**ממ"ן 16 – חלק עיוני**



מגישים:

טל טריביטש - 209396019

יובל בן אליהו - 207386970

תוכן עניינים:

[חלק 1: מסמך תכנון מפורט - מערכת E2EE להעברת הודעות 3](#_Toc186806844)

[הקדמה תאורטית 3](#_Toc186806845)

[מטרות הפרויקט 3](#_Toc186806846)

[תיאור המערכת 4](#_Toc186806847)

[ארכיטקטורת המערכת 4](#_Toc186806848)

[לקוח (Client): 4](#_Toc186806849)

[שרת (Server): 4](#_Toc186806850)

[תכונות עיקריות בפרויקט 5](#_Toc186806851)

[הצפנות 5](#_Toc186806852)

[אימות זהות (Authentication) 5](#_Toc186806853)

[הגנה מפני התקפות MITM 5](#_Toc186806854)

[עבודה במקביל (Multi-Threading) 6](#_Toc186806855)

[הסבר הפרוטוקול 6](#_Toc186806856)

[תהליך רישום 6](#_Toc186806857)

[שליחת הודעה: 7](#_Toc186806858)

[פענוח הודעה: 8](#_Toc186806859)

[סיכום 8](#_Toc186806860)

[חלק 3: 9](#_Toc186806861)

# חלק 1: מסמך תכנון מפורט - מערכת E2EE להעברת הודעות

## הקדמה תאורטית

מערכת הצפנה מקצה-לקצה (End-to-End Encryption, E2EE) נועדה להבטיח שכל התקשורת בין שני לקוחות מוצפנת לחלוטין ואינה נחשפת לגורם שלישי, כולל השרת דרכו מועברות ההודעות. מטרת הפרויקט היא לפתח מערכת להעברת הודעות מוצפנות שתגן מפני התקפות Man-in-the-Middle (MITM), תוך הבטחת Confidentiality (סודיות), Integrity (שלמות המידע), ו-Authentication (אימות).

## מטרות הפרויקט

* פיתוח מערכת הצפנה מבוססת פרוטוקול AES-256 במצב CBC.
* שילוב מנגנון יצירת מפתחות (KDF) להפקת מפתחות הצפנה חזקים.
* יישום מנגנון אימות לקוח המגן מפני התחזות (Spoofing) בעזרת ערוץ מאובטח.
* הגנה מפני התקפות MITM באמצעות תעודות דיגיטליות.
* תמיכה בריבוי תהליכים (Multi-Threading) לשיפור ביצועי המערכת.

## תיאור המערכת

### ארכיטקטורת המערכת

#### לקוח (Client):

* **רישום ראשוני**: ביצוע תהליך רישום ראשוני מול השרת לצורך זיהוי והתחברות למערכת.
* **אימות זהות**: אימות זהות הלקוח מול השרת כחלק מתהליך הגנה על המידע.
* **ניהול מפתחות**: החלפה בטוחה של מפתחות הצפנה עם לקוח אחר לצורך יצירת ערוץ תקשורת מאובטח.
* **שליחת הודעות**: שליחת הודעות מוצפנות ללקוחות אחרים במערכת.

#### שרת (Server):

* **אחסון מפתחות ציבוריים**: שמירת המפתחות הציבוריים של הלקוחות בבסיס נתונים (DB) לצורך תמיכה בתהליכי הצפנה.
* **שליחת OTP**: שליחת קודים חד-פעמיים (One-Time Password) עבור רישום ראשוני של לקוחות באמצעות ערוץ מאובטח.
* **ניהול הודעות מוצפנות**: טיפול בהעברת הודעות מוצפנות בלבד בין לקוחות לשמירה על פרטיות.
* **שמירת הודעות Offline**: אחסון הודעות בבסיס הנתונים (DB) עבור לקוחות שאינם זמינים, לצורך שליחה במועד מאוחר יותר.

