**האוניברסיטה הפתוחה**

**המחלקה למתמטיקה ולמדעי המחשב**



**הבטחת מהימנות הנתונים, מקורם ושלמותם במסדי נתונים במיקור חוץ**

עבודה מסכמת זו הוגשה כחלק מהדרישות לקבלת תואר

"מוסמך למדעים" M.Sc. במדעי המחשב

באוניברסיטה הפתוחה

החטיבה למדעי המחשב

על-ידי

**סלופסקי ערן**

העבודה הוכנה בהדרכתו של פרופ' אהוד גודס

יולי 2017

תוכן עניינים

[רשימת איורים............................................... 3](#_Toc500436028)

[רשימת טבלאות............................................... 4](#_Toc500436029)

[תקציר............................................... 5](#_Toc500436030)

[מבוא............................................... 6](#_Toc500436031)

[1 סקירה מפורטת............................................... 7](#_Toc500436033)

[1.1 שיטות מבוססות חתימות - Authentication and Integrity in Outsourced Databases [‎1] 7](#_Toc500436034)

[1.1.1 הקדמה 7](#_Toc500436035)

[1.1.2 המודל 7](#_Toc500436036)

[1.1.3 הנחות יסוד 8](#_Toc500436037)

[1.1.4 חתימות דיגיטליות - תקורות ותכונות רצויות 9](#_Toc500436038)

[1.1.5 סכמות איחוד חתימות 10](#_Toc500436039)

[1.1.5.1 Condensed RSA 10](#_Toc500436040)

[1.1.5.1.1 RSA סטנדרטי 10](#_Toc500436041)

[1.1.5.1.2 Condensed-RSA 11](#_Toc500436042)

[1.1.5.1.3 אימות Batch-י של חתימות RSA 11](#_Toc500436043)

[1.1.5.1.4 הבטיחות (Security) של Condensed-RSA 12](#_Toc500436044)

[1.1.5.1.5 המתווה 12](#_Toc500436045)

[1.1.5.1.6 פרטים 12](#_Toc500436046)

[1.1.5.1.7 טענה 1 13](#_Toc500436047)

[1.1.5.1.8 טענה 2 13](#_Toc500436048)

[1.1.5.1.9 השוואת תקורות 13](#_Toc500436049)

[1.1.5.1.10 Condensed-RSA in outsourced DB setting....................................... 13](#_Toc500436050)

[1.1.5.2 BGLS 14](#_Toc500436051)

[1.1.5.2.1 סכמת BGLS 14](#_Toc500436052)

[1.1.5.2.2 איחוד חתימות של BGLS 15](#_Toc500436053)

[1.1.5.2.3 ביצועים של BGLS 15](#_Toc500436054)

[1.1.5.2.4 BGLS in outsourced DB setting ....................................................... 15](#_Toc500436055)

[1.1.5.3 אימות Batch-י של חתימותDSA 15](#_Toc500436056)

[1.1.6 השוואת עלויות 16](#_Toc500436057)

[1.1.7 סיכום ועבודות עתידיות (למאמר זה) 18](#_Toc500436058)

[1.2 שיטות מבוססות עצים - Merkle Hash Tree based Techniques for Data Integrity of Outsourced Data ‏[6] 19](#_Toc500436059)

[1.2.1 הקדמה 19](#_Toc500436060)

[1.2.2 טכניקה בסיסית 19](#_Toc500436061)

[1.2.3 Merkle hash trees 20](#_Toc500436066)

[1.2.3.1 Merkle’s Signature Scheme 20](#_Toc500436067)

[1.2.3.2 עצי +B 21](#_Toc500436068)

[1.2.3.3 Data integrity based on Merkle hash tree 21](#_Toc500436069)

[1.2.4 Radix Path Identifier 22](#_Toc500436070)

[1.2.4.1 MHT storage in the database 24](#_Toc500436071)

[1.2.4.1.1 Single Authentication Table (SAT) 24](#_Toc500436072)

[1.2.4.1.2 Level Based Authentication Table (LBAT) 25](#_Toc500436073)

[1.2.4.1.3 השוואת ביצועים בין 2 הסכמות 26](#_Toc500436074)

[1.2.4.2 שליפת נתוני האימות 26](#_Toc500436075)

[1.2.4.2.1 Multi-join 26](#_Toc500436076)

[1.2.4.2.2 Single-join 27](#_Toc500436077)

[1.2.4.2.3 Zero-join 28](#_Toc500436078)

[1.2.4.2.4 Range - Condition 29](#_Toc500436079)

[1.2.4.3 Data Operations 30](#_Toc500436080)

[1.2.4.3.1 Select 30](#_Toc500436081)

[1.2.4.3.2 Update 30](#_Toc500436082)

[1.2.4.3.3 Batch update and optimization 30](#_Toc500436083)

[1.2.4.3.4 Insert ו- Delete 31](#_Toc500436084)

[1.2.5 סיכום ועבודות עתידיות (למאמר זה) 31](#_Toc500436085)

[1.2.5.1 סיכום 31](#_Toc500436086)

[1.2.5.2 עבודות עתידיות 32](#_Toc500436087)

[2 סקירה מקוצרת ........................................................................................................33](#_Toc500436088)

[2.1 Integrity for Join Queries in the Cloud [‎16] 33](#_Toc500436089)

[2.2 Guaranteeing Correctness of Bulk Operations in Outsourced Databases [‎18] 34](#_Toc500436090)

[2.3 Providing Database as a Service [‎19] ..............................................................................36](#_Toc500436091)

[2.4 Crowdsourced Data Integrity Verification for Key-Value Stores in the Cloud ‏ ‎‎[23] 38](#_Toc500436092)

[3 מימוש .....................................................................................................................39](#_Toc500436093)

[3.1 מגבלות והנחות יסוד .................................................................................................39](#_Toc500436094)

[3.2 הרצה.......................................................................................................................40](#_Toc500436095)

[סיכום ......................................................................................................................48](#_Toc500436096)

[ביבליוגרפיה .............................................................................................................49](#_Toc500436097)

[Abstract....................................................................................................................51](#_Toc500436098)

# רשימת איורים

[איור 1 - מודל של משתמש אחד (המשמש גם כבעל המידע וגם כצרכן המידע) 7](#_Toc500586462)

[איור 2 - מודל של בעל מידע אחד ומספר צרכני מידע 8](#_Toc500586463)

[איור 3- מודל של מספר בעלי מידע ומספר צרכני מידע 8](#_Toc500586464)

[איור 4 - Merkle hash tree 20](#_Toc500586465)

[איור 5 - עץ +B מסדר גודל 3 21](#_Toc500586466)

[איור 6 - עץ ה-MHT של טבלת העובדים 22](#_Toc500586467)

[איור 7 MHT עם Radix path identifiers 23](#_Toc500586468)

[איור 8 - עדכון רשומה 30](#_Toc500586469)

[איור 9 – דוגמא להרצת שאילתת join עם שכפול רשומות aו-c 33](#_Toc500586470)

[איור 10 - בניה של קבוצות (sets) והיחסים ביניהם 34](#_Toc500586471)

[איור 11 - זמן CPU ללא הצפנה, הצפנה ברמת השורה וברמת הדף 36](#_Toc500586472)

[איור 12 – בחירת גודל העץ חדש 40](#_Toc500586473)

[איור 13 - בניית העץ חדש 40](#_Toc500586474)

[איור 14 - בחירת מפתח חדש להכנסה לעץ 41](#_Toc500586475)

[איור 15 - לאחר הכנסה של מפתח לעץ 41](#_Toc500586476)

[איור 16 - לאחר הרצת הסקריפט להכנסת מפתחות 42](#_Toc500586477)

[איור 17 - לאחר מחיקת מפתח 70 43](#_Toc500586478)

[איור 18 - לאחר פעולת הכיווץ 44](#_Toc500586479)

[איור 19 - לאחר החלפת מפתח 40 במפתח 90 45](#_Toc500586480)

[איור 20 - הצגת ההיסטוריה 46](#_Toc500586481)

[איור 21 - שגיאת וריפיקציה 47](#_Toc500586482)

[איור 22 - לאחר החלפת מפתח 90 במפתח 100 על ידי גורם לא מורשה 47](#_Toc500586483)

# רשימת טבלאות

[טבלה 1- השוואת עלויות במודל של מסד הנתונים במיקור חוץ 17](#_Toc500586484)

[טבלה 2 - השוואת עלויות אימות וחתימה (ב-msec). t – מספר החתימות. k – מספר החותמים 17](#_Toc500586485)

[טבלה 3 - טבלת העובדים 22](#_Toc500586486)

[טבלה 4 - Single Authentication Table (SAT) 24](#_Toc500586487)

[טבלה 5 - טבלת קישור 25](#_Toc500586488)

[טבלה 6 - עובדים ברמה 2 (Emp\_2) 25](#_Toc500586489)

[טבלה 7 - עובדים ברמה 1 (Emp\_1) 25](#_Toc500586490)

[טבלה 8 –עובדים ברמה 0 (עלים) (Employee) 25](#_Toc500586491)

# תקציר

**מיקור חוץ (outsourcing)** - היא שיטת ניהול מודרנית, שבבסיסה עומד הרעיון להוציא מהארגון את תפעול הפעילויות שאינן נמצאות בבסיסו ולהעבירן לקבלן חיצוני, ולהותיר בארגון רק את תפעול פעילויות הליבה שלו. בשיטה זו יכול הארגון להשקיע את מירב המשאבים בתחום התמחותו, ולחסוך בעלויות הנגזרות מתפעול שאר הפעילויות בתוך הארגון.

העברת מסד הנתונים של הארגון לקבלן חיצוני היא דוגמא לפעילות כזו. העברה זו תחסוך לארגון את הצורך להעסיק מומחים בתחום ניהול מסדי הנתונים DBA)–ים), ואת עלויות החומרה והתוכנה הנדרשות להקמה ולתחזוקה של מסד הנתונים בארגון.

בימינו, ארגונים רבים עושים שימוש במסד נתונים במיקור חוץ בענן כשירות (Database-as-a-service), למשל

על-ידי שימוש בשירות Azure SQL Database של חברת מיקרוסופט.

לצד היתרונות שהוזכרו לעיל, אחסון מסדי נתונים במיקור חוץ כרוך בסיכונים למידע המאוחסן ולמידע המשודר ללקוחות וחזרה לספק. מצד אחד התשדורות בין ספק מיקור החוץ ללקוחותיו וחזרה חשופים להתקפות של גורמים חיצוניים, מצד שני קיים סיכון שהמידע הנשמר אצל הספק או משודר על-ידו ללקוחותיו ישונה או ימחק

על-ידי ספק מיקור החוץ בעצמו. קיימים כיום פתרונות טובים לאבטחת התשדורות בין הספק ללקוחותיו וחזרה כגון SSL, הם אינם נכללים בתכולת עבודה זו. מטרת עבודה זו היא להציג פתרונות המאפשרים לוודא שהספק בעצמו לא ביצע שינויים במידע הנשמר אצלו או במידע שהוא משדר ללקוחותיו מבלי שהשינוי יתגלה.

הפתרונות מוצגים באמצעות סקירה מפורטת של כמה מאמרים עדכניים וכן ע"י הצגת המימוש של אחד האלגוריתמים שמוצגים בעבודה.

# מבוא

עם התחזקות המגמה של אחסון מסדי נתונים באתרים חיצוניים לרבות בענן, מתחזקת החשיבות של שמירה על מהימנות הנתונים (integrity), מקורם (origin authenticity) ושלמותם (completeness).

חשוב להבין מה משמעות 3 המושגים הללו ומה ההבדלים ביניהם:

* שמירה על מהימנות הנתונים (integrity) - שמירה שלא בוצע כל שינוי במידע המאוחסן על-ידי גורם שאינו הבעלים של המידע.
* שמירה על מקור הנתונים (origin authenticity) - שמירה שהמידע הינו המידע המקורי שהוכנס על-ידי הבעלים של המידע, ולא על-ידי כל גורם אחר.
* שמירה על שלמות הנתונים (completeness) - שמירה שצרכן המידע יקבל את כל הרשומות הרלוונטיות לשאילתות שהריץ, ולא רק מידע חלקי.

בעלי המידע וצרכני המידע, מעוניינים במנגנונים שיבטיחו זאת בוודאות מלאה, תוך פגיעה מינימלית ככל הניתן בזמני עדכון ושליפת המידע.

קיימים כיום פתרונות טובים לאבטחת התשדורת ממסד הנתונים במיקור החוץ אל לקוחותיו וחזרה כגון SSL, אך הם אינם נותנים מענה למקרים בהם ספק מיקור החוץ בעצמו אינו אמין, ועלול לבצע שינויים במידע הנשמר אצלו או במידע שהוא מחזיר ללקוחותיו. מטרת עבודה זו היא להציג פתרונות הנותנים מענה לכך.

הפתרונות שיוצגו משתמשים בכלים קריפטוגרפים שונים להצפנת המידע, במטרה להבטיח שגורם בלתי מורשה לא יוכל לשנות את ההצפנה מבלי שהשינוי יהווה חותמת ברת תוקף לשינוי בלתי חוקי.

העבודה מתמקדת ב-2 שיטות הצפנה: הצפנה מבוססת חתימות המשתמשת בטכניקה של איחוד חתימות לאמת את תוצאות השאילתות, והצפנה מבוססת עץ (tree-based) המשתמשת ב-Merkle hash tree או בגרסאות שלו לביצוע ההצפנה והאימות.

לעבודה שלושה פרקים. הראשון שבהם מציג את 2 השיטות באמצעות סקירה מפורטת של 3 מאמרים:

שיטת החתימות המאוחדות ממאמר [‎1] ושיטת MHT ממאמר [‎6] וממאמר [‎17]

הפרק השני מציג בקצרה מגוון פתרונות הקשורים לנושא באמצעות סקירה מתומצתת של 4 מאמרים נוספים: המאמר הראשון [‎16] מציג פתרון של מספר שרתים במיקור חוץ המחלקים את העבודה ביניהם בביצוע Join-ים, המאמר השני [‎19] מציג את האתגרים של ספק במיקור חוץ המציע את מסד הנתונים כשירות, המאמר השלישי [‎‎‎‎18] מציג פתרון התומך בהכנסה ושליפה של כמות גדולה של מידע לשרת במיקור חוץ, והמאמר הרביעי  [‎23 מציג פתרון להבטחת מהימנות הנתונים במסדי נתונים לא רלציוניים.

הפרק השלישי עוסק בתיאור המימוש המעשי.

קריאה נעימה.



# סקירה מפורטת

## 1.1 שיטות מבוססות חתימות - Authentication and Integrity in Outsourced Databases [‎1]

### **הקדמה**

פרק זה המבוסס ברובו על המאמר Authentication and Integrity in Outsourced Databases [‎1] מציג מספר טכניקות הצפנה בעזרת חתימות מאוחדות, תוך אזכור המאמרים המתייחסים לטכניקות אלה. יש לציין, כי מאמר זה פורסם כבר ב-2006, ומכיוון שנושא הבטחת מהימנות הנתונים במיקור חוץ הוא נושא הנחקר ומתפתח כל הזמן, קיימים בוודאי מאמרים חדשים יותר המציגים שיפורים, תוספות ועדכונים לאלגוריתמים המוזכרים בפרק.

הפתרונות שיוצגו בפרק זה נותנים מענה להבטחת מהימנות הנתונים (integrity) ומקורם (origin authenticity), אך אינם נותנים מענה להבטחת שלמותם (completeness).

הם מוגבלים לשאילתות שוויון בלבד. כלומר, ל-SELECT-ים שתוצאתם הינה סט רשומות (או תכונות) התואמות לפרדיקט מסוים או למספר פרדיקטים. הם אינם נותנים מענה לשאילתות סכימה (aggregation) כגון SUM או AVERAGE.

### **המודל**

מודל מיקור החוץ של מסד הנתונים הוא מודל של שרת – לקוח.

ספק מיקור החוץ הינו השרת, המארח את מסדי הנתונים, ומספק מנגנונים המאפשרים ללקוחותיו ליצור, לאחסן, לעדכן ולשלוף את הנתונים. הוא אחראי לתחזוקת מסד הנתונים כולל ביצוע גיבויים והבטחת זמינות הנתונים.

הלקוח, הינו לקוח "לוגי" המייצג ארגון או משתמש/ים. הלקוח יכול להיות בעל מגבלות מחשוב, מגבלות אחסון ו/או בעל רוחב פס מוגבל כגון מכשיר wireless PDA.

במאמר‎1] מתוארים 3 סוגי מודלים:

1. Unified client model - מודל המכיל משתמש אחד בלבד, המשמש בו זמנית גם כבעל המידע (יוצר, מעדכן ומוחק נתונים) וגם כצרכן המידע (מריץ שאילתות על הנתונים). ראה איור 1



איור 1 - מודל של משתמש אחד (המשמש גם כבעל המידע וגם כצרכן המידע)

1. Multi-Querier model - מודל המכיל בעל מידע אחד ומספר צרכני מידע. לדוגמא: מסד נתונים של חברת תעופה המנוהל על-ידי משתמש אחד, אך מאפשר למספר רב של משתמשים לשלוף את המידע ללא אפשרות לעדכן אותו. ראה איור 2



איור 2 - מודל של בעל מידע אחד ומספר צרכני מידע

1. Multi-Owner model - מודל המכיל מספר בעלי מידע ומספר צרכני מידע.

לדוגמא: מסד נתונים המכיל מידע על נתוני מכירות. כל מנהל מכירות אחראי על רשומות מסוימות אותן הוא יצר/עדכן, ובנוסף הוא גם הצרכן ששולף אותן. ראה איור 3.



איור 3- מודל של מספר בעלי מידע ומספר צרכני מידע

כשצרכן המידע מריץ שאילתה על מסד הנתונים, הוא מצפה לקבל תוצאה המורכבת מכל הרשומות העונות על תנאי השאילתה שהריץ. אם מספר הרשומות הכולל במסד הנתונים הוא m, אזי התוצאה יכולה להכיל בין 0 ל-m רשומות, בהתאם לתנאי השאילתה. כלומר, תוצאת השאילתה תהיה מורכבת מאחת מ- התוצאות האפשריות. נשאלת השאלה כיצד ניתן לבצע אימות בטוח ויעיל לכל אחת מ- תוצאות השאילתה האפשריות?

### **הנחות יסוד**

שמירת הנתונים

כפי שנאמר קודם לכן, פרק זה אינו עוסק בנושא של הבטחת שלמות הנתונים (completeness), כלומר לא עוסק בהבטחה שצרכן המידע יקבל את כל הרשומות הרלוונטיות לשאילתות שהריץ, ולא רק מידע חלקי.

רמת ההצפנה ובדיקת מהימנות הנתונים (Granularity of Integrity)

ההנחה היא שבעל המידע שומר את המידע בצורה המתאימה לו, ויכול לקרוא את תוצאות השאילתות ללא תלות ברמת ההצפנה של מסד הנתונים: הצפנה מלאה, הצפנה חלקית או ללא הצפנה בכלל.

בדיקת מהימנות הנתונים (data integrity/authentication) יכולה להתבצע בכמה רמות בהתאם לרמת ההצפנה של הנתונים: ברמת הטבלה, ברמת העמודה, ברמת השורה או ברמת התכונה הבודדת (attribute value). הצפנה/בדיקה ברמת הטבלה/העמודה מחייבת החזרת הטבלה/העמודה למשתמש בכל שאילתה, כדי שיוכל לאמת את מהימנות הנתונים. אין מדובר בהצפנה/בדיקה מעשית מאחר שהיא דורשת העברת מסה גדולה של נתונים למשתמש בכל שאילתה. לעומת זאת, בדיקה/הצפנה ברמת התכונה הבודדת מחייבת בדיקה של מספר חתימות רב, והיא אינה מעשית בשל החישוביות הרבה הנדרשת מבעל המידע החותם על מספר רב של חתימות, ובשל שטח האחסון הגדול הנדרש בצד השרת לשמירת החתימות הללו.

רמת הבדיקה האופטימלית אם כן, היא הבדיקה ברמת השורה. היא דורשת חתימה אחת לכל שורה שאינה מכבידה מבחינה חישובית על החותם, אינה דורשת שטח אחסון גדול מדי בצד השרת, ואינה דורשת העברה מאסיבית של נתונים למשתמשים בכל שאילתה. משמעות ההצפנה ברמת השורה היא שרמת הנתונים הקטנה ביותר שתוחזר למשתמש היא שורה בודדת, גם אם הוא מעוניין רק בתכונה מסוימת אחת מתוכה.

שימוש בחתימות לעומת השימוש בקוד אימות מסרים (Message Authentication Code - MAC):

קוד אימות מסרים (Message Authentication Code - MAC) הוא שם כולל לקבוצה של פונקציות עם מפתח סודי המשמשות לאימות מסרים. פונקציית MAC מקבלת מפתח סודי ומסר באורך שרירותי ומפיקה פיסת מידע קצרה הנקראת תג אימות ((Authentication Code המשמש לאימות המידע.

