האוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב

עבודה מסכמת בנושא

התקפות על רשת ביטקוין

עבודה מסכמת זו הוגשה כחלק מהדרישות לקבלת תואר
"מוסמך למדעים" M.Sc. במדעי המחשב
באוניברסיטה הפתוחה
החטיבה למדעי המחשב

מגיש: רועי דימינטשטיין, ת.ז. 302792361
מנחה: פרופ' אהוד גודס

יולי 2018

תקציר

בעבודה זו נסקור את מבנה רשת ה crypto-currency ביטקוין וכן את ההתקפות העיקריות אליה הרשת חשופה. העבודה תרחיב אודות האלגוריתמים התאורטיים המיושמים ברשת על מנת לאפשר את קיום המטבע, ביצוע העברות בין משתמשים, כריית ביטקוין ואנונימיות ברשת. כמו כן נרחיב אודות מימושים ספציפיים של התוכנה אותה מריצים קליינטים ווריאציות שלה לביצוע התקפות שונות על הרשת לצרכים שונים כגון מיקסום רווחים לא הוגן (selfish mining), ביצוע double spending, מניעת שירות ממשתמש ברשת ודה-אנונימיזציה של משתמש.
בעבודה זו אנחנו סוקרים במפורט באמצעות המאמרים את הנושאים הבאים: מידע כללי על ביטקוין [11, 1, 2, 3] התקפות על פרטיות ודה-אנונימיזציה של ביטקוין [10, 9], התקפות על אבטחת הביטקוין כמו Eclipse ו Selfish mining [4, 5, 6, 7, 8]

החלק היישומי של העבודה כולל פיתוח סימולציה של רשת ביטקוין, שנעשה במסגרת פרויקט הגמר, בין היתר פותחו קליינט (miner) רגיל שהוא צומת מתפקד מלא המטפל בטרנזקציות וכורה ביטקוין, מיינר שעושה את כל אותם דברים אבל ממש selfish mining - כורה באמצעות אסטרטגיה שהיא התקפה על הרשת המתוארת באחד המאמרים וכן פיתוח הסימולציה עצמה שמאפשרת לנו לבחון את האלגוריתם על קונפיגורציות שונות של רשת ביטקוין המסומלצת. לבסוף מיוצגות תוצאות שמראות את אפקטיביות התקפת ה selfish mining תחת הנחות שונות.

תוכן עניינים

תקציר 2

1. מבוא 5

1.1. מבנה הרשת ואופן פעולתה 5

1.2. מבנה העבודה 6

2. רקע טכני 7

2.1. קריפטוגרפיה 8

2.1.1 הצפנת מפתח פומבי 8

2.1.2 חתימות דיגיטליות 8

2.1.5 סכימת שמיר לחלוקת סוד 9

2.1.6 כתובות ביטקוין 9

2.1.7 פונקציות גיבוב 10

2.1.8 overview של תהליך הכרייה 11

2.2. טרנזקציות 12

2.2.1 סקריפטי טרנזקציות 14

2.2.2 סוגי טרנזקציות 15

2.2.4 טרנזקציות סטנדרטיות 17

2.3. בלוקצ'יין 18

2.3.1 proof-of-work 19

2.3.2 בלוקצ׳יין - אבטחת database מבוזר 20

2.3.3 עצי מרקל 22

2.4. ארנקים 23

2.4.1 אבטחת המפתחות הפרטיים 24

2.4.2 ארנקי offline 24

2.4.3 ארנקי web 26

2.4.5 ארנקי brain 26

2.4.6 ארנקים דטרמיניסטיים 27

2.4.7 פרוטוקול simplified payment verification 27

2.4.8 פרוטוקול התשלום 28
2.5. נושאים נוספים בכריית ביטקוין 29
2.5.1 כרייה ב pool 29
2.5.2 עמלות 30
3. התקפות על הרשת 31
3.1. פגיעה בפרטיות 31
3.1.1 הצגת עולם הבעיה 31
3.1.2 אלגוריתם המתבסס על web scraping - מאמר [9] 32
3.1.3 אלגוריתם לביצוע De-anonymization של משתמש - מאמר [10] 36
3.2. selfish mining 44
3.2.1 הצגת עולם הבעיה 44
3.2.2 אלגוריתם לביצוע selfish mining - מאמר [6] 45
3.3. התקפת eclipse 53
3.3.1 הצגת עולם הבעיה 53
3.3.2 ביצוע eclipse attack התוקף את המימוש של bitcoind - מאמר [7] 53
3.3.3 stubborn mining - מאמר [8] 60
4. מימוש השוואתי של selfish miner אל מול מיינר רגיל 72
4.1. מטרת הפרויקט 72
4.2. סוגיות מימוש והיקף העבודה 72
4.3. ממשק המשתמש 73
4.4. מיקום קבצי העבודה והוראות הפעלה 75
4.5. ערכים הנמדדים 76
selfishLeadRevenueRatio 76
selfishConnectivityRatio 77
4.7. מוטיבציה לבחירת הערכים והמדידות 77
4.8. תוצאות 77
5. סיכום ומסקנות 80
6. רשימת מקורות 81

1. מבוא

רשת ביטקוין הנה מערכת שמטרתה לממש מטבע אלקטרוני שהסחר בו יעשה באופן שהוא לגמרי decentralized, זאת אומרת, ללא trusted third party כמו בנק.

הקושי במימוש רשת כזאת טמון בטיפול בבעיית ה double spending. כדי שיהיה decentralized אנחנו צריכים איכשהו לגרום לכולם ברשת להסכים בכל פעם על המצב - כמה כסף יש לכל אחד לפני ואחרי טרנזקציה. במימוש נאיבי, נוכל לתת לכל משתמש להחזיק את רשימת כל הטרנזקציות שאי פעם היו, להגדיר לכל משתמש מפתח פרטי ופומבי וכאשר משתמש אחד ירצה להעביר כסף למשתמש אחר הוא פשוט יחתום עם הפרטי על הודעה שמספרת לכולם למי הוא מעביר וכמה, כולם יוודאו עם הפומבי שלו שזה הוא, יוודאו לפי רשימת הטרנזקציות שבאמת יש לו את הכסף לכך ויעדכנו את רשימת הטרנזקציות. במימוש זה תיהיה בעיה כאשר משתמש זדוני ינסה לקנות במקביל הרבה מוצרים שערכו של כל אחד מהם קטן מממונו אך סכום ערכם עולה על ממונו ועשוי להתבצע double spending של כסף. אלגוריתם ביטקוין פתר בעיה זאת ועל כך נרחיב.

כיום קיים מספר רב של מטבעות אלקטרוניים נוספים, רובם ככולם ממומשים בדרך זהה ומאחר וביטקוין הוא המטבע המוביל, עבודה זאת תסקור רק אותו אבל כנראה שרוב המסקנות והפעולות ניתנות להעברה גם למטבע אחר.

1.1. מבנה הרשת ואופן פעולתה

הרשת בנויה בצורת nodes כאשר כל משתתף המעוניין לסחור בביטקוין הוא צומת כזאת.