### 

### תכונות עיקריות בפרויקט

#### הצפנות

* **AES-256 במצב CBC** 
  + אלגוריתם הצפנה סימטרי מהיר ויעיל.
  + שימוש ב-IV אקראי לכל הודעה להבטחת אקראיות.
  + משמש להצפנת תוכן ההודעות.
* **RSA**
  + הצפנה א-סימטרית להעברת מפתחות סימטריים בצורה מאובטחת.
  + חתימה דיגיטלית לאימות זהות המשתמשים.
* **HMAC**
  + משמש לאימות נתונים ולהבטחת שלמות המידע.
  + מבוסס על פונקציית גיבוב SHA-256

#### אימות זהות (Authentication)

* שימוש בקוד חד-פעמי (OTP) בשלב ההרשמה (אשר יש לו תוקף של 60 שניות).
* יצירת חתימת HMAC לאימות שהמשתמש קיבל את הקוד מהשרת.
* העברת פרטי המשתמש (כולל חתימה) בצורה מוצפנת למניעת גישה לא מורשית.

#### הגנה מפני התקפות MITM

* שימוש בתעודות דיגיטליות (Certificates) על בסיס תשתית מפתחות ציבוריים (PKI).
* הצפנת פרטי ההרשמה באמצעות המפתח הציבורי של השרת.
* אימות התעודות על ידי השרת לפני העברת הודעות.

### 

### עבודה במקביל (Multi-Threading)

התמיכה בריבוי תהליכים מאפשרת לשרת לטפל במספר משתמשים בו-זמנית, לשיפור הביצועים והאמינות.

## הסבר הפרוטוקול

במערכת שלנו יש שרת ולקוחות. הלקוחות מתקשרים זה עם זה בשיטת הצפנה E2E כך שהשרת אינו יכול לפענח את תוכן ההודעות המועברות ביניהם. להלן פירוט הפרוטוקול:

### תהליך רישום

1. **בקשת רישום:**
   1. כל לקוח (טל/יובל) שולח בקשת הרשמה לשרת.
   2. השרת שולח חזרה קוד בן 6 ספרות לערוץ מאובטח (תקף ל-60 שניות).
2. **יצירת מפתחות RSA על ידי הלקוח:**
   1. הלקוח מייצר זוג מפתחות RSA (מפתח ציבורי ומפתח פרטי).
   2. המפתח הציבורי ישמש להצפנה ולאימות, והמפתח הפרטי לחתימה דיגיטלית ולפענוח.
3. **חתימה עם HMAC:**
   1. הלקוח יוצר חתימת HMAC על בסיס הקוד שקיבל מהשרת והמפתח הציבורי שלו.
   2. החתימה מוכיחה שהלקוח אכן קיבל את הקוד מהשרת.
4. **שליחת פרטי הרשמה לשרת:**
   1. הלקוח שולח לשרת בקשה מוצפנת המכילה:
      1. המפתח הציבורי שלו.
      2. החתימה (HMAC).
      3. פרטי משתמש (שם פרטי, שם משפחה, מספר טלפון וסיסמה חזקה).
   2. ההצפנה מתבצעת באמצעות המפתח הציבורי של השרת.
5. **אימות ושמירת הנתונים:**
   1. השרת מפענח את הבקשה, מאמת את הנתונים, ושומר את המפתח הציבורי של הלקוח יחד עם פרטים נוספים:
      1. מספר טלפון (המזהה הייחודי).
      2. סיסמה (מאוחסנת עם KDF ו-SALT אקראי).

### שליחת הודעה:

1. **בקשת מפתח ציבורי:**
   1. טל מבקש מהשרת את המפתח הציבורי של יובל לפי מספר הטלפון שלו.
   2. השרת חותם על המפתח הציבורי של יובל באמצעות המפתח הפרטי שלו.
   3. טל מקבל את המפתח הציבורי של יובל יחד עם החתימה הדיגיטלית של השרת.

**למה זה משמש?**

מבטיח שהמפתח הציבורי שייך ליובל ולא נפרץ או הוחלף על ידי גורם זדוני (מניעת MITM).

**איך זה עובד?**

* השרת יוצר חתימה דיגיטלית על המפתח הציבורי של יובל באמצעות המפתח הפרטי שלו.
* טל משתמש במפתח הציבורי של השרת כדי לאמת את החתימה.