קוד אימות המסרים מתאים לאימות הנתונים המוצפנים ברמת השורה, מאחר שמדובר בד"כ במסרים קטנים ואפקטיביים הן לחישוב והן לאימות. בסכמת MAC בסיסית המשתמש יכול לבקש מהשרת לשמור קוד אימות מסרים (או מפתח hash) עבור כל שורה, שחושב בעזרת מפתח הידוע רק לו. בכל שאילתה, יחזיר השרת בנוסף לתוצאת השאילתה גם מידע נוסף שישמש את המשתמש לביצוע האימות. מידע זה יחושב על-ידי השרת

כ-hash של כל המסרים של הרשומות המוחזרות בתוצאות השאילתה. תקורת רוחב הפס ותקורת החישוביות הנדרשות מהמשתמש הן מינימליות ולכן מדובר בפתרון יעיל למודל של משתמש אחד (Unified client model).

אולם קוד אימות המסרים אינו מעשי ב-2 המודלים האחרים שהוצגו לעיל, מאחר שהוא דורש את שיתופו של המפתח הסודי בין מספר משתמשים/בעלי מידע. לכן עבור 2 המודלים לעיל פתרון מעשי יותר הוא שימוש בחתימות דיגיטליות עם מפתח ציבורי, שהינו בעל תקורות בהיבטים של אחסנה, רוחב פס וחישוביות.

### **חתימות דיגיטליות - תקורות ותכונות רצויות**

גודלה של חתימה דיגיטלית יכול לנוע בין 320 ביטים (כמו ב-DSA) לבין 1024 ביטים (כמו ב-RSA). מאחר שתוצאות השאילתות יכולות להכיל אלפי שורות המוחזרות לצרכני המידע, אימות החתימות יכול להיות מאד יקר עבורם בהיבטים של תקורות החישוביות ורוחב הפס. בנוסף לכך, גם אם אין מגבלת אחסון בצד השרת, יתכן שיש מגבלות של כח המחשוב ו/או שטח האחסון של צרכן המידע, לכן חיוני לצמצם ככל הניתן את תקורת החישוביות ורוחב הפס עבור צרכני המידע.

נדרג את התקורות בהתאם לסדר חשיבותן, כך שבמקומות הראשונים יופיעו התקורות של צרכן המידע אותם ברצוננו לצמצם ככל הניתן, ולאחריהן יופיעו התקורות של השרת ובעל המידע הנמצאים בעדיפות פחותה לצמצום:

1. תקורת כח החישוביות של צרכן המידע – כח החישוביות הנדרש לצרכן המידע על-מנת לאמת את המידע המתקבל בתוצאת השאילתה.
2. תקורת רוחב הפס של צרכן המידע – שליחה/קבלה של הנתונים הנדרשים לאימות המידע (בנוסף לתקורה מקבלת/שליחת רשומות המידע עצמן בתוצאת השאילתה)
3. תקורת כח החישוביות של השרת – כח החישוביות הנדרש מהשרת על מנת לבצע שינויים (אם בכלל) על מידע האימות אותו הוא מעביר לצרכני המידע בנוסף לרשומות המידע עצמן.
4. תקורת כח החישוביות של בעל המידע (החותם) – כח החישוביות הנדרש מבעל המידע לחישוב מידע האימות שיאוחסן בשרת. תקורה זו נמצאת בעדיפות נמוכה, מאחר שההנחה היא שבעלי המידע מבצעים באופן יחסי הרבה פחות פעולות על הרשומות במסד הנתונים (יצירה, מחיקה ועדכון), מאשר כמות השאילתות המתבצעות על-ידי צרכני המידע.
5. תקורת שטח האחסון בצד השרת – שטח האחסון הנדרש לשמירת מידע האימות במסד הנתונים בשרת. תקורה זו נמצאת בעדיפות אחרונה, מאחר שההנחה היא שספק מיקור החוץ מעוניין למכור ללקוחותיו כמה שיותר שטח אחסון, ולכן יעדיף דווקא כמה שיותר להגדיל את שטח האחסון הנדרש.

חתימות דיגיטליות האפקטיביות במיוחד בשלב האימות מתאימות לסדר עדיפות זה, מאחר שתקורת החישוביות ביצורם חשובה הרבה פחות מתקורת האימות שלהם.

הפתרון האידיאלי לסדר העדיפות לעיל הינו בעל תקורה מינימלית (קבועה) על החישוביות של צרכן המידע, ותקורה קבועה (בהיבטי מידע האימות) של רוחב הפס. אך לא ניתן להגיע לתקורת חישוביות קבועה של צרכן המידע, מאחר שעבור כל הרשומות שמחזירה השאילתה, הוא צריך לחשב מחדש לכל הפחות פונקציית hash או פונקציה דומה. לכן יעשה שימוש בסכמות של חתימות דיגיטליות המאפשרות לאחד את כל החתימות המוחזרות לחתימה מאוחדת אחת, כך שתקורת האימות של חתימה מאוחדת כזו תהיה כמעט זהה לתקורת האימות של חתימה בודדת. קיימות סכמות כאלה המאפשרות לאחד חתימות שנחתמו על-ידי אותו החותם, לכן הן מתאימות למודלים שלUnified client ו-Multi-Querier. סכמות אחרות המאפשרות לאחד חתימות שנחתמו על-ידי חותמים שונים יתאימו למודל של Multi-Owner.

### **סכמות איחוד חתימות**

### **Condensed RSA**

סכמה זו הינה הרחבה של סכמת ה-RSA הסטנדרטית. הצפנת ה-RSA מתאימה לאחד חתימות שנוצרו בידי אותו החותם לחתימה "מאוחדת" אחת בשל תכונת ההומומורפית שלה לכפל (multiplicative homomorphic). הומומורפיות לכפל משמעותה שתוצאת המכפלה של 2 הודעות ו- המוצפנות בנפרד על-ידי ההצפנה E תהיה שווה להצפנה של המכפלה שלהם על-ידי אותה ההצפנה E: .

תכונה זו מאפשרת ל"אחד" חתימות בצורה מדורגת, כלומר להוסיף בכל פעם הודעות חתומות נוספות לחתימה ה"מאוחדת". אימות מוצלח של חתימה "מאוחדת" כזו מבטיח שכל ההודעות ה"מאוחדות" בחתימה נחתמו בצורה תקינה בידי החותם המורשה.

### **RSA סטנדרטי**

ניזכר בקצרה בסכמת ה-RSA הסטנדרטית:

1. בוחרים 2 מספרים ראשוניים אקראיים p ו-q בגודל - ביטים (כאשר k הוא בגודל של 1024 ביטים לפחות).
2. מחשבים את .
3. מחשבים את .
4. בוחרים מפתח ציבורי, שלםe שהוא [זר](https://he.wikipedia.org/wiki/%D7%9E%D7%A1%D7%A4%D7%A8%D7%99%D7%9D_%D7%96%D7%A8%D7%99%D7%9D) ל-,המקיים: ,
5. מחשבים את המפתח הפרטי d המקיים: ,

חתימתRSA מחושבת בעזרת פונקצייתhash על הודעת הקלט. תהא , פונקציית hash כזו (כגון MD5 או SHA-1) אשר על הודעת הקלט באורך משתנה m, מפיקה פלט באורך קבוע . חתימתRSA סטנדרטית תחושב על-ידי . אימות חתימת ה-RSA יבוצע על-ידי בדיקה כי . הן ייצור החתימה והן האימות שלה דורשים חישוביות בסדר גודל של פעולה אקספוננציאלית אחת.

### **Condensed-RSA**

בהינתן t הודעות קלט , והחתימות שלהן החתומות בידי חותם אחד, חתימת Condensed-RSA תוגדר כמכפלה של החתימות הבודדות בצורה הבאה:

(1)

המתקבלת, הינה בגודל זהה לגודלה של חתימתRSA סטנדרטית. כדי לאמת את החתימה, המאמת צריך לבצע מכפלה של כל פונקציות ה-hash על הודעות הקלט, ולבצע את הבדיקה הבאה:

(2)

### **אימות Batch-י של חתימות RSA**

סכמת האימות ה- Batch-י נועדה לאמת בצורה Batch-ית (בבת אחת) מספר חתימות שנחתמו על-ידי אותו החותם בצורה יעילה. במקום לאמת כל חתימה בנפרד, סכמת האימות ה- Batch-י מאפשרת לבצע את האימות שלהם בו זמנית.

למעשה האימות בסכמת האימות ה-Batch-י דומה מאד לאימות בסכמה של Condensed RSA.

ההבדל המרכזי בין 2 הסכמות נעוץ בכך שלמאמת באימות ה-Batch-י יש גישה לחתימות הבודדות בשונה מהמאמת של Condensed-RSA. את המכפלה בסכמת האימות ה-Batch-י הוא מחשב בעצמו, ובכך הוא מאיץ את קצב האימות הכולל ע"י צמצום מספר הפעולות האקספוננציאליות הנדרשות לביצוע האימות.

למעשה, התקורה החישובית לאימות של חתימה Batch-ית הינה מאותו סדר גודל של התקורה החישובית לאימות של חתימה בודדת.

בהינתן אוסף החתימות החוקיות על ההודעות הייחודיות , אימות RSA Batch-י יבוצע בעזרת הבדיקה הבאה (המכונה fast screening במאמר [‎20]):

(3)

מאמר [‎2] מתאר את ההסתברות שבה תוקף פוטנציאלי יצליח ליצור אוסף חתימות Batch-י העונה על תנאי משוואה (3), מבלי שיש ברשותו את החתימות הבודדות של כל אחת מההודעות בנפרד. מאחר שלמאמת באימות ה-Batch-י יש גישה לחתימות הבודדות המרכיבות את מכפלת החתימות, זה מאפשר לו לערוך בדיקות נוספות במקרה של כישלון האימות ה-Batch-י. הוא יכול למשל לאתר את החתימה הבעייתית מתוך החתימות הבודדות המרכיבות את המכפלה.

### **הבטיחות (Security) של Condensed-RSA**

כותבי מאמר[‎2]  טוענים כי Condensed-RSA עמיד כנגד התקפות מסוג adaptive chosen message attacks. הם מוכיחים זאת ע"י כך שהם מראים שאם תוקףA יצליח לשבור את ה-Condensed-RSA, אזי הדבר שקול לכך שזייפן B יצליח ליצור בהצלחה batch instance שיעבור את מבחן האימות ה- Batch-י, מבלי שיש בידיו את החתימות הבודדות של כל אחת מהודעות הקלט. כלומר, הם טוענים שה-Condensed-RSA בטוח לפחות כמו האימות Batch-י של חתימות ה-RSA.

הגדרה: תוקף A יצליח לשבור את ה-Condensed-RSA אם הוא יצליח ליצור חתימה "מאוחדת" להודעות , העונה על תנאי משוואה (2) לעיל, מבלי שיש בידיו את החתימות הבודדות של כל אחת מהודעות הקלט. הם מניחים שנעשה שימוש בפונקציית Full Domain Hash (FDH), המתוארת בפירוט במאמר [‎20].

FDH הינה פונקציית hash המוגדרת כ-

### **המתווה**

תוקף A מקבל קלט של ו-, כאשר ו-. במילים אחרות, תוקףA מקבל כקלט t הודעות, וסט המכיל חתימות של מתוך t ההודעות הללו (). A, בהתאם להגדרה, שובר את ה- Condensed-RSAביצירת פלט של חתימת Condensed-RSA ולידית. כעת ניתן להקים את זייפן B שישבור את האימות ה-Batch-י של חתימות ה-RSA.

### **פרטים**

עבור קלט ו-, זייפן B מייצר את הפלט כך ש:

.B מתנהל בצורה הבאה:

1. מייצר מספרים רנדומליים , כאשר .

2. שולח את ההודעות ו-ל-A.

3. נסמן את החתימה המזויפת של ה-Condensed-RSA המוחזרת על-ידיA , ב-X.

4. מחשב את .

5. מייצר כפלט את ה-batch instance, עבור ההודעות .

### **טענה 1**

זייפן B יצר סט חתימות העוברות את מבחן האימות ה- Batch-י של חתימות ה-RSA.

נשים לב כי . כלומר משוואה (3) מתקיימת ומבחן האימות ה- Batch-י עובר בהצלחה.

### **טענה 2**

אםRSA הוא חד-כיווני, אזי Condensed-RSA הינה סכמת חתימה מאובטחת.

האימות ה-Batch-י של חתימות ה-RSA הוכח כמאובטח תחת ההנחה ש-RSA הוא חד-כיווני. לכן, מאחר

שה-Condensed-RSA בטוח לפחות כמו האימות Batch-י של חתימות ה-RSA, כותבי מאמר [‎2] מסיקים כי

Condensed-RSA מאובטח, בהנחה ש-RSA הינו אוסף של פונקציות חד-כיווניות.

### **השוואת תקורות**

מאמר [‎2] משווה את מחירי התקורות של RSA סטנדרטי, Condensed-RSA, ואימות Batch-י של חתימות

RSA במודל של מסד הנתונים במיקור חוץ, תוך התעלמות ממחירו הזניח של חישוב ה-hash.

ההשוואה מתבצעת על תוצאת שאילתה המחזירה t רשומות.

בחתימותRSA סטנדרטיות, צרכן המידע יידרש לקבל ולעבד t חתימות, אחת עבור כל רשומה בתוצאת השאילתה. בכדי לאמת את החתימות, הוא יידרש לבצע t אימותים של RSA, כלומר t פעולות אקספוננציאליות. תקורת רוחב הפס תגיע ל- t\*|n| ביטים.

באימות Batch-י של חתימות ה-RSA, תהליך האימות יכלול חישוב של מכפלת ה-hash-ים של ההודעות, ושל מכפלת החתימות, הכוללת בסה"כ 2(t-1) מכפלות, ועוד חישוב אקספוננציאלי אחד. תקורת רוחב הפס זהה לזו שלRSA סטנדרטי, t\*|n| ביטים.

תקורת רוחב הפס של Condensed-RSA הינה חתימה אחת בגודל ||n ביטים, ואימות במחיר של t-1)) מכפלות לחישוב המכפלה של ה-hash-ים של ההודעות, ועוד חישוב אקספוננציאלי אחד.

כלומר, Condensed-RSA חוסך t-1)\*|n|) ביטים של רוחב פס, ו-t-1)) מכפלות לעומת האימות ה-Batch-י של חתימות RSA.

### **Condensed-RSA in outsourced DB setting**

Condensed-RSA מתאים למודלים של Unified client ושל Multi-Querier מאחר ששניהם מניחים את קיומו של בעל מידע (חותם) אחד. שרת מיקור החוץ המריץ שאילתה נדרש לבצע את הפעולות הבאות: בחירת הרשומות העונות על תנאי השאילתה, שליפת החתימות המתאימות לרשומות שנבחרו, איחוד החתימות לחתימה "מאוחדת" אחת (על-ידי ביצוע מכפלה כפי שהוסבר לעיל) ושליחת תוצאות השאילתה יחד עם החתימה ה"מאוחדת" לצרכן המידע. כמובן ש-Condensed-RSA חסכוני גם בכח החישוביות וגם ברוחב הפס של צרכן המידע כפי שהוסבר בסעיף הקודם. במודל Multi-Owner model בעל מספר בעלי מידע (חותמים), לא ניתן לשמר תקורת רוחב פס קבועה. השרת יכול לאחד חתימות של כל חותם בנפרד, ולשלוח אותם בנפרד. כלומר, תקורת רוחב הפס של צרכן המידע תהיה ליניארית בהתאם לכמות החותמים, והוא יוכל לאמת כל חתימה בנפרד.

### **BGLS**

מאמר[‎3] מתאר את סכמתBGLS  (Boneh-Gentry-Lynn-Shacham) לבניית חתימה מאוחדת, המאחדת חתימות של חותמים שונים החתומים על הודעות שונות לחתימה אחת קצרה. יתרונה בכך שהיא מאפשרת לאמת בפעם אחת חתימות שונות של חותמים שונים על הודעות שונות, בניגוד ל- Condensed-RSAבה נדרש לאמת כל חותם בנפרד.

סכמת BGLS מבוססת על סכמת החתימות הקצרות BLS (Boneh-Lynn-Shacham) המתוארות במאמר[‎22].

כפי ששמה מרמז, החתימה הקצרה הינה באורך 160 ביטים בלבד, וקצרה יותר מהחתימות שתוארו בסכמות האחרות עד כה. גם החתימות המאוחדות בסכמת BGLS הינם באורך של 160 ביטים בלבד ללא קשר לכמות החתימות שאוחדו בכל חתימה.

הבניה בסכמת BGLS מתבצעת בעזרת 2 עקומות אליפטיותו- ומפה בילינארית . בניה המבוססת על מפה ביליניארית היא פשוטה ויעילה יותר מבניה המבוססת על RSA, ומניבה חתימות קצרות יותר.

בסכמה זו נעשה שימוש בפרוטוקול דיפי-הלמן (Difﬁe-Hellman).

נתאר בקצרה את הפרמטרים הרלוונטיים לסכמה זו:

1. ו- הינם 2 חבורות מעגליות מעל שדה סופי מסדר ראשוני p.
2. הוא המחולל של , ו- הוא המחולל של .
3. הינו איזומורפיזם ניתן לחישוב (Computable isomorphism) מ- ל-, כך ש-
4. e הינה מפה בילינארית ניתנת לחישוב המוגדרת כ:

, כך ש: עונה על המאפיינים הבאים:

1. בילינאריות: , מתקיים
2. אי ניוון (Non-degenerate): 1≠.

שני המאפיינים הנ"ל מרמזים כי:

### **סכמת BGLS**

BGLS עושה שימוש בפונקציית hash בתחום מלא: . *יצירת המפתחות כרוכה בבחירה רנדומלית של המפתח הפרטי , וחישוב המפתח הציבורי .* יצירת החתימה על הודעה m כרוכה בחישוב של , כאשר  *ו-*. אימות החתימה מבוצע על-ידי חישוב של ובדיקה האם .

### **איחוד חתימות של BGLS**

"איחוד" t חתימות של BGLS, מתבצע על-ידי חישוב המכפלה של החתימות הבודדות בצורה הבאה:

, כאשר מייצגת את החתימה המתאימה להודעה . גודל החתימה ה"מאוחדת" , הינו כגודלה של חתימה BGLS בודדת, למשל ביטים. בדומה ל-Condensed-RSA, החתימה ה"מאוחדת" יכולה להיבנות בצורה מדורגת על-ידי כל אחד.

אימות החתימה ה"מאוחדת" כרוך בחישוב המכפלה של פונקציות ה-hash של ההודעות, ואימות השוויון הבא:

.

בשל המאפיינים של המפות הבילינאריות, ניתן לפתח את צד שמאל של המשוואה: .

### **ביצועים של BGLS**

כשמחשבים את התקורה של אימות החתימה של BGLS, אנו מבדילים בין 2 פעולות: המכפלה והחישוב של המפה הבילינארית. עבור חותם אחד בעל t הודעות קלט, עלות האימות הינה חישוב המכפלה של הודעות hash ( מכפלות), ולאחריה חישוב של 2 מפות בילינאריות. עבור מספר חותמים (k חותמים בעלי t חתימות לכל חותם), עלות האימות היא מכפלות וגם חישוב של k + 1 מפות בילינאריות.

### **BGLS in outsourced DB setting**

BGLS מתאים ל-3 המודלים שתוארו במאמר [‎2]. השרת מריץ את השאילתה של צרכן המידע על-ידי בחירת הרשומות (יחד עם החתימות שלהן) העונות על תנאי השאילתה, "מאחד" אותן (כפי שהוסבר לעיל), ושולח בחזרה חתימה "מאוחדת" יחד עם הרשומות הרלוונטיות לתוצאות השאילתה. במודלים של Unified client

ו-Multi-Querier ה-BGLS חסכוני הן ברוחב פס והן בכח החישוב של צרכן המידע (בשל החתימה ה"מאוחדת" הקצרה). יחד עם זאת, במודל של Multi-Owner, BGLS אמנם מספק יעילות ברוחב הפס, אך הוא אינו חסכוני בהיבט של כח החישוב של צרכן המידע. מאחר שאימות חתימה של BGLS הבנויה מחתימות של k חותמים שונים, כרוכה בחישוב של k + 1 מפות בילינאריות, שהוא חישוב יקר באופן יחסי.

### **אימות Batch-י של חתימותDSA**

Digital Signature Algorithm DSA – הינו סטנדרט ליצירה של חתימות דיגיטליות שאומץ על-ידי ממשלת ארה"ב. הוא נחשב לאחד המנגנונים ה[קריפטוגרפי](https://he.wikipedia.org/wiki/%D7%A7%D7%A8%D7%99%D7%A4%D7%98%D7%95%D7%92%D7%A8%D7%A4%D7%99%D7%94)ים הטובים והפופולאריים ביותר ליצירת [חתימה דיגיטלית](https://he.wikipedia.org/wiki/%D7%97%D7%AA%D7%99%D7%9E%D7%94_%D7%90%D7%9C%D7%A7%D7%98%D7%A8%D7%95%D7%A0%D7%99%D7%AA). החותם מייצר בעזרת המפתח הפרטי שלו מעין תווית המוצמדת להודעה ומאפשרת את אימותה באמצעות המפתח הציבורי של החותם. החתימה הינה תוספת נפרדת להודעה ואינה הצפנה של ההודעה עצמה. לכן במובן זה אלגוריתם DSA אינו נחשב לאלגוריתם הצפנה כמו ה-RSA.

