האוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב
החטיבה למדעי המחשב

פרויקט מתקדם במדעי המחשב

מימוש (גנרי/אוטומטי) והדגמה של האלגוריתם הדינמי ליצירת "סימן מים" בתוכנה (CT)

מנחה: פרופ' אהוד גודס

מגיש:
יוחאי ליברידר
ת.ז. 031884380

סמסטר ב' 2016

דוח פרויקט זה הוגש כחלק מהדרישות לקבלת תואר "מוסמך (M.Sc.) במדעי המחשב" באוניברסיטה הפתוחה

ינואר 2017

תוכן עניינים

1. **תקציר** 4
2. **מטרת הפרויקט** 4
3. **מבוא** 5
   3.1 הטמעת סימן מים בתוכנה 6
   3.2 CT Algorithm 7
4. **מימוש הפרויקט** 15
   4.1 סביבת הפיתוח 16
   4.2 CTWM ארכיטקטורה 16
   4.3 CTWM ממשק משתמש 27
5. **תוצאות, דוגמאות ומסקנות** 34
   5.1 דוגמאות הרצה 34
   5.2. תוצאות ומסקנות 44
6. **סיכום** 45
7. **רשימת מקורות** 46

רשימת האיורים

איור 1 – אלגוריתם CT 8
איור 2 – גרפי קידוד לסימן המים..... 10
איור 3 – דוגמא – מחלקה לפני ואחרי אלגוריתם CT בסיסי..... 11
איור 4 – התאמת הבנאי של מחלקת הבסיס ל-CT..... 12
איור 5 – אלגוריתם נאיבי לחילוץ סימן המים ב-CT 13
איור 6 – אלגוריתם נאיבי לחילוץ טביעת אצבע המיוצג על ידי גרף מזיכרון הערימה 13
איור 7 - תרשימי המחלקות - Auxilary, MarkPoint 18
איור 8 - תרשים המחלקות - Graph..... 18
איור 9 - תרשים המחלקות - CTWM..... 21
איור 10 – תרשים הרצף – שלב יצירת סימן המים..... 22
איור 11 – תרשים הרצף – שלב הסימון..... 23
איור 12 – תרשים הרצף – שלב המעקב..... 23
איור 13 – תרשים הרצף – שלב שיבוץ הגרף המייצג את סימן המים..... 24
איור 14 – תרשים הרצף – שלב חילוץ הגרף המייצג את סימן המים 25
איור 15 – Calculator - ממשק המשתמש 26
איור 16 – Calculator - תרשים מחלקות..... 26
איור 17 - ממשק המשתמש CTWM 27
איור 18 – ממשק משתמש Creation Tab..... 29
איור 19 – ממשק משתמש Marking Tab..... 29
איור 20 – ממשק משתמש Tracing Tab..... 30
איור 21 – ממשק משתמש Embedding Tab..... 31
איור 22 – ממשק משתמש Extraction Tab..... 32
איור 23 – ממשק משתמש Outputs 33
איור 24 – Creation - פרמוטציה וגרף מייצג עבור סימן המים "2233445566" 34
איור 25 – Creation – קידוד בסיס וגרף מייצג עבור סימן המים "2233445566"..... 35
איור 26 – Marking – סימון התוכנית Calculator2 (1)..... 35
איור 27 – Marking – סימון התוכנית Calculator2 (2)..... 36
איור 28 – Tracing – ביצוע מעקב עם קוד סודי על התוכנית Calculator (1) 36
איור 29 – Tracing – ביצוע מעקב עם קוד סודי על התוכנית Calculator (2) 37
איור 30 – Tracing – ביצוע מעקב עם קוד סודי על התוכנית Calculator (3) 38
איור 31 – Embedding – שיבוץ גרף הפרמוטציה המייצג את סימן המים "2233445566"
בתוכנית Calculator 38
איור 32 – Extraction – חילוץ סימן המים מהתוכנית Calculator (1)..... 39
איור 33 – Extraction – חילוץ סימן המים מהתוכנית Calculator (2)..... 39
איור 34 – חילוץ מוצלח של סימן המים מהתוכנית Calculator, לאחר הזנת הקלט הסודי (3)40
איור 35 – Extraction - סיום לא מוצלח של חילוץ סימן המים מהתוכנית Calculator לאחר
הזנת קלט שאינו הקלט הסודי (4) 41
איור 36 – Extraction - ניסיון ביצוע חילוץ (כושל) על תוכנית Calculator שלא הוטמע בה סימן
מים, ברור שאינה מניבה תוצאות ללא תלות בקלט (5) 42
איור 37 – ILDASM – ניתוח התכנית Calculator שהוטמע בה סימן מים באמצעות ILDASM
..... 43

1. תקציר

הטבעת "סימן מים" בתוכנה לצורך אימות זיהוי היא מעין "טביעת אצבע" המשובצת בצורה נסתרת בקוד, ולמעשה משמשת כרישיון שימוש בתוכנה. רישיון זה ניתן לשלוף כאשר נדרש ולטעון על אותנטיות (או חוסר אותנטיות) של התוכנה.

בפרויקט זה ממומש ומודגם בצורה מלאה האלגוריתם של Collberg–Thomborson (CT) המתואר היטב ב-[1]. האלגוריתם הינו אלגוריתם מלא ל-Watermarking Dynamic Software הממומש באמצעות Dynamic Graph Watermarking (DGW). האלגוריתם מומש תוך בניית תשתית גנרית (ספריות ותכנית) להטמעתו בצורה אוטומטית בתוכניות מארחות.

בסעיף 3 ניתן רקע בנושא סימן מים בתוכנה ומתואר אלגוריתם CT לפרטיו. בסעיף 4.1 מתוארות סביבת העבודה. סעיף 4.2 מתאר את הארכיטקטורה, אלגוריתמי עזר, הספריות שנבחרו למימוש הפרויקט, הגדרת המחלקות, וגרפי הרצף. סעיף 4.3 מתאר את ממשק המשתמש. סעיף 5.1 מתאר ביצוע בפועל של פעולת הטמעה וחילוץ של סימן מים מתוכנית מארחת וסעיף 5.2 דן בתוצאות ומסיק מסקנות. סעיף 6 הוא הסעיף המסכם את הפרויקט.

2. מטרת הפרויקט

מטרת הפרויקט היא לממש ולהדגים בצורה מלאה את האלגוריתם של Collberg–Thomborson (CT) המתואר היטב ב-[1] ואשר הינו האלגוריתם המרכזי שנסקר בעבודת הגמר שבצעתי במסגרת התואר השני.

המימוש יצר כלי שימושי ליוצרי תוכנות בסביבת Microsoft .Net המאפשר להם להטמיע סימן מים (שנבחר על ידם), בתוכנית שלהם בצורה אוטומטית מלאה וללא כל התערבות ידנית בקוד שיצרו על סמך קלט (סודי) שיבחרו. כמו כן, מאפשר הכלי לחלץ את סימן המים מתוכניות שבהם הוא הוטמע, גם כן בצורה אוטומטית.

3. מבוא

בעולם האינטרנט של ימינו ישנה חשיבות הולכת וגדלה לטיפול בנושא גניבת תוכנה. הקלות שבהורדת תוכנות לא מורשות מהרשת מאפשרת ליותר ויותר אנשים לעשות בהם שימוש (במודע או שלא במודע). הסכום המוערך של שווי תוכנות אלה עומד על 52 מיליארד דולר בשנת 2016 [4]. סקר של ה-BSA מ-2016 [4] מראה בנוסף גם על קשר הדוק בין תוכנה שאינה מורשית לתוכנות זדוניות ולמתקפות סייבר.

אחד האמצעים המוצעים להקטנת תופעה זו הינו באמצעות מנגנוני "הוכחת בעלות" (Ownership Authentication). תחום זה נחקר רבות ומאגד בתוכו שיטות ואלגוריתמים רבים, אחת השיטות המוצעות הינה הטבעת "סימן מים" בתוכנה (Watermark embedding).

המונח "סימן מים" מוכר לנו כטכניקה המיושמת בשטרות כסף, ניירות ערך ובולים ואשר שימש ומשמש כאמצעי לזיהוי זיופים על ידי יצירת סימון שאינו נראה ממבט ראשון ויצירתו מצריכה טיפול בנייר לפני הדפסתו.

הטבעת "סימן מים" בתוכנה מנסה ליישם טכניקה זו לשם אותה המטרה בעולם התוכנה. התחום אליו משתייכות טכניקות אלו בתוכנה הוא תחום הסטנוגרפיה (steganography) תחום זה עוסק בטכניקות של החבאת מסר סודי בתוך מסר מארח כאשר המטרה היא לאפשר לשני צדדים לתקשר בחשאיות ומבלי לעורר חשד מצד מצוטטים פוטנציאלים. הפרסום הראשון בנושא הטבעת "סימן מים" בתוכנה הופיע לראשונה ב-1996 במאמרם של [7] Davidson and Myhrvold (ואף נרשם כפטנט). בשנת 1999, Collberg ואחרים הציעו לראשונה אלגוריתם מפורט. מאז פורסמו מספר רב של מאמרים ואלגוריתמים בנושא, ופותחו מספר כלים מסחריים ולמטרות מחקר.

3.1. הטמעת סימן מים בתוכנה

ניתן לתאר את בעיית ה-Software Watermarking בצורה הבאה:

- P – אוסף התוכניות המיועדות לסימון.
- W – אוסף הסימונים.

א. Software Watermarking משבץ w בתוכנית p השייכת ל-P כך שנוצרת תוכנית 'p.

ב. ניתן בצורה אמינה לאתר ולחלץ את w מ-'p גם אם 'p הייתה חשופה לשינויים בקוד כמו תרגום (Translation), מיטוב (Optimization) וההאפלה (Obfuscation).

ג. w – ניתנת להסתרה (Stealth).

ד. ל – w יש תכולת נתונים גבוהה (High Data Rate).

ה. הטמעת w ב-p אינה פוגעת בביצועים של p.

ו. ל – w יש תכונה מתמטית המאפשרת לטעון שנוכחותה ב-'p אינה מקרית אלא מכוונת.

ובשפה תיאורית:

א. לפני שאליס מוכרת את התוכנית שלה p, היא משבצת בה מזהה ייחודי עבור בוב, כך שנוצרת תוכנית חדשה $P_{id=Bob}$:

$$P \xrightarrow{Bob,\ key} P_{id=Bob}$$

ב. כאשר אליס מאתרת עותק של p החשוד כמועתק היא מחלצת את הזהות של הרוכש המקורי מהתוכנית החשודה ויכולה להשתמש בזה כטיעון משפטי נגדו.

$$P_{id=Bob} \xrightarrow{key} Bob$$

ג. במשחק הזה מניחים שבוב יודע את אלגוריתם השיבוץ והחילוץ שבשימוש על ידי אליס אך אינו יודע את סימן המים וגם לא את המפתח הסודי. המטרה של בוב היא לייצר עותק $p'_{id=Bob}$ מ-$p_{id=Bob}$ מבלי לפגוע בישימותה של התכנית כך שאליס לא תוכל לחלץ ממנה את המזהה הייחודי שלו.

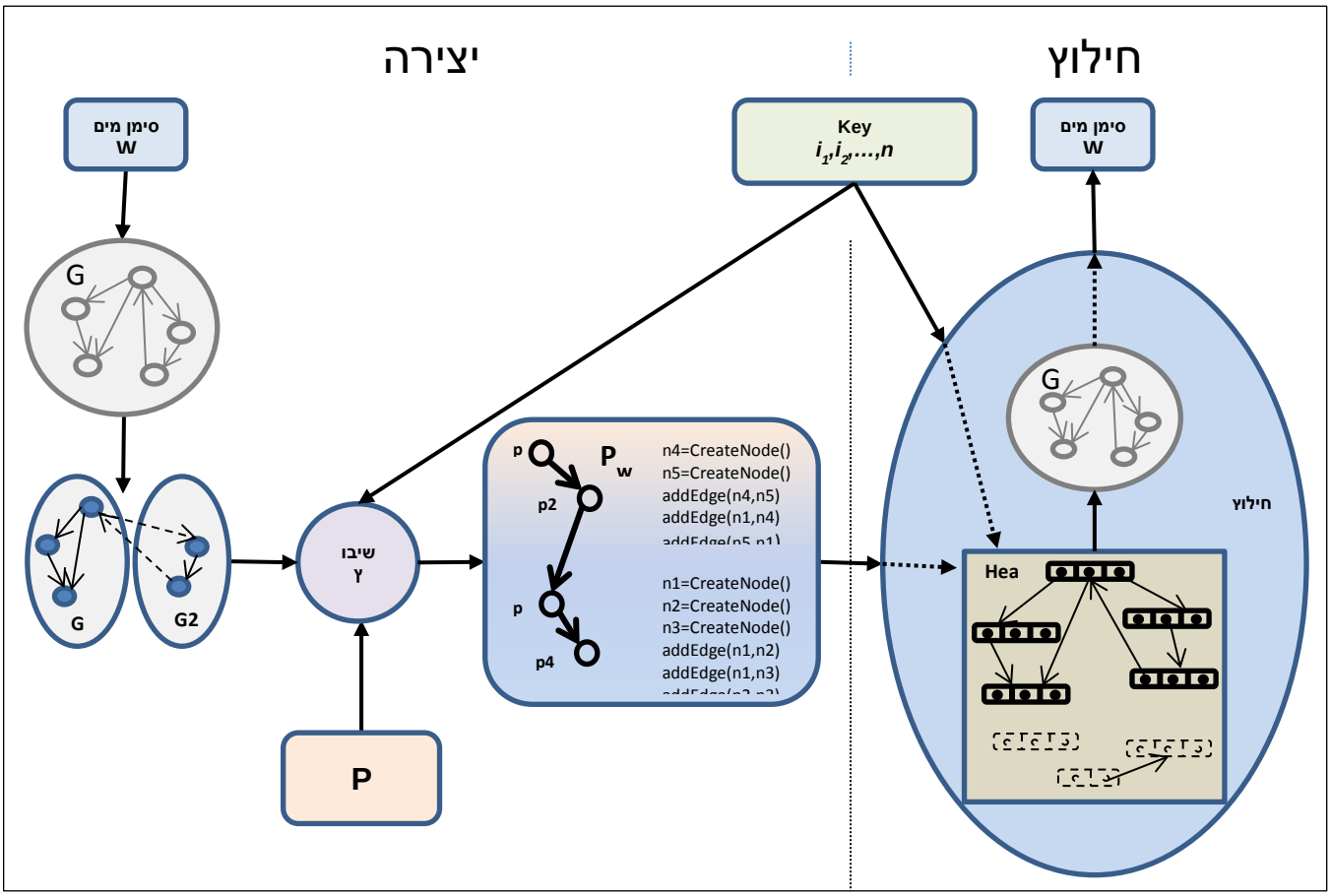
$$P_{id=Bob} \longrightarrow P'_{id=Bob} \overset{key}{\not\longrightarrow} Bob$$

3.2. CT Algorithm

בשנת 2007 פרסמו [1] Collberg and Thomborson מאמר בו הם מרחיבים ומשכללים את האלגוריתם שלהם, הנקרא על שמם CT Algorithm. האלגוריתם בונה בזמן ריצה מבנה נתונים דינמי (גרף) המייצג את טביעת האצבע בהתאם לקלט (סודי) שנבחר מראש. בזמן הריצה, הקלט גורם לקטעי קוד שונים לרוץ וכך נבנית החתימה בזיכרון הערימה (heap), שאותה יש לחלץ במקרה הצורך, ראוי לציין שהאלגוריתם שתיארו הותאם למימוש בשפת java.

תיאור האלגוריתם בצורה כללית:

1. סימן המים W משובץ בטופולוגיה של גרף G, האלגוריתם מונה מספר סוגים פוטנציאליים למימוש:
   - Permutation encoding – מומש בפרויקט.
   - Radix encoding – מומש בפרויקט.
   - Parent-Pointer Trees
   - Reducible Permutation Graphs (RPG)
   - Planted Plane Cubic Trees (PPTC)
2. הגרף מפוצל למספר תתי גרף G1, G2,…
3. עבור כל תת גרף נבנה קטע קוד המשובץ בתוכנית בנתיבים בהם היא עוברת בהינתן קלט סודי I0, I1,….
4. בזמן חילוץ סימן המים מריצים את התוכנית עם הקלט הסודי I0, I1,…, הגרף המייצג את סימן המים נבנה בזיכרון הערימה (heap memory) , מחלצים את הגרף וחושפים את סימן המים.



