



הצעת פרויקט לעבודת גמר

י"ד הנדסאי תוכנה (שאלון: 714918)

בנושא

הצפנה ושיתוף קבצים



שם הסטודנט: אופיר בן שימול

ת"ז: 215964610

מנחה: אילן פרץ

תאריך: 13/1/2026

מכללה: כנפי רוח, קרית נוער ירושלים (סמל מוסד: 140129)

תוכן עניינים

1. תיאור הנושא	2
2. רקע תיאורטי	3
3. תיאור הפרויקט	6
4. הגדרת הבעיה האלגוריתמית	7
5. הליכים עיקריים בפתרון בעיה בטכנולוגיות הנדסה מתקדמות	9
6. הליכים עיקריים בתחום למידת מכונה - לא רלוונטי	11
7. הליכים עיקריים במע' הפעלה / רשתות / תקשורת נתונים / אבטחת מידע	11
8. תיאור פרוטוקולי תקשורת	18
9. פיתוחים עתידיים	19
10. תיאור טכנולוגיה הנדסה	20
11. מסד נתונים	20
12. פרטים פורמליים	23

1. תיאור הנושא

תחום אבטחת המידע עוסק בהגנה על מידע דיגיטלי מפני גישה בלתי מורשית, שינוי, חשיפה או אובדן. בעידן הדיגיטלי המודרני, מידע אישי, עסקי וארגוני מאוחסן ומועבר באופן מקוון כחלק משגרה יומיומית, דבר ההופך אותו לנכס בעל ערך גבוה וליעד מרכזי לאיומים שונים. מידע זה כולל מסמכים, קבצים, נתונים אישיים ונתונים ארגוניים, אשר חשיפתם או פגיעתם עלולה לגרום לנזקים כלכליים, משפטיים ותדמיתיים.

מערכות מודרניות לאחסון ושיתוף קבצים מבוססות לרוב על מודל שרת/לקוח, שבה הקבצים נשמרים בשרת מרכזי ומונגשים למשתמשים דרך רשת האינטרנט. מודל זה מאפשר נוחות, זמינות ועבודה משותפת, אך יוצר בעיה הנדסית משמעותית בכל הנוגע לאבטחת מידע רגיש. במערכות רבות, השרת מחזיק בגישה מלאה לתוכן הקבצים, ולכן מהווה נקודת תורפה במקרה של פריצה, דליפת מידע, או שימוש לרעה מצד בעלי הרשאות לתשתית עצמה. בנוסף, שיתוף קבצים בין משתמשים או ארגונים מגדיל את משטח התקיפה ואת הסיכון לשינוי, העתקה או חשיפה של מידע ללא ידיעת בעליו. כתוצאה מכך, קיים צורך הנדסי בפיתוח פתרונות המאפשרים אחסון ושיתוף קבצים תוך צמצום אמון בשרת, והבטחת סודיות המידע גם בסביבות מרובות משתמשים ובתנאי עבודה יומיומיים.

כיום קיימות מערכות רבות לאחסון ושיתוף קבצים, הן לשימוש פרטי והן לשימוש ארגוני, כגון Google Drive, Dropbox, OneDrive ומערכות ענן ארגוניות דומות. מערכות אלו מאפשרות העלאה, הורדה ושיתוף של קבצים בין משתמשים, ולעיתים כוללות מנגנוני הרשאות וניהול משתמשים בסיסיים. עם זאת, ברוב המקרים המידע נשמר ומנוהל בשרת מרכזי בעל גישה מלאה לתוכן הקבצים, כך שספק השירות או בעלי גישה לתשתית יכולים לקרוא או לעבד את המידע. מצב זה יוצר תלות גבוהה באמינות השרת ובאמצעי האבטחה שלו, ומדגיש את הפער הקיים בין הנוחות התפעולית של מערכות אלו לבין רמת האבטחה הנדרשת בעת עבודה עם מידע רגיש.

כדי להתמודד עם האתגרים באחסון ושיתוף מידע רגיש, תחום אבטחת המידע מגדיר מספר עקרונות יסוד המנחים את תכנון המערכות ההנדסיות. העיקרון המרכזי הוא סודיות (Confidentiality), אשר מטרתו להבטיח כי רק גורמים מורשים יוכלו לצפות בתוכן המידע. עיקרון נוסף הוא שלמות המידע (Integrity), המבטיח שהמידע לא שונה, נמחק או נפגע במהלך האחסון או ההעברה, ללא זיהוי. לצד אלו קיים עיקרון אימות הזהות (Authentication), המאפשר לוודא את זהות המשתמשים במערכת, ועיקרון ההרשאות (Authorization), הקובע אילו פעולות מותרות לכל משתמש ביחס למידע. עקרונות אלו מהווים בסיס תיאורטי הכרחי לכל מערכת העוסקת במידע רגיש, והם מגדירים את הדרישות ההנדסיות שעל הפתרון לעמוד בהן, עוד לפני בחירת טכנולוגיות או אלגוריתמים ספציפיים.

2. רקע תיאורטי

אחד המרכיבים המרכזיים בפתרונות אבטחת מידע הקיימים כיום הוא שימוש בהצפנה לצורך הגנה על מידע דיגיטלי. הצפנה מאפשרת להפוך מידע קריא לייצוג בלתי קריא, כך שרק גורמים מורשים יוכלו לשחזר את המידע למצבו המקורי. בתחום זה מקובל להבחין בין שתי שיטות עיקריות: **הצפנה סימטרית והצפנה אסימטרית**, שלכל אחת מהן מאפיינים, יתרונות ושימושים שונים.

הצפנה סימטרית מבוססת על מפתח יחיד המשמש הן להצפנה והן לפענוח של המידע. שיטה זו מאופיינת בביצועים גבוהים ויעילות חישובית, ולכן מתאימה במיוחד להצפנת נפחי מידע גדולים, כגון קבצים ומסדי נתונים. עם זאת, האתגר המרכזי בהצפנה סימטרית הוא ניהול המפתח, שכן יש להעביר את המפתח בין הצדדים בצורה מאובטחת מבלי לחשוף אותו לגורמים בלתי מורשים. פתרונות קיימים נדרשים להתמודד עם בעיה זו באמצעות מנגנונים משלימים לניהול והעברת מפתחות.

הצפנה אסימטרית, לעומת זאת, מבוססת על זוג מפתחות: מפתח ציבורי ומפתח פרטי. המפתח הציבורי משמש להצפנה, בעוד שהמפתח הפרטי משמש לפענוח. שיטה זו מאפשרת העברת מידע בצורה מאובטחת גם בין גורמים שלא שיתפו סוד מראש, ולכן נפוצה במיוחד בתהליכי אימות זהות והעברת מפתחות. עם זאת, הצפנה אסימטרית דורשת חישובים מורכבים יותר, ולכן אינה מתאימה להצפנת כמויות גדולות של מידע, אלא משמשת לרוב כחלק ממנגנון משולב עם הצפנה סימטרית.

בנוסף להצפנה, פתרונות אבטחת מידע עושים שימוש ב**פונקציות Hash** קריפטוגרפיות, אשר אינן מיועדות להסתרת המידע אלא לאימות שלמותו. פונקציית Hash מקבלת קלט באורך משתנה ומפיקה ערך תקציר באורך קבוע, כך שכל שינוי קטן בקלט גורם לשינוי משמעותי בתוצאה. באמצעות פונקציות אלו ניתן לזהות האם מידע עבר שינוי, פגיעה או זיוף במהלך האחסון או ההעברה. פונקציות Hash מהוות רכיב בסיסי בבדיקת שלמות נתונים, אימות זהויות וחתימות דיגיטליות, והן משלימות את תהליך האבטחה הכולל של המידע.

להלן אלגוריתמים המהווים פתרון אשר הוזכר קודם בנושא זה:

AES הוא אחד האלגוריתמים המרכזיים המספקים פתרון ישיר לבעיה של הגנה על תוכן הקובץ עצמו. זהו אלגוריתם הצפנה סימטרית, המשתמש באותו מפתח להצפנה ולפענוח, ומאפשר להפוך כל קובץ לקריאה בלתי אפשרית ללא מפתח מתאים. היתרון המרכזי שלו הוא יעילות גבוהה ומהירות ביצוע, גם כאשר עובדים עם קבצים גדולים או עם כמות גבוהה של פעולות הצפנה ביום עבודה שוטף. השימוש ב-AES מאפשר להצפין את הקובץ כבר בצד הלקוח, לפני העלאתו לשרת, כך שגם אם הקובץ נחשף, יורט או נשמר בסביבה שאינה מאובטחת לחלוטין, לא ניתן יהיה להבין ממנו דבר. בכך האלגוריתם נותן מענה ברור לבעיה שהוצגה בפרק הקודם: שמירה על סודיות הקובץ גם מחוץ למסגרת הארגונית.

שילוב של חוזק ההצפנה יחד עם ביצועים יציבים הופך את AES לכלי משמעותי במערכות לשיתוף קבצים רגישים, ומספק שכבת הגנה בסיסית אך הכרחית בתהליך העבודה.

Camellia הוא אלגוריתם הצפנה סימטרית נוסף. האלגוריתם מקבל מידע קריא והופך אותו לרצף בלתי מובן באמצעות מפתח יחיד המשמש גם להצפנה וגם לפענוח. Camellia פועל בבלוקים של 128 ביט ומשלב מספר סבבים של החלפה, ערבוב ופיזור נתונים, באופן המבטיח שגם במקרה של יירוט הקובץ לא ניתן יהיה להבין את תוכנו ללא המפתח המתאים. בזכות יעילות גבוהה ויציבות בביצועים, Camellia מתאים להצפנת קבצים שלמים או נתונים בהיקף גדול, ומהווה שכבת הגנה מרכזית בתהליך אבטחת המידע.