הרעיון של מנגנון החתימה של DSA מבוסס על [בעיית הלוגריתם הדיסקרטי](https://he.wikipedia.org/wiki/%D7%91%D7%A2%D7%99%D7%99%D7%AA_%D7%9C%D7%95%D7%92%D7%A8%D7%99%D7%AA%D7%9D_%D7%93%D7%99%D7%A1%D7%A7%D7%A8%D7%98%D7%99)(Discrete Logarithm Problem)  והוא וריאציה של סכמת החתימות של ElGamal. הוא מסתמך על הקושי שבפתרון בעיית חישוב לוגריתמים ב[חבורות (computing [discrete logarithms](https://en.wikipedia.org/wiki/Discrete_logarithm) in the multiplicative group),](https://he.wikipedia.org/wiki/%D7%97%D7%91%D7%95%D7%A8%D7%94_(%D7%9E%D7%91%D7%A0%D7%94_%D7%90%D7%9C%D7%92%D7%91%D7%A8%D7%99)) שהיא כידוע בעיה קשה מבחינה חישובית. ניתן ליצור חתימה מסוג ElGamal בצורה יעילה על-ידי שימוש בטכניקות קדם חישוב. אך האימות הוא בד"כ הרבה פחות יעיל ודורש 2 חישובים אקספוננציאלים. לכן אימות בבת אחת של מספר חתימות DSA יהיה יעיל בהרבה. מאמר [‎2] מתאר את שיטת האימות ה-Batch-י של חתימות DSA המאמתת בצורה Batch-ית (בבת אחת) מספר חתימות DSA. "איחוד" חתימות של DSA יהיה יעיל מאד בתצורת ה-ODB, במיוחד במודל של

Multi-Owner. אך אין דרך בטוחה לבצע "איחוד" של חתימות מבוססי DLP - Discrete Logarithm Problem כפי ש-Boyd ו-Pavlovski הראו במאמר [‎5], בו הם הדגימו כיצד תוקף יכול בקלות להציג חתימות מזויפות שיעברו את מבחן האימות ה- batch-י.

שיטות האימות ה-Batch-י של חתימות DSA הידועות נכון לזמן כתיבת המאמר, משתמשות בטכניקות כגון small exponent test או bucket test הדורשות מהמאמת לבצע פעולות על החתימות הבודדות (למשל

ב- small exponent test המאמת נדרש לבצע פעולה אקספוננציאלית קטנה על כל חתימה לחוד לפני ביצוע

ה-batching). לכן, לא ניתן להרכיב חתימה "מאוחדת" אחת. במילים אחרות, שימוש בסכמת חתימה מבוססת DLP יחד עם טכניקת אימות batch-י במודל ה-ODB תעזור לצמצם את עלויות החישוביות של צרכן המידע, אבל לא את עלויות רוחב הפס שלו.

### **השוואת עלויות**

במאמר [‎2] מבוצעת השוואה בין התקורות של הסכמות השונות שתוארו קודם לכן. תחילה מבוצעת הערכת העלויות של פעולות קריפטוגרפיות בסיסיות, כגון מכפלות, היפוכים ופעולות אקספוננציאליות. בעזרת מדידה של זמני כל פעולה ופעולה, ניתן להראות את התקורה הנגרמת בכל סכמה.

בוצע שימוש במכונת לינוקס P3-977MHZ עם ספריית OpenSSL לחישוב הפעולות הבודדות. נעשה שימוש

ב-1024 ביט muduli ב-RSA ו-DSA: 1024 ביט n ב-RSA, ו-1024 ביט p (יחד עם 160-ביט (q ב-DSA.

השוואת העלויות תבוצע בעזרת הסימונים הבאים:

QC – Querier computation תקורה חישובית של צרכן המידע

QB – Querier bandwidth תקורת רוחב הפס של צרכן המידע

SC – Server computation תקורה חישובית של שרת מיקור החוץ

OC – Owner computation תקורה חישובית של בעל המידע (החותם)

SS – Server storage תקורת האחסון של שרת מיקור החוץ

Multt(k) – t modular multiplications with modulus of size |k|

– t modular exponentiations with modulus of size |k| and exponent of size |l|

Invt(k) – t modular inverses with modulus of size |k|

BM(t) – t bilinear mapping

תוצאת השאילתה כוללת k \* t רשומות, כאשר k הוא מספר החותמים (בעלי המידע) ו-t הוא מספר החתימות (רשומות) המיוצרות על-ידי כל חותם.

|  |  |  |  |
| --- | --- | --- | --- |
| **BGLS** | **Batch-DSA** | **Condensed-RSA** |  |
|  |  |  | **QC** |
| *p* | *k \* t \* ( p + q)* | *k \* n* | **QB** |
|  | *0* |  | **SC** |
|  |  |  | **OC** |
| *k \* t \* p* | *k \* t \* ( p + q)* | *k \* t \* n* | **SS** |

טבלה 1- השוואת עלויות במודל של מסד הנתונים במיקור חוץ

טבלה 1 מתארת את עלויות התקורות של כח החישוביות, האחסון ורוחב הפס בכל אחת מהסכמות בהיבט של פעולות קריפטוגרפיות. הטבלה מציגה את פקטורי התקורה בסדר חשיבות יורד. (כאשר המטרה העיקרית היא להפחית למינימום את תקורות החישוביות ורוחב הפס של צרכן המידע QC ו-QB).

מספר החותמיםk=1 במודלים של Unified Client ו-Multi-Querier. במודלים הללו, ל-Condensed-RSA

ו-BGLS יש רוחב פס קבוע, ללא תלות במספר החתימות. תקורת החישוביות של צרכן המידע (QC) הוא ליניארי במספר החתימות, מאחר ש-2 המודלים כוללים t מכפלות. יתרה מכך, אימות חתימה "מאוחדת" כרוך בפעולה אקספוננציאלית בודדת ב-Condensed-RSA, וב-2 מפות בילינאריות ב-BGLS, ולכן Condensed-RSA יעיל יותר. לעומת זאת, הן QC והן QB הם ליניאריים במספר החתימות עבור Batch-DSA, זאת בשל המבחן האקספוננציאלי הקטן. ב-Batch-DSA, לא מבוצע "איחוד" חתימות, ולכן השרת לא נדרש לבצע עבודה נוספת. תקורת החישוביות של צרכן המידע (QC) הוא ליניארי במספר החתימות. במודל ה-Multi-Owner, ל-BGLS יש תקורת QB קבועה, ואילו ב-Condensed-RSA, תקורת QB היא ליניארית במספר החותמים (K).

ב-Batch-DSA, כמו קודם, .QB הוא ליניארי במספר החתימות.

Condensed-RSA יעיל יותר מ-BGLS מאחר שאקספוננציאליות מודולרית יעילה בהרבה ממיפוי בילינארי, ומכפלה מודולרית ב-(כאשר |n|=1024) זולה יותר מחיבורים סקאלריים בעקומות אליפטיות.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BGLS** | **Batch-DSA** | **Condensed-RSA** |  |  |
| 354 | 3.82 | 6.82 | 1 Signature | **Sign** |
| 62  184.88  463.88  1570.8 | 8.52  1623.59  1655.86  16203.5 | 0.16  44.12  45.16  441.1 | 1 Signature  t=1000, k=1  t=100, k=10  t=1000, k=10 | **Verify** |

טבלה 2 - השוואת עלויות אימות וחתימה (ב-msec). t – מספר החתימות. k – מספר החותמים

טבלה 2 מראה את הזמן הנדרש לייצר ולאמת חתימה בודדת, מספר חתימות על-ידי חותם אחד, ומספר חתימות על-ידי מספר חותמים בשלושת הסכמות. האקספוננט הציבורי e=3 הוצב ב-RSA ונעשה שימוש במשפט השאריות הסיני לזרז את החתימות. לא נעשה שימוש בטכניקות אופטימיזציה כלשהן באף אחת מפעולות האימות.

### **סיכום ועבודות עתידיות (למאמר זה)**

לסיכום, במאמר [‎2] נחקרו מנגנונים השומרים על מהימנות הנתונים (data integrity) ביעילות במסדי נתונים במיקור חוץ. הוצגה סכמת Condensed RSA המתאימה למודלים של Uniﬁed Client ו-Multi-Querier, אך מכיוון שהיא אינה תומכת ב"איחוד" חתימות של חותמים שונים, היא אינה מתאימה למודל Multi-Owner של מספר בעלי מידע (חותמים). מצד שני, סכמת BGLS התומכת ב"איחוד" חתימות של חותמים שונים לחתימה "מאוחדת" אחת, עושה זאת בתקורה חישובית גבוהה. לכן, עבודות עתידיות יתמקדו במציאת סכמות חתימה פרקטיות עבור המודל של Multi-Owner בעל מספר בעלי מידע (חותמים).

## שיטות מבוססות עצים - Merkle Hash Tree based Techniques for Data Integrity of Outsourced Data ‏[‎6]

## הקדמה

פרק זה המבוסס ברובו על המאמרMerkle Hash Tree based Techniques for Data Integrity of Outsourced Data [‎6] מציג מספר טכניקות הצפנה מבוססות Merkle Hash Trees, כולל הטכניקות לשמירה ושליפת מידע האימות משרת מיקור החוץ. הפתרונות שיוצגו בפרק זה נותנים מענה לשמירה על מהימנות הנתונים לרבות שמירה על שלמותם (completeness), נכונותם (correctness) ועדכנותם (freshness).

מאמר [‎6] מציג סכמה בה מעורבים שלושה גורמים: בעלי מידע (DO), צרכני מידע וספק שירות מיקור החוץ (DSP).

ספק השירות אחראי לספק את כל שרותי המידע הנחוצים, לדאוג לזמינות השרתים, לגיבוי שלהם, ולביצוע שחזורים במידת הצורך. לא ניתן להסתמך עליו לגבי שמירה על מהימנות הנתונים. במאמר [‎6] מניחים שהתקשורת בין הגורמים המעורבים מאובטחת בעזרת טכניקה מאובטחת כגון SSL, וכל ניסיון של זיוף המידע בתווך בין הגורמים השונים יתגלה בנקל.

מספר גורמים צריכים להילקח בחשבון בטכניקות לשמירת מהימנות הנתונים בשרת במיקור החוץ:

* התקורה החישובית של בעל המידע.
* התקורה החישובית של ספק שירות מיקור החוץ.
* תקורת האחסון של ספק שירות מיקור החוץ.
* התקורה החישובית של צרכן המידע.
* תקורת האחסון של צרכן המידע.

## טכניקה בסיסית

ההנחה היא שקיים מכניזם להעברת מידע בין בעלי המידע לצרכני המידע בצורה מאובטחת כגון SSL. מידע זה יכול להיות מפתח ציבורי כלשהו או מידע hash-י. רק בעלי המידע יכולים לשנות אותו ואילו עבור צרכני המידע מדובר במידע לקריאה בלבד בשרת מיקור החוץ.

הטכניקה הבסיסית ביותר לשמירה על מהימנות הנתונים תהיה לחתום על כל שדה בטבלה בנפרד, ולשמור את החתימות בטבלה ביחד עם המידע האמיתי. בכל שאילתה ישלחו החתימות הרלוונטיות ביחד עם המידע לצרכן המידע לצורך אימות. טכניקה זו כרוכה בתקורה חישובית גבוהה הן לצרכי המידע והן לבעלי המידע החותמים עליו. תקורת האחסון לעומת זאת היא ליניארית לגודל המידע הנשמר. ספק שירות החוץ יכול למחוק חלק מהשדות או לשלוח תוצאות שאילתה חלקיות מבלי שצרכני המידע יגלו זאת באימות, מאחר שכל שדה שיתקבל בתוצאת השאילתה יהיה חתום בצורה תקינה. כלומר, טכניקה זו אינה שומרת על שלמות הנתונים.

ניתן לחלק טכניקות לשמירה על מהימנות הנתונים ל-2 קטגוריות: הסתברותיות ודטרמיניסטיות.

טכניקות הסתברותיות אינן מצריכות שינוי כלשהו בצד שרת מיקור החוץ, אך תוצאות אימות המידע שלהן מוטלות בספק בהסתברות מסוימות כפי ששם הטכניקה מרמז.

טכניקות דטרמיניסטיות מתבססות על מבני נתונים מאומתיםAuthenticated Data structures (ADS) בהן מתמקד פרק זה.



## Merkle hash trees

בטכניקת ADS חלק מהמידע הנדרש לאימות נשמר בשרת מיקור החוץ. שרת שירות מיקור החוץ מחזיר בנוסף לתוצאות השאילתה גם את המידע הזה המשמש את צרכני המידע לאמת את מהימנות הנתונים.

טכניקות ADS רבות הוצעו לבדיקת מהימנות נתונים, לרבות סכמות "איחוד" חתימות המצריכות שינוי של החתימות בכל הרשומות, והן נסקרו בפרק הקודם.

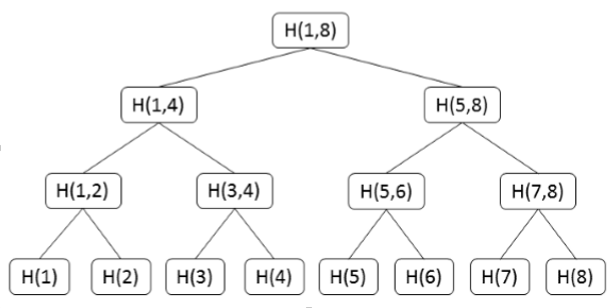
סכמה יעילה לבדיקת מהימנות הנתונים הינה סכמת Modified Merkle Hash Tree (MHT). בסכמה זו

עץ ה-hash המרכזי מחולק לעצים קטנים יותר בהם השורש ה-hash-י שלהם חתום. החלוקה נועדה למנוע חישובים מיותרים עד לשורש של עץ ה-hash המרכזי. מרקל הציע את טכניקת MHT לבדיקת מהימנות הנתונים במיקור חוץ, בה אין צורך בחתימות דיגיטליות לשמירה על מהימנות הנתונים. הטכניקה מבוססת על 2 מרכיבים: סכמת החתימות של מרקל (Merkle’s Signature Scheme) ועצי B+ (B+ Trees).

## Merkle’s Signature Scheme

סכמה זו מבוססת על עצי hash בינאריים.

דוגמא לעץ כזה ניתן לראות באיור הבא (איור 4):



איור 4 - Merkle hash tree

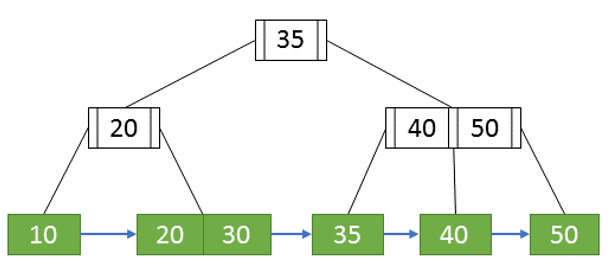
כל עלה מחזיק את ה-hash של בלוק מידע מסוים, לדוגמא H(1) מחזיק את ה-hash של בלוק המידע 1. צמתים פנימיים מחזיקים את ה-hash-ים המשורשרים של ילדיהם, לדוגמא: H(1,2) = H( H(1) | H(2) ) (כאשר | מסמן שרשור). סכמה זו מתבססת על כך שקיימת דרך בטוחה לשתף את שורש העץ בין החותם לבין המאמת. על-מנת לאמת בלוק מידע מסוים, אין צורך לשלוח את כל העץ לאימות, אלא רק את הצמתים המעורבים בנתיב האימות (authentication path) של בלוק המידע בנוסף לשורש העץ ששותף בין החותם לבין המאמת. לדוגמא: H(1), H(3,4) ו- H(5,8)צריכים להישלח לאימות. המאמת צריך לחשב את H(2) מבלוק המידע 2. H(1,2) יחושב לאחר מכן מ- H(1)שהתקבל ו-H(2) שחושב. לאחר מכן H(1,4) יחושב מ- H(1,2)שחושב ו H(3,4)-שהתקבל. H(1,8) יחושב מ- H(1,4)שחושב ו- H(5,8)שהתקבל. המאמת ישווה לבסוף בין שורש העץ H(1,8) ששותף עימו לזה אותו הוא חישב, ואם ה-hash-ים תואמים, אזי הוא אימת בהצלחה את בלוק מידע 2.

מספר עובדות לגבי Merkle’s Signature Scheme:

* ה- securityשל החתימה נגזרת מה-security של פונקציית ה-hash.
* רק פונקציית hash אחת צריכה להיות משותפת בצורה מאובטחת.
* כדי לאמת בלוק מידע מסוים, רק פונקציות hash צריכות להישלח לאימות, כאשר n מייצג את מספר הבלוקים של המידע.
* באימות של טווח ממושך של בלוקים אפילו פחות מ- פונקציות hash צריכות להישלח לאימות.

## עצי +B

עציB+ הם מקרה מיוחד של עצי B כפי שניתן לראות באיור 5 להלן:



איור 5 - עץ +B מסדר גודל 3

שורש העץ יכול להיות עלה או צומת. צומת יכול לשמור מפתחות אבל לא מידע, הנשמר אך ורק בעלים. העלים מקושרים ביניהם על-ידי מצביעים היוצרים מעין רשימה מקושרת, המאפשרת לעבור בצורה סדרתית על המידע.

בהינתן עץ מסוג B+ בגודל n, השורש יכול להחזיק בין 1 ל-n-1 מפתחות (כאשר הוא הצומת היחידי בעץ). אם השורש הינו צומת אזי יכולים להיות לו בין 2 ל-n צמתי בנים. לצמתים פנימיים יכולים להיות בין

ל-n צמתי בנים. עלים יכולים להחזיק בין ל- n-1 מפתחות.

## Data integrity based on Merkle hash tree

בסכמה החתימה המקורית של מרקל נעשה שימוש בעצי +B במקום עצים בינאריים, הדומים לעץ המוצג באיור 5 לעיל עם מספר שינויים: העלים מקושרים ישירות עם מצביעים, והעלים מחזיקים גם את פונקציות ה-hash אליו מצביע המפתח אותו הם מחזיקים בנוסף למפתחות.

בדוגמא באיור 5, העלה 20,30 מחזיק את ה-hash-ים של רשומות המידע של מפתחות 20 ו-30. המצביע הימני של הצומת הפנימי 20 מחזיק את ה-hash של ה-hash-ים המשורשרים עליהם מצביעים מפתחות 20 ו-30, כלומר H(20,30) = H( H(20) | H(30) ).

הבטיחות (security) של סכמת זו תלויה בבטיחות של פונקציית ה-hash בדומה לסכמה החתימה המקורית. סכמה זו פותרת את בעיית העדכניות (freshness): בכל פעם שבעל מידע כלשהו מעדכן מידע בשרת מיקור החוץ, מחושב שורש hash חדש שמשותף עם צרכני המידע לשמירת העדכניות.

## Radix Path Identifier

נדרש מבנה נתונים יעיל לשמירה של ה-MHT בשרת מיקור החוץ. קיימות הרבה שיטות לשמירה של מידע היררכי או דמוי עץ במסד נתונים כגון: adjacency list, path enumeration, nested set, closure tableוכדומה. לכל שיטה יש יתרונות וחסרונות. לדוגמא: יתרונה של מטריצת שכנוּת (adjacency list)מתבטא בכך שקל למצוא את ההורה של מצביע או רשומה מאחר שהקשר הורה-ילד מיוצג בה ישירות. חסרונה הוא שמציאת אבותיו עד לשורש העץ מצריכה מעבר על המטריצה צעד אחר צעד ואינה יעילה. בשיטה של path enumeration הנתיב של כל מצביע או רשומה נשמר ב-string, המשמש לעקוב אחר קשרי ההורה-ילד. בניגוד למטריצת השכנות, קל בשיטה זו למצוא את שורש העץ. אבל כמו בשיטת מטריצת השכנות שיטה זו אינה שומרת את סדר המצביעים או הרשומות בצומת.

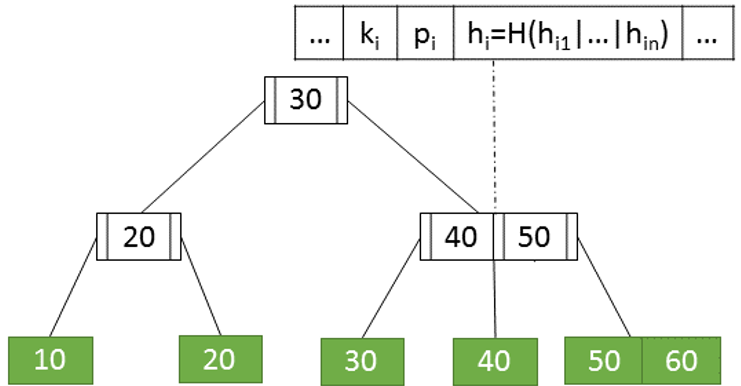
במאמר[‎17] הוצעה שיטה חדשה לאחסון של MHT במסד הנתונים בשם Radix Path Identiﬁers הבאה לתת מענה לחסרונותיהן של השיטות הקיימות. הרעיון הבסיסי הוא להשתמש במספור המבוסס על בסיס מתמטי (radix) מסוים לזיהוי כל מצביע או רשומה בעץ. פרק זה מתאר את השיטה בפירוט.