איור 1 – אלגוריתם CT

האלגוריתם עצמו מורכב מ-4 חלקים, להלן עיקרם :

1. Annotation – (Mark) סורקים את קוד התוכנית ומשבצים בקוד התוכנה קריאות לפונקציית ()mark אשר למעשה אינה עושות דבר בזמן הריצה למעט רישום ל-trace log שהתכנית אכן ביקרה בה. קריאות אלו מפזרים במקומות בהם התוכנית מקבלת החלטות הסתעפות על סמך קלט המשתמש (חלק מפונקציות ()mark אלו יוחלפו בפונקציות לבניית גרף סימן המים).
2. Tracing – לאחר שלב 1, מבצעים ריצה של התכנית עם סדרת קלט I. במהלך הריצה חלק מפונקציות ()mark יבוצעו וירשמו ב-trace.
3. Embedding – כאן מייצרים את הגרף המייצג את סימן המים (בד"כ זה יהיה מספר או מחרוזת שהומרה למספר) ומשבצים אותו בתוכנית. יש לקבל כקלט סימן מים לשיבוץ, לאחר מכן יש צורך

לייצג את המספר באמצעות גרף. על מנת לבצע זאת נעשה שימוש במספר אלגוריתמים שונים בהם ייצוג המספר על ידי פרמוטציה או קידוד בסיס. את תוצאת הפרמוטציה או קידוד הבסיס ניתן לייצג בגרף. לאחר מכן מייצרים את קטעי הקוד לבניית הגרף ומשבצים את קטעי הקוד הללו במקום פונקציות ה-()mark שהופיעו ב-trace ומייצרים תוכנית חדשה. כאן למעשה הסתיים תהליך שיבוץ סימן המים בתוכנה.

4. Extraction – על מנת לחלץ את סימן המים מריצים את התוכנית עם הקלט הסודי, כתוצאה מכך ייבנה גרף סימן המים בזיכרון הערימה (heap). לאחר שהתכנית קיבלה את כל סדרת הקלט הסודי, יש לחלץ את הגרף מזיכרון זה, ומהגרף המחולץ יש לחשוף את סימן המים, כלומר את המספר המייצג אותו.

ובפירוט:

האלגוריתם מניח קיום מפתח סודי k המשמש להטמעה ולחילוץ "סימן המים".

k הוא סדרה של קלטים $I_0, I_1, \ldots$ לתוכנית.

<u>שלב 1</u> – כתיבת הערות או סימון (mark / annotation)

לפני שניתן לשבץ את "סימן המים" בתוכנית המשתמש חייב להוסיף נקודות סימון לתוכנית. נקודות אלו הינם קריאות לפרוצדורות (פונקציות) מהצורה הבאה:

```
mark();
```

הקריאה ()mark למעשה אינה מבצעת דבר. קריאות אלו מהוות אך ורק אינדיקציות בקוד עבור מטביע "סימן המים" והם מסמנות נקודות פוטנציאליות בקוד התוכנית בהם יוטבעו חלקים מ"סימן המים". קריאות אלו עוזרות להבחין בנתיבים בקוד בהם התוכנית עוברת בהינתן סדרת קלטים על ידי המשתמש (אשר למעשה תפעיל את בניית הגרף המייצג את "סימן המים"), נתיבים אלו הם חלק תוצאתי של ריצה נורמלית של התוכנית (הלא מסומנת).

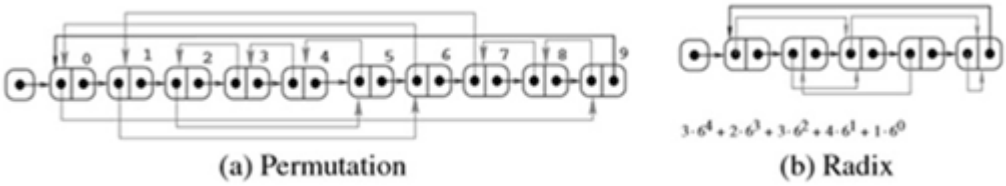
סדרת הקלטים יכולה להיות לחיצות מקלדת, תנועות ולחיצות עכבר/מסך מגע, קבצים, מחרוזות, וידאו, אודיו, נתונים ביומטריים, מסרי תקשורת וכו'.

<u>שלב 2</u> – מעקב (tracing)

לאחר שלב הסימון, מריצים את התוכנית המסומנת ל"ריצת מעקב". התוכנית רצה עם סדרת הקלטים הסודית המוזנת על ידי המשתמש. במהלך הריצה, תתבצע קריאה אחת או יותר של ()mark. חלק מנקודות "הפגיעה" הללו יהיו למעשה המקומות בקוד אשר בהם מטמיע "סימן המים" יטמיע חלקים מסימן המים.

<u>שלב 3</u> – שיבוץ (embedding)

במהלך שלב זה, המשתמש מזין את "טביעת האצבע", למעשה מחרוזת או מספר שלם (Integer). אם הזין מחרוזת, היא מומרת למספר שלם. ממספר זה מייצרים גרף מייצג, כך שטופולוגית הגרף מייצגת את המספר. עתה, קריאות ה- ()mark הרלוונטיות (מאלה שהיו נקודות "הפגיעה") יוחלפו כעת עם הקוד לבניית הגרף.



איור 2 – גרפי קידוד לסימן המים.

```java
// Simple Class
public class Simple
{
    static void P(string i)
    {
        System.Out.println("Hello " + i);
    }
    public static void main(String args[])
    {
        P(args[0]);
    }
}
```

```java
// Simple Class after WM
 public class Simple_W
{
    static void P(string i, Watermark n2)
    {
        // WM Key="World"
        if(i.Equals("World")) {
            Watermark n1 = new Watermark();
            Watermark n4 = new Watermark();
            n4.edge1 = n1;
            n1.edge1 = n2;
            Watermark n3 = (n2 != null) ? n2.edge1 : new Watermark();
            n3.edge1 = n1;
        }
        System.Out.println("Hello " + i);
    }

    public static void main(String args[])
    {
        Watermark n3 = new Watermark();
        Watermark n2 = new Watermark();
        n2.edge1 = n3;
        n2.edge2 = n3;
        P(args[0], n2);
    }
}

class Watermark extends java.lang.Object {
    public Watermark edge1, edge2;
}
```

איור 3 – דוגמא – מחלקה לפני ואחרי אלגוריתם CT בסיסי

איור 3 – דוגמא – מחלקה לפני ואחרי אלגוריתם CT בסיסי מציג דוגמא פשוטה לכיצד תיראה התוכנית לאחר הטבעת "סימן המים". התוכנית המקורית Simple שונתה לתוכנית Simple_W כך שכאשר היא תורץ עם קלט סודי שהוא הארגומנט "World" יבנה בזיכרון הערימה הגרף המייצג את "סימן המים". במימוש טיפוסי Simple_W ו-Watermark היו עוברות ערפול בכדי למנוע מתקפת זיהוי תבניות.

<u>שלב 4</u> – חילוץ (extraction)

במהלך שלב החילוץ, מריצים את התוכנית שוב עם סדרת הקלט הסודית. אותם מקומות בהם היו קריאות ()mark שבוקרו בשלב המעקב ייקראו שוב. אך עתה, מקומות אלו יכילו קוד לבניית הגרף המייצג את "טביעת האצבע". כאשר הוזן הקלט האחרון בסדרה למעשה הושלמה בניית הגרף בזיכרון הערימה. כעת נותר לבדוק את זיכרון הערימה אשר בו נמצא הגרף ולנסות לאתר אותו, לחלץ אותו ולבצע קידוד הפוך מגרף למספר אשר מדווח למשתמש.

שלב זה הוא השלב הקשה והמורכב ביותר למימוש בעיקר משום שיש צורך לחקור את זיכרון הערימה של תוכנית על ידי תוכנית אחרת ולנסות לחלץ ממנו גרף תקף, אשר ייתכן והוא הגרף המייצג את "טביעת האצבע" שהוטמעה.

המאמר שהתאים את האלגוריתם למימוש בשפת java הציע 2 דרכים לחקור את זיכרון הערימה, ואילו במימוש האלגוריתם בפרויקט זה החילוץ התבצע בדרך שלישית, אחרת (כאמור, המימוש כאן הוא בשפה ובסביבה שונה) .

במימוש שלב החילוץ המותאם לשפת java הציעו מפתחי האלגוריתם :

1. לשנות את הבנאי (Constructor) של מחלקת הבסיס של עצמי ג'אווה java.lang.Object ולהכניס בה מונה, בצורה הבאה :

```
package java.lang;
public class Object {
    public static long objCount = 0;
    public long allocTime;
    public Object() {
        allocTime = objCount++;
    }
}
```

איור 4 – התאמת הבנאי של מחלקת הבסיס ל-CT

מכיוון שבנייה של כל עצם בשפת ג'אווה מחייבת קריאה למחלקת הבסיס ולבנאי זה, המשמעות היא שהכנסנו את סדר ההקצאה של העצמים בתוכנית בזיכרון הערימה וזאת בעלות זמן של תוספת השמה אחת ופעולת חיבור אחת ובתוספת זיכרון של משתנה אחד מסוג long (8 בתים) סטטי לכל העצמים ואחד עבור כל עצם.

אך גישה זו אינה מעשית, מכיוון שהיא דורשת לשנות את ספריות הריצה הבסיסיות של שפת ג'אווה, יותר מזה סביר שחלק מהמהדרים הקיימים של השפה יבצעו אופטימיזציה ויבטלו תוספת זו (אופטימיזציה של זמן ריצה וזיכרון).

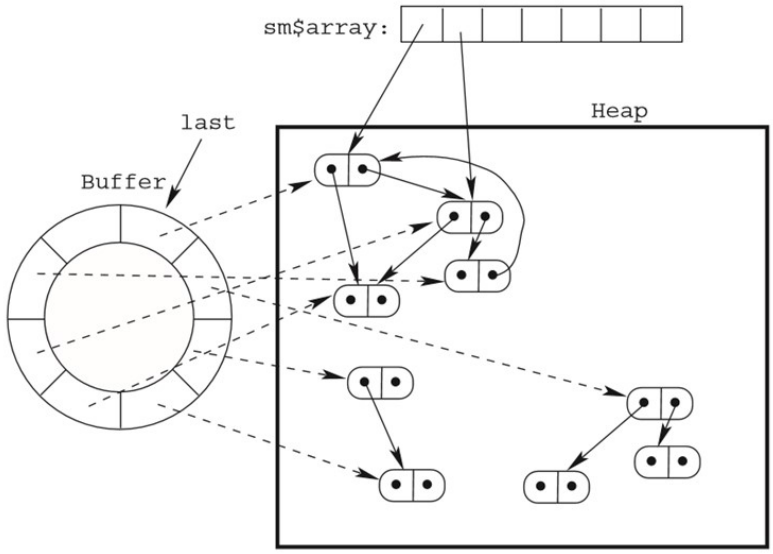
ולכן הוצעה גישה אחרת, יותר קשה למימוש אך יותר יבילה (portable) :

2. על ידי שימוש ב-JDI מוסיפים נקודת עצירה לכל בנאי בתוכנית. בכל פעם שמתרחשת הקצאה של עצם מוסיפים מצביע לחציץ מעגלי אשר מצביע לעצם זה. בצורה כזו ניתן לגשת אל 1000 (או כל

מספר אחר) העצמים האחרונים שהוקצו בזיכרון הערימה. ניתן לראות זאת ב**איור 5 – אלגוריתם נאיבי לחילוץ סימן המים ב- CT**.

החיסרון העיקרי הוא בהאטה משמעותית בביצועי התכנית בזמן החילוץ עצמו עקב הצורך לטפל בנקודות העצירה.

בכדי לחלץ את הגרף המייצג את סימן המים, יש לעבור על העצמים בסדר הפוך לסדר יצירתם ולחפש תת גרף נגיש עד שימצא אחד כזה. חילוץ טביעת האצבע מהגרף ניתן לביצוע על ידי האלגוריתם הנאיבי שמוצג ב**איור 6 – אלגוריתם נאיבי לחילוץ טביעת אצבע המיוצג על ידי גרף מזיכרון הערימה**.



איור 5 – אלגוריתם נאיבי לחילוץ סימן המים ב-CT

ב**איור 5 – אלגוריתם נאיבי לחילוץ סימן המים ב-CT** , ניתן לראות את מצב הזיכרון בזמן החילוץ. חציץ מעגלי מכיל מצביעים לכל העצמים האחרונים שנוצרו בזיכרון הערימה. המחלץ יבדוק את החציץ בסדר הפוך לסדר הקצאתם ויחפש את שורש הגרף שיכול לשמש כגרף המייצג טביעת אצבע.

```
for each graph codec C do
   for each node n on the buffer, in reverse allocation order do
      G = heap graph with root n, edges [e1,e2,...,ek];
      for each pair (ei,ej) of edges in [e1,e2,...,ek] do
         G' = subgraph of G with edges labeled (ei,ej);
         if fingerprint decoding w=C(G') succeeds then
            yield w
```

איור 6 – אלגוריתם נאיבי לחילוץ טביעת אצבע המיוצג על ידי גרף מזיכרון הערימה

3. הדרך השלישית למימוש שלב החילוץ, אשר בוצעה בפרויקט זה נעשתה תוך שימוש בספריית - CLR) Microsoft.Diagnostic.Runtime* MD) [8] , ובצורה נאיבית. רצים על כל העצמים בזיכרון הערימה, כאשר פוגשים בעצם מסוג "סימן מים" מכניסים אותו (את תוכנו וכתובתו) לרשימה. בסיום הריצה רצים על רשימת העצמים ומנסים למצוא את הקשרים ביניהם. בסיום ריצה זו (על הרשימה), נבדקים הקשרים שנמצאו ומנסים למצוא התאמה למבנה הנתונים המייצג גרף תקף (Radix/Permutation), אם נמצא כזה, מציגים את "סימן המים" שהוא מייצג (מספר שלם).

1. תכנית החילוץ מריצה את התוכנית (כתהליך, Process) ממנה יש לבצע את חילוץ סימן המים.
2. באמצעות ספריית [8] Microsoft.Diagnostic.Runtime* (CLR MD) ניגשים ל- Common Language Runtime (CLR) של התהליך.
3. מתוך ה-CLR של התהליך ניגשים לזיכרון הערימה ולסגמנטים השונים בו.
4. עבור כל סגמנט בזיכרון הערימה ניתן לקבל את רשימת העצמים שנמצאים בו.
5. עבור כל עצם ניתן לקבל מידע כמו הסוג שלו (type) מיקומו בזיכרון (כתובת) ואת הערכים שלו בזיכרון (מערך של בתים).
6. ניתן כעת לעבור על העצמים המתאימים ל- type מסוג "סימן המים" ולמצוא את קשרי המצביעים ביניהם וזאת באמצעות התאמת מצביעי הזיכרון של עצמים אלו לכתובות הזיכרון בהם הם נמצאו.
7. עתה ניתן לבדוק האם מבנה הקשרים שנמצאו בין עצמים אלו מייצג גרף תקף המייצג "סימן מים". אם נמצא כזה, ניתן להציג את "סימן המים" אותו הוא מייצג (מספר שלם).

** [8] Microsoft.Diagnostic.Runtime הנה ספרית Open Source אשר נכתבה על ידי Microsoft וניתנת תחת רישיון MIT (רישיון חופשי מתירני). הספרייה מחצינה API המאפשר כתיבה של כלים ותוספים (Plugins) לטובת ביצוע ניפוי שגיאות (debugger) במוצרים מבוססי Net..*

4. מימוש הפרויקט

הרעיון שבבסיס הפרויקט הוא לבנות יישום כללי, מעין כלי תשתיתי, כך שבקלות ובצורה אוטומטית ניתן יהיה לבנות ולהטמיע, ולאחר מכן לחלץ את סימן המים ל/מכל תכנית מארחת (כמובן שרק עבור תכניות שנבנו בטכנולוגיה שנבחרה למימוש כאן, Microsoft .Net).

הפרויקט מורכב מהחלקים הבאים:

א. ניתוח האלגוריתם ודרישותיו ועיצוב הארכיטקטורה הנדרשת למימושו.
ב. מימוש הארכיטקטורה.
ג. בנית תוכנית המדגימה את האלגוריתם והשימוש בו.