הצמתים מתקשרים אחד עם השני בצורת peer-2-peer וכך אין node "מיוחד" שפוגע בעקרון ה decentralized. מספר נושאים ומושגים שיש לדעת על מנת להבין את הבסיס:

- miner - תוכנה הרצה על מחשבים ומריצה את הלוגיקה של להיות node ברשת ביטקוין. תפקידה העיקרי (מעבר לעוד תפקידי צד רבים שהיא מקבלת עם התפתחות על הרשת כמו מניעת התקפות DOS על הרשת) הוא לבצע תהליך הנקרא mining ויוסבר בהמשך. כמו כן, כל מיינר מחזיק את ה blockchain
- blockchain - זוהי רשימה מקושרת המכילה בלוקים של טרנזקציות שעברו בין משתמשי ביטקוין. כל בלוק מכיל מספר טרנזקציות וכל טרנזקציה שאי פעם קרתה נמצאת שם. כל בלוק מחזיק מזהה שמצביע לבלוק הקודם עד לבלוק הראשון שקיבל את השם genesis.
- mining - התהליך של אימות בלוק של טרנזקציות או במילים אחרות שרשור של בלוק חדש לסופו של ה blockchain. אלגוריתם ה mining הוא הרעיון המבריק העומד בבסיסה של רשת הביטקוין, מבחינה תאורטית כמובן. יש צורך להרחיב עליו הרבה וכך יהיה בעבודה אך בגדול:

א. כאשר משתמש מנסה לבצע טרנזקציה היא משודרת ב broadcast לרשת

ב. כל מיינר אוסף מספר טרנזקציות,ובודק שהן חוקיות

ג. מוסיף טרנזקציה שאומרת שנותנים לו ביטקוין כתמריץ

ד. מוסיף את המזהה (hash) של הבלוק האחרון ב blockchain

ה. המיינר יוצר מהמידע הזה בלוק (בינארי)

ו. מתחיל למנות על nonce ולחשב hash של ה nonce משורשר לבלוק עד שיוצא לו hash שמתחיל במספר אפסים ידוע מראש. מספר האפסים מגדיר את הקושי. ברגע שמצא, הוא שולח broadcast עם הבלוק לכל הרשת

ז. שאר ה nodes מאמתים את התוצאה כשהם מקבלים את הבלוק. אם אכן כל הטרנזקציות ואלידיות (כסף שלא בוזבז כבר), הם עוברים לעבוד על הבלוק הבא

חישוב ה hash ובכך תהליך הmining הוא תהליך כבד וקשה והוא בסופו של דבר הגורם המונע את ה double spending. נקודה זאת תובהר לעומק בעבודה

1.2. מבנה העבודה

העבודה תורכב משלושת הפרקים הבאים בהתאם לפירוט שיובא להלן:

א. רקע טכנולוגי מעמיק. הפרק יציג

- מעט היסטוריה, ומוטיבציה ליצירת הרשת.
- המונחים הבאים: blockchain, miner, mining, wallet, proof-of-work, transaction block.
- יוצגו חלקים מ bitcoin core - המימוש הקונקרטי של miner ברשת (חלק זה יסקר ביתר פירוט במסגרת הפרויקט)
- הפרק יתאר לעומק את כלל האלגוריתמים בהם miner נוקט כגון אתחול blockchain, רשימת שכנים, ביצוע mining, אישור/דחייה של בלוק חדש
- ידונו נושאים תאורטיים איתן הרשת מתמודדות כגון forking של טרנזקציות (מקרים בהם ה blockchain "מתפצל" כי בפועל מיינרים שונים אישרו בלוקים שונים), מודל ה privacy וניתוח תאורטי של הסתברות ההצלחה של קבוצת תוקפים הפועלים בצורה "נאיבית" דהיינו לפי חוקי הפרוטוקול

ב. התקפות על הרשת

נסקור מספר מאמרים המתארים התקפות, לוגיות בעיקרן, על רשת ביטקוין. באופן כללי בניתוח התקפות על הרשת, מניחים כי ברשותנו הרבה פחות מחצי מהמיינרים מאחר ובמקרה שיש רוב, הכל אפשרי.

- פגיעה בפרטיות (de-anonymization) - אחת המטרות של רשת ביטקוין היא לשמור על פרטיות המשתמשים במובן שזהותם לא תיחשף. המשתמשים מזוהים במערכת

ע״י המפתח הפומבי שלהם ואת זה חייב לחשוף על מנת לאפשר פעולה תקינה של המערכת. נראה בעבודה כי הפרטיות אינה בהכרח נשמרת

- selfish mining - כפי שנראה בסיום פרק המבוא, קבוצה של מיינרים זדוניים, שיעבדו לפי פרוטוקול הרשת על מנת לייצר chain אלטרנטיבי, כנראה יכשלו. אך מה לגבי מיינרים שסוטים מהפרוטוקול? פרוטוקול כריית ביטקוין מתיימר להיות incentive-compatible במובן שעל מנת להשיג רווח מקסימאלי (פרסים במציאת nonce וכן עמלות שעליהן נדון אל מול עלות חשמל), כדאי לכל מיינר לפעול על פי הפרוטוקול. נציג ניתוח מעמיק של מספר מאמרים, המביאים מספר אלגוריתמים הסותרים טענה זאת
- eclipse attack - התקפה המתבססת על שליטה על מספר nodes הקרובים (רשתית) ל node קורבן כלשהו ברשת ביטקוין. נציג ניתוח של מאמרים המראים ביצוע התקפה מסוג זה ושימוש בה על מנת לגרום ל double spending, השפעה על הקונצנזוס ברשת, ביצוע selfish mining ועוד

ג. פרויקט. מימוש השוואתי של selfish miner אל מול מיינר רגיל

בפרק זה נממש ישום תוכנה שיתפקד כ selfish miner ברשת ביטקוין לפי אחד האלגוריתמים שנסקרו בפרק הקודם. בפרק זה תוצג עבודה מעשית שתשמש גם כפרויקט מתקדם במדעי המחשב. השלבים שיבוצעו ויוצגו בפרק זה:

- סקירת מבנה miner מנקודת מבט של הנדסת תוכנה ומימוש
- מימוש miner
- מימוש selfish miner
- פיתוח סימולציה של רשת ביטקוין וירטואלית המאפשרת לסמלץ עבודה של selfish miners יחד עם מיינרים רגילים והצגת תוצאות (כמה כל אחד הרוויח..)
- סימלוץ טרנזקציות שיכללו בבלוקים ויאומתו על ידי המיינרים ושליחתן ברשת
- איסוף נתונים (כוח חישוב ש״בוזבז״, רווחים בביטקוין)
- הצגת השוואה בין selfish miner ומיינר רגיל

2. רקע טכני

בפרק זה נסקור את הפרטים הטכניים במימוש האלגוריתם המבוזר אותו רשת ביטקוין מממשת. חלק מהנושאים יסקרו בקצרה שכן הם מוכרים ומחוץ ל scope של עבודה זאת אך עם זאת יש צורך להזכיר אותם על מנת לאפשר רציפות לוגית של העבודה וכן מעין ״יישור קו״ לגבי הידע הנדרש להבנה מעמיקה. סדר הנושאים נבחר כך שתתאפשר הבנה הדרגתית - הנושאים המתקדמים בנויים על הנושאים שלפניהם. הידע שיסקר כאן ילווה אותנו במהלך העבודה והינו הכרחי להבנת הפרקים הקשורים להתקפות וכן להבנת הנעשה בפרויקט.

יש לציין כי פרוטוקול ביטקוין מעולם לא תועד במלואו. באופן מעשי, המימוש העיקרי של Bitcoin client שנקרא bitcoind משמש כרפרנס העיקרי בקהילה. בפרק הנוכחי ננסה לתת תיאור שלם ככל הניתן לחלקים העיקריים בהסתמך על פיסות מידע שנלקחו ממאמרים, ספרים וכן מקורות מידע מקובלים בקהילה כמו הפורום bitcointalk.org, הקוד של bitcoind ודף הויקיפדיה הרשמי של ביטקוין
הרפרנס למידע הוא כל המאמרים שציינתי תחת מידע כללי על הרשת בתקציר אבל בעיקר Understanding Bitcoin_ Cryptography, Engineering and Economics - Pedro Franco

2.1. קריפטוגרפיה

2.1.1 הצפנת מפתח פומבי

הצפנת מפתח פומבי הנה הצפנה בה תהליך ההצפנה ותהליך הפענוח דורש מפתח שונה. אלגוריתם יצירת מפתחות של הצפנת מפתח פומבי מייצר זוג מפתחות פרטי ופומבי אשר קשורים מתמטית ביניהם באופן כזה שהצפנה עם אחד ניתנת לפענוח עם השני ולהפך.
כאשר Bob רוצה לקבל הודעות מוצפנות מ Alice הוא מייצר זוג מפתחות פרטי ופומבי ושולח ל Alice את הפומבי שלו. Alice תצפין הודעות עבור Bob עם הפומבי הזה ומאחר שהוא לא שלח לאף אחד את הפרטי, רק הוא ידע לפענח.
ביטקוין לא משתמש בהצפנת מפתח פומבי אבל עושה שימוש בחתימות דיגיטליות המתבססות על אותו רעיון לצורך טיפול בטרנזקציות - נושא שיסקר בפרק הבא

