Secure Banking Transaction System

שחר מרקוביץ

נאור ממן

צוף נויפלד

מכון לב, סמסטר ב' ה'תשפ"א

מערכות להרצת קוד בסביבה בטוחה

בהנחיית ברק עינב

תוכן עניינים

מטרת הפרויקט

חשבון הבנק שלנו הוא משהו שמאוד חשוב ויקר לכל אחד ואחת מאיתנו. לא היינו רוצים שכל אחד יוכל להיכנס בחופשיות לחשבון שלנו ולעשות פעולות בשמנו.

כדי למנוע מצב זה, קיימים היום מנגנונים רבים עבור אימות המשתמש בשני דרכים(2-Factor-Authentication), החל בשליחת SMS לטלפון של המשתמש, דרך שימוש ב-Authenticator Apps עבור 2FA ועד שימוש במפתח פיזי. הבעיה בדרכים אלו – הן לא מספיק מאובטחות:

- הבעיה ב-SMS ואפליקציות היא שהן לא מבודדות ואינן רצות בסביבה מאובטחת, כלומר תוכנה שיושבת על המחשב/מייל/טלפון תוכל לקרוא את הסיסמא החד פעמית(OTP) שנוצרה ולהכניס אותה בעצמה ובכך אין חידוש.
- הבעיה במפתח פיזי – ישנה תלות גבוהה מדי בין המשתמש לבין הרכיב: ומה קורה עם הוא נאבד או נשבר? המשתמש לא יכול להיכנס לחשבון שלו?

באמצעות Intel DAL© ניתן ליצור שיטה מאובטחת יותר עבור אותנטיקציה כפולה.

המטרה, והיתרון בשימוש ב-DAL, היא ההבטחה שרק האדם האמיתי, המשתמש, הוא זה שעושה את הפעולות וחשוף אל הסיסמא החד פעמית, ולא אף ישות/תוכנה אחרת שיכולה להוות סיכון. ומפני כך, אנחנו משתמשים בסביבה זו.

חלקים בפרויקט

הפרויקט שלנו מתחלק לשלושה חלקים עיקרים:

1. Server: שרת, המדמה שרת בנק אמיתי, אשר ניתן להתחבר עליו, להירשם ולעדכן פרטים, לשלוח, להפקיד ולקבל כספים. כתוב ב- Python.
2. Intel DAL: במערכת זו נבצע את כל הפעולות הקריטיות והרגשיות, היא זו שתבצע הצפנות ותחזיק מידע רגיל של הלקוח, ולא תוכנת הלקוח עצמה. כתוב ב-Java.
3. Client: תוכנת צד הלקוח, המדמה את התוכנה שרצה בצד הלקוח, אשר דרכה משתמש הקצה עובד ופונה על מנת לקבל את שירותי הבנק. כתוב ב- #C.

תוכנת צד הלקוח מעבר להיותה רכיב הGUI שאיתו המשתמש מתממשק עם השרת, לפני ואחרי כל הודעה שתשלח לשרת, הא תשלח קודם כל אל רכיב ה-DAL על מנת להצפין או לפענח את ההודעה.

עקרונות פיתוח ואופן עבודת הפרויקט

ארבע עקרונות פעולה כלליים:

1. **הצפנה.** עם הפעלת תוכנת הלקוח, רכיב ה-DAL יוצר מפתח RSA אסימטרי על מנת לקבל בתורה מאובטחת את מפתח ההצפנה של אותה השיחה מהשרת. שאר השיחה מתבצעת על ידי הצפנה סימטרית לפי אלגוריתם AES במודל CBC, מכיוון שהצפנה סימטרית יותר מהירה ופשוטה לחישוב עבור המעבד מאשר הצפנה אסימטרית.
2. **שימוש ב-TOTP**. כאשר נוצר משתמש חדש, השרת שולח למשתמש את shared secret אשר ישמש את שניהם ביצירת ה-TOTP token. ערך זה בצורה הבאה בלקוח, ולא בDAL:

$$pair\ of\ account\ ID\ and\ Base32\left(E_{AES}(shard\ secret)\right)$$

אשר מפתח AES שמור בתוך ה-DAL בצורה מאובטחת.

חשוב: מפתח ההצפנה זהה בין כל המשתמשים השונים עבור אותו רכיב DAL, אח הערך הסודי כמובן שונה בין המשתמשים השונים.