תכולות הפיתוח:

א. הפיכת מספר סודי (המתקבל כקלט) לגרף במספר שיטות שונות (מבני נתונים) Permutation / Radix (באמצעות אלגוריתם unrank1 [6]), וחלוקת הגרף למודולים לשיבוץ.
ב. ויזואליזציה של הגרפים, אפשרות לשנות את פריסתם, לשמור אותם כקובץ (xml) ולהדפיסם.
ג. שלב הסימון - מציאה ושיבוץ של נקודות mark בקוד התוכנית המארחת במקומות תקפים.
ד. שלב המעקב - מציאת נקודות בתוכנית המארחת לשיבוץ הגרף (trace) תוך כדי הרצתה, בהתאם לקלט סודי.
ה. שלב השיבוץ - שיבוץ חלקי הגרף (מודולים) בתוכנית המארחת בהתאם לתוצאות שלב המעקב ובניית התוכנית בה מוטמע סימן המים.
ו. שלב החילוץ - חילוץ הגרף מהתכנית בה מוטמע סימן המים. למעשה חילוץ האובייקטים מזיכרון הערימה (ה-heap) של תהליך אחר תוך כדי ריצתו, וחיפוש הגרף באובייקטים אלו לאחר הזנת הקלט הסודי וביצוע תיקוף של הגרף המתקבל.
ז. ויזואליזציה של הגרף המחולץ והפיכת הגרף חזרה למספר הסודי.
ח. הדגמה על תוכנית אמתית - הדגמה של שימוש בתוכנית שאפתח (סעיפים א – ז) על תוכנית אחרת (כתוכנית מארחת).

4.1. סביבת הפיתוח

אמנם אלגוריתם CT נכתב למימוש בשפת ג'אווה, אולם אני בחרתי לממשו בשפת #C.

בחרתי שלא להיעזר במימושים זמינים לאלגוריתם שפותחו בעבר (כדוגמת הכלי [3] SandMark) ולפתח בעצמי את כל הנדרש, למעט שתי ספריות צד שלישי המשמשות לצורך הצגה ויזואלית של גרף ולצורך חקירת זיכרון ערימה של תוכנית (יפורט בהמשך).

המערכת פותחה עבור מערכת ההפעלה Windows 10, אם כי היא יכולה לרוץ גם על מערכות הפעלה Windows קודמות (Win 7 / Win 8) בהתאמות קלות. כמו כן נבחר Framework 4 Client Profile עבור הפיתוח בסביבת Microsoft .Net.

סביבת הפיתוח (IDE) נבחרה להיות Visual Studio 2015 Community Version.

טכנולוגית ה-UI בה נעשה השימוש הנה Windows Forms.

4.2. ארכיטקטורה CTWM

4.2.1 כללי

הקו המנחה בבניית הארכיטקטורה לביצוע הפרויקט היה הצמדות ל-4 השלבים המפורטים באלגוריתם CT, ולכן המודולים, המחלקות, המתודות וממשק המשתמש תוכננו בהתאם. ניתן לראות זאת בעיקר בעיצוב ממשק המשתמש [4.2.2] ובתרשימי הרצף (sequence Diagrams) [4.2.3].

4.2.2 מודולים, מרחבי השמות והמחלקות

התכנית שפותחה (Collberg Thomborson Water Marking) **CTWM** מורכבת מתוכנית ראשית (executable) וממספר מודולים (Class Libraries/ dlls) חלקם בשימוש אצל היישום המטמיע שפותח (CTWM) וחלקם בשימוש אצל התוכנית המארחת את סימן המים. כמו כן, כאמור, נעשה שימוש בשתי ספריות צד שלישי.

א. שלוש הספריות הבאות נמצאות בשימוש היישום המטמיע והן בתוכנית בה הוטמע סימן המים:

1 Auxilary.dll (Class Library)

- Namespace Auxilary
  - class GeneralDefs

מחזיקה קבועים והגדרות כלליות.

- o class Auxilary

מכילה פרוצדורות כלליות לטיפול בקבצים (העתקות, כתיבות וקריאות), ומתודות כלליות להרצת תהליכים.

2 MarkPoints.dll (Class Library)

- Namespace MarkPoints
- o class MarkPoints

מטפלת בסימון נקודות ה-mark שנקראות בזמן ביצוע ה-trace על התוכנית המארחת.

3 Graphs.dll (Class Library)

- Namespace Graphs
- o class Graph

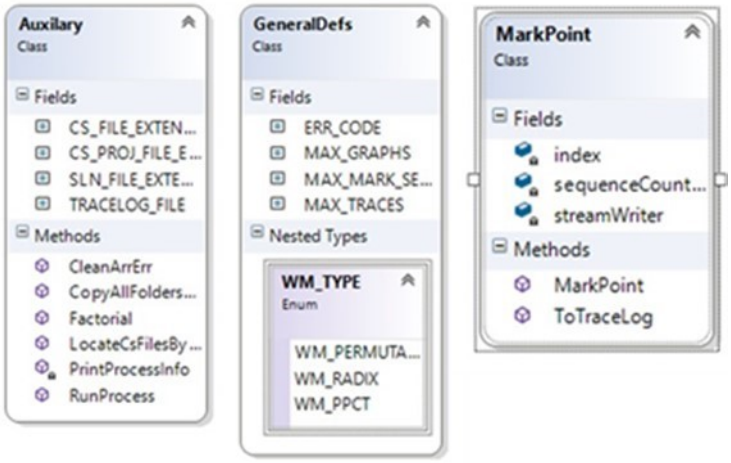
מחלקה כללית למימוש גרף.

- o class PermutationGraph

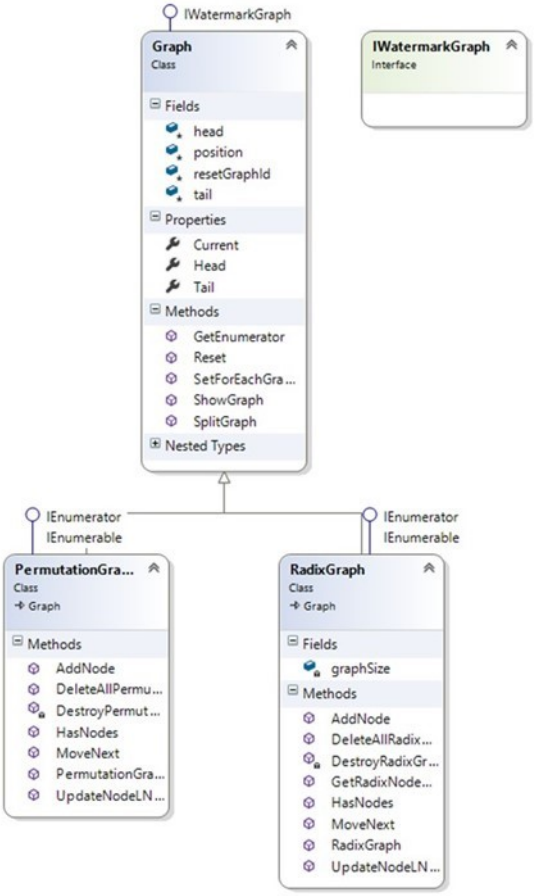
יורשת מ-Graph ומרחיבה אותו לצורך ייצוג וטיפול בגרפי פרמוטציה.

- o class RadixGraph

יורשת מ-Graph ומרחיבה אותו לצורך ייצוג וטיפול בגרפי בסיס.



איור 7 - תרשימי המחלקות - Auxilary, MarkPoint



איור 8 - תרשים המחלקות - Graph

ב. שני המודולים הבאים הם מצד שלישי (3rd Party) :

1 מודול [9] **Microsoft Automatic Graph Layout (MSAGL)** ו- **GraphViewerGDI** [10] לטובת הצגת הגרפים המייצגים את סימן המים.

2 מודול [8] **Microsoft Diagnostic Runtime** לטובת קריאת זיכרון הערימה של תוכנית מתוכנית אחרת (עושה שימוש ב-Reflection).

ג. נבנתה תוכנית ראשית בשם CTWM.exe המכילה את ממשק המשתמש הראשי ובנויה ממרחבי השמות (namespaces) והמחלקות (classes) הבאות:

- namespace CTWM.About
  - o class About : Form

טופס להצגת מספר גרסה, תאריך, ויוצר.

- namespace CTWM.AuxilaryWM
  - o class Constants

קבועי התכנית

  - o class Auxilary

מתודות עזר כלליות לתוכנית

  - o class Exceptions

טיפול בחריגות

  - o class Wizard

טיפול בתהליכי ה- Wizard

- namespace CTWM.CreateWM
  - o class Permutation

יצירת סימן מים מסוג פרמוטציה

  - o class Radix

יצירת סימן מים מסוג קידוד בסיס

- namespace CTWM.EmbedWM
  - o class EmbedWM

ביצוע שלב שיבוץ סימן המים

- namespace CTWM.ExtractWM
  - class ExtractedRawGraph : IEnumerator, IEnumerable
  - class ExtractedRawNode
  - class PeExtractedRawGraph : ExtractedRawGraph
  - class RadixExtractedRawGraph : ExtractedRawGraph

חילוץ ובניית הגרף המייצג את סימן המים

  - class ExtractWM

בניית סימן המים מהגרף שחולץ

- namespace CTWM.MarkWM
  - class MarkWM

ביצוע שלב הסימון

- namespace CTWM.Settings
  - class Settings : FORM

טופס לשינוי הגדרות כלליות.

- namespace CTWM.ShowWM
  - class ShowWM : Form

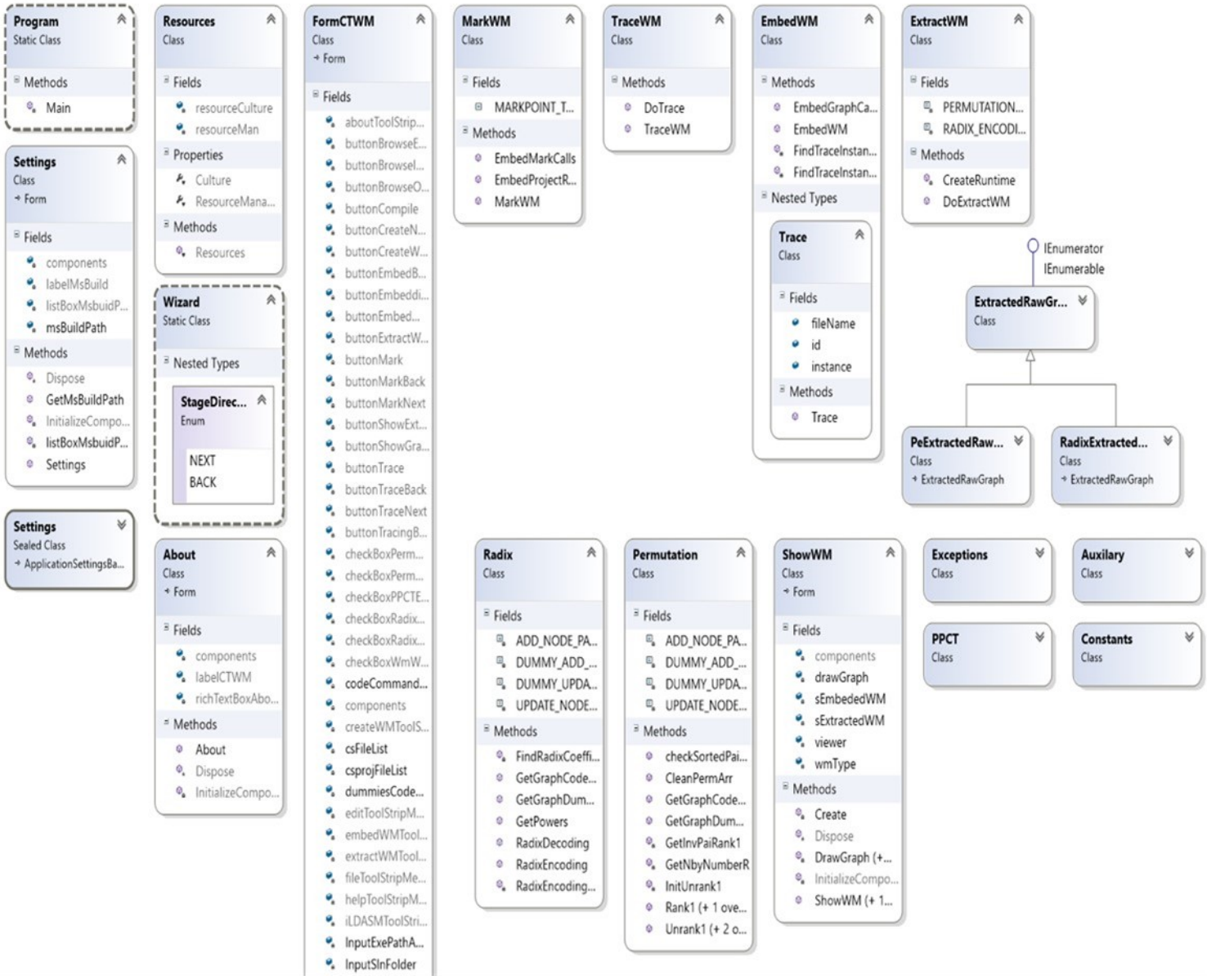
טופס להצגה ויזואלית של הגרף המייצג את סימן המים

- namespace CTWM.TraceWM
  - class TraceWM

ביצוע שלב המעקב

- namespace CTWM
  - class FormCTWM: Form
  - class Program (Main)

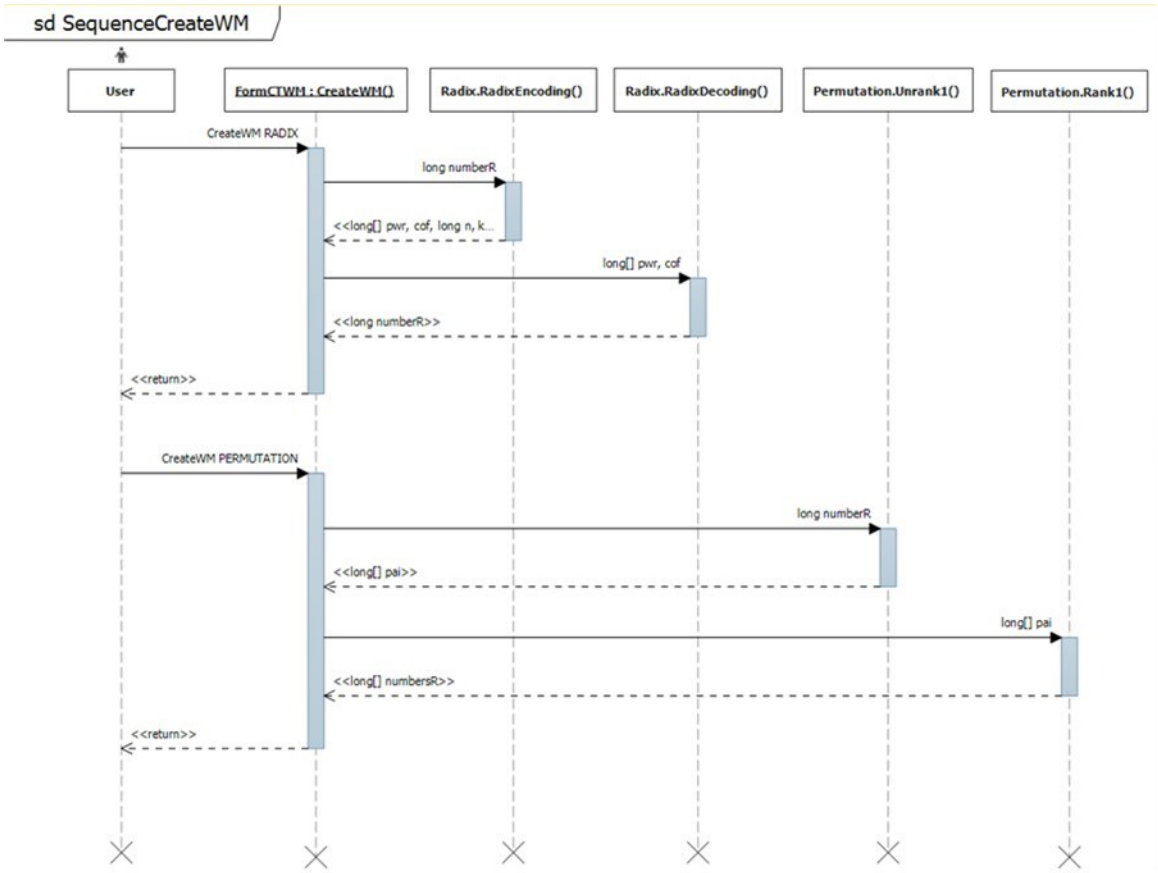
טופס התוכנית הראשי , המכיל את ממשק המשתמש.