2.1.2 חתימות דיגיטליות

חתימה דיגיטלית הנה אפליקציה של הצפנת מפתח פומבי. זוהי בעצם דרך לוודא את זהות השולח של הודעה מסויימת. לאחר יצירת זוג המפתחות על ידי Bob ושליחת הפומבי ל Alice, ניתן באמצעות המפתחות האלה לאפשר ל Alice לוודא שכל הודעה המגיעה אליה מ Bob, אכן הגיעה ממנו ולא מגורם עוין ביניהם (man in the middle).
כאשר בוב שולח הודעה לאליס הוא יכול לצרף להודעה חתימה שהיא תוצאת ההצפנה של ההודעה עם המפתח הפרטי שלו, אליס מקבלת את ההודעה עם החתימה הזאת, מנסה לפענח את החתימה עם הפומבי שלו ולהשוות להודעה עצמה - אם התוצאה שווה אז בוודאי שבוב שלח את ההודעה הזאת כי רק הוא מכיר את המפתח הפרטי שלו.
בפרוטוקול ביטקוין נעשה שימוש בחתימות דיגיטליות. כתובת ביטקוין היא בעצם מפתח פומבי. ניתן להסתכל על המפתחות הפומביים האלה כמספרי חשבונות בנק בעוד המפתחות הפרטיים המתאימים הם חתימות של בעלי החשבון. על מנת לבזבז ביטקוין מחשבון מסויים יש לייצר טרנזקציה החתומה עם המפתח הפרטי המתאים לחשבון זה. תוכנת הארנק (פרק 2.4) מייצרת כתובת ביטקוין באמצעות אלגוריתם ליצירת מפתחות וכך כל משתמש יכול ליצור כמה כתובות ביטקוין שירצה.

נקודה אחת טכנית היא שחתימה באמצעות הצפנה של ההודעה עלולה להיות בזבזנית אם ההודעה ארוכה ולכן משתמשים בפונקציית גיבוב (hash) על מנת לקבל תמצות של ההודעה וחותמים עליו.

חתימה דיגיטלית היא שילוב של אלגוריתם הצפנת מפתח פומבי כלשהו עם סכמת חתימה דיגיטלית. קיימים מספר אלגוריתמי הצפנה פומבית בשימוש (RSA, EC..) וכמו כן מספר סכמות לחתימה דיגיטלית (אל-גמאל, RSA, DSA..)

ביטקוין עושה שימוש ב Elliptic curve כאלגוריתם הצפנה וב DSA כסכמת חתימה דיגיטלית

2.1.5 סכימת שמיר לחלוקת סוד

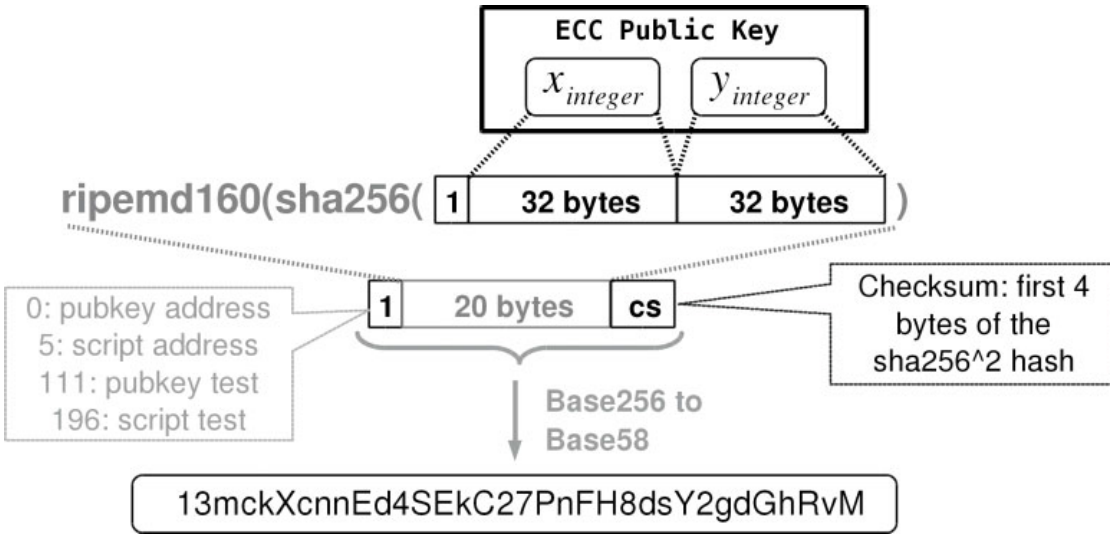
סכימה לחלוקת הסוד של שמיר היא אלגוריתם המאפשר חלוקה של סוד בין k משתתפים במובן שעל מנת לגלות את הסוד יש לאסוף את כל חלקי המידע מכל המשתתפים ביניהם חולק הסוד. אפילו אם בידינו כל חלקי הסוד למעט אחד, לא ניתן יהיה להסיק דבר על הסוד עצמו.

המימוש עצמו נעשה ע״י הגרלת פולינום ממעלה k-1 שבו האיבר החופשי הוא הסוד וחלוקת נקודה אחת (x,y) על הפולינום לכל אחד מהמשתתפים. קל לראות שעל מנת לשחזר את האיבר החופשי יש צורך בכל הערכים שחולקו.

ביטקוין עושה שימוש בסכימה על מנת לשמור cold storage של חלקי המפתח הפרטי וכמו כן לצורכי multisignature - מקרים בהם מספר גורמים חותמים על טרנזקציה מסויימת

2.1.6 כתובות ביטקוין

כתובת ביטקוין הינה ייצוג של המפתח הפומבי של אלגוריתם EC. מפתח פומבי באלגוריתם EC הוא נקודה במרחב (x,y). תהליך תרגום מפתח פומבי לכתובת ביטקוין מתוארת באיור 1 שנלקח מ [11]



איור 1 - מבנה כתובת ביטקוין

במימוש המקורי של מיינר ביטקוין נעשה שימוש ב OpenSSL על מנת להחזיק את המפתח הפומבי, נעשה שימוש ב 65 בתים, 64 כדי להחזיק את x ו y ובית נוסף שמציין את פורמט הנקודה (compresse, uncompressed). השלב הבא הוא יצירת hash של הנקודה, ראשית עם אלגוריתם sha256 ולאחר מכן עם ripemd160 כאשר sha פולט 32 בתים ו ripemd פולט 20 בתים - הפעלת 2 פונקציות הגיבוב על מנת להקטין את גודל ההודעות אך עם זאת להשאיר גודל שמבטיח collision בהסתברות יחסית נמוכה. לאחר מכן מוצמד לhash הזה checksum שהוא 4 בתים ראשונים של הפעלה כפולה של sha256 על ה hash וכמו כן בית נוסף בהתחלה המציין את סוג הכתובת (מעבר לכתובת רגילות המציינות "חשבונות בנק" קיימות כתובת נוספות שיתוארו בהמשך)
מטרתו של ה checksum דומה למטרתם של ספרות אבטחה בכרטיס אשראי - הימנעות משגיאות שיובילו לשליחת כספים לכתובות שגויות. ארנקי ביטקוין מוודאים שה checksum נכון לפני שליחתם.

