

ספר פרויקט גמר:



שם מכללה: מכללת מקיף ה דרכא אשקלון.

סמל מוסד: 644450.

שם הסטודנט: לידור דוקרקר.

ת.ז הסטודנט: 324948058.

שם המנחה: גל בראון.

שם הפרויקט: פורום פומבי עבור ארגונים ממלכתיים

או פרטיים מאומתים באמצעות מנגנון PKI בשילוב

צא'ט מאובטח.

תוכן עניינים

4.....	תיאור הפרויקט:
6.....	רקע תיאורטי בתחום הפרויקט:
6.....	תשתית PKI:
6.....	קריפטוגרפיה:
7.....	הצפנה סימטרית:
7.....	הצפנה אסימטרית:
7.....	תהליך פתיחת חדר/הצטרפות לחדר קיים:
7.....	תהליך החלפת המפתחות:
8.....	תהליך הצפנת המידע ופענוח המידע:
8.....	תהליך סגירת חדר/התנתקות מחדר:
8.....	תהליך הרישום/התחברות:
9.....	תיאור הבעיה האלגוריתמית:
10.....	החלפת המפתחות:
14.....	העברת המידע בין הלקוחות:
14.....	AES – Advanced Encryption Standard:
16.....	מודל שרת לקוח:
16.....	צד השרת:
17.....	צד הלקוח:
17.....	תיאור הטכנולוגיה:
17.....	סביבת עבודה ושפות תכנות:
17.....	מסד הנתונים Mysql:
18.....	פרוטוקולי תקשורת:
18.....	מסד הנתונים:
19.....	לוחות זמנים:
20.....	מבוא:
20.....	הרקע לפרויקט:
21.....	תהליך המחקר:
21.....	אתגרים מרכזיים:
22.....	מטרות/יעדים:
23.....	מדדי הצלחה למערכת:
23.....	רקע תיאורטי:
23.....	קריפטוגרפיה:
23.....	הצפנה סימטרית:
24.....	הצפנה אסימטרית:
24.....	גרפיקה GUI:

24.....	ניתוח דרישות המערכת:
25.....	אילוצים:
25.....	תיאור הרכיבים בפתרון:
36.....	צד שרת:
37.....	צד לקוח:
38.....	תיאור ההצפנות ואלגוריתמים מרכזיים:
38.....	החלפת המפתחות RSA (שיחה בחדר):
40.....	החלפת המפתחות בשיטת RSA בשיחה פרטית:
44.....	PRNG (Pseudorandom number generator) מחולל מספרים פסידו-אקראיים
46.....	OTP (One Time Pad) פנקס חד פעמי -
48.....	HMAC algorithm אלגוריתם HMAC:
53.....	תהליך התחברות או הירשמות למערכת:
56.....	מבני הנתונים בהם עשיתי שימוש:
56.....	בלקוח:
56.....	בשרת:
57.....	עץ מודולים:
57.....	לקוח:
58.....	שרת:
58.....	מודלים שבהם הלקוח והשרת עושים שימוש:
59.....	תרחיש use-case:
60.....	תיאור המחלקות:
63.....	ארכיטקטורת המערכת:
64.....	תיאור התוכנה:
64.....	שפות התכנות:
65.....	מדריך למשתמש:
65.....	תיאור הליך התקנת המערכת:
66.....	משתמשי המערכת:
67.....	אופן הפעלת המערכת:
72.....	סיכום אישי:
72.....	תיאור תהליך העבודה הצלחות ואתגרים:
72.....	מה נלמד מהפרויקט:
72.....	תובנות ומסקנות:
73.....	כלים להמשך:
73.....	ביבליוגרפיה:

הצעת פרויקט:



שם מכללה: מכללת מקיף ה דרכא אשקלון

סמל מוסד: 644450

שם הסטודנט: לידור דוקרקר

ת.ז הסטודנט: 324948058

שם מנחה: גל בראון

שם הפרויקט: פורום פומבי עבור ארגונים ממלכתיים או פרטיים מאומתים

באמצעות מנגנון PKI בשילוב צא'ט מאובטח

תיאור הפרויקט:

במהלך לימודי במכללת מקיף ה דרכא אשקלון, היה עלי לפתח פרויקט גמר. כבר במהלך לימודי ידעתי שאעסוק בבעיות חשיבתיות ומתמטיות העוסקות בקריפטוגרפיה ולכן רציתי שהפרויקט שלי יכלול אלמנטים מתמטיים שלא נגעתי בהם וארצה להעשיר את הידע של תקשורת בין מחשבים הקריפטוגרפיה, ממשק המשתמש ומסד הנתונים.

דוגמא קלאסית לצ'אט מאובטח ברמה גבוהה הוא Whatsapp המשתמש את המשתמש בכך שיש לו שיחות וידיאו, שיתוף קבצים, שיחות פרטיות, שיחות קבוצתיות, הרשמה של משתמשים ועוד...

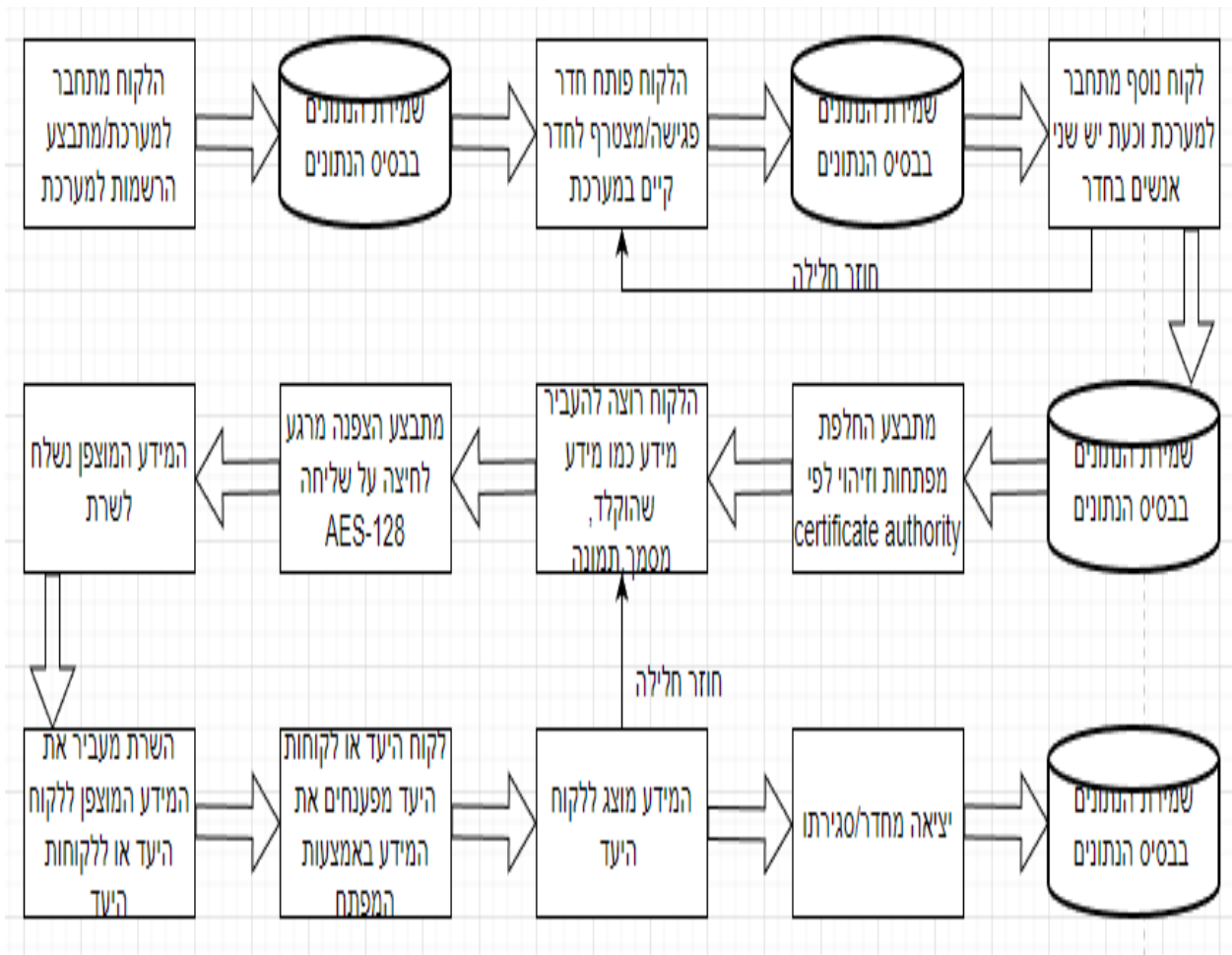
הפרויקט שלי עוסק בתקשורת בין מחשבים ברשת המקומית (LAN), אשלב את קריפטוגרפיה שתיתן אבטח של השיחות מבלי שתוקף יוכל לפרוץ את ההודעות שהוא יחלץ

כלומר לפענח אותן, ארצה בנוסף לפתח ממשק גרפי GUI נעים וברור למשתמש ומסד נתונים DB שיכלול את המידע על החדרים והמשתמש.

הקריפטוגרפיה בפרויקט יהוו פרוטוקול תקשורת מאובטח שיגרום להעברת המידע בין מחשבים ברשת המקומית בתוכנת התקשורת באופן מאובטח כלומר השרת ואנשים חיצוניים שלא קשורים לתקשורת לא יוכלו לראות את המסרים בין המחשבים כלומר לפענח את המסרים השרת אחראי רק להעברה של ההודעות בין שני לקוחות.

הפורום פומבי עבור ארגונים ממלכתיים או פרטיים מאומתים באמצעות מנגנון PKI את זה אממש באמצעות ה- CA זהו ארגון סמכות שיש לו הרשאות וכולם סומכים עליו אותו פורום הוא הארגון והוא פתוח לכולם והוא עבור ארגונים כמו ביטוח לאומי או לקוחות פרטיים אשר מאומתים באמצעות חתימה דיגיטלית שזה בעצם ה- PKI, האימות נועד על מנת לאמת את האדם שאנחנו מדברים למנוע התחזות ולחזק את הפרטיות בין הלקוחות. זהו באופן כללי בהמשך הצעת הפרויקט אני אכנס עמוק יותר לבעיות ולפתרונות ואסביר לעומק על הפרויקט שלי.

תרשים זרימה:

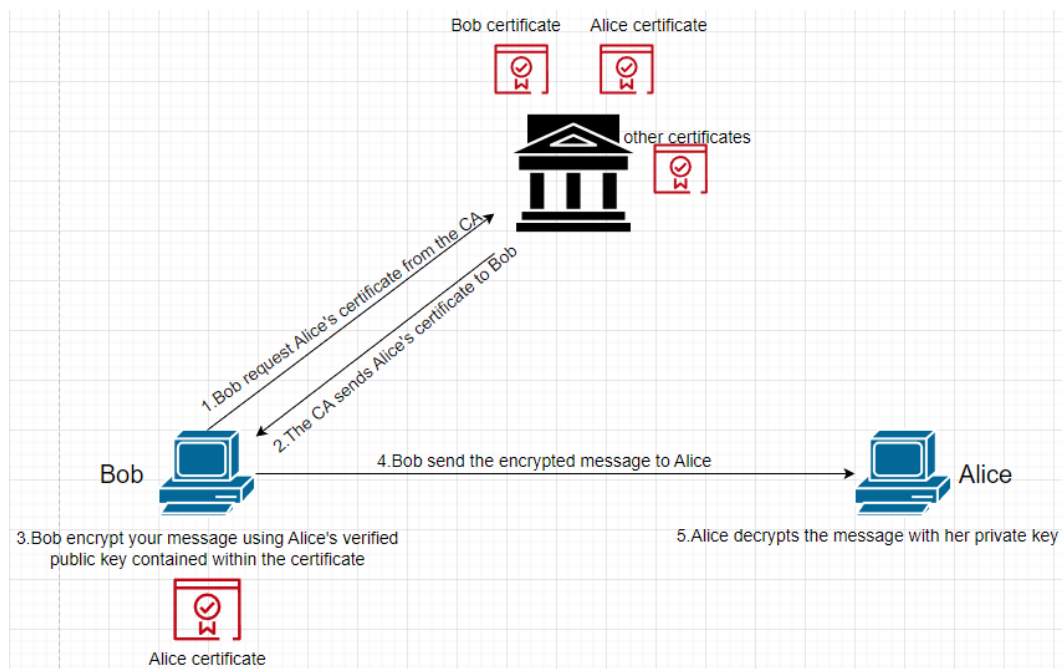


רקע תיאורטי בתחום הפרויקט:

תשתית PKI:

מערכת PKI הינה תשתית המבוססת על הצפנה אסימטרית.

הוא ההסכם המקשר מפתחות ציבוריים על הלקוחות, באמצעות רשויות אמון – Certificate Authority, אשר בסמכותן להנפיק תעודות דיגיטליות. הקשר מיוסד בתהליך של רישום וניפוק. הניפוק מתרחש אצל רשות האמון (באמצעות תוכנת מחשב או פיקוח אנושי), בהתבסס על רמה מסוימת של אמון בין הצדדים. כל הרעיון של חתימה דיגיטלית ו – PKI בנוי בהנחה שכולם סומכים על ה – PKI. והוא עובד כך:



קריפטוגרפיה:

קריפטוגרפיה היא שיטה להגנת מידע ותקשורת באמצעות קודים כך שרק מי שבעבורו מיועד המידע יוכל לקרוא אותו פירוש המילה ה"קריפה" היא "מוסתר" או "כספת" ופירוש המילה "גרפי" היא "כתיבה". כל מערכת שמעבדת את עצמה ואת לקוחותיה משתמשת באבטחה המרבי והמקסימלי.

הצפנה סימטרית:

בקריפטוגרפיה הצפנה סימטרית או צופן סימטרי הוא אלגוריתם הצפנה שבו משתמשים במפתח הצפנה אחד ויחיד הן להצפנה של הטקסט הקריא והן לפענוח של הטקסט המוצפן. בדרך כלל המפתח הוא משותף לשני האנשים או ליותר משתתפים ובדרך כלל מתאים לכמות מוגבלת של נתונים הסיבה שהצופן נקרא סימטרי היא כי נדרש ידע שווה של חומר סודי מפתח משני הצדדים וזהו החיסרון הגדול של הצפנה זאת.

הצפנה אסימטרית:

הצפנה זאת שונה מהצפנה סימטרית בניגוד להצפנה זאת בשביל להצפין ולפענח אנחנו צריכים שני מפתחות כמו שציינתי מפתח פרטי ופומבי כלומר ברגע שנקבל את המפתח הציבורי לא ניתן לפענח את המידע ללא המפתח הפרטי שקשה להשיגו הוא קיים רק אצל הבן אדם שקיבל את המסר. אלגוריתם שפועל בשיטה זו RSA וגם להצפנה הזאת כלומר לאסימטרית יש בעיית פרטיות man in the middle כמו שציינתי אבל היא פתירה באמצעות CA.

תהליך פתיחת חדר/הצטרפות לחדר קיים:

תהליך זה קורה כך כאשר משתמש מתחבר למערכת הוא יוכל לפתוח חדר שיחה חדר שכל אחד יכול לשוחח עם מי שהוא ירצה החדרים יהיו עם סיסמא כלומר על מנת להיכנס לחדר צריך סיסמא של החדר ברגע שהמשתמש יפתח את החדר ישלח לשרת שם החדר, סיסמא, שם המשתמש שפתח את החדר בצורה מוצפנת חשוב להזכיר שמשתמש לא יוכל ליצור חדר שכבר קיים במערכת.

ברגע שהמשתמש ירצה להיכנס לחדר קיים הוא יצטרך להזין סיסמא נכונה ושם הסיסמא נכונה יוכל להיכנס לחדר ולשוחח עם המשתמשים הנמצאים בחדר כעת.

כאשר משתמש יכנס לחדר הוא יקבל מפתח מהשרת באמצעות החלפת המפתחות בשיטת RSA.

תהליך החלפת המפתחות:

כאשר יש שני אנשים משוחחים בחדר תחילה לפני התקשורת בין שני האנשים מתבצע תהליך החלפת מפתחות זאת אומרת מפתח השיחה שבו מוצפנת השיחה כלומר המידע העובר בין שני המשתמשים צריך לעבור בצורה בטוחה מבלי שאף פורץ או תוקף יגלה מה המפתח.

החלפת המפתחות מתבצע באמצעות צופן אסימטרי כלומר לכל אחד יש:

- Public key – מפתח שמשמש להצפנה של ההודעה כולם יכולים לראות אותו

- Private key – מפתח שמשמש לפענוח של ההודעה אסור בתחילת האיסור לשתף מפתח זה הוא פרטי.

בנוסף להחלפת המפתחות אצור רשת אמון שכולם צריכים לסמוך עליה ולתת בה אמון כפי שמה רשת אמון וקוראים לה Certification Authority שתזהה כל אחד מהמשתמשים המחוברים למערכת. תהליך זה נוצר על מנת שנעביר את ה – key שבעזרתו מוצפן ההודעה שאנחנו מעבירים בשיטת AES באמצעות PRNG שפירושו Pseudo Random Number Generator פירושו בעברית מחולל מספריים אקראיים זאת פונקציה שיצרתי במהלך פרויקט בכיתה י'ג ואעשה בה שימוש.

תהליך הצפנת המידע ופענוח המידע:

המשתמשים יוכלו להעביר קבצים, טקסט, תמונות זהו המידע שהמשתמשים יוכלו להעביר.

המערכת שלי אמורה להפוך את המידע ללא קריא ולא ניתנת לפענוח כלומר הוא ניתן לפענוח אבל צריך מיליוני שנים (עם מחשבי – על או רגילים) על מנת לפענח את המידע אנחנו מניחים שהמידע לא ניתן לפענוח ככה זה כל הצפנה חזקה קיום גם השרת לא יוכל לדעת את תוכן המידע המועבר בין משתמשים והוא אחראי רק להעברה של מידע בין משתמשים הוא גשר בין המשתמשים.

באמצעות תהליך החלפת המפתחות הלקוח יוכל לפענח את המידע עם המפתח שניתן לו ובעזרת פונקציית PRNG לייצר את המפתח המלא כלומר עם ה- Seed אפשר לייצר מפתח יחיד לאותו Seed זה בעצם ה – PRNG.

תהליך סגירת חדר/התנתקות מחדר:

המערכת תנתק את הלקוח מהמערכת והוא לא יהיה בה חלק בהתאם המסד נתונים התעדכן לפי צרכי המערכת.

תהליך הרישום/התחברות:

ברגע שמשתמש משתמש בתוכנה שלי עליו להתחבר למערכת או להירשם עליה וזה עובד כך אם הלקוח לחץ על הירשמות עליו להזין את שם המשתמש, סיסמא, שם פרטי, שם משפחה ומייל ולאחר שנרשם למערכת בהצלחה ובהינתן שלא קיים שם משתמש שהוזן יחזור למסך ההתחברות ועליו להקליד את שם המשתמש והסיסמא שבו נרשם. המערכת תעשה בדיקה שהוא קיים במערכת אם הוא קיים הוא יכנס לתוכנה אם לא אז הסיסמא או שם המשתמש לא נכונים.

תיאור הבעיה האלגוריתמית:

בעולם של ימנו ישנו הרבה טכנולוגיה כגון סמארטפונים, מחשבים, אינטרנט ורשתות ובכל הטכנולוגיה הזאת קיימת מעבר של מידע שמתבצע ברשת.