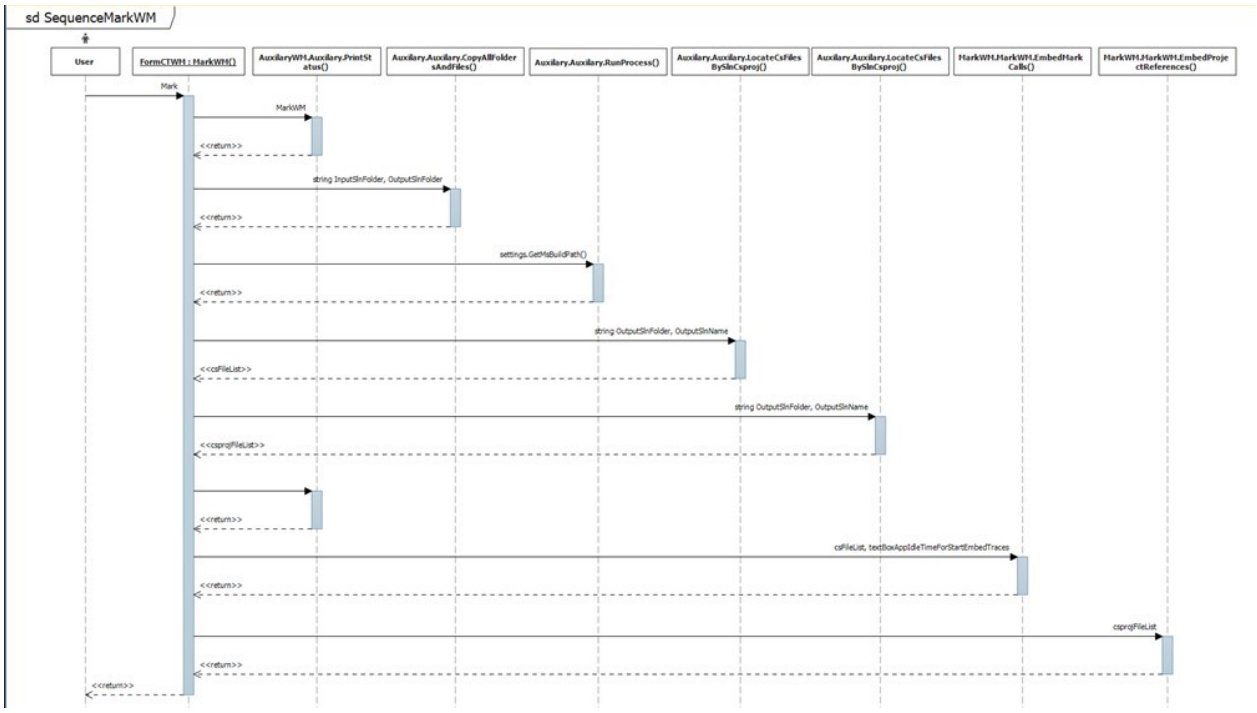
איור 9 - תרשים המחלקות - CTWM

4.2.3 תרשימי הרצף

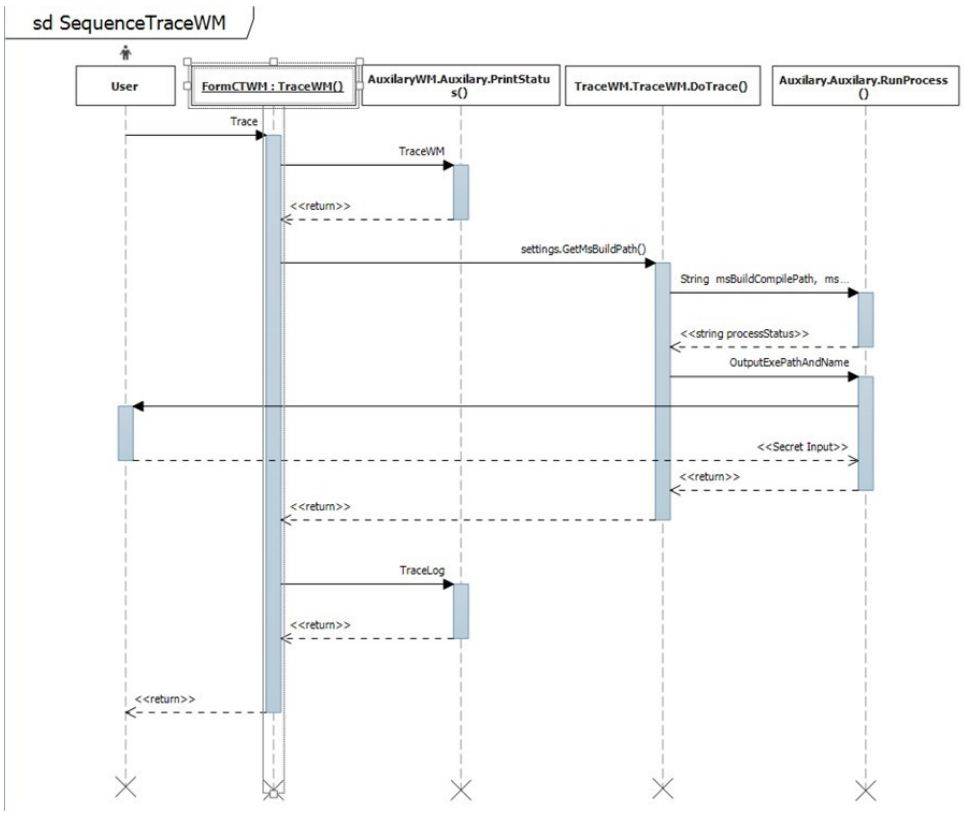
להלן תרשימי הרצף (Sequence Diagrams) עבור השלבים המרכזיים באלגוריתם CT- יצירת סימן המים, סימון התוכנית, המעקב, השיבוץ והחילוץ.



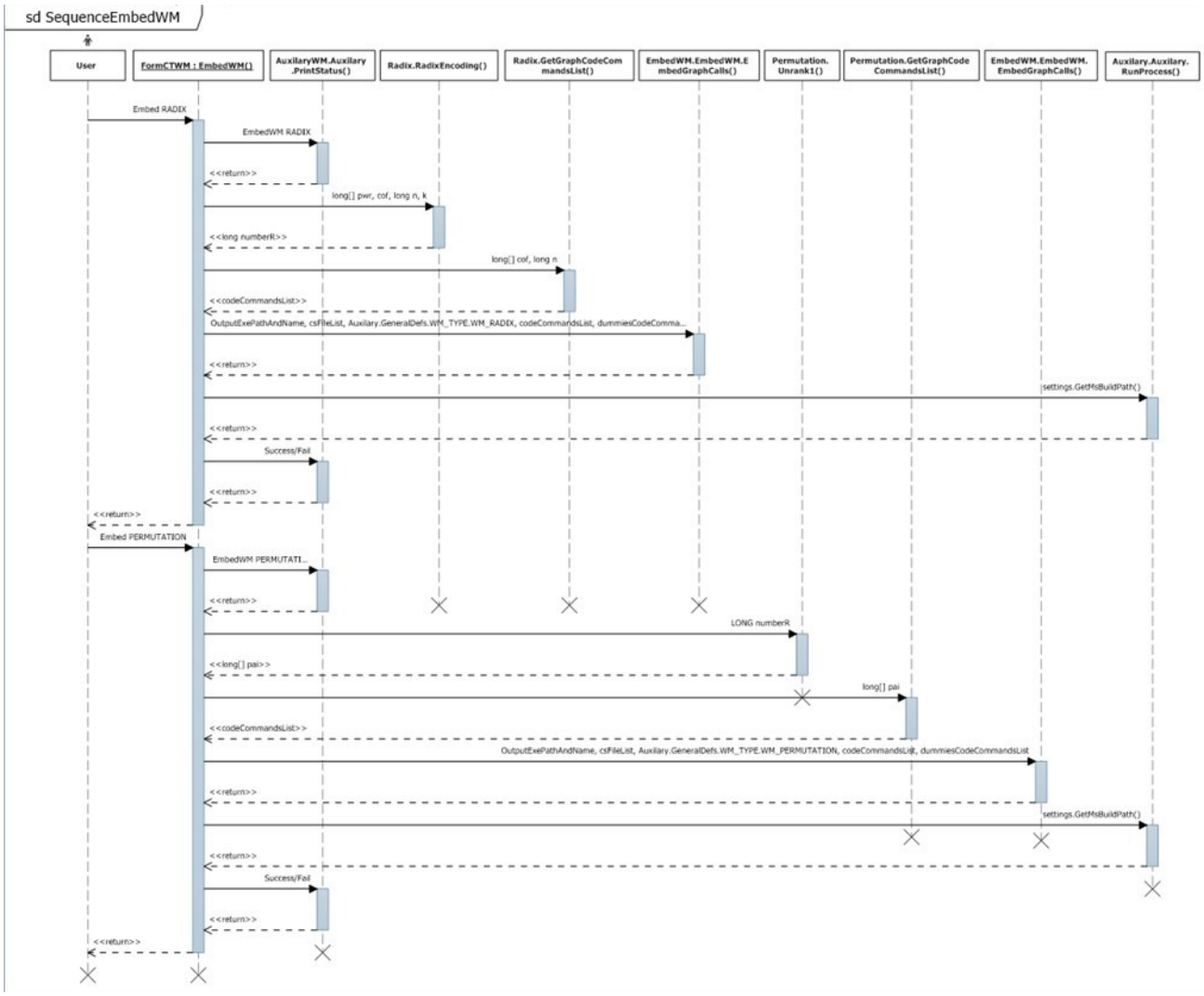
איור 10 – תרשים הרצף – שלב יצירת סימן המים



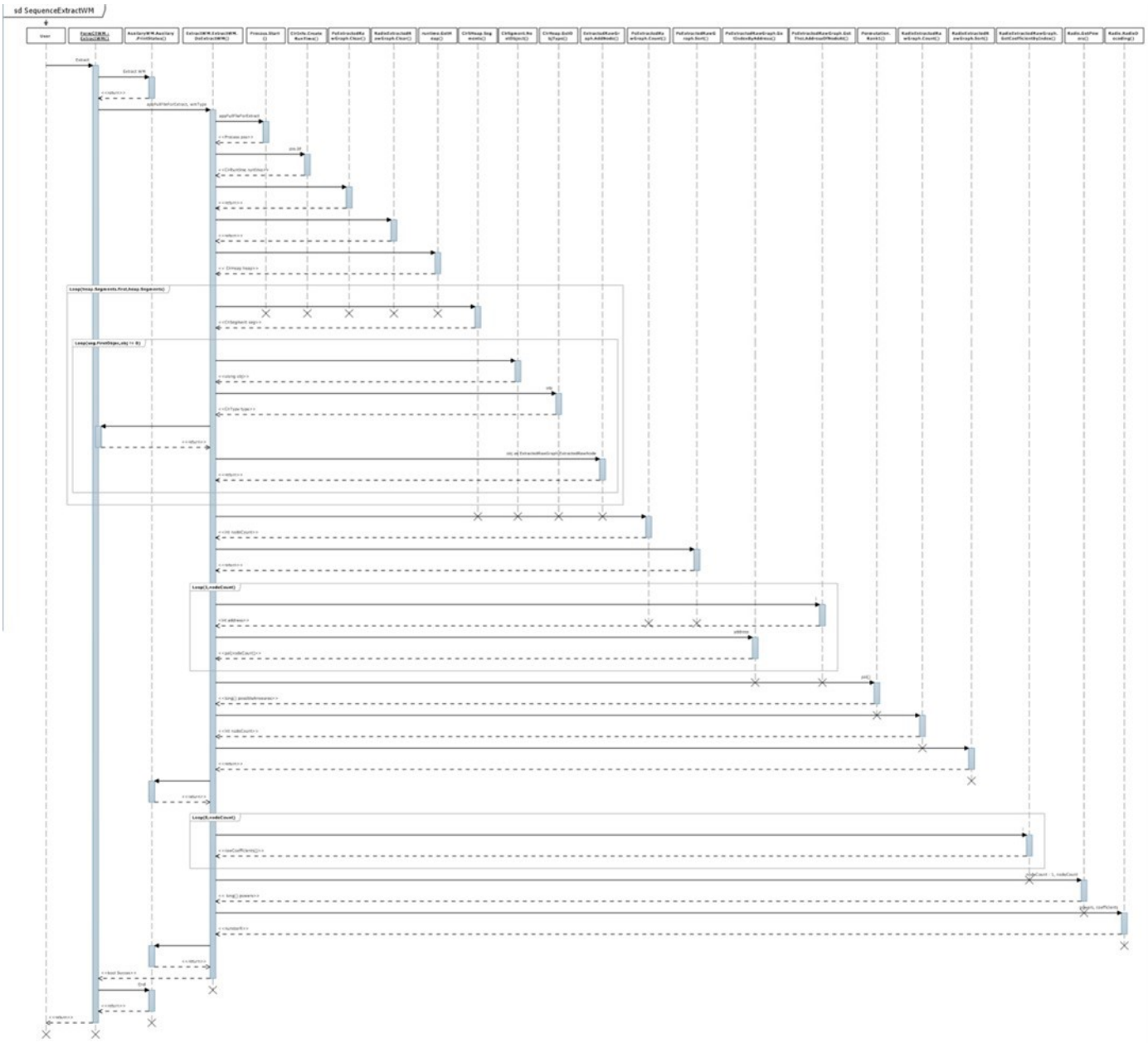
איור 11 – תרשים הרצף – שלב הסימון



איור 12 – תרשים הרצף – שלב המעקב



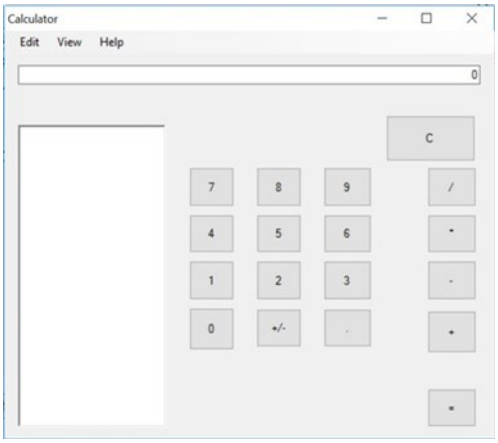
איור 13 – תרשים הרצף – שלב שיבוץ הגרף המייצג את סימן המים



איור 14 – תרשים הרצף – שלב חילוץ הגרף המייצג את סימן המים

4.2.4 תוכנית הדגמה (מארחת)

נבנתה תוכנית נוספת בשם **Calculator** לצורך הדגמת מימוש האלגוריתם. תוכנית זו מממשת מחשבון פשוט.



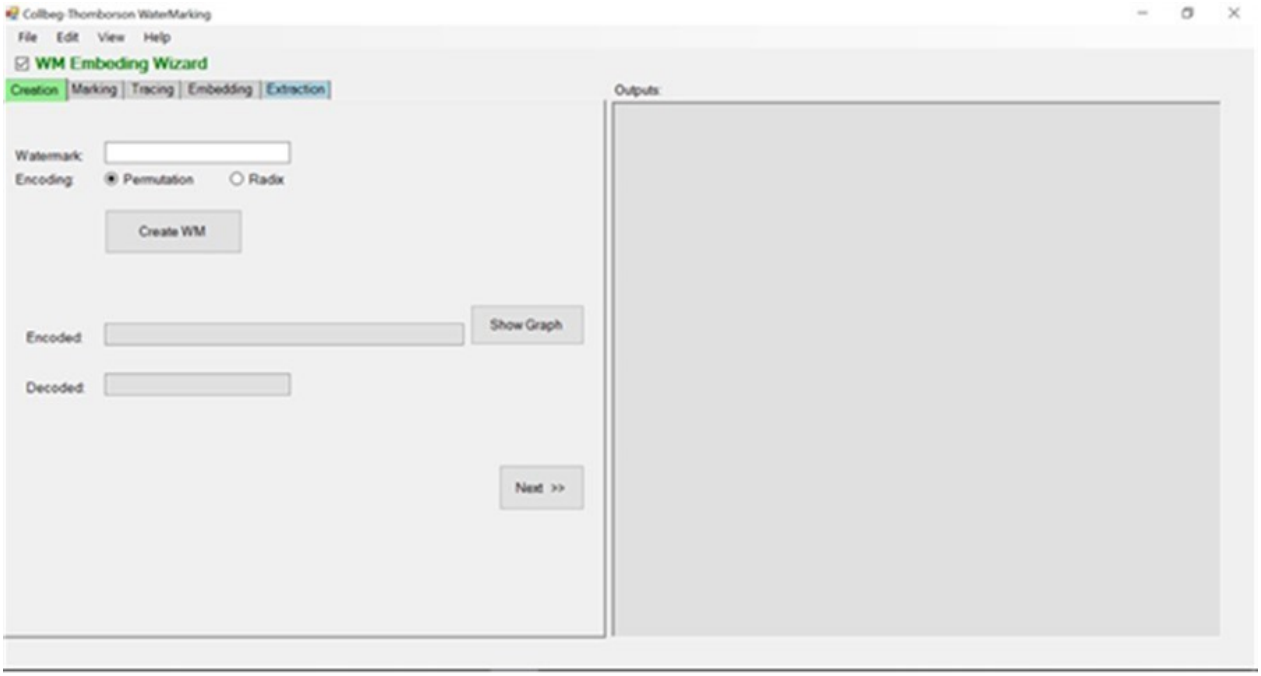
איור 15 – Calculator - ממשק המשתמש

איור 16 – Calculator - תרשים מחלקות

4.3. ממשק משתמש CTWM

ממשק המשתמש מורכב מטופס ראשי אשר מחולק למספר רכיבים:

- תפריטים (Menus)
- הפעלות (Tabs and Buttons)
- דוחות (Outputs)



איור 17 - ממשק המשתמש CTWM

4.2.2 תפריטים

1. File

- WM Wizard – הפעלה/כיבוי ה-Wizard.
- Create WM - פותח את Tab יצירת סימן המים.
- Mark – פותח את Tab סימון התוכנית.
- Trace PTS– פותח את Tab ביצוע המעקב.
- Embed WM - פותח את Tab שיבוץ סימן המים.
- Extract WM – פותח את Tab חילוץ סימן המים.

