# תיאור הפרויקט

## תיאור הבעיה

כיום משתמשים בכונני usbבשביל להעביר מידע ממקום למקום אחר, הם נותנים לנו גישה מהירה מכל מחשב למידע שאנחנו צריכים איתנו, אך אם נאבד אותו המידע שלנו עלול להיחשף לעיניים לא רצויות, ולכן אני רוצה לפתור את הבעיה הזאת, על ידי יצירת תוכנה אשר מצפינה את המידע הנמצא על הכונן.

## האלגוריתם

בשביל לפתור את הבעיה שכתבתי למעלה אני אצטרך, להצפין את המידע. כיום ישנם סוגים רבים של הצפנות , ומשום אינני מעביר את המידע באינטרנט אוכל להשתמש בסוג הצפנה סימטרית, זוהי הצפנה עם אותו מפתח הצפנה ואותו מפתח פיענוח, איננו צריכים שני מפתחות נפרדים משום שאנחנו לא מעבירים אותם בשום מקום.

לכן בשביל פתרון הבעיה שלי בחרתי באלגוריתם AES , אלגוריתם זה הוא אלגוריתם סטנדרטי בתחום ההצפנות הסימטריות, הוא אלגוריתם חזק שאיננו ניתן לפריצה אלה אם כן ידוע הסיסמא איתה יצרו את המפתח.

## מבני נתונים

בתוכנית שלי אני לא אצטרך לשמור מידע רב או בכלל לשמור מידע משום שהמידע ישירות נכתב על הכונן אותו מצפינים או מפענחים.

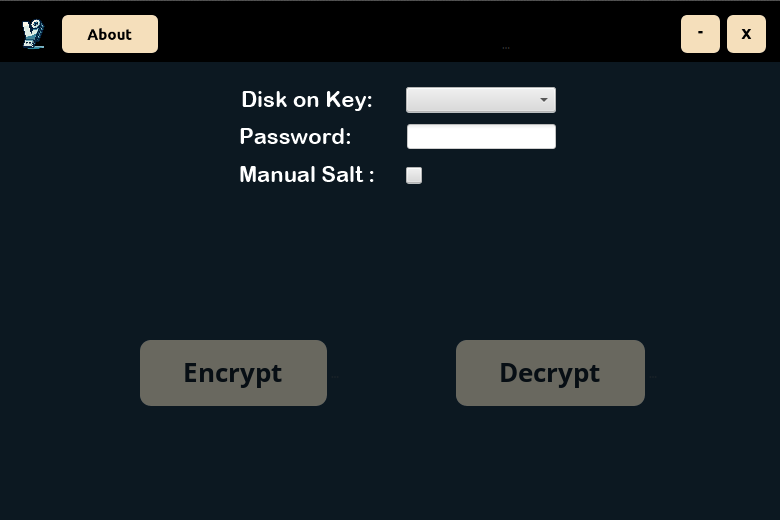
אך אני אשתמש בבסיסי נתונים זמניים לצורך ההצפנה והפענוח.

אלגוריתם הצפנה AES, מצפין חלקי קבצים כאשר הוא מצפין כל פעם חלק של 16 בטים, בשביל זה אני אשתמש במטריצה (מערך דו מימדי) של תווים בו אני אשמור על חלק קטן מן המידע של הקובץ, בנוסף כל קובץ יוגדר במחלקה מיוחדת משלו עם תור של כל הבלוק ים כך שהתוכנה תצפין את חלקי המידע אחד אחרי השני, הסיבה לשימוש מערך היא הנוחות והמהירות הגישה לכל המידע בחלק המסוים בנוסף הסיבה לשימוש בתור כשמרת כל הבלוקים היא משום שתור מאפשר לנו לעקוב אחרי כמות הבלוקים שצריך להצפין, מסדר לנו אוטומטית את סדר הכתיב/קריאה של הבלוקים וכמו כן נותן לנו בקלות לדעת מתי סיימנו לקרוא את כל הבלוקים משום שבמצב זה התור יהיה ריק.

כמו כן על מנת לקרוא מידע מקובץ ולכתוב מידע לקובץ אני אשתמש בבסיס נתונים FileInputStream וFileInputStream, הסיבה לבחירה זו היא משום שמחלקות אלו כבר בנויות אוטומטית לקרוא קבצים כקבצים בינריים כאשר בסופו של דבר כל קובץ הוא קובץ בינארי, כמו כן מחלקה זו מאפשר לי לבנות אובייקט קריאה וכתיבה אחד שמאפשר בקלות לבחור בין קריאת קבצים בינאריים לבין קריאת קבצי zip.

