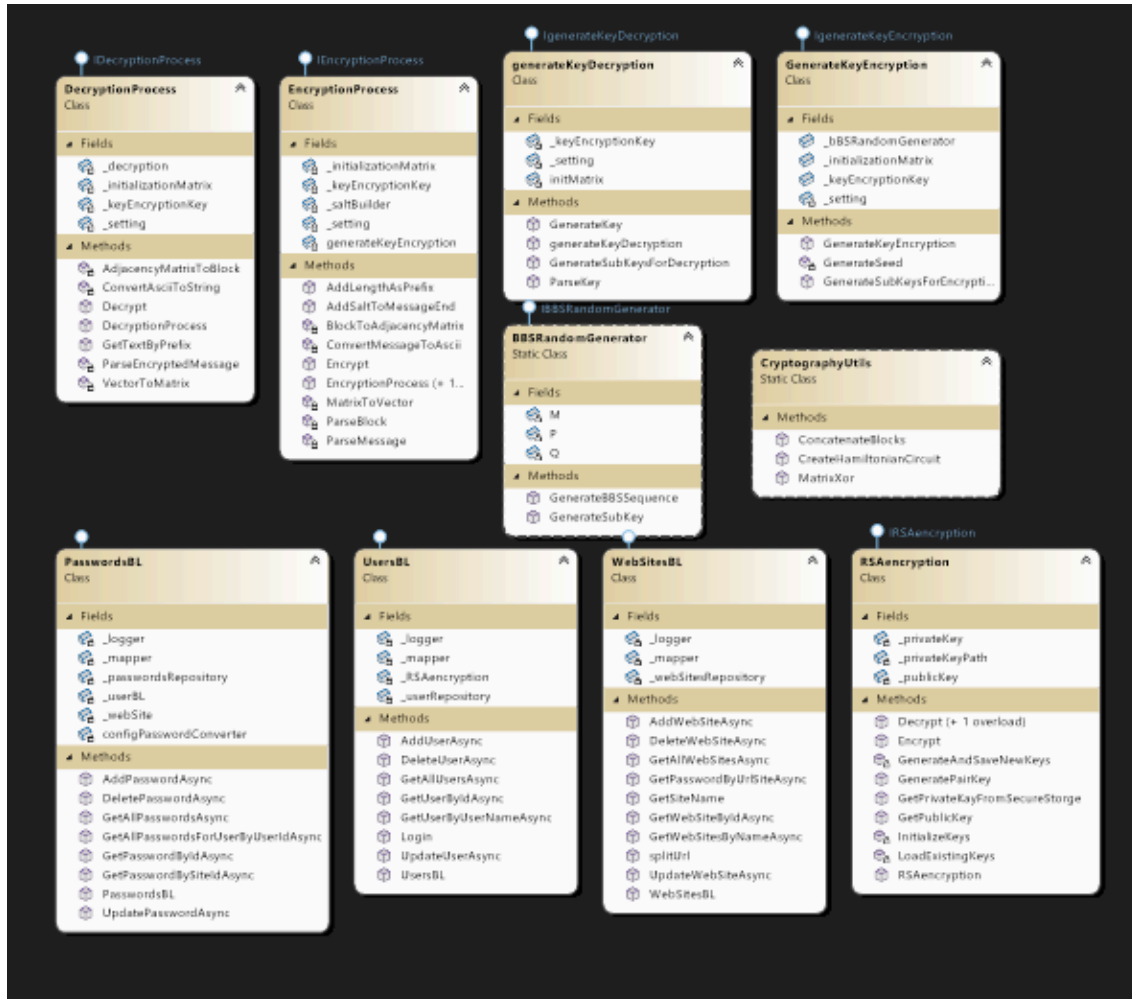
- צד הלקוח (React) מתקשר עם ה-API דרך בקשות HTTP (בעזרת Axios).
- ה-API מקבל את הבקשות ומעביר אותן לשכבת הלוגיקה העסקית (BL).
- הלוגיקה העסקית - BL מבצעת את הפעולות הנדרשות ופונה ל-DAL לשליפת או שמירת נתונים.
- ה-DAL מתממשק עם MongoDB ומחזיר את המידע למעלה עד לצג המשתמש.