RSA נותן מענה לבעיה משלימה בתהליך, העברת מפתחות ההצפנה בצורה מאובטחת בין משתמשים. האלגוריתם מבוסס על זוג מפתחות, ציבורי ופרטי, ומאפשר ליצור ערוץ תקשורת שבו ניתן לשלוח את מפתח ההצפנה מבלי לחשוף אותו לגורם לא מורשה. הרעיון פשוט, המשתמש מצפין את מפתח הסימטרי באמצעות המפתח הציבורי של הנמען, ורק המפתח הפרטי של אותו נמען מסוגל לפענח אותו. בצורה זו, גם אם השרת או כל גורם אחר יירט את ההודעה בדרך, הוא לא יוכל להבין את המפתח. כך האלגוריתם פותר את אחת הבעיות הקריטיות במערכות הצפנת קבצים שהיא: כיצד לשתף מפתחות בצורה בטוחה לחלוטין, בלי לסמוך על השרת מבלי לסכן את סודיות הקובץ. השימוש באלגוריתם זה מפריד בין יכולת האחסון לבין היכולת לקרוא את המידע, ומחזיר את השליטה המלאה למשתמשים המורשים בלבד.

(ECC) Elliptic Curve Cryptography הוא אלגוריתם הצפנה אסימטרית המשמש להעברת המפתח הסימטרי בצורה מאובטחת בין משתמשים. האלגוריתם מבוסס על זוג מפתחות ציבורי ופרטי כאשר המפתח הציבורי משמש להצפנה והמפתח הפרטי משמש לפענוח. האלגוריתם מאפשר לשני צדדים ליצור מפתח משותף מבלי לחשוף את המידע המתמטי העומד מאחוריו, ובכך מונע מגורמים לא מורשים לשחזר את המפתח גם אם הם מאזינים לתקשורת. האלגוריתם מספק רמת אבטחה גבוהה תוך שימוש במפתחות קצרים ויעילים, ולכן מותאם במיוחד לסביבות רשת שבהן יש צורך להעביר מפתחות הצפנה בצורה בטוחה ומהירה.

SHA-256 פותר בעיה שונה לחלוטין מזו של הצפנה, הוא אינו מסתיר את תוכן הקובץ, אלא מבטיח שהקובץ לא שונה במהלך הדרך. האלגוריתם מייצר "טביעת אצבע" ייחודית לכל קובץ, כך ששינוי של אפילו ביט יחיד ייצור טביעת אצבע שונה לחלוטין. בצורה זו ניתן לוודא שהקובץ שהורד או נשמר בשרת הוא בדיוק אותו קובץ שהועלה במקור, ללא זיופים, מניפולציות או פגיעה בשלמות המידע. שימוש באלגוריתם זה מאפשר גם אימות זהויות ויצירת חתימות דיגיטליות, כך שמשתמשים יכולים לבדוק שמפתח, בקשה או פעולה מסוימת אכן הגיעו ממקור אמיתי. אלגוריתם זה מבטיח שהמידע נותר מדויק

ואמין. שילוב של שלושת האלגוריתמים יחד יוצר מערכת מאובטחת שאינה מסתמכת רק על הצפנת התוכן, אלא גם על מנגנון חזק המגן מפני שינויים וטעויות.

Whirlpool הוא אלגוריתם **Hash cryptographic** המשמש ליצירת טביעת אצבע דיגיטלית של הקובץ. האלגוריתם מקבל את תוכן הקובץ ומפיק ממנו ערך ייחודי וחד כיווני באורך 512 ביט, המייצג את מבנה הנתונים באופן מדויק. שינוי קטן ביותר בקובץ, אפילו ביט יחיד, יוצר ערך Hash שונה לחלוטין, ולכן אלגוריתם זה מאפשר לזהות האם הקובץ נשאר שלם או עבר שינוי במהלך ההעברה או האחסון. בשילוב עם חתימה דיגיטלית ניתן גם לאמת את מקור המידע וגם להבטיח שלא בוצעו בו מניפולציות, מה שהופך את האלגוריתם לכלי מרכזי לשמירה על אמינות ושלמות המידע במערכת.

להלן טבלה אשר מציגה השוואה בין אלגוריתמים קריפטוגרפיים שונים, תוך פירוט היתרונות והחסרונות של כל אלגוריתם בהיבטים של אבטחה, יעילות ושימושיות.

אלגוריתם	סוג	יתרונות	חסרונות
AES	סימטרי	<ul style="list-style-type: none"> ביצועים גבוהים בהצפנת נפחי מידע גדולים מבנה פשוט ליישום עמידות גבוהה בפני התקפות ידועות 	<ul style="list-style-type: none"> תלות מלאה בשמירה על סודיות המפתח העברת המפתח דורשת מנגנון אבטחה נפרד
Camellia	סימטרי	<ul style="list-style-type: none"> רמת אבטחה גבוהה עם פיזור ובלבול חזקים מתאים להצפנת קבצים שלמים יציב גם בעומסי עבודה גבוהים 	<ul style="list-style-type: none"> מורכבות פנימית גבוהה יותר של שלבי ההצפנה דורש ניהול מפתח זהה להצפנה ולפענוח
RSA	א-סימטרי	<ul style="list-style-type: none"> מאפשר העברת מפתחות ללא שיתוף סוד מוקדם הפרדה ברורה בין הצפנה לפענוח 	<ul style="list-style-type: none"> חישובים כבדים אינו מתאים להצפנת קבצים גדולים
ECC	א-סימטרי	<ul style="list-style-type: none"> רמת אבטחה גבוהה עם מפתחות קצרים יעילות חישובית טובה בתקשורת רשת 	<ul style="list-style-type: none"> מימוש מורכב יותר מבחינה מתמטית רגיש לטעויות בהגדרת פרמטי העקום

<ul style="list-style-type: none"> • אינו מספק הצפנה או הסתרת מידע • אינו מאפשר שחזור נתונים 	<ul style="list-style-type: none"> • מזהה כל שינוי קטן בנתונים • פלט קבוע באורך ידוע 	בדיקת שלמות ואימות מידע	SHA-256
<ul style="list-style-type: none"> • דורש משאבי חישוב גבוהים יותר • אינו מספק הצפנה או ניהול מפתחות 	<ul style="list-style-type: none"> • פלט ארוך במיוחד המקטין סיכוי להתנגשות רגישות גבהה לשינויים בקלט 	בדיקות שלמות ואימות מידע	Whirlpool

3. תיאור הפרויקט

המערכת היא מערכת לניהול ושיתוף קבצים רגישים, המיועדת למשתמשים וארגונים הזקוקים לרמת אבטחה גבוהה בעת עבודה עם מידע דיגיטלי. מטרת המערכת היא לאפשר אחסון, שיתוף וגישה לקבצים בצורה מבוקרת, תוך שמירה על סודיות המידע ושלמותו גם כאשר הקבצים נשמרים ומועברים דרך תשתיות חיצוניות. המערכת נועדה לתת מענה לצורך של משתמשים המעוניינים לשלוט בגישה למידע שלהם, מבלי לחשוף את תוכן הקבצים לגורמים שאינם מורשים.

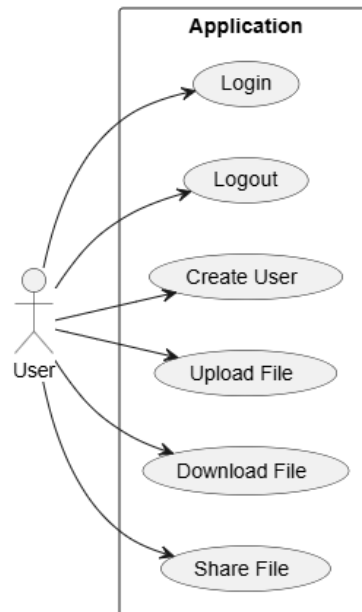
המערכת מיועדת לשימוש של משתמשים פרטיים וארגוניים, כגון עובדים בארגון, צוותים מקצועיים או גופים המטפלים במידע רגיש. כל משתמש פועל מתוך חשבון אישי, ויכול להיות חלק מארגון או קבוצה מוגדרת. המשתמשים יכולים להעלות קבצים, לשמור אותם במערכת, ולשתף אותם עם משתמשים אחרים בהתאם להרשאות שנקבעו. השימוש במערכת מתבצע דרך ממשק Web פשוט וברור, המאפשר עבודה נוחה ללא צורך בידע טכני מוקדם.

מנקודת מבטו של המשתמש, תהליך העבודה במערכת כולל מספר פעולות עיקריות. תחילה המשתמש נרשם למערכת ומתחבר לחשבון. לאחר ההתחברות הוא יכול להעלות קבצים מהמחשב האישי, לצפות ברשימת הקבצים השמורים, ולהגדיר עם מי הקבצים ישותפו. המשתמש יכול לבחור אילו משתמשים אחרים יוכלו לגשת לקובץ מסוים, ולהגביל את הגישה בהתאם לצרכים שלו. כאשר משתמש מקבל גישה לקובץ, הוא יכול להוריד אותו או לצפות בו, בהתאם להרשאות שניתנו לו.

הבעיה המרכזית שהמערכת פותרת עבור המשתמש היא חוסר האמון במערכות אחסון רגילות, שבהן ספק השירות או גורמים בעלי גישה לתשתית יכולים להיחשף לתוכן הקבצים. במערכות רגילות, המשתמש נדרש לסמוך על השרת שישמור על סודיות המידע, דבר שאינו תמיד מספק בעת עבודה עם מידע רגיש. המערכת מצמצמת תלות זו ומאפשרת למשתמשים לעבוד עם קבצים בצורה מאובטחת יותר, תוך שליטה טובה יותר בהרשאות ובגישה למידע.