# מסכי הפרויקט

## מסך ראשי



המסך הראשי נפתח ישירות עם פתיחת הפרויקט, מאפשר לבחור את הכונן להצפנה/פיענוח, המסך מאפשר לכתוב סיסמא (ללא סיסמא כפתורי הפענוח/ההצפנה אינם פעילים), כמו כן המסך מאפשר לכתוב salt אם המשתמש אינו רוצה להשתמש בsalt שנוצר אוטומטית.

## מסך טעינה

A white line on a black background

Description automatically generated

מסך הטעינה מציג את התהליך וכמה נשאר עד סוף הפענוח/הצפנה של כל הקבצים.

## מסך הודות

A screenshot of a computer

Description automatically generated

מסך הודות מציג את שם המתכנת, ומידע נוסף כגון מידע על מנחה הפרויקט.

# תיאור מחלקות

## **מחלקת MainScreenController**

**מטרה:**

* ניהול ממשק המשתמש ואינטראקציות המשתמש עבור אפליקציית הצפנה/פענוח קבצים.
* יישום ממשק AESObserver, המרמז על תבנית עיצוב אסינכרונית שבה הבקר מתעדכן לגבי התקדמות ההצפנה/פענוח.

**משתנים:**

* **ap:** AnchorPane המשמש כנראה כמרכיב פריסת השורש של ממשק המשתמש.
* **disksList:** ChoiceBox המאפשר למשתמש לבחור כונן להצפנה או פענוח.
* **passInput:** TextField עבור המשתמש להזנת הסיסמה המשמשת בהצפנה/פענוח.
* **saltInput:** TextField עבור המשתמש להזנת ערך סולט אופציונלי (משמש לחיזוק ההצפנה).
* **encrypt:** Button שמתניע את תהליך ההצפנה.
* **decrypt:** Button שמתניע את תהליך הפענוח.
* **salt:** אלמנט Text כנראה להצגת מידע על ערך הסולט.
* **model** מופע ModelControl. זה כנראה מייצג את הלוגיקה הליבה של תהליכי ההצפנה/פענוח.

**שיטות:**

* **()Initialize** 
  + ממלא את disksList בכונני USB מזוהים.
  + מצרף מאזינים לפיקוח על קלט סיסמה והפעלה/השבתת לחצני הצפנה/פענוח בהתאם.
  + מחבר את הבקר ל- model על ידי רישום 'this' כ- AESObserver.
* **checkPassword (String newValue)**
  + בודק את חוזק הסיסמה שהוזנה (חייבת להיות לפחות 6 תווים).
  + מפעיל/משבית את לחצני ההצפנה והפענוח בהתבסס על חוזק הסיסמה.
* **onMousePressed(MouseEvent event)**
  + רושם קואורדינטות עכבר התחלתיות עבור פעולות גרירה.
* **onMouseDragged(MouseEvent event)**
  + מאפשר גרירת חלון היישום.
* **Encrypt()**
  + מציג "LoadingPopup" המציין שהתהליך
  + מתחיל אשכול חדש להפעלת פעולת ההצפנה דרך model.
* **Decrypt()**
  + מציג "LoadingPopup" המציין שהתהליך
  + מתחיל אשכול חדש להפעלת פעולת הפענוח דרך model.
* **LoadingPopup(String thing)**
  + שיטת עזר להצגת חלון קופץ של טעינה עם הודעה ניתנת להתאמה אישית.
* **update()**
  + מיישם את ממשק AESObserver. כנראה משמש לעדכון סרגל התקדמות או אלמנט UI אחר מהמודל במהלך הצפנה/פענוח.
  + מסתיר את חלון הקופץ של הטעינה.
* **updateProgress(double n)**
  + מיישם את ממשק AESObserver. מעדכן את תצוגת ההתקדמות בחלון הקופץ של הטעינה.
* **Exit ()**
  + סוגר את חלון היישום.
* **mouseEnter (MouseEvent event)**
  + מוסיף אפקט חזותי (בהירות מופחתת) כאשר העכבר מרחף מעל אלמנט UI.
* **mouseLeave (MouseEvent event)**
  + משחזר את האפקט החזותי המקורי כאשר העכבר עוזב אלמנט UI.
* **handle(MouseEvent me)**
  + מזער את חלון היישום.
* **changeSalt()**
  + מחליף את הנראות של שדה קלט הסולט ואלמנט הטקסט הקשור.