## עץ מודולים

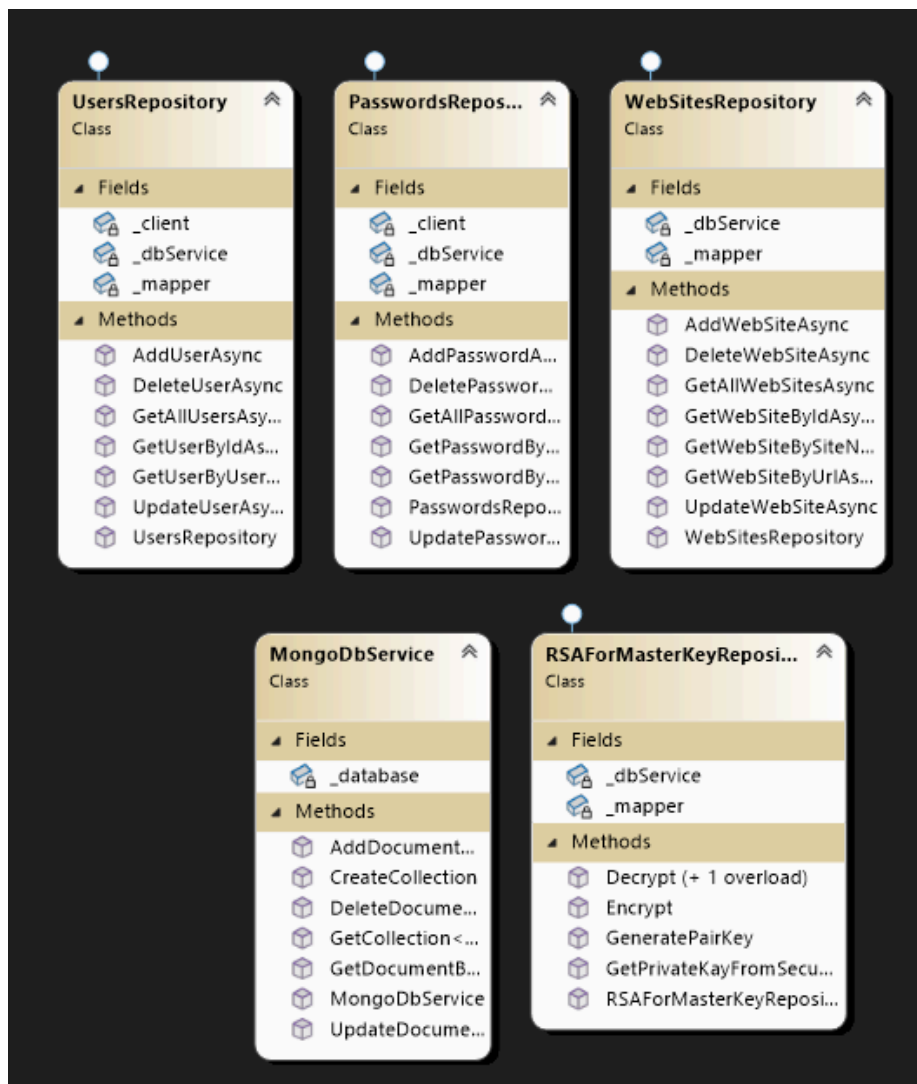


## תרשים UML

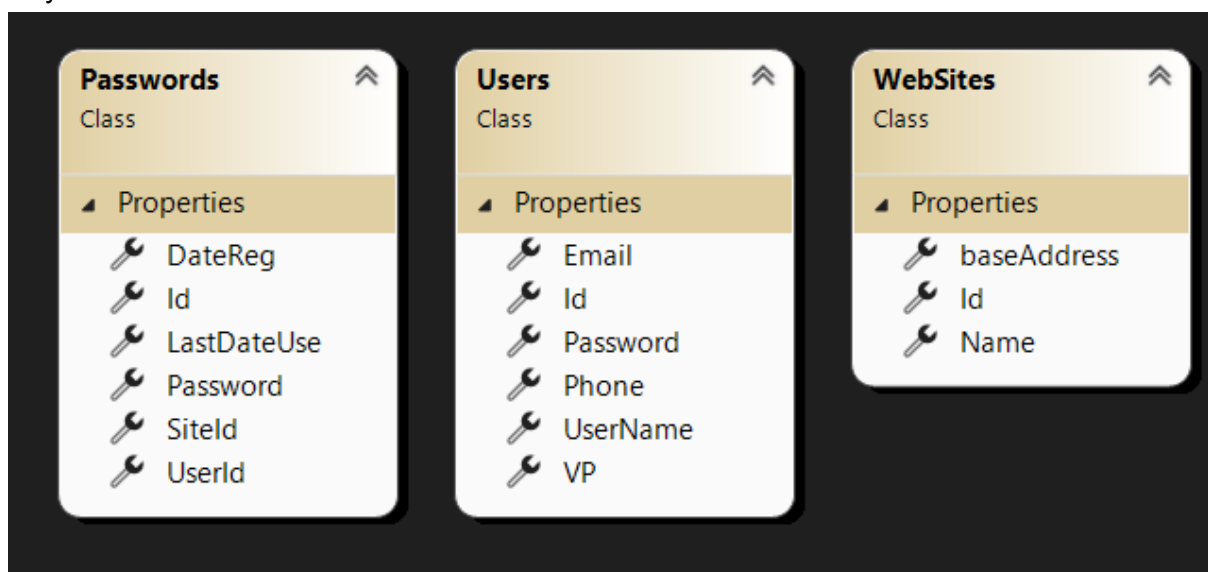
bl layer:



dal layer:

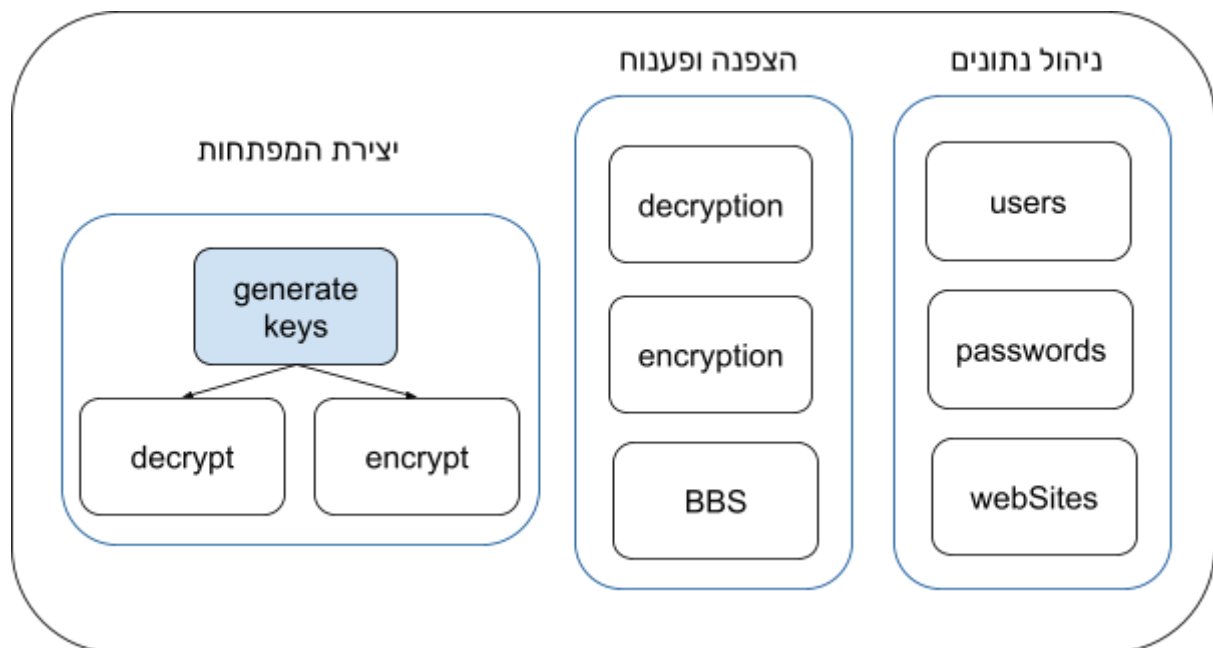


entties:





## design class diagram



## תיאור המחלקות המוצעות

**מחלקת users:** ניהול המשתמשים - הוספה, עדכון, מחיקה

**מחלקת passwords:** ניהול הסיסמאות - הוספה, עדכון, מחיקה

**מחלקת webSites:** ניהול אתרים - הוספה, עדכון, מחיקה.

**מחלקת generateKeyEncryption:** עוסק ביצירת מפתח ההצפנה

**מחלקת generateKeyDecryption:** עוסק ביצירת מפתח הפענוח

**מחלקת encryption:** מצפין את ההודעה המקורית בעזרת מפתח ההצפנה

**מחלקת decryption:** מפענח את ההודעה המוצפנת באמצעות מפתח הפענוח

**מחלקת RSA:** מצפינה מפענחת את מפתח המאסטר - מפתח הכניסה למערכת

**מחלקת BBS:** מחלקת יצירת מספרים פסאודו אקראיים לצורך תהליך ההצפנה והפענוח

## קלט ופלט של היחידות

| מחלקה                        | קלט   | פלט  |
|------------------------------|---|--|
| <b>users</b>                 | <b>התחברות:</b> אימייל וסיסמא<br><b>הרשמה :</b> שם משתמש , אימייל,<br>טלפון , סיסמא | משתמשים רשומים במערכת                              |
| <b>passwords</b>             | כתובת URL של אתר, סיסמא   | סיסמאות שמורות                                     |
| <b>webSites</b>              | כתובת אתר   | אתרים שנשמרו עבורם סיסמאות                         |
| <b>generateKeyEncryption</b> | מפתח ראשי, טקסט המקור   | מחזיר מפתח הצפנה                                   |
| <b>generateKeyDecryption</b> | וקטור המיקומים, מפתח ראשי   | מחזיר מפתח פענוח                                   |
| <b>encryption</b>            | מפתח הצפנה , הודעת המקור  | מחזיר את הסיסמא מוצפנת                             |
| <b>decryption</b>            | מפתח פיענוח , הודעה מוצפנת  | מחזיר את הסיסמא המקורית ע"י פענוח.                 |
| <b>RSA</b>                   | הודעה מקורית  | מצפין ומפענח את מפתח מאסטר (- כניסה למנהל סימסאות) |
| <b>BBS</b>                   | זרע ליצירת מפתח   | מחזיר וקטור/מטריצה עם ערכים פסאודו אקראיים.        |

## רכיבי ממשק

כפתורים

input

כרטיסים

טפסים (Forms)

## תיכון המערכת

### ארכיטקטורת המערכת

---

מערכת ההצפנה תאבטח את הכניסה למנהל סיסמאות באמצעות הצפנה אסימטרית RSA ותשמור את הסיסמאות בצורה מוצפנת באמצעות אלגוריתם הצפנה סימטרי מתקדם.