באופן זה המערכת מספקת פתרון שימושי וברור למשתמשים הזקוקים לשילוב בין נוחות עבודה לבין רמת אבטחה גבוהה, ומאפשרת עבודה יומיומית עם קבצים רגישים בסביבה דיגיטלית מבוקרת.

להלן תרשים מסוג **Usecase** אשר מסביר אילו פעולות המשתמש יכול לבצע.



המשתמש הוא הגורם החיצוני היחיד הפועל מול המערכת, והוא יכול לבצע פעולות של יצירת משתמש (Create User), התחברות (Login) לצורך זיהוי ואימות, וניתוק (Logout) לסיום עבודה מאובטח. לאחר ההתחברות המשתמש רשאי להעלות קבצים למערכת (Upload File), להוריד קבצים אל המחשב האישי (Download File) ולשתף קבצים עם משתמשים אחרים בהתאם להרשאות שניתנו (Share File). התרשים ממחיש בצורה פשוטה וברורה את תחומי האחריות של המערכת ואת הפעולות הזמינות למשתמש, מבלי להיכנס לאופן המימוש הפנימי של כל פעולה.

4. הגדרת הבעיה האלגוריתמית

הבעיה האלגוריתמית שבה עוסקת מערכת זו נוגעת לאבטחת מידע דיגיטלי בעת אחסון ושיתוף קבצים בין משתמשים שונים, בסביבה שאינה בהכרח מאובטחת. מדובר בבעיה מורכבת, הנובעת מהצורך לאפשר זרימה חופשית ונוחה של מידע בין גורמים מורשים, תוך מניעה מוחלטת של גישה, שינוי או ניצול של המידע על ידי גורמים בלתי מורשים. הבעיה אינה עוסקת רק בהגנה על המידע עצמו, אלא גם בשמירה על אמינותו, שלמותו והיכולת לשלוט בגישה אליו לאורך זמן.

באופן פורמלי, ניתן להגדיר את הבעיה כך: נתון קובץ דיגיטלי המיועד לאחסון או שיתוף בין משתמשים, וכן אוסף של משתמשים בעלי רמות הרשאה שונות. נדרש תהליך אלגוריתמי אשר יפיק ייצוג מאובטח

של הקובץ, כך שתוכן הקובץ לא יהיה קריא או ניתן לשחזור על ידי גורם שאינו מורשה, גם אם יש לו גישה לנתונים המאוחסנים או המועברים. במקביל, על התהליך להבטיח שמשתמש מורשה יוכל, בתנאים המתאימים, לשחזר את הקובץ למצבו המקורי ללא פגיעה בנתונים.

מרכיב מרכזי בבעיה האלגוריתמית הוא הצורך להבטיח שלמות מידע. כלומר, יש לוודא כי הקובץ שהתקבל על ידי משתמש הוא זהה לקובץ המקורי שנשמר או נשלח, ללא שינוי, השמטה או הוספה של נתונים. האלגוריתם נדרש לאפשר זיהוי חד-משמעי של כל שינוי שנעשה בקובץ, בין אם במכוון ובין אם כתוצאה מתקלה או תקיפה, ולהתריע על כך לפני שימוש במידע.

מורכבות הבעיה גוברת לאור העובדה שהמידע עשוי לעבור דרך רשתות פתוחות ולהישמר בתשתיות חיצוניות, אשר אינן בשליטת בעלי המידע. לכן, האלגוריתם נדרש לצמצם את התלות באמון בגורמים חיצוניים, ולפעול בצורה שמבטיחה את אבטחת המידע גם בסביבה עוינת. במצב זה, אין להניח שהשרת או אמצעי האחסון הם מהימנים, ועל האלגוריתם להבטיח שהמידע יישאר מוגן גם אם תשתית זו תיפגע.

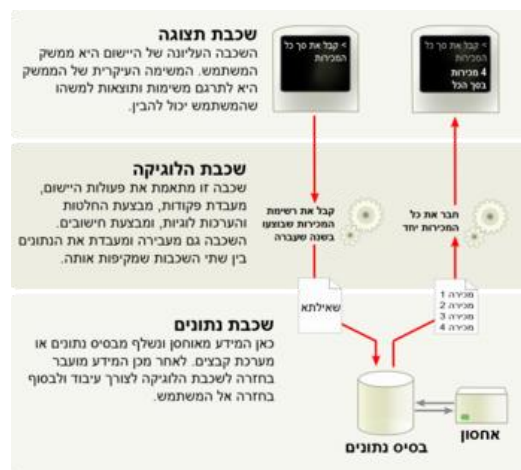
הבחירה באלגוריתמים **Camellia, ECC, Whirlpool** נובעת מהצורך לבנות מערכת מאובטחת, יעילה וגמישה המתאימה לסביבת עבודה מודרנית. Camellia מעניק הצפנה סימטרית חזקה ומהירה, הדומה ברמת האבטחה שלה ל-AES, ומאפשר להצפין קבצים שלמים בצורה יציבה ויעילה גם בהיקפי מידע גדולים. ECC נבחר כפתרון להצפנה אסימטרית ולהעברת מפתחות מאובטחת, בזכות היכולת שלו לספק רמת אבטחה גבוהה באמצעות מפתחות קצרים במיוחד, מה שמיעל את ביצועי המערכת ומקטין עומסי חישוב לעומת חלופות מסורתיות. Whirlpool משלים את המערכת באמצעות מנגנון Hash אמין היוצר טביעת אצבע ייחודית לכל קובץ, ומבטיח יכולת בדיקת שלמות וזיהוי שינויים באופן מדויק. שילוב של שלושת האלגוריתמים יוצר פתרון הכולל הצפנה, העברת מפתחות ובדיקת שלמות שלוש המרכיבים החיוניים לאבטחת מידע רגיש במערכת לשיתוף קבצים.

לסיכום, הבעיה האלגוריתמית של מערכת זו משלבת מספר אתגרים מרכזיים: שמירה על סודיות המידע, הבטחת שלמותו. פתרון הבעיה מחייב הגדרה מדויקת של רצף פעולות אלגוריתמיות, אשר יטפל בכל אחד מהאתגרים הללו באופן משולב ועקבי, ויאפשר עבודה בטוחה עם קבצים רגישים לאורך כל מחזור חייהם.

5. הליכים עיקריים בפתרון בעיה בטכנולוגיות הנדסה מתקדמות

מערכת זו מבוססת על **מודל שלוש שכבות**, אשר מחלק את מבנה התוכנה לשלושה חלקים נפרדים, כך שלכל שכבה יש אחריות מוגדרת וברורה. בנוסף לכך, המערכת פועלת במבנה של שרת לקוח (Client-Server), שבו קיימת הפרדה ברורה בין צד הלקוח (Frontend) לבין צד השרת (Backend). צד הלקוח אחראי על הצגת הממשק למשתמש, קליטת פעולותיו ושליחת בקשות למערכת, בעוד שצד השרת אחראי על עיבוד הבקשות, ניהול הלוגיקה העסקית, בקרת הרשאות ואינטראקציה עם מסד הנתונים. שילוב זה יוצר יישום מסוג **Full-Stack**, שבו חל כל שכבות המערכת, החל מממשק המשתמש ועד לניהול הנתונים והלוגיקה, ממומשות כחלק ממערכת אחת שלמה ופועלות יחד במבנה מבוקר ומסודר.

להלן תרשים המתאר את רצף הזרימה בין קצה לקצה:



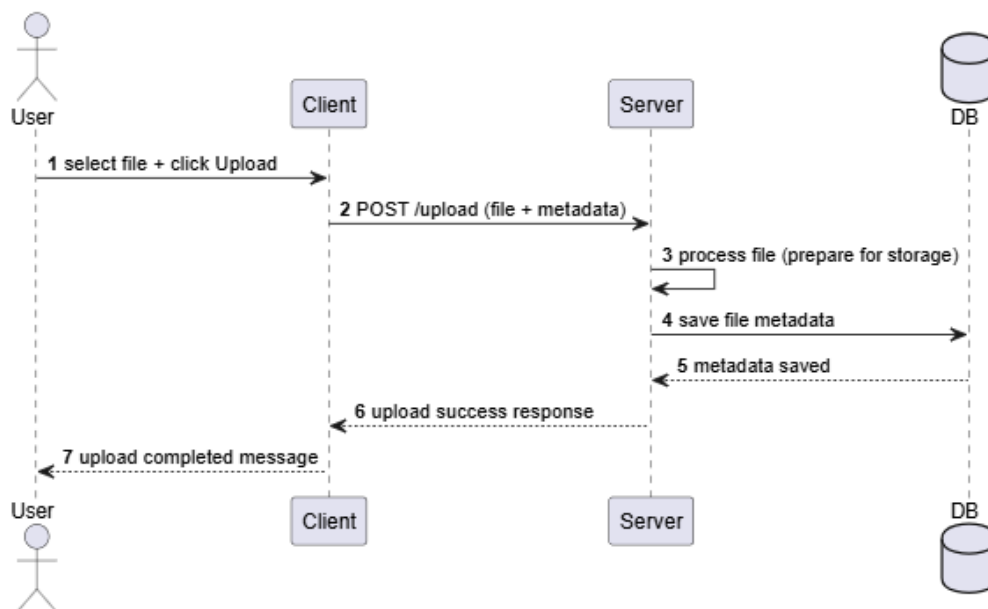
שכבת התצוגה אחראית על הצגת הממשק למשתמש ועל קליטת פעולותיו, כגון התחברות למערכת, העלאת קבצים, צפייה בקבצים ושיתופם. בפרויקט זה שכבה זו ממומשת כחלק מיישום ה-Web המבוסס על Spring Boot, אשר אחראי על ניהול הבקשות מהדפדפן והצגת התגובה המתאימה למשתמש. Spring Boot מטפל בתקשורת בין הדפדפן לשרת ובניהול זרימת הנתונים בין שכבת התצוגה לשכבות הפנימיות של המערכת, ללא צורך במימוש מנגנון תקשורת חיצוני נפרד. שכבה זו מתמקדת באינטראקציה עם המשתמש ואינה כוללת לוגיקה עסקית או גישה ישירה למסד הנתונים.