## **מחלקת FileHolder**

### **מטרה:**

* מייצגת קובץ ומטפלת בתהליכי ההצפנה והפענוח שלו בתוך היישום.
* מספקת פונקציונליות לקריאת קובץ (כולל ארכיבי ZIP) לחסימות ניתנות לניהול, עיבוד חסימות אלה באמצעות אלגוריתם AES וכתיבתן חזרה לקובץ.

### **משתנים:**

* **filepath:** מחרוזת המחזיקה את הנתיב לקובץ.
* **blocks:** תור של אובייקטי Block. כל Block מייצג מקטע מנתוני הקובץ.
* **algo:** מופע של מחלקת AES האחראית על הצפנה/פענוח.
* **mode:** בוליאני המציין מצב פעולה: True להצפנה, False לפענוח.

### **בנאי:**

* **FileHolder(String filepath, AES algo, Boolean mode)**
  + מאתחל אובייקט FileHolder חדש.
  + מגדיר את filepath, קורא את הקובץ ל-blocks ומגדיר את algo ו-mode.

### **שיטות:**

* **readFileIntoBlocks(String filepath)**
  + קורא קובץ לתור של אובייקטים Block.
  + מטפל בארכיבי ZIP: חוזר על ערכי zip, קורא ומתמיר נתונים לחסימות.
  + מטפל בקבצים רגילים: קורא מקטעי נתונים לחסימות.
  + **מחזיר:** תור של אובייקטים Block
* **bytesToHexString(byte[] bytes, int length)**
  + שיטת עזר להמרת מערך בתים לייצוג מחרוזת הקסדצימלית.
* **getFilePath()**
  + גישה ל-filepath.
* **getBlocks()**
  + גישה לתור blocks.
* **writeBlocksToFile(String outputFilePath)**
  + כותב את החסימות המוצפנות/המפוענחות לקובץ פלט.
  + מבחין בין קבצי ZIP וקבצים בינאריים.
  + מטפל בפלט ZIP:
    - מוציא ערכים ZIP לתיקיה זמנית.
    - מצפין/פענח כל קובץ שהופק בנפרד (ThreadBlock מוסבר בהמשך).
    - יוצר ארכיון ZIP חדש ב-outputFilePath המכיל את הקבצים שעובדו.
  + מטפל בפלט קובץ רגיל:
    - מצפין/פענח חסימות (ThreadBlock).
    - כותב את נתוני הבית המתקבלים ישירות ל-outputFilePath.
* **writeBlocksToZipFileWithEncryptedFiles(...)**
  + שיטת עזר לטיפול בקבצי ZIP (ראה הסבר ב-writeBlocksToFile).
* **addFileToZip(...)**
  + שיטת עזר להוספת קבצים לארכיון ZIP.
* **getBlocksAsInputStream**
  + מרכיב את הנתונים מחסימות ל-ByteArrayInputStream לצורך כתיבה.
* **writeBlocksToNonZipFile(...)**
  + שיטת עזר לכתיבה לקבצים רגילים (לא ZIP).
* **hexStringToBytes(String hexString)**
  + שיטת עזר להמרת מחרוזת הקסדצימלית חזרה למערך בתים.

## **מחלקת ThreadBlock**

### **מטרה:**

* מגדירה יחידת עבודה עבור הצפנה או פענוח מרובי-שיטות של אובייקטי Block בודדים בתוך אלגוריתם AES.
* מיישמת את ממשק Runnable, ומאפשרת לה להיות מופעלת כשיטה נפרדת.

### **משתנים:**

* **bl:** אובייקט Block המייצג את הנתונים שיש להצפין או לפענח.
* **algorithm:** אובייקט AES, כנראה מתייחס למופע של מחלקה המממשת את לוגיקת ההצפנה/פענוח של AES.
* **mode:** דגל בוליאני המציין את מצב הפעולה. True מציין הצפנה, False מציין פענוח.

### **בנאי:**

* **ThreadBlock(Block bl, AES algorithm, Boolean mode)**
  + מאתחל את האובייקט עם הנתונים הדרושים:
    - ה-Block שצריך לעבד.
    - אובייקט ה-AES האחראי על ביצוע פעולות קריפטוגרפיות.
    - מצב הפעולה mode.

### **שיטות:**

* **run()**
  + דורסת את שיטת run המוגדרת בממשק Runnable.
  + מכילה את לוגיקת הליבה שתבוצע כאשר השיטה מתחילה:
    - **בודק את דגל mode:**
      * אם mode הוא True, הוא קורא לשיטה algorithm.Encrypt(bl) כדי להצפין את הבלוק.
      * אם mode הוא False, הוא קורא לשיטה algorithm.Decrypt(bl) לצורך פענוח.