בהינתן טבלת העובדים הבאה:

|  |  |  |
| --- | --- | --- |
| Salary | Name | ID |
| 1000 | Alice | 10 |
| 2000 | Bob | 20 |
| 3000 | Cindy | 30 |
| 3000 | Dan | 40 |
| 2000 | Eva | 50 |
| 1000 | Felix | 60 |

טבלה 3 - טבלת העובדים

עץ ה-MHT המתאים יבנה באופן הבא:



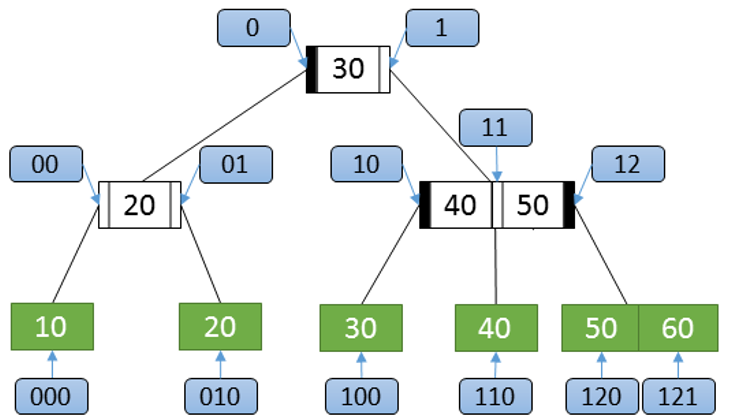
איור 6 - עץ ה-MHT של טבלת העובדים

המידע מוכנס ל-MHT בסדר עולה, כאשר מספר המצביעים של כל צומת (fanout) הוא 3, ולכן כל צומת יכול להחזיק מפתח אחד או שניים.

הרעיון הבסיסי בשיטת Radix Path Identiﬁers מבוסס על השמה של מספרים בכל מצביע של צומת פנימי, ובכל מפתח של עלה, שיאפשרו לזהות אותם בצורה ייחודית ב-MHT. מספרים אלה תלויים בבסיס (Radix) שיבחר, מכל מספר הגדול או השווה ל- fanoutשל ה-MHT. במקרה זה נעשה שימוש ב-Radix 3 השווה ל- fanoutשל

ה-MHT.

הוספה של Radix path identifiers לעץ המתואר באיור 6, יצרו את העץ הבא:



איור 7 MHT עם Radix path identifiers

Radix path identifier של מצביע או מפתח (בעלה) תלוי בדרגה שלו (המסומנת ב-l) ב-MHT ומיקומו בצומת.

דרגת השורש היא 0, ודרגת העלים היא הדרגה המקסימאלית. rb הינו בסיס ה-Radix. f הינו ה-fanout

של ה-MHT.i הינו האינדקס של המצביע/המפתח, היכול להימצא בטווח שבין 0 ל-f.

Radix path identifier או בקיצור rpi מוגדר באמצעות הנוסחא הבאה:

לדוגמא, כדי לחשב את ה-rpi של המפתח 60, יש לקבוע תחילה מהי דרגת המפתח, ומאחר שהיא אינה 0, יש להשתמש בחלק התחתון של הנוסחא המחושבת בבסיס 3. במקרה זה 1i= מאחר ש-60 הוא המפתח השני בעלה. , ו-. הצבתם בנוסחא נותנת תוצאה (בבסיס 3) של 12\*3+1=120+1=121.

מאפייני ה-rpi הם:

* rpi-ים הם מספרים עוקבים באותו הצומת, אך לא בין 2 צמתים שכנים. ניתן לראות באיור 4 כי 10, 11 ו-12 הם עוקבים (בבסיס 3), אך 110 ו-120 אינם עוקבים (בבסיס 3).
* ניתן לחשב את ה-rpi של ההורה בעזרת הבן בעזרת הנוסחא .
* ניתן לחשב את המינימום והמקסימום בכל צומת בעזרת הנוסחאות:

ו-.

* ניתן לחשב את המצביע בעזרת הנוסחא .

## MHT storage in the database

2 מודלים אפשריים לאחסון של Radix path identifiers מבוססיMHT בבסיס הנתונים:

* אחסון כל המידע בטבלת אותנטיקציה אחת הנקראת Single Authentication Table (SAT).
* אחסון כל דרגה של MHT בטבלת אותנטיקציה נפרדת הנקראת Level based Authentication Table (LBAT).

## Single Authentication Table (SAT)

בטכניקה זו מאוחסן כל המידע בטבלת אותנטיקציה אחת כמו בטבלה 4 להלן, המייצגת את העץ מאיור 7:

|  |  |  |  |
| --- | --- | --- | --- |
| Level | Hash | RPI | ID |
| 2 | hash | 0 | -1 |
| 2 | hash | 1 | 30 |
| 1 | hash | 0 | -1 |
| 1 | hash | 1 | 20 |
| 1 | hash | 3 | -1 |
| 1 | hash | 4 | 40 |
| 1 | hash | 5 | 50 |
| 0 | hash | 0 | 10 |
| 0 | hash | 3 | 20 |
| 0 | hash | 9 | 30 |
| 0 | hash | 12 | 40 |
| 0 | hash | 15 | 50 |
| 0 | hash | 16 | 60 |

טבלה 4 - Single Authentication Table (SAT)

כל רשומה בטבלה מיוצגת ע"י מצביע בצומת פנימי או מפתח בעלה של ה-MHT.

עמודת ה-ID בטבלה מייצגת את ה-ID של העובד מטבלה 3.

עבור העלים המיפוי הוא מידי, כל מפתח מייצג את הערך הרלוונטי בעלה.

עבור הצמתים הפנימיים, מספר המצביעים תמיד גדול ב-1 ממספר המפתחות. לכן מפתח אחד מאוחסן בטבלה עם ערך -1 ב-ID (המצביע השמאלי ביותר בכל צומת). שאר המצביעים נשמרים עם ה-ID-ים של המפתחות המתאימים בצומת.

עמודת ה-hash מכילה את פונקציית ה-hash המתאימה למצביעים בצמתים הפנימיים או למפתחות בעלים.

עמודת ה-rpi מכילה את ה- Radix path identifierשל המצביע בצומת הפנימי או של המפתח בעלה.

אמנם המספרים באיור 7 הם מספרים בבסיס 3, אבל בכדי לאחסן אותם בטבלה יש צורך להמירם לבסיס 10 תוך התעלמות מאפסים מובילים. לכן 011 ו-11 יומרו לאותו המספר בבסיס 10. בכדי להבדיל בין 2 rpi-ים זהים, התווספה עמודת הדרגה לטבלה (2 rpi-ים זהים ימצאו בהכרח בדרגות שונות ב-MHT).

## Level Based Authentication Table (LBAT)

בטכניקה זו מאוחסן המידע של כל דרגה של ה-MHT בטבלה נפרדת. לכן נדרשת טבלה קישור נוספת להחזקת המידע על כל טבלאות האותנטיקציה השונות, כמו טבלה 5 להלן.

|  |  |
| --- | --- |
| Table | Level |
| Emp\_2 | 2 |
| Emp\_1 | 1 |
| Employee | 0 |

טבלה 5 - טבלת קישור

|  |  |  |
| --- | --- | --- |
| Hash | RPI | ID |
| Hash | 0 | -1 |
| Hash | 1 | 30 |

טבלה 6 - עובדים ברמה 2 (Emp\_2)

|  |  |  |
| --- | --- | --- |
| Hash | RPI | ID |
| Hash | 0 | -1 |
| Hash | 1 | 20 |
| Hash | 3 | -1 |
| Hash | 4 | 40 |
| Hash | 5 | 50 |

טבלה 7 - עובדים ברמה 1 (Emp\_1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hash | RPI | Salary | Name | ID |
| Hash | 0 | 1000 | Alice | 10 |
| Hash | 3 | 2000 | Bob | 20 |
| Hash | 9 | 3000 | Cindy | 30 |
| Hash | 12 | 3000 | Dan | 40 |
| Hash | 15 | 2000 | Eva | 50 |
| Hash | 16 | 1000 | Felix | 60 |

טבלה 8 –עובדים ברמה 0 (עלים) (Employee)

כל טבלה מייצגת דרגה אחרת בעץ, לכן לא נדרש לשמור את הדרגה בטבלה כפי שנעשה ב-SAT. טבלת העלים

(המייצגת את דרגה 0 בעץ) מכילה בדיוק את אותה כמות רשומות כמו טבלת העובדים (טבלה 3), כך ש-2 הטבלאות ניתנות לאיחוד לטבלה אחת בתוספת עמודות עבור ה-RPI וה-Hash.

## השוואת ביצועים בין 2 הסכמות

בהתחשב בנעילת הטבלאות בפעולות של עדכון והכנסת נתונים, מהיר וקל יותר לעדכן את נתוני האימות ב-LBAT מאשר ב-SAT. בנוסף, מאחר שב-LBAT נתוני האימות מאוחסנים באותה רשומה ביחד עם נתוני העלים, קל יותר לשלוף אותם.

## שליפת נתוני האימות

שליפת נתוני האימות ב-LBAT משמשת לחישוב שורש ה-hash של ה-MHT. ניתן לשלוף את המידע ב-4 דרכים:

Zero-join, Single-join, Multi-join ו-Range-Condition. כל אחת מהדרכים תוצג בעזרת דוגמא על ID 40.

## Multi-join

בגישה זו, כל מידע האימות נשלף מטבלת ה-LBAT הרלוונטית בשליפה אחת, בעזרת שימוש ב-left outer joins. Join-ים אלה יובילו לתוצאות עודפות (redundancy), אותן יצטרך לסנן השולף בכדי לחשב את ה-hash של השורש. המספר 3 המודגש בשליפה הינו הבסיס rb של ה-Radix ששימש לחישובי ה-RPI. RPI/**3** הינו ה-RPI של האבא, RPI/(**3**\***3**) הינו ה-RPI של האבא של האבא וכו'.

בדוגמא להלן, נשלף מידע האימות של הרשומה עם 40 =ID.

select

a0.RPI as RPI0, a0.hash as hash0,

a1.RPI as RPI1, a1.hash as hash1,

a2.RPI as RPI2, a2.hash as hash2

from

Employee emp

left join

Employee a0 on a0.RPI/**3** = emp.RPI/**3**

left join

Emp\_1 a1 on a1.RPI/**3** = emp.RPI/(**3**\***3**)

left join

Emp\_2 a2 on a2.RPI/**3** = emp.RPI/(**3**\***3**\***3**)

where

emp.ID = 40;

## Single-join

בגישה זו, מידע האימות נשלף מכל טבלה בנפרד. עמודת ה-ID אינה מופיעה בטבלאות ולכן בכל שליפה מבוצע Join עם טבלת העובדים עם עמודת ה-RPI.

select

e1.RPI, e1.hash

from

Employee emp

left outer join

Emp\_1 e1 on e1.RPI/3 = emp.RPI/(3\*3)

where

emp.ID = 40;

select

e2.RPI, e2.hash

from

Employee emp

left outer join

Emp\_2 e2 on e2.RPI/3 = emp.RPI/(3\*3\*3)

where

emp.ID = 40;

## Zero-join

בגישה זו כפי ששמה מרמז לא מבוצעים Join-ים לשליפת מידע האימות, אלא מבוצעת שליפה של כל טבלה בנפרד. ה-RPI של הרשומה הרלוונטית נשלף ונשמר במשתנה. לאחר מכן הוא משמש להוצאת מידע האימות מה-LBAT.

declare @rpid as int;

select

@rpid = RPI

from

Employee

where

ID = 40;

select

RPI, hash

from

Employee

where

RPI/3=@rpid/3;

select

RPI, hash

from

Emp\_1

where

RPI/3=@rpid(3\*3)/;

select

RPI, hash

from

Emp\_2

where

RPI/3=@rpid(3\*3\*3)/;

## Range - Condition

בגישה זו יוצרים אינדקס על ה-RPI ומבצעים שימוש ב-Range-condition. השאילתות להלן מדגימות כיצד להשתמש באינדקס על ה-RPI לשליפת מידע האימות מה-LBAT עבורID 40:

declare @rpid as int;

select

@rpid = RPI

from

Employee

where

ID = 40;

select

RPI, hash

from

Employee

where

RPI >= (@rpid/3)\*3 and RPI < (@rpid/3)\*3+3;

select

RPI, hash

from

Emp\_1

where

RPI >= (@rpid/(3\*3))\*3 and RPI < (@rpid/(3\*3))\*3+3;

select

RPI, hash

from

Emp\_2

where

RPI >= (@rpid/(3\*3\*3))\*3 and RPI < (@rpid/(3\*3\*3))\*3+3;

## Data Operations

בסעיף זה יתוארו 5 פעולות על המידע: Select, Update, Batch Update , Insert ו-Delete.

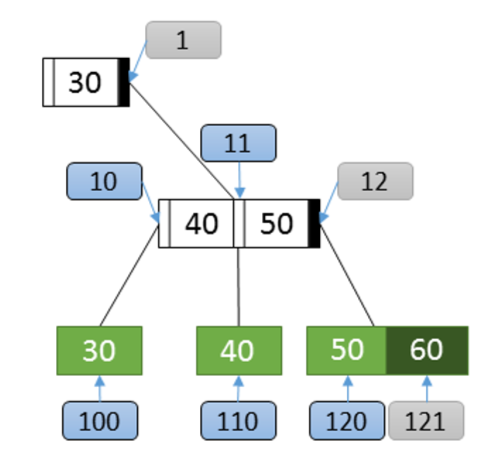
## Select

שליפת טווח של רשומות היא שליפה שונה משליפה של רשומה ספציפית עם מידע האימות שלה. ניקח לדוגמא שליפה המחזירה סט רשומות מ-ID 20 ועד 40. המשתמש יצטרך למצוא את הגבולות של הטווח, כלומר את 10

ו-50. עליו לשלוף את טווח הרשומות מ-ID 20 ועד 40 ללא מידע האימות שלהם. אימות הטווח יתבצע על-ידי אימות המידע של הגבולות שלהם בלבד, כלומר על-ידי אימות המידע של 10 – ו-50 בלבד. אימות הגבולות מבטיח למשתמש את מהימנות המידע של כל הטווח הנשלף.

## Update

עדכון רשומה ספציפית מצריך גם עדכון של פונקציות ה-hash של כל הרשומות המעורבות בנתיב האימות שלה. אם לדוגמא המשתמש מבצע עדכון לרשומה עםID 60, עליו לעדכן את פונקציית ה-hash של הרשומה יחד עם המידע שלה בטבלת העובדים. בנוסף, עליו לעדכן גם את ה-hash של המצביעים 12 ו-1 כפי שניתן לראות באיור 8 להלן:



איור 8 - עדכון רשומה

## Batch update and optimization

נניח שברצוננו לעדכן מספר רשומות בבת אחת. זה דורש עדכון של הרשומות מצד אחד, ושל מידע האימות הרלוונטי שלהן מצד שני. חלק ממידע האימות יעודכן שוב ושוב בשל המבנה ההיררכי של העץ. אם ניקח לדוגמא את העץ באיור 8, אזי עדכון של הרשומה עםID 60, גוררת עדכון של פונקציית ה-hash של הרשומה ושל פונקציות

ה-hash של המצביעים 12 ו-1. עדכון לאחר מכן של ID 50 יגרור עדכון של פונקציית ה-hash של רשומה 50, ושוב עדכון של פונקציות ה-hash של המצביעים 12 ו-1. למעשה, בכל עדכון של אחת הרשומות בתת העץ באיור זה תעודכן פונקציית ה-hash של המצביע 1, המצביע על כל תת העץ באיור. משמעות הדבר שמבוצע עדכון חוזר של אותה רשומת אותנטיקציה בטבלת ה-SQL. אם נדע מהי פונקציית ה-hash המעודכנת ביותר עבור מצביע 1 לאחר כל העדכונים המבוקשים, נוכל לבצע עדכון אחד בלבד של רשומת האותנטיקציה בטבלה. כדי לבצע זאת, עלינו לעקוב אחרי כל הבקשות לעדכונים עבור כל אחת מטבלאות האותנטיקציה (ב-LBAT) או עבור טבלת האותנטיקציה הבודדת (ב-SAT) , להתעלם מהן, ולבצע רק את העדכון האחרון עבור כל אחת מהרשומות הללו. בצורה זו נקטין משמעותית את כמות העדכונים לטבלאות האותנטיקציה. תהליך זה נקרא Merge Update והוא מייעל בצורה משמעותית את העדכון של מספר רשומות בבת אחת.

## Insert ו- Delete

Insert ו- Deleteיכולים להשפיע על מבנה ה-MHT בשלושה דרכים:

1. במקרה הפשוט יושפע רק עלה בודד. לדוגמא כאשר מפתח יתווסף או ימחק מעלה כלשהו. במקרה זה רק המידע בנתיב האימות של העלה המושפע יצטרך להתעדכן.

2. במקרה של שינוי בתת עץ של ה-,MHT כל הרשומות של תת העץ יצטרכו להתעדכן. בנוסף, נתיב האימות של השורש של תת העץ יצטרך להתעדכן.

3. המקרה המסובך ביותר הינו מקרה הכולל הוספה או מחיקה של דרגה שלמה בעץ ה-MHT. הוספה של דרגה למשל מצריכה הקמה של טבלתLBAT חדשה עבורה, הוספה של המידע של הטבלה החדשה בטבלה המכילה את המידע של הLBAT-, ועדכון המידע בכל ה-LBAT.

## סיכום ועבודות עתידיות (למאמר זה)

## סיכום

טכניקות מבוססות Merkle Hash Tree לבדיקת מהימנות המידע נותנות מענה לשלמות (completeness),

נכונות (correctness) ועדכנות המידע (freshness). בנוסף, הן נמנעות משימוש בחתימות דיגיטליות ובכך מונעות תקורה חישובית גבוהה.

השימוש ב-Radix Path Identifier מבוצע תחת הנחות היסוד הבאות:

* רק בעל המידע יכול לשנות אותו.
* כל הבדיקות בוצעו במסדי נתונים מסורתיים כגון SQL server. מסדי נתונים שאינם מבוססיSQL יכולים להוביל לתוצאות אחרות.
* טכניקות cache לביצוע update הן בעלות התקורה הנמוכה ביותר. ללא caching של המידע ע"י בעל המידע, התקורה עלולה לגדול עד ל-100%.
* Insert בסוף הטבלה מובילה לתוצאות טובות יותר מ-insert בתחילת הטבלה. ב-2 המקרים קיימת תקורה משמעותית. לא פורסמו תוצאות עבור Delete.
* כדי לעדכן את המידע, בעל המידע צריך להוריד עותק של כל הטבלה יחד עם נתוני האימות שלה אליו או לחילופין להחזיק את כל המידע ב-cache אצלו.

## עבודות עתידיות

מציאת טכניקה לבדיקת מהימנות המידע שתתבסס על MHT ו-RPI ותהיה בעלת המאפיינים הבאים:

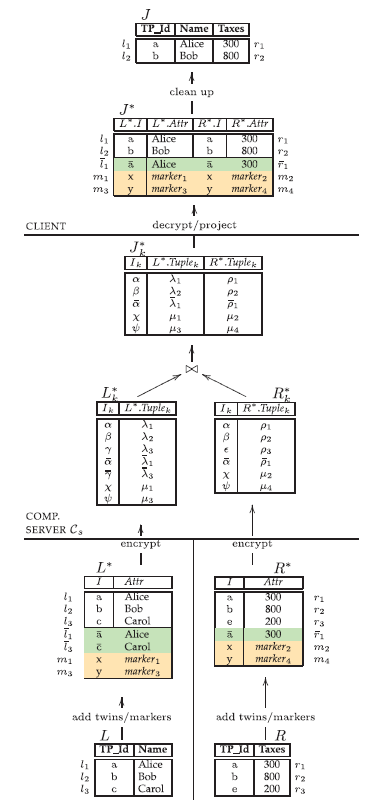
* מספר משתמשים יוכלו לשנות את המידע.
* משתמש לא יידרש לשמור העתק של המידע אצלו בכדי לשנות אותו (נוגד את כל העיקרון של מיקור החוץ של המידע).
* תקורת תקשורת מינימלית שתהיה קרובה ככל הניתן לתקורה של מסדי נתונים מקומיים.
* תמיכה במסדי נתונים של No SQL.