כיום אנשים שולחים הודעות ברשתות החברתיות כגון Whatsapp משתפים מידע שיחות פרטיות, שיחות קבוצתיות, וידיאו, צילומי מסך ושליחת תמונות. אנחנו מגיעים למצב שהמידע הוא מאוד חשוב ומאוד פרטי למשל אם חס וחלילה אני ארצה לשלוח לחברתי את פרטי האשראי בשביל קנייה באינטרנט אני לא ארצה שאדם יקשיב לשיחתנו ויחלץ את ההודעות שלי ויראה את פרטי האשראי על מנת קנייה פרטית שלו שלא בכוונתי כיום זהו עברה פלילית לכל דבר, אנחנו חשופים לפגיעות ויש פגיעה בפרטיות, גניבה של פרטים אישיים וגם לחברות הענקיות יש את הפחד הזה שאנשים יוכלו לחלץ את המידע על משתמשי התוכנה שלהם ולכן פה מגיע הצפנה.

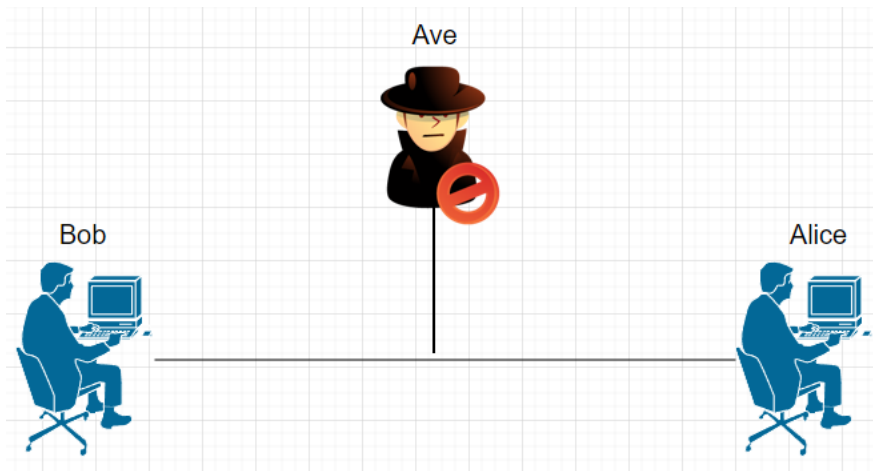
הצפנה חזקה תקרא אם יש תקשורת בין שני מחשבים וכאשר אדם נוסף מחלץ את ההודעות בין שיחתם ולא יצליח לפענח את ההודעות מכיוון שההודעות לא קריאות ההצפנה מטרתה לא לפגוע באבטחה של המערכת.

ההצפנה, כלומר אבטחת המידע אצלי בתוכנה מתבסס על ידי 2 פעולות מרכזיות וחשובות:

- החלפת המפתחות על מנת לשתף את המפתח של ה – AES נצטרך לשתף את המפתח הזה באמצעות אלגוריתם RSA שאותו הסביר בהמשך.
- הצפנת המידע עצמו כלומר תוכן ההודעה והצפנה זאת תתבסס על הצפנה AES שאותה הסביר בהמשך.

המפתח בתהליך החלפת המפתחות יהיה Seed מספר שבהימצאותו אני אצור את המפתח של ה – AES על ידי פונקציית PRNG.

בהמשך הצעת הפרויקט אעזר בדיאגרמת הזאת:



החלפת המפתחות:

כמו שציינתי מקודם על מנת להעביר את המפתח של ה-AES או את ה-Seed של יצירת המפתח של ה-AES אנחנו צריכים לשתף ביניהם מידע שאף איש לא יודע מהו כלומר סוג של סוד על גבי ערוץ שאינו מאובטח אותו ערוץ תקשורת בין שני אנשים כמו בוב ואליס.

סוד זה אני אצור על ידי שיטת RSA (Rivest Shamir Adleman) הסוד יהיה על מנת לתקשר בצורה מאובטחת ושמורה והיא עובדת כך:

RSA זאת שיטת הצפנה אסימטרית המשתמש במפתח ציבורי ומפתח פרטי על מנת להצפין את המידע ולפענח את המידע.

- Public Key – מפתח זה הוא ציבורי כולם רואים אותו, נועד להצפין את המידע העובר בין שני אנשים המתקשרים על גבי ערוץ תקשורת.
- Private Key – מפתח זה פרטי אסור לגלות מה הוא הוא סודי, נועד לפענח את המידע העובר בין שני אנשים המתקשרים על גבי ערוץ תקשורת.

הרעיון המרכזי הוא למשל שבו מצפין מידע עם המפתח הציבורי של אליס ואליס מפענחת את המידע אם המפתח הפרטי שלה!.

להלן האלגוריתם:

כדי להסביר את האלגוריתם אני אדגים עם מספרים קטנים אסור לעבוד איתם כי האלגוריתם יהיה פריץ אבל בפרויקט שלי אעבוד עם מספרים גדולים וענקיים.

Encryption: (5,14)

Text: B -> 2

$$2^5(\text{mod } 14) = 32(\text{mod } 14) = 4(\text{mod } 14)$$

Ciphertext: D

Decryption: (11,14)

$$4^{11}(\text{mod } 14) = 4,194,304$$

$$4,194,304 / 14 = 299,593.142857142857143$$

$$0.142857142857143 \times 14 = 2.0000000000000002$$

2 is the encrypted message.

אבל איך זה עובד?

שלב ראשון:

אלינו לבחור שני מספרים ראשוניים (מספר שמתחלק רק בעצמו ובאחד כלומר אין לו שני מספרים שמוכפלים שווה לו חוץ מאחד והמספר עצמו) חשוב לבחור במספרים ענקיים על מנת שהאבטחה תהיה מרבית ומקסימלית אבל שוב אם ההסבר היה עם מספרים גדולים היה קשה לעקוב אחרי האלגוריתם ולכן ההסבר הוא עם מספרים קטנים ולכן בחרתי 2 ו-7 ונסמן אותם כך:

$$p = 2, q = 7$$

שלב שני:

לאחר שבחרנו אני מספרים ראשוניים נבצע הכפלה בין שני המספרים ונסמן את המכפלה ב-N :

$$N = p \times q = 14$$

אנחנו עכשיו נשים לב למה מופיע 14 בהצפנה ובפענוח ולכן זהו המספר שאנחנו מפרסמים כלומר הוא חלק מה – public key – ו p ו- q לא נפרסם.

שלב שלישי:

נייצר פונקציית אוילר נסביר כעת מהי פונקציית אוילר זאת פונקציה של כמות המספרים שלא משותפים ל – 14 ולמכפלות שלו (2 ו-7) כלומר גורמים משותפים (חוץ מאחד כי הוא משותף לכל מספר).

להלן המספרים בין 1 – 14:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

המספרים באדום יורדים כלומר לא נתייחס אליהם והמספרים בצהוב נספור את הכמות כלומר התוצאה היא 6.

עכשיו יש בעיה איך נבצע סינון עבור מספר N ענקי אנחנו צריכים לבצע מ1 ועד המספר הענקי מי משותף ומי לא. אז לא נצטרך לעשות את העבודה הקשה הזאת ולטחון את ה – CPU יש דרך אחרת ואפילו מהירה בהרבה והיא שווה לזה:

$$\text{Phi function}(N) = (p - 1) \times (q - 1) = (2 - 1) \times (7 - 1) = 1 \times 6 = 6$$

שלב ארבע:

עכשיו אנחנו צריכים לבחור e אותו e הוא ה – 5 Encryption: (14, המספר הזה הוא לא רנדומלי ויש חוקיות והחוקיות עובדת כך:

- המספר חייב להיות בין 1 לפונקציית האוילר כלומר במקרה שלנו המספר 6. כלומר:
 $1 < e < 6$.
- אסור שהמספר בין $1 < e < 6$ יהיה משותף ל – 14 ולגורמים המשותפים לו כלומר (2 ו – 7) ואסור שיהיה משותף לפונקציית האוילר שלו כלומר במקרה שלנו 6.

ולכן לפי חוקים אלו לפי נתון ראשון והשני:

2,3,4,5 -> Coprime with phi function and N -> 5

2 ירד כי הוא משותף ל – 14, 3 ירד כי הוא משותף ל – 6 ו – 4 ירד כי הוא משותף ל – 2 ולכן נשארו רק עם 5.

עד עכשיו הבנו למה Encryption: (5,14) זהו המפתח הציבורי.

שלב חמש:

לאחר שלב ארבע נצטרך לבחור d שמקיים את התנאי הבא:

$$(e \times d) \pmod{\text{phi function}(N)} = 1$$

d	e x d	e x d(mod (phi function(N))
1	5	5
2	10	4
3	15	3
4	20	2

5	25	1
6	30	0
7	35	5

לא נבחר את 5 נבחר את 11 כי הוא מקיים את התנאי ועדיף לא לבחור את הראשון שמקיים את התנאי ולכן $d = 11$.

מפה הבנו למה decryption: (11,14) ולכן אנחנו מניחים שאם בוב רוצה לשלוח מידע לאליס עליו לבקש את המפתח הציבורי של אליס להצפין את המידע שרוצה להעביר לאליס ולאחר מכן ישגר עליה את המידע המוצפן אליס תיקח את המידע ותפענח אותו באמצעות המפתח הפרטי שלה.

עד עכשיו תיארתי והסברתי את אלגוריתם RSA אבל האם יש בעיה עם האלגוריתם? מבחינת אבטחה יש הגנה מלאה כאשר נעבוד עם מספרים גדולים אבל האם יש פרטיות, ישנו מקרה בפגיעה בפרטיות ואני אתאר אותו.

נניח שאליס רוצה לשלוח לבוב מידע אליס מייצרת את צמד המפתחות הציבורי והפרטי ושולחת את המפתח הציבורי לבוב מה מונע מאיב לבוא ולחלץ את המפתח הציבורי של אליס, ליצור מפתח ציבורי משלו ואז לשלוח את המפתח הציבורי שלו לבוב? לאחר מכן כאשר בוב שולח את המפתח הציבורי שלו בחזרה לאליס, איב יכולה ליירט את זה, לאחסן אותו לשלוח את המפתח הציבורי שלו לאליס.

כעת, כאשר אליס שולח מידע מוצפן באמצעות מה שהוא חושב שהוא המפתח הציבורי של בוב (אבל הוא למעשה של איב), איב חילץ את המידע הזה, תפענח אותה ואז יצפין אותה שוב באמצעות המפתח הציבורי האמיתי של בוב.

יש הקוראים לבעיה הזאת man in the middle היא מתוארת כך כאשר יש שני אנשים ויש תוקף הוא בעצם מתחזה כל אחד משני האנשים האלו חושבים שהם מדברים עם השני אבל זה בעצם לא קורה יש מישהו שמתערב ורואה את ההודעות.

זאת בעיה מורחבת אבל אפשרית וצריך לטפל בה הפתרון הוא שאליס צריך לדעת שהמפתח הציבורי שייך בוודאות לבוב ובוב צריך לדעת שהמפתח הציבורי של אליס שייך בוודאות לאליס ולזה קוראים CA היא רשות שכולם סומכים עליה שתפקידה להנפיק certificates למשתמשים במערכת, כאשר כל המשתמשים במערכת יודעים בוודאות את המפתח הפומבי של כל אחד מהמשתמשים האחרים במערכת.

כל המשתמשים מחויבים להאמין ולסמוך על ה-CA שלפני הנפקת ה- certificate הוא צריך לוודא שהזהות שמופיעה ב- Certificate שייכת לבעלים של זוג המפתחות הפומבי והפרטי, ה-CA נוצר כדי שנוכל בסופו של דבר לסמוך על מפתחות פומביים של אנשים אחרים.

התהליך הולך כך כאשר אליס רוצה לשלוח לבוב מידע אליס פונה ל-CA ה-CA בודק האם זה אליס באמת נותן לו את ה- Certificate של בוב המכיל את המפתח הציבורי של בוב ולאחר מכן שולחת את המידע המוצפן עם המפתח הציבורי של בוב לבוב. בוב מפענח עם המפתח הפרטי שלו.

בעזרת (PKI (public key infrastructure)) נעביר, נפרסם ונאמת את המפתחות הציבוריים.

העברת המידע בין הלקוחות:

המידע שעובר בין הלקוחות יוצן בהצפנת AES, שבעזרתו נשיג בטיחות מרבית ומקסימלית לתוכן העובר בין הלקוחות.

AES – Advanced Encryption Standard:

AES הוא צופן בלוקים סימטרי שאומץ על ידי המכון הלאומי לתקנים וטכנולוגיה של ארצות הברית כתקן הצפנה רשמי שהתקבל בעולם כולו, להצפנת נתונים ענקיים. שמו המקורי ריינדל והוא נמצא בשימוש מעשי נרחב בכל העולם הן בתוכנה והן בחומרה וידוע כאלגוריתם בטוח לחלוטין.

זהו צופן בלוקים איטרטיבי עם בלוק ומפתח משתנים בגודלם. ניתן להגדילם ללא תלות אחד בשני 128,192,256 סיביות. ב- AES גודל הבלוק הוא תמיד 128 הוא בטוח לחלוטין והוא יספק אותנו בפרויקט שלי.

ליבת הצופן מורכבת מסבבים שפועלים על מטריצה בגודל 4x4 או 16 בתים המהווה את מצב הביניים של הצופן ונקרת בקיצות המצב (state). מספר הסבבים תלוי באורך המפתח:

- 10 סבבים עבור מפתח של 128 סיביות.
- 12 סבבים עבור מפתח של 192 סיביות.
- 14 סבבים עבור מפתח של 256 סיביות.

בכל סבב מבצעים ארבע פונקציות שונות על ה- state והן ביניהן:

- SubByted
- ShiftRow

- MixColumn
- AddRoundKey

כאשר בסבב האחרון מורידים את הפעולה MixColumn.

SubBytes:

היא פונקציה החלפת בתים פשוטה הפועלת באופן סדרתי על כל בתי ה- state על פי טבלת החלפה קבועה והפיכה הנקראת Sbox ו- RsBox.

ShiftRow:

הזזת שורות בהזזה מעגלית לפי ערכים שונים. השורה הראשונה נותרת במקומה והשורה השנייה מוסטת ימינה פעם אחת והשורה השלישית מוסטת פעמים השורה הרביעית מוסטת שלושה פעמים ימינה.

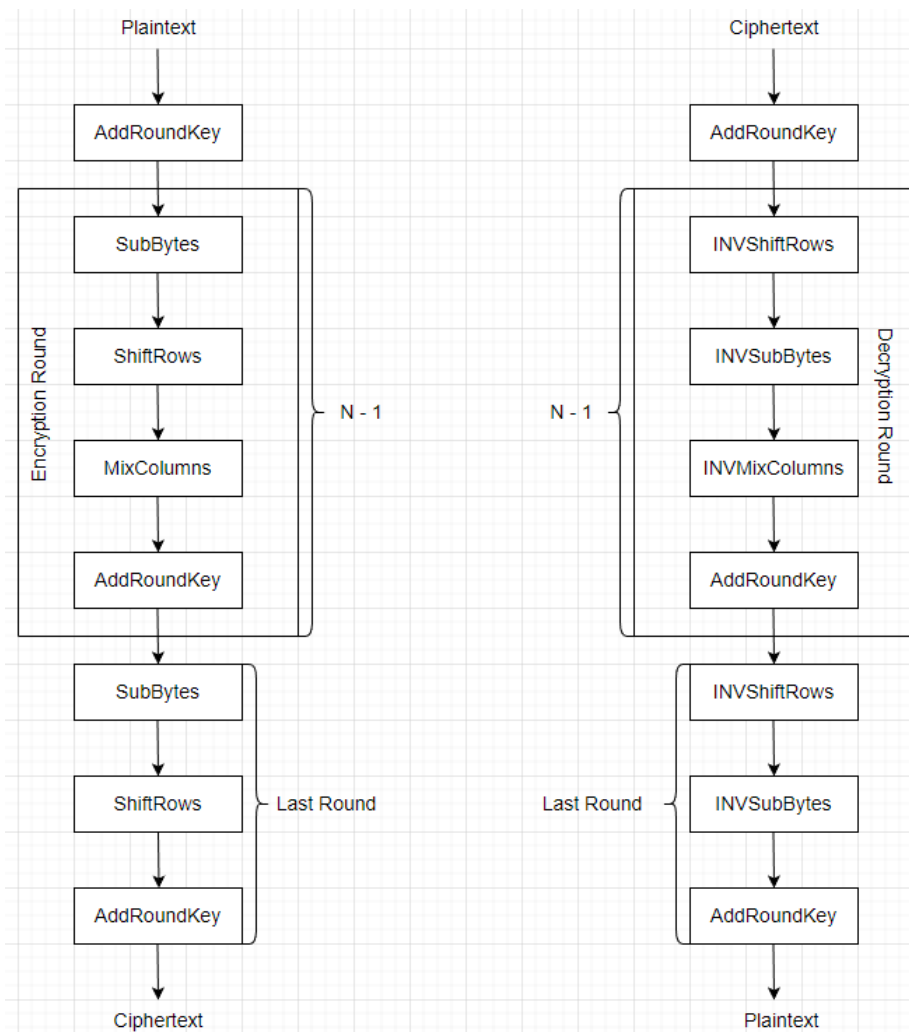
MixColumn:

פעולה זו פועלת על עמודות ב- State. אפשר לתאר פעולה זו ככפל מטריצה $b(x) = c(x) \times a(x)$, c מתארת את המטריצה הקבועה a העמודה הנוכחית ב- State.

AddRoundKey:

פעולה זאת פשוטה היא עושה XOR פשוט בית של המפתח עם הבית של המידע ומעדכן את התוצאה בבלוק.

את המפתח נפתח בעזרת פונקציית PRNG ונציב לו מספר בין 8 ביטים והוא ישגר לנו תוצאת של 16 בתים רנדומליים לחלוטין.



עד עכשיו תיארתי את שיטת ההצפנה שלי ואת המפתח אותו נשלח אותו למשתמשים האחרים באמצעות RSA שהסברתי.

מודל שרת לקוח:

מרבית יישומי האינטרנט בנויים לפי מודל שרת לקוח. ליישום כזה יש שני חלקים:

- יישום לקוח
- יישום שרת

שני חלקי היישום מתקשרים על ידי העברת מידע כפי שנקבע על ידי הפרוטוקול משתמשי הקצה מריצים כמעט תמיד את יישום הלקוח. יישום השרת מורץ באופן טיפוסי המחשב של ארגון נותן שירות.

צד השרת:

במקרה שלנו כפי שצינתי השרת הוא רק הגשר להעברת המידע בין לקוח ללקוח או בין לקוח לרוב לקוחות או לא יוכל לפענח את המידע העובר וגם לא יצליח. ובנוסף יש לו תפקידים

אחרים כמו הירשמות של משתמש ושמירת נתוניו בבסיס הנתונים וגם אם משתמש מצטרף לחדר שיחה השרת שולח את המפתח הפומבי שלו לאחרים להתחלת שיחה כלומר עושה תהליך החלפת מפתחות.

השרת הולך להכתב בשפת python בסביבת עבודה Pycharm.

צד הלקוח:

הלקוח הוא הליבה של הפרויקט הוא המרכזי והחשוב ביותר בפרויקט שלי.

הלקוח בוחר לאיזה חדר להצטרף וברגע שיש יותר מלקוח אחד בחדר מתבצע החלפת המפתחות וברגע זה הכול מוצפן מאובטח ופרטי. הליך זה לא פוגע בהצטרפות של לקוח חדש, כלומר במידה שלקוח אחד מן השניים מתנתק ולקוח חדש מתחבר במקומו ההחלפה תתרחש מחדש וכעת המידע יהיה מוצפן במפתח אחר. ואיך זה מתבצע ברגע שלקוח מתנתק השרת מודיע לכלל הלקוחות שלקוח התנתק.

הלקוח יכתב בשפת Python בסביבת עבודה Pycharm.

תיאור הטכנולוגיה:

סביבת עבודה ושפות תכנות:

שפת התכנות שבה השתמש היא python.

Python היא שפה קלילה ומאתגרת אותי מכיוון שלא פיתחתי פרויקט ברמה כזאת ב – python ואשמח לעשות זאת שפה זאת מציעה הרבה כלים ואפשרויות ויש לה פונקציונליות רבה בנוסף אין לי ממש ידע עמוק עליה אני יותר איש של Java , C , C# ולכן בחרתי שפה זאת.