### תיכון מפורט

---

אלגוריתם המערכת כולל שימוש ב:  
תורת הגרפים  
מעגלים המילטוניים  
הצפנה סימטרית ואסימטרית  
מחולל מספרים פסאודו אקראיים

### חלופות לתיכון המערכת

---

חלופה אפשרית - Python:

שפת Python נחשבת לפופולרית ונוחה מאוד ללמידה, ויש בה ספריות הצפנה מתקדמות (כגון cryptography ו-keyring). עם זאת, היא פחות מתאימה לפיתוח יישומי שולחן עבודה (Desktop) עם ממשק משתמש עשיר, במיוחד כאשר נדרש אינטגרציה הדוקה עם מערכת ההפעלה Windows. בנוסף, יכולות ההצפנה המובנות ב-C# מספקות פתרון מהיר, מאובטח ונוח יותר לפרויקט מסוג זה.

## תיאור התוכנה

---

### סביבת עבודה

visual studio

react vscode

mongoDB atlas

### שפות תכנות:

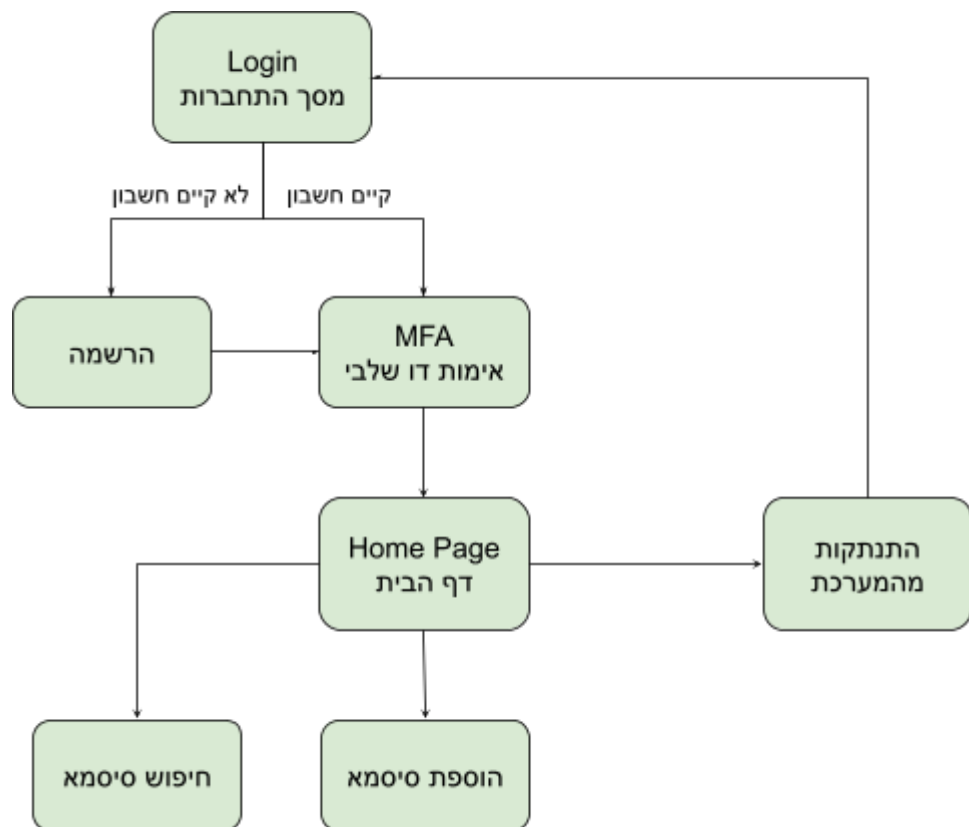
c# .net efcore

css , tsx

## תיאור מסכים

מסך התחברות  
מסך הרשמה למערכת  
דף הבית  
טופס הוספת סיסמא

## תרשים מסכים (diagram flow Screen)




## תיאור מסך הפתיחה-

מסך הפתיחה מאפשר הרשמה או התחברות.  
משם עוברים לדף אימות על מנת לאמת את המשתמש. לאחר מכן עוברים לדף הבית.

## מסכי האפליקציה:


**register** - דף הרשמה למערכת -  
הכנסת פרטי משתמש -  
שם, מייל, טלפון, סיסמה

**login** - דף התחברות -  
המשתמש מכניס פרטי התחברות -  
מייל וסיסמא



### SecureVault

Modern Password Management



## Create account

Sign up to start using our secure password manager

Full Name

Email Address

cell phone


Master Password

Confirm Password

Create Account


Already have an account? [Sign in](#)

© 2025 SecureVault. All rights reserved.



