End-to-End Encryption over TCP Protocol

ירין-מיכאל אזולאי (קבוצה 02) ואורי אברגיל (קבוצה 01)

תוכן עניינים

מטרת הפרוטוקול ואופן פעולתו

מטרת הפרוטוקול היא לאפשר תקשורת מוצפנת מקצה לקצה בין זוגות של לקוחות (2-Party E2EE).
הפרוטוקול עושה שימוש בסכמת הצפנה היברידית (Hybrid cryptosystem) ומסתמך על עקרונות של פרוטוקול הצפנה מקצה לקצה קיים בשם PGP (Pretty Good Privacy).

למעשה, השימוש ב-Hybrid cryptosystem משמעותו שימוש במפתח הצפנה פומבי א-סימטרי (במקרה שלנו - RSA) לצורך הצפנה של מפתח סימטרי שאיתו מוצפנת הודעה ארוכה (הצפנה סימטרית כמו AES-CBC דורשת פחות משאבים עבור הצפנה ופענוח מאשר הצפנה א-סימטרית כמו RSA והיא לא מגבילה אותנו בכמות המידע שניתן להצפין – מכיוון שהיא מצפינה בבלוקים) .

בחלק מהבקשות והתשובות בפרוטוקול שלנו, אנו עושים שימוש במפתח סימטרי מסוג AES-CBC באורך של 128 ביטים לצורך הצפנה של הודעות[1] ואת המפתח הסימטרי אנו מצפינים באמצעות RSA.

(1) את המפתח הסימטרי אנחנו מייצרים מחדש בכל בקשה ולא עושים שימוש חוזר במפתח קיים.

הנחות

1. השרת שכתבנו מהימן ומאובטח.
2. תוכן ההודעות שמועברות הוא הדבר היחידי שמסווג כמידע רגיש (Sensitive information).
3. בעת הרשמה של לקוח מתבצע תהליך אותנטיקציה באמצעות OTP המועבר בערוץ צדדי מאובטח כדי לאמת את הלקוח לפני שהוא נרשם בשרת.
4. אנו מניחים כי כל קוד OTP המוזן על ידי הלקוח הוא תקין היות ולא נדרשנו לממש את המנגנון הנ"ל ומותר לנו להניח שהוא מועבר ונשלח דרך ערוץ בטוח.
5. המפתח הציבורי של השרת קבוע בתוכנת הלקוח, עובדה זו מאפשרת לנו לאמת כי תשובות שקיבלנו אכן נשלחו מהשרת. (באמצעות בדיקת החתימה של ה-Hash בכל תשובה שהשרת מחזיר).
6. אנו מניחים שהמפתחות הפרטיים של הלקוחות לא עשויים לדלוף וכי מפתח ההצפנה הסימטרי שבו אנו עושים שימוש להצפנת הודעות נוצר באופן מאובטח ואקראי. כתוצאה מהנחה זו אין לנו צורך להשתמש ב-Diffie-Helman וב-KDF כדי לגזור מפתח משותף.
7. השרת עושה שימוש ב-RSA-2048 והלקוחות ב-RSA-1024 (כדי שיהיה ניתן להצפין מפתח פומבי של לקוח בעזרת המפתח הפומבי של השרת)

מבנה בקשה

Request	Field	Size	Description
Header	phone_id	10 Bytes	Identifies the sender
	dest_phone_id	10 Bytes	Identifies the destination client
	code	1 Byte	Request code
	timestamp	4 Bytes	Timestamp
	payload_size	4 Bytes	Additional payload size (in bytes)
Payload	payload	Variable size	Additional payload
Signed Hash	hash	128 Bytes	SHA-256 hash of the request signed using sender's private key.

סוגי בקשות

100 – Reconnect

No additional payload.

101 – Register

Field	Size	Description
Public Key	Variable size	RSA Public key (Encrypted using server's public key)

102 – Get public keys

No additional payload.