את הכתיבה אממש בסביבת העבודה Pycharm היא תוכנה חינמית וממש ידידותית למשתמש ויש לה ממשק גרפי מצוין למשתמש היא ברורה ולא קשה והכי חשוב נותן השלמת קוד ומזהה שגיאות.

מסד הנתונים Mysql:

מסד הנתונים יהיה בתוכנת Mysql כי אני מכיר אותה משנה שעברה עבדתי איתה במהלך לימודי ביג, בנוסף השתמש בספריית 3sqlite ב – python שהיא מאפשרת לכתוב פקודות SQL בפייתון דבר שיכול לייעל לנו את העבודה.

פרוטוקולי תקשורת:

- AES הצפנה
- אלגוריתם החלפת המפתחות RSA
- TCP (העברה נכונה של מידע בשרת)

מסד הנתונים:

מסד הנתונים מציין את הפרטים של המשתמשים כל המשתמשים בתוכנה שלי. אני אשתמש בשני טבלאות עיקריות אחד המכיל את פרטי המשתמשים והשני את פרטי השיחות של המשתמשים בתוכנה.

הפרטים של המשתמשים מאוד חשובים ופרטיים והכל קורה בזמן ההתחברות או ההירשמות ברגע שלקוח נרשם למערכת שלי הוא מזין את הפרטים שלו וברגע שלחץ על הירשמות הפרטים נשלחים מוצפנים לשרת באמצעות שימוש בפונקציית hash הצפנה חד-כיוונית כלומר לא ניתן לפענח אותן.

כמו שצינתי אני אשתמש בתוכנת Mysql ובמסד הנתונים שלי יש טבלה אחת שמכילה:

- שם משתמש
- שם פרטי
- שם משפחה
- סיסמא
- מייל

וטבלה שנייה המכילה את הפרטים של החדר והם יהיו:

- שם החדר
- סיסמא של החדר
- שם המשתמש שיצר את החדר

הערה חשוב יכולה להיות שיהיו עוד טבלאות אבל אלו בינתיים הטבלאות המרכזיות.

להלן הטבלאות:

Users:

<u>Username:</u>	Password:	Firstname:	Lastname:	Mail:
Lidor_duk	F1b0c4a1	lidor	dukarker	lidorduk@gmail.com

Rooms:

<u>Roomname:</u>	Password:	Usernameowner:
lidorRoom	F53d294dla1	Lidor_duk

לוחות זמנים:

תאריך סיום השלב	שלב עבודה
1.12.2022	חקירה ולמידה אודות הנושא לעומק
31.12.2022	שליחת הצעת פרויקט
10.01.2023	התחלת עבודה על צד הלקוח
15.01.2023	התחלת עבודה על צד השרת
01.03.2023	שילוב מסד הנתונים
15.03.2023	סיום עבודה על התקשורת בין השרת והלקוח
20.03.2023	שילוב ההצפנה
30.03.2023	הכנה להגשה והכנת ספר פרויקט
10.04.2023	הכנת המערכת להגשה - סיום

חתימה הסטודנט: _____

אישור משרד החינוך:

מבוא:

הרקע לפרויקט:

במהלך לימודי כהנדסאי תוכנה התבקשנו לבחור פרויקט פרויקט גמר, היה ידוע לי שאני הולך לכוון לפרויקט העוסק בעולם הקריפטוגרפיה הכולל בעיות מתמטיות חשיבתיות, רציתי שהפרויקט שלי יכלול אתגרים ומידע שעסקתי בעבר ומידע ואתגרים שלא עסקתי בעבר ולהיכנס לעומק יותר בארגון המידע בין שני לקוחות ויותר המתקשרים עם השרת, הקריפטוגרפיה, מסד הנתונים והממשק משתמש הגרפי המתוכנת בשפת Python.

את הנושא עזר לי לבחור המנחה גל בראון שבין היתר מכיל מערכת קריפטוגרפית בטוחה ומתוחכמת, כלומר מערכת המאפשרת שיחות בין לקוחות מבלי אפשרות הבנת המידע דרך האזנה של המידע העובר בין הלקוחות ברשת במימוש צא'ט.

בחרתי בפרויקט זה מכיוון שהוא קשור לכל התחומים שעסקתי בהם במהלך לימודי בשנים האחרונות בלימודי הנדסת תוכנה, פרויקט זה משלב שני צדדים עיקריים צד שרת ולקוח,

גרפיקה למשתמש על מנת להקל על המשתמש, סייבר, משלב אבטחת מידע ושימוש באלגוריתמים מתחום הקריפטוגרפיה ושלמות העברת הנתונים מלקוח ללקוח.

במהלך התעסקות של הפרויקט נחשפתי למידע של למדתי ולא עסקתי ובכך העשרתי את הידע שלי וקידמתי את הפרויקט שלי בהתאם, העשרתי את הידע בעולם הסייבר, הגרפיקה העיצוב והקריפטוגרפיה.

תהליך המחקר:

במהלך עבודתי על הפרויקט חיפשתי מקורות מידע שאפשר לסמוך עליהם אשר אמורים לעזור לי לקידום הפרויקט, מצאתי מאמרים וספרים העוסקים בקריפטוגרפיה וסייבר גרפיקה עיצוב ועוד, כמו כן גלשתי רבות ברחבי האינטרנט על מנת להבין את האלגוריתמים השונים ומימושם.

בנוסף, המחנה גל בראון עזר לי רבות וסייע לי במידע הכלול בפרויקט שי ושלח לי מאמרים, סרטונים העוסקים בהסברה של אלגוריתמים ומימושם על מנת להעשיר את הידע שלי ושיפור הפרויקט על מנת להגיע לפרויקט גמר הטוב ביותר.

בנוסף במהלך עבודתי על הפרויקט עבדתי בשיטת הצפנה AES וממשתי אותה אבל אם הזמן החלטתי לשנות את שיטת ההצפנה ל – OTP אבל הרעיון האלגוריתמי נשאר אותו הדבר. (OTP מוסבר בהמשך)

אתגרים מרכזיים:

במהלך עבודתי על הפרויקט נתקלתי במספר בעיות וקשיים, הידע שלי בקריפטוגרפיה היה ברמה בינונית, לפני הפרויקט היה לי ידע על הצפנות סימטריות אבל הצפנות אסימטריות היה עולם חדש לגמרי לקח לי זמן להבין איך ההצפנה האסימטרית עובדת לקחתי את הנושא והתחלתי לחקור לבדי ובסופו של דבר התמודדתי עם קושי זה על ידי חקירה מעמיקה ברחבי האינטרנט וצפייה בסרטונים הסברה ומימוש ביוטיוב.

קושי נוסף שהקשה עליי הוא המאמרים והמידע ברחבי האינטרנט על הנושאים הנוגעים על הפרויקט שכולם בשפה האנגלית לא שפת האם שלי אך לאט הבנתי את המידע על ידי למידה איטית וזהירה מה שתרם לי לשיפור השפה האנגלית.

עם למידת המנגנון של TCP שהיא פרוטוקול תקשורת שעליה מושתת רשת האינטרנט, ושם למודל שכבתי המתאר תקשורת ברשתות המחשבים גילתי שהפרוטוקול זה אינו יכול

להבטיח שכל המידע שעובר ב – data link layer יגיע במלואו כלומר ייתכן שחלק מהמידע לא הגיעו בשלמותן וצריך לטפל במצב זה.

בסופו של דבר רוב העזרה קיבלתי מהמנחה גל בראון ששלח לי והכווין אותי למאמרים, קטעי קוד, מקורות מידע ובכך התקדמתי רבות בפרויקט ופתרתי בעיות וקשיים.

במהלך לימודי בתיכון למדתי עסקתי בפרויקט גמר שפיתחתי בכיתה י"ב שבו עסקתי בעולם הקריפטוגרפיה כבר שם ידעתי שעולם זה מעניין אותי כי הוא משלב ידע רב במתמטיקה מקצוע שבו אני מתמקצע ואוהב כשנודע לי על פרויקט גמר בכיתה י"ג ידעתי שאני ימשיך והעשיר את הידע שלי בעולם הקריפטוגרפיה ואעמיק בפרויקט מכיוון שאני אוהב מתמטיקה ובעיות חשיבה מאתגרות.

הפרויקט שבחרתי זה נושא אהוב עליי ומתחבר אליו ובנוסף שפרויקט זה מתקשר לנושא שמאוד מבוקש בתעשיית ההייטק אבטחת מידע, קריפטוגרפיה וסייבר מה שגרם לי עוד יותר להישאב לתוך הפרויקט להמשיך ולשפר אותו לפרויקט הטוב ביותר גם אחרי ההגשה ולהרחיב אותו כמה שאפשר.

הפרויקט עונה על צרכי המערכת, חשוב להזכיר שחלק צורכי המערכת היא תקשורת מקומית לצורך שיחות קבוצתיות ופרטיות, שליחת קבצים ועוד.

מטרות/יעדים:

מטרות הפרויקט – המטרה הראשית של הפרויקט שלי הוא להוות מערכת בטוחה ביותר מערכת בתפעל בזמן אמת ותדע בזמן אמת כאשר לקוחות מתחברים למערכת ותוכל לאתר אותם, להפיק מפתחות להצפנות בצורה בטוחה ומתחכמת והכי חשוב מהירה ככל האפשר, ובנוסף שתאפשר שיחות פרטיות בין משתמשים, ושמירת שיחות חשודות כגון אלימות מילולית, זיהוי האם הקובץ הנשלח הגיע במלואו, ניהול בסיס הנתונים, גריפה פשוטה וקלה למשתמש.

מטרות צד השרת: להיות מרכז בקרה ושליטה של המערכת שמממשת הרבה פעולות שאפשר לעשות במערכת כמו שליחת קבצים ושיחות פרטיות.

מטרות צד הלקוח: להיות קל וברור למשתמש, נוחה, חכמה לממש את כל ההצפנות והאלגוריתמים, יצירת המפתחות והכי חשוב להיות מהירה ככל האפשר.

המטרה העיקרית והכי חשוב עמידה בזמנים ובלוח הזמנים הקיים. עוד מטרה חשובה היא פיתוח הפרויקט ולהרחיב את הפרויקט ואת הידע בתחומי הפרויקט.

מדדי הצלחה למערכת:

למערכת שלי ישנו את מלוא אחוזי ההצלחה 100% היא יכולה לקבל לקוחות ותמיד לבצע את אלגוריתם RSA החלפת מפתחות. היא תמיד מתעדת ושומרת את כל המידע העובר בין ברשת על מנת לייעל את המערכת, ואין מצב שבו המערכת מנסה לבצע פעולה של אילוצי המערכת ולא מצליחה.

רקע תיאורטי:

קריפטוגרפיה:

קריפטוגרפיה היא ענף הקשור למתמטיקה ומדעי המחשב העוסק במחקר ופיתוח שיטות אבטחת מידע ותקשורת נתונים ובנוסף מייעל את האבטחה והתקשורת בין 2 צדדים לפחות, כל מערכת כיום שרוצה שלא יהיה חדירה לפרטיות חייבת לארגן מערכת קריפטוגרפיה טובה וחזקה שיאבטחו את המערכת הכללית והנתונים שלה.

הצפנה סימטרית:

רוב העולם הטכנולוגי מנוהל בצורה מוצפנת הפירוש של המילה הצפנה היא הסתרה החבאת מידע שאנחנו רוצים לשתף לחבר או לקבוצת חברים שאני לא רוצה שגוף זר צד שלישי יוכל לחלץ את המידע ולקרוא את ההודעה. בשביל זה נברא הצפנות רוב ההצפנות כיום הם הצפנות סימטריות מה זאת אומרת למשל יש לי מידע מסוים את המידע אני מצפין עם מפתח אחד ויחיד שאותו מפתח יכול גם לפענח את המידע המוצפן באמצעות פעולות הפוכות אחד האלגוריתמים לפעולות הפוכות הוא AES הפעולה ההפוכה של OTP היא פשוט לעשות פעולת XOR עם הצופן המוצפן ביחד עם ה – Pad.

הצפנה אסימטרית:

הצפנה אסימטרית היא הצפנה שונה מהצפנה סימטרית בניגוד להצפנה סימטרית בהצפנה אסימטרית יש לנו בכל צד במערכת זוג מפתחות הקרואים בשמות:

- מפתח ציבורי
- מפתח פרטי

את המפתח הציבורי אנחנו שולחים לצד שרוצה לשלוח לנו הודעה אותו צד המקבל של המפתח הציבורי ייקח את המפתח הציבורי ויצפין את ההודעה שהוא רוצה לשלוח לנו חשוב להזכיר ההצפנה האסימטרית בפרויקט שלנו נועדה להעברת הסוד. הסוד הוא ה – SEED שאותו נעביר על מנת להצפין את ההודעות בין השרת ללקוח.

המפתח הפרטי נועד לפענח את ההודעה ששלחו לנו חשוב להזכיר את המפתח הפרטי אף אחד לא יודע עליו ואסור לשתפו הוא פרטי.

האלגוריתמים שפועלים בשיטה זו הם RSA ו – Diffie Hellman.

גרפיקה GUI:

על מנת להפוך את התוכנה ואת המערכת עצמה לנוחה יותר לשימוש בעבור המשתמש אנחנו חייבים לממש GUI נוח. על מנת לבנות את ה – GUI יש ספרייה מוכנה ב – python ששמה tkinter שהיא נוחה וטובה לעיצוב ולגרפיקה, הידע שלי יותר רחב ב – Java swing משנה שעברה ואליי ללמוד עיצוב גרפי בפיתוח. הגרפיקה בפרויקט לא מסובכת ואפילו קלה להבנה כי הפרויקט שלי עוסק יותר בתורת הקריפטוגרפיה ולא בעיצוב אך אני אנסה לייעל את הממשק שהוא יהיה נוח וטוב.

ניתוח דרישות המערכת:

הפרויקט שלי מחולק לשני חלקים שונים, חלק אחד הוא השרת יחידת בקרה הוא רץ עצמאית כלומר אינו צריך input מהמשתמש השרת יושב במחשב ברשת המקומית, החלק השני הוא הלקוח שהוא לא עובד עצמאית וכן צריך input מהמשתמש חשוב להזכיר את הדגשים יכולים להיות כמה לקוחות וחשוב להזכיר שיש לוודא שהשרת מחובר כראוי מכיוון שהלקוח לא יצליח להתחבר לשרת עם הוא לא מחובר ודבר אחרון יש לשנות את כתובת ה – ip לכתובת של השרת על מנת שהפרויקט ירוץ.

כמו כן לפני שמריצים צריך לציין שכל החבילות של ה – python כלומר imports צריכים להיות קיימים ומורדים במחשב שבו מורץ הפרויקט.

אילוצים:

למערכת שלי יש שני אילוצים עיקריים והם:

- למערכת יכולים להתחבר עד 10 אנשים.
- המערכת עובדת רק ברשת המקומית.
- אינה תומכת בקבצים גדולים.

תיאור הרכיבים בפתרון:

עשיתי שימוש בתוכנת MySQL מכיוון שבתוכנה זאת השתמשי שנה שעברה ואני מכיר בקיע בתוכנה.

הטבלאות שיש לי בפרויקט הן:

- Client_user
- Client_password_key
- Client_seeds
- Client
- Sus_msg_talking_room
- Sus_msg_work_room
- Sus_msg_dating_room

כל אחת מהטבלאות אני אסביר ואנמק למה עשיתי בה שימוש.

טבלת client_user:

שומרת את כל הלקוחות שנרשמו למערכת בעבר ובהווה. לפי user_name שהוא המפתח הראשי בטבלה יחד עם הרשמות הבאות first_name המייצג את שמו הפרטי של המשתמש , last_name המייצג את שמו המשפחה של הלקוח והכי חשוב הסיסמא של הלקוח שמוצפנת בהצפנה סימטרית שאת המפתח לפיענוח הסיסמא שמורה בטבלה אחרת.

טבלת client_password_key:

טבלה זאת נועדה לשמירת המפתח הסימטרי המצפין את הסיסמא של הלקוח במערכת והוא נשמר לפי user_name שהוא המפתח של הטבלה ושיוכו של key לאותו משתמש עם אותו מפתח, עם אותו מפתח גם מפענחים את הסיסמא שבטבלה השייך לאותו משתמש ומשווים אותו לסיסמא של הלקוח המתחבר אם הם שווים אז הלקוח יכנס למערכת אם לא אז לא.

טבלת client_seeds:

טבלה זאת שומרת את הלקוחות המחוברים כעת למערכת אם היא ריקה אז אף לקוח מחובר כעת למערכת.

היא שומרת את ה – client_socket שהוא המפתח הראשי של הטבלה בין היתר היא שומרת את Public_key שהוא המפתח הציבורי של אותו לקוח שהתחבר ואת ה – modulo_pq שני רכיבים אלה קשורים להחלפת המפתחות הממומשות באלגוריתם RSA ואת ה – setseed_pad שהוא אובייקט שנוצר בעזרת ה – seed שהועבר מהשרת ללקוח ובעזרתו יוצרים את ה – pads להצפנת פנקס חד פעמי.

טבלת client:

טבלה זאת שומרת את כל הלקוחות המחוברים כעת במערכת כלומר אם היא ריקה אז אף לקוח לא מחובר למערכת. היא שומרת את ה – client_socket שהוא המפתח הראשי ביחד עם user_name טבלה זאת היא טבלה מקשרת כלומר טבלת קשר בין client_seeds ו client_user. אחד ההיתרונות של הטבלה הזאת היא מניעה של התחברות כפולה של משתמש כלומר עם משתמש התחבר למערכת או לא יוכל להתחבר שוב למערכת עם משתמש זה.

טבלת sus_msg_talking_room:

טבלה זאת שומרת את ההודעות החשודות כלומר הודעות שיש בהן אלימות מילולית ומילים בוטות שנשלחו בחדר talking.

השרת מחלץ את ההודעות החשודות בכך שאם לקוח שלח בחדר talking מילה בוטה הוא שומר את שם המשתמש ששלח את ההודעה ואת ההודעה החשודה ובנוסף את זמן שליחת ההודעה החשודה.

טבלת sus msg dating room

טבלה זאת שומרת את ההודעות החשודות כלומר הודעות שיש בהן אלימות מילולית ומילים בוטות שנשלחו בחדר dating.

השרת מחלץ את ההודעות החשודות בכך שאם לקוח שלח בחדר dating מילה בוטה הוא שומר את שם המשתמש ששלח את ההודעה ואת ההודעה החשודה ובנוסף את זמן שליחת ההודעה החשודה.

טבלת sus msg work room

טבלה זאת שומרת את ההודעות החשודות כלומר הודעות שיש בהן אלימות מילולית ומילים בוטות שנשלחו בחדר work.

השרת מחלץ את ההודעות החשודות בכך שאם לקוח שלח בחדר work מילה בוטה הוא שומר את שם המשתמש ששלח את ההודעה ואת ההודעה החשודה ובנוסף את זמן שליחת ההודעה החשודה.

פונקציה start_func:

```
def start_func(login_or_create):
    global private_flag
    global data_list_for_private
    global user_pad
    global s
    global seed_server

    # step 1- client connect to server
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('192.168.127.57', 1235))
    print(s.getsockname())
    # Key exchange
    seed_a_b = key_exchange(s, random.choice(primes), random.choice(primes),
                            random.randint(10, 100))
    seed_server = seed_a_b[0]
    user_pad = SetSeed(seed_a_b[0], seed_a_b[1], seed_a_b[2]) # obj that will help to generate pads for the server

    enc_login_or_not = my_functions.recvall(s)
    login_or_not = decrypt_cipher(enc_login_or_not, user_pad)
    print(login_or_not)
    client_ans = login_or_create # what the user click on
    s.send(encrypt_msg(client_ans, user_pad))

    create_account = decrypt_cipher(my_functions.recvall(s), user_pad)
    print(create_account)
```

מקבלת את תגובתו של הלקוח אם ברצונו להתחבר למערכת או להירשם, פותחת socket ומטפלת בתהליך החלפת המפתחות.