# 

# סקירה מקוצרת

# Integrity for Join Queries in the Cloud [‎16]

המאמר מציג גישה בה קיימים מספר שרתים במיקור חוץ המחלקים את העבודה ביניהם: שרת/ים המספק/ים שירותי אחסון (storage services) ללא מגבלת מקום ובעל/י רוחב פס רחב, ושרת/ים המספק/ים שירותי מחשוב (computational services) בעזרת שרתים וירטואליים זולים יחסית, המאפשר/ים גמישות בכוח המחשוב הנדרש בהתאם לצורך. חלוקה זו מצריכה אינטגרציה בין השרתים המעורבים, ובהקשר שלנו, נדרשת אינטגרציה בין מסד הנתונים הנמצא בשרת אחד או במספר שרתים, לבין ההרצה של שאילתות מסוג join על שרת/ים אחר/ים. הפתרון מניח שהשרת/ים המארח/ים את מסד הנתונים הוא/הם אמין/נים, ואילו השרת/ים המריץ/צים את השאילתות איננו/אינם אמין/ נים. הפתרון המוצע מאפשר למריץ השאילתה לאמת את תוצאותיה בשיתוף פעולה עם שרת/י האחסון האמין/נים. המאמר מציג מספר שכבות הגנה: השכבה הראשונה היא ההצפנה ברמת השורה והצפנת המאפיין/נים המשמש/ים לביצוע ה-join. השכבה השניה מתייחסת להוספה של רשומות על-ידי שרת/י האחסון לתוצאת השאילתה (רשומות המכונות markers) לפני העברתם לשרת המחשוב. בצורה זו שרת המחשוב אינו יכול להבדיל בין רשומות "אמיתיות" לרשומות שהתווספו, ואינו יכול לבחור לבצע מניפולציות רק על מידע "אמיתי".. השכבה השלישית מתייחסת לשכפול של רשומות (twinning), המאפשרת להוסיף כמות קטנה יותר של markers. אם רשומה ששוכפלה תופיע בתוצאה "לבד", היא תצביע על כך שתוצאת השאילתה אינה שלמה.



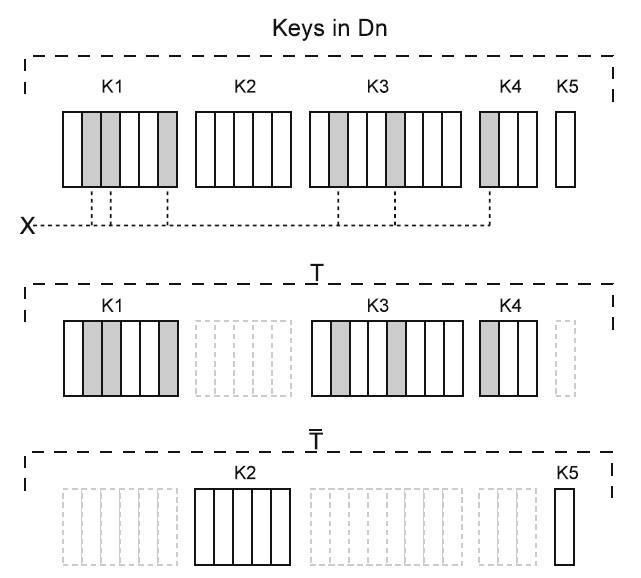
איור9 משמאל מדגים הרצה של שאילתת join עם שכבות ההגנה הנ"ל. הדוגמא מציגה 2 טבלאות L ו-R, הנמצאות על 2 שרתי אחסון. כל אחד מהשרתים מוסיף רשומות markers ומשכפל חלק מהרשומות המקוריות. לאחר מכן מתבצעת הצפנה ברמת כל שורה וברמת מאפייני ה-join, ואת 2 תתי השאילתות המוצפנות מעבירים שרתי האחסון לשרת המחשוב. שרת המחשוב מריץ את ה-join על 2 תתי השאילתות שקיבל, ומעביר את תוצאת ה- joinאל מריץ השאילתה. מריץ השאילתה מפענח את ההצפנה, ומנקה את התוצאה המפוענחת מרשומות ה- markersוהרשומות המשוכפלות לקבלת תוצאת ה-join הרצויה.

עבודות עתידיות - הפתרון המוצע מוגבל לשאילתות join עם תנאי שוויון בלבד, עבודות עתידיות יעסקו בניסיון להרחיב אותו גם לתנאים אחרים כגון תנאי קירוב. בנוסף, הוא מסתמך על העובדה ששרת/י האחסון הוא/הם אמין/אמינים, והוא יורחב בהמשך גם לשרתים שאינם אמינים. מגבלה נוספת של הפתרון המוצע היא ההוספה של אותן רשומות markers ושכפול של אותן הרשומות בכל שרתי האחסון. עבודות עתידיות ינסו למצוא פתרון התומך בשכפול והוספה שונים ובלתי תלויים בכל שרת אחסון.

איור 9 – דוגמא להרצת שאילתת join עם שכפול רשומות aו-c

# Guaranteeing Correctness of Bulk Operations in Outsourced Databases [‎18]

מאמר זה מציג פתרון להבטחת מהימנות הנתונים, שלמותם ועדכנותם עבור נפחים גדולים של מידע. הפתרון המוצג מתוכנן לעבוד ביעילות בעומסי עבודה גדולים, בפעולות של הכנסה (insert) וצירוף בלבד (append-only), תוך כתיבת המידע למסד הנתונים בפעולת כתיבה אחת. הפתרון תומך בשליפה יעילה של נפחים גדולים של מידע, על-ידי הפחתת תקורת התקשורת הנדרשת לאימות הנתונים, הבנויה ממספר רב של מפתחות הנשלפים בפעם אחת. הפתרון מעצב מחדש את בעיית אימות הנתונים משרתים לא אמינים, בעזרת קומבינציה של חתימות מאוחדות של מפות ביליניאריות יחד עם פונקציות hash עמידות להתנגשויות.



איור 10 משמאל מדגים בניה של קבוצות (sets) ואת היחסים ביניהם. מסד הנתונים המכיל מפתחות וערכים מתואר באיור משמאל לאחר 5 פעולות הכנסה של סדרות מידע, כל אחת מהן בעלת מספר רשומות שונה. המפתחות הכלולים במסד הנתונים מסומנים ע"י הקבוצות k1 עד k5. משתמש לגיטימי מעוניין לבצע פעולת קריאה על 6 מפתחות השייכים ל-3 סדרות שונות. קבוצת המפתחות אותו הוא מבקש לקרוא מסומנת ב-X. מאחר ש-X מייצג מפתחות השייכים ל-k1, k3  ו-,k4 כל המפתחות השייכים לסדרות הללו שייכים ל-T. מכיל את המפתחות של יתר הסדרות. קבוצות R1, R3 ו-,R4 מכילות רק את

איור 10 - בניה של קבוצות (sets) והיחסים ביניהם

המפתחות אותם ביקש המשתמש, ששייכים ל- k1, k3 ו-k4 בהתאמה. הקבוצה R מכילה את כל המפתחות מאיחוד של הקבוצות R1, R3 ו-R4. קבוצות Q1, Q3 ו- Q4מכילות את כל המפתחות אותם המשתמש לא ביקש, ששייכים

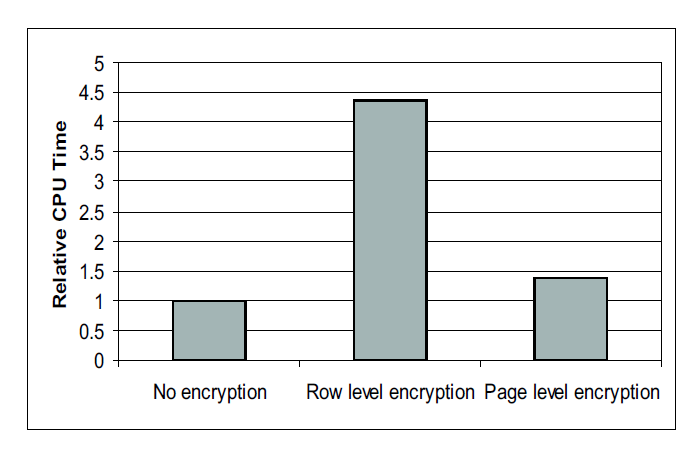
ל- k1, k3 ו-k4. Qמכילה את כל המפתחות מאיחוד של הקבוצות Q1, Q3 ו-Q4. הוכחת האימות מתבצעת על-ידי הנוסחא: ,כאשר מייצג את קבוצת המפתחות הכלולים ב- ( - קבוצת המפתחות השמורות במסד הנתונים לאחר n פעולות הכנסה). אימות יכשל כאשר התוצאה שיחזיר השרת תכיל מפתח אחד לפחות שלא הוכנס על-ידי בעל המידע. נניח לדוגמא ש-R מכיל מפתח כזה, אזי ברור שהנוסחא לא מתקיימת, כיוון ש-R אינו תת-קבוצה של . בכדי להדגים ש-R הוא תת-קבוצה של , על המשתמש להחזיק בכל קבוצת המפתחות . זה אינו אפשרי, מאחר שזה דורש מכל אחד מהמשתמשים לשמור אצלו את כל המפתחות של מסד הנתונים. כדי להימנע מכך, בעל המידע מחזיק את הצובר הקריפטוגרפי , המייצג את מצב המפתחות השמורים במסד הנתונים. צובר זה יעודכן אחרי כל פעולת הכנסה ויהיה זמין לכל המשתמשים. בנוסף, ישלחו *למשתמשים* העדים WQ ו- המייצגים את הקבוצות Q ו-*בהתאמה, ביחד עם התשובה A. המשתמש יחלץ את* R *מ-A, ויבדוק האם העדים שקיבל יחד עם התוצאה מאמתים אותה בהתאם למידע המבוקש ולמצב הנוכחי של מסד הנתונים השמור אצלו.*

*כלומר, הוא יבדוק האם הנוסחא הבאה מתקיימת:* . *הפתרון אינו תומך בפעולות עדכון (*update*), עבודות עתידיות יתרכזו בהרחבתו גם לתמיכה בעדכונים.*

# Providing Database as a Service [‎19]

מאמר זה מציג גישה חדשה לניהול מידע, לפיה ספק מיקור החוץ המארח את מסדי הנתונים של לקוחותיו, מספק להם מנגנונים ליצור, לאחסן, ולגשת למסד הנתונים שלהם כשירות (Database as a Service). שירות זה חוסך ללקוחותיו את הצורך להעסיק מומחים בתחום ניהול מסדי הנתונים DBA)–ים), ואת עלויות החומרה והתוכנה הנדרשות להקמה ולתחזוקה של מסד הנתונים בארגון שלהם, כגון גיבויים, שחזורים ובניית אינדקסים. הוא מציג 3 נקודות מפתח החיוניות להצלחת השירות. ראשית, הנקודה החשובה ביותר הינה שמירה על פרטיות ותקינות המידע של הלקוחות, המתייחסת גם למידע המאוחסן במסד הנתונים וגם למידע המשודר אליו/ממנו. על התשדורות הוא מציע להגן בעזרת פרוטוקולים קיימים ומוכחים כגון SSL ו-TSL. על המידע המאוחסן הוא מציע להגן באחת מ-2 הרמות: ברמת התוכנה, למשל על-ידי אלגוריתם Blowfish שנמצא בניסויים יעיל, קומפקטי ופשוט יותר ליישום לעומת אלגוריתמים ידועים כגון RSA. או ברמת החומרה ע"י חומרה ייעודית כגוןIBM S/390 Cryptographic Coprocessor. המאמר מתאר ניסויים שנערכו ברמת התוכנה לבחינת הצפנות שונות כגון הצפנות ברמת השדה, המגדילות מאד את התקורה החישובית. תוצאות הניסויים הראו שהצפנה של שדות מסוימים משפיעה פחות על התקורה החישובית מהצפנה של שדות אחרים, ולכן הצפנה רק של חלק מהשדות בחלק מהטבלאות יכולה להקטין משמעותית את התקורה החישובית. המאמר מציג מנגנון אוטומטי לכתיבה מחדש של השאילתות המורצות המשפר את זמן הריצה של השאילתות. ברמת החומרה המאמר מציג ניסויים של הצפנות שבוצעו ברמת השורה וברמת הדף. הניסויים הראו שהצפנה ברמת החומרה יעילה יותר מהצפנה ברמת התוכנה בהקשר של מהירות הריצה של השאילתות. הצפנה ברמת הדף מקטינה את תקורת החישוביות לעומת ההצפנה ברמת השורה כפי

שניתן לראות באיור הבא (איור 11):



איור 11 - זמן CPU ללא הצפנה, הצפנה ברמת השורה וברמת הדף

הנקודה השנייה הינה מהירות ביצוע הפעולות על מסד הנתונים המרוחק. הגישה אל מסד נתונים המרוחק אינה מהירה כמו הגישה למסד הנתונים הנמצא בתוך הארגון. לכן יש לאתר את צווארי הבקבוק הגורמים לאיטיות ולטפל בהם כדי להגיע לרמת ביצועים טובה בגישה אל מסד הנתונים המרוחק. הנקודה השלישית מתייחסת לממשק המשתמש. עליו להיות קל לשימוש ונגיש מכל מקום. השירות צריך להיות בנוי כך שעיקר המעמסה החישובית תוטל על צד השרת, ואילו צד הלקוח יצריך רק את הדפדפן לתפעול השירות. בצורה זו השירות זמין מכל מקום מאחר שאינו מצריך התקנה ו/או קינפוג כלשהו מלבד קיומו של הדפדפן. המאמר מציג אב טיפוס של שירות כזה באינטרנט בשם NetDB2. שירות זה מורכב מ-3 שכבות: שכבת תצוגה, שכבת אפליקציה ושכבת ניהול מידע. הפרדה זו מונעת את התלות בין השכבות ומאפשרת לבצע שינויים ביתר קלות בכל שכבה בנפרד, למשל להחליף את שכבת התצוגה בתצוגה אחרת. שכבת התצוגה מורכבת מהדפדפן בצד הלקוח, ומשרת HTTP של

ה-NetDB2. שרת זה מנהל את התקשורת בין הדפדפן לשכבת האפליקציה. שכבת האפליקציה מורכבת מיישומי JAVA והמנוע. היא אחראית ליצור עמודי HTML בהתאם לאינטראקציה עם המשתמש, ולהעביר אותם לשכבת התצוגה. כמו כן היא אחראית על אימות המשתמשים, ניהול ה-session-ים השונים וניהול החיבורים למסד הנתונים. שכבת ניהול המידע אחראי על ניהול מסד הנתונים ועל שרת הגיבויים/שחזורים. כותבי המאמר סבורים ששיטה זו תצבור פופולאריות בשנים הקרובות.

# Crowdsourced Data Integrity Verification for Key-Value Stores in the Cloud ‏[‎23]

מאמר זה עוסק בהבטחת מהימנות הנתונים במסדי נתונים לא רלציוניים מסוג Key-value stores, התומכים באחסון וגישה למידע של צמדי מפתח-ערך. הפתרונות הקיימים להבטחת מהימנות הנתונים של מסדי נתונים רלציוניים אינם מתאימים למסדי נתונים אלה מהסיבות הבאות:

1. נפח המידע המאוחסן במסדי נתונים אלה גבוה בהרבה מזה המאוחסן במסדי נתונים רלציוניים.

2. מודל השאילתות של מסדי נתונים אלה הוא פשוט יותר מזה של מסדי נתונים רלציוניים.

המאמר מציג פתרון השומר על מהימנות הנתונים ב-2 היבטים:

1. נכונות המידע – המידע שיתקבל אצל צרכני המידע אוחסן במקור על-ידי בעל המידע ולא שונה בצד השרת.

2. שלמות המידע – צרכני המידע יקבלו את כל הרשומות הרלוונטיות לשאילתות שהריצו, ולא רק מידע חלקי.

המודל המוצג במאמר מתאפיין בכך שלא מתבצעים עדכונים של המידע בשרת, ולכן שמירה על עדכנות הנתונים אינה רלוונטית למודל זה. הפתרון מושתת על 4 אבני יסוד:

1. פונקציות hash עמידות להתנגשויות, כגון SHA-256, המתאפיינות בכך שקשה חישובית למצוא 2 קלטים שונים המפיקים פלט זהה.

2. 2 מפתחות סודיים אותם חולקים בעל המידע וצרכני המידע להצפנה ואימות המידע.

3. שימוש בקוד אימות מסרים (MAC) לאימות המידע.

4. הצפנת המידע באמצעות הצפנה סימטרית, כגון AES.

הפתרון מבצע שימוש ב-2 טכניקות:

1. קישור רשומות –ההנחה היא שבעל המידע טוען לשרת קבוצה של רשומות בבת אחת, ומקשר בין חלק מהרשומות מאותה הקבוצה. צרכני המידע משתמשים בקישורים הללו לאמת את שלמות המידע. לדוגמא רשומה 1 מקושרת לרשומות 2 ו-3, ולכן אם רשומה 3 לא תוחזר בתוצאת השאילתה יחד עם רשומות 1 ו-2, זו ההוכחה לשינוי בלתי חוקי של המידע. אבל אם השרת לא יחזיר את כל הרשומות מקבוצה מסוימת, הקישורים בין הרשומות בתוך הקבוצה לא יעזרו לזהות זאת. לכן בעל המידע יקשר גם את הקבוצה הקודמת שטען לקבוצה הנוכחית שהוא טוען, שתיקשר בהמשך לקבוצה הבאה שיטען.

2. אימות בעזרת מיקור המונים (Crowdsourced verification) - מספר צרכני מידע משתפים פעולה ביניהם באימות המידע, על מנת לאמת את כל המידע במסד הנתונים החשוב יותר מאימות המידע של שאילתות ספציפיות.

כל צרכן מידע תורם חלק מכוחו החישובי לבניה של שאילתות אימות, הבנויות מהמידע המקושר שקיבל בשאילתות ה"רגילות" שהריץ. השרת אינו יכול להבחין בין שאילתות "רגילות" לשאילתות האימות, מה שיובילו לגילוי שינוי בלתי חוקי של המידע בסופו של דבר. לדוגמא רשומה 1 מקושרת לרשומות 2 ו-3, ומשאילתה "רגילה" בה נשלף שם המשתמש מרשומה 1, יבנו שאילתות אימות לשליפת שם המשתמש מרשומות 2 ו-3, המכילות את שם המשתמש המבוקש. אם תוצאה של לפחות אחת מהשאילתות הללו לא תניב תוצאות, זו ההוכחה לשינוי בלתי חוקי של המידע.

היתרון המרכזי של פתרון זה הוא הקלות שבה ניתן ליישמו במסדי נתונים קיימים מבלי שיש צורך לבצע שינויים כלשהם בצד השרת.

# מימוש

כחלק מעבודה זו, מימשתי Merkle Hash Tree הנבנה אינטראקטיבית על-ידי המשתמש ומציג את ה-Radix path identifiers על העץ, ואת טבלת ה-SAT המתאימה, בעזרתה ניתן לראות בצורה ברורה את החישובים של פונקציית ה-Hash עבור כל עלה וצומת.

בסיס המימוש הינו קוד פתוח של עץ B+ tree מהאתר <http://goneill.co.nz/btree.php>.

הקוד נכתב ב-JS וב-HTML, וכולל 3 קבצים: btree.html, btree.js ו-btree-show.js.

יש לשמור את 3 הקבצים באותה התיקייה. הרצה תתבצע ע"י לחיצה כפולה (double click) על btree.html.

אם מתקבלת אזהרה על חסימה של הרצת סקריפטים בדפדפן של אקספלורר, יש ללחוץ על allow blocked content בכדי להמשיך בהרצה.

# מגבלות והנחות יסוד

Radix Path Identifiersובסיס החישוב של ה- (fanout) גודל העץ

בחירה של גודל העץ (ה-fanout שלו) מוגבלת לגודל מינימלי של 3, במטרה לאפשר להחזיק 2 מפתחות בצומת. בחירה ב-3 משמעותה שמספר המצביעים של כל צומת הוא לכל היותר 3, וכל צומת יכול להחזיק מפתח אחד או שניים.

בסיס החישוב (radix) של ה- ,Radix Path Identifiersשכזכור צריך להיות בגודל של ה- fanoutאו גדול יותר, יהיה בגודל ה-fanout שבחר המשתמש. כלומר, אם נבחר גודל עץ של 3, החישובים יבוצעו לפי .

מפתחות כפולים

התוכנית לא מאפשרת הכנסה של מפתחות כפולים לעץ. אם המפתח 12 קיים בעץ, ניסיון הכנסה של מפתח 12 נוסף לא יבצע דבר. מגבלה זו מקלה על הבנת החישובים.

פונקציית hash

בחרתי בפונקציית hash פשוטה, קרי . המטרה בבחירה של פונקציה פשוטה כזו היא להקל על הבנת החישובים. מימשתי בתוכנית גם פונקציה של Sha1, אך קשה מאד לעקוב אחר החישובים כשמשתמשים בה. בנוסף, אני משרשר את הסימן "|" בין השרשורים, על מנת שניתן יהיה לראות מאיזה שרשורים כל חישוב מורכב.

לדוגמא: עלה עם 2 מפתחות 10 ו-20, להן תוצאות חישוב פונקציות ה-hash התואמות הן 2 ו-4 בהתאמה. פונקציית ה-hash המשורשרת של כל העלה תהיה 4|2, וזה ערך ה-hash שיופיע בטבלת ה-SAT במצביע אל עלה זה. התוכנית מציגה מעל העץ המצויר את ערך ה-hash של שורש העץ לנוחות המשתמש, מאחר שהוא אינו מופיע בטבלת ה-SAT של העץ.