103 – Send message

Field	Size	Description
Encrypted AES key	128 Bytes	AES key encrypted using the end user's public key
AES-IV	16 Bytes	AES Initialization vector
AES encrypted message	Variable size	The encrypted message

אופן הצפנה וחתימה של בקשה:

1. את ה-Header מצפינים עם המפתח הציבורי של השרת (תמיד).
2. את ה- Payload :
   a. במקרה של בקשה 103 (שליחה של הודעה) – ה-Payload מכיל מפתח AES שמוצפן עם המפתח הפומבי של **הנמען**, ה-IV לא מוצפן (אין צורך), תוכן ההודעה מוצפן באמצעות המפתח AES.
   b. בקשת הרשמה – ה-Payload מכיל את המפתח הפומבי של הלקוח שנרשם והוא מוצפן באמצעות המפתח הציבורי של השרת.
   c. ב-2 סוגי הבקשות האחרות (100 ו-102) אין Payload לכן רק ה-Header מוצפן, כאמור, עם המפתח הפומבי של השרת.

3. <u>Hash וחתימה:</u>
   a. במקרה שאין Payload (בקשות 100 ו-102):
   ה-Hash מחושב על ה-Header בלבד לפני ההצפנה שלו ולאחר מכן נחתם בעזרת המפתח הפרטי של השולח.
   b. במקרה שיש Payload (בקשת Register - 101):
   ה-Hash מחושב על ה- Header + Payload לפני ההצפנה של כל אחד מהם בנפרד ולאחר מכן נחתם בעזרת המפתח הפרטי של השולח.
   במקרה של בקשת Send Message (103) ה-Hash מחושב על ה- Header + Payload לפני ההצפנה של ה-Header ולפני ההצפנה של המפתח AES ב-Payload, לאחר מכן חותמים את ה-Hash בעזרת המפתח הפרטי של השולח.

מבנה תשובה

Response	Field	Size	Description
Header	phone_id	10 Bytes	Identifies the sender (zeros if server is the sender)
	dest_phone_id	10 Bytes	Destination user
	code	1 Byte	Response code
	timestamp	4 Bytes	Timestamp (prevent replay-attacks)
	payload_size	4 Bytes	Additional payload size (in bytes)
Payload	payload	Variable size	Additional payload
Signed Hash	hash	128 /256 Bytes	SHA-256 hash of the response signed using server's private key[2]

(2) במקרים שבהם הקוד שמופיע בשדה התשובה הוא 103, ז"א שמדובר בהודעה ששלח משתמש קצה אחר ולכן ה-Hash באותה תשובה נחתם על ידי השולח ולא על ידי השרת וגודלו 128 בתים.

סוגי תשובות

200 – Register success

No additional payload.

201 – Reconnect success

No additional payload.

202 – Share public keys

Field	Size	Description
AES Key (Encrypted w/RSA-1024)	128 Bytes	AES Key
AES-IV	16 Bytes	AES Initialization vector
List of public keys of users* (Encrypted w/AES)	Variable size	RSA Public keys

רשימת המפתחות הציבוריים נשלחת כמחרוזת בפורמט הבא :
List = "phone_id**:**RSA_KEY (ENCODED), phone_id**:**RSA_KEY (ENCODED) ..."

כאשר המפתחות מקודדים בפורמט DER ולאחר מכן מומרים למחרוזת הקסאדצימלית.

203 – Message was transferred to the end-user successfully.

No additional payload.

204 – End-user is offline and will receive the message upon reconnect.

No additional payload.

103 – Send message (Special case – not exactly a response, but a forwarded request)

אופן הצפנה וחתימה של תשובה:

1. את ה-Header השרת מצפין עם המפתח הציבורי של משתמש הקצה.
2. את ה-Payload:
   a. אם מדובר בהעברה של בקשה 103 על ידי השרת לנמען אזי הלקוח ששלח את הבקשה כבר הצפין את מפתח ה- AES בעזרת המפתח הפומבי של הנמען, תוכן ההודעה מוצפן באמצעות מפתח ה-AES.
   b. תשובה לשיתוף מפתחות ציבוריים של משתמשי קצה - התהליך דומה להצפנה של בקשת הודעה.
   ה-Payload מכיל מפתח AES שהוצפן בעזרת המפתח הציבורי של הנמען ותוכן ההודעה מוצפן באמצעות מפתח ה-AES.
   c. בכל מקרה אחר אין payload.
3. <u>Hash וחתימה:</u>
   a. במקרה שאין Payload:
   ה-Hash מחושב על ה-Header בלבד לפני ההצפנה שלו ולאחר מכן נחתם בעזרת המפתח הפרטי של השרת.
   b. במקרה שיש Payload (תשובה 202 - Share public keys):
   ה-Hash מחושב על ה- Header + Payload כאשר ה-Header עדיין לא הוצפן וכאשר ה-Payload הוא ה-Payload לפני ההצפנה של מפתח ה-AES באמצעות המפתח הפומבי של הנמען. לאחר חישוב ה-Hash חותמים אותו בעזרת המפתח הפרטי של השולח. (זהה לתהליך שנעשה בעת חישוב Hash ושליחת בקשה 103 של הודעה שתואר קודם לכן)