פונקציה login_func:

```
def login_func(username, password):
    global user_name
    # step 3 - send to server the username
    s.send(encrypt_msg(username, user_pad))
    s.send(encrypt_msg(password, user_pad))

    try_to_login = decrypt_cipher(my_functions.recvall(s), user_pad)
    print(try_to_login)
    if try_to_login == 'success to connect':
        user_name = username
        print(decrypt_cipher(my_functions.recvall(s), user_pad)) # hello
        return True
    else:
        return False
```

מקבלת את שם המשתמש והסיסמא של הלקוח הרוצה להתחבר למערכת ושולחת לשרת שיאמת שקיים שם משתמש וסיסמא שהזין במערכת ומחזיר תשובה בהתאם לבדיקות.

פונקציה create_account func:

```
def create_account_func(first_name, last_name, username, password):
    s.send(encrypt_msg(first_name, user_pad))
    if first_name == '0':
        return '0'
    time.sleep(0.01)
    s.send(encrypt_msg(last_name, user_pad))
    if last_name == '0':
        return '0'
    time.sleep(0.01)
    s.send(encrypt_msg(username, user_pad))
    if username == '0':
        return '0'
    time.sleep(0.01)
    s.send(encrypt_msg(password, user_pad))
    if password == '0':
        return '0'

    server_ans = decrypt_cipher(my_functions.recvall(s), user_pad)
    print(server_ans)
    if server_ans == 'details saved successfully':
        return '1'
    else:
        return '2'
```

מקבלת את פרטיו של הלקוח שהקליד בחלון ושולחת אחד אחרי השני לשרת, השרת מטפל ברישומם אם הרישום בוצעה בהצלחה או לא ומחזיר תשובה ללקוח בהתאם.

פונקציה select_room func:

```
def select_room_func(room):
    enc_which_room = my_functions.recvall(s)
    which_room = decrypt_cipher(enc_which_room, user_pad)
    print(which_room)
    print('press 1 to exit')

    print("You are connected to the server")

    enc_msg_to_send = encrypt_msg(room, user_pad)
    s.send(enc_msg_to_send)

    return user_name
```

מקבלת את החדר הנבחר על ידי המשתמש בממשק ושולחת לשרת את החדר הנבחר.

פונקציה gui func chat server ongoing msg:recv

```
def receive_ongoing_msg_from_chat_server_func_gui(s, pad, text_box):
    global private_flag
    while True:
        if not private_flag:
            enc_msg_from_server = my_functions.recvall(s)
            msg_from_server = decrypt_cipher(enc_msg_from_server, pad) # msg from the server

            if msg_from_server == 'kill yourself': # exit
                s.close()
                break

            if msg_from_server == 'get file photo': # download a file
                enc_file_name_from_server = my_functions.recvall(s)
                file_name_from_server = decrypt_cipher(enc_file_name_from_server, pad)
                f = open(f'{user_name}_from_{file_name_from_server}', "wb")
                enc_block_file = my_functions.recvall(s)
                block_file = decrypt_cipher_file(enc_block_file, pad)
                digest_maker = hmac.new(str(seed_server).encode(), ''.encode(), hashlib.sha256)
                while block_file != b'0':
                    f.write(block_file)
                    # print(block_file)
                    # print(block_file.encode())
                    digest_maker.update(block_file)

                    enc_block_file = my_functions.recvall(s)
                    block_file = decrypt_cipher_file(enc_block_file, pad)
                    # print(block_file)
                    # print(block_file.encode())
                f.close()

                rec_digest = digest_maker.hexdigest()
                print(f'hmac for the content in the file that sent {rec_digest}')
                enc_real_digest = my_functions.recvall(s)
                real_digest = decrypt_cipher(enc_real_digest, pad)
                print(f'hmac for the real file {real_digest}')

                if real_digest == rec_digest:
                    text_box.configure(state=NORMAL)
```

```

text_box.insert(END, f'{file_name_from_server} file successfully downloaded')
text_box.insert(END, '\n')
text_box.configure(state=DISABLED)
text_box.see(END)
continue
else:
    text_box.configure(state=NORMAL)
    text_box.insert(END, f'{file_name_from_server} file is distorted')
    text_box.insert(END, '\n')
    text_box.configure(state=DISABLED)
    text_box.see(END)
    continue

if msg_from_server == 'start camera':
    enc_from_who = my_functions.recvall(s)
    from_who = decrypt_cipher(enc_from_who, pad)

    data = b""
    payload_size = struct.calcsize("Q")
    while True:
        while len(data) < payload_size:
            packet = my_functions.recvall(s)
            #msg = decrypt_cipher_file(packet, user_pad)
            if not packet:
                break
            data += packet
        packed_msg_size = data[:payload_size]
        data = data[payload_size:]
        msg_size = struct.unpack("Q", packed_msg_size)[0]

        while len(data) < msg_size:
            data += my_functions.recvall(s)
            #msg = decrypt_cipher_file(data, user_pad)
        frame_data = data[:msg_size]
        data = data[msg_size:]
        frame = pickle.loads(frame_data)
        cv2.imshow(f'{from_who} VIDEO', frame)
        key = cv2.waitKey(1) & 0xFF

```

```

        if key == 20:
            break
cv2.destroyAllWindows()

if msg_from_server == 'creator_rsa': # the client that wants to start private session, create the keys
    key_exchange_for_private(s, random.choice(primes), random.choice(primes),
                             random.randint(10, 100))
    private_flag = True

if msg_from_server == 'second_participant': # the second client in the private session create the key, encrypt it and send back
    public_key = int(my_functions.recvall_with_decode(s))
    print(public_key)
    qp = int(my_functions.recvall_with_decode(s))
    print(qp)
    data_list_for_private.append(qp)
    data_list_for_private.append(public_key)
    seed = random.randint(0, 500)
    print(f'seed_for_private2 {seed}')
    data_list_for_private.append(SetSeed(seed, 8888, 9999))
    enc_seed = my_functions.rsa_encryption_decryption(qp, public_key, seed)
    print(f'enc seed_for_private2 {enc_seed}')
    s.send(str(enc_seed).encode())
    private_flag = True

print(msg_from_server)

text_box.configure(state=NORMAL)
text_box.insert(END, msg_from_server)
text_box.insert(END, '\n')
text_box.configure(state=DISABLED)
text_box.see(END)
else: # the client in a private session
    enc_private_msg_from_server = my_functions.recvall(s)
    if enc_private_msg_from_server == '1'.encode(): # exit from the private session
        s.send('stam'.encode())
        private_flag = False
        data_list_for_private.clear()

```

```

print('exit from private session')
text_box.configure(state=NORMAL)
text_box.insert(END, 'exit from private session')
text_box.insert(END, '\n')
text_box.configure(state=DISABLED)
text_box.see(END)
else:
    if private_flag:
        private_msg_from_server = decrypt_cipher(enc_private_msg_from_server, data_list_for_private[-1])

        print(private_msg_from_server)

        text_box.configure(state=NORMAL)
        text_box.insert(END, private_msg_from_server)
        text_box.insert(END, '\n')
        text_box.configure(state=DISABLED)
        text_box.see(END)
exit()

```

פונקציה זאת עובדת ב-thread אחר על מנת לקבל בזמן אמת פקודות מהשרת כמו למשל לפתוח מצלמה או יצירת מפתחות לשיחה פרטית או קבלת קובץ ועד.


```

def session_func(user_msg, textbox):
    global private_flag
    global data_list_for_private
    global hmac_flag_txt
    global hmac_flag_photo
    global hack_flag

    # step 6- client send some msg to server (get the msg as input from the user)
    if private_flag:      # the client in a private session
        if user_msg == '1':
            print('exit from private session')
            textbox.configure(state=NORMAL)
            textbox.insert(END, 'exit from private session')
            textbox.insert(END, '\n')
            textbox.configure(state=DISABLED)
            textbox.see(END)
            private_flag = False
            s.send(user_msg.encode())
            data_list_for_private.clear()
            return 'continue'
        else:
            enc_private_msg_to_send = encrypt_msg(user_msg, data_list_for_private[-1])
            s.send(enc_private_msg_to_send)
            return 'continue'
    else:
        if user_msg == '1' or user_msg == '***delete_account***': # exit
            enc_msg_to_send = encrypt_msg(user_msg, user_pad)
            s.send(enc_msg_to_send)
            return 'exit'

    # FILES
    if hmac_flag_photo: # send file
        try:
            f = open(user_msg)
            f.close()
        except IOError:
            print("File not accessible")
            enc_file_name_to_send = encrypt_msg('File not accessible', user_pad)
            s.send(enc_file_name_to_send)

```

```

        hmac_flag_photo = False
        return 'File not accessible'

enc_file_name_to_send = encrypt_msg(user_msg, user_pad)
s.send(enc_file_name_to_send)

file_name_hmac = hmac.new(str(seed_server).encode(), user_msg.encode(), hashlib.sha256).hexdigest()
enc_file_name_hmac_to_send = encrypt_msg(file_name_hmac, user_pad)
s.send(enc_file_name_hmac_to_send)

digest_maker = hmac.new(str(seed_server).encode(), ''.encode(), hashlib.sha256)
try:
    f = open(user_msg, 'rb')
    try:
        block = f.read(1024)
        while block:
            # print('*')
            digest_maker.update(block)
            print('*')
            print(block)
            enc_block_to_send = encrypt_file(block, user_pad)
            print('**')
            print(enc_block_to_send)
            if hack_flag: # פורץ משבש את תוכן הקובץ
                s.send(len(enc_block_to_send) * b'0')
            else:
                s.send(enc_block_to_send)
            # print(block)
            # print(block.decode()) # block
            # print('***')

            block = f.read(1024)
    finally:
        f.close()
        hack_flag = False
    time.sleep(0.2)
    enc_block_to_send = encrypt_file(b'0', user_pad)
    s.send(enc_block_to_send)

```

```

        time.sleep(0.05)

        digest = digest_maker.hexdigest()
        print(f'hmac for the real file {digest}')

        enc_digest_to_send = encrypt_msg(digest, user_pad)
        s.send(enc_digest_to_send)

    except:
        pass
    hmac_flag_photo = False
    return 'sent'

if user_msg == 'FILE':
    hmac_flag_photo = True

if user_msg == 'HACK':    # disrupt the information
    hack_flag = True
    return

# webcam share
if user_msg == 'WEBCAM':
    c = 0
    enc_msg_to_send = encrypt_msg(user_msg, user_pad)
    s.send(enc_msg_to_send)
    time.sleep(0.01)
    vid = cv2.VideoCapture(0)
    while vid.isOpened():
        c += 1
        img, frame = vid.read()
        frame = imutils.resize(frame, width=320)
        a = pickle.dumps(frame)
        message = struct.pack("Q", len(a)) + a
        # enc_msg = encrypt_file(message, user_pad)
        s.send(message)
        cv2.imshow('MY VIDEO', frame)
        key = cv2.waitKey(1) & 0xFF
        print(c)

```

```

        if c < 600:
            cv2.destroyAllWindows()
            break
        return
    enc_msg_to_send = encrypt_msg(user_msg, user_pad)
    # print(enc_msg_to_send)
    # print(type(enc_msg_to_send))
    s.send(enc_msg_to_send)
    return 'continue'
)

```

מקבלת את ההודעה של הלקוח ומטפלת בה.

שרתי תקשורת:

בפרויקט שלי אני משתמש בפרוטוקול TCP ברשת מקומית (אילוצים תקציבים) אסביר בקצרה על פרוטוקול זה.

TCP/IP הוא חבילה של פרוטוקולי תקשורת המשמשים לחיבור התקני רשת באינטרנט. זהו הבסיס של האינטרנט ומאפשר העברת נתונים בין מכשירים, בין אם הם ממוקמים זה ליד זה או בצדדים מנוגדים של העולם.

TCP אחראית להבטיח שהנתונים מועברים בצורה מהמינה בין מכשירים. הוא מפרק נתונים לחבילות קטנות, מספר אותם ומרכיב אותם מחדש ביעד. הוא גם מטפל בבקרת גודש, ומבטיח שימוש יעיל במשאבי הרשת.

IP אחראי להתייחסות ולניתוב נתונים בין מכשירים. לכל מכשיר באינטרנט יש כתובת IP ייחודית, המאפשרת שליחת נתונים ליעד הנכון.

ביחד, TCP ו-IP מספקים אמצעי חזק ואמין להעברת נתונים בין רשתות, מה שהופך את האינטרנט ורשתות מחשבים רבות אחרות לאפשריות.

צד שרת:

השרת תמיד מחובר וחייב להיות זמין ומחובר ללא זמן מוגבל, הוא חולייה מקשרת בין מסד הנתונים ועדכנו, קשר בין לקוחות במערכת וכאשר לקוח מתחבר למערכת אחרי רישום או התחברות השרת דואג לעדכן את שמו ברשימה ובנוסף שומר מי בחדר ובאיזה חדר הלקוח הנוכחי התחבר ועוד מלא אינפורמציה.

בנוסף השרת אחראי לשמירת פרטי הכניסה של המשתמש כגון סיסמא ושם משתמש. השרת אחראי לקבלת פרטים מהלקוח ושמירתם במסד הנתונים יחד עם השאילתה המתבקשת לביצוע על הפרטים כגון הוספת פרטים, אימות, עדכון, מחיקה.

כל המידע המעובר במערכת בין הלקוחות הוא מוצפן לחלוטין ולא ניתן לגילוי ולפיענוח, במצב של שיחה פרטית בין לקוחות מתרחש מצב של הצפנה מקצה לקצה כך שגם השרת לא יודע לפענח את ההודעות העוברות בין הלקוחות בשיחה הפרטית ממש כמו whatsapp .

השרת נכתב בשפת פייתון בסביבת עבודה Pycharm.

צד לקוח:

הלקוח הוא המרכז הוא החשוב ביותר שדואג להתחבר לשרת ולהתחלת תהליכים כגון התחברות לשרת וישר להתחיל את תהליך החלפת המפתחות RSA.

השרת והלקוח מנהלים תמיד דיאלוג הלקוח מבקש משהו והשרת עונה וכך חוזר, הלקוח דואג לרישום הלקוח או התחברותו למערכת ומקבל תשובה.

לאחר מכן לפי התוכן המתקבל מן השאילתה שהתבצעה הלקוח יודע לנתח את ההודעה האם להביא למשתמש תשובה שלילית או חיובית בנוגע לרישום/התחברות/החלפת מפתחות/ HMAC .

הלקוח נכתב בשפת פייתון בסביבת עבודה Pycharm.

כעת אסביר ואמחיש את האלגוריתמים המרכזיים בפרויקט (קוד + הסבר) בין היתר החלפת המפתחות, OTP, PRNG, HMAC ועוד

תיאור ההצפנות ואלגוריתמים מרכזיים:

החלפת המפתחות RSA (שיחה בחדר):

את אלגוריתם RSA הסברתי בעמוד 7-10 ואותו אסביר בקוד פייתון ממש עכשיו אבל לפני רצוי להסביר למה השתמשתי באלגוריתם זה.

אלגוריתם RSA הוא אלגוריתם הצפנה במקור שמו המקורי הוא RSA encryption כלומר הצפנת RSA בעברית אבל בפרויקט שלי אני משתמש בו על מנת להעביר את ה – SEED האחראי להצפנת המידע בין הלקוח ליתר הלקוחות בחדר אותו SEED אנחנו משתמשים על מנת להצפין ולפענח בצורה סיכרונית והסביר אותו עכשיו.

```
def key_exchange(server_socket, p, q, x):
    y = 0
    while y == 0 or y == 1:
        if y == 1:
            p = random.choice(primes)
            q = random.choice(primes)
            x = random.randint(10, 100)
            if not my_functions.is_prime(p) or not my_functions.is_prime(q):
                raise ValueError('P or Q were not prime')
            eq = (p - 1) * (q - 1) + 1
            y = 1
            xy = x * y
            while xy != eq:
                x += 1
                y = eq // x
                xy = x * y

    data_list.append(p * q)
    data_list.append(x)
    data_list.append(y)
    print("public key, x: " + str(x))
    print("private key, y: " + str(y))
    print("modulo (pq): " + str(p * q))

    send_to_server_public_key_and_pq(server_socket, x, p * q) # send first time

enc_seed = int(my_functions.recvall_with_decode(server_socket)) # get from server the enc seed
print(f'enc seed {enc_seed}')
dec_seed = my_functions.rsa_encryption_decryption(p * q, y, enc_seed) # dec the enc seed
print(f'seed {dec_seed}')

enc_a = server_socket.recv(32).decode()
print(f'1 {enc_a}')

if not enc_a.isdigit():
    sys.exit()

enc_b = server_socket.recv(32).decode()
print(f'2 {enc_b}')
```

זאת פונקציה שהלקוח עושה שימוש והיא נקראת מיד אחרי החיבור לשרת וזה קורה מיד אחרי שלחנו על התחברות למערכת או יצירת חשבון במערכת הלקוח יוצר שני מספרים prime כלומר ראשוניים על מנת לייצר את המודולו q לאחר מכן אנחנו מייצרים את המפתח הציבורי והפרטי של הלקוח לכל לקוח במערכת יהיה זוג מפתחות אחד להצפין ואחד לפענח. ושומרים את המפתחות שייצרנו ביחד עם המודולו ושולחים ישר לשרת את המפתח הציבורי שלנו עם המודולו על מנת שהשרת ישלח לנו את ה – SEED תקשורת על מנת להצפין את ההודעות הנשלחות דרך השרת ולשמור על האבטחת המערכת.

```
def key_exchange(client_socket):
    ab = []
    public_key_and_pq = my_functions.recvall_with_decode(client_socket) # get the public key and the pq
    client_seeds[client_socket] = [int(i) for i in public_key_and_pq.split(' ')] # first public key second pq
    # print(f'public key {client_seeds[client_socket][0]}')
    # print(f'pq {client_seeds[client_socket][1]}')
    logging.info(f'public key {client_seeds[client_socket][0]}')
    logging.info(f'pq {client_seeds[client_socket][1]}')

    # create the seed randomali and send it to the client
    seed = random.randint(0, 5000)
    a = random.randint(10, 99)
    ab.append(a)
    print(f'a {ab[0]}')
    b = random.randint(10, 99)
    ab.append(b)
    print(f'b {ab[1]}')

    client_seeds[client_socket].append(seed) # third seed
    # print(f'real seed {seed}')
    logging.info(f'real seed {seed}')
    # encrypt the seed
    enc_seed = my_functions.rsa_encryption_decryption(client_seeds[client_socket][1], client_seeds[client_socket][0],
                                                       client_seeds[client_socket][2])
    # print(f'enc seed {enc_seed}')
    logging.info(f'enc seed {enc_seed}')
    # print(f'enc_seed {type(enc_seed)}')
    logging.info('-----next_client-----')
    # send it to the client
    client_socket.send(str(enc_seed).encode()) # send the enc seed to the client

    enc_a = my_functions.rsa_encryption_decryption(client_seeds[client_socket][1], client_seeds[client_socket][0], ab[0])
    print(f'enc_a {enc_a}')
    # print(f'enc_a {type(enc_a)}')
    client_socket.send(str(enc_a).encode())

    time.sleep(1)

    enc_b = my_functions.rsa_encryption_decryption(client_seeds[client_socket][1], client_seeds[client_socket][0], ab[1])
    print(f'enc_b {enc_b}')
```

התמונה הנ"ל משקפת את צד השרת, השרת מיד אחרי ההתחברות מקבל את המידע שהלקוח שלח לו כלומר המפתח הציבורי והמודולו שהוא חלק מהמפתח הציבורי והשרת שומר את המידע שהוא קיבל מהלקוח בקובץ הנקרא serverLog.log השומר מידע הכלול בו המפתח הציבורי של הלקוח כולל המודולו המתחבר כולל את הכתובת של הלקוח המתחבר כלומר כתובת ה – IP והפורט שממנו הוא מתחבר, ה – SEED שאותו אנחנו ניצור את ה – Padים של ההודעות שנשלח לשרת ואת ה – SEED המוצפן שאותו נשלח ללקוח בחזרה כל המידע הזה נשמר לצורכי בקרה של המערכת ושיפור השירות.