שכבת הלוגיקה בה מתבצע כל העיבוד ההנדסי שמגדיר את ערך המערכת: קבלת פעולות מהמשתמש (העלאה, שיתוף, הורדה), בדיקת הרשאות, ניהול תהליך ההצפנה והפענוח. בפרויקט זה שכבה זו ממומשת בקוד Java בתוך Spring Boot, באמצעות **מחלקות שירות** (Services) שמרכזות את כל החוקים והשלבים של התהליך הקריפטוגרפי והעסקי, ומבצעות הפרדה ברורה בין שכבת התצוגה לבין שכבת הלוגיקה. בנוסף, שכבה זו אחראית על עבודה עם אובייקטים של נתונים שמועברים בתוך המערכת בפורמט JSON, ועל תרגום בינם לבין המבנים הפנימיים של המערכת, כך שהזרימה נשארת מסודרת, עקבית וניתנת לתחזוקה.

שכבת נתונים אחראית על שמירה, שליפה וניהול של כל המידע שהמערכת מייצרת ומשתמשת בו, כולל נתוני משתמשים וארגונים, מידע של קבצים, הרשאות שיתוף, ערכי Hash, מפתחות מוצפנים ומידע נלווה נוסף. בפרויקט זה שכבה זו ממומשת באמצעות מסד נתונים מסוג NoSQL, המאפשר אחסון גמיש של מסמכים ומבנים משתנים בהתאם לצרכי המערכת, מבלי להיות מוגבל לסכמה קשיחה כמו במסדי נתונים. בתוך Spring Boot, שכבה זו מיושמת באמצעות **ממשקי גישה לנתונים** (Repository), האחראיות לבצע פעולות CRUD בצורה מבודדת משכבת הלוגיקה, כך שהלוגיקה העסקית אינה תלויה בפרטי האחסון. **Maven** משמש לניהול התלויות של הפרויקט ולהטמעת ספריות החיבור למסד הנתונים, מה שמאפשר בנייה מסודרת, תחזוקה קלה והרחבה עתידית של המערכת.

היתרון המרכזי במודל זה הוא הפרדה ברורה בין אחריות השכבות, מה שמאפשר פיתוח יעיל יותר, תחזוקה פשוטה, והרחבת המערכת בעתיד ללא תלות בחלקים אחרים. בכל תהליך עבודה המידע זורם מהמשתמש אל שכבת הלוגיקה, ומשם אל שכבת הנתונים וחוזר אחורה לאחר העיבוד.

התרשים הבא מציג **תרשים רצף** (Sequence Diagram) המתאר את תהליך העלאת קובץ במערכת, מנקודת מבט של זרימת הפעולות בין המשתמש, צד הלקוח, צד השרת ומסד הנתונים. התרשים ממחיש את סדר הפעולות הלוגי מרגע בחירת הקובץ על ידי המשתמש ועד לסיום תהליך ההעלאה ואחסון המידע הנלווה במערכת, תוך הפרדה ברורה בין אחריות הרכיבים השונים.



בתחילת התהליך המשתמש בוחר קובץ ולוחץ על פעולת ההעלאה דרך ממשק הלקוח. צד הלקוח שולח בקשת העלאה לשרת הכוללת את הקובץ ואת המידע הנלווה אליו. השרת מקבל את הבקשה ומעבד את הקובץ לצורך הכנה לאחסון ושומר במסד הנתונים את המידע של הקובץ, כגון מזהה, בעלות ומידע תיאורי. עם השלמת הפעולה, השרת מחזיר ללקוח תגובת הצלחה, והלקוח מציג למשתמש הודעה המאשרת כי הקובץ הועלה בהצלחה למערכת.

6. הליכים עיקריים בתחום למידת מכונה - לא רלוונטי

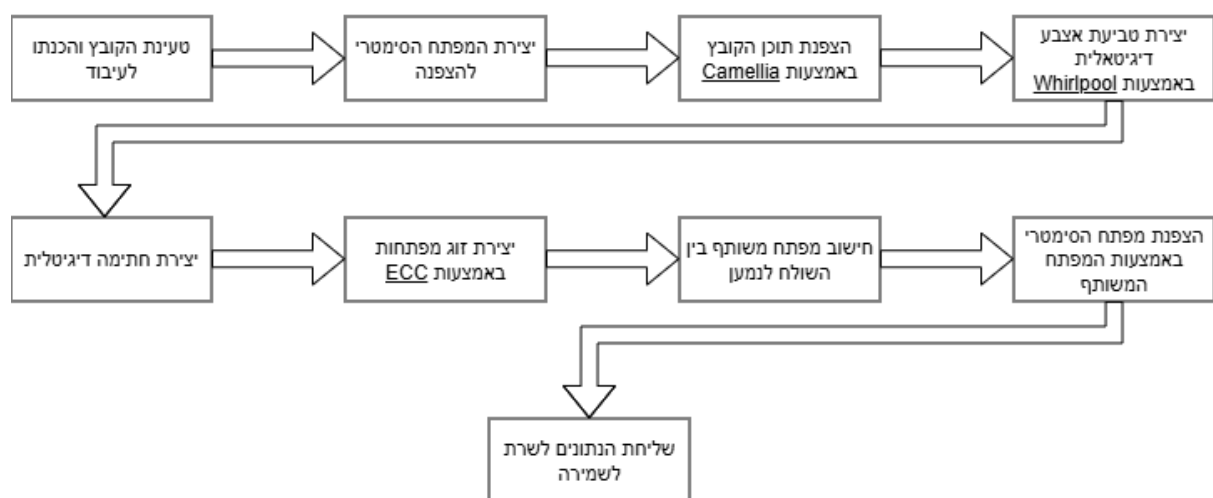
הצעה זו לא עוסקת בלמידת מכונה, לכן סעיף זה לא רלוונטי

7. הליכים עיקריים במע' הפעלה / רשתות / תקשורת / אבטחת מידע

הערה חשובה: ההתייחסות למודל Client/Server מופיעה בהרחבה בסעיף 5 של הצעה.

כדי להמחיש כיצד עקרונות המערכת מיושמים בפועל בתהליך הנדסי מובנה. לצורך כך הוגדר תהליך הצפנת הקובץ, המתואר באמצעות תרשים זרימה, אשר מציג את רצף השלבים מרגע בחירת הקובץ על ידי המשתמש ועד לשמירתו בצורה מאובטחת בשרת. התרשים מאפשר הבנה ויזואלית של חלוקת האחריות בין הלקוח לשרת ושל השימוש באלגוריתמים הקריפטוגרפיים השונים. בהמשך פרק זה יפורט כל שלב בתרשים בנפרד, תוך הסבר מעמיק על פעולתו, תפקידו בתהליך, והאופן שבו האלגוריתמים הקריפטוגרפיים פועלים ברמה העקרונית.

תהליך ההצפנת הקובץ



שלב 1 - טעינת הקובץ והכנתו לעיבוד

בשלב הראשון בתהליך ההצפנה, המשתמש בוחר קובץ מקומי מהמחשב או מהתקן אחסון אחר ומעלה אותו למערכת דרך ממשק ה-Web. הקובץ נטען לזיכרון בצד הלקוח, לפני שליחתו לשרת, וזאת על מנת למנוע חשיפה של תוכן הקובץ במצב גולמי לגורם חיצוני. בשלב זה מתבצעת הכנה ראשונית של הקובץ לעיבוד קריפטוגרפי, הכוללת קריאה של תוכן הקובץ כזרם נתונים בינארי (bytes) והגדרת מבנה אחיד המאפשר הפעלת אלגוריתמים קריפטוגרפיים עליו. שלב זה אינו כולל הצפנה בפועל, אלא

מהווה בסיס הנדסי הכרחי להמשך התהליך, בכך שהוא מבטיח שהקובץ יטופל בצורה מבוקרת, עקבית ובלתי תלויה בסוגו או בגודלו.

שלב 2 - יצירת המפתח הסימטרי להצפנה

בשלב זה המערכת יוצרת מפתח הצפנה סימטרי **אקראי**, אשר ישמש להצפנת תוכן הקובץ. המפתח נוצר בצד הלקוח באמצעות **מחולל מספרים אקראיים קריפטוגרפי**, במטרה להבטיח רמת אבטחה גבוהה ולמנוע אפשרות לניחוש או שחזור המפתח. מפתח זה הוא מפתח זמני וייחודי לכל קובץ, ואינו עושה שימוש במפתחות קבועים או משותפים מראש. הבחירה בהצפנה סימטרית בשלב זה נובעת מהיעילות הגבוהה שלה בעבודה עם קבצים גדולים, ומהיכולת להצפין נפחי מידע משמעותיים בזמן קצר. יצירת המפתח בשלב נפרד מאפשרת שליטה מלאה בתהליך ההצפנה ומהווה בסיס לשילוב מאוחר יותר עם מנגנוני אבטחה נוספים לניהול והגנה על המפתח עצמו.

