האוניברסיטה הפתוחה – החטיבה למדעי המחשב

מימוש והדגמת שיטות בתחום הסטגנוגרפיה בקבצי תוכן

פרויקט מתקדם זה הוגש כחלק מהדרישות לקבלת תואר

מוסמך למדעים במדעי המחשב

באוניברסיטה הפתוחה

החטיבה למדעי המחשב

מאת: דניאל גרפונקל

בהנחית: פרופ' אהוד גודס

תוכן עניינים

1. כללי 6
1.1. מטרת המסמך 6
1.2. מבנה המסמך 6
2. רקע 6
2.1. סטגנוגרפיה 6
2.1.1. מוטיבציה 6
2.1.2. טכניקות 7
2.2. סטגנוגרפיה בקבצים – גישות עיקריות 8
2.2.1. החבאת מידע ב-data של הקובץ 8
2.2.2. החבאת מידע בפורמט הקובץ (Metadata) 8
3. על הפרויקט 8
3.1. מוטיבציה 8
3.1.1. התקפה על שרת הקבצים 8
3.1.2. שימוש בשרת כ-Proxy 9
3.2. תכולות ופיצ'רים 9
3.3. דרכים להרחבה 10
3.3.1. פורמטים נתמכים 10
4. שיטות סטגנוגרפיה הממומשות בפרויקט 11
4.1. End of File 11
4.1.1. פורמטים נתמכים 11
4.1.2. פרטים טכניים 11
4.2. Jpeg COM 12
4.2.1. רקע על JFIF 12
4.2.2. COM Segment 13
4.2.3. פרטים טכניים 13
4.3. Bitmap Gap 14
4.3.1. פורמט Bitmap 14
4.3.2. פרטים טכניים 16
4.4. Bitmap LSB 17
4.4.1. תיאור ה-Data של קבצי Bitmap 17
4.4.2. תיאור השיטה 18
4.4.3. פרטים טכניים 18
4.5. Wave Additional Chunk 19
4.5.1. רקע על פורמט Wave 19
4.5.2. מבנה פורמט Wave 20
4.5.3. תיאור השיטה 20

4.5.4. פרטים טכניים 21
Wave LSB.4.6 21
4.6.1. תיאור ה-Data של קבצי Wave 22
4.6.2. תיאור השיטה 22
4.6.3. פרטים טכניים 23
Zip Beginning.4.7 23
4.7.1. על השיטה 23
4.7.2. פרטים טכניים 24
Zip Additional Record.4.8 25
4.8.1. פורמט Zip 25
4.8.2. על השיטה 25
4.8.3. פרטים טכניים 26
5. תוצאות 27
5.1. End of File 27
5.1.1. תוצאות ויזואליות 27
5.1.2. תוצאות בינאריות 28
5.1.3. ניתוח השיטה 29
Jpeg COM.5.2 29
5.2.1. תוצאות ויזואליות 29
5.2.2. תוצאות בינאריות 29
5.2.3. ניתוח השיטה 30
Bitmap Gap.5.3 30
5.3.1. תוצאות ויזואליות 30
5.3.2. תוצאות בינאריות 31
5.3.3. ניתוח השיטה 33
Bitmap LSB.5.4 33
5.4.1. תוצאות ויזואליות 33
5.4.2. תוצאות בינאריות 34
5.4.3. ניתוח השיטה 35
Wave LSB.5.5 35
5.5.1. תוצאות בינאריות 35
5.5.2. ניתוח השיטה 37
Wave Additional Chunk.5.6 38
5.6.1. תוצאות בינאריות 38
5.6.2. ניתוח השיטה 38
Zip Beginning.5.7 39
5.7.1. תוצאות ויזואליות 39

5.7.2. תוצאות בינאריות 39
5.7.3. ניתוח השיטה 40
5.8. Zip Additional Record 40
5.8.1. תוצאות ויזואליות 40
5.8.2. תוצאות בינאריות 40
5.8.3. ניתוח השיטה 41
6. שימוש ביישום 41
6.1. הטמעת מסר סמוי 42
6.2. חילוץ מסר סמוי 44
7. פירוט טכני 46
7.1. פעולות I/O 46
7.1.1. Stream 46
7.1.2. ReadStream 46
7.1.3. WriteStream 47
7.2. מזהיי קבצים 48
7.3. שיטות סטגנוגרפיה 49
7.4. API 50
7.4.1. FileTypeMatcher 50
7.4.2. StegoManager 51
7.5. הרחבת הפרויקט 51
8. מקורות 52

רשימת איורים

איור 1 : סטגנוגרפיה כתזרים 7
איור 2 : תרשים זרימה להחבאת מסר בקובץ בתוכנת StegoMaster 9
איור 3 : תרשים זרימה לחילוץ המסר הסמוי בתוכנת StegoMaster 10
איור 4 : תרשים זרימה לתיאור החבאת מסר סמוי ב-COM segment 13
איור 5 : תרשים זרימה להחבאת מידע ב-Gap של קובץ Bitmap 16
איור 6 : תיאור חלוקה של פיקסל לערוצי צבע 18
איור 7 : תרשים זרימה להחבאת מידע ב-LSB של הפיקסלים של תמונת Bitmap 19
איור 8 : תרשים זרימה לתהליך החבאת מידע ב-chunk של קובץ Wave 21
איור 9 :החבאת מסר סמוי ב-LSB של דגימות Wave 22
איור 10 : קידוד מסר סמוי ל-LSB בדגימות קובץ Wave 23
איור 11 : החבאת מסר סמוי בתחילת קובץ ה-Zip 24
איור 12 : מבנה קובץ Zip 25
איור 13 : תהליך החבאת מידע בקובץ Zip על ידי הוספת Record 26
איור 14 : תצוגה בינארית של קובץ ה-Jpeg בעל המסר הסמוי בשיטת End of File 28
איור 15 : מבנה קובץ ה-Jpeg בעל המסר הסמוי בשיטה Jpeg COM 30
איור 16 : תצוגה בינארית של קובץ ה-Bitmap בעל המסר הסמוי בשיטת Bitmap Gap 31
איור 17 : תצוגת ה-Header של קובץ ה-Bitmap בעל המסר הסמוי בשיטה Bitmap Gap 32
איור 18 : השוואה בינארית בעורך 010editor בין קבצי ה-Bitmap בשיטה Bitmap LSB 34
איור 19 : השוואה בינארית בעורך 010editor בין קבצי ה-Wave בשיטה Wave LSB 36
איור 20 : מבנה קובץ ה-Wave לאחר הטמעת המסר הסמוי בשיטה Wave LSB 36
איור 21 : תצוגה בינארית של קובץ ה-Wave לאחר הטמעת המסר הסמוי בשיטה Wave LSB. 37
איור 22 : מבנה קובץ ה-Wave לאחר הטמעת המסר הסמוי בשיטה Wave Additional Chunk 38
איור 23 : תצוגה בינארית של קובץ ה-Wave בעל המסר הסמוי בשיטת Wave Additional Chunk 38
איור 24 : תצוגה של קובץ ה-Zip בעל המסר הסמוי בשיטת Zip Beginning בתוכנת WinRAR 39
איור 25 : תצוגה בינארית של קובץ ה-Zip בעל המסר הסמוי בשיטת Zip Beginning 39
איור 26 : תצוגה של קובץ ה-Zip בעל המסר הסמוי בשיטת Zip Additional Chunk בתוכנת WinRAR 40
איור 27 : מבנה קובץ ה-Zip בעל המסר הסמוי בשיטה Zip Additional Chunk 41
איור 28 : תצוגה בינארית של קובץ ה-Zip בעל המסר הסמוי בשיטת Zip Additional Chunk . 41
איור 29 : חלון StegoMaster האחראי על החבאת מסרים 42
איור 30 : תווית החבאת המסרים 42
איור 31 : כפתור בחירת נתיב הקובץ הנשא להחבאת המסר 43
איור 32 : חלון שגיאת בחירת קובץ מפורמט לא מזוהה 43
איור 33 : רשימת בחירת השיטה הסטגנוגרפית 43
איור 34 : בחירת נתיב קובץ הפלט וכתיבת המסר הסמוי 44
איור 35 : הודעת סיום הטמעת מסר מסוי בהצלה 44
איור 36 : תווית חילוץ מסרים סמויים 44
איור 37 : כפתור בחירת נתיב הקובץ הנשא חילוץ המסר 45
איור 38 : רשימת בחירת השיטה הסטגנוגרפית לחילוץ המסר הסמוי 45
איור 39 : איזור הטקסט שבו מופיע המסר הסמוי 46
איור 40 : תרשים מחלקות המתאר streams 48
איור 41 : תרשים מחלקות של מזהיי הקבצים 48
איור 42 : תרשים מחלקות המתאר את השיטות הסטגנוגרפיות 49
איור 43 : תרשים מחלקות המתאר את מחלקת ה-FileTypeMatcher 50
איור 44 : תרשים מחלקות המתאר את מחלקת ה-StegoManager 51

רשימת טבלאות

טבלה 1 : השוואה ויזואלית של תמונות Jpeg בשיטה End of File 27
טבלה 2 : השוואה ויזואלית של תמונות Bitmap בשיטה End of File..... 28
טבלה 3 : השוואה ויזואלית של תמונות Jpeg בשיטה Jpeg COM..... 29
טבלה 4 : השוואה ויזואלית של תמונות Bitmap בשיטה Bitmap Gap 31
טבלה 5 : תצוגה בינארית של ה-Headers של קובץ ה-Bitmap בעל המסר הסמוי בשיטה Bitmap Gap 32
טבלה 6 : השוואה ויזואלית של תמונות Bitmap בשיטה Bitmap LSB 33
טבלה 7 : השוואה בין מבני קבצי Bitmap לפני ולאחר הטמעת מסר סמוי בשיטה Bitmap LSB 34
טבלה 8 : השוואה ויזואלית של תמונות Bitmap בעומק סיביות 24 בשיטה Bitmap LSB..... 34

1. כללי

1.1. מטרת המסמך

מסמך זה בא לתאר את פרויקט המתקדם במדעי המחשב באוניברסיטה הפתוחה במחלקה למדעי המחשב. פרויקט זה עוסק בשיטות סטגנוגרפיות בקבצי תוכן, תיאורן, ניתוחן ומימושן. המסמך מלווה את בניית הפרויקט StegoMaster משלב המחקר, תיאור השיטות הסטגנוגרפיות ומבנה הקבצים באופן מפורט לכדי הבנת המימוש באופן מיטבי.

1.2. מבנה המסמך

ראשית נסקור במסמך זה את עולם הסטגנוגרפיה, מקורות ואפיון התהליך. כמו כן, מסמך זה יגע בהסברים כלליים על הפרויקט, מטרותיו ותכונותיו הכלליות. לאחר מכן נסקור את הקבצים בהם תוכנת StegoMaster תומכת תוך כדי התמקדות בדרגות החופש בהם. לאחר מכן, מסמך זה יסקור מספר שיטות סטגנוגרפיות בכל אחד מן הפורמטים שראינו.

לבסוף, מסמך זה יסקור לעומק את תוכנת StegoMaster, את תכונותיה, הבחירות המימושיות שנעשו בתהליך העיצוב התוכנה ובתהליך הפיתוח. מסמך זה מלווה בתוכנת StegoMaster עצמה שביחד משלימים אחד את השני לתוצר הפרויקט המתקדם.

2. רקע

2.1. סטגנוגרפיה

2.1.1. מוטיבציה

תחום הסטגנוגרפיה עוסק בטכניקות של החבאת מסר סודי בתוך מסר מארח כאשר המטרה היא לאפשר לשני צדדים לתקשר בחשאיות, מבלי לעורר חשד מצד מצוטטים פוטנציאלים. בעבודתי, הנחת היסוד היא שהמסר המארח הוא קובץ, מפורמט ידוע מראש כלשהו.

בניגוד לקריפטוגרפיה, אמנות ההצפנה, בה תוכן ההודעה הוא הסוד, כאשר מדובר בסטגנוגפריה, הסוד הוא עצם קיומו של ערוץ התקשורת. כלומר, בקריפטוגרפיה, צופה מהצד יכול להבחין במידע שעובר בערוץ התקשורת אך לא ידע את משמעותו, בעוד שבסטגנוגרפיה, אותו צופה לא ידע על קיום הערוץ. כמו כן נציין, כי לעתים נעשה שימוש בשילוב של שתי השיטות הנ״ל ומחביאים באמצעות סטגנוגרפיה מידע מוצפן.

2.1.1.1. מקורות הסטגנוגרפיה

בעיית החבאת ערוצי התקשורת קיימת עוד מימי הרומאים לפני הספירה, כאשר האדונים היו שולחים את עבדיהם עם מסר מקועקע לקרקפתם אל עבר יעד ההודעה. כאשר העבד הגיע אל הנמען, הוא גולח ומקבל ההודעה היה יכול לקרוא את ההודעה בשלמותה.

2.1.1.2. סטגנוגפריה בעולם המודרני

עם התפתחות הטכנולוגיה והתקדמות עולם המחשוב, תחום הסטגנוגרפיה נכנס גם לערוצי התקשורת הממוחשבים וכיום ניתן למצוא התייחסויות לנושאי החבאת המסרים בפרוטוקולי תקשורת, תמונות, וידאו, שמע ועוד.

2.1.2. טכניקות

הטמעת הערוץ הסמוי בעזרת סטגנוגרפיה בתוכנה, מתוארת ב- [1] בצורה הבאה:

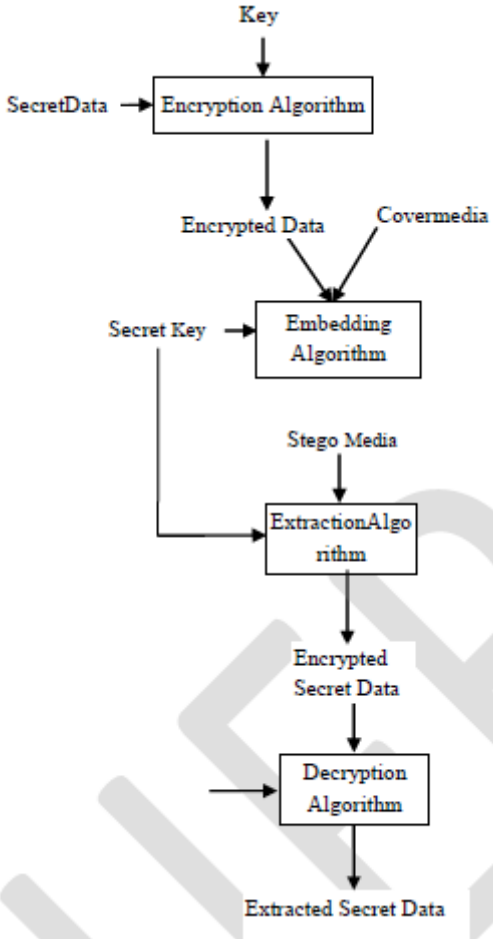
קלט האלגוריתם: קובץ נשא, מסר סמוי

פלט האלגוריתם: קובץ הנושא בתוכו ערוץ סטגנוגרפי

מהלך האלגוריתם:

1. קבל מהמשתמש את המסר הסמוי
2. קבל מהמשתמש את קובץ הנשא
3. קדד את המסר הסמוי בקידוד הנתמך על ידי אלגוריתם השילוב
4. הצפן את המסר הסמוי
5. הכנס את המסר הסמוי אל תוך קובץ הנשא
6. החזר את הקובץ הנושא בתוכו את הערוץ הסטגנוגרפי.

ראה גם איור 1 הנלקח ממאמר [1]



איור 1: סטגנוגרפיה כתזרים

שלבים 3 ו-4 באלגוריתם הם אופציונליים, כלומר, קידוד המסר הסמוי בקידוד הנתמך על ידי המערכת הוא עניין טכני בלבד ולכן אין בו עניין בשיח תיאורטי. הצפנת המסר הסמוי (כפי שכתוב בסעיף ד') תורמת רבות לחוזק הערוץ, אך חוזק זה אינו נובע מחוזק הסטגנוגרפיה אלא מחוזק קריפטוגרפי.

שלב 5 באלגוריתם המוצג לעיל הוא השלב שילווה אותנו לכל אורך העבודה, כלומר, איך בעל המסר יכול להזריק מידע כרצונו (או בעל מגבלות מסוימות) אל קובץ מסוים מבלי שהמחזיק, או המתבונן בו ירגיש.

2.2. סטגנוגרפיה בקבצים – גישות עיקריות

קימות שתי גישות עיקריות להטמעה של תוכן סמוי בקבצים על ידי שימוש בסטגנוגרפיה. אחת עושה שימוש במבנה הקובץ וביכולותיו השונות, כלומר לרוב Metadata, והשנייה מחביאה את הערוץ הסמוי במידע הקובץ עצמו, כלומר ב-data.

ההפרדה בין הגישות נובעת מצרכים ומקשיים הגנתיים במניעת (גילוי או שיבוש) הערוצים המוחבאים. כלומר, בקבצים בהם ניתן למנוע בקלות יחסית את הערוצים הסמויים – צריך להחביא את המידע ב-data עצמו, בעוד שבקבצים סבוכים, בהם קשה לגלות קיום של מידע מוחבא, יש דרכים רבות להסתיר את המידע הסמוי ב-Metadata.

2.2.1. החבאת מידע ב-data של הקובץ

חלק מהמאמרים שאסקור בעבודה, כגון [1] , [2] , [3] ו- [4] עוסקים בהסתרת ערוצים סטגנוגרפיים בקבצי מולטימדיה. החבאת המידע בקבצי מולטימדיה לרוב תעשה ב-data עצמו משום שקבצים אלה לרוב יעברו המרות או עיבוד כלשהו בעת השימוש – לדוגמה העלאה לרשתות חברתיות עשויה להמיר את פורמט הקובץ. כל עוד אותו עיבוד משמר מידע, הערוץ הסמוי החבוי ב-data לא יפגע.

2.2.2. החבאת מידע בפורמט הקובץ (Metadata)

בפורמטים עשירים יותר, לדוגמה פורמטי Open Office XML או פורמט HTML, ניתן להחביא בבטחה את המידע הסמוי ב-Metadata. סביר להניח כי קבצים מן הפורמטים הללו יערכו במהלך חייהם על ידי אדם, ופחות סביר כי יעברו המרות כלשהן. לכן, החבאת מידע ב-data עלולה להיפגם, בעוד שהחבאה ב-Metadata תשרוד.

3. על הפרויקט

3.1. מוטיבציה

העולם כפי שאנו חיים בו היום נשען באופן בולט מאוד על המרחב הקיברנטי בכל תחומי החיים. עובדה זו פתחה דלת עבור עבריינים ופושעים להשיג את מטרותיהם באמצעות התקפות סייבר. סוג מסוים של התקפות כאלה, הוא הזלגת מידע – כלומר, גישה למידע רגיש מרחוק.

פרויקט זה מדגים יכולות של החבאת ערוצי מידע סמויים בקבצי תוכן עבור העולם האקדמי. ערוצי המידע הסמויים יכולים לשמש לצורך התקפות סייבר.

דוגמה ממשית להזלגת מידע על גבי ערוץ סטגנוגרפי היא בשרת קבצים, מסוג כלשהו – בין אם מתקשרים איתו בפרוטוקול FTP להורדת הקבצים או שהוא מיחצן שירות Web שצורכים ב-HTTP. בפסקה זו אתאר שני מתארי תקיפה באמצעות הזלגת מידע על גבי ערוץ סמוי באמצעות שרת קבצים.

3.1.1. התקפה על שרת הקבצים

בהינתן כלי תקיפה שהונח על שרת הקבצים, כזה שמטרתו להחזיר מידע אל התוקף אודות השרת או הרשת בה הוא נמצא – ניתן לתאר תקיפה בה הזלגת המידע החוצה נעשית באמצעות קבצים.