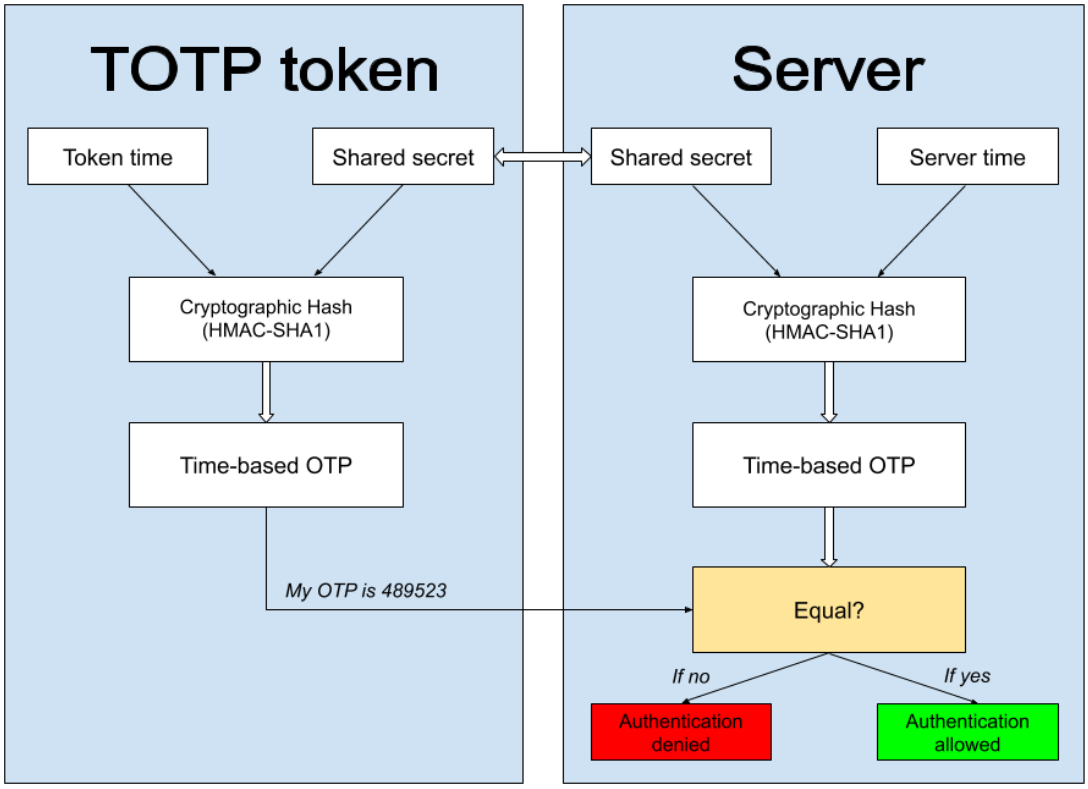


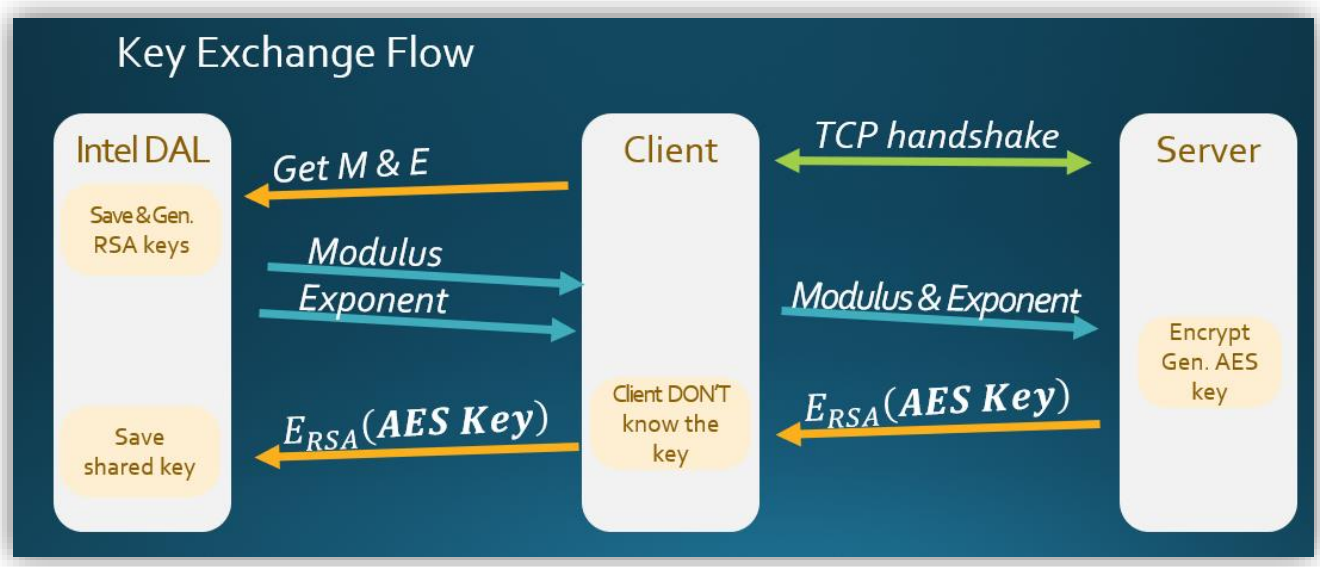
Figure 1 TOTP Algorithm Illustration

3. **העברת מידע חיוני בלבד.** רק מידע חיוני ושבאמת צריך נשלח אל המשתמש מהשרת.

4. **שמירה על חווית משתמש נוחה.** אין שימוש ב-2FA לפני ביצוע כל משימה ומשימה. קיים אח ורק לפני ביצוע משימות קריטיות – העברת כסף בין חשבונות, הפקדה ומשיכה של מזומנים.

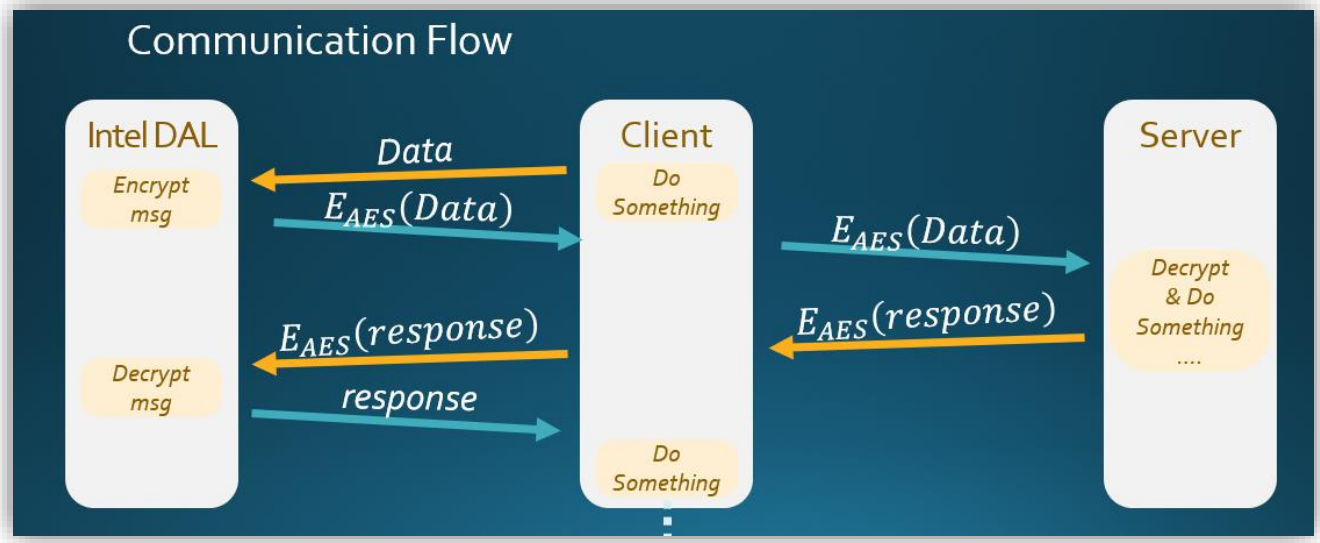
כיצד התקשורת עובדת

<u>תהליך החלפת המפתחות:</u>



לאחר שביצענו את לחיצת היד המשולשת של TCP, תוכנת הלקוח פונה לDAL כדי ליצור צמד מפתחות של RSA. כעת DAL מחזיר ללקוח את מפתח RSA ציבורי E ואת המודולו M המתאים. שני פרטים אלו נשלחים לשרת בצורה לא מוצפנת. כעת השרת יצור מפתח סימטרי ע"פ אלגוריתם AES, עבור השיחה עם אותו לקוח, ויצפין אותו בעזרת הפרמטרים שנאמרו לעיל את המפתח הציבורי וישלח אותו חזרה ללקוח. חשוב להדגיש כי הלקוח **אינו** יודע מה מפתח ההצפנה של AES – הוא פשוט מעביר את ההודעה המוצפנת ל-DAL הוא יפענח אותה, יסיק שמדובר במפתח השיחה וישמור אותו אצלו.