שלב 3 - הצפנת תוכן הקובץ באמצעות Camellia

לאחר יצירת המפתח הסימטרי, מתבצעת הצפנת תוכן הקובץ באמצעות אלגוריתם Camellia. אלגוריתם זה הוא אלגוריתם הצפנה סימטרי מסוג Block Cipher, הפועל על בלוקים בגודל קבוע של נתונים ומצפין כל בלוק באמצעות המפתח שנוצר בשלב הקודם. תהליך ההצפנה כולל חלוקה של תוכן הקובץ לבלוקים, והפעלת סדרת סבבים (Rounds) על כל בלוק, כאשר בכל סבב מתבצעות פעולות מתמטיות ולוגיות כגון ערבוב נתונים, החלפות והצפנה באמצעות תתי מפתחות הנגזרים מהמפתח הראשי.

המטרה של תהליך זה היא להפוך את הנתונים המקוריים לנתונים בלתי קריאים לחלוטין עבור כל גורם שאין ברשותו את המפתח המתאים. הבחירה ב-Camellia נובעת מהיותו אלגוריתם מאובטח ויעיל, המתאים להצפנת קבצים גדולים, ומאפשר שמירה על סודיות תוכן הקובץ גם אם הנתונים המוצפנים נחשפים או נשלחים דרך רשת לא מאובטחת.

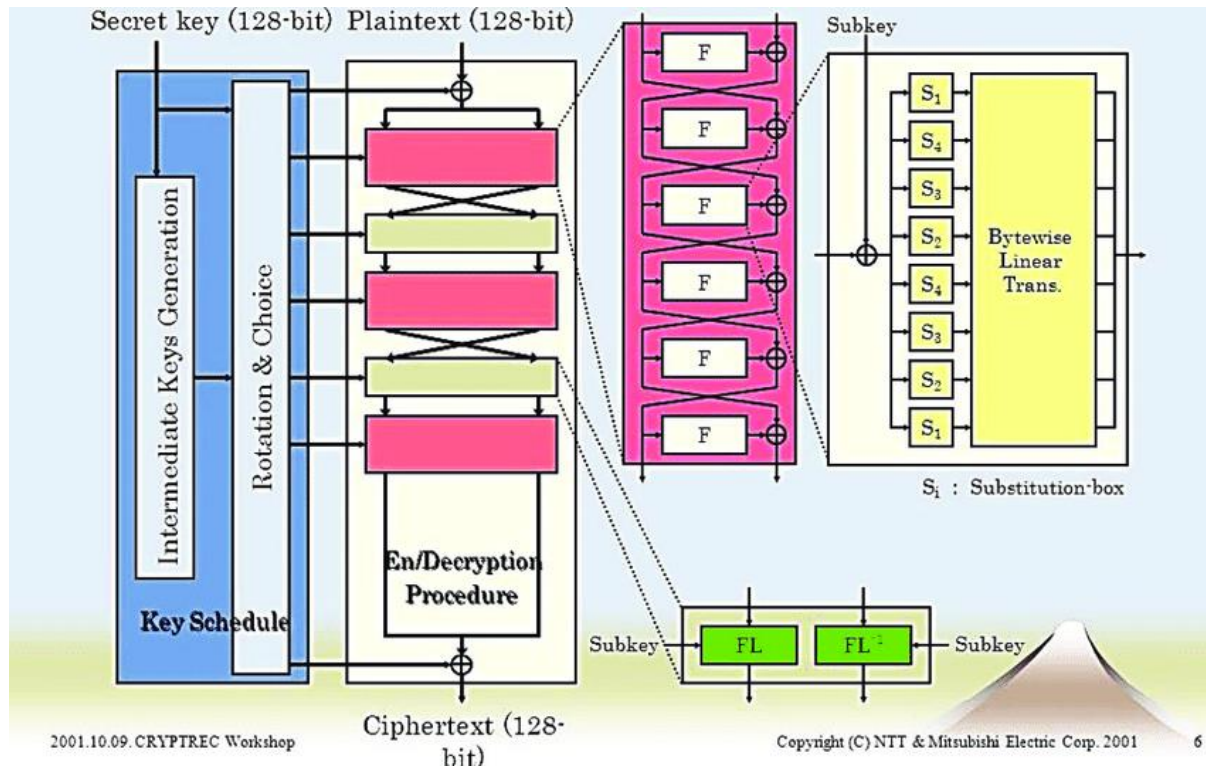
אלגוריתם Camellia

אלגוריתם זה משמש להצפנה סימטרית מסוג Block Cipher, אשר פועל על בלוקים קבועים של נתונים בגודל 128 ביט. בשלב הראשון, תוכן הקובץ מחולק לרצף של בלוקים בינאריים בגודל זהה, כאשר כל בלוק מוצפן בנפרד באמצעות אותו מפתח סימטרי. המפתח הראשי עובר תהליך של הרחבת מפתח (Key Schedule) שבמהלכו נגזרים ממנו תתי מפתחות שונים המשמשים בכל אחד מסבבי ההצפנה.

תהליך ההצפנה עצמו מתבצע במספר סבבים עוקבים, כאשר בכל סבב מופעלות פונקציות קריפטוגרפיות הכוללות פעולות של החלפה (Substitution) באמצעות טבלאות קבועות, ערבוב ביטים (Permutation), ופעולות XOR בין נתוני הבלוק לתתי המפתחות. פעולות אלו נועדו ליצור בלבול (Confusion) ופיזור (Diffusion) של הנתונים, כך שכל שינוי קטן בנתוני הקלט או במפתח גורם לשינוי

משמעותי בפלט. בסיום הסבבים מתקבל בלוק מוצפן שאינו משמר קשר ישיר או ברור לנתונים המקוריים. תהליך זה חוזר על עצמו עבור כל בלוק בקובץ, עד להצפנת הקובץ כולו.

להלן תמונה אשר מייצגת את תהליך ההצפנה:



התרשים מציג את מבנה ההצפנה של אלגוריתם Camellia ואת זרימת המידע בתהליך ההצפנה והפענוח. בצד שמאל מופיע שלב יצירת תתי-המפתחות (Key Schedule) שבו המפתח הסימטרי הראשי עובר עיבוד ונגזרים ממנו מפתחות משנה המשמשים לאורך הסבבים. במרכז התרשים מוצג תהליך ההצפנה עצמו, שבו בלוק הנתונים עובר סדרת סבבים חוזרים הכוללים פונקציות פנימיות (F) ופעולות טרנספורמציה, כאשר בכל סבב משולב תת מפתח שונה. בצד הימני של התרשים ניתן לראות את מבנה פונקציית ההחלפה והערבוב, הכוללת שימוש ב-S-boxes ופעולות ליניאריות, שמטרתן ליצור פיזור ובלבול של הנתונים. השילוב בין סבבים חוזרים, תתי מפתחות וטרנספורמציות קריפטוגרפיות מבטיח שהקשר בין הקלט לפלט יטושטש לחלוטין, כפי שנדרש מאלגוריתם ההצפנה סימטרי חזק.

שלב 4 - יצירת חתימה דיגיטלית באמצעות Whirlpool

לאחר הצפנת תוכן הקובץ, המערכת יוצרת חתימה דיגיטלית שמטרתה להבטיח את שלמות המידע ולאפשר זיהוי של שינוי או פגיעה בקובץ המוצפן. בשלב זה מופעלת פונקציית ה-Hash הקריפטוגרפית Whirlpool על תוכן הקובץ המוצפן, ומתקבל ערך Hash באורך קבוע המייצג באופן ייחודי את הנתונים. פונקציית Whirlpool מבוססת על עקרונות של ערבוב והפצה של ביטים, כך שכל שינוי קטן בתוכן הקובץ יגרום לשינוי משמעותי בערך ה-Hash המתקבל.

החתימה הדיגיטלית משמשת כטביעת אצבע של הקובץ המוצפן, ומאפשרת בשלב מאוחר יותר לוודא שהקובץ לא שונה מאז יצירת החתימה. שלב זה אינו מצפין מידע נוסף, אלא מוסיף שכבת אבטחה המשלימה את תהליך ההצפנה, בכך שהוא מאפשר אימות שלמות הנתונים לפני תהליך הפענוח והשימוש בקובץ.

אלגוריתם Whirlpool

אלגוריתם זה הוא אלגוריתם Hash קריפטוגרפי, שמטרתו להפיק ערך תקציר (Hash) באורך קבוע של 512 ביט מתוך קלט באורך משתנה. בשלב הראשון, תוכן הקובץ המוצפן עובר תהליך הכנה הכולל השלמה (Padding) וחלוקה לבלוקים בגודל קבוע של 512 ביט. כל בלוק מעובד באופן סדרתי, כך שהתוצאה של עיבוד בלוק אחד משפיעה על עיבוד הבלוק הבא.

ליבת האלגוריתם מבוססת על פונקציית דחיסה קריפטוגרפית הפועלת במספר סבבים עוקבים. בכל סבב מבוצעות פעולות של החלפה לא ליניארית באמצעות טבלאות קבועות (S-boxes), ערבוב ליניארי של בתיים ופעולות XOR בין נתוני הבלוק לערך הביניים המצטבר. פעולות אלו נועדו ליצור פיזור ובלבול של המידע, כך שכל שינוי קטן בקלט גורם לשינוי משמעותי בערך ה-Hash המתקבל. בסיום עיבוד כל הבלוקים מתקבל ערך Hash סופי, המשמש כטביעת אצבע ייחודית של הקובץ, ואינו מאפשר שחזור של המידע המקורי מתוכו.

