שם: רז דהן 208952622

ממן 16/תכנון פרוטוקול הצפנה מקצה לקצה

מבוא

במטלה עלינו לתכנן פרוטוקול מאובטח להעברת הודעות קצרות מקצה לקצה, עם תמיכה במשתמשים שאינם מחוברים ברגע שליחת ההודעה. הפרוטוקול יבטיח סודיות, שלמות ואמינות, ויהיה עמיד נגד מתקפות MITM בכל השלבים.

שיטת ההצפנה ואופן יצירת המפתחות

נשתמש בהצפנה סימטרית לצורך הצפנת תוכן ההודעות עצמן. נבחר את האלגוריתם AES-256-CBC, שהוא אחד האלגוריתמים החזקים והמאובטחים ביותר כיום להצפנה סימטרית. האלגוריתם פועל במצב CBC (Galois/Counter Mode), שמתאים לפרוטוקול זה שבו שלמות ההודעה מאומתת בנפרד, וכל הודעה מוצפנת בצורה שונה, גם אם התוכן זהה, על ידי IV (Initialization Vector).

תהליך הרישום הראשוני
כאשר משתמש חדש מתקין את האפליקציה ורוצה להירשם, הוא יוצר לעצמו זוג מפתחות א-סימטריים: מפתח פרטי שאותו הוא שומר לעצמו בסודיות מוחלטת במכשיר שלו, ומפתח ציבורי שאותו הוא שולח לשרת יחד עם מספר הטלפון שלו. השרת מקבל את המפתח הציבורי ושומר אותו במאגר נתונים, מקושר למספר הטלפון של המשתמש. לאחר מכן, השרת מייצר קוד אימות קצר (6 ספרות) ושולח אותו למשתמש בערוץ תקשורת מאובטח אחר. המשתמש מאמת את הקוד באפליקציה, ובכך מושלם תהליך הרישום והקישור בין מספר הטלפון למפתח הציבורי בשרת. איננו משתמשים בKDF, משום שהקוד משמש ליצירת רישום ראשוני בלבד של מספר הטלפון, ולא ליצירת מפתחות קריפטוגרפים לשימוש מתמשך.

יצירת והעברת מפתחות שיחה
כאשר משתמש רוצה להתחיל שיחה עם משתמש אחר, הוא יוצר מפתח שיחה סימטרי חדש באופן אקראי. מפתח זה ישמש להצפנה ופענוח של כל ההודעות באותה שיחה ספציפית. את מפתח השיחה הזה הוא מצפין באמצעות המפתח הציבורי של הנמען (אותו הוא מקבל מהשרת) ושולח אותו יחד עם ההודעה המוצפנת הראשונה. הנמען, בעזרת המפתח הפרטי שלו, יכול לפענח את מפתח השיחה ולהשתמש בו לפענוח ההודעה. מרגע זה, שני הצדדים משתמשים במפתח השיחה הזה להצפנה ופענוח של הודעות עתידיות בשיחה, עד לסיומה.

הצפנה ושליחת הודעה : כאשר משתמש רוצה לשלוח הודעה, המערכת מצפינה אותה באמצעות AES-256-CBC. בנוסף, המערכת מחשבת HMAC (Hash-based Message Authentication Code) על ההודעה המוצפנת בעזרת מפתח אימות שנגזר ממפתח השיחה ה-HMAC, משמש לווידוא שלמות ההודעה ומונע שינויים זדוניים בה. לאחר מכן, השולח שולח לנמען את ההודעה המוצפנת, את הHMAC, ואת מפתח השיחה המוצפן (רק בהודעה הראשונה).

קבלת ופענוח הודעה :כאשר הנמען מקבל את ההודעה, הוא קודם כל מפענח את מפתח השיחה (אם זו ההודעה הראשונה) בעזרת המפתח הפרטי שלו. לאחר מכן, הוא מפענח את ההודעה המוצפנת בעזרת מפתח השיחה. לבסוף, הוא מחשב HMAC באופן עצמאי על ההודעה המפוענחת ומשווה אותו ל HMAC-שהתקבל עם ההודעה. אם שני ה HMACs-זהים, זה מוכיח שההודעה לא שונתה במהלך השידור והיא אותנטית. אם האימות הצליח, הנמען יכול לשלוח אישור קבלה מוצפן בחזרה לשולח.

שימוש בHMAC: הוא קוד אימות הודעה המבוסס על פונקציית גיבוב קריפטוגרפית. הוא מבטיח את שלמות ההודעה. הוא מחושב על ידי השולח על ההודעה המוצפנת ומועבר יחד איתה לנמען. הנמען מחשב HMAC באופן עצמאי על ההודעה שקיבל ומשווה אותו ל HMAC-שהתקבל. אם הם זהים, סימן שההודעה לא שונתה.