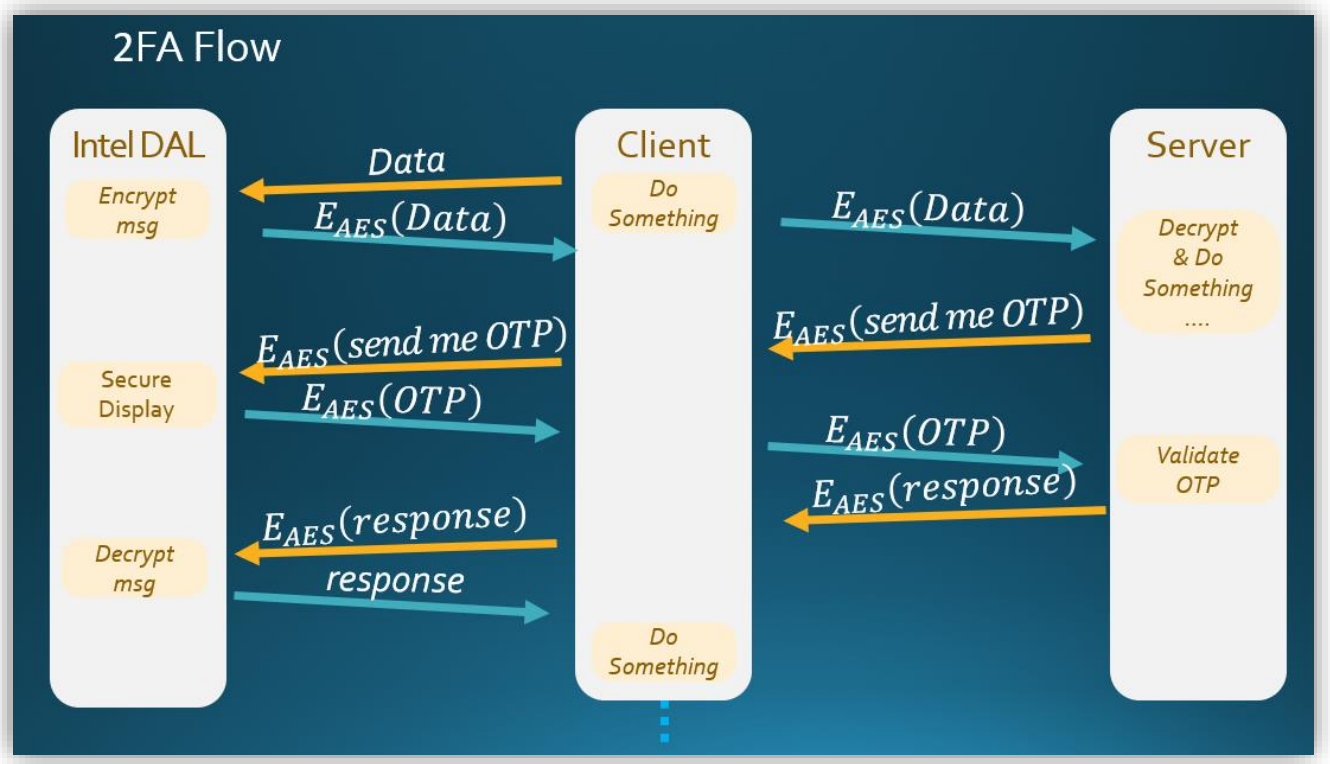
<u>תקשורת פשוטה בין השרת ללקוח:</u>



לאחר שהלקוח מבצע פעולה כל שהיא, לפני שליחתה לשרת לצורך עיבוד, היא תשלח ל-DAL על מנת לעבור תהליך הצפנה. לאחר מכן, תשלח ההודעה המוצפנת לשרת. כשהשרת יקבל את ההודעה המוצפנת, הוא יפענח אותה, יפעל בהתאם וישלח את התגובה המוצפנת המתאימה. כפי שהסברנו לעיל, הלקוח לא יודע מה לעשות עם המידע המוצפן, לכן הוא ישלח אותו ל-DAL כדי שיפענח אותו.

כל הפעולה הזאת חוזרת חלילה עבור כל אופרציה ואופרציה שהלקוח עושה מול השרת. עם זאת, ישנו הבדל קטן ומהותי עבור פעולות קריטיות אשר דורשות אותנטיקציה נוספת.