2. Edit

- Settings – פותח את טופס ההגדרות הכללי המכיל אפשרות לבחור את סוג המהדר (msbuild.exe) ותצורת ההידור (debug/release).

3. View

- ILDASM – פותח את התוכנית ILDASM.exe (Intermediate Language Disassembler) עם התוכנית עבורה בוצע שיבוץ או חילוץ סימן המים.

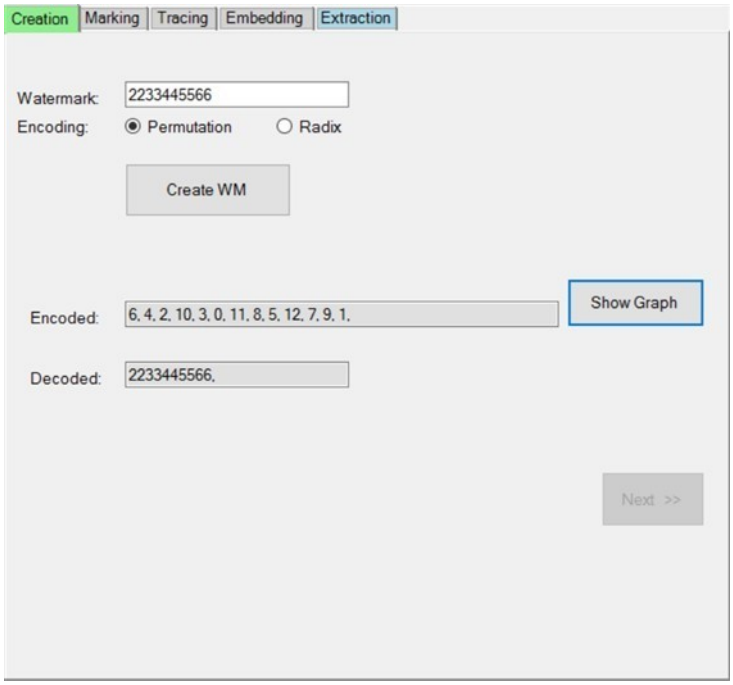
4. Help

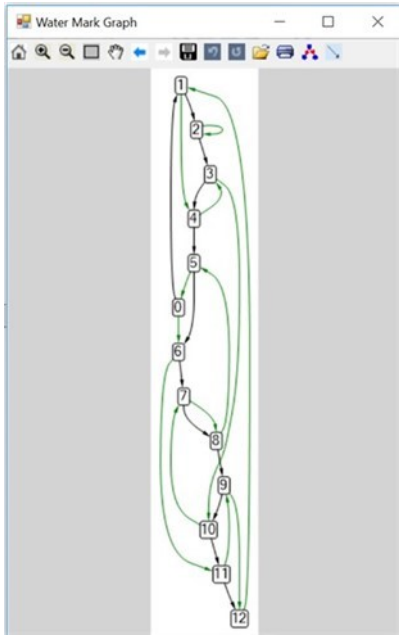
- About – פותח את טופס About המכיל מספר גרסה, תאריך, והיוצר.

4.2.2 הפעלות

רכיב ההפעלות מורכב מ:

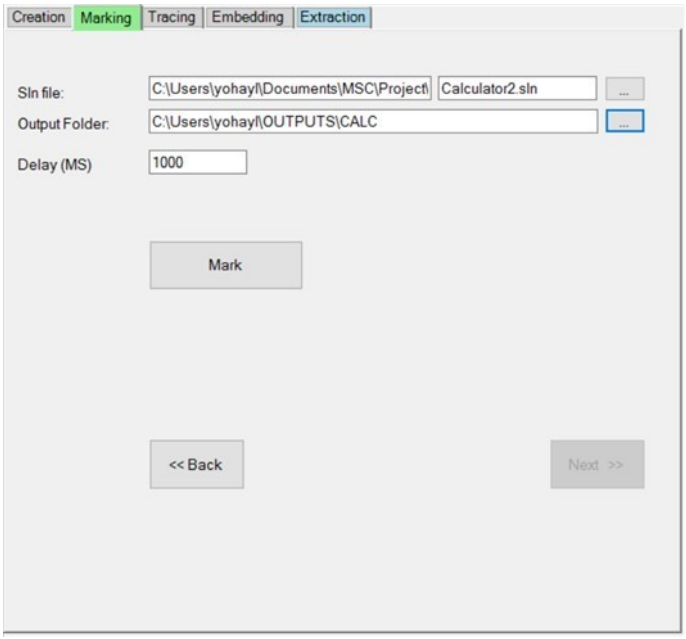
1. תיבת סימון להפעלה או כיבוי WM Embedding Wizard.
2. Creation Tab - מאפשר הזנת מספר (סימן המים) והפיכתו לגרף המייצג אותו ב-2 אלגוריתמים שונים – פרמוטציה וקידוד בסיס. כמו כן ניתן לצפות וויזואלית בגרף שנוצר, לשנות את גודלו ואת פריסתו, לשמור אותו לקובץ, להדפיסו וכו'.





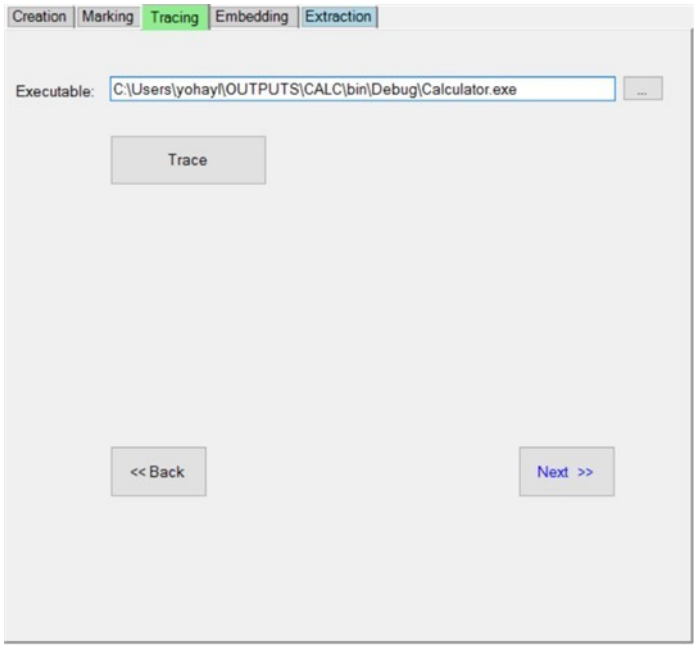
איור 18 – ממשק משתמש Creation Tab

3. Marking Tab – (Annotation) פעיל רק כאשר נמצאים במצב Wizard והמשתמש יצר סימן מים ב- Creation Tab. מאפשר לבחור תוכנית קלט אשר עבורה יבוצעו הסימונים של כל המקומות האפשריים והרלוונטיים להטמיע את חלקי הגרף המייצג את סימן המים שנבחר. ניתן לצפות ברכיב הדוחות (Outputs) מה התבצע והיכן בתוכנית בשלב זה.



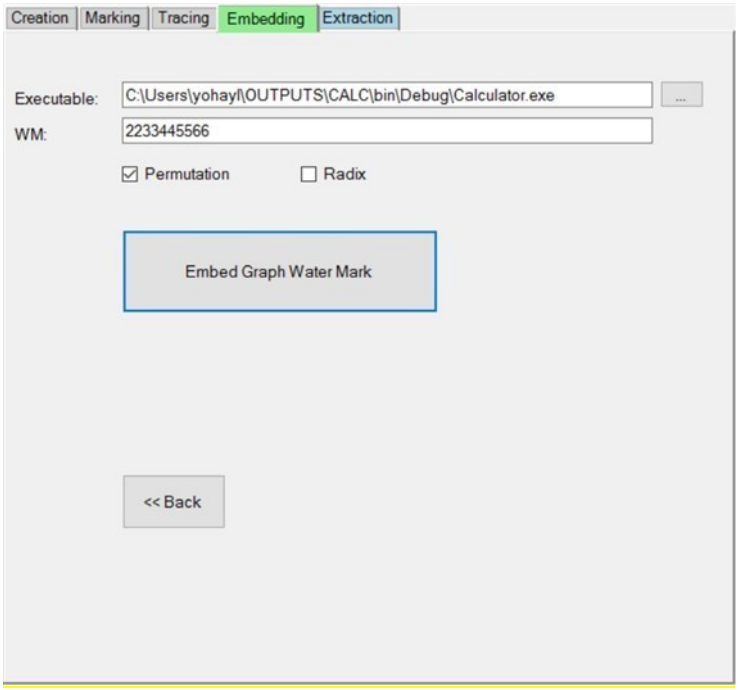
איור 19 – ממשק משתמש Marking Tab

4. Trace Tab - פעיל רק כאשר נמצאים במצב Wizard והמשתמש ביצע את שלב ה-Marking. מאפשר לבחור את התוכנית בה בוצע הסימון, מריץ אותה ומבצע Trace על סמך הקלט שקיבלה (הקוד הסודי של המשתמש). .ניתן לצפות ברכיב הדוחות (Outputs) מה התבצע והיכן בתוכנית בשלב זה.



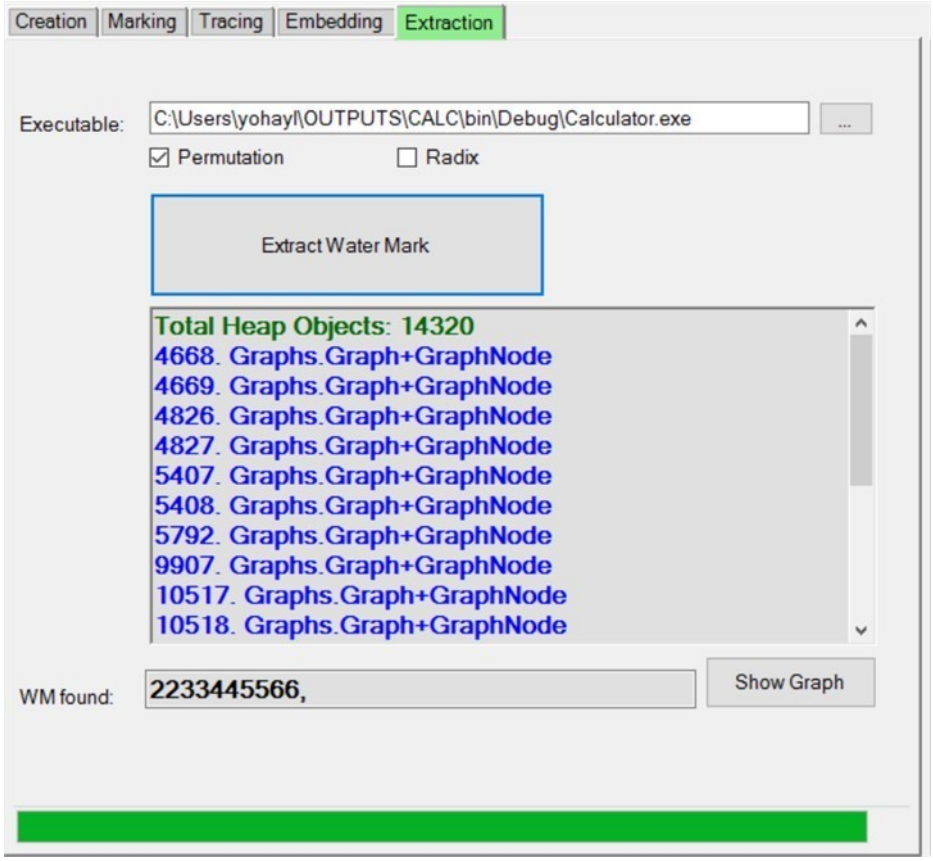
איור 20 – ממשק משתמש Tracing Tab

5. Embedding Tab - פעיל רק כאשר נמצאים במצב Wizard והמשתמש ביצע את שלב ה-Trace. מאפשר לשבץ את סימן המים (ניתן עכשיו לבחור אחד אחר) התוכנית שעברה Mark ו-Trace. בסיום שלב זה מסתיים שלב השיבוץ, ונוצרת תוכנית חדשה אשר הגרף המייצג את סימן המים שובץ בתוכה על סמך הקוד הסודי שהתקבל כקלט בתוכנית.



איור 21 – ממשק משתמש Embedding Tab

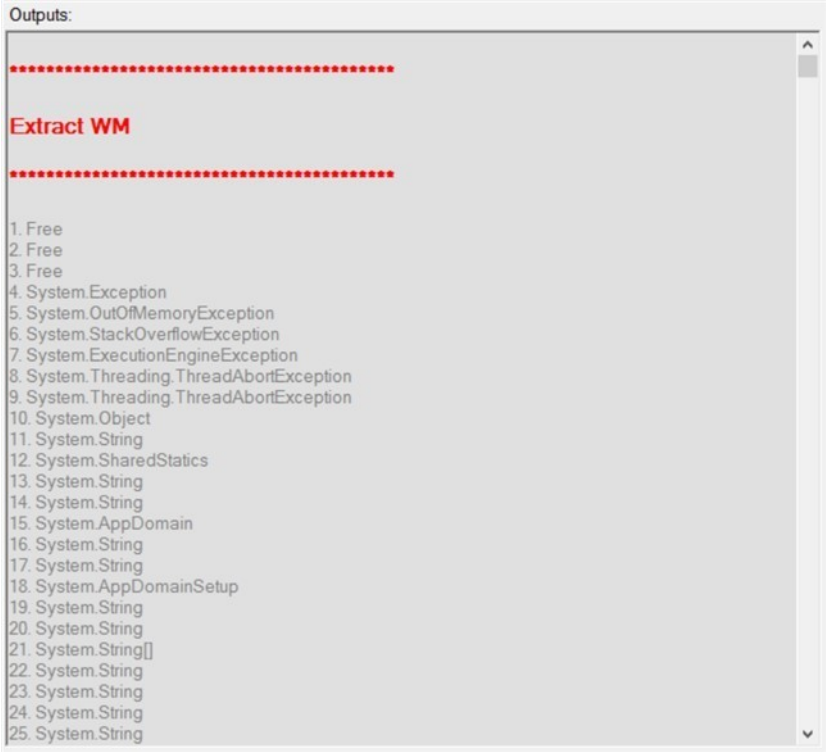
6. Extraction Tab – אינו תלוי במצב ה-Wizard. מאפשר לחלץ (ליתר דיוק - לנסות לחלץ) את הגרף המייצג את סימן המים מתוכנית ולהמירו חזרה למספר שנבחר להיות סימן המים וזאת תוך כדי ריצת התוכנית המסומנת המקבלת את הקלט הסודית. המשתמש יתבקש לבחור את התוכנית ממנה ירצה לחלץ את סימן המים, ואת סוג סימן המים שהוטמע בה (לא הכרחי), לאחר מכן בלחיצה על "Extract Water Mark" תפתח התוכנית שנבחרה והמשתמש יתבקש להזין לה את הקלט הסודי. לאחר שהוזן הקלט הסודי יודיע על כך המשתמש לתוכנית החילוץ והיא תתחיל לחקור את זיכרון הערימה של התוכנית המחולצת (שעדיין רצה וחייבת לרוץ) ותציג בר התקדמות בתחתית ה-tab. תוך כדי ריצה יודפסו כל העצמים שנמצאו בזיכרון הערימה ברכיב הדוחות, ובטופס ב-Extraction יודפסו כל העצמים שנמצאו בזיכרון הערימה מסוג הגרף שהוטמע. בסוף הריצה על זיכרון הערימה יתבצע ניסיון להרכיב מהעצמים שנמצאו גרף מייצג חוקי, אם נמצא כזה יתבצע אלגוריתם לחילוץ סימן המים (מספר) מהגרף שנמצא. אם נמצא כזה המספר יוצג ב-WM found. ניתן גם להציג וויזואלית את הגרף שנמצא (אם נמצא).