אימות

המסך מציג 2 ערכי hash: את ערך ה-hash הנוכחי (Current Root hashcode), ואת ערך ה-hash הנכון

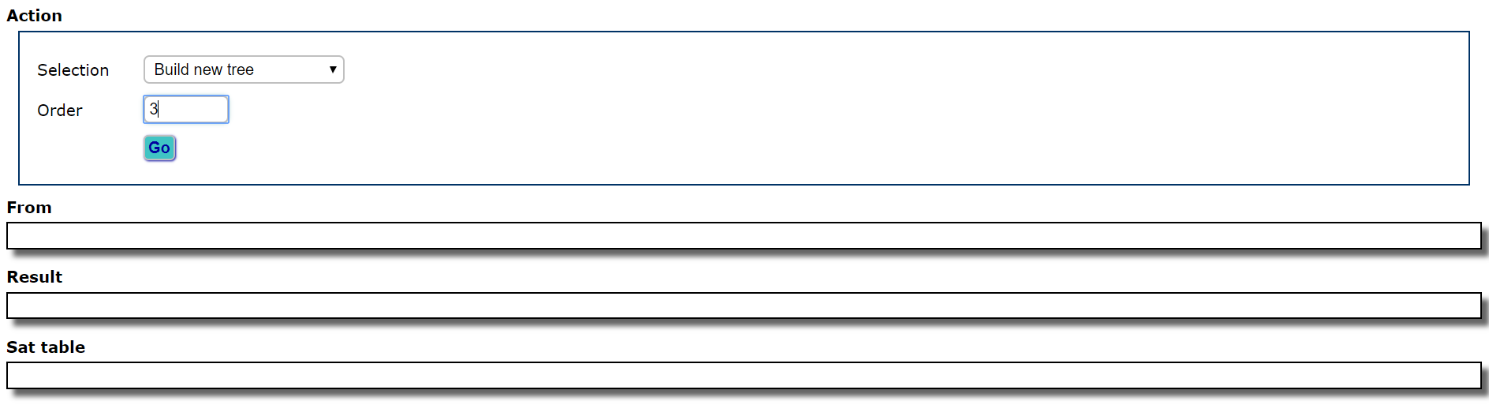
(Correct Root hashcode). שינוי של מפתח על ידי גורם שאינו מורשה, יוביל לכך שערך ה-hash הנוכחי יהיה שונה מהערך ה-hash הנכון ולכן יצבע באדום. כל עוד הם שווים, הוא יצבע בצבע ירוק.

# הרצה

בניית עץ חדש

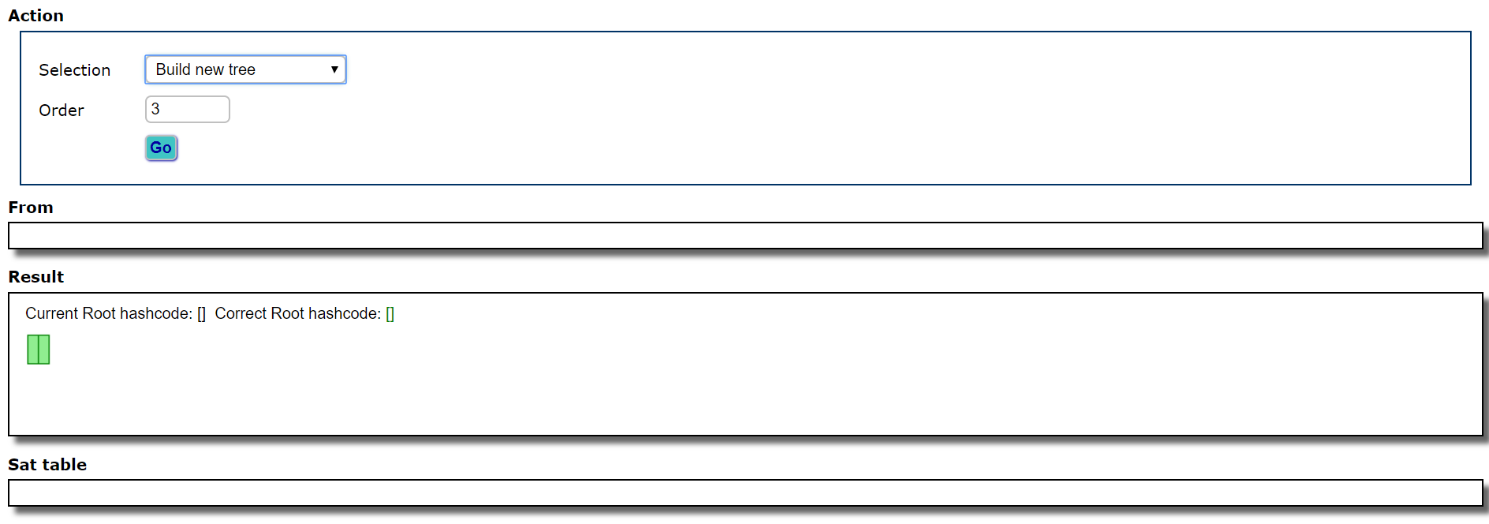
הפעולה הראשונה שיש לבצע היא בחירה של גודל העץ החדש (בחירת ה-fanout שלו). הגודל המינימלי לבחירה הוא 3, משמע מספר המצביעים של כל צומת הוא לכל היותר 3, וכל צומת יכול להחזיק מפתח אחד או שניים.

יש לבחור ב-Build new tree, להקליד את הגודל הרצוי בתיבה ,Order וללחוץ על כפתור ה-Go.



איור 12 – בחירת גודל העץ חדש

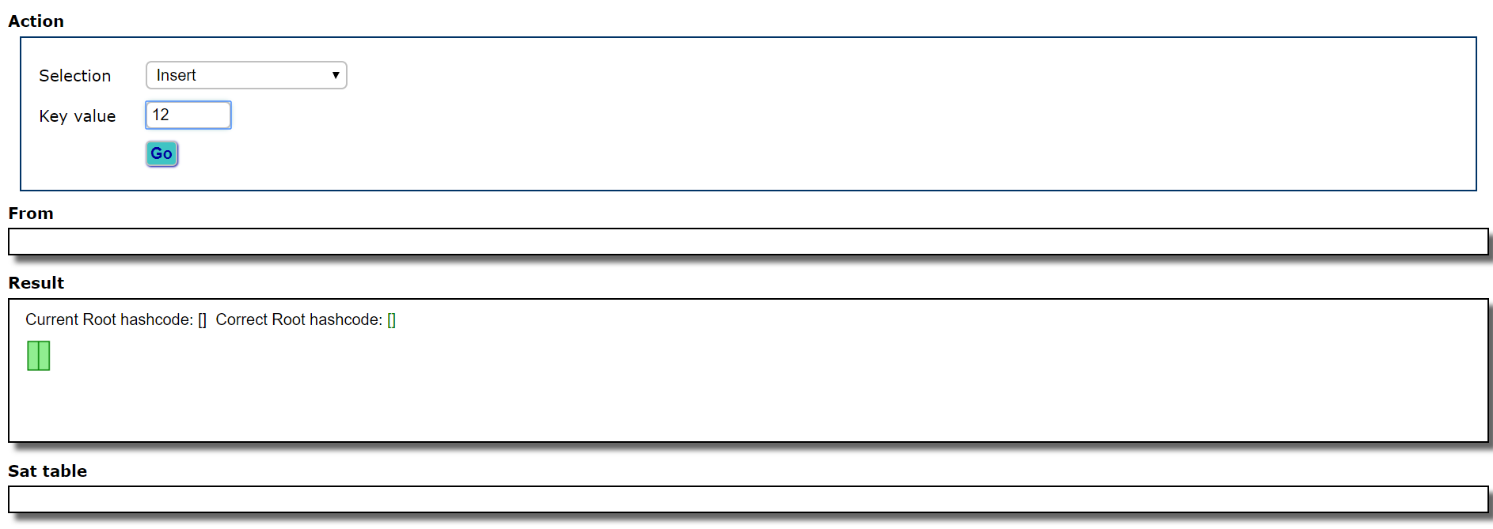
יבנה עץ חדש ויוצג המסך הבא:



איור 13 - בניית העץ חדש

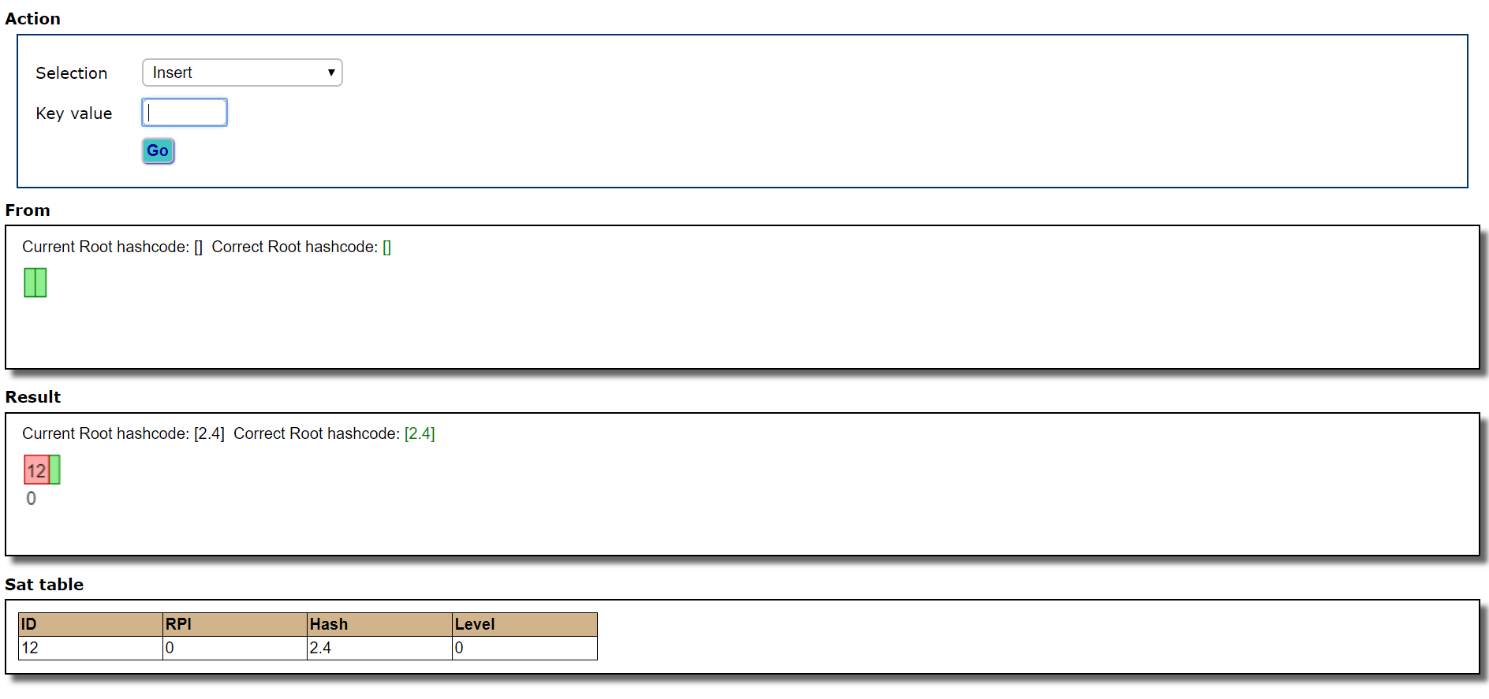
הכנסת מפתחות חדשים

כעת יש להכניס מפתחות חדשים לעץ. יש לבחור ב-Insert, להקליד את המפתח הרצוי בתיבה ,Key value וללחוץ על כפתור ה-Go, כפי שניתן לראות באיור הבא:



איור 14 - בחירת מפתח חדש להכנסה לעץ

המפתח החדש יוכנס לעץ ויוצג המסך הבא:



איור 15 - לאחר הכנסה של מפתח לעץ

בחלונית from ניתן לראות את העץ טרם ביצוע הפעולה האחרונה. בחלונית Result ניתן לראות את העץ המתקבל לאחר הכנסת המפתח 12, ובחלונית Sat table את טבלת ה-SAT התואמת לעץ.

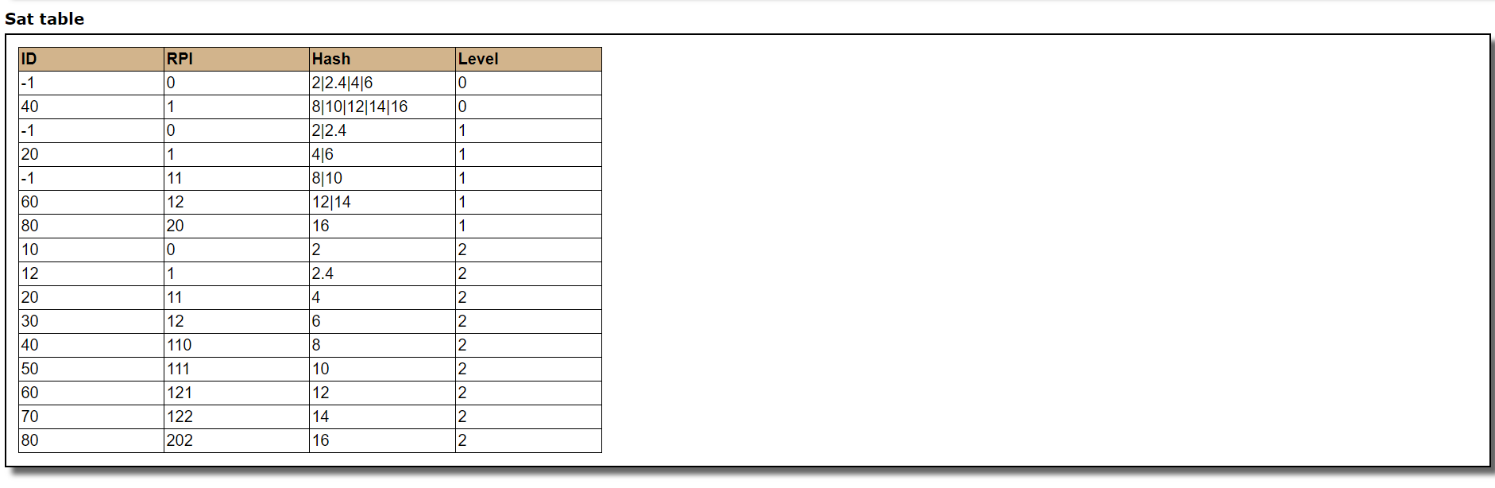
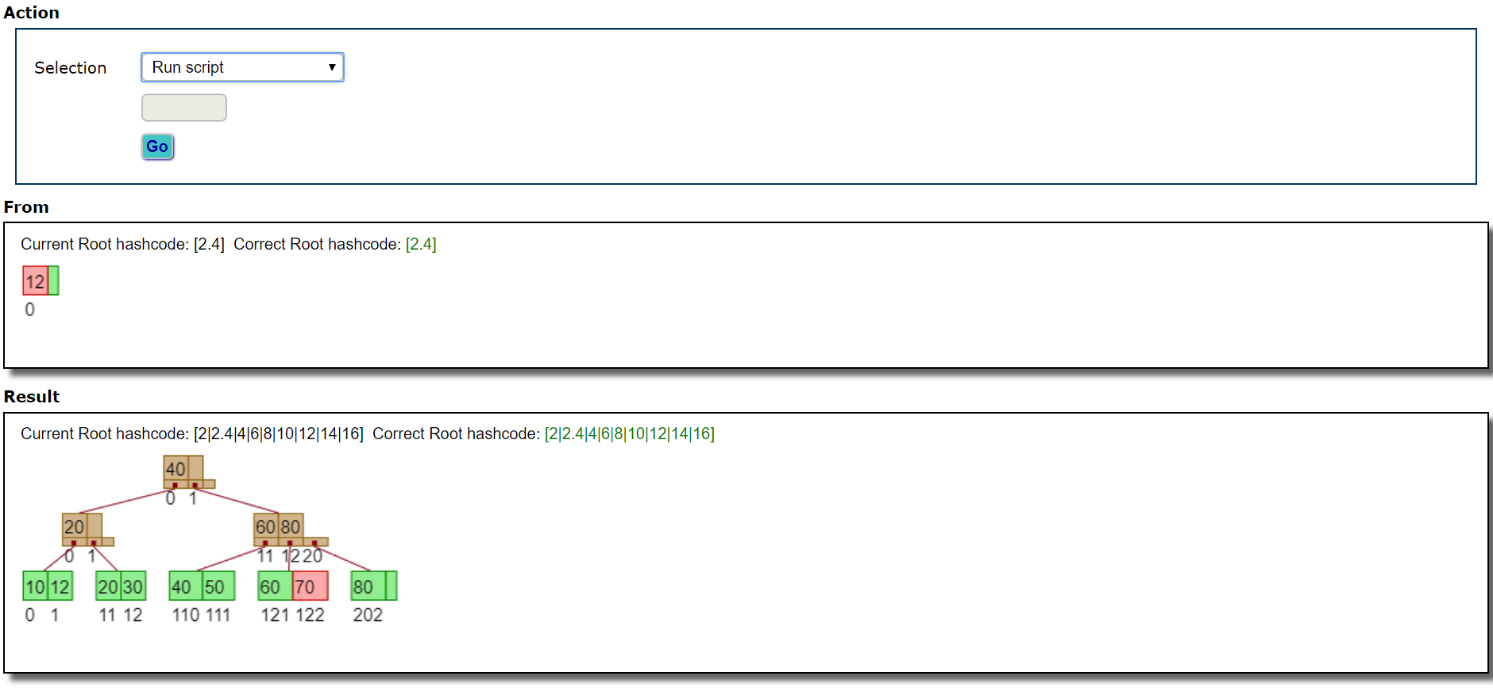
העלה מהווה גם את שורש העץ, לכן ה-hashcode של שורש העץ שווה לזה של העלה והוא .

הכנסת מפתחות חדשים על-ידי סקריפט

ניתן להכניס מספר מפתחות בריצה אחת, על-ידי בחירה ב-Run script. בפועל מתבצעת הכנסה של 8 מפתחות

10, 20, 30, 40, 50, 60, 70 ו-80 שנבחרו hard coded בתוכנית.