<u>תקשורת בין השרת ללקוח אשר מצריכה 2FA:</u>



תחילת התקשורת היא בדיוק כמו בחלק הקודם. אולם אחרי שהשרת מפענח את ההודעה ומסיק שמדובר בפעולה שדורשת אותנטיקציה נוספת, הוא ישלח הודעה מתאימה ל-DAL, שתגרום להפעלת ה-Secure Display. דרכה, הלקוח יוכל להזין בצורה מאובטחת את ה-TOTP. לאחר מכן, ה-TOTP יעבור בצורה מוצפנת לשרת על מנת שהוא יאמת אותו. אם ה-OTP ווליד – תשלח הודעת אישור לפעולה, אחרת – תשלח הודעת שגיאה והלקוח יצטרך להזין את ה-TOTP מחדש או לבטל את הפעולה.

כמו כן, התהליך יכול להתבצע הלוך וחוזר על עצמו במקרה הצורך.

איך משתמשים בפרויקט

1. יש לוודא כי סביבת DAL Emulator מותקנת ועובדת. יש לפחות ל-Intel על מנת לקבל עותק מהם.
2. התקנת Eclipse
3. חילוץ הקבצים המצורפים לתיקייה נוחה במחשב.
4. ווידוא כי הנתיב הבא קיים, ואם לא - ליצור אותו: C:\bin
5. התקנת Python
6. הרצת השרת
7. הרצה של פרויקט DAL ב- Eclipse **בלבד**
8. הרצה של הלקוח

לאחר ביצוע שמונה פעולות פשוטות עלו יפתח לפניכם תפריט ראשי אשר באמצעותו אפשר לפתוח חשבון חדש, להתחבר לקיים ולבצע שלל פעולות על משתמש רשום.

יעדי אבטחה עיקריים

מטרת הפרויקט היא כשלעצמה יעד אבטחתי – ביצוע 2FA בצורה המאובטחת ביותר שאפשר ובצורה מבודדת, כלומר תוך הבטחה כי המשתמש, בן אדם, הוא זה שיכניס את הקוד ולא גורם שלישי אחר לא רצוי.

אופן העבודה

העבודה על הקוד חולקה כך שכל אחד תרם את חלקו והוסיף מהידע שלו. נתקלנו בקטעים המסובכים יותר של העבודה על הקוד בבאגים שדרשו מאיתנו זמן רב כדי לתקנם, אך באמצעות עבודת צוות וחקירה משותפת הצלחנו להתגבר על כך.

עבודה עתידית

1. הפרויקט שלנו לא נייד כרגע ועובד אח ורק על מחשב יחיד.

הגורם לבעיה: באת יצירת משתמש חדש **בלבד**, נשלח secret מהשרת לDAL לצורך חישוב TOTP. ערך זה שמור בצורה מוצפנת במחשב של המשתמש ולא ניתן לפתוח אותו על מחשב אחר. כלומר, החלפת מחשב תגרום לכך שלא ניתן לבצע 2FA. נרמה למצוא צורה אחרת לשמור ערך זה אשר תאשר גם את ניידות הפרויקט.

2. שילוב מנגנון האותנטיקציה שלנו עם שיטות 2FA אשר כבר קיימות בשוק.

תועלת: המערכת שלנו תהיה חזקה ומאובטחת יותר. כמו כן היא תהפוך מ-2FA ל-MFA.

3. תמיכה במספר לקוחות המקביל. נכון לנקודת זמן זו, השרת תומך רק בלקוח אחד שמחובר אליו, דבר שלא יתכן באתר אמיתי, בטח לא בשרת של בנק. נרצה לשפר ולהוסיף עבודה במקביל עם ריבוי לקוחות.

4. עמידה בפני התקפת *'אדם שבתווך'* (Man In The Middle). הוספה של סרטיפיקטים (Certificate) בשלב החלפת המפתחות הקריפטוגרפים. בעולה זו תוכל לאפשר לכל אחד מהצדדים לדעת בוודאות שהוא משוחח עם צד שני לגיטימי ולא עם מתחזה.

5. שימוש ב-Secure Input. כאשר המשתמש מכניס מידע קריטי, לדוגמא כמות הכסף שהוא מעביר לחשבון אחר וכן מספר החשבון אליו יש להעביר את הכסף, מידע זה מוכנס לא בצורה מאובטחת, בצד הלקוח. כלומר – אם יש קוד זדוני בעל הרשאות מספיק גבוהות והוא יכול לגשת לזיכרון של התוכנה שלנו – הוא יכול לשנות את כמות הכסף שיש להעביר/ את מס' החשבון אליו יעבור הכסף. כדי למנוע בעיה זו, יש להשתמש ב-Secure Input אשר יבטיח, בדיוק כמו Secure Display שמי שמכניס את המידע הוא אכן משתמש לגיטימי.