כלי התקיפה אשר הונח על השרת דולה את המידע ממנו, ולאחר מכן מטמין את המידע בקבצים אותם הוא מיחצן. ככה התוקף, יכול לגשת אל הקבצים מרחוק, להוריד אותם ולפענח את המסר הסמוי.

תפקיד הסטגנוגרפיה הוא כמובן להחביא את התקשורת בין התוקף אל השרת, דבר שללא סטגנוגפריה היה ניתן לראות בכלים המנטרים את התקשורת כמו לדוגמה WireShark. אם התוקף רוצה לשפר את התקפה זו הוא יכול להשתמש בשרת Proxy או ב-Tor, ככה שאפילו לא תהיה הוכחה שהוא עצמו ניגש לשרת הקבצים.

3.1.2. שימוש בשרת כ-Proxy

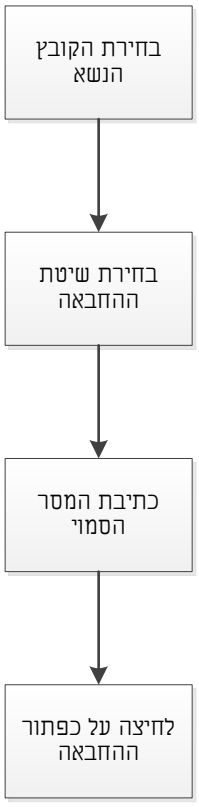
שימוש נוסף אפשרי לפי התרחיש הבא, בו שני גורמים רוצים להחביא את ערוץ התקשורת ביניהם, כך שצופה מהצד לא יראה כי השניים כלל מתקשרים.

הצד השולח יכול להעלות קובץ אל שרת קבצים, ובו מסר סמוי שברצונו להעביר את הגורם המקבל. והצד המקבל צריך פשוט להוריד את הקובץ משרת הקבצים ולפענח את המסר. פעולה זו יכולה להיעשות אף ברשת חברתית בה הצד השולח מעלה תמונה או קטע וידאו המכיל מסר סמוי, והצד המקבל פשוט מחלץ את המסר לוקאלית.

3.2. תכולות ופיצ'רים

הפרויקט StegoMaster מהווה תשתית יציבה וניתנת להרחבה של הטמנה וחילוץ מסרים סמויים מקבצים. התשתית פועלת בסביבת Windows x64 ומעל סט קבצים מצומצם נכון לעת כתיבת המסמך.

הפרויקט מספק למשתמש שתי פעולות עיקריות והן החבאת מסר בקובץ או חילוץ מסר בקובץ. תהליך החבאת מסר בקובץ מתואר באיור 2:



איור 2: תרשים זרימה להחבאת מסר בקובץ בתוכנת StegoMaster

כפי שניתן לראות בשלבים השני והרביעי באיור 2, על המשתמש לבחור באיזו שיטה הוא רוצה להחביא את המסר בקובץ. שלבים אלה כומסים בחובם את הצעדים שעושה התשתית:

1. זיהוי סוג הקובץ על ידי True Type
2. בדיקת סוגי השיטות המתאימות לסוג הקובץ
3. הפעלת השיטה על הקובץ

צעדים אלה חוזרים על עצמם גם בשלבי הפענוח של המסר, כפי שניתן לראות באיור 3.

איור 3: תרשים זרימה לחילוץ המסר הסמוי בתוכנת StegoMaster

יש לשים לב שבשלב השני של איור 3 יש לבחור את שיטת הסטגונגרפיה שנעשה בה שימוש בשלב ההחבאה – אחרת StegoMaster תחפש את המסר במקום בו הוא לא קיים.

היישום אינו מהווה כלי תקיפה בסייבר, ואין כנה להריץ אותו על כל מחשב שהוא.

3.3. דרכים להרחבה

כפי שצוין לעיל, הפרויקט נכתב בראיה לעתיד כתשתית ניתנת להרחבה. כלומר, התשתית כולה בנויה על טעינה דינאמית של ספריות DLL לצורך התכולות:

- זיהוי סוג הקובץ – TrueType
- החבאת המידע באופן סטגנוגרפי

לכן, על מנת להרחיב את התשתית, אין צורך לערוך את הקוד של הפרויקט, אלא לממש מנשק (interface) המוגדר בקובץ header של הפרויקט, ולהוסיף את הספרייה הדינאמית המתקבלת לתיקיה המתאימה.

כך, מפתח יכול להרחיב את תשתית StegoMaster כך שתתמוך בסוגי קבצים נוספים ובשיטות סטגנוגפריה נוספות בקלות ובלי להיכנס לקוד התשתית המקורי.

3.3.1. פורמטים נתמכים

רשימת הפורמטים הנתמכים על ידי תשתית StegoMaster נכון לעת כתיבת מסמך זה הינה:

1. Jpeg

2. Bitmap
3. Wave
4. Zip

4. שיטות סטגנוגרפיה המומשות בפרויקט

בסעיף זה יוצגו שיטות הסטגנוגרפיה המוצעות על ידי תשתית StegoMaster נכון לעת כתיבת מסמך זה.

4.1. End of File

שיטה זו מסתמכת על הפער האבטחתי בתוכנות שמטרתן לפתוח קבצים, שהוא לא לקרוא את כל קובץ הקלט. לכל פורמט יש מבנה מסוים שהוא אמור להיכתב לפיו, סדר של נתונים וחוקים מסוימים, אך מה קורה כאשר אותם חוקים טוענים שהגיע סוף הקובץ, אך הקובץ לא באמת נגמר?

ישנן תוכנות שכאשר הן רואות את סוף הקובץ על פי הפורמט, הן מפסיקות לקרוא את הקלט ומציגות את הקובץ. עבור אותם פורמטים, ניתן לכתוב מסרים סמויים בסוף הקובץ, לאחר "שנגמר הפורמט" וכך המידע הנ"ל לא יוצג.

שיטה זו אמנם פשוטה אך יעילה. בשיטה זו נעשה שימוש בתקיפת Duqu שמתוארת ב-[12], תקיפה שהתרחשה בשנים 2014-2015 שבין היתר תקפה גורמים המקושרים לתוכנית הגרעין האיראנית. התקיפה השתמשה בשיטת End of File על גבי קבצי Jpeg כדי להזליג מידע חזרה אל התוקף.

4.1.1. פורמטים נתמכים

1. Jpeg – בפורמט Jpeg ישנה תגית הנקראת EOF, המייצגת את End Of File, שלאחריה תוכנות רבות מפסיקות לקרוא את הקובץ. לכן, StegoMaster רושמת את המידע הסמוי אחרי תגית זו.
2. Bitmap – בקובץ זה, כחלק מאחד ה-headers, רשום גודל התמונה כגודל מטריצה – גובה על רוחב. לאחר מכן, כתובים הפיקסלים המתאים את אותה המטריצה. משום שגודלה של המטריצה ידוע מראש, התוכנה המציגה את תמונה יודעת מתי להפסיק לקרוא את המידע. לכן, מיד לאחר הפיקסלים, ניתן לכתוב את המסר הסמוי.

4.1.2. פרטים טכניים

המסר הסמוי שמוחבא בסוף הקובץ מורכב מ-3 חלקים:

1. Magic – רצף בתים קבוע שנועד לזהות בסבירות גבוהה שקיים מסר שהוחבא על ידי StegoMaster. במקרה זה הרצף הוא הקידוד ב-ASCII של המחרוזת 'stgo'
2. אורך המסר – מקודד אל תוך הקובץ על מנת שנוכל לדעת כמה בתים עלינו לקרוא. אורך המסר מקודד כ-Unsigned Int 32 bit.
3. תוכן המסר הסמוי

כאשר, 3 הוא מוגדר על ידי המשתמש והוא בעצם המסר אותו מקליד המשתמש ב-GUI. והחלקים כתובים בסדר הפוך בקובץ, כלומר, ה-Magic יופיע ממש בסוף הקובץ, לפניו אורך הקובץ ולפניו המסר הסמוי.

4.2. Jpeg COM

Joint Photographic Expert Group (JPEG) הוא סטנדרט לכיווצי תמונה, כלומר האלגוריתם המתמטי, בעוד ש-JPEG File Interchange Format (JFIF) הוא פורמט קבצי תמונה. סיומת קבצי JFIF היא לעתים jpeg ולעתים jpg אך את שניהם מבוטאים Jpeg, דבר שיוצר בלבול בין שם הקובץ לשם הכיווץ. בעבודה זו אתייחס לעתים ל-JPEG בתור פורמט הקובץ לשם הנוחות.

4.2.1. רקע על JFIF

בסעיף זה אתאר את המבנה של קובץ JFIF בקצרה כפי שתואר ב- [5]. יש לציין כי לפורמט זה הרחבות רבות שלא יתוארו כאן, וכמו כן מספר תכונות ושדות שאינן רלוונטיות לעבודה או לקריאה בסיסית של הקובץ.

4.2.1.1. מבנה כללי

קובץ JFIF בבסיסו בנוי מסגמנטים (Segment), כאשר כל סגמנט מכיל מידע שונה אודות הקובץ. את תחילת כל סגמנט פותח מרקר (Marker) בגודל שני בתים. ערך הבית הראשון של המרקר הוא תמיד 0xFF וערך הבית השני מייצג את סוג המרקר.

הקובץ נפתח תמיד במרקר הנקרא Start Of Image (SOI) ונגמר במרקר End Of Image (EOI). מרקרים אלה הם מרקרים ללא סגמנט ומטרתם היא לציין את תחילת/סוף התמונה בלבד.

לרוב, לאחר המרקר, יופיע אורך הסגמנט הנכתב ב-2 בתים בקידוד Big-Endian. כך, ניתן "לרפרף" על חלקים מן הקובץ מבלי לקרוא את הסגמנטים עצמם, משום שלאחר קריאת האורך ניתן לדלג ישר אל המרקר הבא וכך הלאה. יש לשים לב כי אורך הסגמנט כולל את שדה האורך (2 בתים) אך לא כולל את המרקר עצמו.

4.2.1.2. מרקרים עיקריים

בטבלה המוצגת מטה ניתן לראות את הסגמנטים העיקריים בקובץ Jpeg.

שם המרקר	מזהה	תיאור
SOI	0xd8	**Start of Image, מסמן את תחילת התמונה**
APP0	0xe0	**JFIF application segment, המרקר הפותח את הסגמנט הראשון בקובץ ומכיל פרטים כלליים על התמונה**
APPn	0xe1 – 0xef	**Other APP segments, מרקרים שנועדו לשימוש אפליקציות שונות. לדוגמה, המזהה של מרקר APP14 הוא 0xee והוא מייצג מרקר של Adobe.**
DQT	0xdb	**Quantization Table, מכיל את טבלת הקוונטיזציה**
SOF0	0xc0	**Start of Frame, מכיל מידע על Frame ספציפי – לעתים יש גם SOF1 שהמזהה שלו הוא 0xc1**
DHT	0xc4	**Huffman Table, מכיל את טבלת ה-Huffman לצורך פתיחת הכיווץ**
SOS	0xda	**Start of Scan, מכיל את ה-data של הקובץ**
EOI	0xd9	**End of Image, מסמן את סוף התמונה**

COM Segment .4.2.2

כמו להרבה פורמטים, גם לפורמט JFIF ישנה דרך לכתוב בתוכו הערות. סגמנט COM הוא סגמנט הנועד על מנת לכתוב בתוכו הערות, לרוב השימוש בסגמנט נעשה על מנת לכתוב את שם הצלם של התמונה או החברה שממנה התמונה הגיעה וכו' המידע הנמצא בסגמנט אינו מוגבל ואף אינו חייב להיות printable. מבנה הסגמנט כדלקמן:

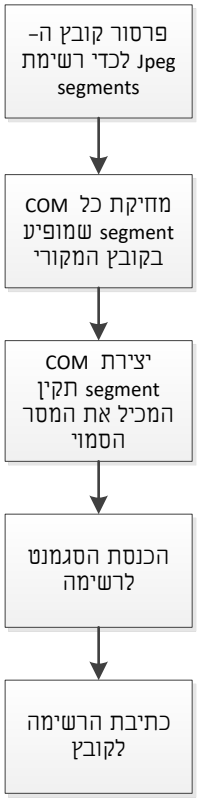
שם השדה	גודל	תיאור מילולי
Marker Identifier	2 Bytes	ערך קבוע של 0xfffe
Length	2 Bytes	אורך הסגמנט (לא כולל ה-Marker Identifier)
Comment	Non-constant	הערה בעלת תוכן חופשי באורך של Length -2

שדה ה-Length כאמור יכול להיות גדול כרצוננו ולכן ערוץ המידע גדול כרצוננו.

ערוץ מידע זה קל למימוש אך גם קל לראות שהוא קיים, בנוסף יש לזכור כי סגמנט ה-COM הינו חלק מן הפורמט, ולכן אינו חשוד, כלומר, אם נחביא מסר מוצפן תחת סגמנט COM לא יהיה ניתן להגיד בוודאות שמדובר במסר סטגנוגרפי. עם זאת, סביר להניח כי גורמים הנזהרים מתכנים סטגנוגרפיים יסירו את סגמנט ה-COM.

4.2.3. פרטים טכניים

כדי לממש שיטה סטגנוגרפית זו, תוכנת StegoMaster יודעת לפרסר קובץ Jpeg לרשימת סגמנטים. וככתוב, לכתוב רשימה של סגמנטים לכדי קובץ Jpeg. בהינתן היכולות האלה תרשים הזרימה באיור 4 מתאר את פעולת ההחבאה.



איור 4: תרשים זרימה לתיאור החבאת מסר סמוי ב-COM segment

לכן, לאחר תהליך זה, יתקבל קובץ Jpeg כפלט, ובו הערה אחת בלבד – זו המכילה את המסר הסמוי. החיסרון בשיטת יישום זו הוא כמובן שלא ניתן להשתמש במתודה זו באופן סדרתי, אך אם נקרא את הערות הקובץ המקורי ונכתוב אותן אל קובץ הפלט – נפתור גם את בעיה זו.

4.3. Bitmap Gap

שיטה זו מתבססת על ניצול דרגת חופש שניתנת לנו על ידי פורמט Bitmap. על מנת להבין את שיטה זו לעומק, כמובן נצטרך לסקור את פורמט Bitmap.

4.3.1. פורמט Bitmap

בסעיף זה אפרט את מבנה קובץ ה-Bitmap באופן מפורט, לרמת ה-Byte כפי שמתואר ב- [6], אך מכיוון שזהו אינו מיקוד העבודה, אתמקד בתכונות ושדות עיקריים בקובץ.

4.3.1.1. Bitmap Header

קובץ ה-Bitmap תמיד יתחיל ב-Header המכיל את השדות הבאים:

שם השדה	גודל	תיאור מילולי
Header Field	2 Bytes	מאין Magic Number שמסמל שאכן מדובר בקובץ BMP, לרוב ערכו יהיה BM (בקידוד Ascii)
File Size	4 Bytes	מתאר את גודל הקובץ בבתים
Reserved	2 Bytes	ערכו צריך להיות 0
Reserved	2 Bytes	ערכו צריך להיות 0
Data Offset	4 Bytes	ההיסט בו נמצאת המטריצה המכילה את ערכי הפיקסלים

4.3.1.2. DIB Header

לאחר התחילית שראינו קודם (Bitmap Header) ישנו Header נוסף שנקרא Device Independent Bitmap Header (DIB). והוא מתאר את תכונות הקובץ התלויות בתוכנה היוצרת, ולכן יש מספר סוגי DIB Headers. הסוג שיתואר בעבודה זו הוא BITMAPINFOHEADER שבו נעשה שימוש ברוב מערכות הWindows (החל מ-Windows NT 3.1).

כל DIB Header, מכל סוג שהוא, מתחיל בשדה בן 4 בתים המתאר את גודלו. לכל סוג של DIB Header יש גודל שונה, וכך התוכנה המציגה את התמונה יודעת להבחין בסוג ה-DIB Header הנכון למרות ששמו לא כתוב במפורט בשום מקום בקובץ.

להלן תיאור ה-BITMAPINFOHEADER:

שם השדה	גודל	תיאור מילולי
Size	4 Bytes	גודל ה-DIB Header, עבור סוג BITMAPINFOHEADER שדה זה חייב להיות בעל ערך 40
Width	4 Bytes	רוחב התמונה בפיקסלים (יכול להיות שלילי – אומר שסדר הפיקסלים בתמונה הוא מימין לשמאל ולא להפך)
Height	4 Bytes	גובה התמונה בפיקסלים (יכול להיות שלילי – אומר שסדר הפיקסלים בתמונה הוא מלמעלה למטה ולא להפך)
Number Of Color Plains	2 Bytes	חייב להיות 1
Color Depth	2 Bytes	עומק הסיביות של התמונה – 1, 4, 8, 16, 24 או 32
Compression Method	4 Bytes	לרוב יהיה BI_RGB, כלומר הערך 0 – ללא כיווץ
Image Size	4 Bytes	גודל ה-Data בתמונה (בתמונות לא מכווצות, ערך זה יכול להיות 0 משום שניתן להסיק אותו מהערכים האחרים)
Horizontal Resolution	4 Bytes	יחס אופקי של פיקסל למטר
Vertical Resolution	4 Bytes	יחס אנכי של פיקסל למטר
Number Of Colors in Palette	4 Bytes	יכול להיות כל ערך בין 0 לבין $2^{biBitCount}$. (הערך 0 אומר גודל ברירת מחדל)
Number of Important Colors	4 Bytes	לרוב מתעלמים מערך זה – 0 אומר שכל הצבעים חשובים

Color Palette .4.3.1.3

לפי תיאור הקובץ המתואר לעיל, בו סוג הכיווץ הוא BI_RGB (כלומר ה-data אינו מכווץ), לאחר ה-DIB Header עשויה לבוא פלטה (Palette) שתחזיק את טבלת הצבעים לתמונה במקרה ועומק הסיביות הוא 1,4 או 8. מספר הצבעים בפלטה הוא כפי שצוין ב-DIB Header ועשוי להיות כל מספר בין 1 לבין $2^{biBitCount}$. כל תא בפלטה מכיל את הערכים (משמאל לימין):

Blue, Green, Red, Alpha

כאשר כל אחד מהערך מתואר על ידי byte בודד, והערך Alpha הוא 0x00 קבוע (לכל צבע בפלטה).

Gap .4.3.1.4

לאחר הפלטה עשוי להופיע רווח, "ערכי זבל" של בתים, המפריד בין ה-Headers ל-data. לערכים אלה אין שום משמעות ולכן כל תוכנה הקוראת קבצי Bitmap מדלגת על ערכים אלה באמצעות ה-offset שמופיע ב-Bitmap Header המורה היכן מתחיל ה-data ה"מעניין".

בשיטה זו, תוכנת StegoMaster כותבת את קובץ ה-Bitmap מחדש כאשר ה-Gap מוגדל לפי צרכי הסטגנוגרפיה, כלומר באורך ההודעה שברצונינו להכניס.

Pixel Matrix .4.3.1.5

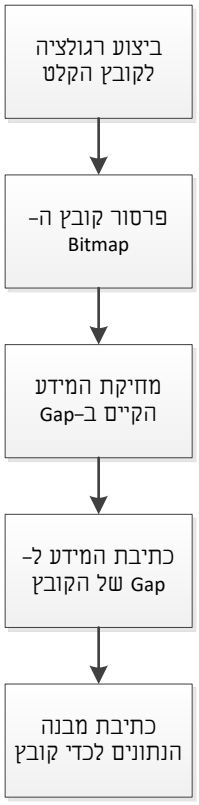
לאחר ה-Gap, ובהיסט כפי שתואר ב-Bitmap Header, נמצאת מטריצה של פיקסלים, הפיקסלים יכולים להיכתב לפי ערכיהם או כמצביעים לפלטה שהוזכרה קודם.