השרת מגריל מספר 0-5000 שהוא ה – SEED וביחד איתו עוד שני מספרים בני 2 ספרות אותם מספרים משומשים על מנת ליצר את ה – Pad ימים של ההודעות (את יצירת ה – Pad הסביר בהמשך) השרת שולח ללקוח בצורה מוצפנת בהמצאות המפתח הציבורי של אותו לקוח ללקוח והלקוח מפענח בהמצאות המפתח הפרטי שלו ושומר את ה – SEED ואת שני המספרים לייצור ה – Pad ימים.

במילים אחרות ובמשפט אלגוריתם RSA עוזר לנו במערכת לשיתוף סוד אותו סוד קטן הוא ה – SEED לייצור ה – Pad ימים.

החלפת המפתחות בשיטת RSA בשיחה פרטית:

החלפת המפתחות בשיחה פרטית היא בדיוק כמו החלפת המפתחות בין השרת ללקוח על מנת שיוכל לדבר באחד מחדרי המערכת אבל פה החלפת המפתחות מתבצעת בין הלקוח היוצר ללקוח השני מבלי שהשרת יוכל לפענח את ההודעות העוברות בשרת כשפה המקצועית הצפנה מקצה לקצה ממש כמו whatsapp .

על מנת שהלקוח ירצה ליצור שיחה פרטית הוא צריך לשלוח לשרת "private!" השרת ידע שהלקוח היוצר רוצה ליצור שיחה פרטית ולאחר מכן השרת יחכה לשם משתמש שבו הלקוח היוצר רוצה ליצור איתו שיחה. השרת יבצע כמה בדיקות ראשית האם הלקוח המבוקש כבר בשיחה פרטית עם אדם אחר אם כן יעדכן את הלקוח היוצר. לאחר מכן יבדוק האם הלקוח היוצר רוצה ליצור שיחה פרטית אם עצמו משהו שהוא לא אפשרי ולכן ישלח לו הודעה בהתאם ובדיקה אחרונה בדיקה האם שם המשתמש שהלקוח היוצר רוצה ליצור איתו שיחה פרטית אם הוא קיים.

```
elif msg_from_client == '!private': # trying to start private msg # part 1
    print('*')
    enc_to_who = my_functions.recvall(client_socket)
    to_who = decrypt_cipher(enc_to_who, client_seeds[client_socket][3])
    if to_who in clients.keys() and to_who != user_name:
        # enc_private_msg = my_functions.recvall(client_socket)
        # private_msg = decrypt_cipher(enc_private_msg, client_seeds[client_socket][3])
        # private_msg = msg_header_private + private_msg
        if to_who in private_sessions.keys():
            client_socket.send(encrypt_msg(f'ERROR: {to_who} already in private session', client_seeds[client_socket][3]))
        else:
            private_sessions[to_who] = [user_name, 0]
            accept_to_private_session_msg = f'{user_name} wants to start a private session with you. (confirm_p/decline_p)'
            clients[to_who].send(encrypt_msg(accept_to_private_session_msg, client_seeds[clients[to_who]][3]))
    else:
        if to_who == user_name:
            client_socket.send(encrypt_msg(f'ERROR: you can not start a private session with yourself',
                                           client_seeds[client_socket][3]))
        else:
            client_socket.send(encrypt_msg(f'ERROR: there is no client called {to_who}', client_seeds[client_socket][3]))
    else:
        msg_from_client_to_others = msg_header + msg_from_client
        for others_clients in current_room.values():
            if others_clients is not client_socket:
                if others_clients not in [clients[c] for c in private_sessions.keys()] and others_clients not in [clients[private_sessions[c][0]] for c in private_sessions.keys()]:
                    others_clients.send(encrypt_msg(msg_from_client_to_others, client_seeds[others_clients][3]))
```


אם הלקוח המבוקש לשיחה פרטית קיים במערכת ולא בשיחה פרטית עם לקוח אחר השרת ישלח לו את שם המשתמש שברצונו ליצור איתו שיחה פרטית ויבקש ממנו אישור או סירוב אנחנו מעדכנים בנוסף במערך את שם המשתמש היוצר ושם המשתמש המבוקש והשלב שאנחנו נמצאים בו כרגע 1.

שיחה בין הלקוח השני לשרת:

```
if user_name in private_sessions.keys(): # part 2
    if private_sessions[user_name][1] == 0: # need to answer to private session
        print('**)')
        if msg_from_client == 'decline_p':
            clients[private_sessions[user_name][0]].send(encrypt_msg(f'{user_name} declined to start a private session with you', client_seeds[clients[private_sessions[user_name][0]]][3]))
            private_sessions.pop(user_name)
        elif msg_from_client == 'confirm_p':
            clients[private_sessions[user_name][0]].send(encrypt_msg(f'{user_name} confirmed to start a private session with you', client_seeds[clients[private_sessions[user_name][0]]][3]))
            private_sessions[user_name][1] = 1
            clients[private_sessions[user_name][0]].send(encrypt_msg('creator_rsa', client_seeds[clients[private_sessions[user_name][0]]][3]))
```

השרת בודק אם השלב הקודם הושלם בהצלחה ומחכה לתשובה מהלקוח השני אם הלקוח השני סירב השרת שולח ללקוח היוצר שהלקוח המבוקש סירב לשיחה הפרטית אם הלקוח המבוקש כלומר השני אישר את השיחה הפרטית הוא שולח ללקוח היוצר הודעה בהתאם ובנוסף מעדכן שהשלב השני הושלם ושולח ללקוח היוצר את ההודעה "creator_rsa" שאומר ללקוח היוצר תייצר שני מפתחות להצפנת ה – SEED המיועד להצפנת המידע העובר בין הלקוחות המנהלים שיחה פרטית.

הלקוח היוצר:

```
if msg_from_server == 'creator_rsa': # the client that wants to start private session, create the keys
    key_exchange_for_private(s, random.choice(primes), random.choice(primes),
                             random.randint(10, 100))
    private_flag = True
```

הלקוח היוצר מקבל את הודעתו של השרת ויוצר מפתחות בהתאם לבקשתו ושולח לשרת את המפתח הציבורי שנוצר בעקבות האלגוריתם RSA וגם את המודולו ובתוך פונקציית key_exchange_for_private ישנו גם את ה – SEED ששלח הלקוח השני מוצפן עם המפתח הציבורי שכרגע יצר הלקוח היוצר הלקוח היוצר מפענח את ה – SEED ושומר אותו במערך ה – data_list_for_private.

להלן פונקציית key_exchange_for_private:

```
def key_exchange_for_private(server_socket, p, q, x):
    y = 0
    while y == 0 or y == 1:
        if y == 1:
            p = random.choice(primes)
            q = random.choice(primes)
            x = random.randint(10, 100)
            if not my_functions.is_prime(p) or not my_functions.is_prime(q):
                raise ValueError('P or Q were not prime')
            eq = (p - 1) * (q - 1) + 1
            y = 1
            xy = x * y
            while xy != eq:
                x += 1
                y = eq // x
                xy = x * y

    data_list_for_private.append(p * q)
    data_list_for_private.append(x)
    data_list_for_private.append(y)
    print("public key_for_private, x: " + str(x))
    print("private key_for_private, y: " + str(y))
    print("modulo_for_private (pq): " + str(p * q))

    print('*')

    send_to_server_public_key_and_pq_for_private(server_socket, x, p * q)

    print('*')

    enc_seed = int(my_functions.recvall_with_decode(server_socket))
    print('***')
    print(f'enc seed_for_private1 {enc_seed}')
    dec_seed = my_functions.rsa_encryption_decryption(p * q, y, enc_seed)
    print(f'seed_for_private {dec_seed}')

    data_list_for_private.append(SetSeed(dec_seed, 8888, 9999))
```

השרת מקבל את המפתח הציבורי ואת המודולו של הלקוח היוצר ושולח ללקוח שני שייצר SEED משותף ביניהם ונוסף מעדכן שאנחנו עוברים לשלב השלישי:

```
else:
    if [user_name, 1] in private_sessions.values(): # part 3
        for (sec_user, me) in private_sessions.items():
            if me[0] == user_name:
                break
        print('***')
        public_key_private = my_functions.recvall(client_socket)
        print(f'public key from c {public_key_private};')
        pq_private = my_functions.recvall(client_socket)
        print(f'pq key from c {pq_private};')
        clients[sec_user].send(encrypt_msg('second_participant', client_seeds[clients[sec_user]][3]))
        clients[sec_user].send(public_key_private)
        time.sleep(0.05)
        clients[sec_user].send(pq_private)
        private_sessions[sec_user][1] = 2
```

הלקוח השני מקבל את מפתחו של הלקוח היוצר עם המודולו ומצפין את ה-SEED שהגדיר
בשיטת הצפנה RSA ושולח בחזרה לשרת

```
if msg_from_server == 'second_participant':
    public_key = int(my_functions.recvall_with_decode(s))
    qp = int(my_functions.recvall_with_decode(s))
    data_list_for_private.append(qp)
    data_list_for_private.append(public_key)
    seed = random.randint(0, 500)
    print(f'seed_for_private2 {seed}')
    data_list_for_private.append(SetSeed(seed, 8888, 9999))
    enc_seed = my_functions.rsa_encryption_decryption(qp, public_key, seed)
    print(f'enc seed_for_private2 {enc_seed}')
    s.send(str(enc_seed).encode())
    private_flag = True
```

השרת בודק האם אנחנו בשלב השלישי ומקבל את ה-SEED שהוגרל כמובן מוצפן ושולח
את ה-SEED שהוצפן ללקוח היוצר.

```
if user_name in private_sessions.keys() and private_sessions[user_name][1] == 2: # part 4
    print('****')
    enc_seed = my_functions.recvall(client_socket)
    clients[private_sessions[user_name][0]].send(enc_seed)
    private_sessions[user_name][1] = 3
```

הלקוח היוצר מקבל את ה-SEED שהוגרל מוצפן עם המפתח הציבורי שיצר בפונקציה
key_exchange_for_private ומפענח אותו עם המפתח הפרטי שיצר ומייצר אובייקט של
SetSeed על מנת להצפין את ההודעות בין הלקוח היוצר לשני.

ברגע שיוצר השיחה הפרטית שולח הודעה הוא שולח לשרת והשרת מקבל את ההודעה
וישר שולח ללקוח השני.

```
if [user_name, 3] in private_sessions.values(): # part 5
    for tuple_private in private_sessions.items(): # tuple_private = (sec_user, me)
        if tuple_private[1][0] == user_name:
            sec_user = tuple_private[0]
            break
    print('c')
    enc_private_msg_from_client = my_functions.recvall(client_socket)
    if enc_private_msg_from_client == '1'.encode():
        private_sessions.pop(sec_user)
    clients[sec_user].send(enc_private_msg_from_client)
```

ברגע שהלקוח השני שולח הודעה השרת מקבלת את ההודעה המוצפנת ושולח ישירות ללקוח היוצר.

```
if user_name in private_sessions.keys() and private_sessions[user_name][1] == 3: # part 5 # second
    sec_userr = private_sessions[user_name][0]
    if user_name not in private_sessions.keys():
        continue
    print('s')
    enc_private_msg_from_client = my_functions.recvall(client_socket)
    if enc_private_msg_from_client == '1'.encode():
        private_sessions.pop(user_name)
    clients[sec_userr].send(enc_private_msg_from_client)
```

PRNG (Pseudorandom number generator) מחולל מספרים פסידו-אקראיים

מחולל מספרים פסידו-אקראיים (Random Number Generator, RNG) הוא כלי תוכנה או חומרה שמייצר מספרים אקראיים בצורה סטטיסטית. המטרה של המחולל הוא ליצור מספרים שאינם חוזרים ואקראיים לחלוטין.

מחוללי מספרים פסידו-אקראיים משמשים במגוון רחב של יישומים כגון משחקי מחשב, סוכנויות ממשלתיות, מוסדות פיננסיים וכל מקום אחר שדרושה יצירת מספרים אקראיים.

אני אעשה שימוש ב – PRNG על מנת לייצר את ה – Pad להצפנת ההודעות.

```
class SetSeed:
    def __init__(self, seed, a, b):
        self.start_seed = seed
        self.A = a
        self.B = b
        # print(self.A)
        # print(self.B)

    def print_p(self):
        print(self.start_seed)

    def my_random_pad_generator(self, a, b, len_key, final_key):
        while len_key != 0:
            a_s = int((str(a) + str(self.start_seed)) + (str(a) + str(self.start_seed)))[4]
            self.start_seed = (a_s + b) % (2 ** 32)
            if (self.start_seed % 2) == 0:
                final_key.append(format(1, 'b'))
            else:
                final_key.append(format(0, 'b'))
            len_key = len_key - 1
        return final_key

    def my_key_stream_create_pad(self, key_len):
        pad = ''.join(self.my_random_pad_generator(self.A, self.B, key_len, []))
        # print(f'pad: {pad}')
        return pad
```

בשיטת המחולל מספרים פסידו-אקראיים אעשה שימוש גם בשלושה ארגומנטים כמו שנראה בתמונה SEED התחלתי ושני מספרים a ו- b . ה-`start_seed` משתנה בהתאם לגודל ההודעה שהלקוח רוצה לשלוח.

את פונקציית ה-`PRNG` אנסה לעשות אותה כמה שיותר מהירה כלומר שיצירת ה-`Pad` יהיה מהיר כי יצירת ה-`Pad` לוקח מלא זמן ואנחנו לא רוצים את זה.

בהתחלה נקרא לפונקציה `my_key_stream_create_pad` לפי גודל ההודעה והוא יחזיר לנו את ה-`Pad` לפי `SetSeed` שקראנו לו.

הפונקציה `my_random_pad_generator` רצה ככמות גודל ההודעה שהלקוח רוצה לשלוח ומייצר את ה-`Pad` בהתאם כשהפונקציה לוקחת את a וממירה אותו למחרוזת ומשרשת אליו את `start_seed` ולתוצאה משרשת את האיבר הרביעי של השירשור לדוגמא:

אם $a = 56$ ו-`start_seed = 571` אז `a_s` שווה ל:

$$a_s = \text{int}((56 + 571) + (56 + 571)[4]) = \text{int}(56571 + 1) = \text{int}(565711) = 565711$$

את התוצאה אנחנו מחברים עם b ומבצעים מודולו של 2^{32} ושומרים ב-`start_seed` ובכך אנחנו משנים כל פעם את `start_seed` בצורה רנדומלית כך שאף גורם זר לא יוכל לדעת את הצעד הבא.

לאחר הפעולה האחרונה אנחנו בודקים אם התוצאה היא זוגית אם היא זוגית אז אנחנו משרשרים ל-`final_key` 1 אם התוצאה שלילית אז 0 ובסוף הפונקציה אנחנו מחזירים את ה-`final_key`.

OTP (One Time Pad) פנקס חד פעמי –

הצפנת פנקס חד פעמי היא צורת הצפנה המשתמשת במפתח אקראי באותו אורך כמו הודעת הטקסט המקורי. המפתח משתמש פעם אחד בלבד, ומכאן השם "חד פעמי". מיותר לציין ששיטת הצפנה זאת היא הצפנה סימטרית כאשר המפתח שלה הוא הפנקס שאנחנו מציינים וקרוי באנגלית Pad. המפתח או הפנקס נוצר באמצעות תהליך אקראי באמת, כגון הטלת מטבע, ולעולם לא נעשה בו שימוש חוזר.

כדי להצפין הודעה באמצעות פנקס חד פעמי, הודעת הטקסט המקורי משולבת עם המפתח האקראי על ידי ביצוע פעולת XOR בלעדית בסיביות על כל סיביות של הטקסט המקורי עם הסיביות המקבילות של המפתח. התוצאה היא הודעת טקסט צופן, אותה ניתן לפענח רק על ידי ביצוע אותה פעולת XOR על טקסט המוצפן עם המפתח.

אחד השמות המתארים את אלגוריתם OTP הוא הצפנה מושלמת כי הוכח מתמטית שאם המפתח המשתמש להצפנה נבחר אקראי והשימוש בו הוא חד פעמי, ההצפנה בלתי ניתנת לשבירה אפילו עם ליריב בעל עוצמת חישוב בלתי מוגבלת.

על מנת להקל אלינו אנחנו ממירים את הודעת הטקסט למחרוזת בינארית של אפס אחד שהוא הייצוג המחשבי במחשב. המפתח המשמש להצפנה גם מומר לאפס ואחד שאורכו כמובן כאורך הטקסט המקורי.

$$C = M \text{ XOR } K$$

$$M = C \text{ XOR } K$$

- M מציין את ההודעה המקורי
- K מציין את המפתח
- C מציין את ההודעה המוצפנת

אחד הסיבות שאסור להשתמש בפנקס יותר מפעם אחת היא ששימוש חוזר בפנקס יהפוך את הצופן לקל מאוד לשבירה בהינתן שני מסרים שהוצפנו עם אותו מפתח ביצוע XOR בין ההודעות המוצפנות יחזיר את פעולת ה XOR בין שני ההודעות הלא מוצפנות כלומר ההודעות המקוריות מה שיגרום לביטול השפעת המפתח עליהם ובהינתן מסר אחד מבין השניים אפשר לבטא את השני בקלות. לדוגמא:

$$C1 \text{ XOR } C2 = (M1 \text{ XOR } K) \text{ XOR } (M2 \text{ XOR } K) = M1 \text{ XOR } M2 \text{ XOR } K \text{ XOR } K = M1 \text{ XOR } M2 \text{ XOR } 0 = M1 \text{ XOR } M2$$

אחד הסיבות לשימוש ב XOR במקום ב AND או OR נובעת מהסיבה ש XOR אקראי אפשר להוכיח זאת בכמה דרכים, אבל מספיק להתבונן בטבלה של XOR ולראות

שהיא מכילה בדיוק 50 אחוז אס ו – 50 אחוז אחד לעומת AND ו – OR שיחס ההתפלגות או 1 ל – 3.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

פונקציית ההצפנה בשיטת OTP:

```
def encrypt_msg(msg, pad):
    msg_bin = text_to_byte(msg)
    pad_bin = pad.my_key_stream_create_pad(len(msg_bin)) # create a pad
    # print(f'encryption pad for {msg}:\n{bytes(pad_bin, "ascii")}')

    enc_msg_to_send = encrypt(bytes(msg_bin, 'ascii'), bytes(pad_bin, 'ascii'))
    # print(f'enc msg: {enc_msg_to_send}')

    # dec_msg = decrypt(enc_msg_to_send, bytes(pad_bin, 'ascii'))
    # print(f'*****: {byte_to_text(dec_user_name)}')
    return enc_msg_to_send
```

פונקציית ההצפנה בשיטת OTP:

```
def decrypt_cipher(cipher, pad):
    # print(f'enc res: {cipher}')
    pad_bin = pad.my_key_stream_create_pad(len(cipher))
    # print(f'decryption pad for {cipher}:\n{bytes(pad_bin, "ascii")}')

    cipher_dec = decrypt(cipher, bytes(pad_bin, 'ascii'))
    msg_from_server = byte_to_text(cipher_dec)
    # print(f'decrypt msg: {msg_from_server}')
    return msg_from_server
```

כפי שניתן לראות בפונקציית ההצפנה אנחנו מכניסים אליה את ההודעה המקורית יחד עם האובייקט ששמו Pad זה לא כל כך Pad אלה אובייקט שמייצר את ה – Pad העושה שימוש בפונקציית ה – PRNG על פעם שנרצה להצפין נקרא לפונקציה יחד עם הפרמטרים שציניתי. ההצפנה עובדת כך אנחנו ממירים את ההודעה לבתים ואז יוצרים Pad בגודל כמות

מספר הבתים שיש בטקסט המקורי ועושים פעולה XOR (bit by bit) ואת התוצאה שולחים לשרת.