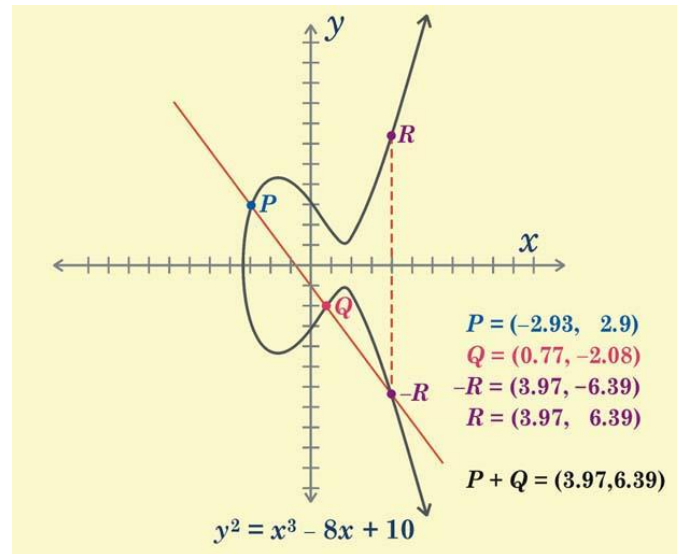
שלב 5 - הצפנת מפתח ההצפנה באמצעות ECC

לאחר הצפנת תוכן הקובץ באמצעות מפתח סימטרי, נדרש להגן על מפתח זה מפני חשיפה לגורמים לא מורשים. לצורך כך נעשה שימוש בהצפנה אסימטרית מבוססת ECC, אשר פועלת באמצעות זוג מפתחות ציבורי ופרטי. בשלב זה, המפתח הסימטרי המשמש להצפנת הקובץ מוצפן באמצעות המפתח הציבורי של המשתמש המורשה, כך שרק בעל המפתח הפרטי התואם יוכל לפענח אותו. תהליך זה מבטיח שהמפתח הסימטרי אינו נשמר או מועבר בצורה גלויה, ואף גורם בעל גישה לשרת אינו מסוגל לשחזרו.

אלגוריתם ECC

אלגוריתם זה הוא אלגוריתם הצפנה אסימטרי המבוסס על תכונות מתמטיות של **עקומים אליפטיים** מעל שדות סופיים. האלגוריתם עושה שימוש בזוג מפתחות, מפתח ציבורי ומפתח פרטי, כאשר המפתח הציבורי נגזר מתמטית מהמפתח הפרטי באמצעות פעולת כפל נקודה על העקום. תהליך זה יוצר קשר חד-כיווני, שבו חישוב המפתח הציבורי מהמפתח הפרטי הוא פשוט, אך שחזור המפתח הפרטי מתוך המפתח הציבורי נחשב לבלתי ישים חישובית. במהלך פעולת ההצפנה, נעשה שימוש במפתח הציבורי כדי להצפין מידע או מפתח אחר, כך שרק בעל המפתח הפרטי התואם יוכל לבצע את פעולת הפענוח. הביטחון של ECC נשען על הקושי שבפתרון בעיית הלוגריתם הדיסקרטי בעקומים אליפטיים, אשר מקשה על תוקף להסיק את המפתח הפרטי גם כאשר המפתח הציבורי ידוע. יתרונו

המרכזי של ECC הוא ביכולת לספק רמת אבטחה גבוהה באמצעות מפתחות קצרים יחסית, דבר המאפשר יעילות חישובית ושילוב נוח במערכות מודרניות לאבטחת מידע וניהול מפתחות. האיור מציג את אופן הפעולה של אלגוריתם ECC באמצעות חיבור נקודות על עקום אליפטי. נקודת



הבסיס P היא נקודה קבועה וידועה מראש, המשמשת כנקודת המוצא לכל החישובים הקריפטוגרפיים במערכת. נקודה Q מייצגת נקודה נוספת על העקום, אשר לרוב מתקבלת כתוצאה מכפל נקודת הבסיס במפתח פרטי סודי, ולכן משמשת כמפתח הציבורי. לצורך חיבור נקודות, מועבר ישר דרך הנקודות P ו-Q, והנקודה השלישית שבה הישר חותך את העקום מסומנת כ-R. נקודה זו משוקפת ביחס לציר ה-x ומתקבלת הנקודה R, שהיא תוצאת החיבור $P + Q$. פעולת חיבור זו מהווה את הבסיס לכל נקודות בעקום אליפטי, אשר משמש ליצירת מפתחות והצפנה ב-ECC. המבנה הגיאומטרי המוצג באיור מדגים כיצד פעולות אלו פשוטות לביצוע בכיוון אחד, אך קשות לשחזור לאחר ללא ידיעת המפתח הפרטי.

שלב 6 - חישוב מפתח משותף בין השולח לנמען באמצעות ECC

בשלב זה מתבצע חישוב של **מפתח משותף (Shared Secret)** בין השולח לנמען, המבוסס על מנגנון חילופי מפתחות באמצעות ECC. כל אחד מהצדדים עושה שימוש במפתח הפרטי שלו ובמפתח הציבורי של הצד השני, ומבצע כפל נקודה על העקום האליפטי. למרות שכל צד משתמש בערכים שונים, תוצאת החישוב המתקבלת זהה עבור שניהם ומייצרת מפתח משותף שאינו נשלח בפועל דרך הרשת. מפתח זה משמש כבסיס קריפטוגרפי מאובטח להמשך התהליך, ומבטיח כי רק השולח והנמען יוכלו לגזור את אותו ערך, גם אם המידע המועבר ביניהם נחשף לגורם חיצוני. שלב זה מאפשר הקמת סוד משותף ללא העברת מפתח גלוי ומהווה מרכיב מרכזי באבטחת תהליך שיתוף הקבצים.

שלב 7 - הצפנת המפתח הסימטרי באמצעות המפתח המשותף

נעשה שימוש במפתח המשותף לצורך הצפנה נוספת של המפתח הסימטרי המשמש להצפנת הקובץ. המפתח המשותף משמש כבסיס להצפנה סימטרית קלה ומהירה, כך שהמפתח הסימטרי של הקובץ אינו נשמר או מועבר בצורה גלויה בשום שלב. תהליך זה יוצר שכבת הגנה נוספת, שבה רק גורמים שהצליחו לגזור את אותו מפתח משותף יוכלו לפענח את מפתח ההצפנה של הקובץ. שילוב זה מבטיח שגם במקרה של גישה לנתונים המאוחסנים בשרת, לא ניתן יהיה להשתמש במפתח הסימטרי ללא ביצוע מוצלח של שלב חילופי המפתחות הקריפטוגרפים.

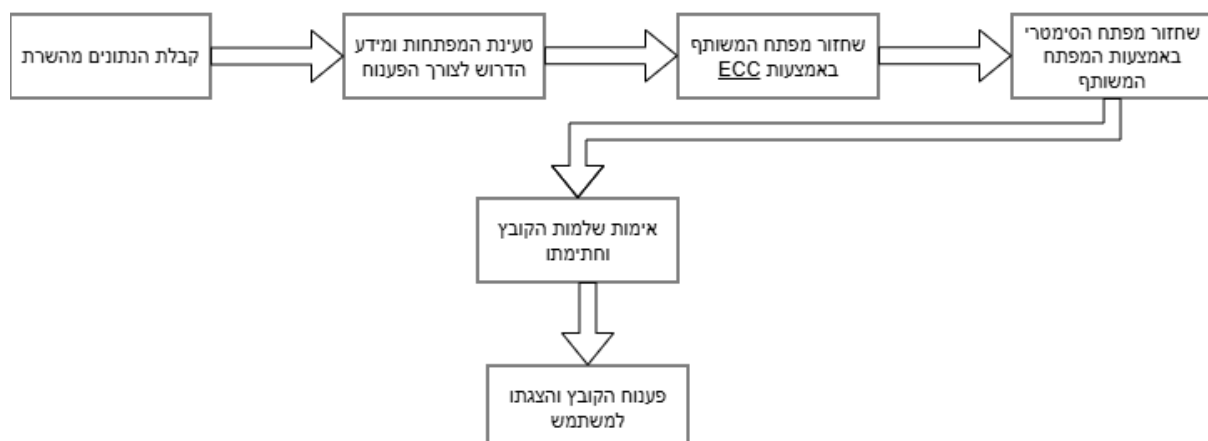
שלב 8 - שליחת הנתונים המוצפנים לשרת ושמירתם

בשלב הסופי בתהליך, הנתונים המוצפנים נשלחים לשרת לצורך אחסון. נתונים אלו כוללים את תוכן הקובץ לאחר הצפנה סימטרית, את המפתח הסימטרי כשהוא מוצפן באמצעות המפתח המשותף, וכן את ערך ה-Hash שנוצר לצורך בדיקת שלמות. השרת משמש בשלב זה כגורם אחסון בלבד, ואינו מחזיק ביכולת לפענח את הקובץ או את מפתח ההצפנה, מאחר שאין ברשותו את המפתחות הפרטיים הנדרשים לכך. הפרדה זו מבטיחה שגם במקרה של גישה לא מורשית לשרת או לנתוניו, תוכן הקבצים נשאר חסוי. שמירת הנתונים בצורה זו מאפשרת גישה מאובטחת לקבצים על ידי משתמשים מורשים בלבד, תוך שמירה על עקרון של מינימום אמון בשרת.

תהליך פענוח הקובץ

לאחר השלמת תהליך ההצפנה והעברת הקובץ המאובטח, נדרש תהליך פענוח מבוקר בצד הנמען על מנת לשחזר את הקובץ המקורי בצורה בטוחה. תהליך זה מבוסס על אימות שלמות ומקור המידע, שחזור מפתח ההצפנה הסימטרי והפעלת פעולת הפענוח עצמה. כל שלב בפענוח נשען על נתונים שנוצרו בשלב ההצפנה, ומטרתו לוודא שהקובץ שהתקבל לא שונה בדרך, אכן נשלח מגורם מורשה, וניתן לפענוח רק על ידי הנמען המתאים. כעת יפורטו שלבי הפענוח שמבצעת המערכת בצד המקבל, לפי סדר ביצועם.