על מנת לתמוך ב2 רשתות ביטקוין, אחת לפיתוח ונקראת test net והשנייה היא הרשת ה"אמיתית" ונקראת main net, בכל כתובת מסומן באמצעות בית אחד האם היא שייכת לרשת האחת או לשנייה. טרנזקציות המסומנות עבור test net לא משפיעות על המאזנים של משתמשים בmain net ולהפך. מידי פעם המפתחים של ביטקוין מאפסים את מצב הבלוקצ'יין ב test net על מנת שיהיה קל יותר לכרות ביטקוינים שם כדי לאפשר טסטים וכמו כן על מנת שאנשים לא יסתמכו על הערכים שם.
בסה"כ התהליך מפיק כתובת ביטקוין באורכים בין 27 ל 34 תווים

2.1.7 פונקציות גיבוב

פונקצית גיבוב (תיקרא פונקצית האש מעכשיו) היא אלגוריתם המקבל data באורך כלשהו ומחזיר מחרוזת ביטים באורך קבוע. ערך זה תמיד זהה עבור אותו קלט.
פונקציית האש היא בעצם מיפוי בין קבוצת הקלטים באורך כלשהו לקבוצת הפלטים מאורך מסויים כאשר ישנן מספר דרישות להן מצפים:

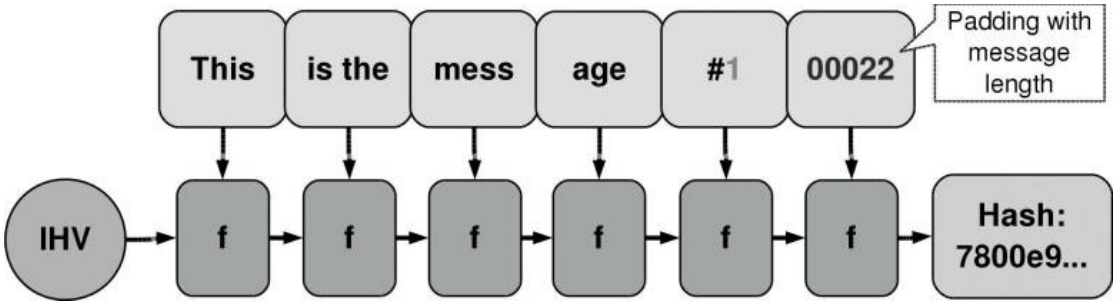
- על החישוב להיות מהיר
- שינויים קטנים בקלט יגרמו לשינויים גדולים בפלט (תוצאות הפעלת הפונקציה על שני קלטים דומים מאוד לא יחשוף כל קשר ביניהם)
- בהינתן פלט, זאת בעיה חישובית קשה לחשב מה היה הקלט
- בהינתן קלט, זאת בעיה חישובית קשה למצוא עוד קלט שהפעלת הפונקציה על שניהם תיתן תוצאה זהה
- זאת בעיה חישובית קשה למצוא שני קלטים כלשהם שהפעלת הפונקציה עליהם תיתן תוצאה זהה

ביטקוין עושה שימוש בפונקצית ההאש $(SHA256)^2$ שהיא הפעלה כפולה של פונקציית ההאש SHA256 שתוכננה על ידי ה NSA, עונה על הדרישות ומפיקה פלט באורך 256 ביט. ביטקוין משתמש בפונקציה זאת כפונקצית ה proof-of-work שלו. מאחר ועבור קלט מסויים, החישוב של ערך ההאש הוא מהיר (תכונה 1) אבל יחד עם זאת בהינתן פלט רצוי קשה מאוד למצוא קלט מתאים (תכונה 3) אז במידה ומישהו מציג בפנינו קלט כלשהו שערך הגיבוב שלו עומד בתנאים שהגדרנו מראש - כנראה שהדרך היחידה שהייתה לו להגיע לקלט הזה היא באמצעות מנייה וזאת הוכחה לכך שהוא "עשה עבודה". הקונספט של proof-of-work יוסבר יותר לעומק בפרק של תיאור הבלוקצ'יין ויחד עם הקונטקסט המתאים גם יהיה מובן יותר.

שאלה מעניינת לגבי פונקציית ההאש היא מדוע $(SHA256)^2$ ולא פשוט SHA256?

התשובה תמונה בשיטת מרקל-דמגרד ופונקציות דחיסה. שיטה זאת היא מעין מתכון לבניית פונקציות האש קריפטוגרפיות בהן פונקציות רבות משתמשות וביניהן גם SHA256. כמו כן שיטה זאת משתמשת בפונקציית דחיסה שבאופן פשטני היא פונקציה המקבלת קלט באורך קבוע כלשהו ומפיקה קלט באותו האורך אבל מסקרומבל באופן דטרמיניסטי אבל קשה לשחזור. מבנה מרקל-דמגרד מקבל קלט באורך כלשהו, "מפרק" אותו לבלוקים בגודל הקלט לפונקציית דחיסה ומפעיל אותה על כל בלוק.

במקרה של SHA256, פונקציית הדחיסה מקבל שני קלטים באורך 256 - האחד בלוק מהקלט והשני הוא ערך ההאש המצטבר עד כה ומפיקה ערך האש מצטבר חדש שיוזן כחלק מהקלט להפעלת הפונקציה הבאה וכן הלאה. איור 2 שנלקח מ [11]



איור 2 - שיטת מרקל-דמגרד

מבנה זה פגיע להתקפה שנקראת length extension attack. בהינתן ערך ההאש של הודעה m - H(m) תוקף יכול להזין אותו כ IHV ולהכניס קלט 'm כרצונו ולחשב את H(m||m') . ביצוע האש פעמיים מונע את ההתקפה הזאת

2.1.8 overview של תהליך הכרייה

לפני שניכנס לפרטים נוספים, נציג ב high level איך מתנהל תהליך הכרייה:

<u>כל מיינר בכל זמן מבצע את העבודה הבאה שנקראת כרייה</u>

- בחר מספר טרנזקציות ששמעת עליהן והוסף אליהן טרנזקציית פרס עבורך
- אמת את הטרנזקציות
- צור בלוק המכיל את הטרנזקציות, האש של הבלוק האחרון בבלוקצ׳יין ומספר nonce
- קדם את ה nonce עד שתקבל בלוק שההאש שלו מכיל מספיק 0-ים בהתחלה
- אם מצאת בלוק כזה, פרסם אותו לרשת והמשך לעבוד על הבלוק הבא

<u>תוצאות של תהליך הכרייה</u>

- Double spending נפתר בסבירות גבוהה

1. התהליך ארוך, בלוק נכרה ברשת אחת ל10 דקות, עד שיכרה הבלוק הנוסף שמכיל את ה double spend, הוא כבר כנראה לא יאושר כי הטרנזקציה הראשונה כבר נכנסה לבלוקצ׳יין
2. במקרה וכן נכנסו שני הבלוקים במקביל אז נוצר split בבלוקצ׳יין והבלוק הבא שמישהו ימצא יכריע לגבי האמת

- בעיית שינוי הקונצנזוס נפתרת

1. דורש כוח חישוב בסדר גודל של הרשת על מנת ״להשיג״ אותה בקצב ייצור שרשרת אחרת

- יש תמריץ למיינרים לעבוד על אימות וההעברת ביטקוין ולהביא לכך שהרשת מתפקדת כתשתית למטבע הדיגיטלי
- קבוצת מיינרים מרוויחה ביחס ישר לכוח החישוב שלה (מספרי החישובים של האשים לשנייה..)