השרת מקבל את ההודעה המוצפנת ומכניס אליה את ההודעה המוצפנת יחד עם האובייקט של יצירת ה- PRNG מצייר את אותו Pad שייצר הלקוח להצפנת ההודעה המקורית לפי גודל ההודעה המוצפנת עושה XOR (bit by bit) עם ההודעה המוצפנת ובכך מקבל את ההודעה המקורית.

HMAC algorithm אלגוריתם HMAC:

אלגוריתם HMAC הוא אלגוריתם קריפטוגרפי המשמש לאימות אותנטיקציה והשלמות ההודעה. זהו סוג של קוד אימות הודעות (MAC) הכולל פונקציית hash קריפטוגרפית בשילוב מפתח סודי. אלגוריתם HMAC לוקח הודעה ומפתח סודי כקלט, ומייצר פלט בגודל קבוע, המכונה ה-HMAC digest. בדרך כלל נעשה שימוש בתקציר HMAC כדי לאמת את האותנטיקציה והשלמות של ההודעה.

אלגוריתם HMAC פועל על ידי החלת תחילה פונקציית גיבוב קריפטוגרפית על ההודעה, אשר מייצרת ערך גיבוב. לאחר מכן, המפתח הסודי משורשר עם ערך ה-hash והתוצאה שוב מועברת באמצעות אותה פונקציית hash קריפטוגרפית. הפלט הסופי הוא ה-HMAC digest.

HMAC נמצא בשימוש נרחב בפרוטוקולי אבטחה שונים. הוא מספק דרך לאמת את ההודעה, מבלי לחשוף את המפתח הסודי המשמש לאימות. זה הופך אותו לכלי יעיל להגנה מפני גישה לא מורשית לנתונים רגישים.

אעשה שימוש באלגוריתם HMAC על מנת לאמת מסמכים הנשלחים לשרת על מנת לאבטח את המערכת בכך שגורמים זרים לא יוכלו לשלוח לי מסמכים מזויפים משהו שאנחנו לא נרצה במערכת.


```

if hmac_flag_photo: # send file
    try:
        f = open(user_msg)
        f.close()
    except IOError:
        print("File not accessible")
        enc_file_name_to_send = encrypt_msg('File not accessible', user_pad)
        s.send(enc_file_name_to_send)
        hmac_flag_photo = False
        return 'File not accessible'

    enc_file_name_to_send = encrypt_msg(user_msg, user_pad)
    s.send(enc_file_name_to_send)

    file_name_hmac = hmac.new(str(seed_server).encode(), user_msg.encode(), hashlib.sha256).hexdigest()
    enc_file_name_hmac_to_send = encrypt_msg(file_name_hmac, user_pad)
    s.send(enc_file_name_hmac_to_send)

    digest_maker = hmac.new(str(seed_server).encode(), ''.encode(), hashlib.sha256)
    try:
        f = open(user_msg, 'rb')
        try:
            block = f.read(1024)
            while block:
                # print('*')
                digest_maker.update(block)
                print('*')
                print(block)
                enc_block_to_send = encrypt_file(block, user_pad)
                print('*')
                print(enc_block_to_send)
                if hack_flag: # יציאה מלול נא לשלוח פלט
                    s.send(len(enc_block_to_send) * b'0')
                else:
                    s.send(enc_block_to_send)

```

```

        block = f.read(1024)
    finally:
        f.close()
        hack_flag = False
    time.sleep(0.2)
    enc_block_to_send = encrypt_file(b'0', user_pad)
    s.send(enc_block_to_send)

    time.sleep(0.05)

    digest = digest_maker.hexdigest()
    print(f'hmac for the real file {digest}')

    enc_digest_to_send = encrypt_msg(digest, user_pad)
    s.send(enc_digest_to_send)

except:
    pass
hmac_flag_photo = False
return 'sent'

```

הלקוח פותח את הקובץ עם הקובץ לא קיים יזרוק שגיאה שהקובץ לא קיים וישלח לשרת שמשוהו קרה לקובץ ויבטל את המנגנון של שליחת הקובץ ואימותו.

```
elif msg_from_client == 'FILE': # file
    enc_file_name = my_functions.recvall(client_socket)
    file_name = decrypt_cipher(enc_file_name, client_seeds[client_socket][3])

    if file_name == 'File not accessible': # if there is no file called that
        continue

    enc_file_name_hmac = my_functions.recvall(client_socket)
    file_name_hmac = decrypt_cipher(enc_file_name_hmac, client_seeds[client_socket][3])

    if hmac.new(str(client_seeds[client_socket][2]).encode(), file_name.encode(), hashlib.sha256).hexdigest() == file_name_hmac:
        f = open(f"server_temp_file.{file_name.split('.')[-1]}", "wb")

        real_digest_maker = hmac.new(str(client_seeds[client_socket][2]).encode(), ''.encode(), hashlib.sha256)

        enc_block_file = my_functions.recvall(client_socket)
        # print(enc_block_file)
        block_file = decrypt_cipher_file(enc_block_file, client_seeds[client_socket][3])
        # print(block_file)
        # print(type(block_file))

        while block_file != b'0': # get the file
            print('*')
            f.write(block_file)
            # print('*')
            real_digest_maker.update(block_file)

            enc_block_file = my_functions.recvall(client_socket)
            block_file = decrypt_cipher_file(enc_block_file, client_seeds[client_socket][3])
            print(block_file)
            # print(block_file)
            # print(block_file.encode())

        f.close()

    real_digest = real_digest_maker.hexdigest()
    print(f'hmac for the content in the file that sent {real_digest}')
```

הלקוח בודק אם הקובץ הנבחר קיים אם הוא לא קיים אז הוא שולח לשרת הודעה בהתאם אם לא ממשיך את תהליך שליחת הקובץ הלקוח שולח את שם הקובץ מוצפן ב – OTP ושם הקובץ מוצפן ב – HMAC שהתוצאה היא Hmac digest .

השרת מקבל את שני הנתונים לוקח את השם הקובץ המוצפן ב – OTP מפענח ומבצע עליו HMAC התוצאה תהיה HMAC digest ואתה אותה תוצאה נשווה את ה – HMAC digest ששלח הלקוח למה שהשרת יצר אם הם שווים אז הכל טוב ואין התערבות של גורם שלישי אם לא שולח ללקוח הודעה בהתאם ומפסיק את תהליך שליחת הקובץ.

לאחר מכן הלקוח מייצר אובייקט של HMAC שהמפתח הסימטרי יהיה ה – SEED שהעברנו בהחלפת המפתחות עם השרת ההודעה תהיה בהתחלה ריקה וה – HMAC digest תהיה בת 16 בתים. את הקובץ אנחנו קוראים כל פעם 1024 בתים ושולחים לשרת מוצפן באמצעות OTP חשוב לציין שכל פעם אנחנו מבצעים update ל – HMAC יחד עם

הבלוק הנקרא פונקציה מחזירה HMAC digest המורכב מההודעה הקודמת שעשינו לה
HMAC + הבלוק ובכך אנחנו משנים כל פעם את החתימה בהתאם לבלוק על מנת לבדוק כל
הקובץ הגיע במלואו לשרת.

השרת מקבל את הקובץ בבלוקים ומפענח ובנוסף מייצר HMAC digest כמו הלקוח וכותב
לקובץ עזר בסוף הוא מקבל את ה – HMAC digest של הלקוח ומשווה בין ה – HMAC
digest שהוא ייצר אם הם שווים אז אפשר לשלוח את הקובץ לשאר הלקוחות ולהתחיל את
אותו תהליך שהסביר רק שהפעם השרת או שולח הקובץ והלקוחות המיועדים הם מקבלי
הקובץ.

```
enc_body_file_hmac = my_functions.recvall(client_socket)
body_file_hmac = decrypt_cipher(enc_body_file_hmac, client_seeds[client_socket][3])
print(f'hmac for the real file {body_file_hmac}')

if real_digest == body_file_hmac: # check the integrity of the file body
    print(f'The FILE sent from {user_name} are original')
    for others_clients in current_room.values():
        if others_clients is not client_socket:
            if others_clients not in [clients[c] for c in private_sessions.keys()] and others_clients not in [clients[private_sessions[c][0]] for c in private_sessions.keys()]:
                others_clients.send(encrypt_msg('get file photo', client_seeds[others_clients][3]))
                time.sleep(0.5)
                others_clients.send(encrypt_msg(f'{user_name}_{os.path.basename(file_name)}', client_seeds[others_clients][3]))
    time.sleep(0.1)

    hmac_for_clients = {}
    for others_clients in current_room.values():
        if others_clients is not client_socket:
            hmac_for_clients[others_clients] = hmac.new(str(client_seeds[others_clients][2]).encode(), ''.encode(), hashlib.sha256)

    f = open(f"server_temp_file.{file_name.split('.')[-1]}", "rb")
    block = f.read(1024)

    while block: # sending the file to all clients
        # print('*')
        # print(block)
        # print(block.decode())
        for others_clients in current_room.values():
            if others_clients is not client_socket:
                if others_clients not in [clients[c] for c in private_sessions.keys()] and others_clients not in [clients[private_sessions[c][0]] for c in private_sessions.keys()]:
                    hmac_for_clients[others_clients].update(block)
                    others_clients.send(encrypt_msg_file(block, client_seeds[others_clients][3]))
                time.sleep(0.1)

        block = f.read(1024)
    f.close()
```

```
# os.remove("server_temp_file.txt")
time.sleep(0.5)
for others_clients in current_room.values():
    if others_clients is not client_socket:
        if others_clients not in [clients[c] for c in private_sessions.keys()] and others_clients not in [clients[private_sessions[c][0]] for c in private_sessions.keys()]:
            others_clients.send(encrypt_msg_file(b'0', client_seeds[others_clients][3]))

time.sleep(0.1)
for others_clients in current_room.values():
    if others_clients is not client_socket:
        if others_clients not in [clients[c] for c in private_sessions.keys()] and others_clients not in [clients[private_sessions[c][0]] for c in private_sessions.keys()]:
            others_clients.send(encrypt_msg(hmac_for_clients[others_clients].hexdigest(), client_seeds[others_clients][3]))

hmac_for_clients.clear()

else:
    print(f'Something change in the FILE sent from {user_name} 2')
else:
    print(f'Something change in the FILE sent from {user_name} 1')
```

חשוב להזכיר ברגע שהגענו לשלב הזה כל התהליך קורה שוב הפעם בין השרת ללקוח המיועד לקבל את הקובץ אבל פה השרת הוא שולח הקובץ והלקוח המיועד מקבל את הקובץ.

פונקציה המצפינה את הקובץ המיועד לשליחה:

```
def encrypt_msg_file(block, pad):
    pad_bin = pad.my_key_stream_create_pad(len(block)) # create a pad
    # print(f'encryption pad for {msg}:\n{bytes(pad_bin, "ascii")}')

    enc_msg_to_send = encrypt(block, pad_bin.encode())
    # print(f'enc msg: {enc_msg_to_send}')

    # dec_msg = decrypt(enc_msg_to_send, bytes(pad_bin, 'ascii'))
    # print(f'*****: {byte_to_text(dec_user_name)}')
    return enc_msg_to_send
```

הפונקציה עובדת בדיוק כמו פונקציית ההצפנה של הודעת טקסט רק על קבצים אותו הדבר פענח של הקובץ.

```
def decrypt_cipher_file(cipher, pad):
    # print(f'enc res: {cipher}')
    pad_bin = pad.my_key_stream_create_pad(len(cipher))
    # print(f'decryption pad for {cipher}:\n{bytes(pad_bin, "ascii")}')

    cipher_dec = decrypt(cipher, pad_bin.encode())
    # print(f'decrypt msg: {msg_from_client}')
    return cipher_dec
```

תהליך התחברות או הירשמות למערכת:

```
# step 1 - server send msg 1 to client
msg_to_send = "to login send 1, to create account send 2"
enc_msg_to_send = encrypt_msg(msg_to_send, client_seeds[client_socket][3])
client_socket.send(enc_msg_to_send)
enc_ans = my_functions.recvall(client_socket)
if enc_ans == -999:
    return
ans = decrypt_cipher(enc_ans, client_seeds[client_socket][3])

if ans == '2':
    msg_to_send = "choose your first name, last name, user name and the password you want(with a enter between each) to login send 0"
    client_socket.send(encrypt_msg(msg_to_send, client_seeds[client_socket][3]))
elif ans == '1':
    msg_to_send = "what is your user name and your password?"
    client_socket.send(encrypt_msg(msg_to_send, client_seeds[client_socket][3]))

success = False
while not success and ans == '2': # create account
    first_name = decrypt_cipher(my_functions.recvall(client_socket), client_seeds[client_socket][3])
    # print(first_name)
    if first_name == '0':
        break
    last_name = decrypt_cipher(my_functions.recvall(client_socket), client_seeds[client_socket][3])
    # print(last_name)
    if last_name == '0':
        break
    user_name = decrypt_cipher(my_functions.recvall(client_socket), client_seeds[client_socket][3])
    # print(user_name)
    if user_name == '0':
        break
    password = decrypt_cipher(my_functions.recvall(client_socket), client_seeds[client_socket][3])
    # print(password)
    if password == '0':
        break
```

לאחר ההתחברות כאשר הלקוח בא לבחור בין התחברות להירשמות למערכת השרת שואל את הלקוח כמובן בצורה מוצפנת להתחברות שלח 1 אם לא להירשמות שלח 2. הלקוח שולח את הודעה המתאים בהתאם למה שלחץ השרת מעבד את זה ושולח לו מענה אם לחץ הירשמות ישלח אליו הודעה כזאת "אנא תבחר שם פרטי שם משפחה, שם משתמש וסיסמא" ואם לחת על התחברות ישלח לו "מה שם המשתמש שלך והסיסמא". הלקוח מעבד ושולח לו את הנתונים בהתאם.

השרת מקבל את הנתונים ועושה כמה בדיקות למשל עם השם משתמש שהלקוח רוצה ליצור קיים ישלח אליו הודעה ששם משתמש כך וכך קיים במערכת אם לא ישמור את הנתונים במערכת.

בדיקה נוספת כאשר הלקוח רוצה להתחבר למערכת השרת בודק עם שם המשתמש והסיסמא קיימים במסד הנתונים אם הם קיימים אז השרת ישלח ללקוח שההתחברות בוצע בהצלחה ואם אז ישלח הודעה בהתאם.

אחד השאלות הנשאלות זה איך השרת יודע איזה לקוח בחדר מסוים ואיך לשלוח את ההודעה שלקוח רוצה לשלוח בחדר לשאר חברי הקבוצה בחדר?

הפתרון הוא פשוט השרת שולח הודעת שאלה ללקוח הנמצא בחלון בחירת החדרים "איזה חדר ברצונך להתחבר" הלקוח ילחץ על חדר מהחדרים וישלח את שם החדר שהתחבר אליו השרת מעבד את הנתון ומעדכן במבנה הנתונים של החדר, מוסיף את הלקוח למבנה הנתונים.

```
enc_msg_to_send = encrypt_msg('Which room would you like to join?(work, talking, dating) ', client_seeds[client_socket][3])
client_socket.send(enc_msg_to_send)

enc_room = my_functions.recvall(client_socket)

room = decrypt_cipher(enc_room, client_seeds[client_socket][3])

if room == 'work':
    clients_room_work[user_name] = client_socket
    current_room = clients_room_work
elif room == 'dating':
    clients_room_dating[user_name] = client_socket
    current_room = clients_room_dating
else:
    clients_room_talking[user_name] = client_socket
    current_room = clients_room_talking
    room = 'talking'
```

הפעלת מצלמה ושליחת המצלמה בזמן אמת ליתר חברי הקבוצה:

```
# webcam share
if user_msg == 'WEBCAM':
    c = 0
    enc_msg_to_send = encrypt_msg(user_msg, user_pad)
    s.send(enc_msg_to_send)
    time.sleep(0.01)
    vid = cv2.VideoCapture(0)
    while vid.isOpened():
        c += 1
        img, frame = vid.read()
        frame = imutils.resize(frame, width=320)
        a = pickle.dumps(frame)
        message = struct.pack("Q", len(a)) + a
        # enc_msg = encrypt_file(message, user_pad)
        s.send(message)
        cv2.imshow('MY VIDEO', frame)
        key = cv2.waitKey(1) & 0xFF
        print(c)
        if c < 600:
            cv2.destroyAllWindows()
            break
    return
enc_msg_to_send = encrypt_msg(user_msg, user_pad)
# print(enc_msg_to_send)
# print(type(enc_msg_to_send))
s.send(enc_msg_to_send)
return 'continue'
```

על מנת לפתוח מצלמה ולשדר לשאר חברי הקבוצה נצטרך לרשום WEBCAM נצפין את ההודעה WEBCAM ונשלח לשרת השרת ידע שברצוננו לפתוח מצלמה וישלח לשאר חברי הקבוצה בצא"ט שמישהו רוצה לפתוח מצלמה ואת שמו של הלקוח בחדר שרוצה לפתוח מצלמה.

הלקוח ישלח כל פעם frame וישלח לשרת בצורה מוצפנת השרת ישלח ישר ללקוח המיועד מבלי לראות את ה – frame הלקוח המיועד מקבל את ההודעה יפענח את ה – frame ויראה מה היה ב – frame כי הרי וידיאו או כאשר אנחנו מפעילים מצלמה אנחנו בעצם מצלמים כל פעם תמונה frame. וידיאו מורכב מכמה frames , frame הוא חלק מהפאזל שהוא המצלמה בזמן אמת.

השרת:

```
elif msg_from_client == 'WEBCAM': # camera share

    for others_clients in current_room.values():
        if others_clients is not client_socket:
            if others_clients not in [clients[c] for c in private_sessions.keys()] and others_clients not in [clients[private_sessions[c][0]] for c in private_sessions.keys()]:
                others_clients.send(encrypt_msg('start camera', client_seeds[others_clients][3]))
                others_clients.send(encrypt_msg(user_name, client_seeds[others_clients][3]))

    enc_frame = my_functions.recvall(client_socket)
    # print('*')
    while enc_frame:
        # print('*')
        for others_clients in current_room.values():
            if others_clients is not client_socket:
                if others_clients not in [clients[c] for c in private_sessions.keys()] and others_clients not in [clients[private_sessions[c][0]] for c in private_sessions.keys()]:
                    others_clients.send(enc_frame)

    enc_frame = my_functions.recvall(client_socket)
```

הלקוח המיועד לקבל את ההודעה:

```
if msg_from_server == 'start camera':
    enc_from_who = my_functions.recvall(s)
    from_who = decrypt_cipher(enc_from_who, pad)

    data = b""
    payload_size = struct.calcsize("Q")
    while True:
        while len(data) < payload_size:
            packet = my_functions.recvall(s)
            #msg = decrypt_cipher_file(packet, user_pad)
            if not packet:
                break
            data += packet
        packed_msg_size = data[:payload_size]
        data = data[payload_size:]
        msg_size = struct.unpack("Q", packed_msg_size)[0]

        while len(data) < msg_size:
            data += my_functions.recvall(s)
            #msg = decrypt_cipher_file(data, user_pad)
        frame_data = data[:msg_size]
        data = data[msg_size:]
        frame = pickle.loads(frame_data)
        cv2.imshow(f'{from_who} VIDEO', frame)
        key = cv2.waitKey(1) & 0xFF
        if key == 20:
            break
    cv2.destroyAllWindows()
```

מבני הנתונים בהם עשיתי שימוש:

מבני הנתונים שבהם השתמשי הם:

בלקוח:

רשימה data_list – שומרת את המפתח הפרטי, המפתח הפרטי והמודולו של החלפת המפתחות שהתבצע עם השרת בתחילת הקשר.