### SecureVault

Modern Password Management



## Welcome back

Sign in to access your secure vault

Email Address

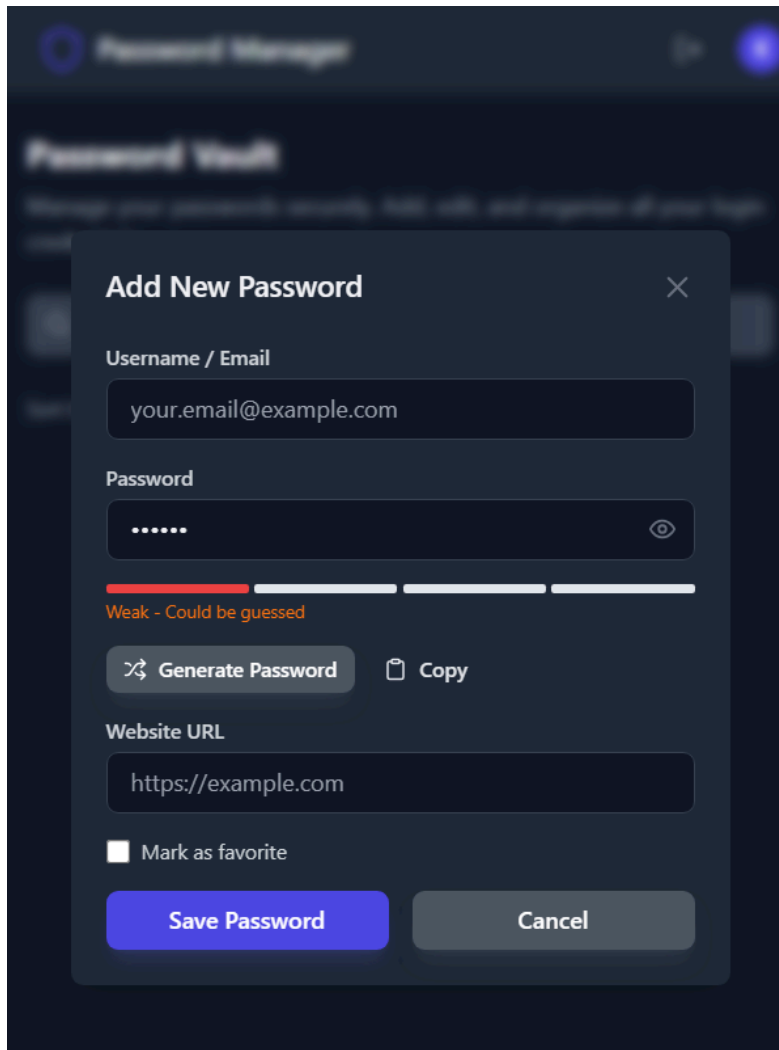
Master Password

Sign In

Don't have an account? [Create account](#)

© 2025 SecureVault. All rights reserved.

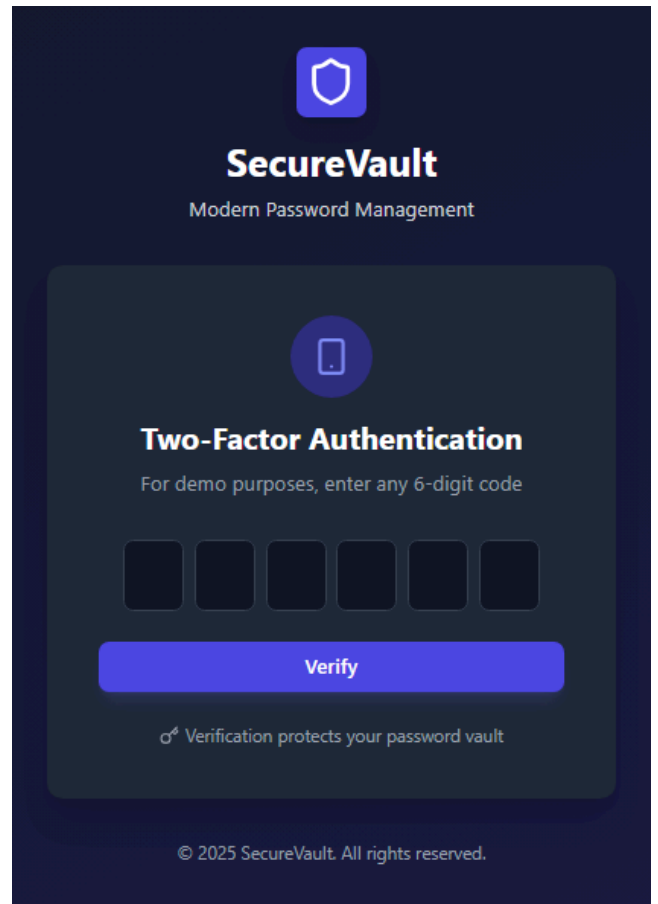
**הוספת סיסמה:**  
(חובה לבחור סיסמה חזקה)



The screenshot shows a dark-themed application window titled "Password Manager". A modal dialog titled "Add New Password" is open. It contains the following fields and controls:

- Username / Email:** A text input field containing "your.email@example.com".
- Password:** A password input field with masked characters "....." and an eye icon to toggle visibility. Below the field is a strength indicator bar that is mostly red, with the text "Weak - Could be guessed" in orange.
- Website URL:** A text input field containing "https://example.com".
- Buttons:** "Generate Password" (with a key icon), "Copy" (with a clipboard icon), "Save Password" (in blue), and "Cancel" (in grey).
- Checkbox:** "Mark as favorite" with an unchecked checkbox.

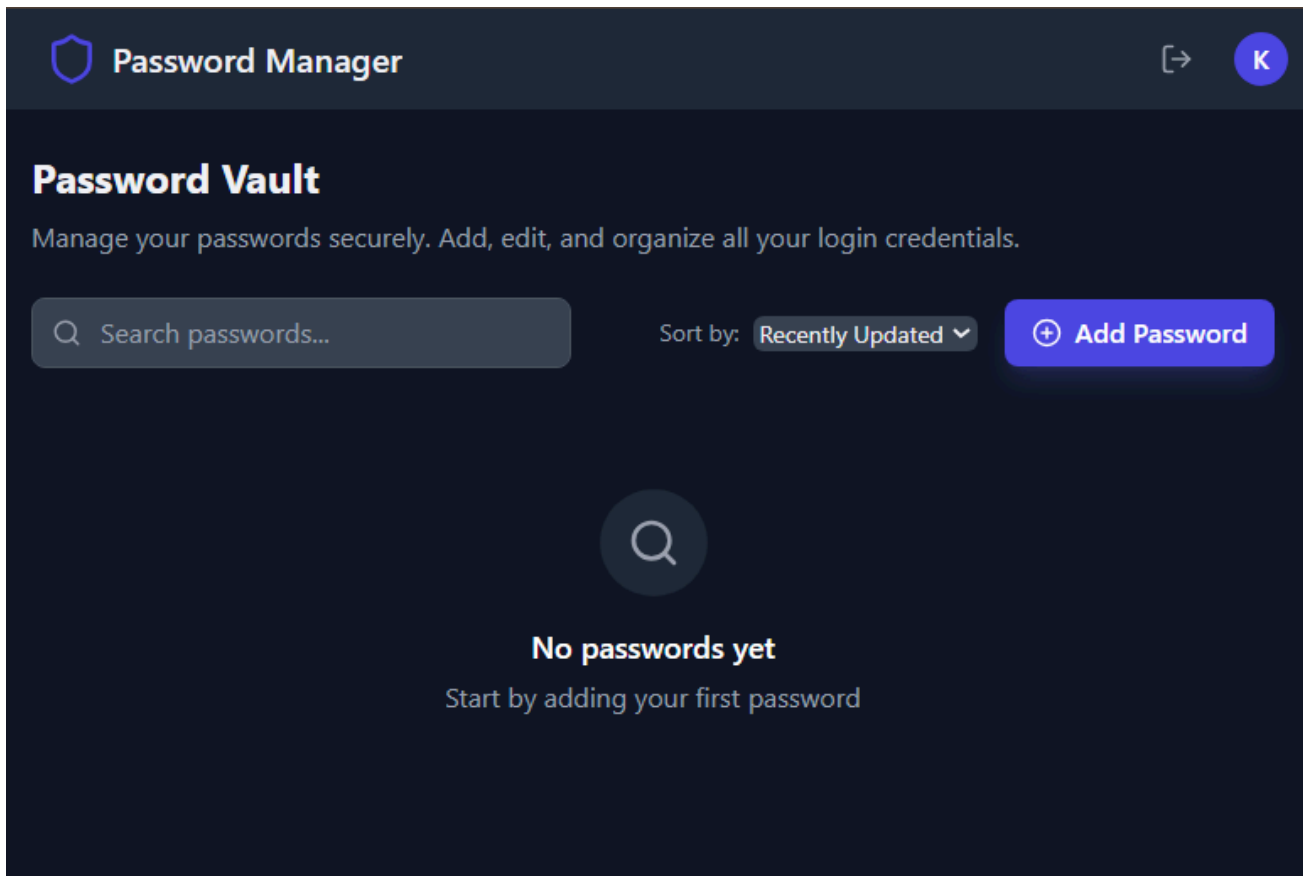
אימות דו שלבי - MFA  
לוודא שאכן מדובר במשתמש הנכון