4.3.2. פרטים טכניים

כדי לממש שיטה סטגנוגרפית זו, תוכנת StegoMaster יודעת לפרסר קובץ Bitmap בעל BITMAPINFOHEADER כפי שתואר בסעיף 4.3.1.2 ובעל עומק של 24 סיביות. לכן, על StegoMaster לבצע רגולציה לכל קובץ Bitmap בו היא פוגשת. הרגולציה נעשית באמצעות תוכנת Open Source ששמה ImageMagick [7], העוסקת בקריאה והמרה של קבצים ובעיקר תמונות. השימוש ב-ImageMagick נעשה על ידי שורת הפקודה:

```
Convert.exe <original_file_path> –type truecolor BMP3:<converted_file_path>
```

כמו כן, תוכנת StegoMaster יודעת לכתוב את מבנה ה-Bitmap שפרסרה לכדי קובץ. בהינתן היכולות האלה תרשים הזרימה באיור 5 מתאר את פעולת ההחבאה:



איור 5: תרשים זרימה להחבאת מידע ב-Gap של קובץ Bitmap

כך ניתן לראות שתוכנת StegoMaster תומכת במגוון גרסאות של Bitmap, הודות ל-ImageMagick, ובכולן היא יכולה להחביא מידע ב-Gap. אך יש לשים לב שהפלט של StegoMaster תמיד מאותו סוג, דבר שלעתים עלול להגדיל את נפח הקובץ המקורי, גם אם המסר קצר.

Bitmap LSB .4.4

בסעיף זה בניגוד ל-4.3, החבאת המידע נעשית ב-Data של קובץ ה-Bitmap ולא ב-metadata. לכן, על מנת לתאר שיטה זו באופן מיטבי, נצטרך לסקור חלקים נוספים בקובץ ה-Bitmap, שלא הוצגו ב-4.3.1.

4.4.1. תיאור ה-Data של קבצי Bitmap

בסעיף זה נתאר את החלק הארי של הקובץ – ה-data עצמו (או הצבעות אל הפלטה אם קיימת, אך גם הצבעות שכאלה נחשבות data). הפיקסלים בתמונה מחולקים לשורות. מספר השורות הוא ערך מוחלט של השדה גובה התמונה שנמצא ב-DIB Header שתואר קודם לכן. לכל שורה בתמונה מוסיפים ריווח (Padding) כך שגודל השורה בבתים יתחלק ב-4 ללא שארית. כך, גודל כל שורת פיקסלים בתמונה בבתים מתואר על ידי הנוסחה הבאה:

$$RowSize = \left\lfloor \frac{BitsPerPixel \cdot \mathrm{Im}ageWidth + 31}{32} \right\rfloor \cdot 4$$

לכן, גודל כל ה-data בתמונה מתואר על ידי: $PixelDataSize = RowSize \cdot |\mathrm{Im}ageHeight|$

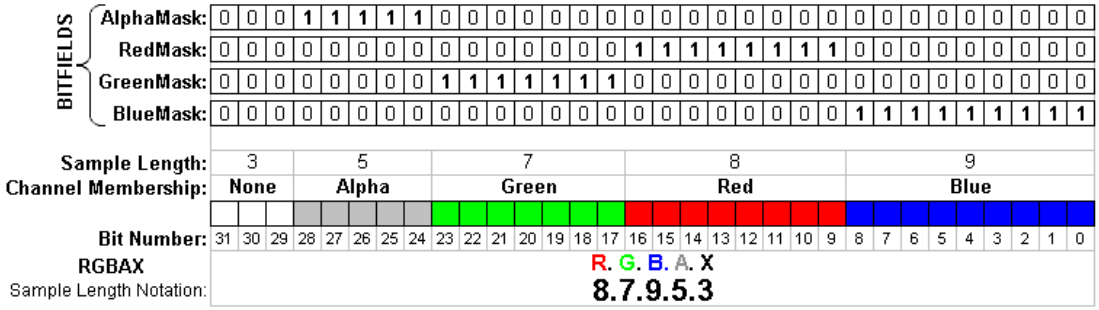
לרוב, סידור הפיקסלים של התמונה מתחיל מהפינה השמאלית התחתונה שלה, כלומר, הפיקסל הראשון שכתוב בתמונה הוא הפיקסל שמוצג בפינה השמאלית התחתונה. לאחר מכן, כיוון ההתקדמות הוא בהמשך השורה (לכיוון ימין), ולאחר מכן למעלה. כך שהפיקסל האחרון שמקודד בקובץ הוא הפיקסל המופיע בפינה הימנית העליונה של התמונה. הנאמר בפסקה זו עשוי להשתנות אם שדות הגובה והרוחב של התמונה כפי שמוצגים ב-header הינם שליליים (כפי שנאמר לעיל ב-4.3.1.2).

BitFields .4.4.1.1

ייצוג פיקסל בתמונת Bitmap יכול להיעשות באופנים שונים, וכך ניתן לייצג פיקסל בדיוקים שונים. נניח ואנו מייצגים פיקסל על ידי 32 סיביות, איך נדע אילו מהסיביות שייכות לאיזה ערוץ צבע?

לצורך כך, נמצא שדה ה-BitField אשר מכיל mask שאומר כיצד יש לקרוא כל פיקסל, כלומר איזה ביטים שייכים לאיזה ערוץ צבע.

לדוגמה באיור 6 הלקוח מתוך [6] בו ניתן לראות masks עבור תמונה עם עומק סיביות של 32.



איור 6: תיאור חלוקה של פיקסל לערוצי צבע

ה-mask לכל ערוץ צבע (כולל alpha) מופיע לפי הסדר Alpha, Red, Green, Blue (משמאל לימין), אך הסיביות המייצגות כל ערוץ צבע יכולות לשנות את הסדר – כמו בדוגמה שראינו בה הביטים של הערוץ הירוק יופיעו לפני הביטים של הערוץ האדום. ה-masks לא יכולים לעלות אחד על השני (מצב זה נקרה overlapping).

הדוגמה לעיל מתארת תמונה בעלת BitFields ועומק סיביות של 32, אך ניתן להשתמש בשדה זה גם עבור תמונות בעלי עומקי סיביות אחרים.

4.4.2. תיאור השיטה

שיטה סטגנוגרפית זו מבוססת על [2], [1] ועל [3] והיא, בניגוד לשיטות שנדונו עד כה, מחביאה את המסר הסמוי ב-data של הקובץ ולא ב-meta-data.

עיקר שיטה זו הוא להחביא חלקים קטנים מהמסר בכל אחד ממרכיבי התמונה (פיקסלים) באופן כזה שלא נראה לעין. כך, בעת קריאת התמונה ניתן לחלץ מאותם פיקסלים את המידע ולחברו יחדיו לכדי מסר שלם.

את השיטה ניתן לממש כך שבתוך כל byte של data מחביאים כל כמות של סיביות (עד 8 כמובן), כאשר שככל שמחביאים פחות – הערוץ אינו נראה לעין, וככל שמחביאים יותר כך נפח הערוץ הסמוי גדל. במימוש StegoMaster בחרתי להחביא סיבית אחת בכל byte של data.

יש לשים לב ש**לכאורה** השינויים שנעשים לקובץ בשיטה זו יהיו נראים לעיניי המתבונן, אך כאשר מדובר בסיבית אחת של שינוי בכל byte של data מקורי השינוי הוא זעיר מאוד ואינו נראה לעין כלל.

4.4.3. פרטים טכניים

כפי שתואר ב-4.3.2, הצעד הראשון בקריאת תמונת Bitmap הוא לנרמל אותה לעומק סיביות 24, לכן, מעתה ואילך, כל ההתייחסויות לתמונה יעשו רק עבור תמונות בעלות עומק סיביות של 24.

בהינתן קובץ Bitmap כקלט, תשתית StegoMaster תבצע את הצעדים המתוארים באיור 7.

איור 7: תרשים זרימה להחבאת מידע ב-LSB של הפיקסלים של תמונת Bitmap

כפי שניתן לראות באיור 7, בשיטה סטגנוגפית זו נמצא שלב נוסף באלגוריתם והוא קידוד המידע. המידע כאמור מגיע אלינו כרצף בתים, ואנו רוצים לפצלו, כך שלכל byte של data נקודד בדיוק סיבית אחת.

אל המסר עצמו, מתלווים מחרוזת magic (שערכה הוא 'Stego' בקידוד ASCII) ואורך תוכנו של המסר הסמוי בבתים (האורך מקודד על ידי unsigned integer בגודל 32 bit).

חסרון בולט בשיטה זו הוא שהמסר הסמוי מוגבל בגודלו על ידי כמות ה-data בתמונה. לדומה, במימוש של StegoMaster אורך המסר הסמוי מוגבל ל-$\frac{3x}{8}$ כאשר x הוא מספר הפיקסלים בתמונה המקורית. (בספירה הזנחתי את אורך ה-magic ואת קידוד אורכו של המסר).

4.5. Wave Additional Chunk

בסעיף זה נציג שיטה סטגנוגרפית הפועלת על קבצי Audio מסוג Wave (או WAV). קבצי Wave שייכים למשפחת הפורמטים RIFF. שיטה זו עושה ניצול לדרגות חופש המאופשרות לנו על ידי מבנה RIFF המתירני. על מנת להבין טוב יותר את השיטה נדרשת הקדמה קלה על הפורמט כפי שמתואר ב- [8].

4.5.1. רקע על פורמט Wave

פורמט Resource Interchange File Format או בקיצור RIFF הוא פורמט הנועד לאחסן מידע. הגדרת הפורמט היא כללית ולא במקרה, במשפחת פורמטים אלה נמצאים פורמטי Audio, וידאו, תמונות וכו'.

המבנה העיקרי בבסיס פורמט RIFF הוא ה-chunk והוא מאין אוגדן מידע מכל סוג שהוא. מבנה ה-chunk כדלקמן:

שם השדה	גודל	תיאור מילולי
ckID	4 Bytes	מזהה של ה-chunk (4 תווים ב-ASCII)
ckSize	4 Bytes	גודל ה-chunk בבתים
ckData	<ckSize> Bytes	ה-data עצמו שמחזיק ה-chunk

סידור ה-chunks הוא היררכי, כלומר, ה-chunks בקובץ מכילים אחד את השני בצורת עץ.

ישנו chunk עיקרי שחוזר על עצמו במספר קבצי RIFF והוא ה-Format Chunk. עבור chunk זה מתקיים:

1. ckID = 'RIFF'
2. ה-4 בתים הראשונים ב-data של ה-chunk הם ב-ASCII והם מייצגים את סוג הפורמט.

לרוב ה-Format Chunk הוא ה-chunk הראשון והעיקרי בקובץ (שורש העץ), כלומר, כל שאר ה-chunks מוכלים בו.

ישנם עוד chunks שעשויים להופיע בקבצי RIFF כלליים אך ה-Format Chunk הוא העיקרי בהם ולכן עבודה זו תתאר אותו בלבד – לשם סקירה נרחבת יותר של שאר ה-chunks מומלץ להסתכל במפרט עצמו [9].

4.5.2. מבנה פורמט Wave

בסעיף זה אתאר את מבנה קובץ סטנדרטי של Wave, כלומר אתמקד בפריטים הנפוצים שנמצאים בקבצי Wave ולאו דווקא ארחיב בתיאור על חלקים שהם אופציונאליים או בשימוש שאינו נרחב.

לרוב ה-data של קבצי Wave מקודד בקידוד Pulse Code Modulation או בקיצור PCM. קידוד זה הוא העברה של גלי הקול (האנלוגיים) אל העולם הדיגיטלי. כלומר, ניקח גל קול נתון ונדגום את האמפליטודה שלו במרווחי זמן קבועים. את התוצאה נעגל (קוונטיזציה) וכך נקבל את הייצוג של הגל כאות דיגיטלי. קידוד PCM הוא הנפוץ ביותר בקרב קבצי Wave ונדיר לראות קובץ שה-data שלו מקודד בקידוד אחר, לכן בעבודה זו אתמקד בקבצי Wave שה-data שלהם מקודד בקידוד PCM.

כמו שצוין בסעיף הקודם, קובץ Wave נפתח ב-Format Chunk כאשר ה-4 בתים הראשונים ב-data שלו הם המחרוזת ב-ASCII: "WAVE" (ללא null-terminator). ולאחר מכן ישנם chunks שהם בנים של ה-Format Chunk.

תחת ה-Format Chunk נמצא את ה-fmt chunk, את ה-data chunk וכו'. ה-chunks בנויים במבנה שתואר קודם, כך שקודם כל מופיע שם ה-chunk, לאחר מכן אורכו ולבסוף data חופשי של ה-chunk כאשר המגבלה היחידה עליו היא האורך.

4.5.3. תיאור השיטה

השיטה מנצלת תכונה חשובה מאוד של פורמט Wave, והיא הרצון להתחדש. פורמט Wave הוא פורמט שניתן להרחבה, כלומר ניתן להוסיף לו chunks חדשים – ומדי פעם, כאשר הפורמט מתעדכן, גם נוספים כאלה.

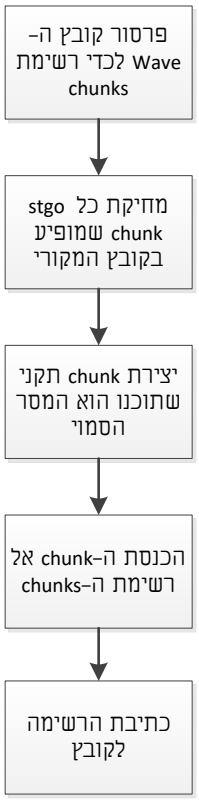
לכן, קוראי קבצי Wave יודעים שכאשר הם נתקלים ב-chunk שאינם מכירים, זה לא אומר שהקובץ לא תקין – אלא מניחים כי הוא מגרסה עדכנית יותר. לכן, הם מדלגים על ה-chunk הזה ומנסים לנגן את הקובץ מבלי להתייחס אליו כלל.

בשיטה זו הוספנו לקובץ chunk חדש המכיל את המידע הסמוי. כאמור, ניגון קבצי ה-Wave לא אמור להשתנות כלל.

4.5.4. פרטים טכניים

תשתית StegoMaster יודעת לקרוא קבצי Wave לכדי רשימה של chunks, ועבור chunks עיקריים אף לקרוא את ערכי שדותיהם. כמו כן, התוכנה גם יודעת לקבל רשימת chunks ולכתוב אותם לכדי קובץ Wave בר נגינה.

בהינתן היכולות המתוארות לעיל, סכימת פעולת StegoMaster מתוארת באיור 8.



איור 8: תרשים זרימה לתהליך החבאת מידע ב-chunk של קובץ Wave

שלב 3 באיור 8 מתאר יצירת של chunk בשם "stgo", וכפי שניתן לראות, בסוף תהליך זה יתקבל קובץ Wave בעל stgo chunk יחיד. החיסרון בשיטה זו הוא שלא ניתן להשתמש בה באופן סדרתי, כלומר, לא ניתן להחביא בה 2 מסרים אחד אחרי השני – משום שההחבאה השנייה תמחק את ההחבאה הראשונה. אך חסרון זה הוא החלטת מימוש בלבד וכמובן שאפשר לממש אחרת, כך שדבר זה יהיה אפשרי.

4.6. Wave LSB

בסעיף זה בניגוד ל-4.5, החבאת המידע נעשית ב-Data של קובץ ה-Wave ולא ב-metadata. לכן, על מנת לתאר שיטה זו באופן מיטבי, נצטרך לסקור חלקים נוספים בקובץ ה-Wave, שלא הוצגו ב-4.5.1.

4.6.1. תיאור ה-Data של קבצי Wave

בתוך ה-Format Chunk שתואר בסעיף 4.5.2, מופיע ה-data chunk המכיל בתוכו את ה-data של הקובץ. מבנה ה-chunk פשוט ומתואר בטבלה הבאה:

שם השדה	גודל	תיאור מילולי
ckID	4 Bytes	ערכו הוא "data" בקידוד ASCII
ckSize	4 Bytes	גודל ה-chunk בבתים
Sampled data	<ckSize> Bytes	ה-samples של קובץ השמע (מלווים ב-padding byte במקרה הצורך)

אם אורך ה-data הוא אי זוגי, בסוף ה- data chunk יופיע byte נוסף לצרכי padding.

ה-Sampled Data שמצוין בטבלה לעיל הוא הדגימות ה-data, כלומר האמפליטודות של גלי הקול. ה-Sampled Data מסודר בלוקים כך שכל בלוק מכיל nChannels דגימות. כלומר, סדר ה-data בקובץ הוא כך שבבלוק הראשון יופיעו הדגימה הראשונה של כל הערוצים, בבלוק השני הדגימה השנייה של כל ערוץ וכן הלאה. כך יכול הקובץ להתנגן במעבר אחד על ה-data משום שהוא מחולק לפי זמן.

4.6.2. תיאור השיטה

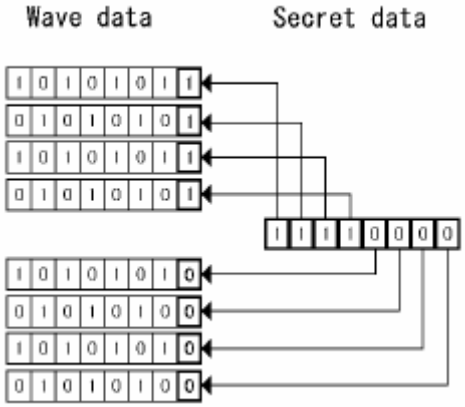
שיטה סטגנוגרפית זו מבוססת על אחת השיטות המתוארות ב- [4] והיא, בניגוד לשיטות שנדונו עד כה, מחביאה את המסר הסמוי ב-data של הקובץ ולא ב-meta-data.

עיקר שיטה זו הוא להחביא חלקים קטנים מהמסר בכל אחד ממרכיבי קובץ השמע (דגימות) באופן שאינו נשמע. כך, בעת קריאת הקובץ ניתן לחלץ מאותן הדגימות את המידע ולחברו יחדיו לכדי מסר שלם.

את השיטה ניתן לממש כך שבתוך כל דגימה מחביאים כל כמות של סיביות (עד 8 כמובן), כאשר שככל שמחביאים פחות – הערוץ פחות בולט ופחות נשמע, וככל שמחביאים יותר כך נפח הערוץ הסמוי גדל. במימוש StegoMaster בחרתי להחביא סיבית אחת בכל דגימת PCM.

יש לשים לב שלכאורה השינויים שנעשים לקובץ בשיטה זו יהיו חשופים וניתנים לגילוי, אך כאשר מדובר בסיבית אחת של שינוי בכל דגימה, השינוי הוא זעיר מאוד ואינו נראה נשמע כלל.

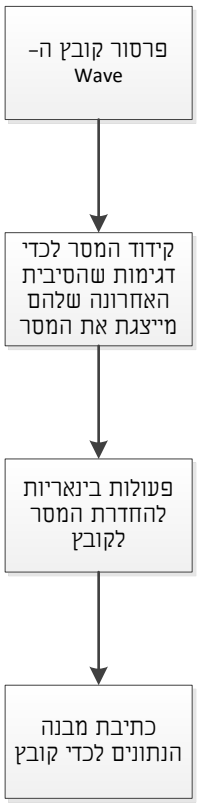
איור 9 הנלקח מתוך [4] מתאר החבאה של byte של מידע סמוי בתוך ה-data של קובץ Wave.