רשימה data_list_for_private – רשימה זאת מיועדת ללקוחות בשיחה פרטית והיא שומרת את המפתח הציבורי של יוצר השיחה, המפתח הפרטי והמודולו של החלפת המפתחות עם הלקוח שני ואת האובייקט שנוצר עם ה – SEED של השיחה הפרטית. ללקוח השני יהיה בדיוק אותו הדבר רק בלי המפתח הפרטי של יוצר השיחה כי בכל זאת זה פרטי.

בשרת:

מילון clients_room_work – מילון המאחסן את המשתמשים הנמצאים בחדר work ומחזיק את השם המשתמש הנמצא בחדר עם ה – socket שלו. שם המשתמש הוא המפתח וה – socket ערכו.

מילון clients_room_talking – מילון המאחסן את המשתמשים הנמצאים בחדר talking ומחזיק את השם המשתמש הנמצא בחדר עם ה – socket שלו. שם המשתמש הוא המפתח וה – socket ערכו.

מילון clients_room_dating – מילון המאחסן את המשתמשים הנמצאים בחדר dating ומחזיק את השם המשתמש הנמצא בחדר עם ה – socket שלו. שם המשתמש הוא המפתח וה – socket ערכו.

מילון clients – מילון המכיל את שם המשתמש המחובר עם ערך ה – socket של ההתחברות לשרת כאשר שם המשתמש המפתח וה – socket ערכו.

מילון client_seeds – מבנה נתונים המכיל נתונים בין היתר ה – socket שהוא המפתח, המפתח הציבורי מודול וה – SEED עם האובייקט SetSeed מתאר את ערכו של המפתח של המילון.

מילון clients_users – המפתח הוא שם המשתמש וערכו הוא שאר נתוניו של שם המשתמש בין היתר שם פרטי, שם משפחה והסיסמא המוצפנת של אותו משתמש.

מילון client_password_key – המפתח הוא שם המשתמש וערכו המפתח של הסיסמא

עץ מודולים:

לקוח:



שרת:

database_mengment.py

```
import mysql.connector
from cryptography.fernet import Fernet
import datetime
```

server_chat.py

```
import os
import random
import socket
import threading
import time
#from click import style
import my_functions
from opt_functions import *
from database_management import *
from cryptography.fernet import Fernet
import logging
import detectEnglish
import hashlib
import hmac
import pickle
import imutils
import struct
import cv2
```

מודלים שבהם הלקוח והשרת עושים שימוש:

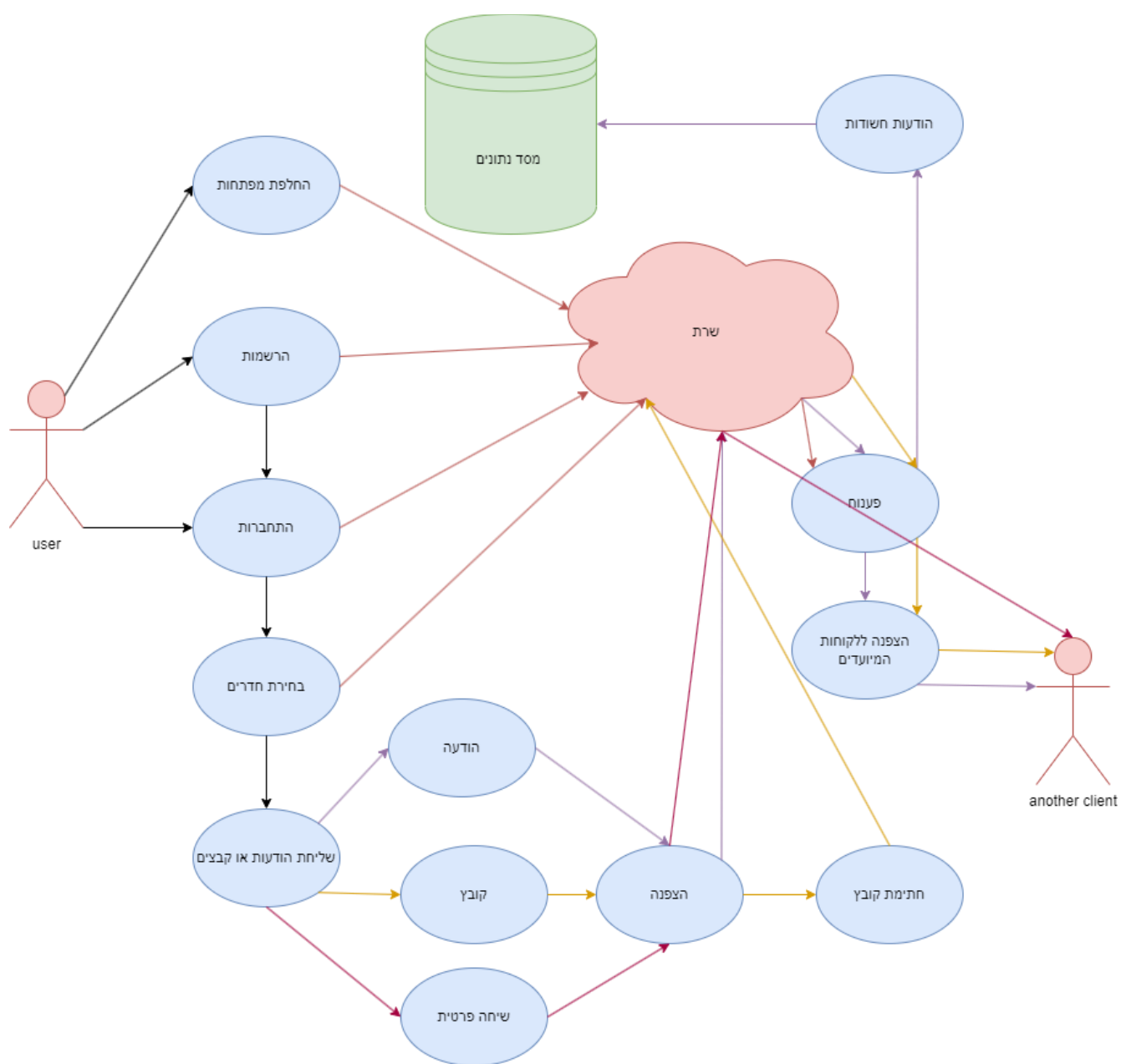
My_functions.py

```
import decimal
import math
from decimal import *
```

otp_functions.py

```
import random
```

תרשים use-case:



תיאור המחלקות:

קובץ client_chat.py :

קובץ אחראי על הכול הוא מהווה את כל האלגוריתמים שהלקוח עושה שימוש כולל לוגיקה וארגון המידע בין השרת ללקוח הקלט שהקובץ מקבל הוא מהמשתמש ממשק הגרפי של המערכת ובנוסף מה – socket . הפלט שהיא מחזירה חלקם עבור ממשק הגרפי וחלקם עבור השרת.

קובץ client_start_gui.py :

תפקידו של הקובץ הזה הוא יצירת חלון ההתחלה הוא ממשק משתמש עבור הלקוח והוא קורא לפונקציית start_func שהיא בעצם מתחילה את ההתחברות לשרת כולל החלפת המפתחות. הקלט שלה הוא מורכב משניים האחד יצירת משתמש והתחברות למערכת והפלט שלה הוא הפעלת תהליך החלפת המפתחות כולל פלט בהתאם למה שהלקוח לחץ התחברות / הירשמות.

קובץ client_create_account_gui :

התפקיד של הקובץ הנ"ל הוא חלון ההירשמות למערכת, הוא ממשק גרפי המקבל נתונים מהמשתמש ושולח את נתוניו לשרת בעזרת הפונקציה create_account_func מקובץ client_chat.py. בין היתר קלטיו יהיו שם פרטי, שם משפחה, שם משתמש וסיסמא ובנוסף מקבלת מהשרת האם ההירשמות בוצע בהצלחה או לא. הפלט שלה הוא פתיחת דף ההתחברות. לחיצה על כפתור ה – submit ישלח את נתוניו של המשתמש לשרת.

קובץ client_login_gui.py :

יחידה זאת היא מהווה דף ההתחברות למערכת, היא מהווה ממשק גרפי למשתמש ושליחה של הנתונים שהקליד לשרת אחרי לחיצת הכפתור submit והיא עושה זאת בעזרת פונקציית login_func מקובץ client_chat. הקלט שלה מורכב משניים האחד הוא שם משתמשו של הלקוח וסיסמתו הפלט שלה הוא האם ההתחברות בוצעה בהצלחה או לא על ידי השרת.

קובץ client_select_room_gui.py :

הקובץ פותח חלון של בחירת חדרים ובחלון זה יהיה 3 כפתורים כל כפתור הוא חדר אחר, חלון זה הוא ממשק גרפי עבור המשתמש והוא נועד לשלוח את הקלט שקיבל על ידי המשתמש ושליחת החדר הנבחר על ידי המשתמש לשרת על ידי פונקציה select_room_func מ – client_chat.py.

קובץ client_gui:

זהו קובץ מאוד חשוב ומורכב הוא מהווה דף דיאלוג במערכת, הוא חלון ממשק למשתמש והוא מורכבת משני חלקים האחד הוא שליחת הנתונים שהקליד המשתמש על ידי הכפתור sent על ידי הפונקציה session_func מ – client_chat.py והשני הוא קבלת הודעות ממשתמשים אחרים הנמצאים בחדר או מאדם שמתנהל איתו שיחה פרטית על ידי הפונקציה recive_ongoing_msg_from_chat_server_func_gui מ – client_chat.py שמופעלת על ידי thread.

קובץ database_management.py:

הקובץ מהווה גשר בין השרת למסד הנתונים ומכיל שאילתות, עדכון, מחיקה של בסיס הנתונים. הקלט שלה הם נתונים הניתנים על ידי השרת והכנסתם למסד הנתונים כמו למשל לטבלת client_seeds כמו כן לעוד טבלאות. הפלט שלה הוא עדכון הטבלאות במסד הנתונים ושיחת הודעה בהתאם לשרת האם הפעולה שבוצעה או לא.

קובץ detect_english:

תפקידו של הקובץ זיהוי הודעות חשודות או אסורות ולאחר מכן תיעוד ההודעות במסד הנתונים על ידי database_management.py. הקלט שלה הוא תוכן של קובץ מילים חשודות suspicious_words_dictionary.txt. הפלט הוא מילון של כל המילים החשודות, כדי שיהיה ניתן לבדוק האם בנתונים שהלקוח העביר הייתה מילה חשודה ולתעד אותה.

קובץ my_functions.py:

הקובץ משמש לקליטת הנתונים מן הלקוח באמצעות socket וזיהוי מספרי prime כלומר ראשוניים וטיפול בהצפנה ובפענוח של ה – RSA. הקלט שלה הוא ה – socket שממנה היא קיבלה מידע ואת הפרמטרים להצפנה/פענוח של ה – RSA. הפלט הוא הצופן או הפענוח של ה – RSA והנתונים שקיבלה מה – Socket.

קובץ otp_functions.py:

תפקידו של הקובץ הוא טיפול בהודעות כגון פיענוח והצפנה של הנתונים שעוברים ברשת בעזרת OTP. כלומר הוא מייצר את ה – pads המיועדים להצפנה או פענוח.

הקלט שלה הוא ה – seed ההתחלתי ואתחול האובייקט שמייצר כל פעם את ה – pads לפי אורך ה – data שצריך להצפין אותו.

הפלט הוא ה – pads שצריך כדי להצפין או לפענח את ה – data ומחזירה אותו למי קרא לו זה יכול להיות השרת או הלקוח.

קובץ server_chat.py:

זהו השרת יחידת המרכז של כל המערכת, המערכת לא עובדת בלעדיו, כל לקוח המתחבר לשרת וכל המידע בין הלקוחות עובר דרכו, בעיקרון הוא יחידה מרכזית במערכת הוא יודע על כל מידע שעובר דרכו חוץ מהשיחות הפרטיות.

כאשר לקוח מתחבר למערכת כלומר לשרת הוא יוצר thread לטיפולו כך שיהיה אפשר לעבוד עם הרבה לקוחות בלי בעיה. הקלט של הוא מידע ונתונים בין היתר הודעות וקבצים ושידור סטרים המתקבלים על ידי הלקוחות. הפלט של היא שליחת הנתונים ליתר הלקוחות המיועדים לקבל אותם במערכת בנוסף עדכון בסיס הנתונים.

ארכיטקטורת המערכת:

צד השרת:

בפרויקט שלי השרת כפי שציינתי ממוש באמצעות Socket בפרוטוקול TCP , אליו כל הלקוחות מתחברים בעת שהשרת מורץ על מנת להריץ את השרת נריץ את קובץ server_chat.py , והוא משתמש ועושה שימוש בכמה מודלים שיצרתי כגון: my_functions.py כדי להיעזר בתהליך החלפת המפתחות שנעשה על ידי אלגוריתם RSA ולקבל מידע מהלקוח באמצעות הפונקציה recvall.

בנוסף השרת משתמש במודול otp_function.py שאחראי לכל המידע המוצפן והמפוענח כלומר המודול מקבל מידע ומצפין או מקבל מידע מוצפן ומפענח אותו בנוסף הוא גם יוצר את ה – pads להצפנת OTP.

השרת משתמש גם במודול הנקרא database_mangement.py שאחראי לכל הכנסת המידע, עדכון המידע וגם למחיקת המידע במסד הנתונים.

צד הלקוח:

הלקוח במערכת מהווה חלק מרכזי במערכת שלי וצריך להיות ידידותי למשתמש ולכן בנית אותו כממשק גרפי באמצעות tkinter של פייתון היא סיפרייה בתוך פייתון המיועדת לשימוש גרפי שמאפשר ליצור חלונות, כפתורים, טקסטים, כתורות ועוד.

החלק המרכזי בצד הלקוח הוא client_chat.py שאת הפונקציות הנמצאות שם מפעיל ממשק הגרפי על ידי המשתמש. על מנת לעלות את הלקוח נריץ את הקובץ client_start_gui.py ששם נקרא לפונקציה בשם start_func מ – client_start_gui.py שתנהל את תהליך התחברות לשרת ותהליך החלפת המפתחות.

לאחר מכן יפתח מסך הרשמות או התחברות על ידי הפונקציות create_account_func ו – login_func שיממשו ב – client_start_gui.py אם הכל בוצע בהצלחה יופעל חלון בחירת החדרים על ידי הפונקציה select_room_gui.py ואז לבסוף יפתח החלון הסופי והמרכזי שלי שהוא חדר הצא'ט שיפעיל שני פונקציות אחת בטרייד ששמה start_recice שתפקידו לקבל פקודות מהשרת בזמן אמת והשנייה session_func שתפקיד להעביר מידע לשרת.

כמו בצד השרת שהוא משתמש במודלים כגון otp_functions.py ו – my_functions.py גם הלקוח עושה בהם שימוש.

תיאור התוכנה:

סביבת העבודה שעשיתי בה שימוש היא PycharmPyCharm Community Edition 2023 שלדעתי מאוד נוחה וידידותית למשתמש והיא הטובה ביותר לתכנות בשפת python היא נותנת עצות ועזרה ובנוסף השלמת קוד בין היתר גם debug מצוין ואולי אחד הטובים שראיתי.

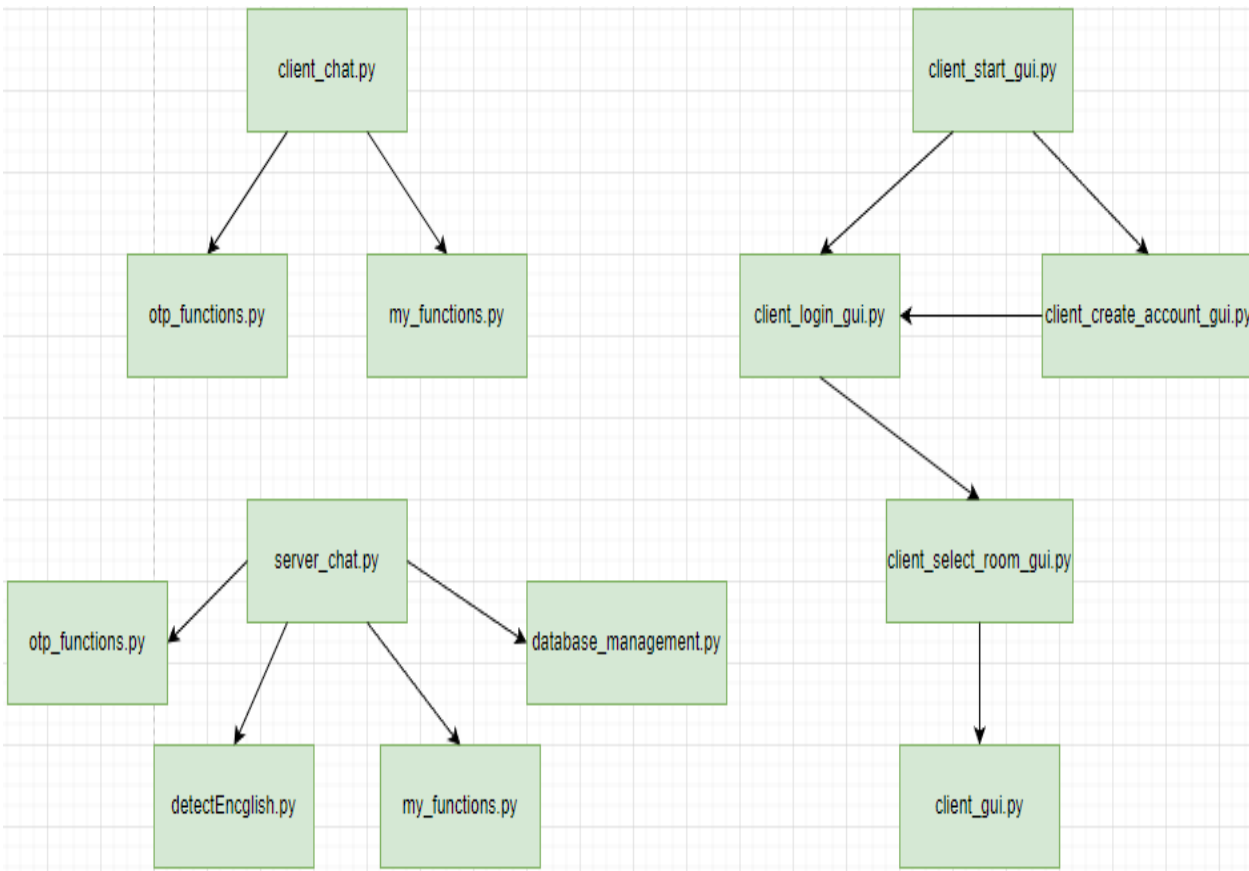
השתמשתי גם בתוכנה MySQL Workbench לשרת ה – database.

שפות התכנות:

שפת התכנות היא כמובן Python. הכרתי אותה שנה שעברה ומיד ידעתי שהפרויקט גמר שלי יהיה בשפה הזאת היא מאוד נוחה ומאוד אנגלית הכוונה שיש בה הרבה מילים שמורות באנגלית למשל print בניגוד ל – console.writeline c# שקצת קשה לזכור, בפיתון יש מלא אפשרויות ובנוסף תמיד רציתי ללמוד אותה. הפרויקט נתן לי ללמוד את פיתון לעומק ועכשיו אני בקי בפיתון ובמה שהיא מציע למשתמש.

מדריך למשתמש:

עץ קבצי המערכת:



תיאור הליך התקנת המערכת:

המערכת מתוכננת בסביבת עבודה python בתוכנת pycharm המערכת עובדת על כל מחשב בסביבת עבודה של ווינדוס המערכת בנוסף עושה שימוש במסד נתונים מסד הנתונים של המערכת היא MYSQL שצריכה להיות מורדת בצד של השרת ומיותר לציין שהסכמה והטבלאות צריכות להיות בצד השרת.