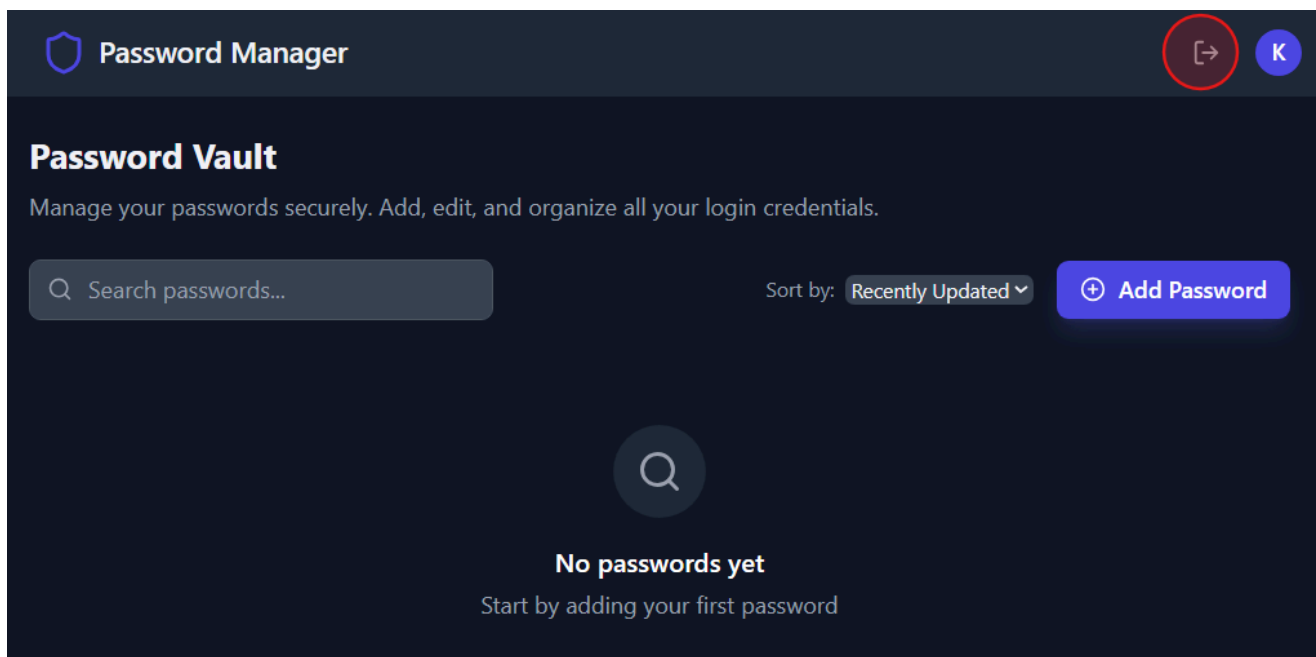
The screenshot shows a dark-themed application window titled "SecureVault" with the subtitle "Modern Password Management". A modal dialog titled "Two-Factor Authentication" is open. It contains the following elements:

- Icon:** A blue circle containing a white smartphone icon.
- Text:** "For demo purposes, enter any 6-digit code".
- Input:** Six empty square boxes for entering a 6-digit code.
- Button:** A large blue "Verify" button.
- Text:** A small icon of a key and the text "Verification protects your password vault".
- Footer:** "© 2025 SecureVault. All rights reserved."

דף הבית



התנתקות מהמערכת:



## ממשק משתמש

בממשק המשתמש השתמשתי בצבעים שמשדרים אמינות, לתת למשתמשים ביטחון שהסיסמאות שלהם שמורות במקום בטוח. גם הפונט, העיצוב, הכל בקונספט שמשדר אבטחה גבוהה.

## קוד התוכנית

בקטע קוד זה אני מצפינה את מפתח הכניסה למנהל סיסמאות באמצעות הצפנת RSA. הקוד מקבל את הסיסמא המקורית ומפתח ציבורי ומחזיר לי את הסיסמא מוצפנת.

```
2 references | swennly, 1 day ago | 1 author, 2 changes
public byte[] Encrypt(string plainText, string publicKey)
{
    if (string.IsNullOrEmpty(plainText))
        throw new ArgumentException("Plain text cannot be null or empty");

    try
    {
        using (RSA rsa = RSA.Create())
        {
            rsa.FromXmlString(publicKey);
            byte[] plainBytes = Encoding.UTF8.GetBytes(plainText);
            return rsa.Encrypt(plainBytes, RSAEncryptionPadding.Pkcs1);
        }
    }
    catch (Exception ex)
    {
        throw new Exception($"Encryption failed: {ex.Message}", ex);
    }
}
```

הצפנת סיסמאות שהוסיף המשתמש:

- הכנת מפתח ההצפנה:

הסיסמא שאותה מצפינים מחולקת לכמה בלוקים כך שכל בלוק מוצפן באמצעות מפתח הצפנה שונה. חלק ראשון של הפונקציה:

```
public (int[,] SubKeys, List<int> VectorOfPositions) GenerateSubKeysForEncryption(int[] block)
{
    int blocksCount = block.Count();

    // חלוקת ההודעה לבלוקים
    //List<int[]> blocks = ParseMessage(messageAsAscii, blocksCount);
    int[,] subKeys = new int[blocksCount, 1];
    List<int> vectorOfPositions = new List<int>();

    // בחירה אקראית של תו מהבלוק
    int randomIndex = RandomNumberGenerator.GetInt32(block.Length);
    int selectedChar = block[randomIndex];

    // ASCII שלילי, נעשה אותו חיובי (אם זה לא בא מתוך סיוח selectedChar אם 0)
    if (selectedChar < 0)
    {
        selectedChar = Math.Abs(selectedChar); // אם זה לא מספיק - לאתחל אותו ל-0
    }

    // הוספה לוקטור המיקומים
    vectorOfPositions.Add(selectedChar);
}
```



המשך-

```
// הוספה לוקטור המיקומים
vectorOfPositions.Add(selectedChar);

int index = selectedChar % _setting.keySize; // מודול 256

// אם האינדקס לא בסדר, תפסול את החישוב
if (index < 0 || index >= _keyEncryptionKey.Length)
{
    throw new ArgumentOutOfRangeException($"האינדקס {index} חורג מהמגבלה של המערך");
}

// קבלת ערך מהמפתח הראשי
int n = _keyEncryptionKey[index];
// יצירת וקטור זרע באורך 13
int[] seedVector = GenerateSeed(n);
// יצירת תת-מפתח
subKeys = BBSRandomGenerator.GenerateSubKey(seedVector, _setting);

return (subKeys, vectorOfPositions);
}
```