איך אנחנו משיגים שלמות במובן הרחב (מקור ותוכן ההודעה):

אימות מקור: אימות המקור מתבצע בשני שלבים :

1. **רישום ראשוני :** במהלך הרישום, השרת מקשר את המפתח הציבורי של המשתמש למספר הטלפון שלו. זה מבטיח שהמפתח הציבורי אכן שייך למספר הטלפון הרשום.
2. **חתימה דיגיטלית:** בנוסף לHMAC, ניתן להשתמש בחתימה דיגיטלית של השולח על ההודעה. זה מספק רמת אימות גבוהה יותר של המקור, מכיוון שרק השולח בעל המפתח הפרטי המתאים יכול ליצור חתימה תקפה. חתימה דיגיטלית תתבצע על הגיבוב (Hash) של ההודעה המוצפנת.

אימות תוכן: הHMAC מבטיח את שלמות תוכן ההודעה. כל שינוי בהודעה לאחר יצירת ה HMAC יגרום לאי התאמה באימות בצד הנמען.

מבנה הנתונים של השרת (במימוש):

השרת משתמש בשני מבנים מרכזיים:

1. **טבלת משתמשים (self.users)**
מילון שבו:
   - **מפתח :**מספר טלפון של המשתמש (מחרוזת).
   - **ערך :**מפתח ציבורי של המשתמש (אובייקט מסוג RSA Public Key)
2. **טבלת הודעות (self.messages)**
מילון שבו:
   - **מפתח :**מספר טלפון של הנמען (מחרוזת).
   - **ערך :**רשימה של הודעות מוצפנות (עד שתי הודעות לכל משתמש).

מבנה שדות ההודעה:

encrypted_session_key (bytes): מפתח השיחה הסימטרי (AES) מוצפן באמצעות המפתח הציבורי של הנמען (RSA).

iv (bytes): וקטור אתחול עבור הצפנת AES במצב CBC.

encrypted_message (bytes): ההודעה המוצפנת עם מפתח השיחה (AES-256-CBC).

hmac (bytes) : קוד אימות הודעה (HMAC) שנוצר על בסיס ההודעה המוצפנת, באמצעות מפתח השיחה.

<u>סעיף ב'</u>

במימוש בפייתון (מצורפים הקבצים), הנחנו כמה הנחות:

לכל לקוח יעד, השרת שומר עד שתי הודעות בלבד. אם מגיעה הודעה שלישית, ההודעה הישנה ביותר נמחקת כדי לפנות מקום להודעה החדשה.

כל משתמש חייב להירשם מראש לשרת עם מספר טלפון ייחודי (המשמש כמפתח ראשי) ומפתח ציבורי. אין אפשרות לרשום את אותו מספר טלפון פעמיים. ניסיון כזה יוביל לשגיאה.

השרת תומך בריבוי לקוחות (Multithreading) באמצעות פתיחת שרשור (Thread) נפרד לכל חיבור נכנס. השרת אינו תומך בניהול מורכב כמו שמירת היסטוריית הודעות מעבר לשתי ההודעות האחרונות. המערכת אינה תומכת בהודעות קבוצתיות או ניהול קבוצות.

<u>סעיף ג'</u>

זמינות (Availability) במערכת היא היכולת להבטיח שהמערכת והשירותים שלה יהיו נגישים ומשמשים את המשתמשים בזמן הדרוש להם, גם בתנאי עומס או תקלה.

במימוש הנוכחי בפייתון, **לא ניתן להבטיח זמינות מלאה** באופן מוחלט בגלל מספר מגבלות:

1. **תלות בתשתית הרשת והשרת**

   השרת תלוי במחשב שמריץ אותו, ואם יש כשל חומרה או בעיות רשת (כמו ניתוק אינטרנט או עומס ברשת), המערכת לא תהיה זמינה

2. **ריבוי חיבורים**

   המימוש הנוכחי תומך בריבוי חיבורים באמצעות **Threading,** אך מספר השרשורים המקסימלי תלוי במגבלות מערכת ההפעלה. בעומס גבוה, השרת עלול לקרוס או להפוך לאיטי.

3. **ניהול הודעות מוגבל**

   השרת שומר עד שתי הודעות לכל לקוח. אם הלקוח לא מתחבר בזמן ומגיעות הודעות נוספות, ההודעות הישנות נמחקות, מה שיכול להוביל לאובדן מידע עבור המשתמש.

4. **עדכון מפתחות ואימות**

   אם מפתח ציבורי של משתמש משתנה (למשל, בעקבות אובדן מפתח פרטי), יש לבצע רישום מחדש. אם לא מתבצע עדכון כזה, הלקוח לא יוכל לקבל הודעות.
   פתרון אפשרי: שימוש ב **Certificate Authorities (CA)** או מנגנון הפצת מפתחות דינמי כדי להבטיח זמינות תקשורת מאובטחת בכל עת.