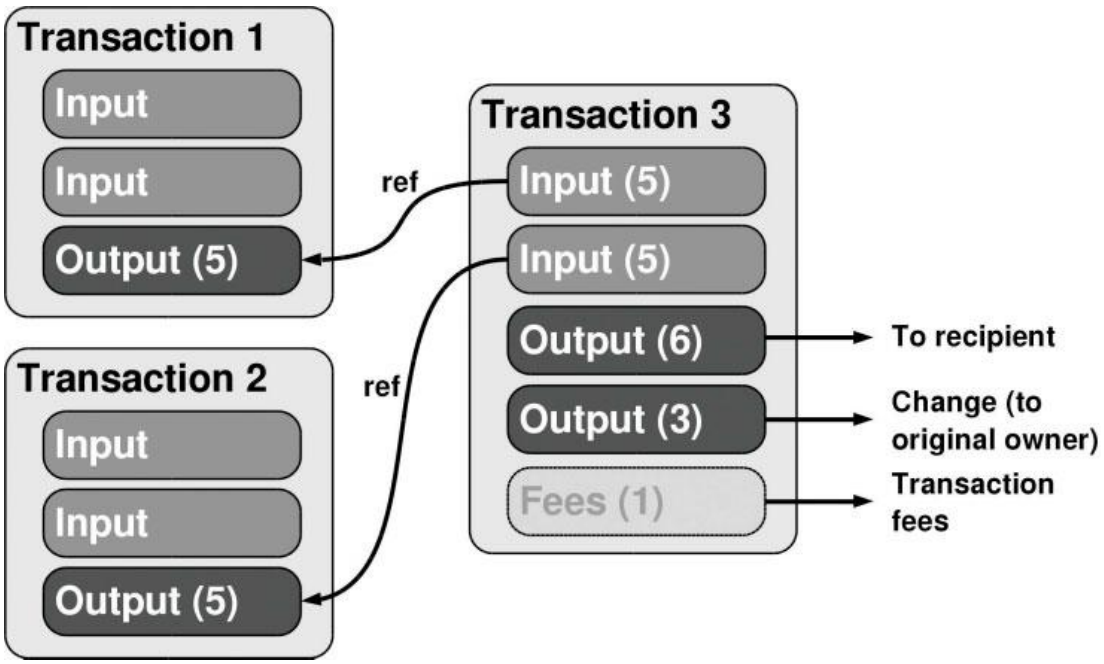
2.2. טרנזקציות

ביטקוינים אינם ״שוכנים״ במחשב של המשתמש, ביטקוינים הם רשומות במסד נתונים מבוזר הנקרא בלוקצ׳יין (יתואר לעומק בחלק הבא). בניגוד למטבעות אחרים, פרוטוקול ביטקוין לא שומר מידע על חשבונות ועל מאזנים בבלוקצ׳יין. הבלוקצ׳יין מכיל טרנזקציות. טרנזקציה היא מבנה נתונים המורכב מטרנזקציות נכנסות וטרנזקציות יוצאות (TransactionInput ו TransactionOuput. נהוג גם בקיצור לקרוא להם TxOut, TxIn) כאשר המטרה של טרנזקציה היא לבצע העברה של כספים. הטרנזקציות היוצאות של טרנזקציה יתארו כמה ביטקוין עוברים ולמי (העברה אחת עבור על טרנזקציה יוצאת) , והטרנזקציות הנכנסות הן בעצם הצבעה להעברות כסף קודמות שהגיעו אל השולח ועדיין לא בוזבזו. הרעיון הוא בעצם שימוש בביטקוינים שהגיעו אלי (טרנזקציות נכנסות) על מנת לשלוח אותם או את חלקם למשתמש אחר (טרנזקציות יוצאות).
מבחינת מימוש, טרנזקציה יוצאת מחזיקה 2 פיסות מידע - כמות וכתובת היעד. מאחר וכתובת היעד נגזרת מהמפתח הפרטי כפי שתואר בפרק הקודם, רק היעד המחזיק במפתח הפרטי יוכל לעשות שימוש במשאבים בטרנזקציה יוצאת זאת - באמצעות חתימה עליה עם המפתח הפרטי שלו ושליחתה לכתובת ביטקוין אחרת כחלק מטרנזקציה אחרת.

טרנזקציה נכנסת גם היא מחזיקה 2 פיסות מידע - רפרנס לטרנזקציה יוצאת קודמת וחתימה המוכיחה שניתן לבזבז את הטרנזקציה היוצאת הזאת.
על מנת שטרנזקציה תיהיה תקינה צריכים להתקיים התנאים:

- החתימות שתוארו צריכות להיות נוכחות ותקינות
- סכום הביטקוינים בטרנזקציות הנכנסות צריך להיות גדול או שווה לסכום הביטקוינים בטרנזקציות היוצאות
- הטרנזקציות היוצאות לא בוזבזו כבר בטרנזקציה אחרת

אין דרך לבזבז חלק מטרנזקציות יוצאות אלא את כל הסכום של הטרנזקציה היוצאת במלואו. אם הסכום של הטרנזקציות הנכנסות גדול מהסכום הנשלח, ניתן לייצר עודף בדמות טרנזקציה יוצאת נוספת.
מעבר לטרנזקציות נכנסות שהן קלטים המצביעים על משאבים קיימים של המשתמש הרוצה לשלוח ביטקוין, טרנזקציות יוצאות המסבירות לאן לשלוח וטרנזקציה של עודף כפי שתיארנו, ניתן להשאיר עמלה בצורת הפרש בין סכום הקלטים לסכום הפלטים שיהווה תמריץ לצמתים אחרים ברשת לבצע תהליך של עיבוד ואישור של הטרנזקציה.
איור 3 שנלקח מ [11] מראה דוגמא של יצירת טרנזקציה חוקית



איור 3 - טרנזקציות והיחסים ביניהן

מה שמתואר כאן הוא מצב בו המשתמש בוב רוצה לשלוח 6 ביטקוין למשתמש אליס. כמו כן, בעבר כבר עובדו ואושרו 2 טרנזקציות (Transaction 1 ו Transaction 2). ניתן לראות שבכל אחת מהטרנזקציות הקודמות, נשלחו אל בוב 5 ביטקוין. בוב יצר את טרנזקציה 3, בחר כטרנזקציות נכנסות, כקלטים של הטרנזקציה בעצם, את 2 הטרנזקציות היוצאות שהגיעו אליו. אליהן הוא הוסיף את הטרנזקציה היוצאת של 6 ביטקוין הנשלחים אל אליס. קל לראות שכרגע יש עודף של 4 ביטקוין (הוא הולך לבזבז 2 טרנזקציות של 5 ביטקוין שנשלחו אליו), הוא מייצר טרנזקציה יוצאת נוספת ששולחת אליו 3 ביטקוין מהעודף והשאר יהווה עמלה בגובה 1 ביטקוין.

לאחר מכן הטרנזקציה נשלחת לרשת, כל צומת ברשת (מיינר) שמקבל אותה מנסה לעבד ולאשר אותה. במידה והוא מצליח, הוא משדר אותה לצמתים נוספים ברשת שהוא מכיר.

על מנת לאשר את הטרנזקציה הוא מבצע את השלבים הבאים:

- מוודא שהטרנזקציות היוצאות המתוארות כטרנזקציות נכנסות (הקלטים לטרנזקציה) קיימות ושהן לא בוזבזו כבר בעבר. הוא עושה זאת על ידי שימוש במבנה הנתונים שנקרא Unspent transaction output cache או בקיצור UTXO שיוצג בקרוב
- בודק שסכום הקלטים גדול מסכום הפלטים
- בודק את תקינות החתימות. ז״א, שכל אחת מהטרנזקציות הנכנסות חתומה עם המפתח הפרטי משוייך למפתח הפומבי ממנו הגיעה הטרנזקציה הזאת

הסיבה שהוחלט על ארכיטקטורה בה לא ניתן לבזבז חלקית טרנזקציה יוצאת היא מימוש יעיל יותר של מבנה הנתונים UTXO. מבנה נתונים זה מתוחזק על ידי כל צומת ברשת ומכיל בכל רגע את כל הטרנזקציות היוצאות שעדיין לא בוזבזו מבחינתו. כאשר טרנזקציה חדשה מגיעה למיינר והוא רוצה לאמת אותה, הוא מחפש את כל מזהי הטרנזקציות הנכנסות שבה ב UTXO ומוודא שהוא מוצא את כל הטרנזקציות היוצאות המתאימות למזהים האלה וזה מאשר לו שבאמת הן עדיין לא בוזבזו.
איך מתחזקים ומעדכנים את כל מבנה הנתונים שהוא הבלוקצ׳יין נסביר בהמשך אבל כרגע נניח רק שמספר טרנזקציות מאוגדות למבנה שנקרא בלוק, הבלוקצ׳יין שהוא מבנה נתונים מבוזר מחזיק את כל הבלוקים וכל טרנזקציה שנמצאת בבלוקצ׳יין היא טרנזקציה שכבר בוזבזה ולכן המיינר דואג שהיא כבר לא תימצא ב UTXO שלו.

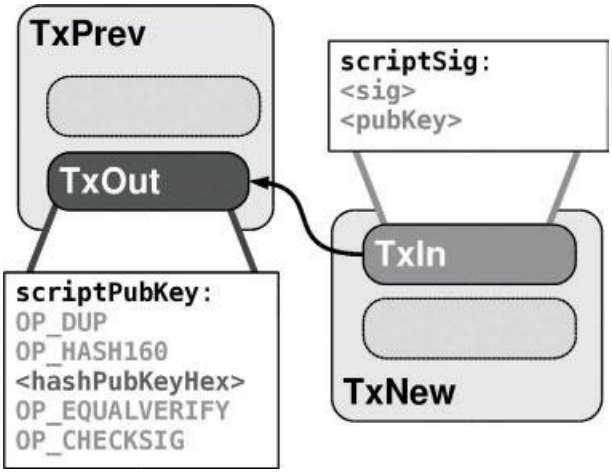
היתרון בשמירת UTXO הוא שהוא הרבה יותר קטן משמירת כל הטרנזקציות שזה מה שהבלוקצ׳יין עושה ולכן ניתן לשמור אותו ב RAM ולבצע גישה מהירה ובכך אימות מהיר של טרנזקציות חדשות שמגיעות.