איור 9: החבאת מסר סמוי ב-LSB של דגימות Wave

4.6.3. פרטים טכניים

השלבים למימוש שיטה זו מתוארים באיור 10.



איור 10: קידוד מסר סמוי ל-LSB בדגימות קובץ Wave

כפי שניתן לראות באיור 10, בשיטה סטגנוגפית זו נמצא שלב נוסף באלגוריתם והוא קידוד המידע. המידע כאמור מגיע אלינו כרצף בתים, ואנו רוצים לפצלו, כך שלכל דגימה של PCM נקודד בדיוק סיבית אחת.

אל המסר עצמו, מתלווים מחרוזת magic (שערכה הוא 'Stego' בקידוד ASCII) ואורך תוכנו של המסר הסמוי בבתים (האורך מקודד על ידי unsigned integer בגודל 32 bit).

חסרון בולט בשיטה זו הוא שהמסר הסמוי מוגבל בגודלו על ידי כמות ה-data בתמונה. לדומה, במימוש של StegoMaster אורך המסר הסמוי מוגבל ל- $\frac{x}{8}$ כאשר x הוא מספר הדגימות בקובץ.

(בספירה הזנחתי את אורך ה-magic ואת קידוד אורכו של המסר).

4.7. Zip Beginning

4.7.1. על השיטה

שיטה זו מנצלת תכונה מעניינת בקריאת קבצי Zip. פורמט Zip בנוי כך שיש לקרוא אותו מהסוף אל ההתחלה. קריאה זו נעשית על ידי כך שלא מסתמכים מראש על מבנה הקובץ או מקומות לחפש ערכים בתוך הקובץ – אלא בכל פעם שברצוננו לקרוא מידע מהקובץ, רשום לנו בדיוק היכן למצוא אותו (היסט מתחילת הקובץ).

לדוגמה, בתוך קובץ ה-Zip הרי, כתובים כל הקבצים הפנימיים (באופן מכווץ). הגישה אליהם נעשית דרך offset. בסוף הקובץ נמצא ה-end-locator שאומר מה ההיסט של טבלת ה-entries, שמוסרים לנו meta-data אודות הקבצים המכווצים. כל entry מחזיק אצלו את ההיסט של הקובץ. כך ניתן לקרוא את הקובץ מסופו, משום שכל קריאה שהיא לא רציפה – נעשית על ידי היסט שכתוב בקובץ.

לכן, אם נכתוב בתחילת הקובץ את המסר הסמוי ונתקן את ה-offsets הכתובים בקובץ – תוכנה הקוראת את קובץ ה-Zip לא תראה את המסר, משום שאף offset לא מצביע עליו.

4.7.2. פרטים טכניים

תרשים הזרימה שמובא בפנינו באיור 11 מתאר את תהליך החבאת המסר בקובץ Zip.

איור 11: החבאת מסר סמוי בתחילת קובץ ה-Zip

תהליך פרסור קובץ ה-Zip כפי שכתוב באיור 11 נעשה כמובן מסוף הקובץ להתחלתו כמו קורא קבצי Zip אמתי. פרט זה חשוב מאוד לפעולת התוכנית משום שאם נרצה להחביא פעמיים מסר באותו הקובץ, בתחילתו – לא היינו יכולים לעשות זאת, משום ששלב הפרסור היה נכשל.

אל המסר הסמוי מתלווים בתחילת הקובץ גם:

1. רצף הבתים `0x50, 0x4B, 0x03, 0x04`
2. מחרוזת magic שערכה הוא "stgo" בקידוד ASCII
3. 4 בתים שמייצגים את גודל המסר הסמוי כ-Unsigned 32-bit Integer

רצף הבתים `0x50, 0x4B, 0x03, 0x04` מייצג magic של קובץ Zip שאומר שכאן מתחיל Zip Record, כלומר קובץ מכווץ. כתיבת הרצף הנ"ל חשובה משום שכך תוכנת StegoMaster מזהה כי מדובר בקובץ Zip (אם הוא מתחיל ברצף הבתים הנ"ל). לכן, אם נרצה להעביר ב-StegoMaster קובץ Zip שכבר עבר תהליך של הטמעת מסר בתחילת הקובץ – חשוב שרצף זה יהיה נוכח.

4.8. Zip Additional Record

על מנת להסביר את שיטה זו לעומק, נצטרך "לצלול" מעט יותר עמוק אל תוך פורמט Zip.

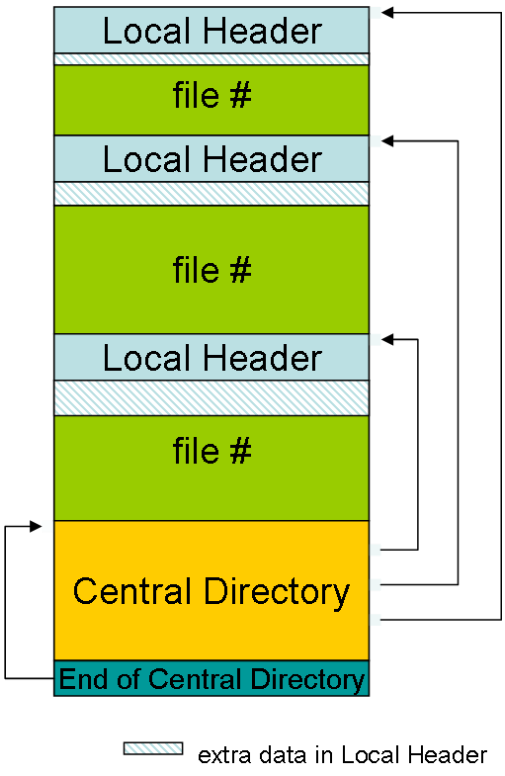
4.8.1. פורמט Zip

בסעיף זה אתאר את מבנה פורמט Zip בקצרה כפי שמתואר ב- [10] ולא אכנס לפרטי פרטים על כל מבנה ומבנה.

איור 12 (הנלקח מתוך [11]) מתאר את מבנה קובץ Zip. כפי שנאמר ב-4.7.1, על מנת לקרוא קובץ Zip באופן נכון יש לקרוא מהסוף להתחלה. בסוף הקובץ נמצא ה-End of Central Directory, שמצביע לתחילתה. כך אנו יכולים להגיע לתחילת ה-Central Directory.

ה-Central Directory הינה טבלה שבכל כניסה (entry) שלה נמצא אודות קובץ מכווץ שהיא מייצגת. כמו כן, היא מכילה את שם הקובץ ואת גודלו.

קורא קבצי Zip הולך לסוף הקובץ, קורא את ה-End of Central Directory, הולך בעקבות ה-offset המתואר בו אל ה-Central Directory, קורא את כולו – כך הוא יודע איך קבצים נמצאים ב-Zip ומה שמותיהם, ואת תוכנם הוא מחלץ מה-data המכווץ שנמצא ב-Record, אליו מצביעים ה-entries ב-Central Directory.



איור 12: מבנה קובץ Zip

4.8.2. על השיטה

שיטה סטגנוגרפית זו מנצלת את תהליך קריאת קובץ Zip שתואר בסעיף 4.8.1 בשביל להחדיר מסר סמוי אל תוך קובץ ה-Zip. בעצם התוכנה הפותחת את הקובץ מציגה את תתי הקבצים המופיעים בפנים לפי ה-Central Directory ולאו דווקא את הקבצים הנמצאים בארכיב.

כלומר, ניתן להוסיף קובץ לארכיב, וכל עוד הוא לא מופיע ב-Central Directory הוא יהיה "בלתי נראה" בעיניי התוכנה הפותחת את קובץ ה-Zip. בשיטה סטגנוגפית זו ניתן להחביא קבצים שלמים בארכיביי Zip, אך כמובן ב-StegoMaster מתמקדים בהחבאת מחרוזות בקבצים – לכן אנו מחביאים "קובץ מדומה".

4.8.3. פרטים טכניים

כפי שצוין ב-4.8.2 שיטה זו נועדה כדי להחביא קבצים שלמים בתוך Zip, אך אנו עושים בה שימוש להחבאת מסר סמוי (מחרוזת) שנקלט מן המשתמש. לכן עלינו ליצור קובץ מדומה שיכיל את המסר הסמוי, וכן, להוציא את המסר מן הקובץ המדומה.

הקובץ שאנו מחביאים ב-StegoMaster אינו מכווץ וזאת על מנת שנוכל לראות אותו בבהירות לשם הדו"ח. כמובן שבמימוש אחר – ניתן לכווץ גם את המסר הסמוי כך שהוא גם יהיה פחות בולט לעין.

איור 13: תהליך החבאת מידע בקובץ Zip על ידי הוספת Record

איור 13 מתאר לנו את תהליך הכנסת הקובץ עם המסר המוסתר אל קובץ ה-Zip. חשוב לזכור לעדכן את ה-offsets בקובץ בשביל שהוא יוכל להיפתח לאחר מכן.

במימוש ב-StegoMaster הקובץ הנוסף הוא "stego.file" כך ניתן לזהותו בקלות בעת החילוץ.

5. תוצאות

בסעיף זה אפרט את תוצאות הפרויקט StegoMaster ועבור כל אחת משיטות הסטגנוגרפיה שהצגתי אגע בתכונות הבאות:

- נפח הערוץ הסמוי – מהו גודל המידע שניתן להעביר על ידי שימוש בשיטה
- איכות הערוץ לעין "בלתי מיומנת" – האם הערוץ נחשף בשימוש סביר בקובץ
- איכות הערוץ הסמוי בבדיקה מעמיקה יותר – בהסתכלות על הקובץ כתוכן בינארי

5.1. End of File

שיטה זו היא שיטה כללית שאינה תלויה בסוג הקובץ, אך בפרויקט StegoMaster נמדוד שיטה זו עבור תמונות מפורמטיי Jpeg ו-Bitmap בלבד.

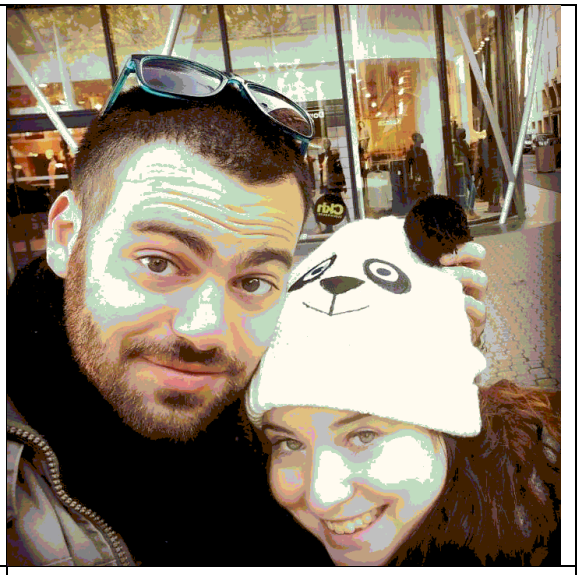
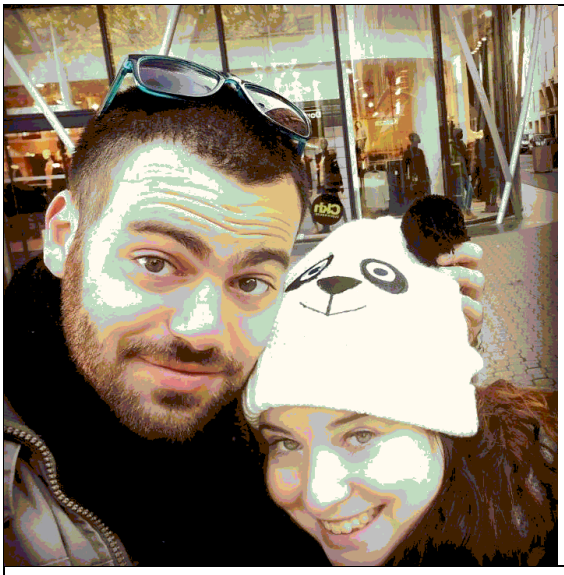
5.1.1. תוצאות ויזואליות

להלן תמונות מהפורמטים לעיל, לפני ואחרי תהליך שילוב הערוץ הסמוי בהם. נביט תחילה בתמונות מפורמט Jpeg:

תמונת ה-Jpeg המקורית	תמונת ה-Jpeg עם המסר הסמוי

טבלה 1: השוואה ויזואלית של תמונות Jpeg בשיטה End of File

בדוגמה לעיל הוחבא המסר "This is a hidden message" וכפי שניתן לראות בטבלה 1 לעיל, כאשר מיישמים שיטה זו בתמונות Jpeg, לא ניתן לראות א המסר הסמוי כלל. כעת נביט בטבלה 2 המציגה תמונות מפורמט Bitmap שבהן הוחבא אותו המסר.

	
תמונת ה-Bitmap המקורית	תמונת ה-Bitmap עם המסר הסמוי

טבלה 2: השוואה ויזואלית של תמונות Bitmap בשיטה End of File

גם בתמונת ה-Bitmap ניתן להבחין כי לא רואים שינוי בעקבות החבאת המסר הסמוי. הסיבה לכך שלא רואים שינויים בתמונה לאחר החבאת המסר היא שהמסר הוא אינו חלק מן הפורמט, אלא נכתב בסוף הקובץ – התוכנה המציגה את הקובץ לא קוראת כלל את האזור הזה בקובץ, ולכן לא מציגה את הקובץ באופן שונה.

יחד עם המסר הסמוי, ניתן להחביא גם את אורכו כך שהמקבל של התמונה ידע לקרוא קודם את אורכו ורק לאחר מכן את המסר. לדוגמה, אם נחליט שארבעת הבתים האחרונים ייצגו את גודל המסר x – נוכל לדעת מהיכן להתחיל לקרוא את המסר האמיתי, $x+4$ בתים לפני סוף הקובץ.

5.1.2. תוצאות בינאריות

אם נסתכל בקבצי ה-Jpeg בעורך הבינארי 010editor [12] נוכל לראות בבירור את המסר הסמוי בסוף הקובץ, כפי שניתן לראות באיור 14.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
1:60B0h:	31	EE	56	81	A2	0E	4A	CB	A7	DC	06	A1	60	B8	B2	38	1îV.¢.JË§Ü.¡`¸²8
1:60C0h:	60	A0	F0	45	AB	A8	F1	50	76	CB	AC	45	C4	BA	83	85	` ðE«¨ñPvË¬EÄº…
1:60D0h:	7E	25	A8	B2	EA	0B	87	72	ED	AF	42	31	94	B5	E2	3A	~%¨²ê.‡rí¯B1"µâ:
1:60E0h:	1C	C7	64	94	C7	CC	34	AE	65	E5	6D	8D	0F	88	E8	94	.Çd"ÇÌ4®eåm..ˆè"
1:60F0h:	34	F9	9F	FF	D9	54	68	69	73	20	69	73	20	74	68	65	4ùŸÿÙThis is the
1:6100h:	20	68	69	64	64	65	6E	20	6D	65	73	73	61	67	65	1A	hidden message.
1:6110h:	00	00	00	73	74	65	67	6F									...stego

איור 14: תצוגה בינארית של קובץ ה-Jpeg בעל המסר הסמוי בשיטת End of File

בגלל שהמסר מקודד ב-plain text, כלומר, אינו מוצפן – ניתן לראותו בבירור בסוף הקובץ – "This is the hidden message." כמו כן, ניתן לראות את ה-marker ה-EOI שמסיים את תמונת ה-Jpeg: FFD9. אדם בעל הבנה בסיסית בפורמט JFIF יודע כי זהו ה-marker שמסיים את הקובץ ואחריו לא אמור לבוא מידע נוסף.

תוצאות דומות ניתן לראות גם בהטמעת המסר הסמוי בשיטה זו בתמונות Bitmap.

5.1.3. ניתוח השיטה

כפי שניתן לראות לעיל, אין שינוי ויזואלי בתמונות לאחר החבאת המסר הסמוי, ולכן שיטה זו טובה נגד עין בלתי מיומנת.

אך בגלל ששיטה זו מכניסה את המסר הסמוי "מחוץ לכותלי הפורמט" בעזרת עורך קבצים בינארי ניתן לראות את המסר כפי שהוא, במיוחד אם הוא נכתב plain text, ולכן שיטה זו חלשה מול בדיקות מעמיקות יותר.

גודל המסר הסמוי הוא בלתי מוגבל, משום שניתן להכניס בסוף הקובץ מסר ארוך כרצוננו מבלי לחשוף אותו לעין בלתי מיומנת. החיסרון במסרים ארוכים הוא שכל בי של מסר סמוי מגדיל את גודל הקובץ הכולל – לכן מסרים גדולים מדי יכולים לעורר חשד גם עבור "עין בלתי מיומנת"

5.2. Jpeg COM

שיטה זו כאמור עובדת על קבצי Jpeg בלבד, והיא ניצול של אחת מתכונות הפורמט על מנת להחביא בתוך הקובץ מסר סמוי.

5.2.1. תוצאות ויזואליות

להלן שתי תמונות, אחת לפני החבאת המסר הסמוי ואחת לאחר שימוש ב-StegoMaster להחבאת המסר "This is the hidden message." :

תמונת ה-Jpeg המקורית	תמונת ה-Jpeg עם המסר הסמוי

טבלה 3: השוואה ויזואלית של תמונות Jpeg בשיטה Jpeg COM

כפי שניתן לראות בטבלה 3, גם בשיטה זו לא רואים שינויים ויזואליים בתמונה, וזאת משום שהמידע העודף שהכנסנו לקובץ הוא בסגמנט COM שאינו משנה את אופן הצגת הקובץ.

5.2.2. תוצאות בינאריות

אם נסתכל בקובץ ה-Jpeg לאחר החבאת המסר הסמוי בתוכנת 010editor [12] תוך הפעלת template של קבצי Jpeg, כפי שניתן לראות באיור 15, נראה שקל למצוא את המסר הנסתר שלנו, וגם לקרוא אותו. לכן ניתן להגיד ששיטה זו אינה מוגנת מפני עורכים בינאריים לסוגם.

Name	Value	Start	Size
struct JPGFILE jpgfile		0h	1610Eh
enum M_ID SOIMarker	M_SOI (FFD8h)	0h	2h
struct APP2 app2		2h	21Eh
struct APP0 app0		220h	12h
struct APP13 app13		232h	96h
struct DQT dqt[0]		2C8h	45h
struct DQT dqt[1]		30Dh	45h
struct DQT dqt[2]		352h	45h
struct SOFx sof2		397h	13h
struct DHT dht[0]		3AAh	1Eh
struct DHT dht[1]		3C8h	1Eh
struct DHT dht[2]		3E6h	1Bh
struct COMMENT comment	This is the hidden message.	401h	1Fh
enum M_ID CommentMarker	M_JPG14 (FFFEh)	401h	2h
WORD szSection	29	403h	2h
char comment[27]	This is the hidden message.	405h	1Bh
struct SOS scanStart		420h	Bh
char scanData[89313]		42Bh	15CE1h
enum M_ID EOIMarker	M_EOI (FFD9h)	1610Ch	2h

איור 15: מבנה קובץ ה-Jpeg בעל המסר הסמוי בשיטה Jpeg COM

5.2.3. ניתוח השיטה

גם בשיטה זו כמו בשיטה הקודמת המתוארת ב-5.1, המסר הסמוי לא נראה ל"עין בלתי מיומנת" אך חשוף לעורכים בינאריים.