תהליך פענוח הקובץ



קבלת הנתונים מהשרת

בשלב הראשון של תהליך הפענוח, המערכת בצד הנמען מקבלת מהשרת את חבילת הנתונים המלאה שנשלחה בתהליך ההצפנה. חבילה זו כוללת את הקובץ המוצפן, מפתח ההצפנה הסימטרי כשהוא מוגן באמצעות ECC, טביעת האצבע הדיגיטלית של הקובץ, החתימה הדיגיטלית והמידע הנלווה. בשלב זה המערכת עדיין אינה מפענחת את הקובץ, אלא רק טוענת את כל הנתונים הדרושים להמשך תהליך האימות והפענוח. שלב זה מהווה את נקודת הכניסה של המידע המוצפן אל תהליך השחזור המאובטח.

טעינת מפתחות והכנת תהליך האימות

בשלב זה המערכת טוענת את זוג המפתחות הקריפטוגרפים של הנמען, הכוללים מפתח פרטי ומפתח ציבורי, וכן את המפתח הציבורי של השולח הנדרש לאימות החתימה הדיגיטלית. שלב זה נועד להכין את כל החומרים הקריפטוגרפים הדרושים לביצוע פעולות האימות ושחזור מפתח ההצפנה בהמשך. בשלב זה עדיין לא מתבצעת בדיקה או פענוח של הקובץ עצמו, אלא רק הכנה של סביבת העבודה הקריפטוגרפית לצורך המשך התהליך בצורה מאובטחת.

שחזור המפתח המשותף באמצעות ECC

בשלב זה המערכת משתמשת במפתח הפרטי של הנמען ובמפתח הציבורי של השולח כדי לבצע את תהליך ה-ECC ולחשב את המפתח המשותף. חישוב זה מתבצע באופן מקומי בצד הנמען ואינו מצריך העברת מפתח כלשהו דרך הרשת. המפתח המשותף שמתקבל זהה למפתח שנוצר בצד השולח בשלב ההצפנה, והוא משמש כבסיס לשחזור מפתח ההצפנה הסימטרי. שלב זה מהווה את החוליה המרכזית בתהליך הפענוח, שכן הוא מאפשר גישה למפתח ההצפנה מבלי לחשוף אותו לגורמים אחרים.

שחזור מפתח ההצפנה הסימטרי

בשלב זה המערכת משתמשת במפתח המשותף שחושב באמצעות ECC כדי לשחזר את מפתח ההצפנה הסימטרי של Camellia. המפתח הסימטרי שהיה שמור בשרת בצורה מוגנת מפוענח כעת באופן מקומי בצד הנמען, מבלי שהמפתח נחשף במהלך ההעברה. עם סיום שלב זה, המערכת מחזיקה לראשונה במפתח ההצפנה המקורי הדרוש לפענוח הקובץ עצמו, אך עדיין אינה מפענחת את הקובץ עד לאימות שלמותו ומקורו.

אימות שלמות הקובץ והחתימה הדיגיטלית

בשלב זה המערכת מחשבת מחדש ערך Hash על הקובץ המוצפן שהתקבל, באמצעות אותו אלגוריתם Hash שבו נעשה שימוש בשלב ההצפנה. לאחר מכן מתבצעת בדיקה של החתימה הדיגיטלית באמצעות המפתח הציבורי של השולח, כדי לוודא שטביעת האצבע אכן נוצרה על ידו ולא שונתה בדרך. אם ערך ה-Hash המחושב תואם לערך החתום והחתימה מאומתת בהצלחה, ניתן להסיק שהקובץ שלם, לא עבר שינוי, ונשלח מגורם מורשה. רק במקרה שבו שלב זה מסתיים בהצלחה, המערכת ממשיכה לשלב הפענוח של הקובץ עצמו.

פענוח הקובץ והצגתו למשתמש

בשלב הסופי של תהליך הפענוח, ולאחר שהמערכת אימתה בהצלחה את שלמות הקובץ ואת מקורו, מתבצע פענוח הקובץ עצמו באמצעות אלגוריתם Camellia והמפתח הסימטרי ששוחזר בשלב הקודם. פעולת הפענוח מחזירה את הקובץ למצבו המקורי והקריא. לאחר מכן המערכת שומרת את הקובץ המפוענח במיקום שנבחר על ידי המשתמש או מציגה אותו בהתאם לצורך. שלב זה מסכם את תהליך האבטחה כולו ומבטיח שהקובץ נפתח רק לאחר שכל בדיקות האבטחה עברו בהצלחה, ללא חשיפה מוקדמת של מידע רגיש וללא תלות באמינות השרת.

8. תיאור פרוטוקולי תקשורת

המערכת עושה שימוש בפרוטוקולי תקשורת סטנדרטיים להעברת מידע בין צד הלקוח לבין צד השרת בצורה מאובטחת, אמינה ומבוקרת. פרוטוקולים אלו מאפשרים תקשורת רציפה בין רכיבי המערכת, תוך שמירה על סודיות ושלמות המידע המועבר.

• פרוטוקול HTTP / HTTPS

התקשורת בין הלקוח לשרת מתבצעת באמצעות פרוטוקול HTTP בגרסתו המאובטחת HTTPS. שימוש ב-HTTPS מבטיח כי כל הנתונים הנשלחים בין הצדדים מועברים בערוץ מוצפן, כך שגם במקרה של יירוט תעבורת הרשת, לא ניתן יהיה לקרוא או לשנות את המידע. בנוסף, הפרוטוקול מאפשר אימות של זהות השרת ומספק הגנה מפני שינוי נתונים במהלך ההעברה.

• TCP/IP

ברמה הבסיסית יותר, התקשורת בין הלקוח לשרת נשענת על פרוטוקול TCP/IP האחראי על העברת המידע ברשת בצורה אמינה ומסודרת. TCP/IP מבטיח הגעה מלאה של הנתונים, שמירה על סדר המנות, וזיהוי תקלות בהעברה. פרוטוקול זה מהווה את התשתית שעליה פועלים שאר פרוטוקולי התקשורת במערכת.

• REST API

בנוסף לתקשורת בין הלקוח לשרת, המערכת עושה שימוש ב-REST API חיצוני לצורך קבלת

מידע אבטחתי תומך החלטה. בעת יצירת חשבון משתמש או שינוי סיסמה, השרת שולח בקשת REST לשירות Pwned Passwords של Have I Been Pwned, באמצעות נקודת הקצה הייעודית לבדיקה אנונימית של סיסמאות. הבקשה מתבצעת באמצעות קריאה ל- endpoint `GET /range/{hashPrefix}` בפורמט `{hashPrefix}` מייצג חלק מערך ה-Hash של הסיסמה, בהתאם למנגנון בדיקה מבוסס k-anonymity בתגובה מוחזרת רשימת ערכי Hash חלקיים המאפשרת לשרת לקבוע האם הסיסמה מופיעה במאגרי דליפות ידועים, מבלי להעביר את הסיסמה עצמה או את ערך ה-Hash המלא. תוצאת הבדיקה מאפשרת למערכת להתריע למשתמש, לחסום שימוש בסיסמאות שנפרצו, ולהעלות את רמת האבטחה של חשבונות המשתמשים. שימוש ב- endpoint זה מאפשר שילוב מידע אבטחתי חיצוני בתהליך קבלת ההחלטות של המערכת, תוך שמירה על סודיות המידע הרגיש והימנעות מחשיפת פרטי הזדהות.

• פורמט JSON

JSON הוא פורמט נתונים טקסטואלי מובנה שמקובל מאוד ב-REST API והוא משמש לייצוג הבקשות והתגובות בין מערכות בצורה אחידה. בהקשר של REST API, הלקוח שולח בקשה שכוללת נתונים בפורמט JSON, והשרת מחזיר תשובה בפורמט JSON שמכילה שדות ברורים כמו סטטוס, פרטים ותוצאות. היתרון כאן הוא שהפורמט פשוט לפירוק ולעיבוד, כך שקל מאוד לוודא מה התקבל ומה צריך לעשות עם זה בצד המקבל.

9. פיתוחים עתידיים

- בשלב עתידי ניתן להרחיב את מערכת במגוון יכולות מתקדמות שיחזקו את חוויית השימוש ואת רמת האבטחה. אפשרות מרכזית היא הוספת צפייה ישירה בקבצים דרך הדפדפן, כך שמשתמשים יוכלו לצפות במסמכים ובתמונות ללא הורדה. פיתוח נוסף עשוי לכלול תמיכה בשיתוף קבצים קבוצתי, המאפשר להעניק הרשאות גישה למספר משתמשים בארגון בצורה פשוטה ומובנית. בנוסף, ניתן לשלב מנגנון מחיקה אוטומטית לקבצים לאחר זמן קצוב או מספר הורדות מוגדר, כדי להקטין את הסיכון לדליפות מידע.
- אפשרות אחרת היא הטמעת מערכת ניהול ארגונית, שתאפשר למנהלים לצפות ברשימות קבצים, לקבוע מדיניות הצפנה, ולהגדיר רמות הרשאה שונות לכל משתמש. בעתיד ניתן יהיה להוסיף גם דוחות פעילות, שיציגו היסטוריית גישה לקבצים, ניסיונות התחברות, ופעולות שבוצעו במערכת. פיתוחים נוספים עשויים לכלול תמיכה בהעלאת קבצים גדולים במיוחד באמצעות פרוטוקולי חלוקה למקטעים, או שילוב זיהוי דו שלבי להגברת אבטחת הגישה למערכת. כל רעיון כזה נועד לשפר את המערכת מבלי לשנות את ליבת העבודה הקיימת.