במובן מסויים ניתן להגיד שהביטקוינים השייכים למשתמשים נמצאים ב UTXO.

2.2.1 סקריפטי טרנזקציות

עד לנקודה זאת הנחנו כי כל טרנזקציה יוצאת נשלחת לכתובת ביטקוין. הפרוטוקול הפועל גמיש הרבה יותר, כל טרנזקציה יוצאת מגדירה חידה מתמטית שיש לפתור על מנת לבזבז את המשאבים שלה.
החידה והפתרון שלה מיוצגים בצורת שני חלקי סקריפט. הסקריפט שיוצר את החידה הוא scriptPubKey בגלל שהוא החלק של החידה שמכיל את המפתח הפומבי. חלק הסקריפט שפותר את החידה נקרא scriptSig מאחר והוא החלק שמכיל את החתימה (signature).

זהוי הטרנזקציה השכיחה ביותר שבה האימות מסתמך על חישוב של כתובת הביטקוין.
איור 5 שנלקח מ [11] מתאר טרנזקציה נכנסת ויוצאת המתאימות לסוג הזה



איור 5 - מבנה Pay-To-Address

איור 6 שנלקח מ [11] מתאר את הפקודות המבוצעות ומצב המחסנית לאחר ביצוע כל פעולה (נזכיר ש scriptSig רץ ומיד אחריו scriptPubKey) :

Stack	Command
	<sig> <pubKey>
<sig> <pubKey>	OP_DUP
<sig> <pubKey> <pubKey>	OP_HASH160
<sig> <pubKey> <pubKeyHash>	<hashPubKeyHex>
<sig> <pubKey> <pubKeyHash> <hashPbKeyHex>	OP_EQUALVERIFY
<sig> <pubKey>	OP_CHECKSIG
1	

איור 6 - מצב המחסנית בעת ביצוע הסקריפט

לאחר ביצוע החלק של scriptSig נכנסים למחסנית 2 ערכים - הראשון היא חתימה על ההאש של הטרנזקציה עם המפתח הפרטי המתאים למפתח הפומבי של שולח הטרנזקציה, הערך השני הוא המפתח הפומבי.
לאחר מכן scriptPubKey מחשב את כתובת הביטקוין מהמפתח הפומבי (ע"י שכפול המפתח הפומבי, ביצוע OP_HASH160 שהוא אלגוריתם לחישוב הכתובת כפי שתואר

בפרק הקודם), לאחר מכן הוא משווה את הכתובת שחישב עם הכתובת של השולח ע"י השוואת שני הערכים האחרונים במחסנית. לבסוף הפקודה OP_CHECKSIG מחשבת את ההאש של הטרנזקציה ובודקת שהערך sig הוא חתימה מתאימה על ההאש הזה.
לסיכום, מבנה זה אוכף את הדרישה הבאה: על מנת לבזבז את הטרנזקציה היוצאת, יש ליצור טרנזקציה נכנסת חתומה עם המפתח הפרטי המקושר לכתובת הביטקוין hashPubKeyHex. על מנת לעשות זאת, הטרנזקציה הנכנסת צריכה לספק שני ערכים ב scriptSig:

- את המפתח הפומבי pubKey (מפתח פומבי של אלגוריתם ההצפנה EC) שקידודו ככתובת ביטקוין תפיק את hashPubKeyHex
- חתימה sig על הטרנזקציה עם המפתח הפרטי המתאים למפתח הפומבי pubKey שמוכיחה את הבעלות על הביטקוינים המתבזבזים

2. Pay-To-Public-Key

מבנה זה דומה ל Pay-To-Address רק שבמקום ש scriptPubKey יכלול את כתובת הביטקוין, הוא מחזיק ישירות את המפתח הפומבי ואז scriptSig מכיל פקודה אחת שהיא <sig> ו scriptPubKey מכיל רק 2 פקודות <pubKey> ואז OP_CHECKSIG.
בעצם מתבצע פה רק השלב האחרון של Pay-To-Address. בפועל מעט פשוט יותר אבל כמעט ולא נמצא בשימוש מאחר והפורמט הזה פגיע להתקפה תאורטית של חישוב קוונטי ובנוסף מפתח פומבי של הצפנת EC גדול מכתובת ביטקוין ולכן הטרנזקציה תיהיה גדולה יותר (למרות ש scriptSig כן יהיה קטן יותר..)

3. Multisignature

פרוטוקול ביטקוין, באמצעות סקריפטי הטרנזקציות, מאפשר מנגנון של חתימה של יותר מגורם אחד על טרנזקציה הנקרא m-of-n multisignature. הסקריפט scriptPubKey של טרנזקציות אלה יציין n מפתחות פומביים כאשר משתמש המעוניין לבזבז טרנזקציה כזאת יצטרך לספק m חתימות של המפתחות הפומביים האלה.
חתימה כזאת באמצעות מפתחות מרובים נותנת את היכולת בפועל למספר גורמים להחזיק בביטקוין באופן משותף לצורכי אבטחה או לצורכי חלוקת משאבים וזה שימושי במקרים רבים כגון העברת כספים ליעד לא אמין תוך כדי trust בצד שלישי אמין כך לדוגמא אם מערבים 3 חתימות באמצעות multisignature 2-of-3, של השולח , היעד הלא אמין וצד שלישי שסומכים עליו, אם השולח והיעד מסכימים ביניהם שניהם יכולים לחתום ואין צורך במתווך אך המקרה של אי הסכמה, הצד המתווך יכול לחתום ומספיקה רק חתימה של צד אחד נוסף, לדוגמא הצד הלא אמין כדי לאשר את הטרנזקציה או שאם המתווך מחליט על ביטול הטרנזקציה הוא פשוט לא יחתום. דבר זה פותח את הדלת לשוק של תיווך וגורמים אמינים

2.2.4 טרנזקציות סטנדרטיות

נכון לכתיבת עבודה זאת המימוש של ביטקוין מאפשר 6 סוגי טרנזקציות שונות

- pay-to-public-key
- pay-to-address
- pay-to-script-hash שבו נעשית אופטימיזציה על מנת לגרום ל scriptPubKey להיות קטן מאוד ביחס ל scriptSig מטעמי גודל הודעות אבל דומה מאוד ל multisignature. לא ניכנס לפרטיו בעבודה זאת מאחר ולא נזכיר אותו בהמשך
- multisignature
- null_data בו נעשה שימוש כאשר מנצלים את המערכת לשמירת מידע ולאו דווקא טרנזקציות של ביטקוין וגם הוא לא יסקר כאן
- non_standard שאר הטרנזקציות

באופן כללי, רק ה5 הראשונות נחשבות סטנדרטיות ורק טרנזקציות מהסוג הזה יאושרו על ידי מיינרים ויועברו ברשת. איור 7 שנלקח מ [11] מציג נתונים סטטיסטיים לגבי טרנזקציות שעברו ברשת עד לשנת 2014

Transaction type	Percentage
Pay-to-address	99.062%
Pay-to-pubkey	0.847%
Multisignature	0.056%
P2SH	0.032%
OP_RETURN	0.002%
Unknown	0.002%

איור 7 - פילוח סוגי טרנזקציות עד ל2014

2.3. בלוקצ'יין

בלוקצ׳יין הוא כנראה החידוש הגדול ביותר שביטקוין הציג, הוא הקונספט שמאפשר קיום של מטבעות המבוססים peer-to-peer ללא רשות מרכזית.

בלוקצ׳יין הוא מסד נתונים מבוזר המכיל את כל הטרנזקציות שנעשו ברשת ביטקוין מאז הקמת הרשת ובנוסף למסד הנתונים הוא מספק גם אמצעים לאבטח את המידע הזה.