עוד דבר חשוב הוא שינוי כתובת ה IP – אצל השרת ושינוי ה IP- אצל הלקוח אשר השרת שרוצה להתחבר אליו.

ובנוסף הוא התקנת ה – packages שדרושות המערכת בין היתר cv2, random, socket ועוד.

משתמשי המערכת:

משתמש קצה:

המשתמש קצה יהיה הלקוח הוא שהוא עושה שימוש בקבצים הבאים:

- Client_chat.py
- Otp_functions.py
- My_functions.py
- Client_start_gui.py
- Client_login_gui.py
- Client_create_account_gui.py
- Client_selecet_room_gui.py

על מנת להריץ את הלקוח המשתמש קצה יצטרך להריץ את client_start_gui.py והתוכנה תעשה את שלה היא תדע מתי לקרוא לקבצים אחרים.

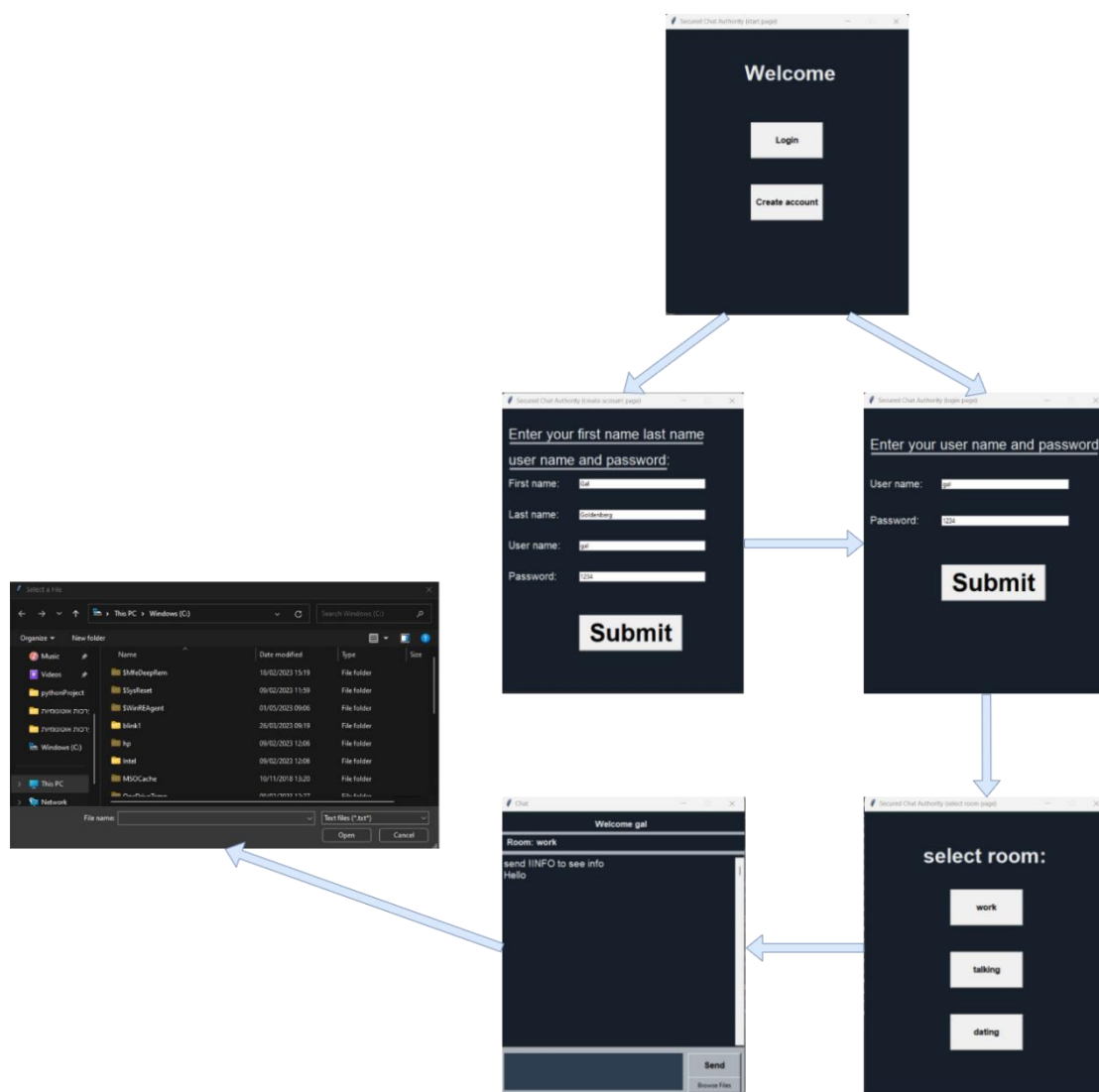
מנהל המערכת:

מנהל המערכת הוא השרת שהוא עושה שימוש בקבצים הבאים:

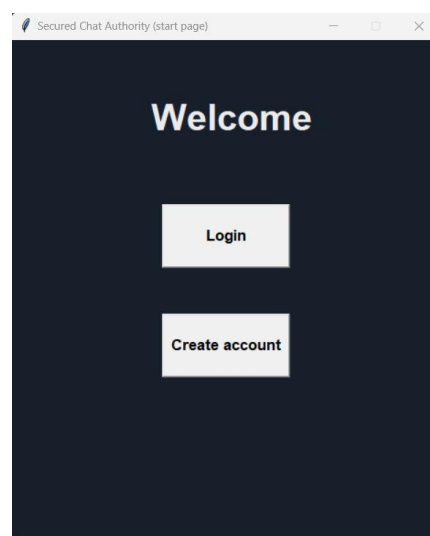
- Server_chat.py
- Otp_functions.py
- My_functions.py
- Database_manegment.py

על מנת להריץ את השרת, השרת צריך להריץ את הקובץ server_chat.py והתוכנה תדע מתי לקרוא לפונקציות מקבצים אחרים ולעשות בהן שימוש.

אופן הפעלה המערכת:

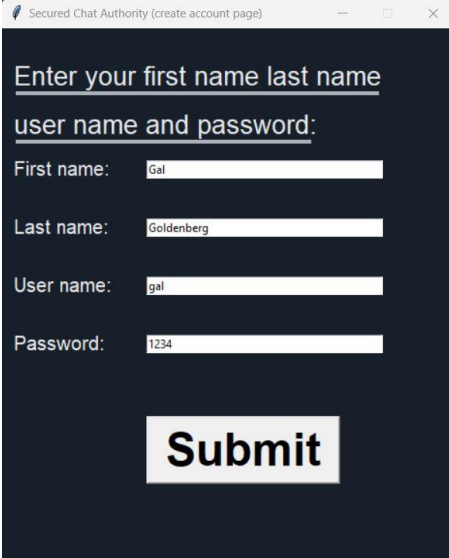


מסך פתיחה:



מסך הפתיחה נותן למשתמש בחירה של הירשמות למערכת או התחברות למערכת. המסך מורכב משני חלקים שני כפתורים login וכפתור create account שכאשר המשתמש ילחץ על אחד משניהם יתרחש תהליך ההתחברות או ההירשמות בהתאם למה שלחץ המשתמש.

מסך ההירשמות:



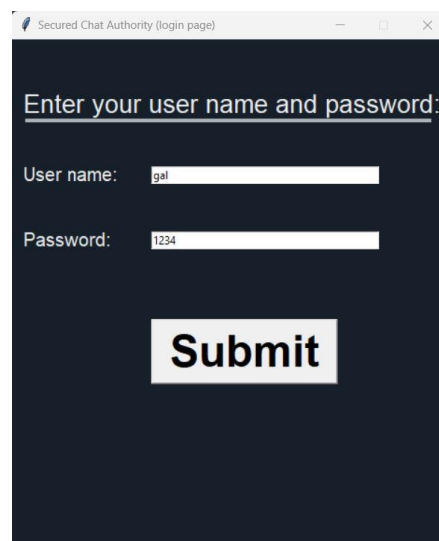
The screenshot shows a web browser window titled "Secured Chat Authority (create account page)". The page has a dark blue background. At the top, it says "Enter your first name last name" and "user name and password:". Below these are four input fields: "First name:" with the value "Gal", "Last name:" with the value "Goldenberg", "User name:" with the value "gal", and "Password:" with the value "1234". At the bottom of the form is a large white button with the text "Submit" in black.

חלון זה הוא חלון יצירת משתמש במערכת שימש להתחברות עתידית למערכת עם הנתונים שהקליד המשתמש בין היתר המסך מכיל ארבע נתונים:

- שם פרטי
- שם משפחה
- שם משתמש
- סיסמא

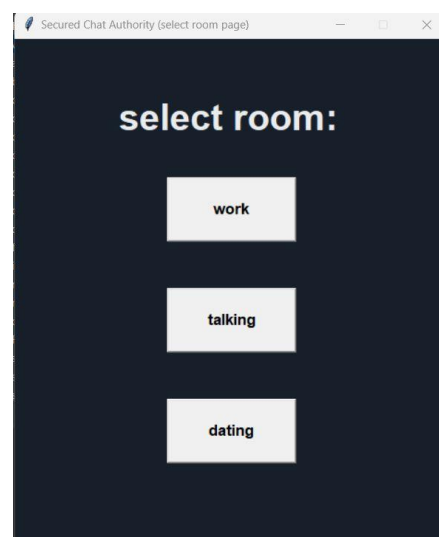
כפתור ה – submit מעביר את הנתונים לשרת ולאחר מכן יפתח מסך ההתחברות אם הכל בוצע בהצלחה, אם לא המשתמש יצטרך להירשם שוב שנית.

מסך התחברות:



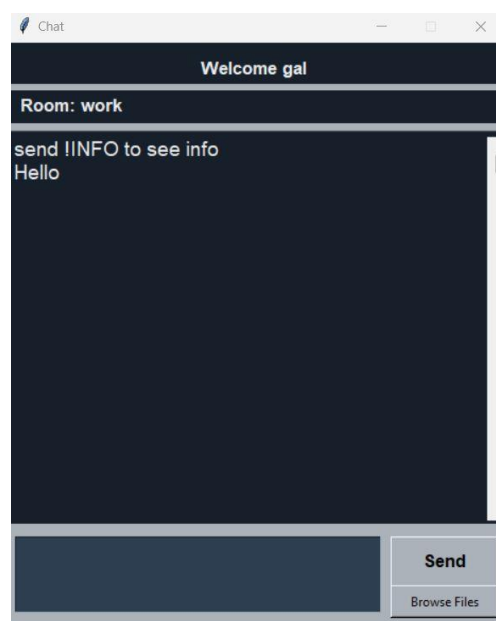
חלון זה הוא חלון העוסק בתהליך ההתחברות של המשתמש למערכת שיצר בעבר. המסך מכיל שתי נתונים וביניהם שם משתמש וסיסמא לזיהוי המשתמש, וכפתור submit שיגרום לנתונים להישלח לשרת ועיבודם שם ואם הכל בוצע בהצלחה אז נעבור למסך של בחירת החדרים.

מסך בחירת החדרים:



מסך זה מספק למשתמש לבחור את החדר שיחה שברצונו להשתתף בו. במערכת ישנו 3 חדרים וביניהם חדר work, חדר talking ו- dating. כל לחיצה משלושת החדרים יוביל אותנו לחדר הנבחר ותעביר למסך ההשתתפות של החדר.

מסך הצא'ט:



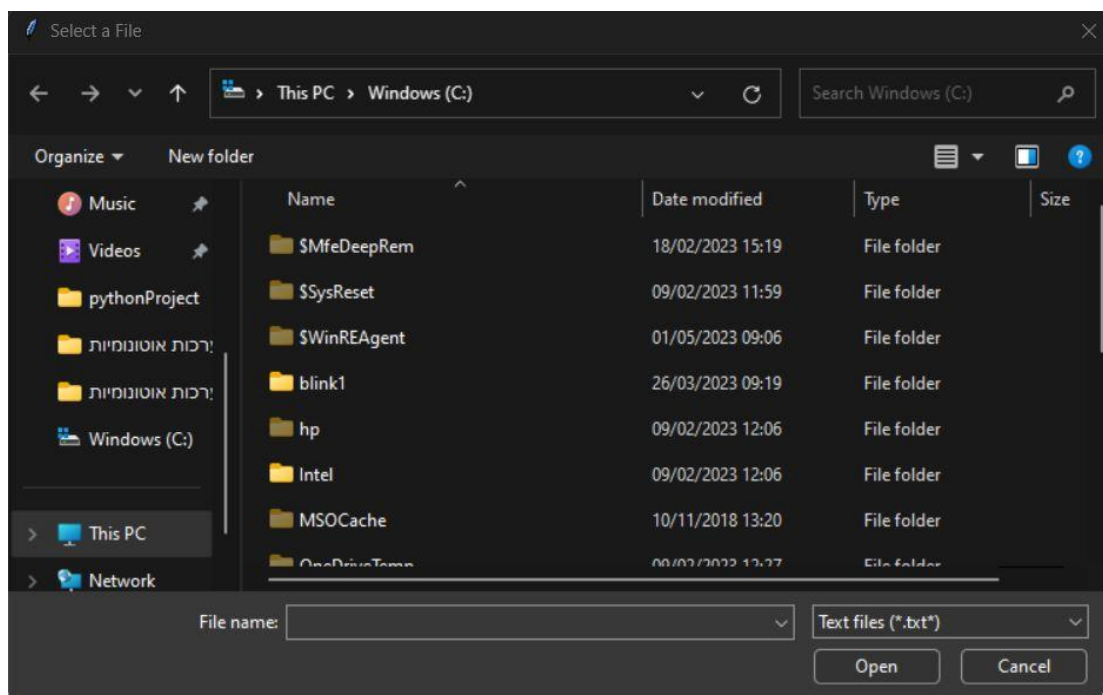
מסך זה הוא המסך המרכזי של המערכת כל ההודעות שהלקוחות אחרים בחדר שלחו בחדר יופיעו פה בצורה יפה וגרפית, בנוסף אפשר גם לשלוח הודעות ללקוחות הנמצאים בחדר.

מסך זה מכיל 2 כפתורים ביניהם send שלוקח את מה שהקליד המשתמש בתיבת הטקסט ושולח לשרת והשרת מציץ את ההודעה לשאר חברי המשתמשים הנמצאים בחדר ובנוסף ישנו כפתור בחירת קובץ לשליחה לשאר חברי השיחה בחדר.

בחלון זה אפשר לכתוב פקודות ולקבל מידע במערכת להלן הפקודות:

- אם נקליד '1' המערכת נוציא את הלקוח מהמערכת ואם הוא בשיחה פרטית אז המערכת תוציא אותו מהשיחה הפרטית.
- אם נקליד '***delete_account***' המערכת תמחק את המשתמש מהמערכת וממסד התונים.
- אם נקליד 'online!' המערכת תציג לנו את כל המשתמשים המחוברים למערכת כרגע.
- אם נקליד 'room online!' יציג לנו את כל המשתמשים המחוברים למערכת בחדר שאנחנו נמצאים בו כעת.
- אם נקליד 'profile!' ואז את שם המשתמש המערכת תציג לנו את הנתונים על המשתמש ביניהם שם פרטי שלו ושם המשפחה שלו אם לא קיים משתמש כזה תוצג הודעה בהתאם.
- אם נקליד 'private!' ואז את שם המשתמש המערכת תפתח בתהליך החלפת המפתחות מקצה לקצה בשיחה פרטית.

חלון בחירת הקובץ לשליחה:



חלון זה עוסק בחלון פתיחת הקבצים שליחה בחדר ומתחיל בתהליך ה – HMAC במערכת על מנת לאמת שהקובץ הנשלח הגיע מאיתנו ולא מגורם זר.

סיכום אישי:

תיאור תהליך העבודה הצלחות ואתגרים:

במהלך העבודה נתקלתי בקשיים רבים כמו באגים כגון בעיות קומפילציה ובעיות זמן ריצה הייתי צריך לחקור רבות ברחבי האינטרנט על מנת לפתור את אותם באגים. אחד הבאגים שהיו לי זה רבות זה ההמרת אובייקט לאובייקט אחר מכיוון שאני לא רגיל לתכנת בפייטון היה לי ידע בסיסי בפייטון אבל היה לי תמיד מוזר אני שלמשתנים אין טיפוס כלומר אין int לפני כל מספר לדוגמא אבל על קושי זה התגברתי על ידי למידת פייטון ואיך משתמשים במשתנים כמו שצריך עוד קושי זה נוסף שהיה לי הוא אלגוריתם החלפת המפתחות בין משתמשים הרוצים להשתמש בשיחה פרטית וארגון המידע והעברת ההודעות ביניהם, קושי זה לקח לי זמן לפתור ופתרתי אותו על ידי מחקר באינטרנט ותמיכה של המרצה גל בראון שכיוון אותי לפתרון.

מה נלמד מהפרויקט:

למדתי על הפרויקט רבות בחיים שלי לא פיתחתי פרויקט ברמה הזאת שיפרתי את החשיבה האלגוריתמית שלי עישרתי את הידע שלי בקריפטוגרפיה נושא שאני ממש אוהב ובנוסף הכי חשוב נהנתי מהלמידה על נושאי הפרויקט.

למדתי מהי הצפנה אסימטרית ולמה היא שימושית בנוסף איזה אלגוריתמים עושים שימוש בהצפנה אסימטרית, מה זה HMAC ולמה הוא שימושי לעולם של היום, מימוש GUI בשפת פייטון על ידי שימוש ב – tkinter, שימוש נכון ב – threads.

במילים אחרות למדתי רבות והכי חשוב בניגוד למבחן שלומדים אליו וישר שוכחים את מה שהיה במבחן יום אחרי, פרויקט הוא משהו גדול שבחיים לא נשכח משהו שהולך איתנו לכל החיים אני תמיד אומר שפרויקט אפשר ללמוד הרבה בניגוד למבחנים לפחות לפי דעתי.

תובנות ומסקנות:

אחד התובנות והמסקנות שלמדתי הוא לא לוותר בחיים גם אם קשה לנסות בכל הכוח עד שנגיד להצלחה כי בסך הכל הצלחה בונה את האדם ולא רק הצלחות בונות את האדם אלה גם כישלונות שאנחנו בני האדם תמיד ננסה לצאת מהם עוד תובנה שהגעתי אליה בפרויקט הוא בחיים לא להגיד זה לא אפשרי אין דבר כזה לא אפשרי.

כלים להמשך:

קיבלתי המון כלים להמשך הדרך במהלך הפרויקט כלים שהם ממש שימושיים לעולם של היום למשל HMAC נמצא בכל רשת חברתית אם נשלח קובץ ב – whatsapp נשים לב שיש שימוש באלגוריתם של HMAC וגם ב – discord – zoom כל רשת חברתית שמכבדת את עצמה יש אלגוריתמי הצפנה ואימות נתונים על מנת לשפר את האבטחה של המערכת.

ביבליוגרפיה:

RSA algorithm:

<https://www.youtube.com/watch?v=4zahvcJ9glg&t=2s>

<https://www.youtube.com/watch?v=oOcTVTpUsPQ>

[https://en.wikipedia.org/wiki/RSA_\(cryptosystem\)](https://en.wikipedia.org/wiki/RSA_(cryptosystem))

OTP encryption:

https://en.wikipedia.org/wiki/One-time_pad

<https://www.youtube.com/watch?v=FIIG3TvQCBQ>

HMAC:

<https://en.wikipedia.org/wiki/HMAC>

<https://www.youtube.com/watch?v=wISG3pEiQdc>

GUI python tutorial:

<https://www.youtube.com/watch?v=YXPyB4XeYLA>

PRNG:

https://en.wikipedia.org/wiki/Pseudorandom_number_generator