לאחר הרצת הסקריפט יוצג המסך הבא:

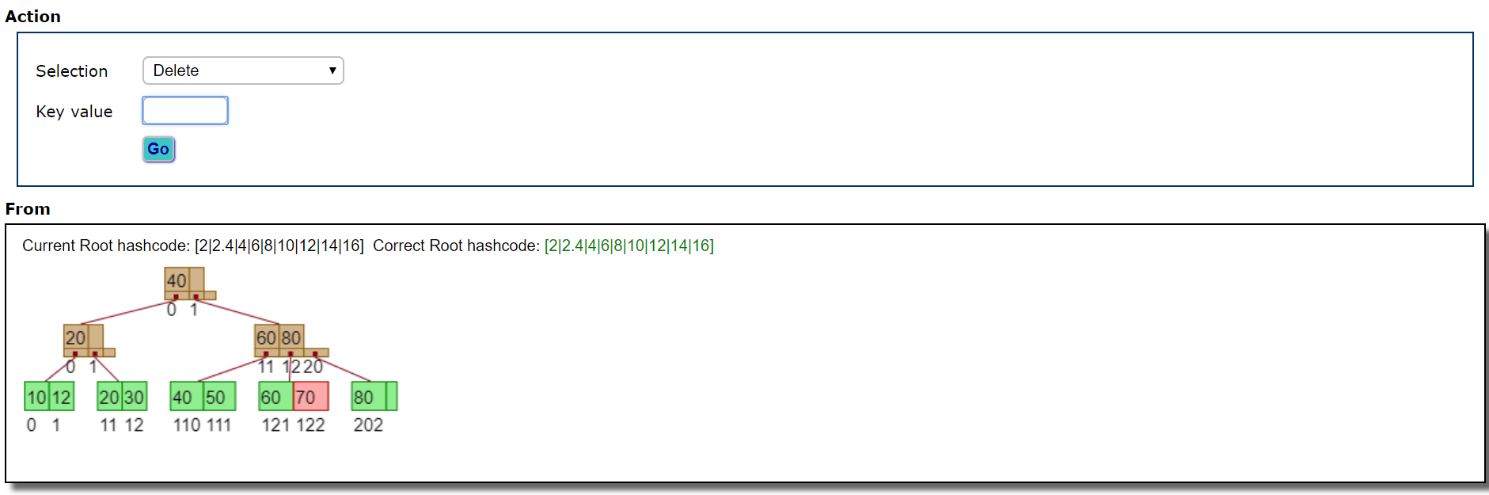


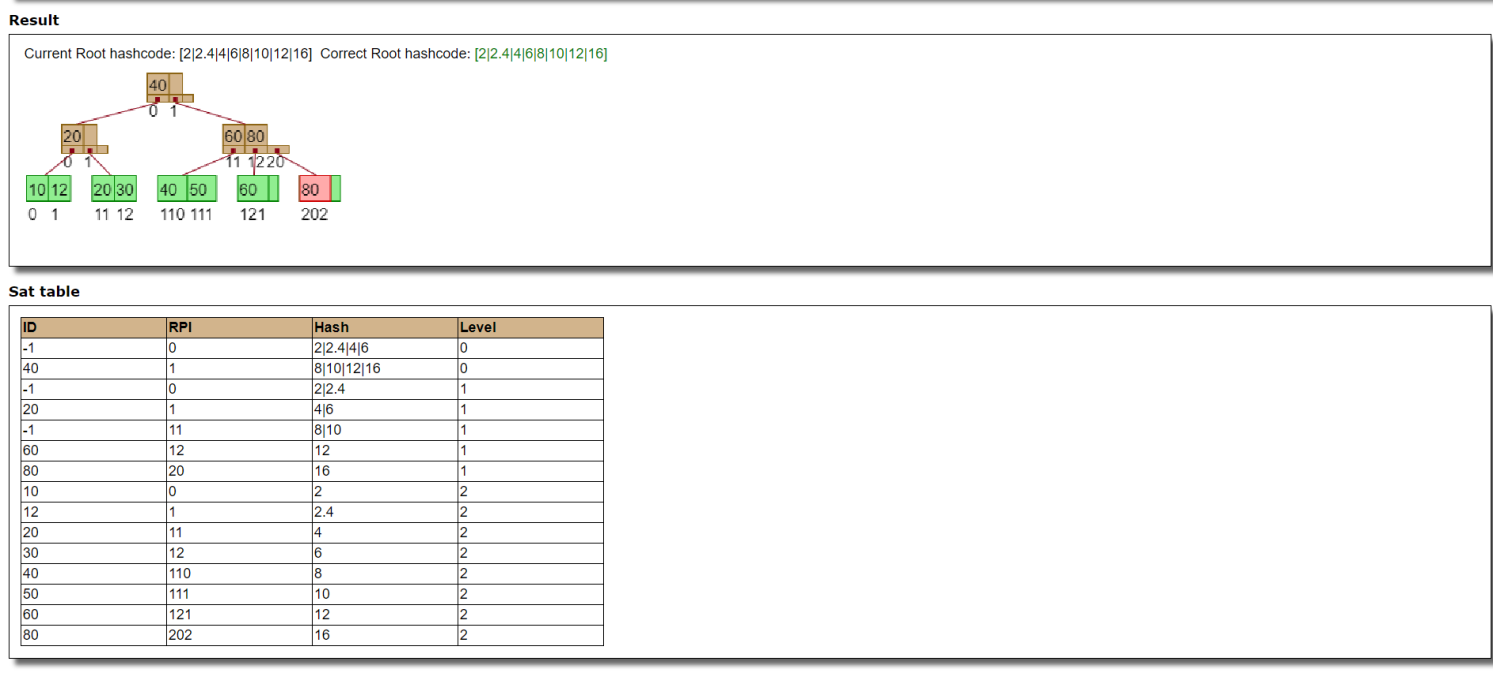
איור 16 - לאחר הרצת הסקריפט להכנסת מפתחות

מחיקת מפתחות

כדי למחוק מפתחות מהעץ, יש לבחור ב-Delete, להקליד את המפתח שברצוננו למחוק בתיבה ,Key value וללחוץ על כפתור ה-Go.

לאחר המחיקה של מפתח 70 יוצג המסך הבא:



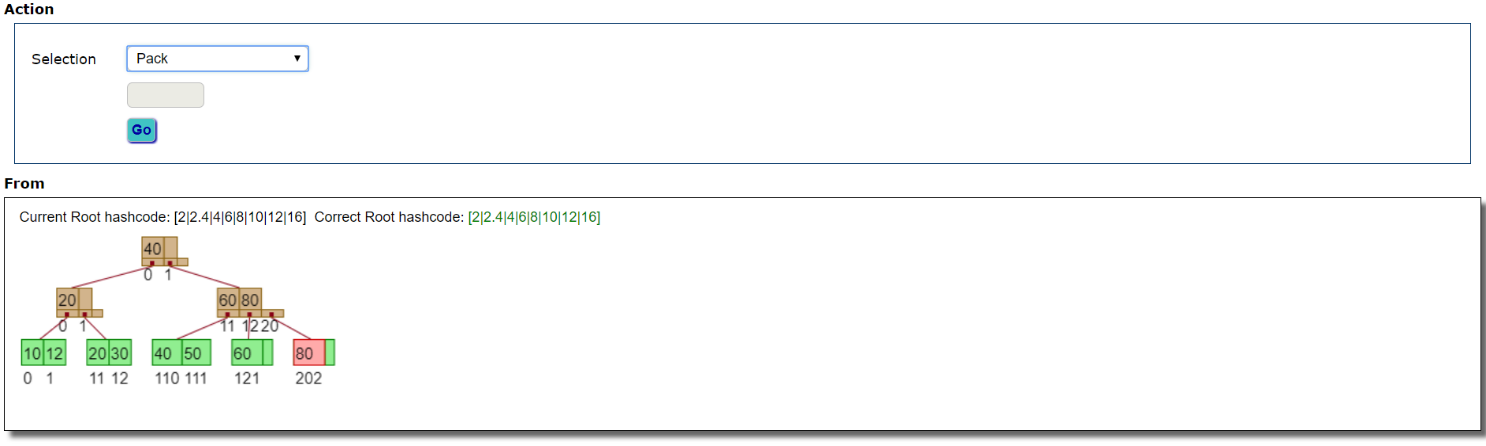


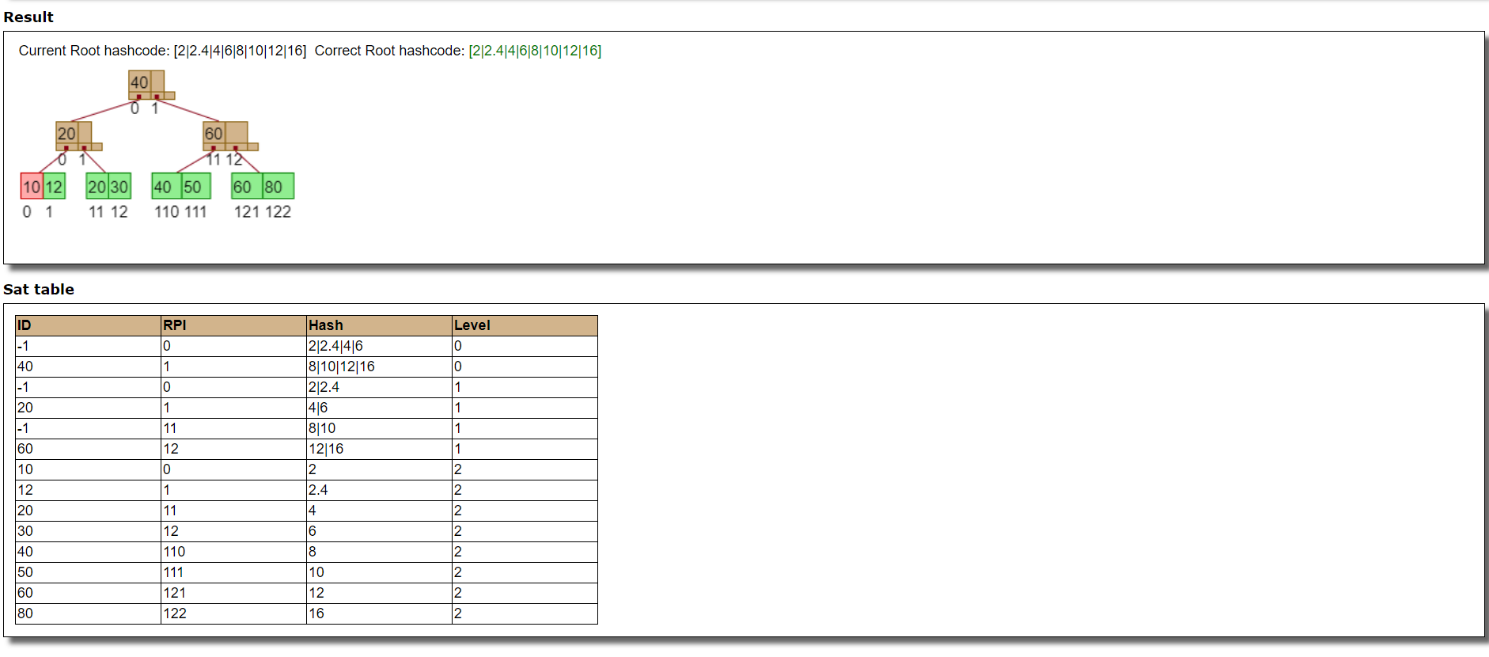
איור 17 - לאחר מחיקת מפתח 70

כיווץ העץ (pack)

בעקבות הכנסה ומחיקה של מפתחות, יהיו צמתים ועלים בעץ שאינם מלאים. פעולת הכיווץ מכווצת את העץ למספר עלים וצמתים מינימלי ככל הניתן.

לאחר הכיווץ יוצג המסך הבא:





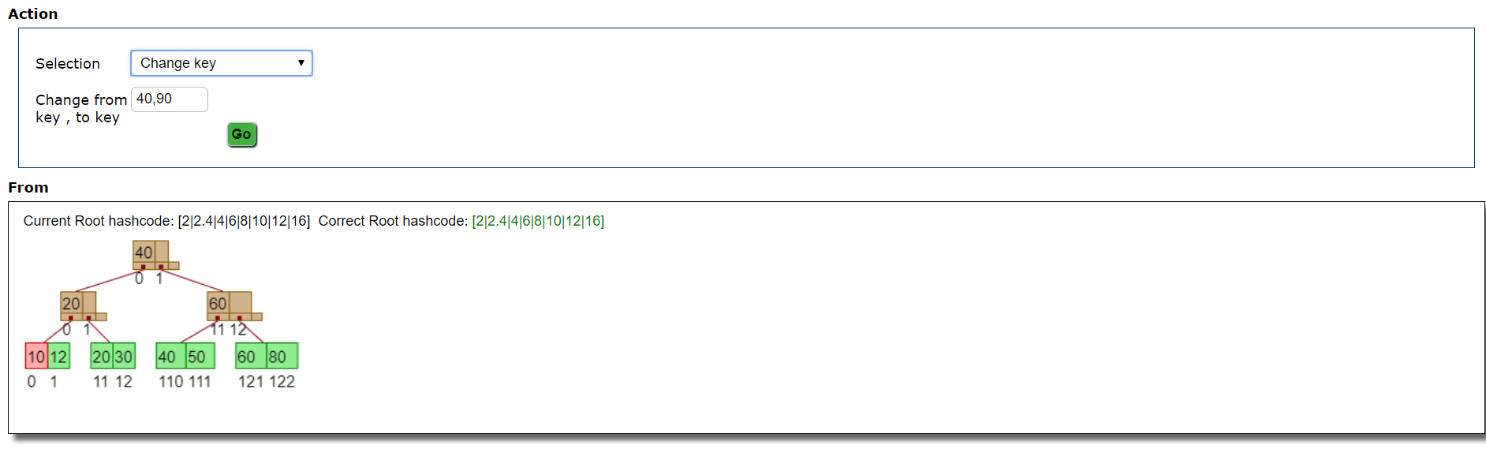
איור 18 - לאחר פעולת הכיווץ

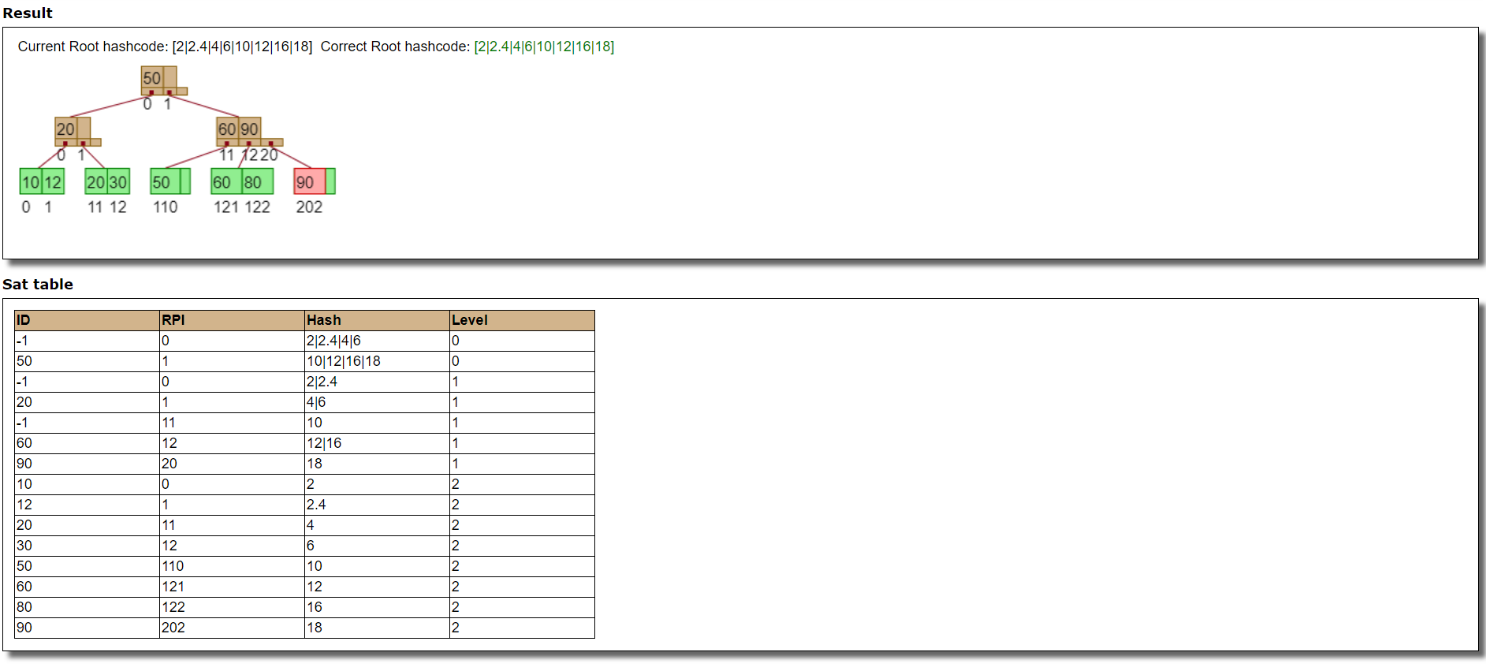
ניתן לראות שכיווץ העץ לא השפיע על חישוב ה- hashcodeשל השורש שלו, שנותר כשהיה טרם הכיווץ.

החלפת מפתחות

כדי להחליף מפתחות בעץ יש לבחור ב-Change key, להקליד בתיבה Change from key , to key"" את המפתח המוחלף מופרד בפסיק במפתח שברצוננו להחליף, וללחוץ על כפתור ה-Go.

נקליד בתיבה 90 , 40 כדי להחליף את מפתח 40 במפתח 90, לאחר מכן יתקבל המסך הבא:





איור 19 - לאחר החלפת מפתח 40 במפתח 90

הצגת ההיסטוריה

ניתן לצפות בהיסטוריית הפעולות שבוצעו עד כה, על-ידי בחירה ב-Show history ואח"כ ללחוץ על כפתור ה-Go.

נקבל את תצוגת ההיסטוריה הבאה:

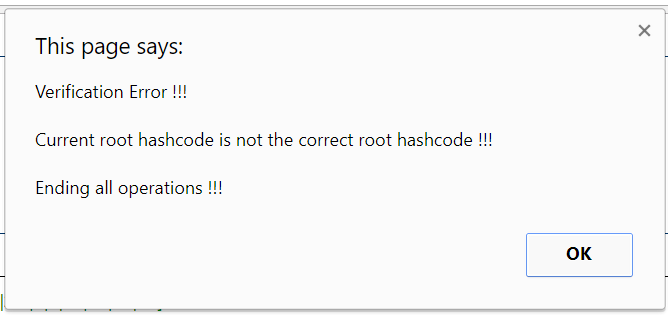


איור 20 - הצגת ההיסטוריה

החלפת מפתחות על ידי גורם לא מורשה

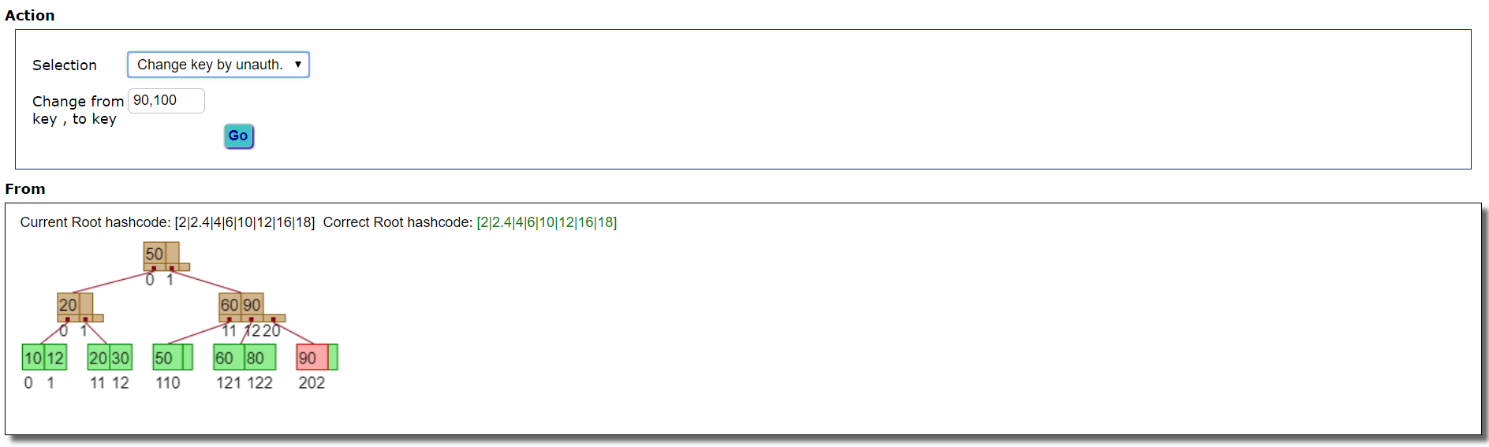
כדי לדמות החלפת מפתחות בעץ על ידי גורם לא מורשה, יש לבחור ב-Change key by unauth., להקליד בתיבה Change from key, to key"" את המפתח המוחלף מופרד בפסיק במפתח שברצוננו להחליף, וללחוץ על כפתור ה-Go.

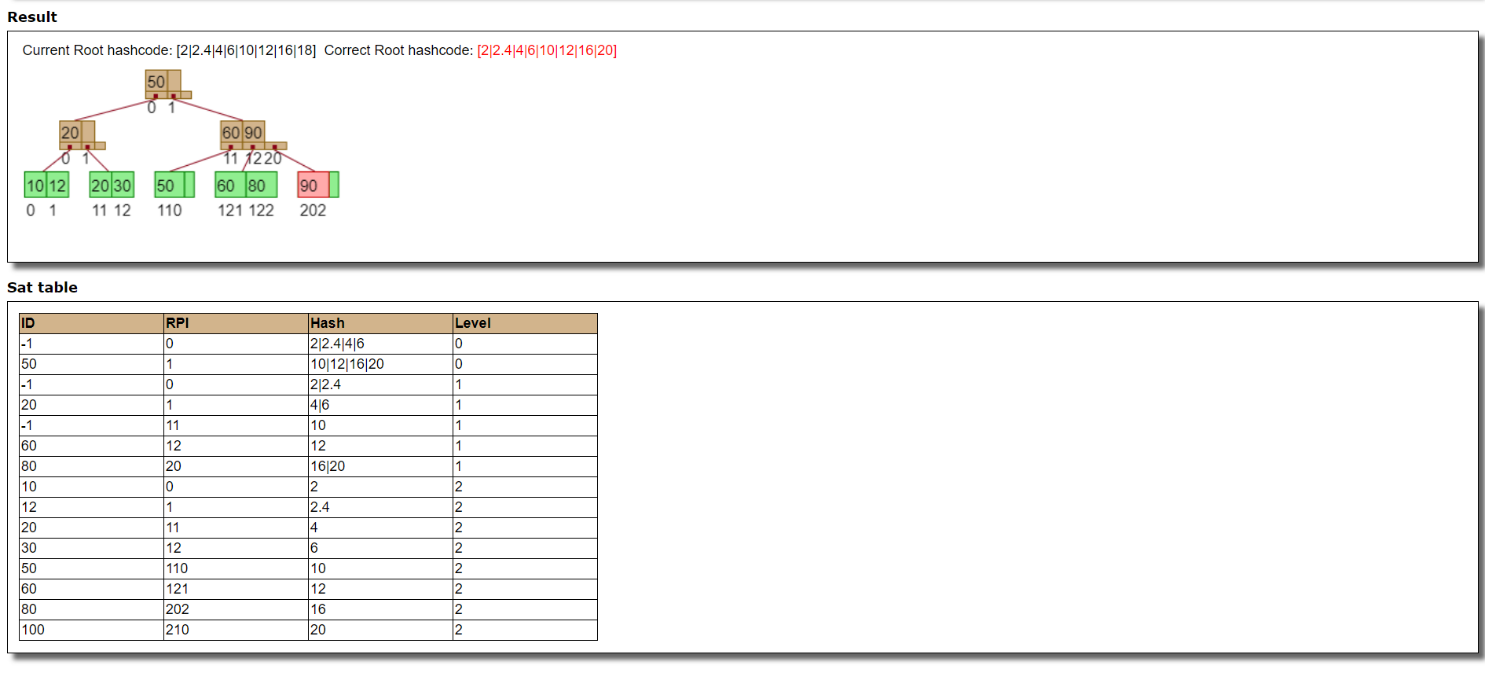
נקליד בתיבה 100 , 90 כדי להחליף את מפתח 90 במפתח 100, לאחר מכן תתקבל הודעת השגיאה הבאה:



איור 21 - שגיאת וריפיקציה

והמסך הבא:





איור 22 - לאחר החלפת מפתח 90 במפתח 100 על ידי גורם לא מורשה

ה- hashcode של השורש הוא 20 במקום 18 כפי שרואים בטבלה, וזה יגרום לגילוי השגיאה !