ביצירת המפתח הצפנה השתמשתי בפונקציות האלו:  
1. בפונקציה זו נייצר וקטור באורך 13

```
/// <summary>
/// מייצר וקטור זרע באורך 13 מתוך ערך התחלתי
/// </summary>
1 reference | swennly, 4 days ago | 1 author, 3 changes
private int[] GenerateSeed(int initialValue)
{
    return BBSRandomGenerator.GenerateBBSSequence(initialValue, _setting.subBlockSize, _setting);
}
```

2. פונקציה זו מייצרת את תתי מפתחות ההצפנה:

```
public static int[,] GenerateSubKey(int[] seedVector, MySetting _setting)
{
    int[,] subKey = new int[_setting.graphOrder, _setting.graphOrder];

    for (int i = 0; i < _setting.graphOrder; i++)
    {
        int[] rowValues = GenerateBBSSequence(seedVector[i], _setting.graphOrder, _setting);

        for (int j = 0; j < _setting.graphOrder; j++)
        {
            subKey[i, j] = rowValues[j];
        }
    }

    return subKey;
}
```

3. בפונקציה זו נקבל זרע, אורך הפלט המבוקש ונחזיר מערך עם ערכים פסאדו אקראיים

```
3 references | 0 changes | 0 authors, 0 changes
public static int[] GenerateBBSequence(int seed, int length, MySetting _setting)
{
    int[] sequence = new int[length];

    // הבטחה שהזרע הוא חיובי
    BigInteger x = new BigInteger(Math.Abs(seed));
    x = (x * x) % M; //  $x_0 = \text{seed}^2 \bmod m$ 

    for (int i = 0; i < length; i++)
    {
        x = (x * x) % M; //  $x_{n+1} = x_n^2 \bmod m$ 
        sequence[i] = (int)(x % 256); // מיפוי לטווח 0-255
    }

    return sequence;
}
```

המפתח מוכן.

### הצפנת הסיסמה:

בשלב הראשון אני מטפלת בסיסמה ומכינה אותה להצפנה:

- פונקציה זו מוסיפה תחילית לסיסמה שהיא תייצג את אורך הסיסמה וכן מוסיפה מלח לסיסמה :

```
1 reference | swennly, 22 hours ago | 1 author, 2 changes
public string AddSaltToMessageEnd(string message)
{
    message = AddLengthAsPrefix(message);
    Random _random = new Random();
    int targetLength = _setting.BlockSize;
    string saltChars = _setting.saltChars;
    if (message.Length > targetLength)
        throw new ArgumentException("ההודעה ארוכה מ-" + _setting.BlockSize + " תווים");
    int saltLength = targetLength - message.Length;
    StringBuilder saltBuilder = new StringBuilder(saltLength);
    for (int i = 0; i < saltLength; i++)
    {
        char randomChar = saltChars[_random.Next(saltChars.Length)];
        saltBuilder.Append(randomChar);
    }
    return message + saltBuilder.ToString();
}
```

לאחר מכן מחלק את הסיסמה לכמה בלוקים:

```

1 reference | swenny, 22 hours ago | 1 author, 4 changes
private List<int[]> ParseBlock(int[] block)
{
    List<int[]> subBlocks = new List<int[]>();
    int validLength = Math.Min(block.Length, _setting.BlockSize);

    for (int i = 0; i < _setting.BlockSize/_setting.subBlockSize; i++)
    {
        int[] subBlock = new int[_setting.subBlockSize];
        int startIndex = i * _setting.subBlockSize;

        // למקרה של בלוק אחרון שאינו מלא
        if (startIndex < validLength)
        {
            int copyLength = Math.Min(_setting.subBlockSize, validLength - startIndex);
            Array.Copy(block, startIndex, subBlock, 0, copyLength);

            // מילוי עם אפסים אם צריך
            if (copyLength < _setting.subBlockSize)
            {
                for (int j = copyLength; j < _setting.subBlockSize; j++)
                {
                    subBlock[j] = 0;
                }
            }
        }
        else
        {
            // מילוי תת-בלוק ריק באפסים
            for (int j = 0; j < _setting.subBlockSize; j++)
            {
                subBlock[j] = 0;
            }
        }

        subBlocks.Add(subBlock);
    }

    return subBlocks;
}

```

המרת בלוק למטריצת סמיכויות:

```

private int[,] BlockToAdjacencyMatrix(List<int[]> subBlocks)
{
    // יצירת מטריצת סמיכות התחלתית מלאה באפסים
    int[,] adjacencyMatrix = new int[_setting.graphOrder, _setting.graphOrder];

    // מעבר על כל תת-בלוק ויצירת מעגל המילטוני
    for (int subBlockIndex = 0; subBlockIndex < subBlocks.Count; subBlockIndex++)
    {
        int[] subBlock = subBlocks[subBlockIndex];

        // יצירת מסלול המילטוני עבור תת-הבלוק
        List<int> path = CreateHamiltonianCircuit(subBlockIndex);

        // הוספת המשקלים למטריצת הסמיכות
        for (int i = 0; i < path.Count - 1; i++)
        {
            int value = subBlock[i];
            int fromVertex = path[i];
            int toVertex = path[i + 1];

            adjacencyMatrix[fromVertex, toVertex] = value;
        }

        // סגירת המעגל - חיבור בין הקדקוד האחרון לראשון
        int lastValue = subBlock[path.Count - 1];
        int lastVertex = path[path.Count - 1];
        int firstVertex = path[0];

        adjacencyMatrix[lastVertex, firstVertex] = lastValue;
    }

    return adjacencyMatrix;
}

```

ביצוע פעולת XOR

1. עם המטריצה הקודמת 2. עם תת המפתח הצפנה

```
private int[,] MatrixXor(int[,] matrix1, int[,] matrix2)
{
    int rows = matrix1.GetLength(0);
    int cols = matrix1.GetLength(1);
    int[,] result = new int[rows, cols];

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            result[i, j] = matrix1[i, j] ^ matrix2[i, j];
        }
    }

    return result;
}
```

לאחר שהבלוק הוצפן נשרשר אותו לרשימת הבלוקים המוצפנים:

```
2 references | 0 changes | 0 authors, 0 changes
public static int[] ConcatenateBlocks(List<int[]> blocks)
{
    int totalLength = blocks.Sum(block => block.Length);
    int[] result = new int[totalLength];

    int index = 0;
    foreach (int[] block in blocks)
    {
        Array.Copy(block, 0, result, index, block.Length);
        index += block.Length;
    }

    return result;
}
```