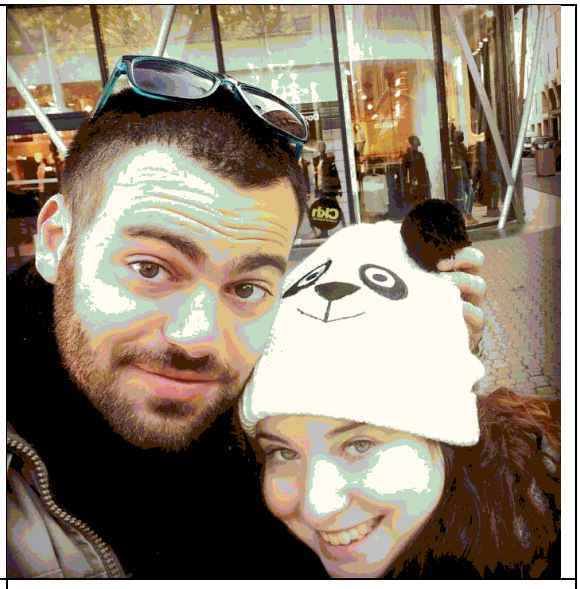
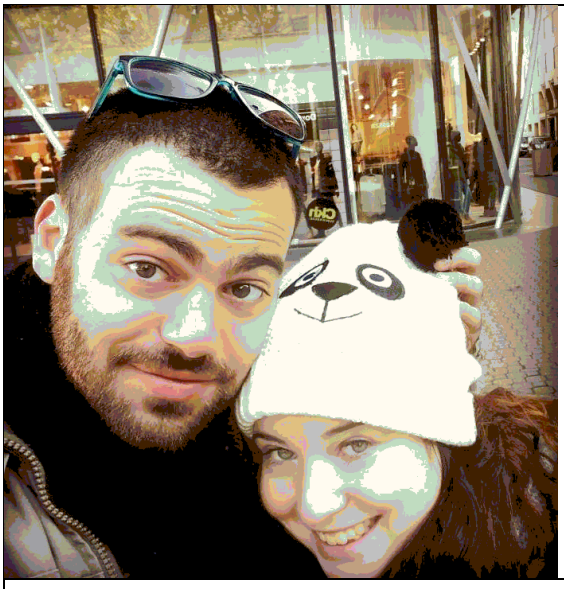
אך כאן, נפח הערוץ הסמוי מוגבל. אורך סגמנט COM בקבצי Jpeg מוגבל ל-2^{16} בתים. וכמו בשיטה הקודמת גם כאן – כל byte של מסר סמוי מגדיל את גודל הקובץ הכולל ב-byte. לכן, מסר גדול מדי עלול להוות חשד גם עבור "עין בלתי מיומנת".

5.3. Bitmap Gap

בשיטה זו המסר הסמוי מוחבא בין חלקי הפורמט, כלומר בין החלק המוגדר כ-metadata לבין החלק המוגדר כ-data.

5.3.1. תוצאות ויזואליות

אם נסתכל על תמונות Bitmap לפני ואחרי החבאת המסר הסמוי "Can you see this message?":

	
תמונת ה-Bitmap המקורית	תמונת ה-Bitmap עם המסר הסמוי

טבלה 4: השוואה ויזואלית של תמונות Bitmap בשיטה Bitmap Gap

כפי שניתן להבחין בטבלה 4, בתמונות – אין שינוי ויזואלי, זאת משום שהמסר הסמוי מוחבא בין החלקים בעלי המשמעות בתמונה, ולכן לא בא לידי ביטוי בהצגתה.

5.3.2. תוצאות בינאריות

אם נביט בתמונת ה-Bitmap בעל המסר הסמוי בעורך הבינארי 010editor [12], כפי שרואים באיור 16, נוכל להבחין במסר הסמוי.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	42	4D	4E	BB	17	00	00	00	00	00	4E	00	00	00	28	00	BMN»......N...(.
0010h:	00	00	D0	02	00	00	D0	02	00	00	01	00	18	00	00	00	..Ð...Ð.........
0020h:	00	00	00	BB	17	00	C3	0E	00	00	C3	0E	00	00	00	00	...»..Ã...Ã.....
0030h:	00	00	00	00	00	00	43	61	6E	20	79	6F	75	20	73	65	......Can you se
0040h:	65	20	74	68	65	20	6D	65	73	73	61	67	65	3F	40	40	e the message?@@
0050h:	40	40	20	40	40	20	40	40	20	40	40	20	40	40	20	40	@@ @@ @@ @@ @@ @
0060h:	40	20	40	40	20	40	40	20	40	40	20	40	40	20	40	00	@ @@ @@ @@ @@ @.
0070h:	20	40	00	20	40	00	20	40	00	20	40	00	20	40	00	20	@. @. @. @. @.
0080h:	40	00	20	40	00	20	40	00	20	40	00	20	40	00	20	40	@. @. @. @. @. @
0090h:	00	20	40	00	20	40	00	20	40	00	20	40	40	20	20	00	. @. @. @. @@ .
00A0h:	20	40	00	20	40	40	20	20	40	20	40	40	20	40	40	20	@. @@ @ @@ @@
00B0h:	40	40	20	40	40	20	40	40	20	40	00	20	40	00	20	40	@@ @@ @@ @. @. @

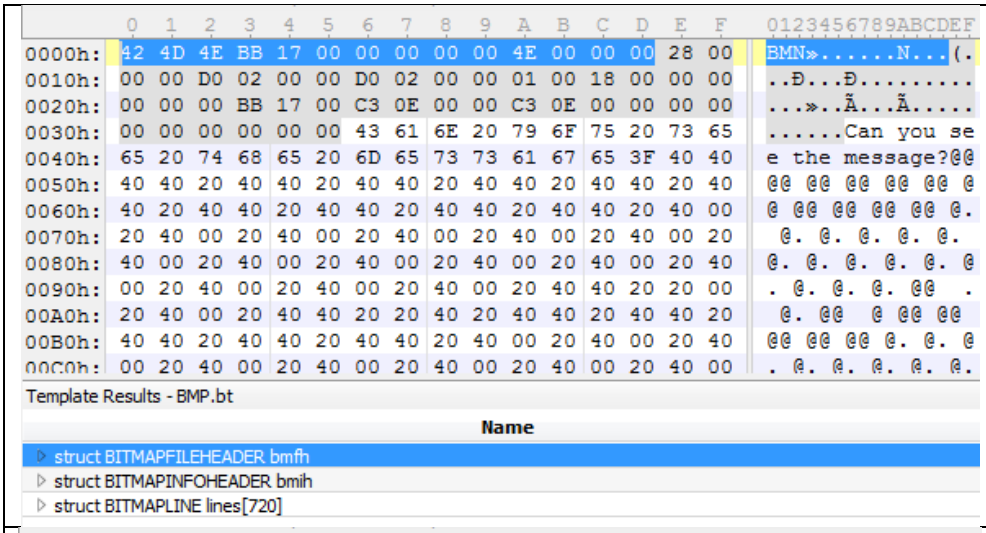
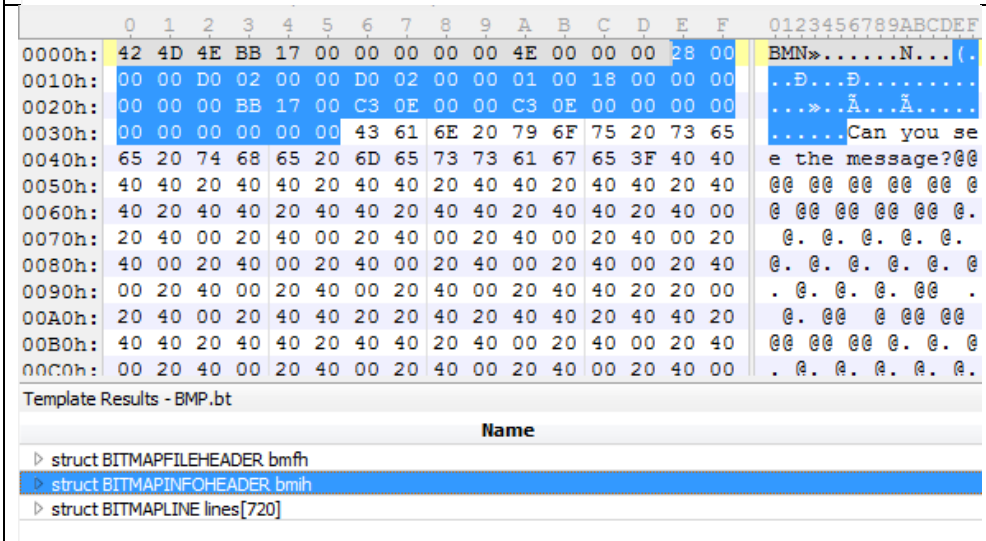
איור 16: תצוגה בינארית של קובץ ה-Bitmap בעל המסר הסמוי בשיטת Bitmap Gap

כמו כן, ניתן להבחין שהמסר הסמוי מתחיל בהיסט 0x36 מתחילת הקובץ, ונגמר בהיסט 0x4E. אם נסתכל ב-BitmapFileHeader, כפי שמוצג באיור 17, נוכל לראות כי אכן ה-data של התמונה מתחיל בהיסט 78 (או 0x4E) כלומר, מיד לאחר המסר.

Name	Value
struct BITMAPFILEHEADER bmfh	
CHAR bfType[2]	BM
DWORD bfSize	1555278
WORD bfReserved1	0
WORD bfReserved2	0
DWORD bfOffBits	78
struct BITMAPINFOHEADER bmih	
struct BITMAPLINE lines[720]	

איור 17 : תצוגת ה-Header של קובץ ה-Bitmap בעל המסר הסמוי בשיטה Bitmap Gap

נסמן בעורך את ה-headers של התמונה כדי לראות שהם אכן נגמרים בהיסט 0x36 :

	Bitmap File Header
	Bitmap Info Header

טבלה 5: תצוגה בינארית של ה-Headers של קובץ ה-Bitmap בעל המסר הסמוי בשיטה Bitmap Gap

כמו שניתן לראות בטבלה 5, ה-Headers אכן מגיעים בתחילת הקובץ ונגמרים בדיוק בהיסט 0x36, היכן שמחיל המסר הסמוי.

כמו כן, גם ניתן להבחין כי העורך אינו מודע לכך שיש חלק נוסף בקובץ (ה-gap), והוא לא מציג אותו ברשימת חלקי הקובץ מטה, זאת משום שה-gap הוא אינו חלק רשמי מן הפורמט.

5.3.3. ניתוח השיטה

גם כאן, כמו בשיטות הקודמות שראינו עד כה, המסר הסמוי גלוי לעורכים בינאריים אך נסתר מעיניי המשתמש הסביר.

משום שגם שיטה זו מוסיפה את המסר הסמוי לקובץ ולא "דורסת" אף מידע שקיים בקובץ מראש – כל בית של מסר סמוי מגדיל בבית את הקובץ.

גודל הערוץ הסמוי כאן מוגבל על ידי תיאור ה-offset של תחילת ה-data, ולכן, גודל המסר הסמוי חייב לקיים את המשוואה הבאה:

$$sizeof\left(BitmapFileHeader\right)+sizeof\left(BitmapInfoHeader\right)+sizeof\left(HiddenMessage\right)<2^{32}$$

משום שההיסט של תחילת ה-data חייב להיות מתואר על ידי DWORD.

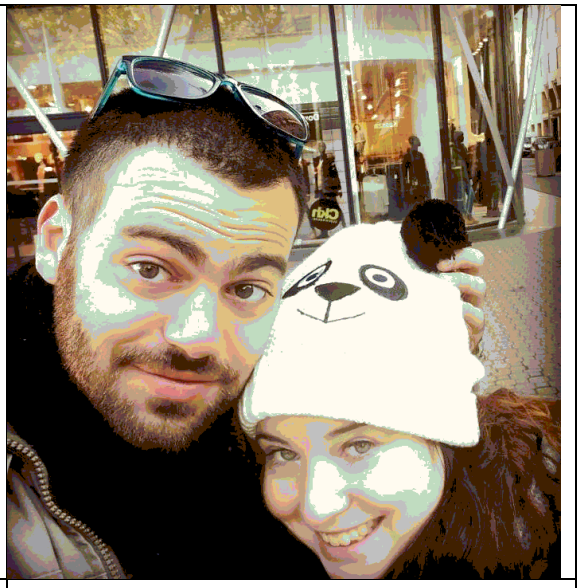
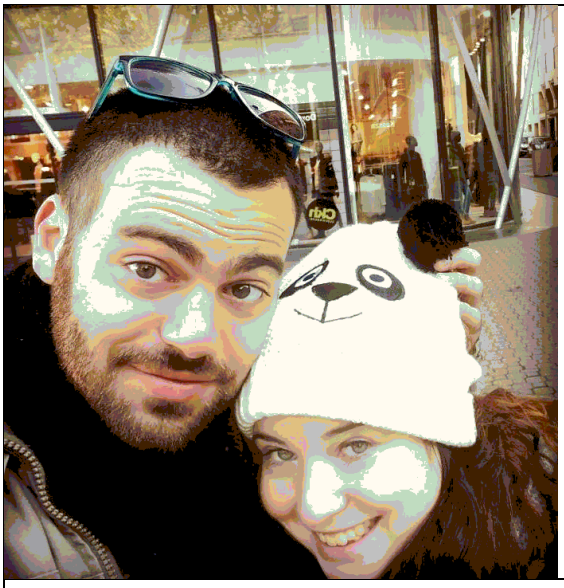
5.4. Bitmap LSB

בשיטה זו בניגוד לשאר השיטות שסקרנו עד כה, אנו משנים את ה-data של התמונה, דבר שגורר תכונות סטגנוגרפיות שטרם ראינו.

5.4.1. תוצאות ויזואליות

בסעיף ה נביט בתוצאות הויזואליות של שיטה זו כפי שמומשה ב-StegoMaster אך גם נדון במה היה יכול להיות במימושים שונים.

להלן תמונת Bitmap לפני ואחרי החבאת המסר הסמוי "The message is in the data now":

	
תמונת ה-Bitmap המקורית	תמונת ה-Bitmap עם המסר הסמוי

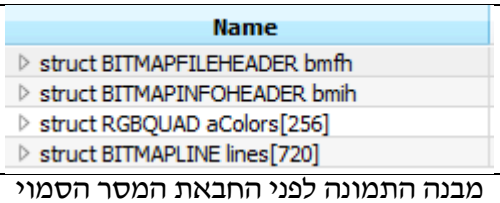
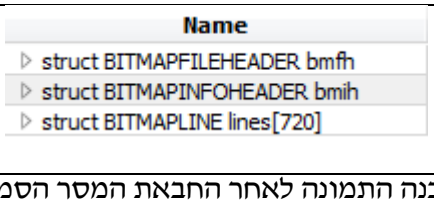
טבלה 6: השוואה ויזואלית של תמונות Bitmap בשיטה Bitmap LSB

כפי שניתן להבחין בטבלה 6 לעיל, גם כאן לא ניתן לראות שינוי ויזואלי בתמונה וזאת למרות שנעשה שינוי ב-data. השינוי שמומש ב-StegoMaster משנה את הביט התחתון של כל ערוץ צבע של כל פיקסל – דבר שאינו נראה לעין. אך, אם נרצה להחריב את ערוץ המידע ולהחביא יותר ביטים בכל ערוץ צבע – המסר הסמוי יראה לעין יותר ויותר ככל שהנפח הערוץ יגדל.

5.4.2. תוצאות בינאריות

בניגוד לשיטות הקודמות מכיוון שאנו משנים data קיים, לא ניתן לראות את הערוץ הסמוי בבירור גם בעזרת עורך בינארי. אך נוכל לראות כי יש שינוי בינארי בין הקבצים.

התמונה המקורית היא בעלת פלטת צבעים, וכפי שתואר ב-4.4, תוכנת StegoMaster ממירה את התמונה המקורית לתמונה בעלת עומק סיביות 24, כלומר ללא פלטה. לכן – נוכל לראות בטבלה 7 כי בתמונה לאחר החבאת המסר הסמוי לא קיימת פלטה.

	
מבנה התמונה לפני החבאת המסר הסמוי	מבנה התמונה לאחר החבאת המסר הסמוי

טבלה 7: השוואה בין מבני קבצי Bitmap לפני ולאחר הטמעת מסר סמוי בשיטה Bitmap LSB

אך גם אם נטמיע מסר סמוי בתמונה ללא פלטה נוכל לראות הבדל ב-data בעורך הבינארי.

תמונת ה-Bitmap המקורית	תמונת ה-Bitmap עם המסר הסמוי

טבלה 8: השוואה ויזואלית של תמונות Bitmap בעומק סיביות 24 בשיטה Bitmap LSB

למרות שלא ניתן לראות הבדל בין התמונות המוצגות בטבלה 8, אם נשווה בין התמונות בעורך בינארי, כפי שניתן לראות באיור 18, נוכל לראות הבדל בין הקבצים.

Result	Address A	Size A	Address B	Size B
Match	0h	26h	0h	26h
Difference	26h	5h	26h	5h
Match	2Bh	Bh	2Bh	Bh
Difference	36h	2Eh	36h	2Eh
Match	64h	8h	64h	8h
Difference	6Ch	EAh	6Ch	EAh
Match	156h	Ah	156h	Ah
Difference	160h	Ch	160h	Ch
Match	16Ch	1BA30Ah	16Ch	1BA30Ah

איור 18: השוואה בינארית בעורך 010editor בין קבצי ה-Bitmap בשיטה Bitmap LSB

– הבדל זה נובע עקב השימוש בספריית 0x26בנוסף, נבחין בהבדל של 5 בתים בכתובת שתוארה בסעיף ImageMagick 4.4.

שאר הסגמנטים הנבדלים (Difference) הם מה-data של הקובץ וניתן לראות כי אכן הם גדולים יחסית, וביניהם קטעים יחסית קטנים של התאמות (Match) – מידע שבמקרה התאים לביט התחתון של ה-data המקורי.

5.4.3. ניתוח השיטה

כפי שנאמר קודם, אם משתמשים בשיטה זו באופן לא אחראי, כלומר מחביאים יותר מידי סיביות בכל פיקסל, ערוץ המידע יכול להתגלות לעין גם עבור משתמש ללא עורך בינארי.

יתרון משמעותי של השיטה היא שהקובץ לבד לא נראה חשוד כלל, משום שלא ניתן לראות בו את המסר ללא השוואה לקובץ המקורי.

גודל הערוץ הסמוי חסום על ידי: $\frac{3 \cdot numberOfPixels}{8}$ משום שבכל פיקסל בכל אחד משלוש ערוצי הצבע ניתן להחביא סיבית אחת.

5.5. Wave LSB

שיטה זו דומה לקודמתה אך כאן מדובר בקבצי שמע, לכן לא נוכל להראות תוצאות ויזואליות במסמך זה, אלא רק נתאר את ההבדל בשמע ואת התוצאות הבינאריות.

5.5.1. תוצאות בינאריות

בשיטה זו החבאנו את המסר "It's time for audio files", ונתבונן בהשוואה בינארית בין הקבצים שנערכה בעורך הבינארי 010editor [12], כאשר Address A מתייחס להיסט מסוים בקובץ בעל המסר הסמוי, ו-Address B להיסט בקובץ המקורי.