# סיכום

מיקור חוץ של מסדי נתונים הוא נושא מרתק שהפך בשנים האחרונות לפופולארי בעיקר בענן. יחד עם השימוש הגובר בהם, מתחזק גם הצורך לשמור על המידע המאוחסן בהם, ועל המידע המשודר מהם ללקוחותיהם וחזרה. אמצעי הגנה הן מתוקפים מבחוץ התוקפים את התשדורות ממסדי הנתונים ואליהם, והן מתוקפים מבפנים כלומר מספקי מיקור החוץ בעצמם.

קיימים פתרונות טובים לאבטחת תשדורות הנתונים כנגד תוקפים מבחוץ כגון SSL, אך הם אינם בתכולה של עבודה זו.

בעבודה זו הוצגו פתרונות להגנה על המידע מספק מיקור החוץ עצמו, המשתמשים בכלים קריפטוגרפים שונים להצפנת המידע, על מנת להבטיח שגורם בלתי מורשה לא יוכל לשנות את ההצפנה מבלי שיתגלה השינוי.

הפרק הראשון סקר בפירוט 2 שיטות הצפנה: הצפנה מבוססת חתימות המשתמשת בטכניקה של איחוד חתימות לאמת את תוצאות השאילתות, והצפנה מבוססת עץ (tree-based) המשתמשת ב-Merkle hash tree לביצוע ההצפנה והאימות.

הפרק השני הציג בקצרה מגוון פתרונות הקשורים לנושא: הוצג פתרון של מספר שרתים במיקור חוץ המחלקים את העבודה ביניהם בביצוע Join-ים, הוצגו האתגרים העומדים בפני ספק מיקור החוץ המציע את מסד הנתונים כשירות, הוצג פתרון התומך בהכנסה ושליפה של כמות גדולה של מידע לשרת מיקור החוץ, והוצג פתרון להבטחת מהימנות הנתונים במסדי נתונים לא רלציוניים.

הפרק השלישי תיאר את המימוש המעשי.

הוצאת מסדי נתונים למיקור חוץ מעניקה יתרונות לארגונים, היכולים למקד את פעילותם בפעילויות הליבה שלהם, ולחסוך בעלויות הנגזרות משמירת המידע בארגון, אך יוצרת סיכונים למידע המאוחסן. מגוון הפתרונות הקיימים להצפנת המידע מכילים חסרונות כגון: התאמה לסוגי שאילתות מסוימות בלבד, תקורת חישוביות גבוהה ואי התאמה למודל של מספר חותמים. יש צורך להמשיך ולחקור את הנושא בכדי למצוא פתרונות חדשים שיתנו מענה לחסרונות הללו.

# ביבליוגרפיה

1. Einar Mykletun, [Maithili Narasimha](http://dblp.uni-trier.de/pers/hd/n/Narasimha:Maithili), [Gene Tsudik](http://dblp.uni-trier.de/pers/hd/t/Tsudik:Gene):  
   **Authentication and integrity in outsourced databases**. [TOS 2(2)](http://dblp.uni-trier.de/db/journals/tos/tos2.html#MykletunNT06): 107-138 (2006).
2. L. Harn:

**Batch verifying rsa signatures. Electronic Letters, vol.34, pp. 1219–1220, Apr 1998.**

1. [Dan Boneh](http://dblp.uni-trier.de/pers/hd/b/Boneh:Dan), [Craig Gentry](http://dblp.uni-trier.de/pers/hd/g/Gentry:Craig), [Ben Lynn](http://dblp.uni-trier.de/pers/hd/l/Lynn:Ben), [Hovav Shacham](http://dblp.uni-trier.de/pers/hd/s/Shacham:Hovav): **Aggregate and Verifiably Encrypted Signatures from Bilinear Maps.** Proceedings of [EUROCRYPT 2003](http://dblp.uni-trier.de/db/conf/eurocrypt/eurocrypt2003.html#BonehGLS03): 416-432.
2. David Naccache, [David M'Raïhi](http://dblp.uni-trier.de/pers/hd/m/M=Ra=iuml=hi:David), [Serge Vaudenay](http://dblp.uni-trier.de/pers/hd/v/Vaudenay:Serge), [Dan Raphaeli](http://dblp.uni-trier.de/pers/hd/r/Raphaeli:Dan):

**Can D.S.A. be improved? Complexity Trade-Offs with the Digital Signature Standard.** Proceedings of [EUROCRYPT 1994](http://dblp.uni-trier.de/db/conf/eurocrypt/eurocrypt94.html#NaccacheMVR94): 77-85.

1. [Colin Boyd](http://dblp.uni-trier.de/pers/hd/b/Boyd:Colin), [Chris Pavlovski](http://dblp.uni-trier.de/pers/hd/p/Pavlovski:Chris):  
   **Attacking and Repairing Batch Verification Schemes.** Proceedings of [ASIACRYPT 2000](http://dblp.uni-trier.de/db/conf/asiacrypt/asiacrypt2000.html#BoydP00): 58-71.
2. Muhammad Saqib Niaz, [Gunter Saake](http://dblp.uni-trier.de/pers/hd/s/Saake:Gunter):  
   **Merkle Hash Tree based Techniques for Data Integrity of Outsourced Data.** Proceedings of [GvD 2015](http://dblp.uni-trier.de/db/conf/gvd/gvd2015.html#NiazS15): 66-71.
3. Qingji Zheng, [Shouhuai Xu](http://dblp.uni-trier.de/pers/hd/x/Xu:Shouhuai), [Giuseppe Ateniese](http://dblp.uni-trier.de/pers/hd/a/Ateniese:Giuseppe):   
   **Efficient query integrity for outsourced dynamic databases.** Proceedings of [CCSW 2012](http://dblp.uni-trier.de/db/conf/ccs/ccsw2012.html#ZhengXA12):

71-82.

1. [Feifei Li](http://dblp.uni-trier.de/pers/hd/l/Li:Feifei), [Marios Hadjieleftheriou](http://dblp.uni-trier.de/pers/hd/h/Hadjieleftheriou:Marios), [George Kollios](http://dblp.uni-trier.de/pers/hd/k/Kollios:George), [Leonid Reyzin](http://dblp.uni-trier.de/pers/hd/r/Reyzin:Leonid):  
   **Dynamic authenticated index structures for outsourced databases.** Proceedings of [SIGMOD Conference 2006](http://dblp.uni-trier.de/db/conf/sigmod/sigmod2006.html#LiHKR06): 121-132.

1. [HweeHwa Pang](http://dblp.uni-trier.de/pers/hd/p/Pang:HweeHwa), [Jilian Zhang](http://dblp.uni-trier.de/pers/hd/z/Zhang:Jilian), Kyriakos Mouratidis:  
   **Scalable Verification for Outsourced Dynamic Databases.** [PVLDB 2(1)](http://dblp.uni-trier.de/db/journals/pvldb/pvldb2.html#PangZM09): 802-813 (2009).
2. Roberto Tamassia, [Nikos Triandopoulos](http://dblp.uni-trier.de/pers/hd/t/Triandopoulos:Nikos):  
   **Certification and Authentication of Data Structures.** Proceedings of [AMW 2010](http://dblp.uni-trier.de/db/conf/amw/amw2010.html#TamassiaT10).

1. [Giuseppe Ateniese](http://dblp.uni-trier.de/pers/hd/a/Ateniese:Giuseppe), [Seny Kamara](http://dblp.uni-trier.de/pers/hd/k/Kamara:Seny), [Jonathan Katz](http://dblp.uni-trier.de/pers/hd/k/Katz:Jonathan):  
   **Proofs of storage from homomorphic identification protocols.** Proceedings of [ASIACRYPT 2009](http://dblp.uni-trier.de/db/conf/asiacrypt/asiacrypt2009.html#AtenieseKK09): 319-333.

1. [Giuseppe Ateniese](http://dblp.uni-trier.de/pers/hd/a/Ateniese:Giuseppe), [Randal C. Burns](http://dblp.uni-trier.de/pers/hd/b/Burns:Randal_C=), [Reza Curtmola](http://dblp.uni-trier.de/pers/hd/c/Curtmola:Reza), [Joseph Herring](http://dblp.uni-trier.de/pers/hd/h/Herring:Joseph), [Lea Kissner](http://dblp.uni-trier.de/pers/hd/k/Kissner:Lea), [Zachary N. J. Peterson](http://dblp.uni-trier.de/pers/hd/p/Peterson:Zachary_N=_J=), [Dawn Xiaodong Song](http://dblp.uni-trier.de/pers/hd/s/Song:Dawn_Xiaodong):  
   **Provable data possession at untrusted stores.** Proceedings of [ACM Conference on Computer and Communications Security 2007](http://dblp.uni-trier.de/db/conf/ccs/ccs2007.html#AtenieseBCHKPS07): 598-609.

1. [Hovav Shacham](http://dblp.uni-trier.de/pers/hd/s/Shacham:Hovav), [Brent Waters](http://dblp.uni-trier.de/pers/hd/w/Waters:Brent):  
   **Compact Proofs of Retrievability.** [J. Cryptology 26(3)](http://dblp.uni-trier.de/db/journals/joc/joc26.html#ShachamW13): 442-483 (2013).

1. [Mihir Bellare](http://dblp.uni-trier.de/pers/hd/b/Bellare:Mihir), [Gregory Neven](http://dblp.uni-trier.de/pers/hd/n/Neven:Gregory):  
   **PMulti-signatures in the plain public-Key model and a general forking lemma.** Proceedings of [Computer and Communications Security 2006](http://dblp.uni-trier.de/db/conf/ccs/ccs2006usa.html#BellareN06): 390-399.

1. [Ari Juels](http://dblp.uni-trier.de/pers/hd/j/Juels:Ari), [Burton S. Kaliski Jr.](http://dblp.uni-trier.de/pers/hd/k/Kaliski_Jr=:Burton_S=):  
   **Pors: proofs of retrievability for large files.**Proceedings of [ACM Conference on Computer and Communications Security 2007](http://dblp.uni-trier.de/db/conf/ccs/ccs2007.html#JuelsK07): 584-597.
2. Sabrina De Capitani di Vimercati z, [Sara Foresti](http://dblp.uni-trier.de/pers/hd/f/Foresti:Sara), [Sushil Jajodia](http://dblp.uni-trier.de/pers/hd/j/Jajodia:Sushil), [Stefano Paraboschi](http://dblp.uni-trier.de/pers/hd/p/Paraboschi:Stefano), [Pierangela Samarati](http://dblp.uni-trier.de/pers/hd/s/Samarati:Pierangela):  
   **Integrity for Join Queries in the Cloud.** [IEEE Trans. Cloud Computing 1(2)](http://dblp.uni-trier.de/db/journals/tcc/tcc1.html#VimercatiFJPS13): 187-200 (2013).

1. [Wei](http://dblp.uni-trier.de/pers/hd/w/Wei:Wei), [Ting Yu](http://dblp.uni-trier.de/pers/hd/y/Yu:Ting):  
   **Integrity Assurance for Outsourced Databases without DBMS Modification**. Proceedings of DBsec 2014: 1-16.

1. [Luca Ferretti](http://dblp.uni-trier.de/pers/hd/f/Ferretti:Luca), [Michele Colajanni](http://dblp.uni-trier.de/pers/hd/c/Colajanni:Michele), [Mirco Marchetti](http://dblp.uni-trier.de/pers/hd/m/Marchetti:Mirco):  
   **Guaranteeing Correctness of Bulk Operations in Outsourced Databases**. Proceedings of DBsec 2016: 37-51.

1. [Hakan Hacigümüs](http://dblp.uni-trier.de/pers/hd/h/Hacig=uuml=m=uuml=s:Hakan), [Sharad Mehrotra](http://dblp.uni-trier.de/pers/hd/m/Mehrotra:Sharad), [Balakrishna R. Iyer](http://dblp.uni-trier.de/pers/hd/i/Iyer:Balakrishna_R=):  
   **Providing Database as a Service**. Proceedings of [ICDE 2002](http://dblp.uni-trier.de/db/conf/icde/icde2002.html#HacigumusMI02): 29-38.

1. [Mihir Bellare](http://dblp.uni-trier.de/pers/hd/b/Bellare:Mihir), [Juan A. Garay](http://dblp.uni-trier.de/pers/hd/g/Garay:Juan_A=), [Tal Rabin](http://dblp.uni-trier.de/pers/hd/r/Rabin:Tal):  
   **Fast Batch Verification for Modular Exponentiation and Digital Signatures.** Proceedings of [EUROCRYPT 1998](http://dblp.uni-trier.de/db/conf/eurocrypt/eurocrypt98.html#BellareGR98): 236-250.
2. Einar Mykletun, [Maithili Narasimha](http://dblp.uni-trier.de/pers/hd/n/Narasimha:Maithili), [Gene Tsudik](http://dblp.uni-trier.de/pers/hd/t/Tsudik:Gene):  
   **Providing Authentication and Integrity in Outsourced Databases using Merkle Hash Trees.** UCI-SCONCE Technical Report, 2003.
3. Dan Boneh, [Ben Lynn](http://dblp.uni-trier.de/pers/hd/l/Lynn:Ben), [Hovav Shacham](http://dblp.uni-trier.de/pers/hd/s/Shacham:Hovav):  
   **Short Signatures from the Weil Pairing**.Proceedings of [ASIACRYPT 2001](http://dblp.uni-trier.de/db/conf/asiacrypt/asiacrypt2001.html#BonehLS01): 514-532

1. [Grisha Weintraub](http://dblp.uni-trier.de/pers/hd/w/Weintraub:Grisha), [Ehud Gudes](http://dblp.uni-trier.de/pers/hd/g/Gudes:Ehud): **Crowdsourced Data Integrity Verification for Key-Value Stores in the Cloud.** Proceedings of [CCGrid 2017](http://dblp.uni-trier.de/db/conf/ccgrid/ccgrid2017.html#WeintraubG17): 498-503.

# Abstract

**Outsourcing -** Is a modern management method, in which an organization contracts-out a part of their existing internal activity to an external organization, leaving only their main activities in house. This enables the organization to concentrate on their core tasks, rather than incurring substantial hardware, software and personnel costs involved in maintaining other activities in house.

Database outsourcing is a recent and important example for outsourcing such an activity.

Today, may organizations outsource their databases by using Database-as-a-service in the cloud, i.e. using the popular Microsoft’s Azure SQL Database.

Beside the advantages mentioned above, outsourcing databases triggers some important security concerns to the stored data and to the data transmitted to the customers and back. On the one hand the transmissions are exposed to external attacks, on the other hand there is a risk that the stored data or transmitted data will be changed or deleted by the service provider itself. External attacks can be handled by establishing secure channels using standard tools like SSL, these tools are not part of the scope of this paper. This paper reviews several techniques to ensure that the service provider cannot change the stored or transmitted data without being discovered.

These techniques are presented by detailed reviewing relevant articles and by implementing one of the algorithms presented in this paper.

Table of Contents

[List of figures 3](#_Toc491496011)

[List of tables 4](#_Toc491496012)

[Abstract (hebrew) 5](#_Toc491496013)

[Introduction 6](#_Toc491496014)

[1 Detailed review 7](#_Toc491496016)

[1.1 Signature-based methods -Authentication and Integrity in Outsourced Databases[1] 7](#_Toc491496017)

[1.1.1 Introduction 7](#_Toc491496018)

[1.1.2 System model 7](#_Toc491496019)

[1.1.3 Assumptions 8](#_Toc491496020)

[1.1.4 Digital signatures – overhead factors and desired features 9](#_Toc491496021)

[1.1.5 Suitable signature schemes 10](#_Toc491496022)

[1.1.5.1 Condensed RSA 10](#_Toc491496023)

[1.1.5.1.1 Standard RSA 10](#_Toc491496024)

[1.1.5.1.2 Condensed-RSA 11](#_Toc491496025)

[1.1.5.1.3 Batch veriﬁcation of RSA signatures 11](#_Toc491496026)

[1.1.5.1.4 Security of Condensed-RSA 12](#_Toc491496027)

[1.1.5.1.5 Outline 12](#_Toc491496028)

[1.1.5.1.6 Details 12](#_Toc491496029)

[1.1.5.1.7 Claim 1 13](#_Toc491496030)

[1.1.5.1.8 Claim 2 13](#_Toc491496031)

[1.1.5.1.9 Comparing overhead costs 13](#_Toc491496032)

[1.1.5.1.10 Condensed-RSA in outsourced DB setting 13](#_Toc491496033)

[1.1.5.2 BGLS 14](#_Toc491496034)

[1.1.5.2.1 BGLS scheme 14](#_Toc491496035)

[1.1.5.2.2 BGLS signatures aggregation 15](#_Toc491496036)

[1.1.5.2.3 BGLS performance 15](#_Toc491496037)

[1.1.5.2.4 BGLS in outsourced DB setting 15](#_Toc491496038)

[1.1.5.3 Batch veriﬁcation of DSA signatures 15](#_Toc491496039)

[1.1.6 Comparing overhead costs 16](#_Toc491496040)

[1.1.7 Summary and future work (for this article) 18](#_Toc491496041)

[1.2 ‏Tree-Based Methods - Merkle Hash Tree based Techniques for Data Integrity of Outsourced Data‏[6] .](#_Toc491496042).. 19

[1.2.1 Introduction 19](#_Toc491496043)

[1.2.2 Basic technique 19](#_Toc491496044)

[1.2.3 Merkle hash trees 20](#_Toc491496049)

[1.2.3.1 Merkle’s signature scheme 20](#_Toc491496050)

[1.2.3.2 B+ trees 21](#_Toc491496051)

[1.2.3.3 Data integrity based on Merkle hash tree 21](#_Toc491496052)

[1.2.4 Radix Path Identifier 22](#_Toc491496053)

[1.2.4.1 MHT storage in the database 24](#_Toc491496054)

[1.2.4.1.1 Single Authentication Table (SAT) 24](#_Toc491496055)

[1.2.4.1.2 Level Based Authentication Table (LBAT) 25](#_Toc491496056)

[1.2.4.1.3 Performanc ecomparison between both schemes 26](#_Toc491496057)

[1.2.4.2 Authentication data extraction 26](#_Toc491496058)

[1.2.4.2.1 Multi-join 26](#_Toc491496059)

[1.2.4.2.2 Single-join 27](#_Toc491496060)

[1.2.4.2.3 Zero-join 28](#_Toc491496061)

[1.2.4.2.4 Range - Condition 29](#_Toc491496062)

[1.2.4.3 Data Operations 30](#_Toc491496063)

[1.2.4.3.1 Select 30](#_Toc491496064)

[1.2.4.3.2 Update 30](#_Toc491496065)

[1.2.4.3.3 Batch update and optimization 30](#_Toc491496066)

[1.2.4.3.4 Insert and Delete 31](#_Toc491496067)

[1.2.5 Summary and future work (for this article) 31](#_Toc491496068)

[1.2.5.1 Summary 31](#_Toc491496069)

[1.2.5.2 Future work 32](#_Toc491496070)

[2 Short review 33](#_Toc491496071)

[2.1 Integrity for Join Queries in the Cloud [‎16] 33](#_Toc491496072)

[2.2 Guaranteeing Correctness of Bulk Operations in Outsourced Databases ‎[18] 34](#_Toc491496073)

[2.3 Providing Database as a Service ‎]19] 36](#_Toc491496074)

[2.4 Crowdsourced Data Integrity Verification for Key-Value Stores in the Cloud ‎23]] 38](#_Toc491496074)

[3 Implementation 39](#_Toc491496075)

[3.1 Limitations and assumptions 39](#_Toc491496076)

[3.2 Executing the program 40](#_Toc491496077)

[Summary 48](#_Toc491496078)

[Bibliography 49](#_Toc491496079)

[Abstract (English) 51](#_Toc491496013)

**The Open University of Israel**

**Department of Mathematics and Computer Science**



**Authentication and Integrity in Outsourced Databases**

Final Paper submitted as partial fulfillment of the requirements

towards an M.Sc. degree in Computer Science

The Open University of Israel

By

**Slupsky Eran**

Prepared under the supervision of Prof. Ehud Gudes

July 2017