תהליך הרשמה

כאשר לקוח לא מחזיק זוג מפתחות (Key-pair) הוא צריך לבצע הרשמה לשרת:

1. הלקוח נדרש להזין מספר טלפון שישמש אותו בתור מזהה.
2. הלקוח מאומת באמצעות המספר טלפון שהוא הזין דרך קוד OTP (מניחים שזה אכן כך)
3. מספר הטלפון של הלקוח נשמר בקבוץ לוקאלי הנקרא number.txt
4. תוכנת הלקוח מייצרת זוג מפתחות RSA פרטי וציבורי באורך 1024 ביטים, הנשמרים לוקאלית בקבצים client_private.pem ו-client_public.pem בהתאמה.
5. נשלחת בקשת הרשמה לשרת המכילה את המפתח הציבורי של הלקוח.
6. אם הבקשה הצליחה, השרת יחזיר תשובה שההרשמה הצליחה. אחרת, השרת יסגור את החיבור.

אם הבקשה לא תואמת לפרוטוקול או ה-Hash לא נכון אז השרת סוגר את החיבור אל מול אותו לקוח

תהליך חיבור מחדש

תוכנת הלקוח תעשה שימוש בקבצים client_private.pem, client_public.pem והקובץ number.txt כדי לשלוח בקשת התחברות מחדש.

1. במידה והקבצים קיימים בתיקיית תוכנת הלקוח :
   a. תשלח בקשת Reconnect לשרת (code 100).
   b. אם ההתחברות מחדש צלחה, השרת יחזיר תשובה חיובית (קוד 201).
   אחרת, השרת יסגור את החיבור.

תהליך שליחת הודעה

לאחר הרשמה או התחברות מחדש של לקוח:

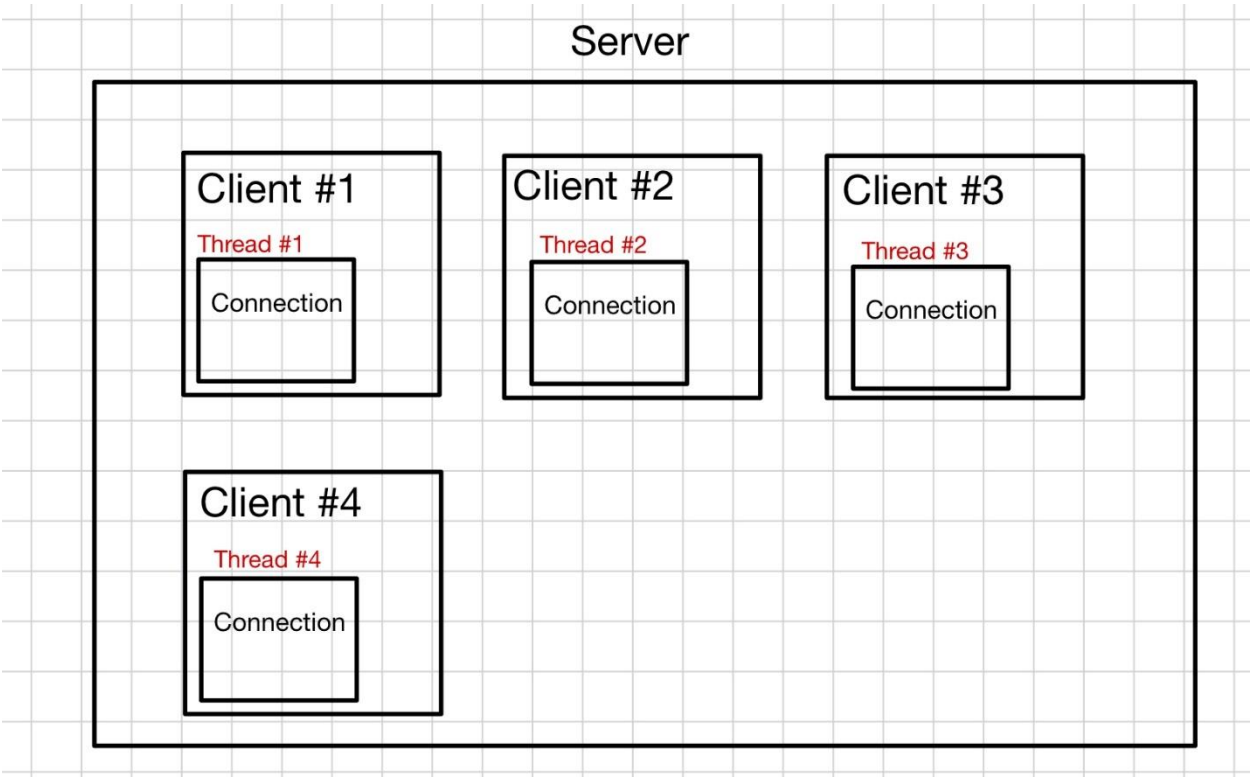
1. תוכנת הלקוח מבצעת בקשה יזומה לקבלת כל המפתחות הפומביים שנמצאים בשרת.
2. הלקוח נשאל מיהו הנמען אליו הוא מעוניין לשלוח הודעה.
3. תוכנת הלקוח מחפשת האם היא קיבלה את המפתח הפומבי של אותו נמען.
4. במידה והנמען נמצא, הלקוח יתבקש להזין את ההודעה שהוא מעוניין לשלוח.
5. נשלחת בקשת הודעה לשרת לצורך העברת ההודעה לנמען (קוד 103).
6. במידה והלקוח מחובר, מועברת אליו ההודעה ומובטח לנו שהיא תתקבל אצלו (הפרוטוקול ממומש מעל TCP). לאחר העברת ההודעה נשלח חיווי לשולח על כך שההודעה נמסרה לנמען בהצלחה (קוד 203).

אחרת, במידה והנמען לא מחובר, ישלח חיווי לשלוח על כך שהנמען לא מחובר כעת וההודעה תשמר ברשימת הודעות ממתינות עבור אותו נמען. ברגע שהנמען יתחבר מחדש לשרת יועברו אליו כל ההודעות הממתינות.

אופן החזקת המידע בשרת

בכל התחברות של לקוח מיוצר עבורו בשרת אובייקט מסוג Client המכיל פרטים המתארים את אותו לקוח כמו מספר הטלפון, המפתח הציבורי שלו, רשימה של הודעות הממתינות עבורו ואובייקט מסוג Connection שדרכו השרת מתקשר עם הלקוח (אובייקט זה מחזיק רפרנס ל-socket שדרכו הלקוח מחובר לשרת). בכל התחברות מחדש של לקוח מעדכנים את אובייקט ה-Connection של אותו לקוח כך שיהיה מעודכן ויחזיק את האובייקט של הסוקט שדרכו התחבר הלקוח מחדש (כדי שנוכל להעביר לו הודעות).

אילוסטרציה:



שמירה על שלמות המידע (Integrity)

לכל בקשה ותשובה מחושב Hash אלגוריתם SHA-256 כך שנוכל להבטיח את שלמות המידע המועבר דרך הסוקט.
לפני הצפנה של כל בקשה או תשובה, מתבצע חישוב של פונקציית ה-Hash על הבקשה \ תשובה הלא מוצפנת, לאחר מכן היא מוצפנת וה-Hash נחתם על ידי המפתח הפרטי של השולח.
כל המידע לאחר מכן מועבר אל הנמען דרך הסוקט.
הנמען מקבל את המידע מפענח אותו, לאחר מכן הוא מחשב את Hash ובודק האם הוא זהה למה שקיבל ומתאים לחתימה של מי ששלח אותו. אם כן, ניתן להיות בטוחים שהמידע לא שונה או נפגם במהלך ההעברה. אם הערכים אינם תואמים, הדבר מעיד על שינוי אפשרי הנובע מתקלה או ניסיון זדוני לשנות את הבקשה.
כאשר השרת מקבל בקשה מלקוח שבה ה-Hash לא תקין הוא סוגר את החיבור אל מול אותו לקוח מחשש שמדובר בלקוח שמנסה לעשות פעולות זדוניות.