## מחלקת Key

### **מטרה:**

* מייצגת מפתח הצפנה המשמש בתהליך ההצפנה/פענוח של AES.
* מאחסנת את המפתח הראשוני ומספקת שיטה לחישוב המפתח המורחב, שהוא רצף של מפתחות המשמשים בסיבובים שונים של AES.

### **משתנים:**

* **key:** מערך דו-מימדי של תווים (char[][]) המאחסן את המפתח הראשוני בגודל 32 סיביות.
* **extendedKey:** מערך דו-מימדי של תווים (char[][]) המאחסן את המפתח המורחב (סדרה של 16 'מילים', כאשר כל מילה היא 16 סיביות).

### **בנאי:**

* **Key(char[] key)**
  + מקבל מערך תווים (key) המייצג את המפתח הראשוני בגודל 32 סיביות כקלט.
  + מאתחל את משתנה החבר key על ידי המרת התווים ההקסדצימליים במערך הקלט לערכי התווים המתאימים.

### **שיטות:**

* **extendKey()**
  + יוצר את המפתח המורחב על סמך המפתח הראשוני. זה כולל סדרה של שלבים:
    1. **סיבוב:** מזיז את הבתים בעמודה של המפתח המורחב.
    2. **החלפה:** מחליף בתים עם ערכים תואמים מטבלת חיפוש (SBox).
    3. **XOR עם Rcon:** מבצע XOR של בית עם קבוע סיבוב (RCon) המשמש בהרחבת מפתח.
    4. **XOR עם מילה קודמת:** מבצע XOR של מילה מהמפתח המורחב עם מילים מסיבובים קודמים.

## מחלקת Salt

### **מטרה:**

* כלי שירות פשוט ליצירת ערך Salt על סמך תכונות קובץ נתון.

### **שיטות:**

* **generate(String filePath)**
  + **שיטה סטטית:** משמעותה שניתן לקרוא לה בלי ליצור מופע של המחלקה (לדוגמה, SaltGenerator.generate(...)).
  + **מחשב Salt:**
    1. יוצר אובייקט File מה-filePath שסופק.
    2. משלב את שם הקובץ (file.getName()) עם הנתיב המוחלט שלו (file.getAbsolutePath()) כדי ליצור את מחרוזת ה-Salt.
  + **מחזיר:** את מחרוזת ה-Salt שנוצרה.

## מחלקת Block

### **מטרה:**

* מייצגת בלוק יחיד של נתונים (16 בתים) בתוך תהליך ההצפנה/פענוח של AES.
* מאחסנת נתונים במבנה מערך תווים דו-מימדי המתאים למבנה אלגוריתם AES.

### **משתנים:**

* **data:** מערך תווים דו-מימדי (char[][]) עם ממדים 4 על 4, שתוכנן להחזיק בלוק נתונים של 16 בתים.

### **בנאי:**

* **Block(String text)**
  + מקבל מחרוזת (text) כקלט.
  + מאתחל את משתנה החבר data.
  + מעבד את מחרוזת הקלט:
    1. **מילוי:** אם text קצר מ-32 תווים (16 בתים), הוא מוסיף תווים '0' כדי להגיע לאורך הנדרש.
    2. **המרה:** ממיר זוגות של תווים הקסדצימליים מה-text המרופד לערכי התווים המתאימים שלהם ומאחסן אותם במערך data.

### **שיטות:**

* **getData()**
  + שיטת getter פשוטה לאחזור את המשתנה data.

## מחלקת AES

### תיאור

מחלקת ההצפנה/הפיענוח, מחלקה זו היא המחלקה עם הלוגיקה הראשית של התוכנה ונועדה כדי להצפין בלוקים של מידע ולפענח בלוקים של מידע.

### משתנים

Key – משתנה אשר שומר את מידע המפתח איתו אנחנו מצפינים את המידע, אנחנו צריכים משתנה זה בשביל המפתח המורחב בכל רמה.

Data – משתנה זה שומר את המידע שצריך להצפין/לפענח והקוד משנה את הבלוק בהתאם למבוקש מהאלגוריתם, נצטרך משתנה זה כדי להחזיר בסוף הפעולות את התוצאה.