1. **אימות המפתח הציבורי:**
   1. טל מאמת את החתימה באמצעות המפתח הציבורי של השרת והמפתח הציבורי של יובל, ובכך מוודא שהמפתח לא שונה ע"י צד שלישי.
2. **יצירת מפתח סימטרי:**
   1. טל ויובל מייצרים סוד משותף באמצעות ECDH (Elliptic Curve Diffie-Hellman) על בסיס המפתחות הפרטיים והציבוריים שלהם.

**למה זה משמש?**

ליצירת מפתח סימטרי ייחודי שמאפשר הצפנה מאובטחת לתקשורת בין טל ליובל.

**איך זה עובד?**

כל צד משתמש במפתח הציבורי של הצד השני ובמפתח הפרטי שלו כדי ליצור את אותו סוד משותף.

1. **הצפנת ההודעה:**
   1. טל מצפין את ההודעה באמצעות AES-256 במצב CBC.
   2. לכל הודעה נוצר IV אקראי להגברת האבטחה.
2. **שליחת ההודעה:**
   1. טל שולח ליובל את ההודעה המוצפנת, יחד עם ה-SALT וה-IV.

### פענוח הודעה:

1. **קבלת מפתח ציבורי:**
   1. יובל מבקש מהשרת את המפתח הציבורי של טל (חתום על ידי השרת כמו בתהליך שליחת ההודעה).

**למה זה משמש?**

מבטיח שהמפתח הציבורי של טל הוא אותנטי ולא זויף.

**איך זה עובד?**

יובל מאמת את החתימה על המפתח הציבורי של טל באמצעות המפתח הציבורי של השרת.

1. **יצירת מפתח סימטרי:**
   1. יובל יוצר סוד משותף עם המפתח הפרטי שלו, המפתח הציבורי של טל, וה-SALT.
2. **פענוח ההודעה:**
   1. יובל מפענח את ההודעה באמצעות המפתח הסימטרי וה-IV שנשלח.

## סיכום

המערכת המתוכננת מספקת פתרון מאובטח להעברת הודעות באמצעות שילוב הצפנה סימטרית וא-סימטרית. הפרוטוקול מבטיח עמידות בפני התקפות MITM ושמירה על פרטיות ושלמות המידע.

# חלק 3:

**האם ניתן להבטיח זמינות מלאה?**  
זמינות מלאה אינה ניתנת להבטחה, אך המערכת עושה מאמצים להבטיח זמינות גבוהה, בזכות המנגנונים הבאים:

1. **שמירת הודעות בשרת:**  
   כאשר אחד הקליינטים אינו זמין, השרת שומר את ההודעות המוצפנות עד שהלקוח הופך לזמין שוב. מנגנון זה מבטיח שלא יאבדו הודעות בשל חוסר זמינות זמני של קליינטים, ותורם לזמינות המידע.
2. **תמיכה בריבוי תהליכים (Multi-threading)**   
   התמיכה בריבוי תהליכים בשרת מאפשרת טיפול במספר רב של משתמשים במקביל, מה שמשפר את זמינות השירות גם במצבי עומס.
3. **אבטחת הנתונים:**  
   המערכת מטפלת רק בהודעות מוצפנות, מה שמפחית את הסיכוי לשיבוש נתונים או לפגיעה במידע, ועקב כך משפר את הזמינות.

**מגבלות אפשריות:**

* במקרה של כשל בשרת עצמו (למשל, קריסה או חוסר גישה לשרת), שמירת ההודעות לא תספיק כדי להבטיח זמינות מלאה.
* המערכת אינה מכילה מנגנוני יתירות (redundancy) או התאוששות מכשל (failover), ולכן קיימת תלות בתפקוד התקין של השרת.

**סיכום:**  
המערכת מתוכננת להתמודד עם חוסר זמינות זמני של לקוחות באמצעות שמירת הודעות בשרת, אך זמינות מלאה תלויה גם בתקינות השרת עצמו. על מנת לשפר זמינות ניתן לשלב מנגנוני יתירות או שרתים חלופיים (backup servers).