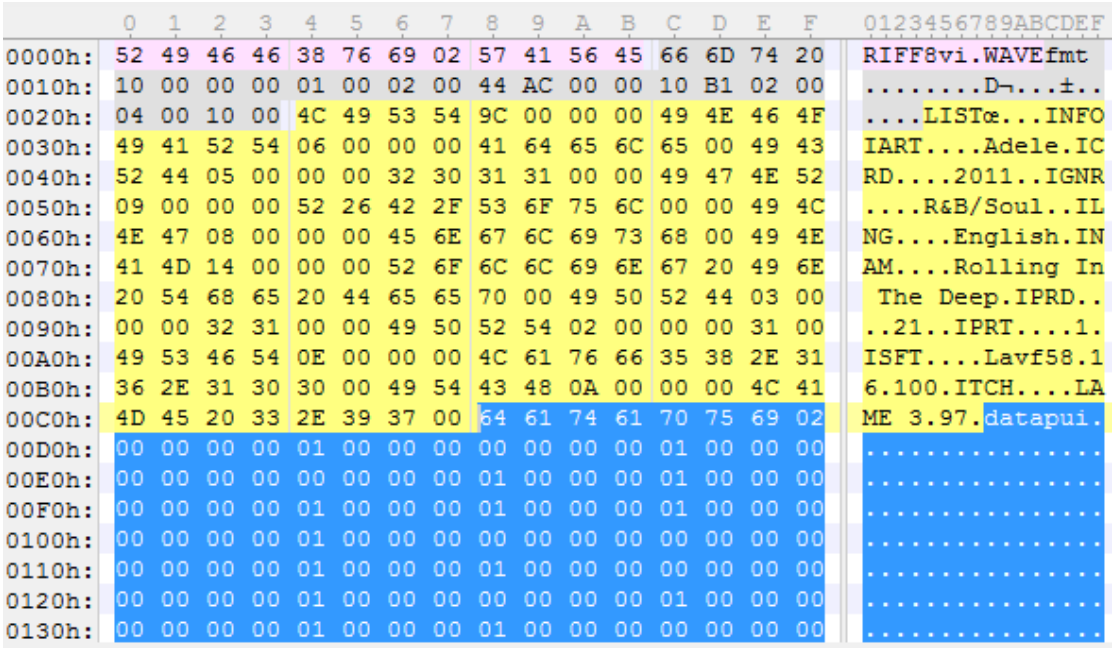
Result	Address A	Size A	Address B	Size B
Match	0h	D4h	0h	D4h
Difference	D4h	7h	D4h	9h
Match	DBh	Bh	DDh	Bh
Difference	E6h	17h	E8h	1Dh
Match	FDh	Fh	105h	Fh
Difference	10Ch	3h	114h	5h
Match	10Fh	Bh	119h	Bh
Difference	11Ah	11h	124h	15h
Match	12Bh	Bh	139h	Bh
Difference	136h	20h	144h	29h
Match	156h	Fh	16Dh	Fh
Difference	165h	3h	17Ch	5h
Match	168h	Bh	181h	Bh
Only in B			18Ch	1h
Match	173h	67h	18Dh	67h
Only in B			1F4h	1h
Match	1DAh	Bh	1F5h	Bh
Only in B			200h	1h
Match	1E5h	Bh	201h	Bh
Difference	1F0h	14h	20Ch	19h
Match	204h	13h	225h	13h
Only in B			238h	1h
Match	217h	Bh	239h	Bh
Difference	222h	13h	244h	19h
Match	235h	Bh	25Dh	Bh
Difference	240h	3h	268h	5h

איור 19: השוואה בינארית בעורך editor010 בין קבצי ה-Wave בשיטה Wave LSB

כפי שניתן לראות באיור 19, החל מהיסט 0xD4 מתחילים לצוץ מקטעים לסירוגין של התאמה ושוני בין הקבצים. אנו מצפים שמקור השוני הוא אכן ה-data של הקובץ בו שונו הדגימות. נביט באיור 20 המציג את ניתוח אחד הקבצים על ידי Wave template בעורך 010editor בשביל לברר האם אכן היסט 0xD4 נמצא בתוך ה-data chunk:

Name	Value	Start	Size
struct WAVRIFFHEADER header		0h	Ch
struct FORMATCHUNK format		Ch	18h
struct LISTCHUNK list		24h	A4h
struct DATACHUNK data		C8h	2697578h

איור 20: מבנה קובץ ה-Wave לאחר הטמעת המסר הסמוי בשיטה Wave LSB



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	52	49	46	46	38	76	69	02	57	41	56	45	66	6D	74	20	RIFF8vi.WAVEfmt
0010h:	10	00	00	00	01	00	02	00	44	AC	00	00	10	B1	02	00	........D¬...±..
0020h:	04	00	10	00	4C	49	53	54	9C	00	00	00	49	4E	46	4F	....LISTœ...INFO
0030h:	49	41	52	54	06	00	00	00	41	64	65	6C	65	00	49	43	IART....Adele.IC
0040h:	52	44	05	00	00	00	32	30	31	31	00	00	49	47	4E	52	RD....2011..IGNR
0050h:	09	00	00	00	52	26	42	2F	53	6F	75	6C	00	00	49	4C	....R&B/Soul..IL
0060h:	4E	47	08	00	00	00	45	6E	67	6C	69	73	68	00	49	4E	NG....English.IN
0070h:	41	4D	14	00	00	00	52	6F	6C	6C	69	6E	67	20	49	6E	AM....Rolling In
0080h:	20	54	68	65	20	44	65	65	70	00	49	50	52	44	03	00	The Deep.IPRD..
0090h:	00	00	32	31	00	00	49	50	52	54	02	00	00	00	31	00	..21..IPRT....1.
00A0h:	49	53	46	54	0E	00	00	00	4C	61	76	66	35	38	2E	31	ISFT....Lavf58.1
00B0h:	36	2E	31	30	30	00	49	54	43	48	0A	00	00	00	4C	41	6.100.ITCH....LA
00C0h:	4D	45	20	33	2E	39	37	00	64	61	74	61	70	75	69	02	ME 3.97.datapui.
00D0h:	00	00	00	00	01	00	00	00	00	00	00	00	01	00	00	00	....
00E0h:	00	00	00	00	00	00	00	00	01	00	00	00	01	00	00	00	....
00F0h:	00	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00	....
0100h:	00	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	....
0110h:	00	00	00	00	01	00	00	00	01	00	00	00	00	00	00	00	....
0120h:	00	00	00	00	01	00	00	00	00	00	00	00	01	00	00	00	....
0130h:	00	00	00	00	01	00	00	00	01	00	00	00	00	00	00	00	....

איור 21: תצוגה בינארית של קובץ ה-Wave לאחר הטמעת המסר הסמוי בשיטה Wave LSB

ואכן ניתן לראות באיור 21 כי היסט 0xD4 מתחילת הקובץ שייך ל-data chunk הנמשך עד סוף הקובץ. כלומר, השינוי היחיד בקובץ נעשה ב-data chunk.

הקובץ לאחר הטמעת המסר הסמוי בהיבט הבינארי נראה למשתמש כמו קובץ תקין כך שאין דרך לדעת בוודאו שמדובר בקובץ נשא למסר סמוי.

למרות שאין דרך לקבוע בוודאות כי אכן מדובר בקובץ נשא, ניתן ליישם היוריסטיקה. הרבה קבצי audio מתחילים ברצף לא קטן של דגימות (samples) מאופסות – לכן כאשר רואים דגימות שערכן 1 ניתן להניח שמדובר בקובץ שאינו "טבעי".

גם את ההיוריסטיקה ניתן לעקוף על ידי כך שלא נתחיל לקודד את המסר הסמוי בתחילת ה-data אלא רק אחרי רצף של דגימות מאופסות.

5.5.2. ניתוח השיטה

המימוש בתוכנת StegoMaster נעשה על ידי הטמעת הערוץ הסמוי באחד מבין ערוצי השמע, דבר שמגביל את נפח הערוץ הסמוי על ידי $\frac{NumberOfSamples}{8}$.

בעזרת שינויים במימוש, החבאת המסר בכל אחד מערוצי השמע, היה ניתן ליצור ערוץ סמוי שמוגבל על ידי $\frac{NumberOfChannels \cdot NumberOfSamples}{8}$.

גם כאן, כמו שדנו בסעיף 5.4.3, ניתן להגדיל את נפח הערוץ הסמוי על חשבון מספר הסיביות שדורסים בכל דגימה של audio. כמובן שככל שנדרוס יותר סיביות כך הערוץ הסמוי ישמע יותר בבירור כאשר מנגנים את הקובץ. במימוש השיטה ב-StegoMaster נדרסה רק סיבית אחת בכל דגימה והערוץ הסמוי לא נשמע כלל.

5.6. Wave Additional Chunk

שיטה זו גם פועלת על קבצי audio מסוג Wave, אך בניגוד לשיטה הקודמת שסקרנו, שיטה זו מחביאה את המידע ב-metadata של הקובץ ולא ב-data. דבר שיהפוך את המסר הסמוי לברור יותר במובנים מסוימים.

5.6.1. תוצאות בינאריות

נחביא את המסר "hidden message in the metadata? ok!" בקובץ wave, ונסתכל עליו בעורך הבינארי 010editor, ראו איור 22.

Name	Value	Start	Size
struct WAVRIFFHEADER header		0h	Ch
struct FORMATCHUNK format		Ch	18h
struct LISTCHUNK list		24h	A4h
struct DATACHUNK data		C8h	2697578h
struct UNKNOWNCHUNK unknown		2697640h	2Bh
ID chunkID[4]	stgo	2697640h	4h
long chunkSize	35	2697644h	4h
uchar unknownData[35]		2697648h	23h

איור 22: מבנה קובץ ה-Wave לאחר הטמעת המסר הסמוי בשיטה Wave Additional Chunk

נוכל לראות שישנו chunk שהעורך איננו מכיר וזה מסמן אותו כ-"unknown", ה-chunk הנ"ל מתחיל בהיסט 0x2697640 מתחילת הקובץ.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
269:75B0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
269:75C0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
269:75D0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
269:75E0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
269:75F0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
269:7600h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
269:7610h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	(00	00)	.....(..)
269:7620h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
269:7630h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
269:7640h:	73	74	67	6F	23	00	00	00	68	69	64	64	65	6E	20	6D	stgo#...hidden m
269:7650h:	65	73	73	61	67	65	20	69	6E	20	74	68	65	20	6D	65	essage in the me
269:7660h:	74	61	64	61	74	61	3F	20	6F	6B	21						tadata? ok!

איור 23 : תצוגה בינארית של קובץ ה-Wave בעל המסר הסמוי בשיטת Wave Additional Chunk

אנו יכולים לראות באיור 23 שה-chunk הנ"ל אכן מכיל את המסר הסמוי הרצוי – דבר שאומר שהמסר הסמוי שהחבאנו אכן חשוף לקריאה בעורך בינארי.

5.6.2. ניתוח השיטה

ניגון הקובץ לאחר הטמעת המסר הסמוי נשמע בדיוק כמו הקובץ המקורי משום שכאשר נגני מוזיקה נתקלים ב-chunk שאינם מכירים בפורמט wave הם מדלגים עליו ולא מייחסים לו משמעות. על כן, השיטה מוגנת מפני שימוש סביר של הקובץ.

כפי שראינו, בעזרת עורך בינארי ניתן לראות את המסר הסמוי בבירור.

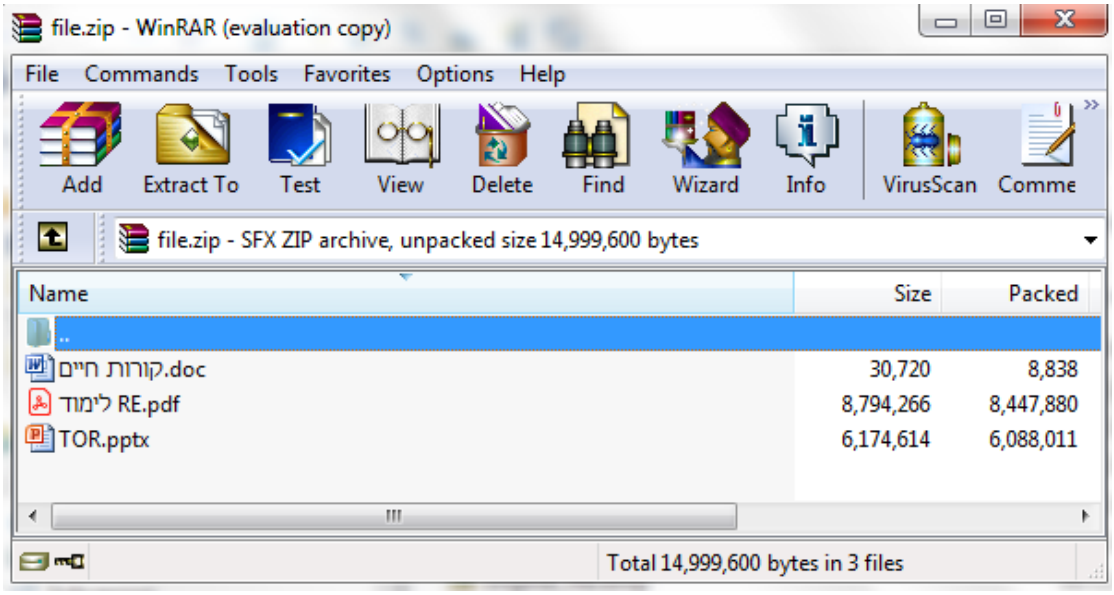
נפח הערוץ הסמוי בשיטה זו מוגבל אך ורק על ידי גודל chunk מקסימלי לפי פורמט wave והוא 2^{32} בתים – נפח גדול מאוד. רק יש לזכור שכל byte של ערוץ סמוי שמטמיעים בקובץ מגדיל את הקובץ ב-byte.

Zip Beginning .5.7

שיטה זו דומה בבסיסה לשיטה שראינו ב-5.1, רק שהפעם מדובר על הוספת הערוץ הסמוי בתחילת הקובץ ולא בסופו. שיטה זו אפשרית משום שקריאת קובץ zip נעשית מהסוף להתחלה.

5.7.1. תוצאות ויזואליות

איור 24 מתאר פתיחת קובץ Zip בתוכנת WinRAR [13] שבו הוחבא המסר: "Let's try it with a zip file".

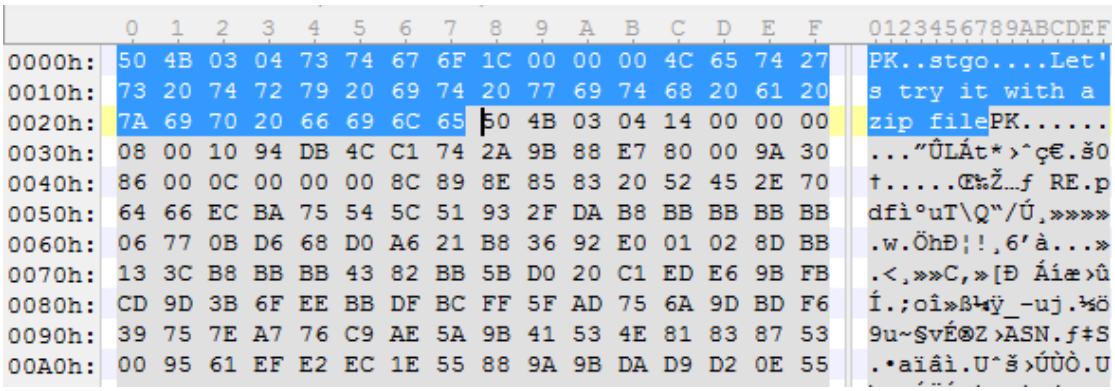


איור 24: תצוגה של קובץ ה-Zip בעל המסר הסמוי בשיטת Zip Beginning בתוכנת WinRAR

כפי שניתן לראות – הקובץ נפתח כשורה ואין סימן למסר סמוי מסוג כלשהו.

5.7.2. תוצאות בינאריות

כאשר נפתח את הקובץ בעורך הבינארי 010editor נוכל לראות בתחילת הקובץ את המסר הסמוי שלנו ולאחר מכן את תחילת קובץ ה-zip, כפי שרואים באיור 25.



איור 25: תצוגה בינארית של קובץ ה-Zip בעל המסר הסמוי בשיטת Zip Beginning

המסר בתמונה מסומן בכחול, ומיד אחריו מתחיל קובץ ה-zip המקורי.

מכאן נובע כי שיטה זו חשופה בפני עורכים בינאריים וקל לראות באמצעותם את המסר הסמוי.

5.7.3. ניתוח השיטה

ניתוח השיטה מבחינה הן מבחינה ויזואלית והן מבחינה בינארית כבר נעשה בסעיפים הקודמים, נפח הערוץ הסמוי הוא בלתי מוגבל בשיטה התאורטית, אך מוגבל ל-2^{32} במימוש ב-StegoMaster משום שזה מתאר את אורך המסר הסמוי על ידי DWORD.

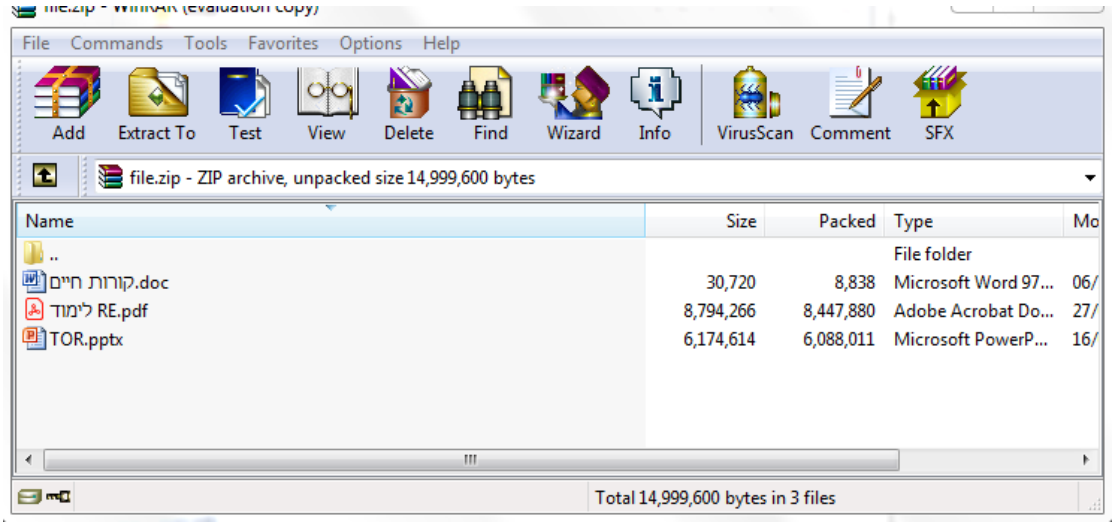
5.8. Zip Additional Record

שיטה זו, בניגוד לשיטה שהוצגה ב-5.7, מתייחסת יותר לעומק לפורמט Zip משום שהיא משתמשת במבנים פנימיים של הקובץ על מנת להחדיר את המסר הסמוי פנימה. שיטה זו משתמשת ב-Zip כפורמט נשא לקבצים ומשתילה בו קובץ שאינו נראה על ידי תוכנות הקוראות קבצי Zip.

5.8.1. תוצאות ויזואליות

גם בשיטה זו לא ניתן להבחין במסר הסמוי בקובץ, למרות שבשיטה זו, אנו מחביאים קובץ שלם ולא רק מסר כמחרוזת.

בשיטה זו נחביא את המסר "Hiding an entire file in the zip", ונראה באיור 26 שהקובץ לאחר הטמעת המסר הסמוי לא מכיל את הקובץ הנוסף – לפחות למראית עין בתוכנת WinRAR.