שמירה על אותנטיות

השימוש המשולב ב- One-Time Password (OTP) ובחתימות דיגיטליות מאפשר לנו להבטיח שמירה על אותנטיות המידע המתקבל מלקוחות.
שכן, בעת ההרשמה הראשונית באמצעות ה-OTP מובטח לנו זיהוי של המשתמש אל מול מספר הפלאפון הנייד שלו כתוצאה מזיהוי מוצלח באמצעות OTP הלקוח משויך לזוג מפתחות RSA שישמש אותו לצורך ההתקשרות עם השרת והלקוחות.
מכיוון שהלקוח חותם כל בקשה שהוא שולח בעזרת המפתח הפרטי שלו, מובטח לנו שכל הודעה שישלח הלקוח תהיה מזוהה איתו בלבד ולא נוכל להפריך את אותנטיות הבקשה.
לכן, מנגנון ה-Hashing בשילוב עם החתימה מבטיח לנו שלמות במובן הרחב, שכן לא ניתן לשנות את מקור או תוכן ההודעה מבלי שנבחין בכך.

שמירה על סודיות המידע

סודיות המידע מובטחת לנו באמצעות השימוש במנגנוני ההצפנה ובאמצעות ההנחה שהשרת מהימן ומאובטח (בין אם באופן פיזי ובין אם באופן דיגיטלי).
הצפנת המידע שנשלח על גבי הסוקט מאפשרת לנו להבטיח שהמידע שעובר ברשת לא ניתן לפענוח אלא על ידי מי שהמידע מיועד עבורו.
ההנחה שמפתחות פרטיים של משתמשים לא דולפים מספיקה לנו כדי להבטיח שתוכן ההודעות (המידע הרגיש) יהיה מוצפן מפני מי שאינו רשאי לקרוא אותו.
במידה ודלף המפתח הפרטי של השרת (למרות שאנחנו מניחים שהוא מאובטח), תוקף יהיה מסוגל לפענח רק מתי נשלחות הודעות ולמי, אך לא יהיה מסוגל לקרוא אותם או לשנות את תוכנן.

זמינות

השרת מבטיח זמינות על ידי תמיכה ב-Multithreading, לכל חיבור שנוצר אל מול השרת מוקצה Thread שהוא ייעודי לאותו חיבור.
אם דרך חיבור מסוים נשלחות בקשות לא מזוהות או בקשות עם Hash לא תקין, השרת דואג לסגירת אותו חיבור כדי שלא יעמיס על השרת.
הזמינות מובטחת לנו גם באמצעות מנגנון האותנטיקציה (Authentication) והאותוריזציה (Authorization) שהשרת נוהג על פיו :
כל משתמש שרשום בשרת הוא משתמש שעבר אימות ובפרט רק משתמשים רשומים ומזוהים רשאים לבצע אינטראקציה אל מול השרת.
משתמש שזהותו חשופה בדרך כלל אינו נוטה לבצע פעולות זדוניות כאשר הוא יודע שניתן יהיה להתחקות אחריו.
כל משתמש שלא עבר אותנטיקציה איננו רשאי לבצע אינטראקציה עם השרת ללא ביצוע התחברות מחדש ובפרט כל ניסיון שכזה יעלה לו בסגירת החיבור מצד השרת.
<u>הערה</u>: ניתן להגביר את הבטחת הזמינות על ידי מנגנונים נוספים כמו IP Blacklisting / Whitelisting כדי לחסום לצמיתות חיבורים נכנסים מכתובות IP שמהן מגיעות בקשות סתמיות.