**פענוח** הסיסמא מתבצע בסדר הפוך:  
יצירת מפתח פענוח בהתבסס על וקטור המיקומים והמפתח הראשי (KEK):

```
public int[,] GenerateSubKeysForDecryption(List<int> vectorOfPositions)
{
    int blocksCount = vectorOfPositions.Count;

    // יצירת מפתח מלא מתוך וקטור המיקומים והמפתח הראשי
    int[] key = GenerateKey(_keyEncryptionKey, vectorOfPositions);

    // חלוקת המפתח לוקטורים
    List<int[]> seedVectors = ParseKey(key, blocksCount);

    // יצירת תת-מפתחות מהוקטורים
    int[,] subKeys = new int[blocksCount][,];
    // again not blockCount. k'.
    for (int i = 0; i < blocksCount; i++)
    {
        subKeys[i] = BBSRandomGenerator.GenerateSubKey(seedVectors[i], _setting);
    }
    return subKeys;
}
```

נחלק את הבלוקים

```
// חלוקת ההודעה המוצפנת לבלוקים
List<int[]> encryptedBlocks = ParseEncryptedMessage(encryptedMessage, blocksCount);

// מערך לאחסון הבלוקים המפוענחים
List<int[]> decryptedBlocks = new List<int[]>();

// (מסריצת אתחול לבלוק הראשון) CBC מסריצה קודמת עבור
int[,] previousMatrix = _initializationMatrix;
```

המשך פונקציית הפענוח:  
נמיר את הבלוק המוצפן שקיבלנו למסריצה "i הפונקציה הזו:

```
private int[] AdjacencyMatrixToBlock(int[,] adjacencyMatrix)
{
    // שחזור כל 6 תת-הבלוקים מתוך המעגלים ההמילטוניים
    List<int[]> subBlocks = new List<int[]>();

    for (int subBlockIndex = 0; subBlockIndex < _setting.BlockSize / _setting.subBlockSize; subBlockIndex++)
    {
        int[] subBlock = new int[_setting.subBlockSize];
        List<int> path = CryptographyUtils.CreateHamiltonianCircuit(subBlockIndex);

        // הוצאת הערכים מהמטריצה לפי המסלול
        for (int i = 0; i < path.Count - 1; i++)
        {
            int fromVertex = path[i];
            int toVertex = path[i + 1];
            subBlock[i] = adjacencyMatrix[fromVertex, toVertex];
        }

        // הערך האחרון - מהקודם האחרון לראשון
        int lastVertex = path[path.Count - 1];
        int firstVertex = path[0];
        subBlock[path.Count - 1] = adjacencyMatrix[lastVertex, firstVertex];

        subBlocks.Add(subBlock);
    }

    // איחוד כל תת-הבלוקים לבלוק אחד
    int[] block = new int[_setting.BlockSize];
    int index = 0;

    foreach (int[] subBlock in subBlocks)
    {
        Array.Copy(subBlock, 0, block, index, subBlock.Length);
        index += subBlock.Length;
    }

    return block;
}
```

נפעיל פעולת XOR עם המפתח פענוח שייצרנו, נפעיל פעולת XOR עם הבלוק הקודם(עבור הבלוק הראשון יצרנו מטריצת אתחול)

```
// פענוח כל בלוק
for (int i = 0; i < blocksCount; i++)
{
    // המרת וקטור לבלוק המוצפן למטריצה
    int[,] encryptedMatrix = VectorToMatrix(encryptedBlocks[i]);

    int[,] nextPreviousMatrix = encryptedMatrix;
    // עם תת-המפתח XOR ביצוע
    int[,] modifiedMatrix = CryptographyUtils.MatrixXor(encryptedMatrix, subKeys[i]);

    // עם המטריצה הקודמת XOR ביצוע (CBC)
    int[,] adjacencyMatrix = CryptographyUtils.MatrixXor(modifiedMatrix, previousMatrix);

    // עדכון המטריצה הקודמת לבלוק הבא
    previousMatrix = nextPreviousMatrix;

    // המרת מטריצת הסמיכות לבלוק
    int[] decryptedBlock = AdjacencyMatrixToBlock(adjacencyMatrix);

    decryptedBlocks.Add(decryptedBlock);
}
```

המשך-

נשתמש בפונקציה לשרשור הבלוקים המוצפנים, נוריד את התחילית של הסיסמא ואת המלח, נמיר לאסקי ונחזיר את הסיסמא המפוענחת.

```
// איחוד כל הבלוקים המפוענחים להודעה אחת
int[] decryptedMessage = CryptographyUtils.ConcatenateBlocks(decryptedBlocks);
// שליפת הסיסמה בלי המלח והתחילית
int[] parse = GetTextByPrefix(decryptedMessage);
// לטקסט והסרת אפסים ASCII המרת מערך
string convertAscii = ConvertAsciiToString(parse);

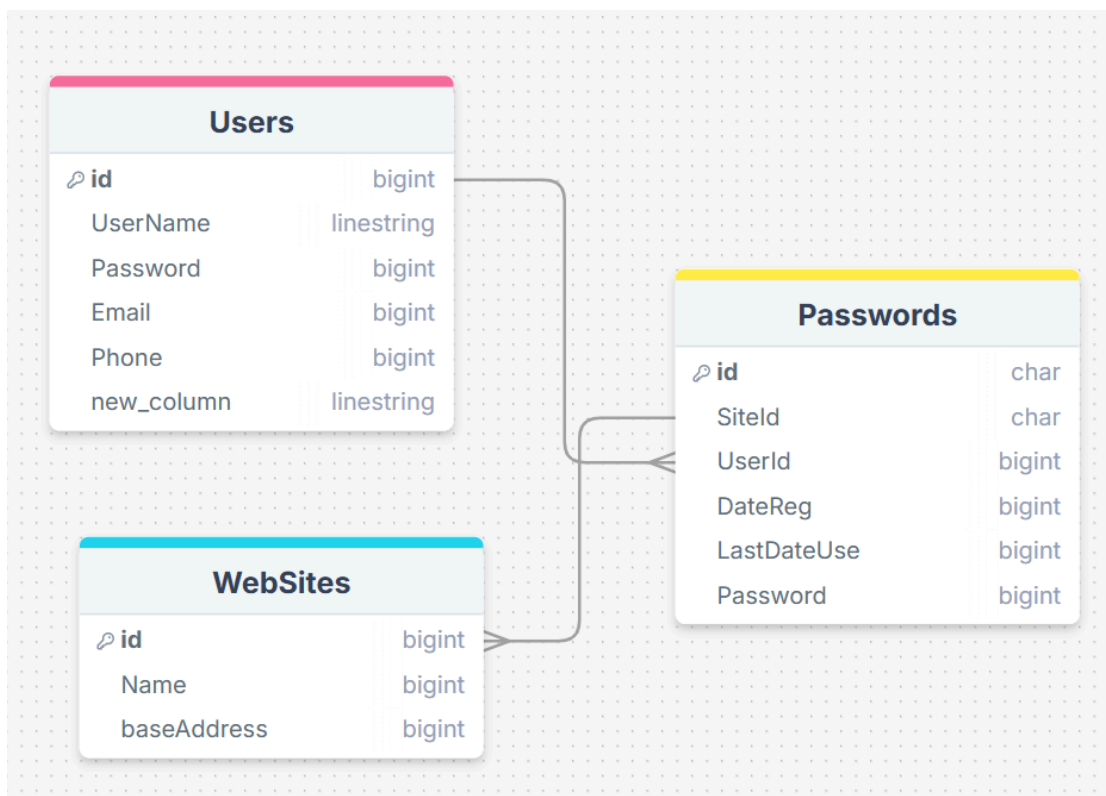
return convertAscii;
}
```

## תיאור מסד הנתונים

השתמשתי במסד נתונים mongoDB לניהול הנתונים. מונגו DB מיועדת לאחסון נתונים בצורה גמישה ולא רלציונית. היא משתמשת במבנה מסמכים (documents) כדי לאחסון נתונים, בניגוד למסדי נתונים רלציוניים המשתמשים בטבלאות.