הבלוקצ׳יין בעצמו לא מספק אמצעים ממשיים על מנת לענות על כל השאלות שיש לענות עליהן בזמן סביר, לדוגמא האם טרנזקציה מסויימת כבר בוזבזה בעבר ולכן על מנת לממש את פרוטוקול ביטקוין, על מיינר לפרסר את הבלוקצ׳יין ולשמור אותו בצורות שונות לדוגמא ב DB או חלקי מידע כמו ב UTXO שהוצג בקצרה בפרק הקודם.

הבלוקצ׳יין משתמש בקונספט של proof-of-work על מנת להגן עליו מפני שינויים של משתמשים זדוניים במובן שעל מנת לשנות אותו לגרום לכל הרשת להאמין בשינוי, על המשתמש הזדוני ״להילחם״ בכוח החישוב הכולל של הרשת שהושקע על מנת ליצור את הרשומות. בנוסף, התוקף יצטרך לנצח את כוח החישוב הנוכחי של הרשת שכולה עובדת על מנת ליצור רשומות חדשות המתבססות על העבודה הקודמת אותה הוא רוצה לשנות.

proof-of-work 2.3.1

באופן כללי, proof-of-work היא דרך להתמודדות נגד התקפות ברשת. דוגמא טובה לכך הלקוחה דווקא לא מהעולם של מטבעות דיגיטליים היא התמודדות נגד התקפת DOS. בהתקפה זאת מנסים לגרום לשרת להיות מוצף בבקשות ובכך לא לעמוד בקצב שירות הבקשות עד הגעה למצב שבו הוא לא יכול לשרת אף אחד, גם לא לקוחות לגיטימיים. דרך אחת להתמודדות עם התקפה זאת היא דרישה מכל לקוח המבקש שירות לשלוח הוכחה כלשהי לכך שהוא עשה עבודה , proof of work, לפני שהוא מקבל שירות. עבודה זאת יכול להיות פתרון של חידה כלשהי לדוגמא כפי שהוצג בפרק בו הוסברו פונקציות ההאש - מציאת מקור שההאש שלו עומד בקריטריונים מסויימים. באופן זה, יקח מעט זמן למשתמש שרוצה לקבל שירות על מנת לפתור את החידה אבל מאחר וכל מה שהוא רוצה זה לקבל שירות פעם אחת, הוא יכול לספוג את העלות הזאת אבל מצד שני, משתמש זדוני שרוצה להציף את השרת במיליוני בקשות יצטרך לפתור מיליוני חידות ואלא אם כן ברשותו כוח מחשוב עצום, כנראה שיהיה לו קשה מאוד לבצע את ההתקפה.

בנוסף, עבור השרת קל מאוד לבדוק את נכונות פתרון החידה ולכן קל להבין למה ההתקפה תימנע במצב הזה.

בפועל ניתן לראות מנגנונים דומים, המוכר ביניהם הוא ה captcha.

במקרה של ביטקוין, ההתקפה שמנסים למנוע היא אינה DOS במקרה הזה, אלא שינוי של הבלוקצ׳יין (שהוא בעצם שינוי של היסטורית ההעברות וההסכמה בקונצנזוס של הרשת המבוזרת בין המיינרים).

הבלוקצ׳יין מורכב מבלוקים כאשר בלוק הוא מבנה נתונים המכיל מספר טרנזקציות ונתוני מסגרת נוספים שיוזכרו בהמשך, ביטקוין משתמש ב partial hash inversion בתור פונקציית proof-of-work. פונקציה זאת דורשת שההאש של כל בלוק יתחיל במספר מסויים של ׳0׳-ים.

כפי שתיארנו בפרק הקודם, פונקציית ההאש שנעשה בה שימוש היא $(SHA256)^2$.

כל בלוק מכיל מספר טרנזקציות ונתוני מסגרת שנקבעים באופן דטרמיניסטי בעת יצירת הבלוק והם פונקציה של הטרנזקציות וזמן יצירת הבלוק אבל בנוסף, בלוק מכיל integer שנקרא nonce, כאשר מיינר כורה בלוק (מנסה לפתור proof-of-work עבורו) הוא מונה על ערך ה nonce עד שמתקבל ערך כזה שעבורו ההאש של הבלוק מכיל את מספר ה ׳0׳-ים הנדרש כעת בתחילתו.

התהליך של מציאת nonce כזה הוא קשה מבחינה חישובית אך התהליך של וידוא נכונות הפתרון לאחר מציאתו הוא קל מאחר והוא מצריך רק חישוב האש יחיד.

מספר האפסים הנדרש בכל עת עבור פתרון ה proof-of-work הוא משתנה והוא מגדיר את קושי החידה - ככל שמספר זה גדול יותר, קיימים פחות ערכי nonce מתאימים ולכן יקח זמן רב יותר על מנת לפתור את החידה (זה נכון מבחינת תוחלת, התהליך הסתברותי כמובן).

הרשת עושה fine-tuning כל הזמן לרמת הקושי על מנת לשאוף כל הזמן להיות בנקודת עבודה בה כל 10 דקות בממוצע נכרה בלוק חדש ברשת אפילו שכוח החישוב בה הולך וגדל

2.3.2 בלוקצ'יין - אבטחת database מבוזר

הבלוקצ׳יין הוא רשימה מקושרת של בלוקים שגודלת. כל בלוק מכיל מספר טרנזקציות ומקושר לבלוק הקודם בשרשרת. טרנזקציות חדשות ברשת נאספות לתוך בלוקים שלאחר ביצוע proof-of-work עליהם, משורשרים לבלוקצ׳יין.

כל הטרנזקציות שאי פעם אושרו נמצאות בבלוקים בבלוקצ׳יין, בלוקים לעולם אינם מוסרים מהבלוקצ׳יין ולכן הוא יכול רק לגדול.

נכונותו של כל בלוק מוגנת באמצעות ה proof of work שנעשה עליו.

הגופים או המחשבים שתורמים את כוח החישוב שלהם על מנת לפתור את ה partial hash inversion נקראים מיינרים. עבודתם קריטית לצורך אבטחת הבלוקצ׳יין ועל כן הפרוטוקול ״מפצה״ אותם בביטקוין על עבודתם. כל בלוק מכיל טרנזקציה מיוחדת שנקראת coinbase. טרנזקציה זאת היא הראשונה בבלוק, יש לה רק טרנזקציה נכנסת אחת והיא אינה מקושרת לאף טרנזקציה יוצאת. ה coinbase כן יכולה להכיל מספר כלשהו של טרנזקציות יוצאות שסכומן שווה לערך הפרס בתוספת לסכום העמלות בבלוק - טרנזקציות יוצאות לחשבונות של המיינר אליהם ״נשלחים״ הרווחיםשלו מכריית הבלוק. בלוק בדר״כ מכיל מספר טרנזקציות אבל הוא גם יכול להיות ריק ולהכיל רק את ה coinbase וזהו המצב שבאמת היה כשביטקוין רק התחיל - כל המיינרים התחרו ביניהם על כריית בלוקים ריקים במטרה להרוויח ביטקוין. קצב חישוב ההאשים (hash rate) של מיינר הוא כוח החישוב שלו הנמדד בהאשים לשנייה. קצב החישוב של כל הרשת הוא הסכום ועומד כיום על כ 41,765,953,155 GH/s.

בכל פעם שמיינר פותר חידת hash inversion עבור בלוק מסויים הוא מקבל את הפרס שנקרא block reward. הפרס עצמו קטן באופן אקספוננציאלי כך שסכום הביטקוין שיהיו קיימים בסוף התהליך יהיה בערך 21 מיליון. בהתחלה הפרס עמד על 50 ביטקוין לבלוק, כיום הוא כבר 12.5. נכון לכתיבת המאמר הזה קיימים כבר כ17 מיליון ביטקוין שנכרו. היסטורית כ99% מהרווחים של מיינרים הגיע מהפרסים, לאחר שיופקו כל ה21 מיליון ביטקוין האפשריים מפרסים, הרווחים של מיינרים, שהם בעצם התמריץ שלהם לבצע את העבודה, יגיעו מהעמלות.