איור 26 : תצוגה של קובץ ה-Zip בעל המסר הסמוי בשיטת Zip Additional Chunk בתוכנת WinRAR

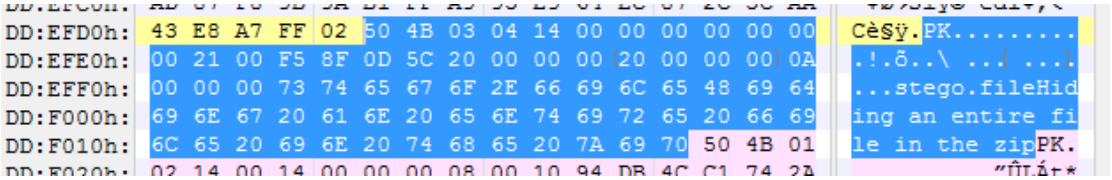
5.8.2. תוצאות בינאריות

אם נפתח את הקובץ בעורך הבינארי 010editor כפי שמתואר באיור 27 ונפעיל עליו template של zip, נוכל לראות כי אכן נראה כי נוסף record נוסף למארז ללא entry מתאים, ואכן אותו record מכיל את המסר הסמוי שלנו.

Name	Value	Start	Size
struct ZIPFILERECORD record[0]	Œ‰...ƒ R...	0h	80E7B2h
struct ZIPFILERECORD record[1]	—...˜...š ‡...	80E7B2h	22B2h
struct ZIPFILERECORD record[2]	TOR.pptx	810A64h	5CE571h
struct ZIPFILERECORD record[3]	stego.file	DDEFD5h	48h
struct ZIPDIRENTRY dirEntry[0]	Œ‰...ƒ R...	DDF01Dh	3Ah
struct ZIPDIRENTRY dirEntry[1]	—...˜...š ‡...	DDF057h	3Ch
struct ZIPDIRENTRY dirEntry[2]	TOR.pptx	DDF093h	36h
struct ZIPENDLOCATOR endLocator		DDF0C9h	16h

איור 27 : מבנה קובץ ה-Zip בעל המסר הסמוי בשיטה Zip Additional Chunk

אנו רואים שה-record הנוסף מתחיל בהיסט 0xDDEFD5 מתחילת הקובץ, ואם נסתכל בהיסט הנ"ל נוכל לראות מבנה record שהתוכן שלו הוא המסר הסמוי, כפי שניתן לראות באיור 28.



איור 28: תצוגה בינארית של קובץ ה-Zip בעל המסר הסמוי בשיטת Zip Additional Chunk

5.8.3. ניתוח השיטה

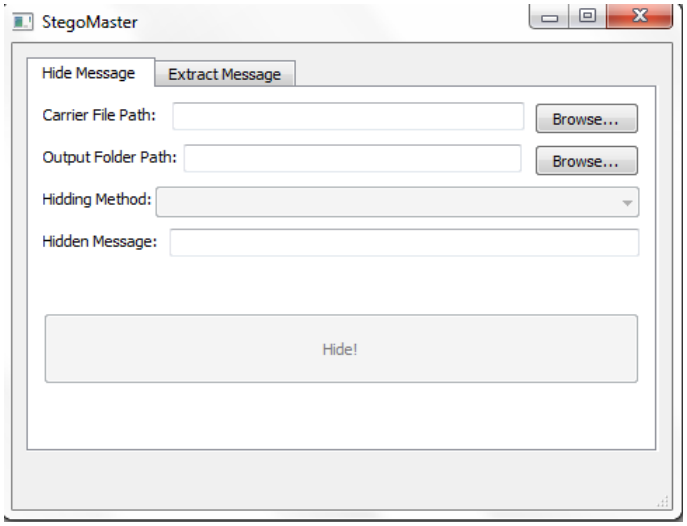
כפי שניתן לראות בסעיפים הקודמים, השיטה לא ניתנת לגילוי על פתיחת הקובץ בתוכנות הקוראות קבצי zip, אך המסר כן חשוף בפני עורכים בינאריים למיניהם.

היתרון המרכזי של השיטה הוא שניתן להחביא כך קבצים רבים, או אפילו היררכיה שלמה. כל זאת כל עוד לוקחים בחשבון את מגבלת הגודל המארז שהיא 2^{32} בתים.

6. שימוש ביישום

השימושים של פרויקט StegoMaster הם, כפי שצוין, החבאת מסר סמוי בקובץ נתון והוצאת המסר מן הקובץ בהינתן השיטה בה הוא הוחבא. השימוש בפרויקט נעשה דרך Graphic User Interface תמציתי ופשוט למשתמש.

כאשר פותחים את תוכנת StegoMaster דרך הקובץ StegoMasterGui.exe ייפתח החלון המוצג באיור 29.



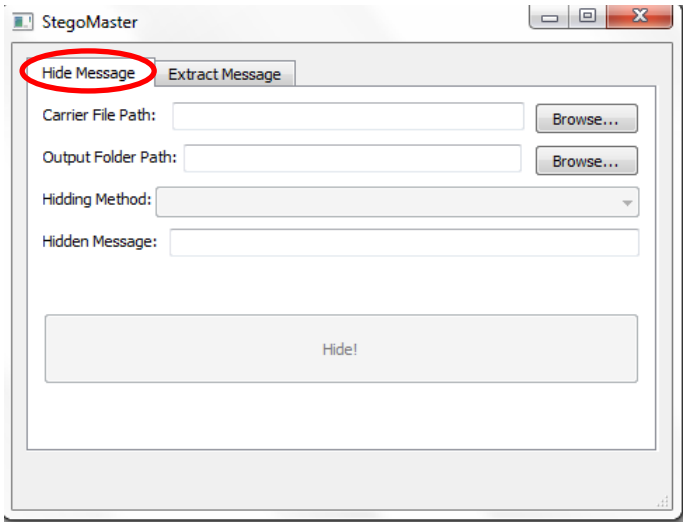
איור 29: חלון StegoMaster האחראי על החבאת מסרים

דרכו ניתן לבחור בלשונית כרצוננו, להחביא מידע בקובץ או לחלץ מידע מקובץ שכבר הוחבא בו המידע קודם לכן.

חשוב לזכור את השיטה בה מחביאים את המסר הסמוי, משום שיש לנקוב בשמה כאשר רוצים לחלץ את המסר מן הקובץ – טעות במסר יגרום לשגיאה בתוכנה וכמובן לאי חילוץ המסר.

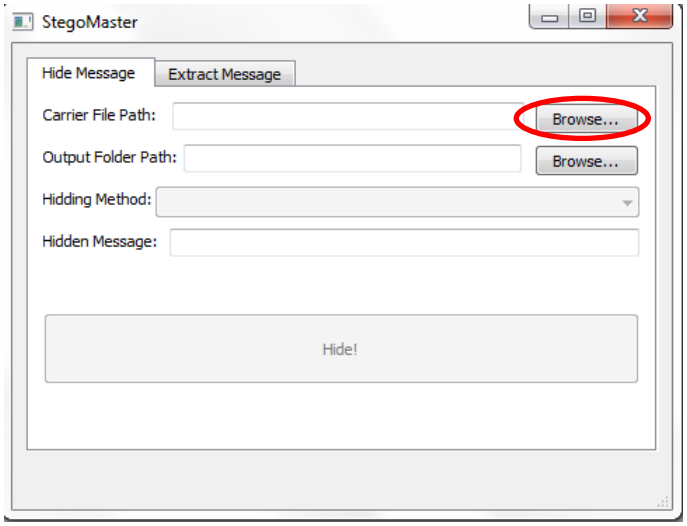
6.1. הטמעת מסר סמוי

על מנת להטמיע מסר סמוי בקובץ, יש לבחור בלשונית Hide Message בחלק העליון של המסך, כפי שניתן לראות באיור 30.



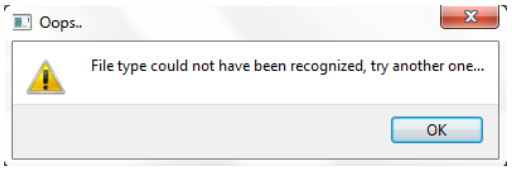
איור 30: תווית החבאת המסרים

לאחר מכן יש לבחור את הקובץ שאנו רוצים להטמיע בו את המסר על ידי הכפתור המוצג באיור 31.



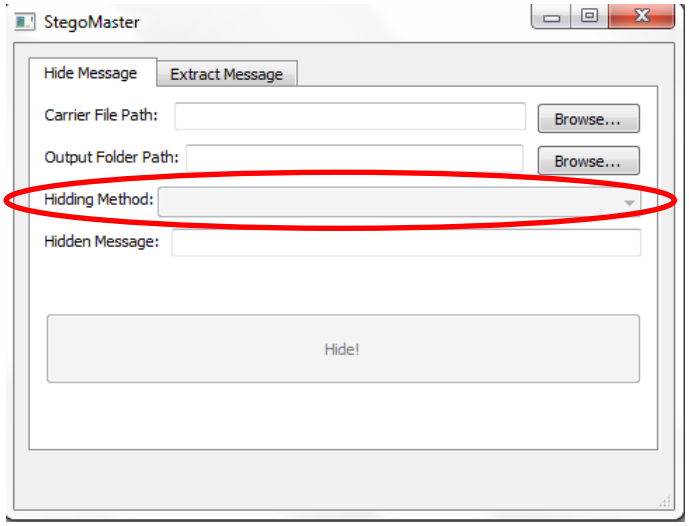
איור 31: כפתור בחירת נתיב הקובץ הנשא להחבאת המסר

אם הקובץ לא מזוהה על ידי תוכנת StegoMaster עלול לצוץ חלון השגיאה המופיע באיור 32.



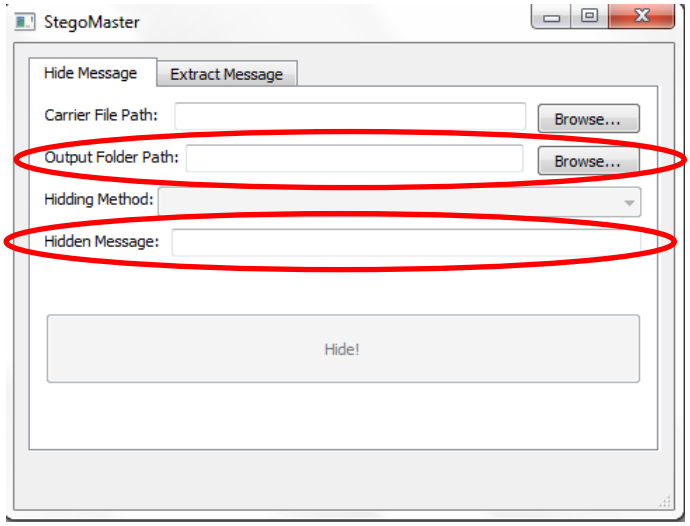
איור 32 : חלון שגיאת בחירת קובץ מפורמט לא מזוהה

ולכן יש להשתמש רק בקבצים מהפורמטים הנתמכים בתוכנה כפי שתוארו במסמך זה. אם נבחר קובץ מפורמט נתמך יופיעו שיטות להחבאת המידע באותו הקובץ בשדה Hidden Method כפי שמוצג באיור 33.



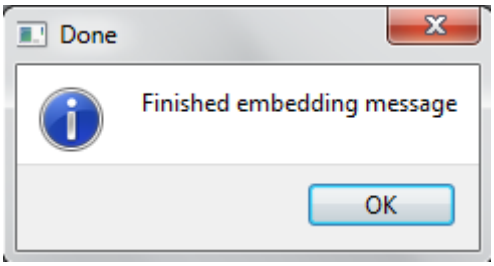
איור 33 : רשימת בחירת השיטה הסטגנוגרפית

לאחר מילוי תיקיית היעד (היכן לשמור את הקובץ לאחר הטמעת המסר הסמוי), והמסר עצמו לפי איור 34.



איור 34 : בחירת נתיב קובץ הפלט וכתיבת המסר הסמוי

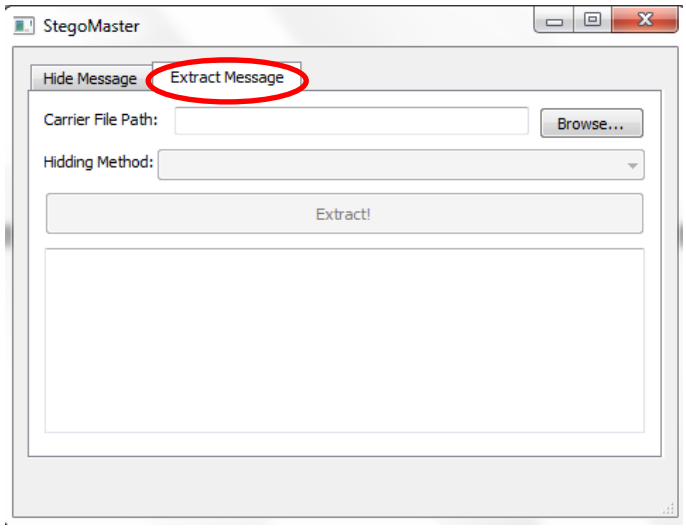
לאחר מילוי כל השדות הנ"ל כפתור ה-"Hide!" יתאפשר ללחיצה, בעת הלחיצה כל החלון הראשי יהיה לא זמין עד סיום התהליך. כאשר המסר הוטמע בהצלחה תתקבל ההודעה המוצגת באיור 35.



איור 35 : הודעת סיום הטמעת מסר סמוי בהצלה

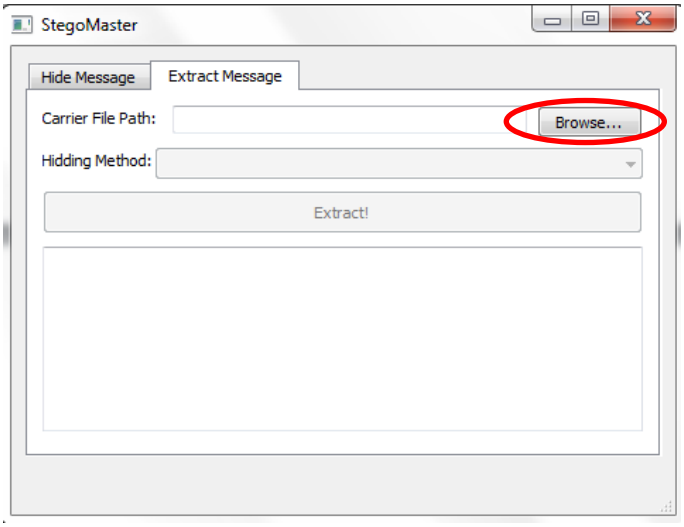
6.2. חילוץ מסר סמוי

על מנת לחלץ מסר סמוי מקובץ יש לבחור בתווית Extract Message בחלק העליון של המסך, כפי שניתן לראות באיור 36.



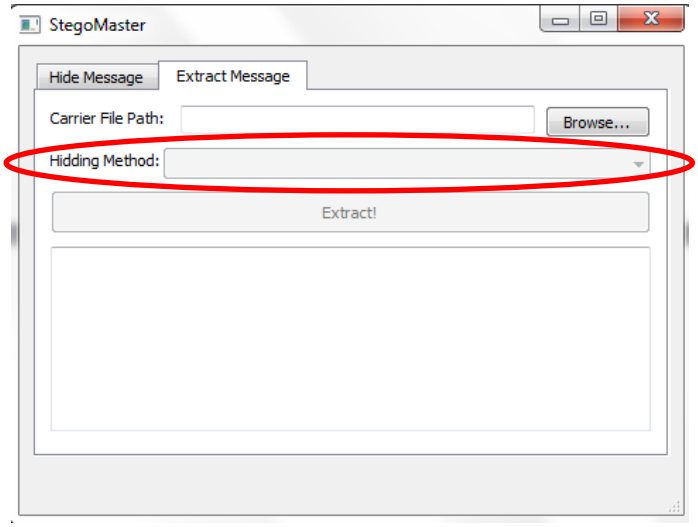
איור 36 : תווית חילוץ מסרים סמויים

לאחר בחירת הלשונית המתאימה, יש לבחור את הקובץ שממנו אנו רוצים לחלץ את המסר הסמוי על ידי לחיצה על הכפתור כפי שמופיע באיור 37.



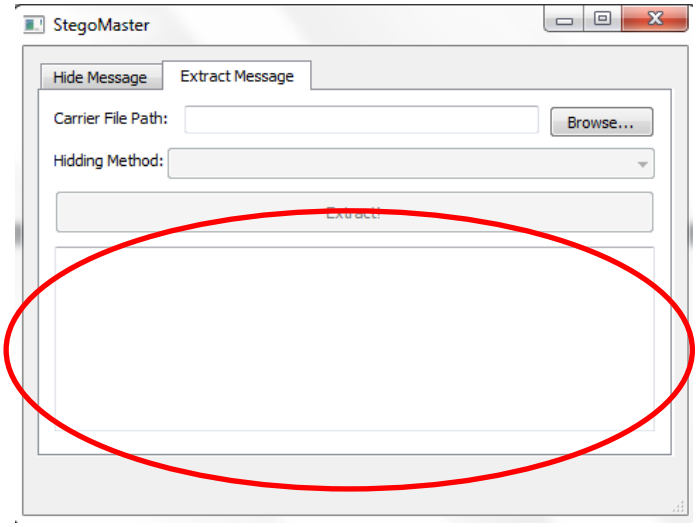
איור 37: כפתור בחירת נתיב הקובץ הנשא חילוץ המסר

לאחר מכן, יופיעו שיטות סטגנוגרפיה המתאימות לסוג הקובץ הנבחר. יש לבחור את השיטה בה הוחבא המסר מתוך רשימת השיטות, כפי שמוצג באיור 38.



איור 38: רשימת בחירת השיטה הסטגנוגרפית לחילוץ המסר הסמוי

לאחר בחירת השיטה, יש ללחוץ על הכפתור "Extract!" שכעת הפך לפעיל. בעת חילוץ המסר החלון הראשי יהיה חסום ללחיצות ובסוף החילוץ המסר הסמוי יופיע בתיבת הטקסט המסומנת באיור 39.



איור 39: איזור הטקסט שבו מופיע המסר הסמוי

7. פירוט טכני

תוכנת StegoMaster נכתבה בשפת C++ בסביבת Windows. נכון לעת כתיבת מסמך זה התוכנה מתקמפלת עבור פלטפורמות של 64 bit הן בתצורת Debug והן בתצורת Release.

7.1. פעולות I/O

פעולות input-output במערכת StegoMaster נעשות על בסיס stream, בעצם זרמים של מידע שמייחצנים פעולות input-output על הקבצים בהם אנו מתעסקים. ה-streams מתחלקים לשני סוגים, זרמי קריאה וזרמי כתיבה כך שכל פעולות ה-I/O נעשות אך ורק דרך אותם ה-streams.

7.1.1. Stream

כפי שניתן לראות בתרשים המחלקות באיור 40, בראש היררכית המחלקות עומדת המחלקה האבסטרקטית Stream, שמייחצנת פעולות בסיסיות על streams שאינם מסווגים עדיין לקריאה או כתיבה. הפעולות הן can_read ו-can_write, ואכן, ייתכן שיהיה קיים stream שהוא גם לקריאה וגם לכתיבה, ולכן יש לממש את הפעולות הנ"ל – כמובן שהמימוש ברירת המחדל הוא להחזיר false.

המחלקות ReadStream ו-WriteStream, גם הן אבסטרקיות והן לא מציינות כיצד הפעולות הבסיסיות ממומשות – רק הנגשתן באמצעות פעולות נוחות יותר.

7.1.2. ReadStream

ראשית נסקור את המחלקה ReadStream, ובה הפעולות האבסטרקטיות הן safe_read ו-seek וייעודם כדלקמן:

- can_read – שאותה דורסים מהמחלקה Stream והיא מחזירה true
- safe_read – לבצע קריאה מה-stream ולהחזיר את כמות הבתים שנקראה בפועל (ייתכן, ובעת שגיאה – זו אינה אותה כמות שנתבקשנו לקרוא)
- seek – להזיז את סמן הקובץ לפי רצוננו

ושאר הפעולות המיוחצנות הם לצרכי נוחות המשתמש במחלקות.

