PATH ORAM

Yosef Revivo

https://github.cs.huji.ac.il/yosef-revivo1/pathorm

מצורף פרויקט path Oram המכיל קליינט וסרבר כך שלקליינט יש יכולת לשמור מידע על גבי השרת כאשר השרת הינו untrusted, דרישות המערכת הן שלשרת לא תהיה כל יכולת לראות לשנות או ללמוד את ה access pattern של המשתמש.

אתחול

בשביל לאתחל את הפרויקט לאחר התקנת NODE בסביבת העבודה כל שנדרש הוא – לפתוח CMD או TERMINAL בתיקייה הנוכחית, ואזי להריץ

1. Npm install
2. Npm start
3. Cd client
4. Npm start

עיצוב ותכנון

1. סרבר – הסרבר מאותחל עם מערך ריק, ומאפשר ב API בסה"כ שתי פעולות **כתיבה** לאינדקס במערך **וקריאה** מאינדקס במערך.
   - ✓ Framework – node js with express
2. קליינט – כלל הלוגיקה ממומשת בקליינט על מנת למנוע אפשרות לשרת לדעת מידע כלשהו. כלומר הקליינט מתייחס למערך בזיכרון של הסרבר כאל עץ בינארי BST.

   כעת, בפעולת access עבור פעולת קריאה\כתיבה אנחנו נפעל כדלקמן -

   A. נשאב את מספר העלה והבלוק המתאים מה position map.
   B. נבצע קריאה של כל ה path מהשורש עד לעלה.
   C. נפענח את המידע.
   D. נכתוב\נקרא ל\את הבלוק המתאים.

E. נפעפע את הבלוק שהשתמשנו בו לשורש ונמפה אותו לעלה חדש.

F. נצפין מחדש את ה path.

G. נכתוב את כולו חזרה לסרבר.

השתמשתי בפונקציות המובנות של js עבור כל הפעולות הקריפטוגרפיות, ובנוסף הנחתי (וכתבתי למשתמש) שבפעולת מחיקה המידע נמחק לוקלית אצל הקליינט, וזה לא משנה כי אין הבחנה לסרבר בין בלוק שהוא dummy לבלוק המכיל מידע.

✓ Framework – react with typescript, mantine for ui/ux and crypto package.

multithreading system

שפת JS היא שפה שלא עובדת עם כמה threads אלא משתמשת באחד ויחיד, ולכן לא יכולתי להשתמש בכמה threads במימוש שלי ל path oram, במימוש אחר שכן היה משתמש ב multithreading, אכן הייתה יכולה להיות הטבה משמעותית בזמני הריצה למשל בעת הקריאה מהסרבר, הייתי יכול לקרוא בבת אחת את כל path ולחלק את העבודה בין threads שונים, כך שהזמן היה מתקצר משמעותית.

בנוסף באותה מידע בעת ההצפנה והפיענוח של ה path ובכלל בכל פעולה כזאת שצריכה להתבצע באופן בלתי תלוי על כל ה path הייתי יכול להשתמש ב multithreading בשביל לעשות את כל הפעולות בו זמנית.

performance

db size	m_sec	seconds
3	952.9	1.0
4	1185.9	1.2
5	1497.3	1.5
6	1785.0	1.8
7	2095.4	2.1
8	2384.5	2.4
9	2656.9	2.7
10	2959.1	3.0
11	3267.1	3.3
12	3560.5	3.6