איור 22 – ממשק משתמש Extraction Tab

4.2.2 דוחות

רכיב הדוחות מציג דוח פעילות עבור כל שלב בביצוע הטבעת סימן המים וכן בשלב חילוץ סימן המים



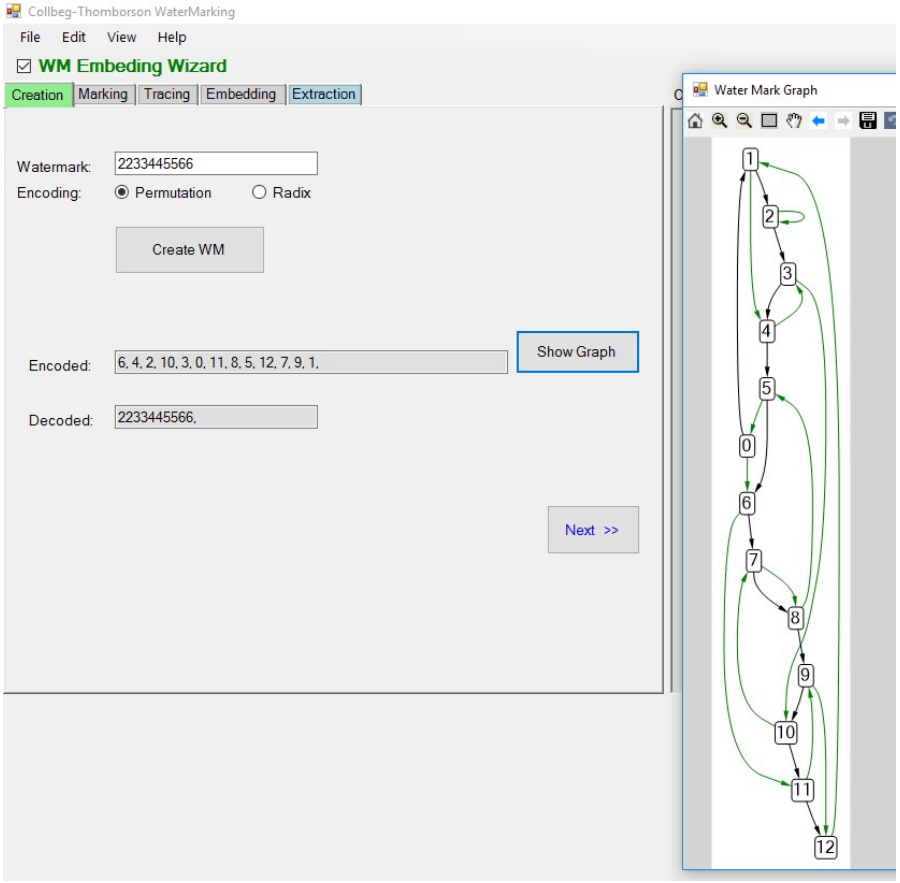
איור 23 – ממשק משתמש Outputs

באיור 13 – ממשק משתמש Outputs ניתן לראות את הדוח לשלב החילוץ, כאן מוצגים כל העצמים שחולצו מזיכרון הערימה של התוכנית, בכחול בולט, יוצגו העצמים שקשורים לגרף סימן המים.

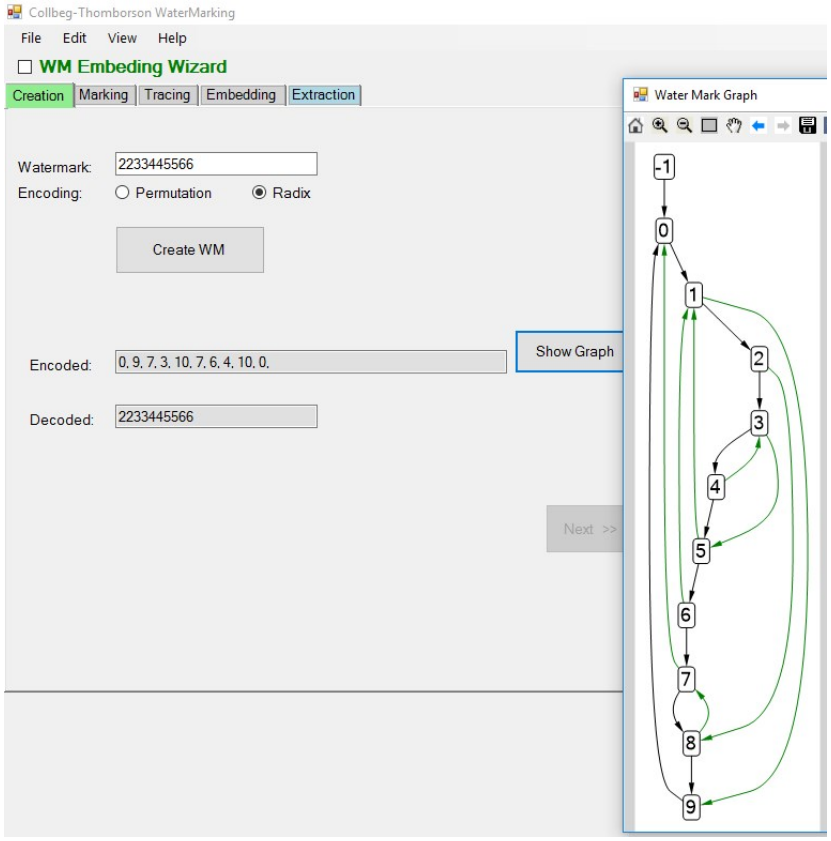
5. תוצאות, דוגמאות ומסקנות

ההרצה בוצעה באמצעות CTWM על התוכנית Calculator.

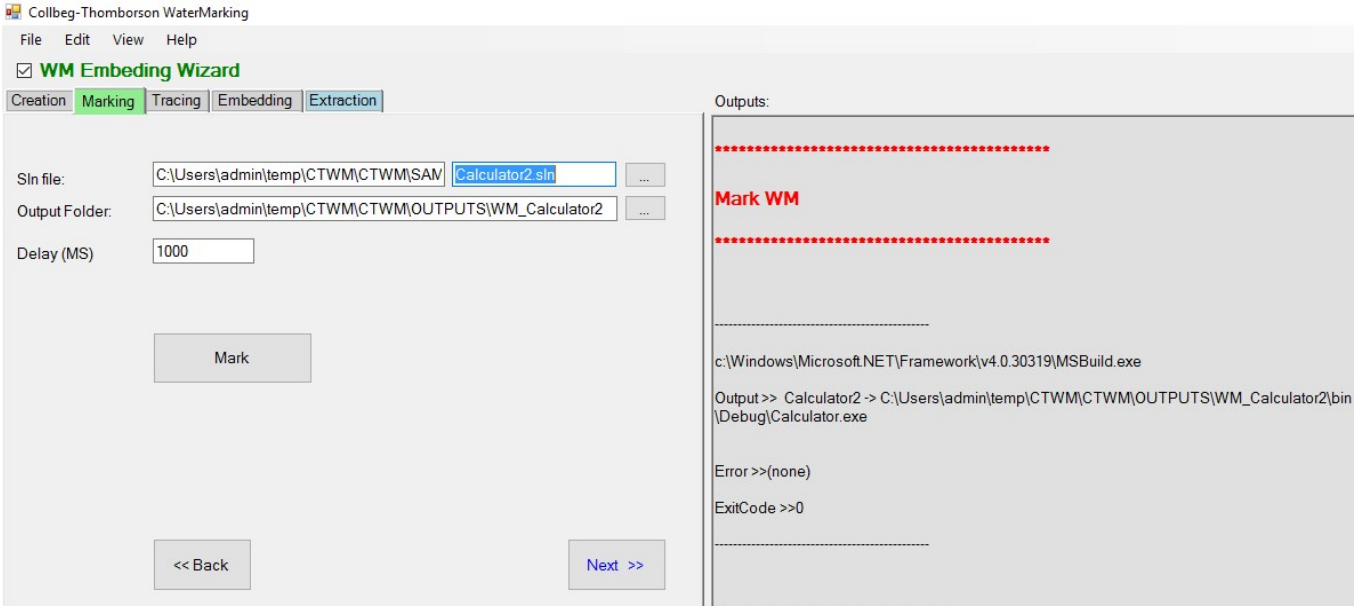
5.1 דוגמאות הרצה



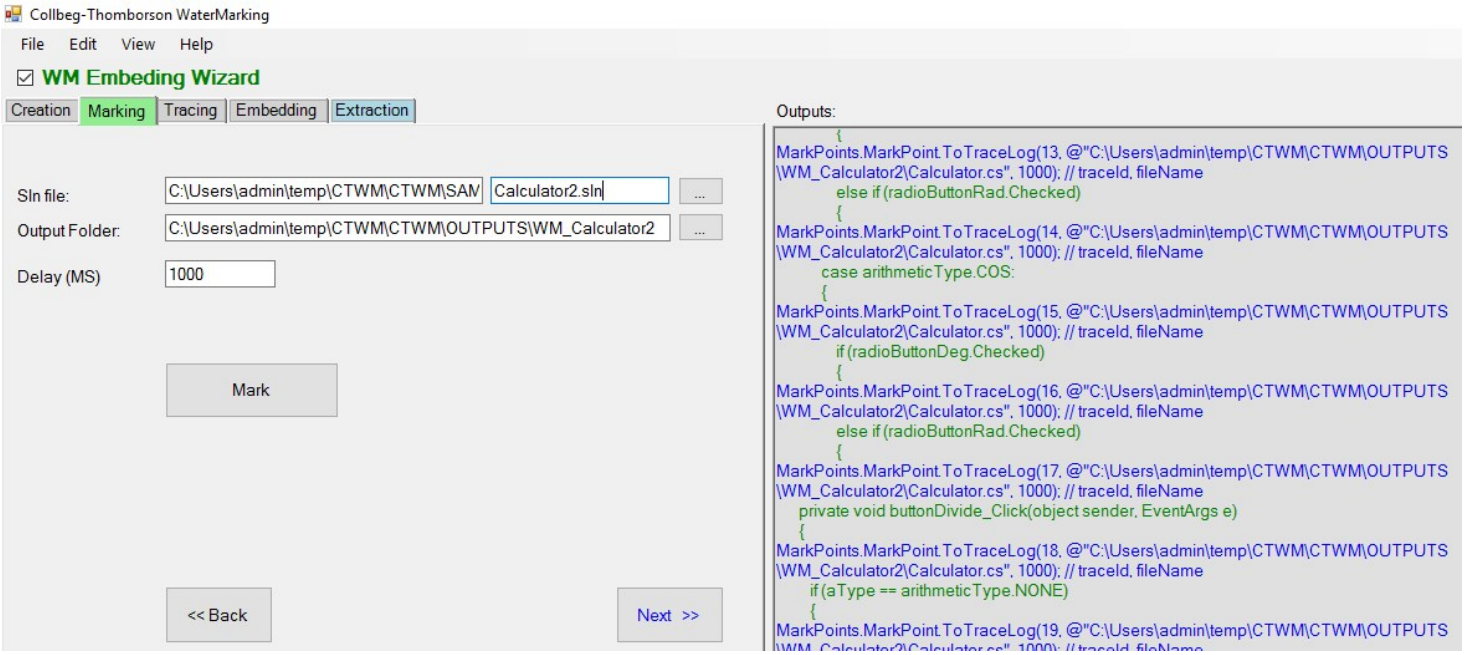
איור 24 – Creation - פרמוטציה וגרף מייצג עבור סימן המים "2233445566"



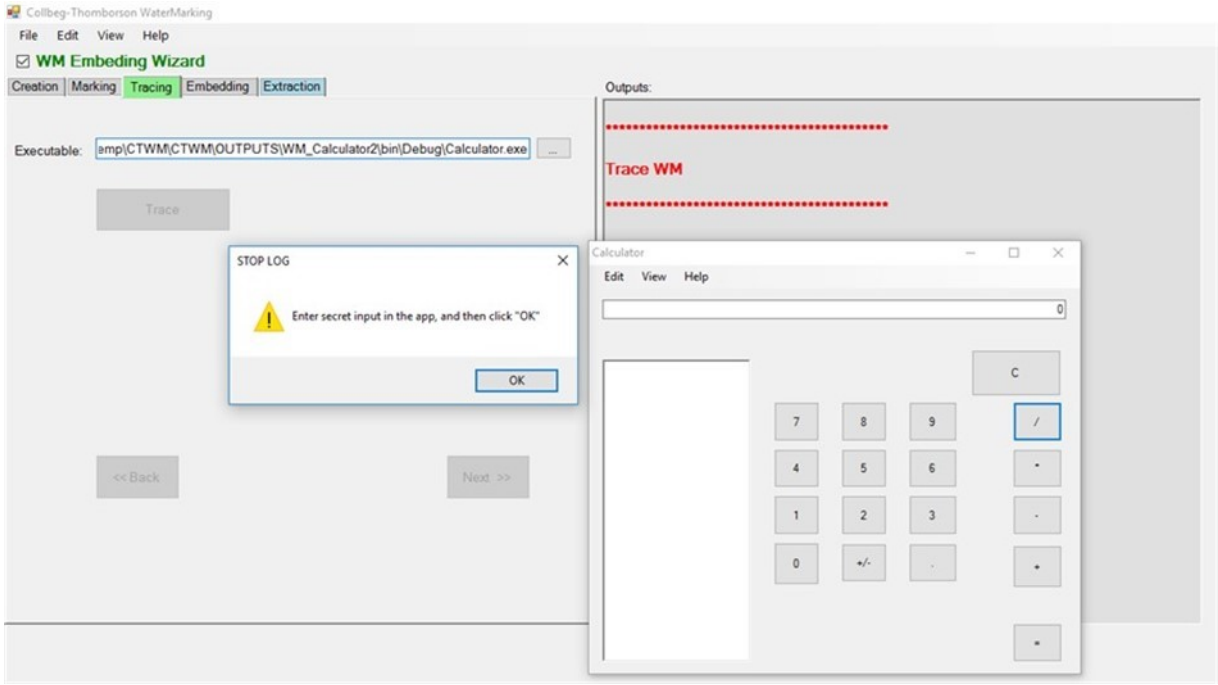
איור 25 – Creation – קידוד בסיס וגרף מייצג עבור סימן המים "2233445566"



איור 26 – Marking – סימון התוכנית Calculator2 (1)



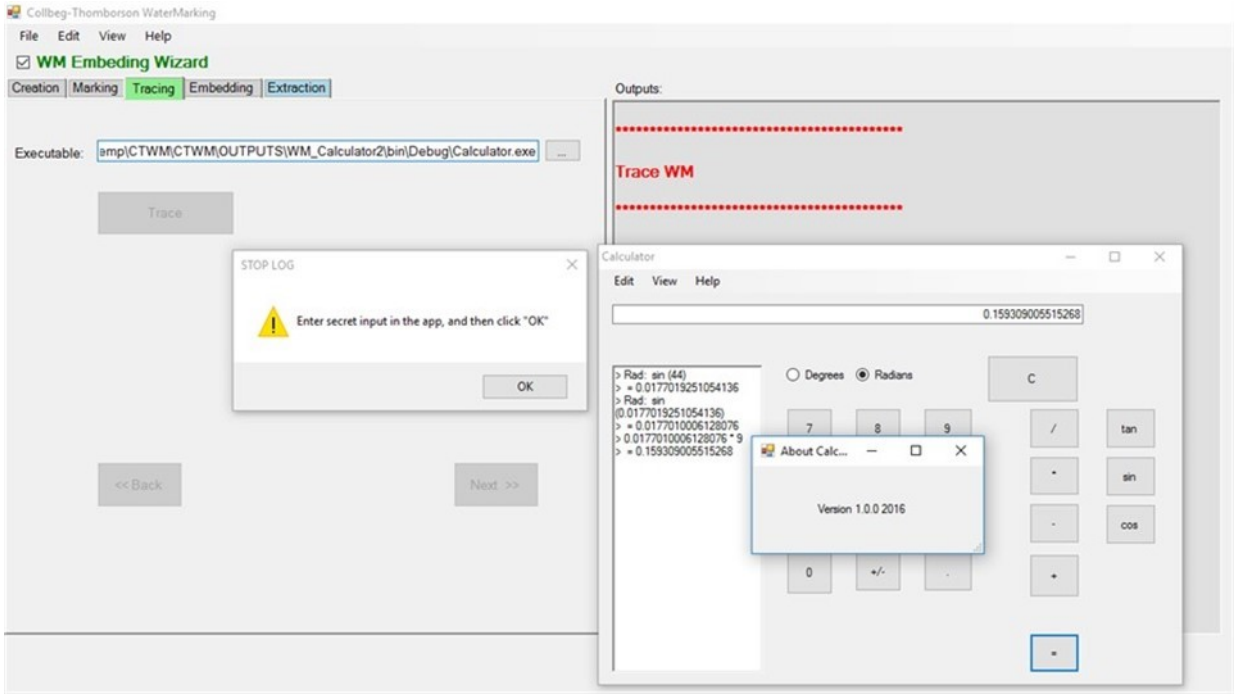
איור 27 – Marking – סימון התוכנית Calculator2 (2)