- הפעולה read משתמשת ב-safe_read, אך זורקת exception אם הגודל שנקרא בפועל לא תואם לגודל המבוקש
- read_integer – עוטפת את הקריאה הרגילה כך שתחזיר מספר שלם במקום רצף בתים

המחלקה FileReadStream כאמור יורשת מהמחלקה ReadStream וממשת את פעולותיה האבסטרקטיות עבור קריאה מקבצים.

WriteStream .7.1.3

המחלקה WriteStream כמובן נועדה לבצע כתיבות לתוך streams וגם כאן, המחלקה מייחצנת מתודות שחלקן אבסטרקטיות שנועדו לדריסה וחלקן משתמשות באותן הפעולות על מנת לייחצן פעולות נוחות למשתמש.

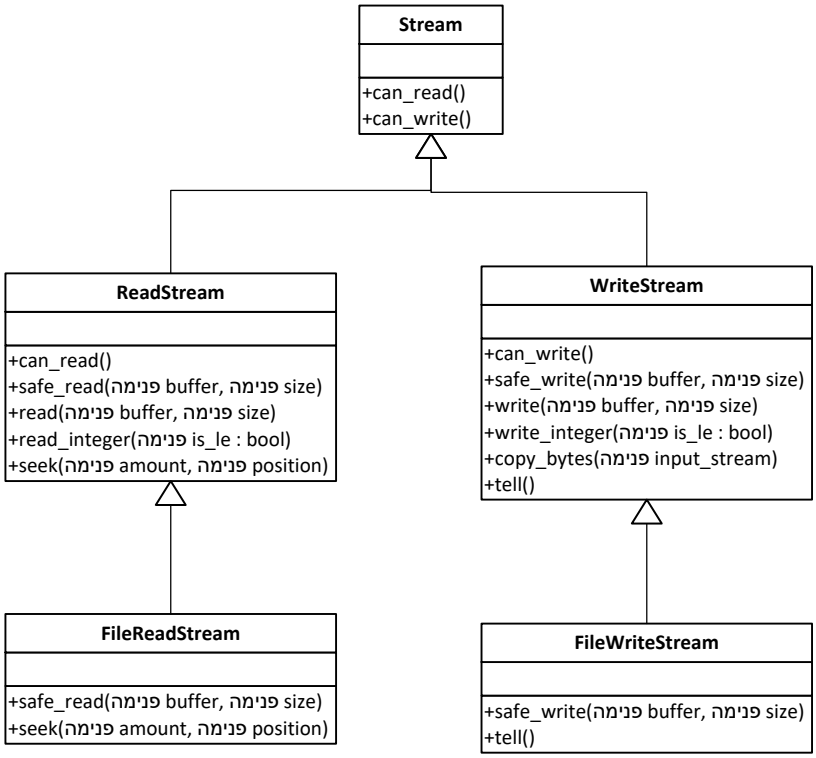
הפעולות האבסטרקטיות במחלקה WriteStream הם כדלקמן:

- safe_write – מתודה המקבלת buffer לכתיבה ל-stream ומחזירה את מספר הבתים שהיא כתבה בפועל
- tell – מחזירה את המיקום הנוכחי של הסמן בתוך ה-stream

הפעולות האחרון שממומשות ב-WriteStream:

- can_write – דורסת את המתודה של המחלקה Stream והיא מחזירה true
- write – מקבלת buffer לכתיבה ל-stream וזאת משתמשת במתודה safe_write על מנת לכתוב אותו ל-stream. במידה וכמות הבתים שנכתבה בפועל לא שווה לכמות הבתים שנדרשנו לכתוב – נזרוק exception.
- copy_bytes – מקבלת כקלט ReadStream ומעתיקה מתוכו את המידע אל ה-WriteStream.

המחלקה FileWriteStream יורשת מן המחלקה WriteStream וממשת עבורה את המתודות האבסטרקטיות כך שהן יפעלו על מנת לבצע קריאה מקבצים.

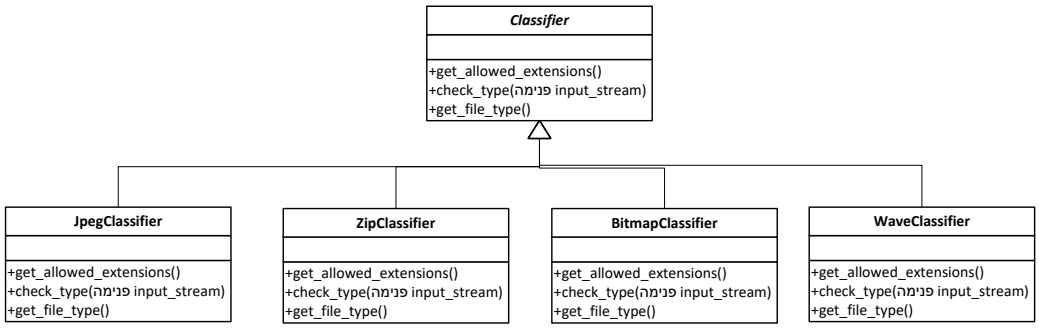


איור 40: תרשים מחלקות המתאר streams

7.2. מזהיי קבצים

צעד ראשון בהליך הסטגנוגרפיה ב-StegoMaster, בין אם מדובר בהחבאת מסרים סמויים ובין אם מדובר בחילוצם, הוא זיהוי פורמט הקובץ שעומד בפנינו. הקלט לצורך המשימה הוא תוכן הקובץ והנתיב אליו.

לכל פורמט יש סממנים ייחודיים שניתן לזהות בצורה מהירה יחסית מבלי לצלול עמוק אל תוך נבכי הפורמט. כך, נוכל לבדוק מהו הפורמט בדיוק, במעבר על כלל שיטות הבדיקה, מבלי לבזבז זמן רב.



איור 41: תרשים מחלקות של מזהיי הקבצים

המחלקה Classifier, המתוארת באיור 41, היא מחלקה אבסטרקטית שמייחצנת פעולות לדריסה על ידי המחלקות היורשות כדלקמן:

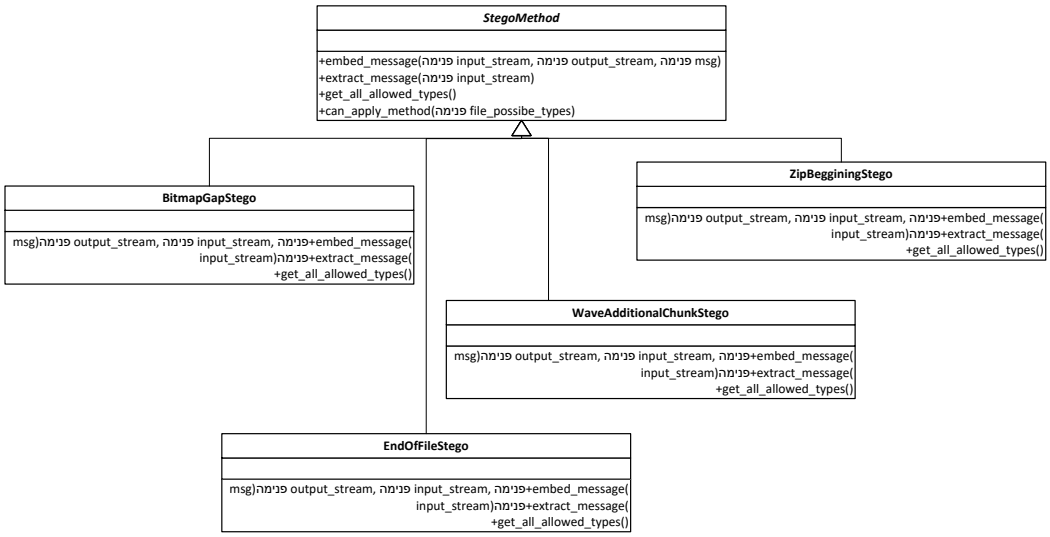
- get_allowed_extension – מתודה המחזירה רשימה של הסיומות המותרות עבור פורמט זה, לדוגמה עבור פורמט Jpeg הסיומות הן jpeg ו-jpg.
- get_file_type – המתודה מחזירה מהו הפורמט שאותו ה-classifier מזהה.

- check_type – הפעולה העיקרית של ה-classifier. מתודה זו בודקת האם ה-ReadStream שהתקבל כולל את הסממנים של הפורמט שאותו אנו בודקים.

עבור כל טיפוס שנתמך על ידי מערכת StegoMaster יש לספק classifier מתאים, אחרת המערכת לא תדע לזהות קבצים מהסוג הנ"ל.

7.3. שיטות סטגנוגרפיה

זהו חלק הארי של הפרויקט, ובמחלקות המתוארות בסעיף זה ובאיור 42 שוכנת כל הלוגיקה. שיטות הסטגנוגרפיה מייחצנות שתי פעולות עיקריות, החבאת המידע וחילוצו. השימוש בשיטות הסטגנוגרפיה כפי שיוצגו בסעיף זה, נעשה ישירות על ידי המשתמש ב-StegoMaster, לאחר שקיבל אותן מפעולות ה-API שיפורטו בהמשך.



איור 42: תרשים מחלקות המתאר את השיטות הסטגנוגרפיות

כל שיטה סטגנוגרפית בפרויקט, אם בהווה או בעתיד, צריכה לממש את ה-interface כפי שמוגדר במחלקה האבסטרקטית StegoMethod. המתודות המיוחצנות על ידי ה-interface הם כדלקמן:

- embed_message – המקבלת קובץ ומסר ומטמיעה את המסר בתוך הקובץ
- extract_message – המקבלת קובץ ומחזירה את המסר המוחבא בקובץ
- get_all_allowed_types – מחזירה רשימה של טיפוסים שהשיטה הזו רלוונטית לגביהם
- can_apply_method – הפעולה מחזירה האם ניתן להפעיל את השיטה הסטגנוגרפית על הקובץ, לפי סוגי הקבצים שנמצאו עבורו.

לאחר ביצוע פעולות API שיפורטו בהמשך, המשתמש יקבל רשימה של שיטות סטגנוגרפיות, ועליו לבחור מתוכן ולהפעיל את השיטה הנבחרת בעצמו.

API .7.4

ה-API של StegoMaster בנוי על מעט מחלקות ודרכן, ואך ורק דרכם מפעילים את השיטות הסטגנוגרפיות שתוארו קודם. בעצם התהליך בנוי על 2 מחלקות עיקריות והן:

```
FileTypeMatcher _matcher;
StegoManager _stego_manager;
```

אשר כשמן כן הן:

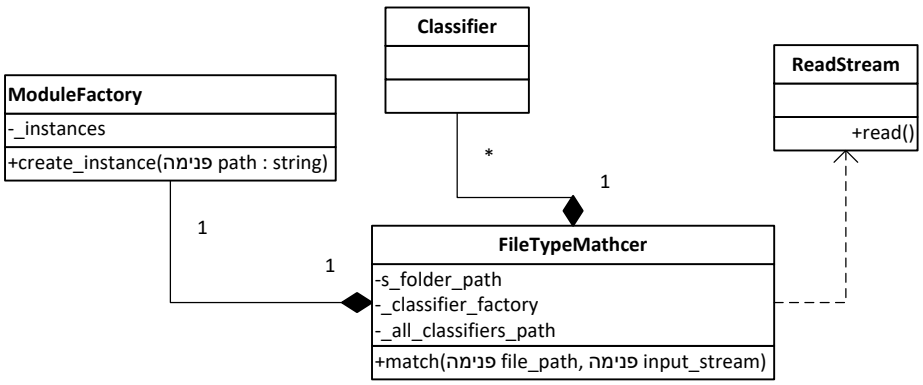
- FileTypeMatcher – אחראית על התאמה של סוגי פורמטים לקובץ נתון
- StegoManager – מתאים שיטות סטגנוגרפיות לפי רשימה של סוגי פורמטים

בתרשימי המחלקות המופיעים באיור 43 ובאיור 44 אתאר את המחלקות הנ"ל ואת המחלקות איתן הן מקושרות. המחלקות המקושרות יתוארו בהרחבה בהמשך אך מובאות כאן לצורך ראיה רחבה של התמונה.

FileTypeMatcher .7.4.1

למחלקת FileTypeMatcher, המוצגת באיור 43, יש מתודה אחת המקבלת את הנתיב אל הקובץ ואת ה-data שלו על ידי ReadStream (תזרים קריאה). פלט ההתאמה הוא רשימה של פורמטים מתאימים לסוג הקובץ הנ"ל.

החיפוש אחר הטיפוס הנכון נעשה על ידי מעבר על כל ה-Classifiers ומציאת הטיפוס נעשית על ידי וידוא צולב של הרצת ה-Classifier ובדיקת סיומת הקובץ.



איור 43: תרשים מחלקות המתאר את מחלקת ה-FileTypeMatcher

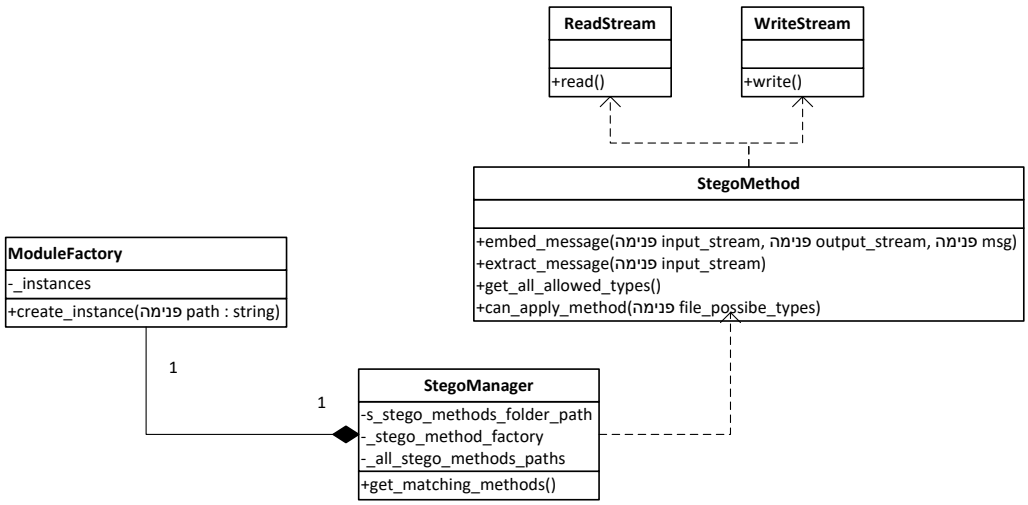
ה-FileTypeMatcher שומר אצלו _classifierFactory שמשמש כמעין cache עבור ה-Classifiers השונים. ה-Classifiers כאמור, נטענים דינאמית לזיכרון בשימוש הראשון, ונשארים בו. ה-factory מוודא שאכן הם נטענים פעם אחת – לאחר טעינה הוא שומר את המופע שלהם ומאחזר אותו בכל שימוש נוסף.

המתודה הראשית של המחלקה היא match, והיא כשמה כן היא, מקבלת קובץ כ-stream ואת הנתיב אל הקובץ ומחזירה את הפורמטים המתאימים לאותו הקובץ. טיפוס ההחזרה הוא רשימה ולא פורמט בודד לצורך הרחבת StegoMaster לפורמטים מורכבים יותר – כמו לדוגמה docx שעשוי להזדהות גם כ-zip.

StegoManager .7.4.2

StegoManager היא מחלקה שמטרתה היא להתאים שיטות סטגנוגרפיות, הן להטמעת מסרים והן לחילוץ מסרים. לכן, השימוש במחלקה זו נעשה לאחר השימוש ב-FileTypeMatcher – לאחר שמצאנו מהם הפורמטים שהקובץ שייך אליהם, אנו צריכים למצוא מהם השיטות הסטגנוגרפיות בהן ניתן להשתמש.

למחלקת StegoManager המוצגת באיור 44, כמו ל-FileTypeMatcher, יש שדה מטיפוס ModuleFactory, אך כאן, מדובר ב-cache של שיטות סטגנוגרפיות. אותו factory שומר את כל השיטות הסטגנוגרפיות בהן השתמשנו משום שמערכת StegoMaster טוענת דינאמית את כל השיטות הסטגנורפיות, ואין אנו רוצים לטעון שיטה יותר מפעם אחת. לאחר השימוש הראשון בשיטה מסוימת, ה-factory יאחזר את השיטה שנטענה כבר בעבר לזכרון.



איור 44: תרשים מחלקות המתאר את מחלקת ה-StegoManager

המתודה הראשית של המחלקה StegoManager היא get_matching_methods שמקבלת את רשימת הטיפוסים כפי שיצרה המחלקה FileTypeMatcher ומחזירה רשימה של StegoMethods. את השיטות הסטגנוגרפיות אותן מאחזרת המתודה get_matching_methods יש להפעיל באופן חיצוני, כפי שפורט קודם.

7.5. הרחבת הפרויקט

כפי שנאמר קודם, פרויקט StegoMaster נכתב תוך צפייה לעתיד באופן שניתן להרחבה. כלומר, ניתן להוסיף תמיכה בסוגי קבצים חדשים והוספת שיטות סטגנוגרפיות חדשות. הדרך לעשות זאת היא כמובן על ידי מימוש ה-interfaces השונים, קימפול המחלקות ב-VisualStudio 2017, ולשים את הספריות הדינאמיות בתיקיות הנכונות.

על מנת להוסיף סוג חדש של קובץ, יש להוסיף Classifier חדש המממש את ה-interface מן המחלקה האבסטרקטית Classifier ולשים את תוצאת הקימפול בתיקיה classifiers.

וכדי להוסיף שיטה סטגנוגרפית חדשה יש לממש את ה-interface המובא במחלקה האבסטרקטית StegoMethod, ולשים את תוצאת הקימפול תחת התיקיה stegomethods.

8. מקורות

[1] B. Dipalee“ ,New robust LSB steganographic technique for increased security ”, *International Journal of Engineering Research and General Science 3, no. 2* ,pp. 112-113, 2015 .

[2] A. Chaudhary, J. Vasavada, J. L. Raheja, S. Kumar ו M. Shama“ ,A hash based approach for secure keyless steganography in lossless RGB images ”,.*arXiv* .2012 ,

[3] T. Morkel, J. H. Eloff ו M. S. Olivier“ ,An overview of image steganography ”, *ISSA* .2005 ,

[4] M. Wakiyama, Y. Hidaka ו K. Nozaki“ ,An audio steganography by a low-bit coding method with wave files -ב ”,*Intelligent Information Hiding and Multimedia Signal Processing (IH-MSP), 2010 Sixth International Conference* , .2010

[5] E. Hamilton ,*JPEG file interchange format* .2004 ,

[6] “Bitmap File Format ”,Available: http://en.wikipedia.org/wiki/BMP_file_format.

[7] “Image Magick Homepage ”,Available: https://www.imagemagick.org./

[8] P. Kabal“ ,Audio File Format Specification .2017 ”,Available: http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html.

[9] Multimedia Programming Interface and Data Specifications 1.0, 1991 .

[10] *ZIP File Format Specification* .2004 ,

[11] E. Ruffaldi“ ,Extracting files from a remote ZIP archive 30 ”,October 2004 . Available: https://www.codeproject.com/Articles/8688/Extracting-files-from-a-remote-ZIP-archive.

[12] 010“Editor Product Page ”,SweetScape ,Available: https://www.sweetscape.com/010editor./

[13] “Rarlabs - home of WinRAR and RAR archivers ”,Available: https://www.rarlab.com./

[14] “Duqu 2.0: The Most Sophisticated Malware Ever Seen .2018 ”,Available: https://resources.infosecinstitute.com/duqu-2-0-the-most-sophisticated-malware-ever-seen/#gref.