10. תיאור טכנולוגיה הנדסה

- שפת תכנות:
 - **Java 21** שפת תכנות עילית OOP המאפשרת לכתוב קוד גם בצד השרת וגם בצד הלקוח.
- מסגרות עבודה (Frameworks):
 - **Spring Boot 3.5.9** היא מסגרת עבודה לפיתוח יישומי Web בצד השרת, המשמשת לבניית הלוגיקה המרכזית של המערכת וניהול תהליכי העבודה בה. המסגרת מאפשרת קבלת בקשות מהמשתמשים, עיבודן בהתאם לכללי המערכת והחזרת תוצאות מתאימות, תוך מתן תשתית מובנית לניהול בקשות וזרימת נתונים בצורה מסודרת. במודל שלוש שכבות מסגרת זו פועלת בשכבת הלוגיקה (Business Logic Layer) ומשמשת כמתווך בין שכבת התצוגה לשכבת הנתונים. היא אחראית על מימוש כללי המערכת, ניהול הרשאות ואינטגרציה עם מסד הנתונים, תוך שמירה על הפרדה ברורה בין הצגת המידע לבין אופן הטיפול והאחסון שלו. הפרדה זו תורמת לאבטחת המערכת, לתחזוקה נוחה ולהרחבה עתידית.
 - **Vaadin 24.9.5** היא ספריית עזר לבניית ממשקי משתמש ליישומי Web, אשר פועלת בצמוד ל-Spring Boot בצד השרת. ייעודה של Vaadin הוא לאפשר יצירת ממשק משתמש גרפי מלא באמצעות קוד Java בלבד, ללא צורך בכתיבה ישירה של HTML, CSS, JavaScript. במסגרת עבודה עם Spring Boot, Vaadin משתלבת כשכבת התצוגה של המערכת, ומאפשרת להציג טפסים, כפתורים, טבלאות ומסכים שונים, ולקשר אותם ישירות ללוגיקה העסקית שבשרת. שילוב זה מאפשר זרימת מידע ישירה ומאובטחת בין הממשק למערכת, תוך שמירה על הפרדה בין שכבת התצוגה לשכבת הלוגיקה.
- סביבת פיתוח (IDE):
 - **VSCode 1.108** היא סביבת פיתוח קלה וגמישה המשמשת לכתיבה, ניהול ודיבאג של קוד בפרויקט תוכנה. היא תומכת בעבודה עם פרויקטי Java וכלי בנייה כגון Maven, ומאפשרת פיתוח יעיל ונוח באמצעות הרחבות וכלים מובנים.
- ספריות עזר: אין ספריות כרגע לשימוש בפרויקט.

11. מסד נתונים

בפרויקט זה נעשה שימוש במסד נתונים **MongoDB** מסוג **NoSQL**, המתאים במיוחד למערכות שבהן מבני הנתונים אינם קבועים מראש ועלולים להשתנות בהתאם לצרכים תפעוליים ואבטחתיים. בניגוד למסדי נתונים רלציוניים, מסד נתונים זה מאפשר אחסון מידע במבנה מסמכי גמיש, שבו כל

רשומה יכולה להכיל שדות שונים ומבנים מקוננים, ללא תלות בסכמה קשיחה. תכונה זו מתאימה לאופי המערכת, שבה נשמרים סוגים שונים של נתונים כגון פרטי משתמשים, מפתחות ציבוריים, מידע נלווה על קבצים והרשאות גישה, אשר אינם בהכרח אחידים במבנם.

השימוש במסד נתונים NoSQL מאפשר שמירה נוחה של מידע הקשור לקבצים מוצפנים, מבלי לאחסן את תוכן הקבצים עצמם במצב קריא. המידע המאוחסן כולל נתונים תיאוריים בלבד, כגון מזהים, בעלות על קבצים, תאריכי פעולה ומידע אבטחתי נלווה, דבר המאפשר הפרדה ברורה בין תוכן רגיש לבין נתוני ניהול. בנוסף, מבנה מסמכי מאפשר קישור טבעי בין משתמשים, קבצים והרשאות, באמצעות מזהים ושדות מקוננים, ללא צורך בקשרים רלציוניים מורכבים.

יתרון נוסף של שימוש ב-NoSQL בפרויקט הוא היכולת להרחיב את מבנה הנתונים בעתיד בקלות, למשל הוספת שדות אבטחה חדשים, נתוני בקרה או מידע תפעולי נוסף, מבלי לפגוע בנתונים קיימים. גישה זו תורמת לתחזוקה נוחה של המערכת, להתאמה לדרישות משתנות ולפיתוח עתידי, תוך שמירה על ביצועים ויציבות. הבחירה במסד נתונים NoSQL תומכת במטרות האבטחתיות של הפרויקט ומאפשרת ניהול מידע גמיש, מאובטח ומבוקר.

להלן פירוט האוספים המרכזיים במסד הנתונים של המערכת והאופן שבו כל אוסף משמש לניהול המידע ולתמיכה בתהליכי העבודה והאבטחה של הפרויקט.

• אוסף פרטי משתמש

אוסף זה משמש לשמירת נתוני הזיהוי והאימות של המשתמשים במערכת, כולל פרטי התחברות, מפתח ציבורי והקשר לקבצים השייכים להם. המידע באוסף זה מאפשר ניהול הרשאות, זיהוי משתמשים ושיוך פעולות וקבצים לבעליהם בצורה מבוקרת ומאובטחת.

• אוסף קבצים מוצפנים

אוסף זה משמש לשמירת מידע נלווה לקבצים המאוחסנים במערכת, כאשר תוכן הקבצים נשמר במצב מוצפן בלבד. המידע באוסף זה מאפשר ניהול קבצים, בדיקת שלמותם, שליטה בגישה אליהם ושחזור מאובטח על ידי משתמשים מורשים בלבד.

להלן מידע מפורט אשר יוצג במסד הנתונים:

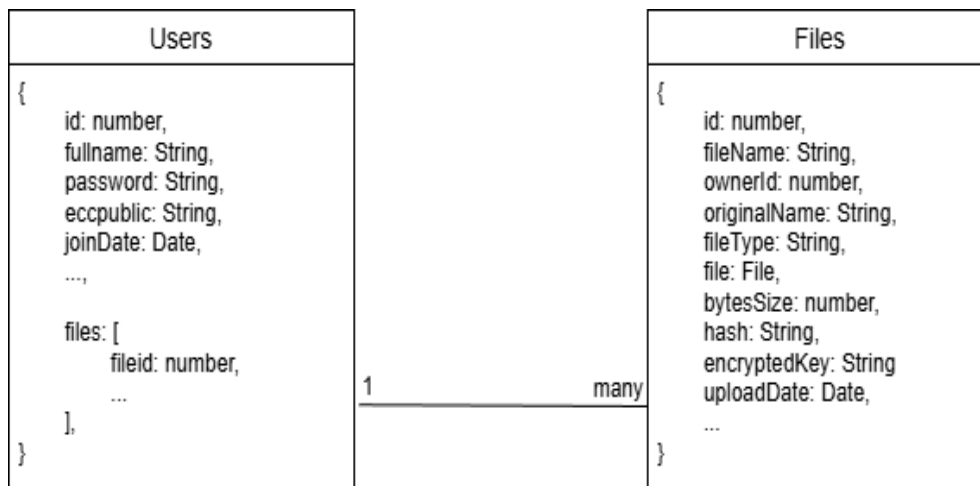
מידע של פרטי משתמש:

- שם המשמש
- שם מלא
- סיסמא
- מפתח ECC ציבורי
- קבצים שמורים במערכת

מידע של קבצים מוצפנים:

- מספר מזהה
- שם המשתמש של הבעלים
- שם הקובץ המקורי
- סוג הקובץ
- קובץ מוצפן
- תאריך העלאה
- גודל בטים
- ערך ה-Hash של הקובץ המוצפן
- מפתח סימטרי מוצפן

להלן תרשים ERD של האוספים המציג את המידע:



ניתן לראות בתרשים שני אוספים, האוסף השמאלי הוא אוסף המשתמשים (Users) המציג את המידע לכל משתמש כגון: שם מלא, סיסמא, תאריך הצטרפות, ואוסף שנמצא בצד ימין של התרשים שהוא אוסף קבצים מוצפנים (Files) ששם מאוחסן כל המידע על הקובץ כגון: שם הקובץ, סוג הקובץ, וכדומה, והקובץ עצמו, הקשר בין האוספים הללו הוא N:1, כאשר יש משתמש אחד שמקושרים אליו כמה קבצים.

הערה חשובה: ייתכן שבמהלך בניית המערכת יתווסף מידע נוסף או ישתנה מעט, לכן התרשים והמידע המופיעים בסעיף זה מהווים תצוגה כללית ולא דבר וודאי.

12. פרטים פורמליים

לוח זמנים:

לסיים עד תאריך	שלבי עבודה
1.12.2025	בחירת פרויקט, חקירה ולמידה לעומק של נושאי הפרויקט
11.12.2025	כתיבה והגשת הצעת הפרויקט לאישור משרד החינוך
13.1.2026	מימוש הקוד של האלגוריתם המרכזי, ביצוע בדיקות ושיפורים
3.2.2026	בניית צד שרת
17.2.2026	בניית מסד הנתונים ושילובו
3.3.2026	בניית צד לקוח
24.3.2026	כתיבת ספר הפרויקט
7.4.2026	הגשת הפרויקט כולו (ספר + קוד) להגנה וקבלת ציון מגן

חתימת מנחה הפרויקט:



חתימת הסטודנט:

חתימת רכז המגמה :