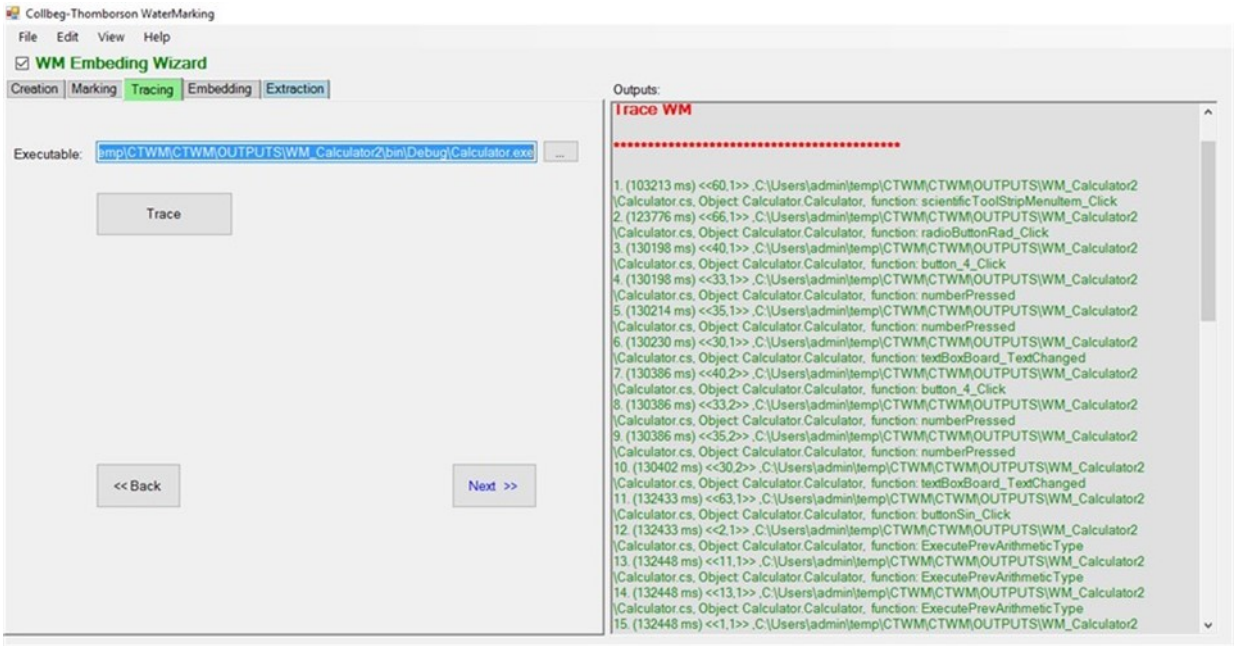
איור 28 – Tracing – ביצוע מעקב עם קוד סודי על התוכנית Calculator (1)

עם הקוד הסודי (קלט סודי) -

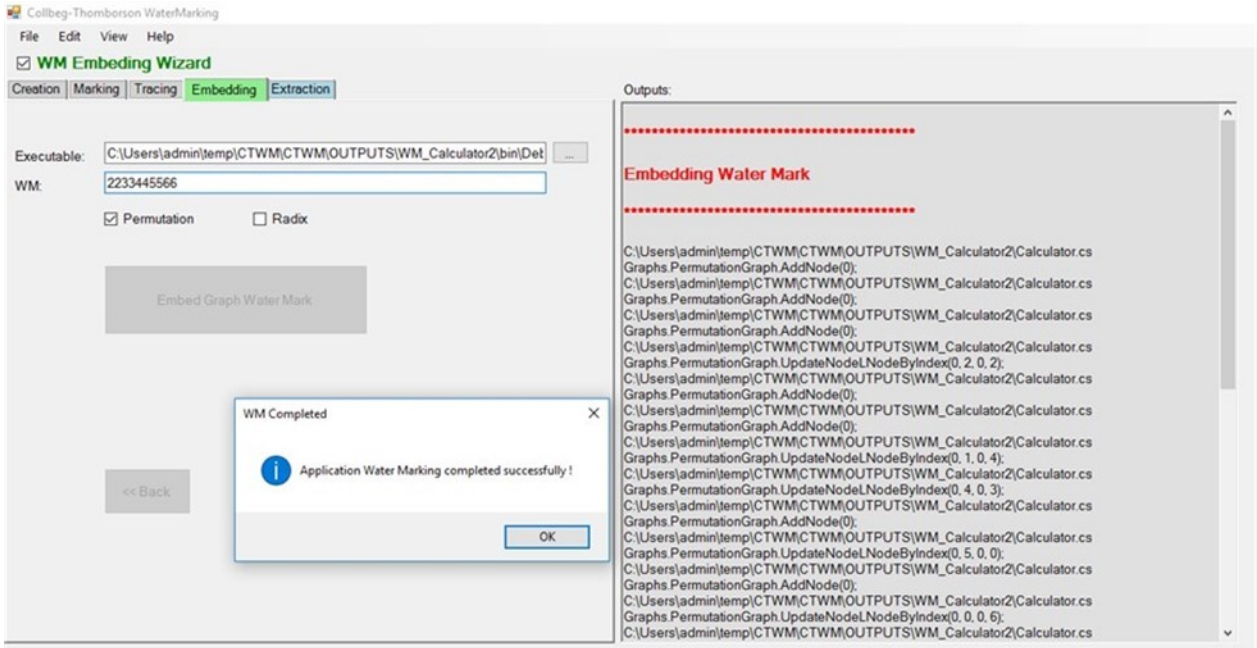
1. *Edit -> Scientific*
2. *Radians*
3. *44*
4. *Sin*
5. ***
6. *9*
7. *=*
8. *View ->History*
9. *Help -> About*



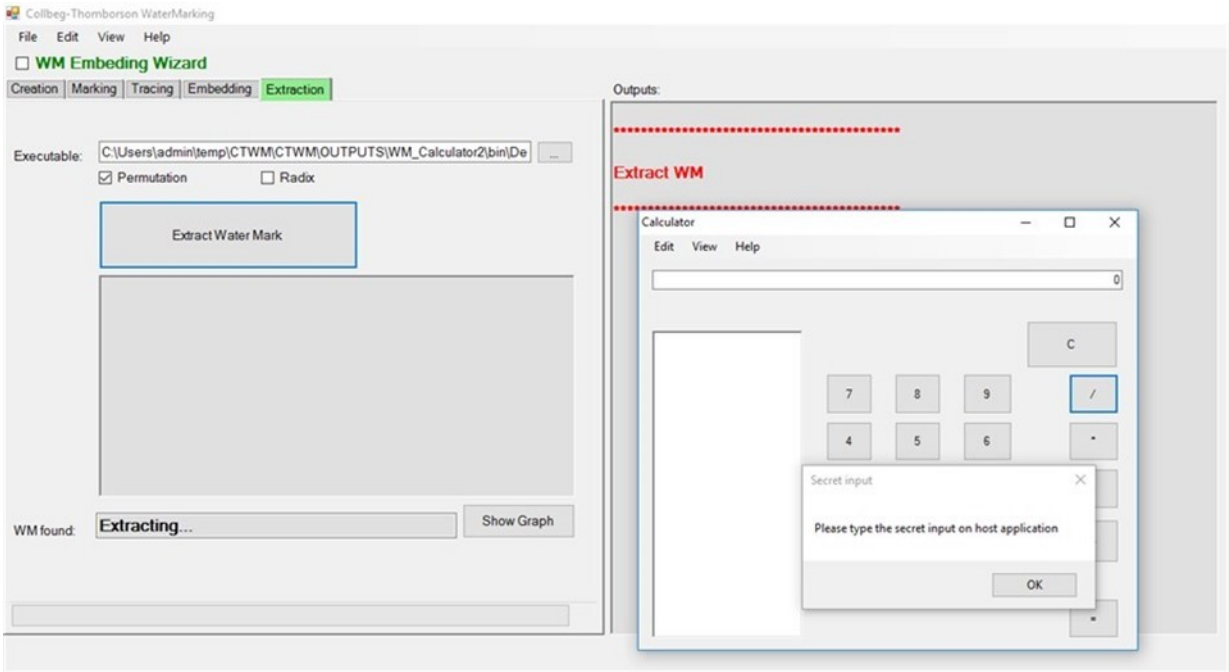
איור 29 – Tracing – ביצוע מעקב עם קוד סודי על התוכנית Calculator (2)



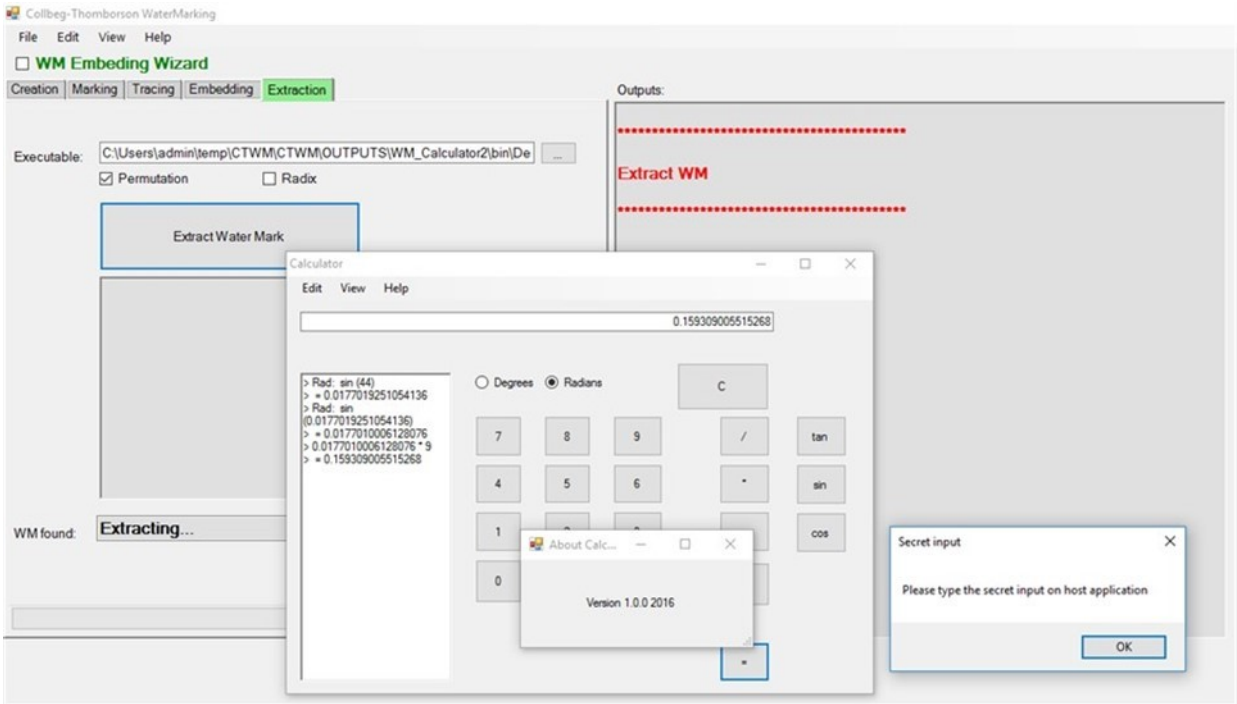
איור 30 – Tracing – ביצוע מעקב עם קוד סודי על התוכנית Calculator (3)



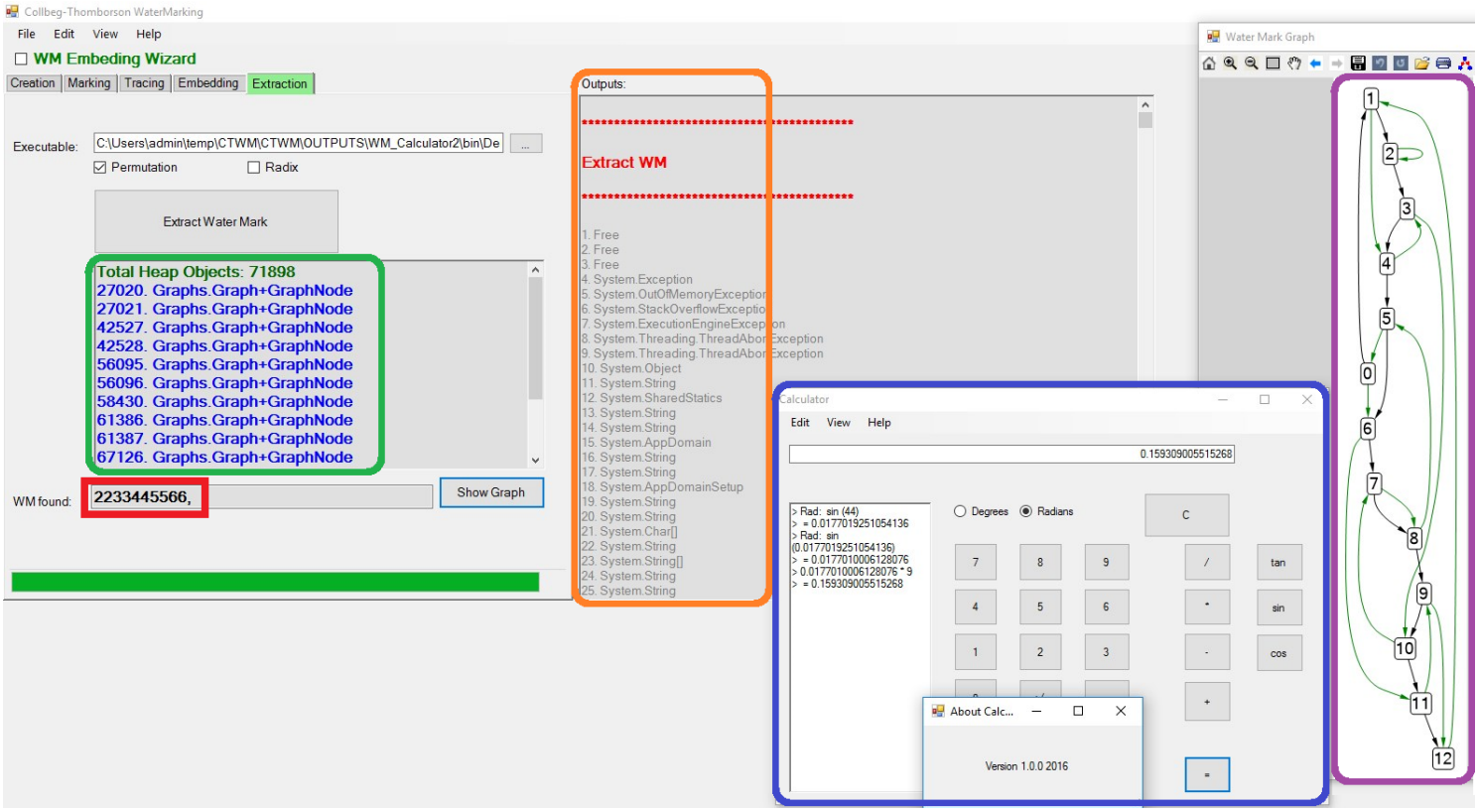
איור 31 – Embedding – שיבוץ גרף הפרמוטציה המייצג את סימן המים "2233445566" בתוכנית Calculator



איור 32 – Extraction – חילוץ סימן המים מהתוכנית Calculator (1)

