פרוטוקול:  
  
שלב 1: רישום לקוחות ואימות ראשוני

1.1 יצירת מפתחות בצד הלקוח

- כל לקוח יוצר זוג מפתחות RSA:

- מפתח פרטי וציבורי נשמרים מקומית.

- מפתח ציבורי, שישלח לשרת, מפתח פרטי, בעזרתו יפענח ויחתום.

1.2 בקשת הרשמה

- הלקוח שולח בקשת הרשמה לשרת יחד עם מספר הטלפון שלו.

1.3 שליחת OTP

- השרת שולח ללקוח קוד חד-פעמי (OTP) בן 6 ספרות דרך ערוץ בטוח (SMS).

- ה-OTP תקף ל-5 דקות בלבד.

1.4 יצירת Salt משותף

- השרת והלקוח יוצרים Salt זהה באמצעות HMAC:

- 

1.5 יצירת מפתח קריפטוגרפי חד פעמי ( )

- הלקוח והשרת יוצרים את אותו (  ) באמצעות - KDF :

1.6 שליחת מפתח ציבורי ואימות

- הלקוח:

1. מחשב חתימה דיגיטלית על המפתח הציבורי שלו עם  באמצעות HMAC.

2. שולח לשרת את:

- *Public Key*

- Signature

- השרת:

1. מחשב את החתימה *Public Key* של הלקוח באופן עצמאי.

2. משווה את החתימה עם זו שהתקבלה מהלקוח.

3. אם יש התאמה, ולא עברו 5 דקות משליחת הOTP, המפתח הציבורי של הלקוח נשמר בטבלה עם המזהה של הלקוח (מספר הטלפון).

שלב 2: החלפת מפתחות בין לקוחות A ו- B

2.1 בקשת תחילת שיחה חדשה

- לקוח A מודיע לשרת שהוא רוצה להתחיל תקשורת עם לקוח B.

2.2 אישור זמינות של לקוח B

- השרת מוודא שלקוח B מחובר ושולח לו הודעה שלקוח A מעוניין להתחיל תקשורת, לקוח B מאשר או דוחה את הבקשה

- אם לקוח B לא מחובר או שדחה את הבקשה, לקוח A מקבל שגיאה בהתאם.

2.3 השרת מעביר ללקוח A את המפתח הציבורי של לקוח B וללקוח B את המפתח הציבורי של לקוח A

- השרת: (פעמיים פעם שולח ל-A ופעם שולח ל-B)

1. מפעיל SHA-256 על המפתח הציבורי של לקוח B ואז מצפין עם ה-*Private Key* שלו (של השרת).

2. שולח ללקוח A:

- *Public Key* של לקוח B.

- Signature

2.3 אימות המפתח הציבורי שנשלח ללקוחות המעוניינים בתקשורת

- שני הלקוחות:

1. מפענחים את החתימה באמצעות המפתח הציבורי של השרת.

2. מפעילים SHA-256 על המפתח הציבורי שהשרת שלח ומשווים עם החתימה .

3. אם יש התאמה, המפתח הציבורי מאומת והלקוחות שומרים אותו.

2.4 יצירת מפתח סימטרי ושילוב חתימה

- לקוח A:

1. יוצר מפתח סימטרי אקראי K.

2. מצפין את K בשני שלבים:

- פעם ראשונה עם Public Key של B.

- פעם שנייה מפעיל על K פונקציית ריבוב SHA-256 ומצפין עם ה-Private Key שלו (של A) – על מנת ליצור חתימה דיגיטלית (Signature).

3. שולח לשרת:

- K המפתח הסימטרי מוצפן במפתח הציבורי של B.

- Signature

2.5 העברת המפתח ללקוח B

- השרת מעביר את המפתח הסימטרי המוצפן ואת החתימה ללקוח B.

2.6 פענוח ואימות המפתח הסימטרי

- לקוח B:

1. מפענח את המפתח הסימטרי המוצפן באמצעות המפתח הפרטי שלו (של B).

2. מפעיל על K פונקציית ריבוב SHA-256 מפענח את החתימה באמצעות המפתח הציבורי של A ומשווה בינהם לאימות.

3. אם האימות מצליח, המפתח הסימטרי נשמר בקובץ חיצוני.

שלב 3: שליחת הודעות

3.1 הצפנת ההודעה

- לקוח A:

1. כותב הודעה M .

