**מעקב פרויקט בדפנסיבי**

**שרת בפייתון**

1. ~~להתחיל Template לשרת בסיסי רצח, שמתחיל בפונק' main.~~
2. ~~לעשות שהוא יעבוד~~ **~~בריבוי משתמשים~~** ~~ע"י threads או selector.~~
3. ~~גרסת השרת תהיה 1~~ (אם אני לא משתמש בSQL).
4. ~~לקרוא מקובץ, להיעזר בדרישות ובפונק' שכבר עשיתי בממן 14, לשים לב לדרישה אם הקובץ לא קיים.~~
5. ~~ליצור מחלקה לבסיס נתונים, עם פונק' בסיסיות – לפי מה שדרוש.~~
6. עשיתי מחלקה של בקשה, שמקבלת בקשה ומפרסרת אותה:  
   א' לוודא שעשיתי כמו שצריך ושהנתונים נאספים בתצורה הנכונה (אולי צריך להמיר חלק למספר / מחרוזת או משהו בסגנון? או בכלל לא להמיר? לעבור בדיוק על איך עשיתי ב14 ולתקן בהכול!!!! (גם בrequest וגם בmessage).  
   ב' ~~לכתוב פונק' שממשיכה לפרסר כמו שצריך את הmessages~~
7. ~~לכתוב פונקציות המשך לכל אחת מסוגי הבקשות שם, לפי מה שעשיתי בTODO, שממש עושות כל אחד מהדברים כמו שדרוש.~~
8. ~~לעשות את הכנ"ל רק עם ההודעות^^^^^^^~~
9. ~~להוסיף בדיקה שהusername הוא לפי הפורמט איפה שהוא.~~
10. להוסיף בדיקה שכל המספרים הם תקינים, לפי מה שעשיתי בממ"ן 14.
11. לוודא האם כדאי לי לשמור את השם משתמש בבסיס נתונים כTEXT או varchar(255).
12. להוסיף מחלקה של response (או מחלקות ויורשות וכו?) ולהתחיל להכין את התגובות בחזרה, **לוודא** שאני שולח את הכול כמו שצריך, עם padding של \0 לstr שקטן מ255 (למשל שם משתמש).
13. לעשות Unit test לדברים האלו איכשהו.

**Client בCPP**

1. להתחיל להכין את המחלקות במבנה דומה לשרת, ובהשראה למה שעשיתי ל-14.
2. להכין את הממשק הכללי והלולאה הכללית, עם כל הפונק' הדרושות.
3. להתחיל לעבוד על ההצפנה!
4. כשאני מכין את הבקשה מסוג הודעה, לזכור להוסיף **גם** לגודל של שpayload **וגם** לשל הmessage content size את הגודל של ההודעה!
5. כשאני בונה את הקוד שמקבל את רשימת הלקוחות, ליצור וקטור זמני שמכיל את הclass של clientlistresponse **או** שאולי עדיף בעצם לעשות רק לattributes של זה, בלי כל הversion,request וכו... – קיצר לעבוד על זה.
6. לכל החלק של ההתחברות עם הרשת ובדיקת קובץ וכו – לעבור שם **כמו** שצריך- לסדר את הפונקציות, מתי אני מתחבר לרשת, להוסיף ניהול שגיאות, ולהעביר את **כל** הנתונים כמו שצריך.

**הצפנה כללי**

-**כל ההצפנות והפענוחים קורים אצל הCPP, אז בקיצור – להתחיל CPP!!!!**  
**החלפת הודעות**

1. לקוח A מבקש מהשרת את המפתח **הציבורי** של לקוח B. - **בקשה רגילה 602.**
2. לקוח A שולח הודעה (דרך השרת) ללקוח B מסוג "בקשת מפתח הצפנה סימטרי". ההודעה מוצפנת ע"י המפתח הציבורי של B. – **בקשת הודעה 603 -> בקשת מפתח סימטרי 1.**
3. השרת מקבל את ההודעה ושומר אותה. – **שומר את הבקשה בDB, כרגיל.**
4. לוקח B מושך מהשרת את ההודעות הממתינות לו. – **בקשת שליפת הודעות 604.**
5. לקוח B מפענח את ההודעה באמצעות המפתח **הפרטי** שלו. – **קורה אצל הלקוח.**
6. לקוח B מבקש מהשרת את המפתח **הציבורי** של לקוח A. – **בקשה 602.**
7. לקוח B שולח **תשובה** מסוג "מפתח הצפנה סימטרי" ללקוח A. התשובה מוצפנת ע"י המפתח הציבורי של A. – **בקשת הודעה 603 -> שליחת מפתח סימטרי 2.**
8. השרת מקבל את ההודעה ושומר אותה. – **כרגיל.**
9. לקוח A מושך מהשרת את ההודעות הממתינות לו. – **בקשה 604.**
10. לקוח A מפענח את ההודעה באמצעות המפתח **הפרטי** שלו. – **קורה אצל לקוח A.**
11. כעת, לקוח A ולקוח B יכולים לשוחח באמצעות מפתח הצפנה **סימטרי**. – **להעמיק עוד.**

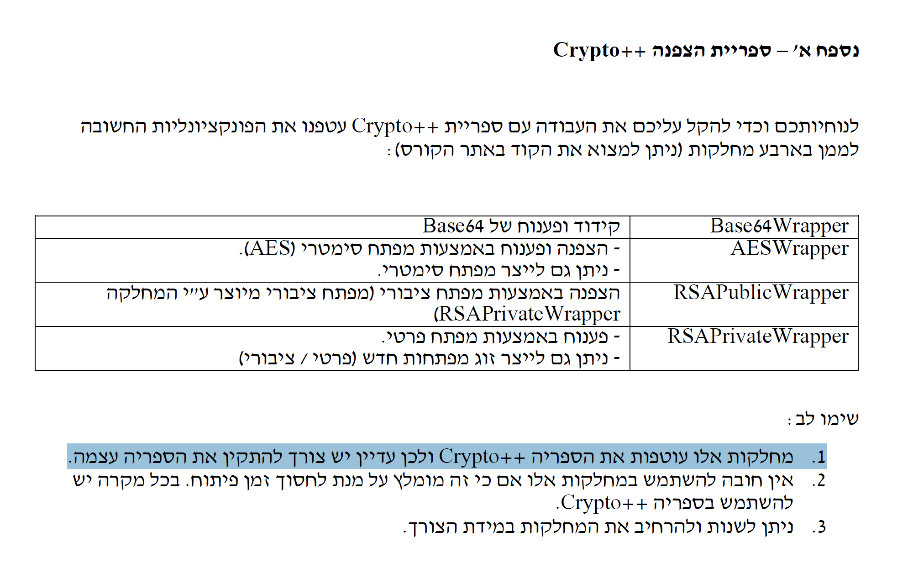
**גודל כל אחד מסוגי המפתחות:**

מפתח סימטרי = 128 סיביות (bits), כלומר – 16 בתים.  
מפתח א-סימטרי = 1024 סיביות, כלומר – 128 בתים.

מפתח ציבורי = 160 בתים.

תמונה שמכילה טקסט, צילום מסך, גופן, מסמך

התיאור נוצר באופן אוטומטימפתח פרטי = **לא רלוונטי?**



**חולשות אפשריות:**

שרת

* קלט שמגיע בגדלים שונים
* קלט שמגיע בפורמט אחר ומנסים לפענח אותו
* קלט שמגיע קצר מדי ולא בודקים את זה
* Payload/message\_content שמגיע עם size לפני, ולא מוודאים שה-size והגודל ממש של הפיילואד שווים.

לקוח