### פעולות

encrypt() – הפעולה מקבלת מפתח ובלוק המידע שצריך להצפין, הפעולה שומרת את המפתח ואת הבלוק בזמן שימוש האלגוריתם, ומחזירה את בלוק המידע מוצפן.

Decrypt() – הפעולה מקבלת מפתח ובלוק המידע שצריך לפענח, הפעולה שומרת את המפתח ואת הבלוק בזמן שימוש האלגוריתם, ומחזיקה את הבלוק המידע המפוענח.

## מחלקת SBox

### תיאור

מחלקה זו שומרת במטריצה של תווים (מערך דו מימדי של תווים) את טבלת SBox, שמשתמשים בה באלגוריתם AES, מחלקה זו מאפשרת לקבל ערכים של הטבלה וערכים ההפוכים של הטבלה.

### משתנים

sboxTable – טבלת הSbox שמשתמשים באלגוריתם AES.

rsboxTable – טבלת הSbox ההפוכה שמשתמשים באלגוריתם AES.

### פעולות

getSboxValue(char num)– פעולה שמחזירה ערך בטבלת הSbox לפי הערך שקיבלה.

getSboxInvert(char num) – פעולה שמחזירה ערך בטבלת הSbox ההפוכה לפי הערך שקיבלה.

## מחלקת Rcon

### תיאור

בהצפנת AES בזמן ההצפנה בכל שלב מרחיבים ומשנים את המפתח כך שמצפינים את המידע 15 פעמים עם 15 מפתחות שונים שנוצרים מהמפתח הראשי, בשביל לעשות הרחבה משתמשים בטבלת הRcon או בשמה המלא Round Constant ובכך בכל שלב יש לנו משתנה גלובלי איתו אנחנו משנים את המפתח.

### משתנים

rconTable – טבלת הRcon שמשתמשים באלגוריתם AES.

### פעולות

getRconValue(char num)– פעולה שמחזירה ערך בטבלת ה Rcon לפי הערך שקיבלה.

## מחלקת LogTable

### תיאור

אלגוריתם AES בזמן ההצפנה משנה את הערכים לפי Log-ים ולכן נצטרך גם את טבלת הלוגים.

### משתנים

logTB – מערך חד מימדי של הלוגים של המספרים במידע.

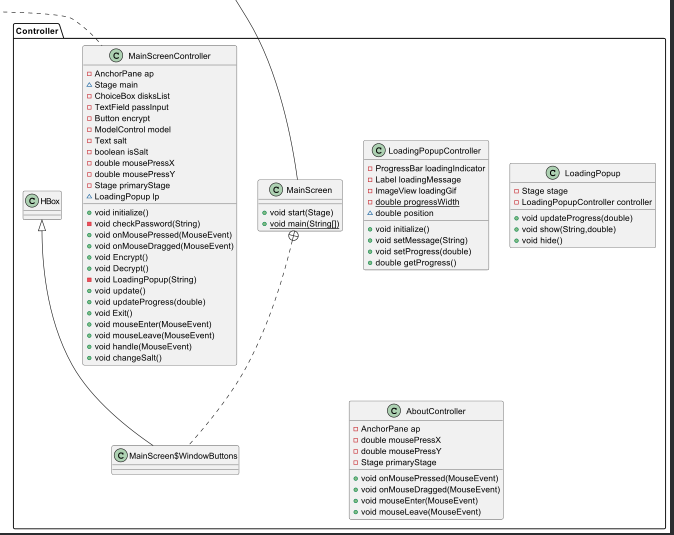
aLogTable – מערך חד מימדי של היפוך הלוגים.

### פעולות

getLog(int num)– פעולה שמחזירה ערך בטבלת הלוגים לפי הערך שקיבלה.

getALog(int num)– פעולה שמחזירה ערך בטבלת ההיפוך הלוגים לפי הערך שקיבלה.

# תרשים מחלקות (UML)



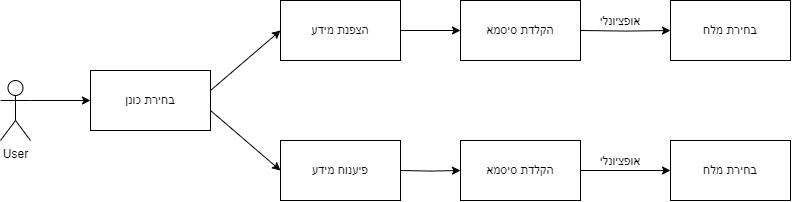
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# תרשים זרימה לשימוש התוכנה (Use Cases UML)



# תרשים מחלקות עליון (Top Down Level )