2. מצפין את M עם המפתח הסימטרי באמצעות AES-256.

3. מחשב HMAC-SHA256 מהמפתח הסימטרי ומ-M.

4. יוצר IV חדש

3.2 שליחת ההודעה

- הלקוח:

שולח לשרת:

-  ההודעה המוצפנת.

- HMAC: החתימה על ההודעה.

- IV

- השרת:

- מקבל את ההודעה ללא יכולת לפענחה אותה

- בודק שלקוח B מחובר ומעביר את ההודעה ללקוח B ללא שינוי.

- אם לקוח B לא מחובר השרת שומר את ההודעה בטבלה עד שהלקוח יתחבר.

3.3 פענוח ואימות בצד לקוח B

- לקוח B:

1. מפענח את  עם המפתח הסימטרי והIV.

2. מחשב HMAC ומשווה לערך שהתקבל.

3. אם יש התאמה, ההודעה מאומתת ונקראת.

שלב 4: שליחת אישור ((AC

4.1 יצירת ACK

- לקוח B:

1. יוצר הודעת אישור.

2. מצפין את ה-ACK עם המפתח הסימטרי באמצעות AES-256.

3. יוצר HMAC-SHA256 על 

4.2 שליחת ה-ACK לשרת

- הלקוח שולח לשרת את ה-ACK המוצפן וה-HMAC.

4.3 העברת ה-ACK ללקוח A

- השרת מעביר את ה-ACK ללקוח A.

4.4 פענוח ואימות ה-ACK

- לקוח A:

1. מפענח  עם המפתח הסימטרי.

2. מאמת את ה-HMAC.

3. אם יש התאמה, האישור מתקבל.

מה יש לנו?:

AES – א יוצר מפתח ושולח ל-ב בדרך מוצפנת – אוו ששניהם מסכימים על מפתח מבלי לשלוח אותו

RSA – א יוצר מפתח פרטי וציבורי , שולח בדרך לא מוצפנת את המפתח הציבורי, רק הוא יכול לפענח

KDF – פונקציה ליצירת מפתחות קריפטוגרפי מערך קטן, למשל הפונקציה PBKDF2

SHA-256 – פונקציית גיבוב

בעצם הפונקציה תקבל:  
OTP – 6 ספרות מSMS  
 – ערך רנדומלי כל שהוא ששני הצדדים מסכימים עליו  
 – מספר איטרציות שהפונקציה מבצעת  
 – אורך המפתח

חתימה דיגיטלית בעזרת RSA – לאימות השולח

א' רוצה לשלוח הודעה לב'

א' כותב הודעה M ומפעיל עליה פונקציית HASH לקבלת hash   
א' מצפין את הhash עם המפתח RSA הפרטי שלו

א' שולח את ההודעה + hash המוצפן

ב' מקבל את ההודעה + hash המוצפן   
ב' מפענח את הhash בעזרת המפתח RSA הציבורי של א'

ב' מפעיל פונקציית HASH על ההודעה ומשווה בין ה-hash

HMAC – פונקציית HASH שמקבלת בנוסף מפתח – למשל

פונקציה לא חד-חד ערכית כך שאם הhash נופל לידי תוקף זדוני יהיה לו קשה מאוד לשחזר להודעה המקורית

סודיות – confidentiality

הבטחה שהמידע הרגיש נגיש אך ורק לגורמים מורשים - מתבצע בעזרת הצפנה (AES\RSA)

שלמות – integrity

הבטחה שהמידע מגיע בשלמותו ולא שונה או נהרס בדרך – מתבצע בעזרת חתימות וMAC (HMAC)

מקוריות – authenticity

הבטחה שהמידע שנשלח הגיע מהמקור הרצוי ולא מגורם אחר – מתבצע בעזרת חתימות (SIGNATURE)

הגנה מפני MITM בכל השלבים

הסבר בקצרה את הבחירה ואיך היא משיגה את המטרה:  
א. שיטת ההצפנה (סימטרית או א-סימטרית) ואופן יצירת מפתחות.

השתמשנו בהצפנה א-סימטרית (RSA) עבור החלפת המפתחות מכיוון שבדרך זו רק בעל המפתח הפרטי יכול לפענח את ההודעה ולא צריך להעביר מפתח סודי לצד המצפין ובהצפנה סימטרית (AES) עבור שליחת ההודעות מקצה לקצה בצורה מהירה ויעילה. המפתחות נוצרים בצורה רנדומלית שרק יוצר המפתח יודע מהו.