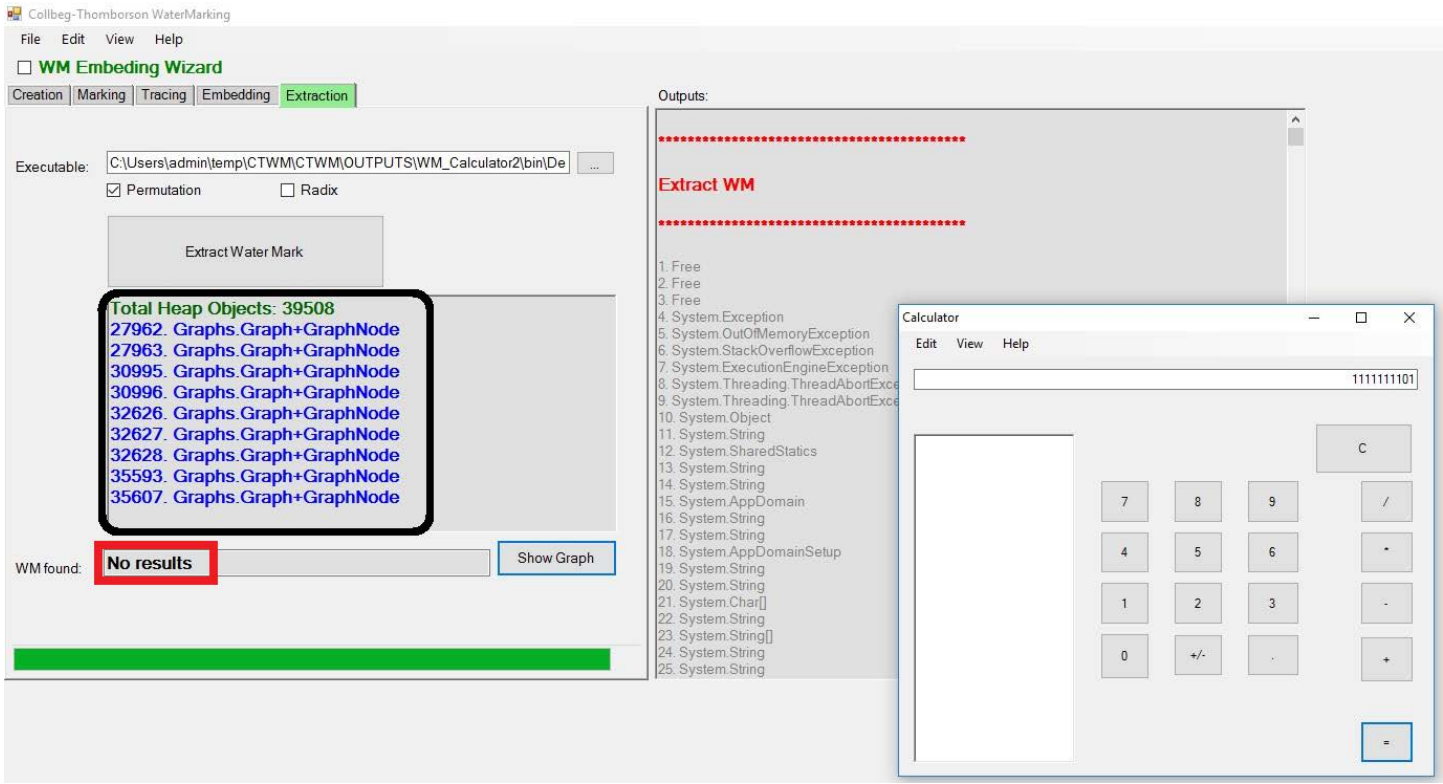


איור 33 – Extraction – חילוץ סימן המים מהתוכנית Calculator (2)



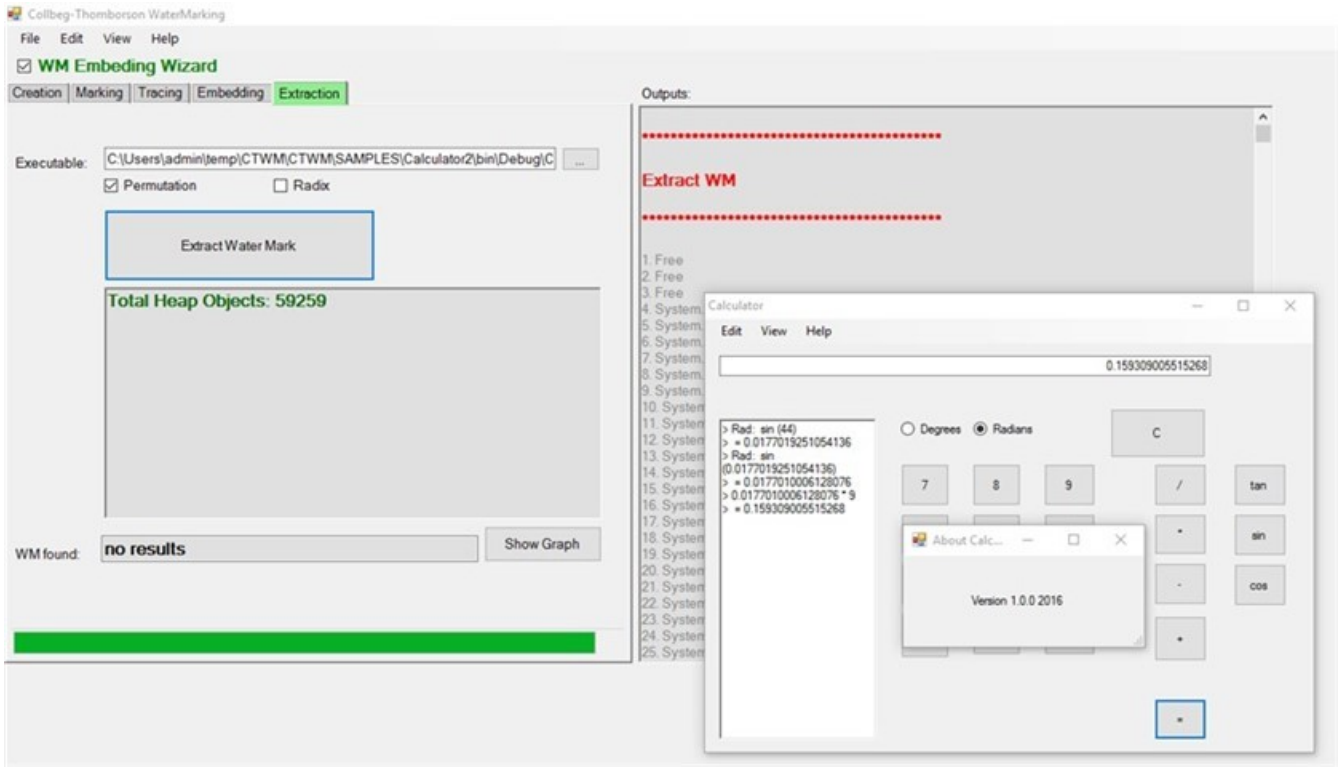
איור 34 – חילוץ מוצלח של סימן המים מהתוכנית Calculator, לאחר הזנת הקלט הסודי (3)

ניתן לראות את רשימת כל העצמים שחולצו מזיכרון הערימה (מוקפים במסגרת **כתומה**) של התוכנית Calculator (מוקפת במסגרת **כחולה**), את אלו מביניהם שמרכיבים את הגרף המייצג את סימן המים (מוקפים במסגרת **ירוקה**), את הגרף שחולץ (מוקף במסגרת **סגולה**) ואת סימן המים עצמו (מוקף במסגרת **אדומה**)

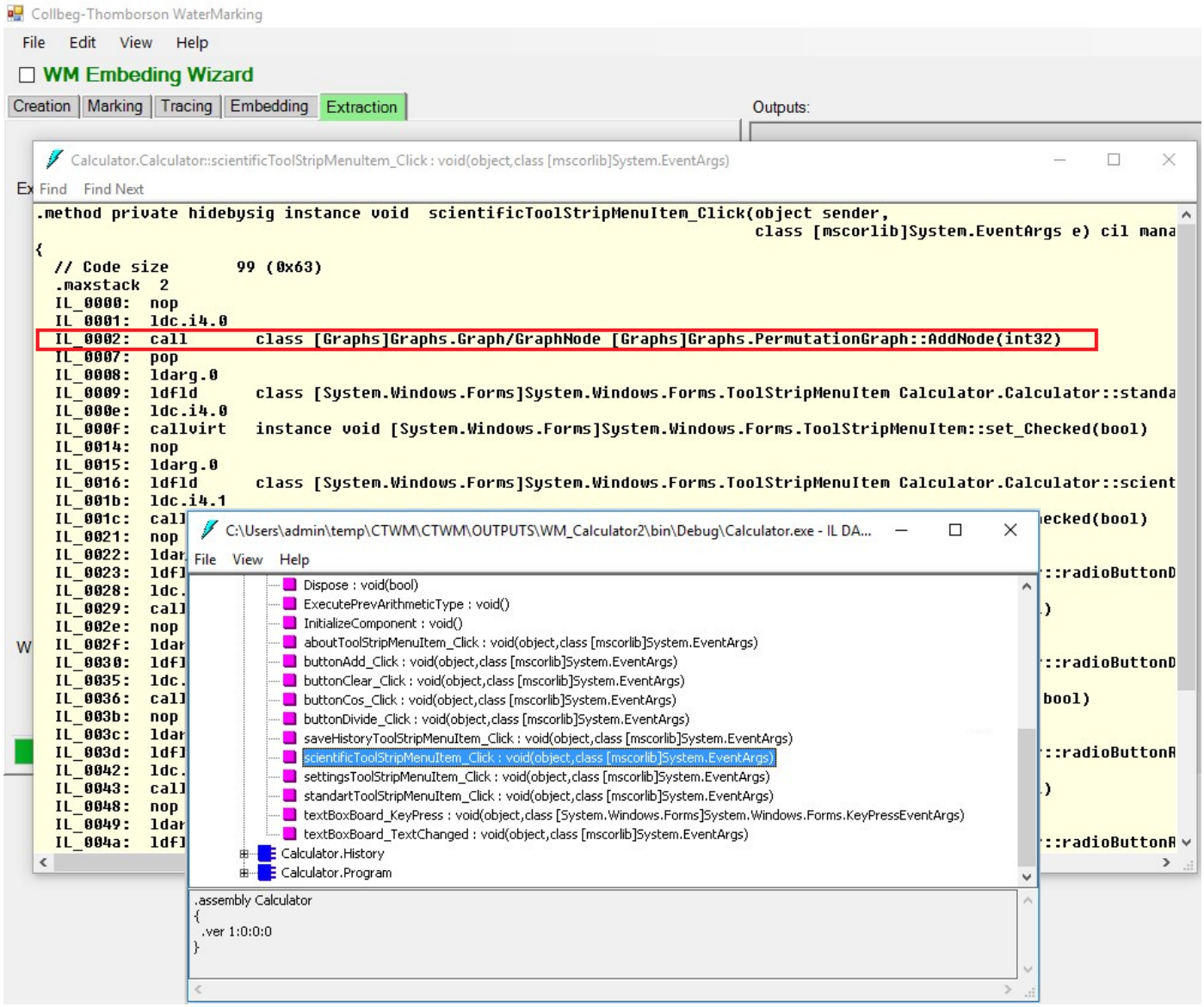


איור 35 – Extraction - סיום לא מוצלח של חילוץ סימן המים מהתוכנית Calculator לאחר הזנת קלט שאינו הקלט הסודי (4)

שימו לב שבזיכרון הערימה נבנו עצמים המרכיבים את הגרף (מוקפים במסגרת **שחורה**) וזאת מכיוון שהקלט שהוזן מכיל רכיבים הקיימים בקלט הסודי, אולם עצמים אלה אינם מייצגים גרף תקין ולכן התוכנית לא חילצה את סימן המים (התוצאה – No results מוקפת במסגרת **אדומה**).



איור 36 – Extraction - ניסיון ביצוע חילוץ (כושל) על תוכנית Calculator <u>שלא</u> הוטמע בה סימן מים, ברור שאינה מניבה תוצאות ללא תלות בקלט (5)



איור 37 – ILDASM – ניתוח התכנית Calculator שהוטמע בה סימן מים באמצעות ILDASM

ניתן להבחין בקריאות ליצירת הגרף בפונקציות של התוכנית. כאן בפונקציה ***ScientificToolStripMenuItem_Click*** ישנה קריאה להוספת צומת לגרף (מוקפת במסגרת **אדומה**).

5.2. תוצאות ומסקנות

המימוש עובד היטב ומוצא תוצאות, בכל המקרים שנבדקו.

האלגוריתם והמימוש אינם נטולי בעיות, הבעיות העיקריות שהתגלו בזמן המימוש:

- סימן המים אינו חבוי, ניתן לראות את הקריאות ליצירתו באמצעות ILDASM, ניתן להתגבר על כך על ידי ביצוע ערפול (Obfuscation). כמו כן ניתן להכניס דמויים (המימוש אינו טריוויאלי), כלומר קריאות שאינן אמתיות ליצירת הגרף, או כאלו שיגרמו לגרף להיווצר בצורה שאינה תקינה כאשר מוזן קלט שאינו תקין.
- בבחירת המקומות בקוד בהם יש לבצע פעולות הקשורות ליצירת הגרף, יש לבחור מקומות בהם התכנית עברה פעם אחת ויחידה במהלך הזנת הקלט הסודי, אחרת יש צורך בסימון מיוחד המציין את מספר הביקורים באותה שורה בתכנית בזמן ריצה, דבר המסבך מאוד את הטמעת סימן המים ובנוסף פעולה חריגה זו עלולה לעורר חשד אצל תוקף שעלול לזהות כי נקודה זו קשורה ליצירת סימן המים.
- סדר הזנת הקלט אינו משנה במימוש זה, לצורך העניין אם הקלט הסודי (על תוכנית המחשבון) הוא "5 * 8 + 3 =" אזי גם הקלט "3 + 8 * 5 =" יעבוד.
- אם הקלט הסודי שהוזן הוא תת קבוצה של הקלט שנבדק אזי גם ייתכן שייבנה הגרף המתאים בזיכרון הערימה וייתכן שיחולץ סימן המים.
- נצפו מקרים בהם חולץ יותר מגרף אחד שמתאים לקידוד ולכן, במקרים כאלו הוצגו כל סימני המים שנמצאו. תופעה זו קוראת לרוב כאשר סימן המים הוא מספר קטן.
- זמני הריצה בכל שלבי השיבוץ הם זניחים, ועולים מעט ככל שגודל התוכנית עולה.
- זמן הריצה בזמן החילוץ הוא משמעותי ותלוי בכמות העצמים שהוקצו בערימה בזמן ביצוע החילוץ ובעצמת המחשב המבצע. עבור הדוגמא בפרק 5, במחשב עם מעבד אינטל I7 וזיכרון של 8 gb ram משך החילוץ בסריקה של סה"כ 56,518 עצמים היה כ-2:30 דקות (cpu 13%).
- גודל התוכנית בה מוטמע סימן המים גדל. בתוכנית המחשבון בדוגמה שניתנה בפרק 5, עבור סימן מים מסוג פרמוטציה, הייתה עליה של כ- 2% אחוזים בהידור בתצורת debug (מ 26,112 בתים ל -26,624 בתים), ובכ-4% בהידור בתצורת release (מ 24,576 בתים ל-25,600 בתים).
- ככל שסימן המים המוטמע גדול יותר התוכנית המוטמעת גדולה יותר. לדוגמא, הטמעת סימן המים "223344556677889911" מסוג פרמוטציה בתצורת release הגדילה את תוכנית המחשבון, ביחס לדוגמא בסעיף 5 בכ- 2% (מ-25,600 בתים ל-26,112 בתים).
- הטמעת סימן מים באמצעות קידוד בסיס יוצרת לרוב תוכנית מעט קטנה יותר מאשר פרמוטציה (עד כ-2% בתוכנית המחשבון).

6. סיכום

הפרויקט מימש את אלגוריתם CT מתחילתו ועד סופו והדגים אותו על תכנית אמיתית. התשתית שנוצרה בפרויקט יכולה לשמש כאמצעי להזנת סימן מים בתוכניות הכתובות ב-Net. #C.

למרות זאת, לגבי השאלה האם שימוש בהטמעת סימן מים בתוכנה באמצעות אלגוריתם CT הוא יעיל, שימושי או עמיד, התשובה אינה חד משמעית שכן הוא מראה חולשות ובעיות שיש לתת עליהם את הדעת, הרחבה בנושא ניתנת בעבודה המסכמת.

7. רשימת מקורות

[1] C. S. Collberg, C. Thomborson and G. M. Townsend
Dynamic Graph-Based Software Fingerprinting, ACM Transactions on Programming Languages and Systems, Vol. 29,No. 6, Article 35, Publication date: October 2007.

[2] J. Hamilton and S. Danicic, A Survey of Static Software Watermarking. Internet security (WorldCIS), 2011 World Congress on, 100-107 (IEEE).

[3] C. S. Collberg, G. Myles and A. Huntwork. Sandmark—A Tool for Software Protection Research, IEEE Security & Privacy 2003, vol.1, Issue No.04, pp. 40–49, July-August

[4] globalstudy.bsa.org/2016/ (BSA, The software alliance)

[5] www.Wikipedia.org

[6] W. Myrvold and F. Ruskey 2001. Ranking and unranking permutations in linear time. Inf.Proc. Lett. 79, 6 (Sept.), 281–284.

[7] R. Davidson and N. Myhrvold. Method and system for generating and auditing a signature for a computer program, June 1996. Microsoft Corporation, US Patent 5559884.

[8] https://github.com/Microsoft/clrmd/blob/master/Documentation/WalkingTheHeap.md

[9] https://www.microsoft.com/en-us/research/project/microsoft-automatic-graph-layout/

[10] https://github.com/Microsoft/automatic-graph-layout/issues/35