## תרשימי ERD

משתמשים - אוסף של כל המשתמשים הקיימים במערכת  
 סיסמאות - אוסף המכיל את כל הסיסמאות של כל המשתמשים  
 אתרים - אוסף המכיל את כל האתרים של הסיסמאות.



## מדריך למשתמש

**דף כניסה** - התחברות למערכת יש להקיש שם משתמש וסיסמא  
**אימות דו שלבי** - לאחר התחברות הרשמה תקבלו קוד אישור במייל לאימות.  
**הרשמה** - יש להכניס פרטים להרשם למערכת יש לבחור סיסמה חזקה.  
**דף הבית** - חיפוש סיסמאות, הוספת סיסמא, יציאה מהמערכת.  
**הוספת סיסמא** - יש ללחוץ על כפתור הוספת סיסמא ולמלא את הפרטים הנדרשים

## בדיקות והערכה

**בדיקת רגישות לטקסט קריא (Plaintext Sensitivity Test):**  
 בדיקה זו נועדה לוודא את תכונת הדיפוזיה. היא מודדת את אחוז הסיביות שמשתנות בטקסט המוצפן כאשר משנים סיבית אחת או יותר בטקסט הקריא המקורי. במערכת המוצעת, אחוז ההבדלים בין הסיביות נע בין 48.16% ל-51%, לעומת 47.1%-50.80% בצופן AES-128, מה שמצביע על דיפוזיה טובה.

**בדיקת רגישות למפתח (Key Sensitivity Test):**  
 בדיקה זו משמשת לבחינת תכונת הקונפוזיה. היא מודדת את השינוי בטקסט המוצפן כאשר משנים סיבית אחת או יותר במפתח הסודי, בעוד הטקסט הקריא נותר קבוע. התוצאות עבור המערכת המוצעת הראו אחוז שינוי ממוצע של 49.64%-50.79%, בהשוואה ל-48.25%-50.73% בצופן AES-128. ממצאים אלו מעידים כי תהליך יצירת המפתחות באלגוריתם המוצע חזק יותר מזה של AES-128.

בוצעה בדיקת **DIEHARD** - אחת מהבדיקות הידועות ביותר בתחום - כדי לבחון את איכות האקראיות של המערכת. מטרת הבדיקה היא לוודא שהפלט של מערכת ההצפנה אינו ניתן להבחנה סטטיסטית מפלט אקראי, דבר החשוב לעמידות בפני התקפות סטטיסטיות. הבדיקה מתמקדת בערכי p-value בטווח [0.025, 0.975]. זרם הסיביות שהופק מהמערכת עבר בהצלחה את כל הבדיקות, מה שמעיד על רמת אקראיות מספקת

**ניתוח התקפת כוח גס (Brute Force Attack):**



הניתוח מעריך את הקושי לנסות את כל המפתחות האפשריים. בהנחה של שימוש במכונה חזקה, מספר המפתחות האפשרי ( $256^{1000}$ ) הופך כל ניסיון חיפוש ממצה לבלתי אפשרי בפרק זמן סביר.

בנוסף, הקושי במציאת מספרים ראשוניים למחולל BBS (שמשמש ליצירת המפתח) תורם אף הוא לחוזק ההצפנה.

בוצעה השוואת ביצועים בין המערכת המוצעת לבין צופני בלוקים מוכרים אחרים, כמו AES ו-Triple DES, עבור גדלי הודעות שונים. המערכת המוצעת הציגה תוצאות טובות יותר מבחינת זמן ריצה, בהשוואה לאלגוריתמים הסטנדרטיים.

## אבטחת מידע

---

הכניסה לאתר מתבצעת עם אימות דו שלבי, את מפתח הכניסה אשמור מוצפן ב-RSA לאחר שהשוונו עם מפתח המקור ישלח מייל עם קוד אימות לאמת את המשתמש. את הסיסמאות אשמור מוצפנות ע"י אלגוריתם ההצפנה.

## מסקנות

---

העבודה מציגה מערכת הצפנה חדשה שנעזרת בעקרונות תורת הגרפים, אשר מאפשרת רמת אבטחה גבוהה תוך שמירה על ביצועים של עיבוד נתונים. הצפנת הבלוק המוצעת משתמשת במיוחד במעגלים המילטוניים המופרדים אשר אומצו לייצוג של הטקסט הפשוט בשלב הקדם-הצפנה. התהליך עושה שימוש במפיק מפתח משנה ספציפי שהוקם כדי לייצר את מפתחות ההצפנה בהתאם לדרישות של המערכת המוצעת.

## פיתוחים עתידיים

---

- פיתוח תוסף לדפדפן שמזהה אם המשתמש זקוק לסיסמא ונותן לו אותה מיידית.
- שימוש במחולל פסאודו אקראי מתקדם יותר
- הוספת ששן לאתר. לאחר כמה דקות של חוסר פעילות יצטרכו להתחבר שוב למערכת.

## ביבליוגרפיה

---

<https://www.fortinet.com/resources/cyberglossary/encryption>

[https://www.hamichlol.org.il/%D7%9E%D7%97%D7%95%D7%9C%D7%9C\\_%D7%9E%D7%A1%D7%A4%D7%A8%D7%99%D7%9D\\_%D7%A4%D7%A1%D7%90%D7%95%D7%93%D7%95-%D7%90%D7%A7%D7%A8%D7%90%D7%99%D7%99%D7%9D\\_%D7%A7%D7%A8%D7%99%D7%A4%D7%98%D7%95%D7%92%D7%A8%D7%A4%D7%99](https://www.hamichlol.org.il/%D7%9E%D7%97%D7%95%D7%9C%D7%9C_%D7%9E%D7%A1%D7%A4%D7%A8%D7%99%D7%9D_%D7%A4%D7%A1%D7%90%D7%95%D7%93%D7%95-%D7%90%D7%A7%D7%A8%D7%90%D7%99%D7%99%D7%9D_%D7%A7%D7%A8%D7%99%D7%A4%D7%98%D7%95%D7%92%D7%A8%D7%A4%D7%99)

[https://www.academia.edu/15298020/Encryption\\_Algorithm\\_Using\\_Graph\\_Theory?email\\_work\\_card=view-paper](https://www.academia.edu/15298020/Encryption_Algorithm_Using_Graph_Theory?email_work_card=view-paper)

<https://www.geeksforgeeks.org/computer-networks/rsa-algorithm-cryptography/>