ב.  תהליך הרישום הראשוני.

לקוח שרוצה להירשם לשירות מקבל OTP דרך ערוץ מאובטח, בעזרתו השרת והלקוח מסכמים על מפתח קריפטוגרפי זמני, הלקוח שולח את המפתח הציבורי חתום על ידי HMAC עם המפתח הזמני בצורה זו השרת יכול לאמת את זהות השולח ולהימנע ממתקפת MITM.

ג. אופן יצירת והחלפת מפתחות, מקום שמירתם (שרת/לקוח) ובטיחות החלפתם.

לכל לקוח יש זוג מפתחות פרטי וציבורי שהוא יצר באקראיות, השרת מקבל את המפתח הציבורי של כל לקוח תוך כדי וידוי אימות. בתחילת סשן בין שני לקוחות השרת שולח לכל צד את המפתח הציבורי של הצד השני.

הלקוח שביקש להתחיל סשן יוצר מפתח בצורה אקראית ושולח לצד השני את המפתח מוצפן וחתום – כך שרק הצד השני יוכל לפענח את המפתח ולוודא אימות ושלמות.

ד. אופן השגת השלמות במובן הרחב מקור ותוכן ההודעה, פרטו את השיטה ואת דרך יישומה.

כל הודעה מוצפנת שנשלחת מקצה לקצה חתומה על ידי HMAC עם המפתח הסימטרי שיש רק לשני הצדדים, בצורה זו הצד המקבל יכול לשחזר את ה-HMAC ולהשוות - על מנת לוודא שההודעה לא שונתה בדרך וגם את זהות שולח ההודעה.

ה. מהלך שליחת ההודעה והאישור על קבלתה.

לקוח א' שולח לשרת הודעה המוצפנת במפתח סימטרי שרק ללקוח ב' יש – כך שום גורם אחר כולל השרת אינו יכול לפענחה. בנוסף נשלחת חתימה, וקטור IV, ומזהה של לקוח ב' (היעד).

השרת בודק לפי המזהה אם היעד מחובר ושולח אליו את תוכן ההודעה ואת מזהה המקור (אם לא מחובר שומר בטבלה עד שמתחבר).

לקוח ב' מפענח את ההודעה בעזרת הIV והמפתח הסימטרי, מחשב חתימה ומשווה לבדיקת אימות ושלמות.

לאחר קבלה תקינה של ההודעה, לקוח ב' יוצר הודעת אישור שגם כן מוצפנת וחתומה ושולח לשרת שמעביר ללקוח א'.

ו. תארו בצורה מפורטת את מבנה (שדות) ההודעה

ההודעה מחולקת ל-5 שדות:

יעד(מזהה הלקוח המקבל), מקור (מזהה הלקוח השולח), מלל מוצפן, חתימה (לשם אימות ושלמות) , וקטור (IV)

ז. תארו בצורה מפורטת את מבנה הנתונים בשרת (מה ואיך נשמר).

- טבלת לקוחות : { מזהה לקוח , מפתח ציבורי }

- טבלת הודעות שלא נמסרו : {יעד ,מקור, ההודעה המוצפנת , חתימה , IV }

חלק 3

האם ניתן להבטיח זמינות?

לא ניתן להבטיח זמינות מהסיבות הנל:

* השרת צריך להיות למעלה תמיד  
  אם השרת מושבת, לא ניתן לבצע שום פעולה במערכת. פעולות כמו רישום לקוחות חדשים, שליחת הודעות, או קבלת הודעות אינן אפשריות במקרה של קריסת השרת.
* השרת מוגבל ל10 לקוחות  
  המערכת תומכת בעד 10 לקוחות בלבד. אם כבר רשומים עשרה לקוחות במערכת, לקוח חדש שרוצה להירשם לא יוכל לבצע רישום או פעולות אחרות כמו שליחת הודעות.
* השרת שומר עד 2 הודעות  
  השרת שומר עד שתי הודעות בלבד עבור לקוחות לא מחוברים. במקרה שבו לקוח לא מחובר וכבר יש שתי הודעות בתור, הודעות נוספות לא יישמרו והן עלולות ללכת לאיבוד