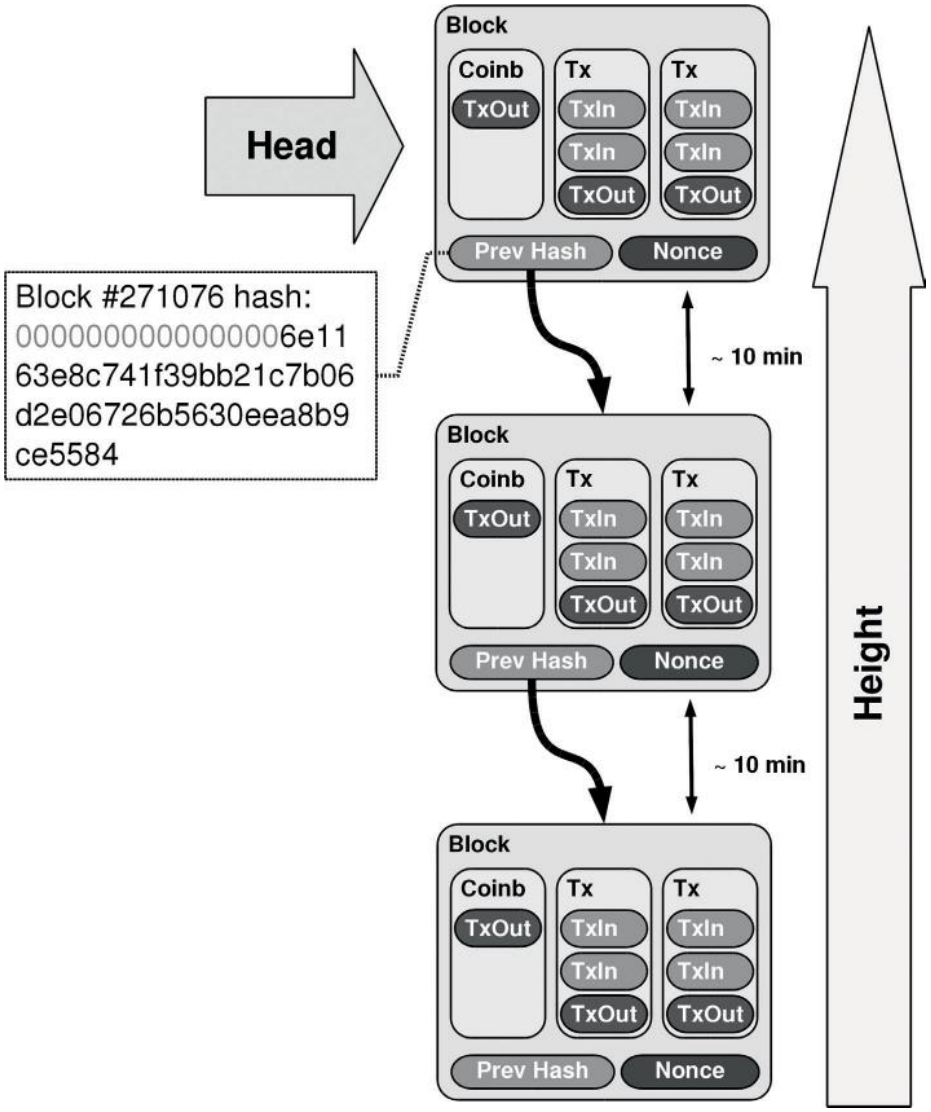
התהליך של מציאת nonce מתאים נקרא כרייה (mining) כאנלוגיה לכריית מתכות יקרות אך אנלוגיה זאת קצת מטעה מאחר ובהינתן השקעה גדולה יותר בציוד ומאמצי כרייה, קצב מציאת הזהב בעולם תגדל אבל במקרה של ביטקוין זה לא המצב מאחר וקצב הכרייה יתאזן והרווח הכולל של הרשת ישאר תמיד זהה

הבלוק בבלוקצ׳יין הקודם לבלוק חדש נקרא בלוק האב שלו, בלוקים חדשים מכילים רפרנס לאבא שלהם באמצעות כתיבת ההאש של האב כחלק מהבלוק וכך הבלוקצ׳יין שומר על סדר

כרונולוגי של הטרנזקציות. לבלוק הראשון בבלוקצ׳יין קוראים genesis block והוא נוצר על ידי הממציא satoshi (שם בדוי) ב2009.

הסדר של בלוק החלק מבלוק ה genesis (המרחק ביניהם) נקרא הגובה שלו, הבלוק האחרון בבלוקצ׳יין נקרא ה head, בלוקים חדשים מתווספים כבנים של ה head.

איור 8 שנלקח מ [11] מציג דוגמא של בלוקצ׳יין



איור 8 - בלוקצ׳יין

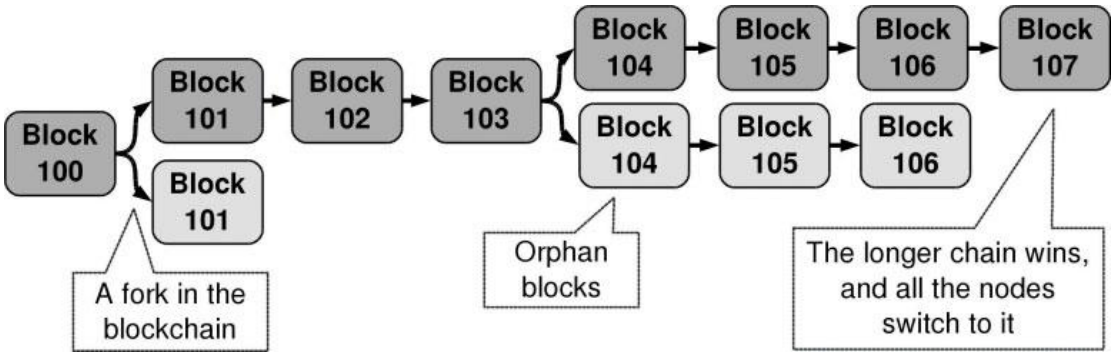
תופעה שעשויה לקרות נקראת fork ובה שני מיינרים מצליחים לכרות במקביל שני בלוקים x ו y שהם שונים ואביהם הוא ה head. במקרה זה יש בעיה מאחר ורק אחד הבלוקים יכול להיות הבן של ה head. ה״סכסוך״ הזה יפתר לבד באופן טבעי על ידי האלגוריתם ובאופן ספציפי הוא יפתר על ידי כלל מרכזי בפרוטוקול - מיינרים תמיד מקבלים את הבלוקצ׳יין הארוך ביותר כבלוקצ׳יין ה״נכון״ כאשר אורך הוא בעצם הקושי המצטבר החל מה genesis

ועד ה head, חשוב לשים לב שזה לא שקול לגובה ה head מאחר ואפשר די בקלות לייצר בלוקצ'יין ארוך מאוד וגם חוקי של בלוקים שנכרו ברמת קושי נמוכה (מעט מאוד '0'י- בהתחלה). לכן במקרה כזה של סכסוך, חלק מהמיינרים יחשבו ש x הוא ה head כרגע וחלק יחשבו ש y הוא ה head כרגע (תלוי ב propagation של ההודעות ברשת ועל איזה בלוק הם שמעו קודם) וכאשר מיינר מסויים ימצא בלוק חדש שהוא הבן של x או y הוא יפרסם אותו והוא הבלוק שיכריע. במידה ונכרה בלוק נוסף ש x הוא האב שלו, כל המיינרים שחשבו ש y הוא ה head יזנחו אותו וימשיכו לכרות בראש הבלוקצ'יין המכיל את x והבן שלו. בלוק y אז יקרא orphan.

כמובן ש fork כזה יכול, הסתברותית, להימשך גם יותר מבלוק יחיד, הנתונים מראים שכל כ50 בלוקים נוצר fork של בלוק יחיד אבל fork של יותר מבלוק יחיד הוא נדיר מאוד.

כאשר fork נפתר, הענף המכיל את הבלוקים שלא התקבלו מכילים טרנזקציות שיוחזרו אל ה pool של הטרנזקציות שעדיין לא טופלו ואל ה UTXO בהתאם.

איור 9 שנלקח מ [11] מציג מצב של fork בבלוקצ'יין.



איור 9 - התפצלות בבלוקצ'יין

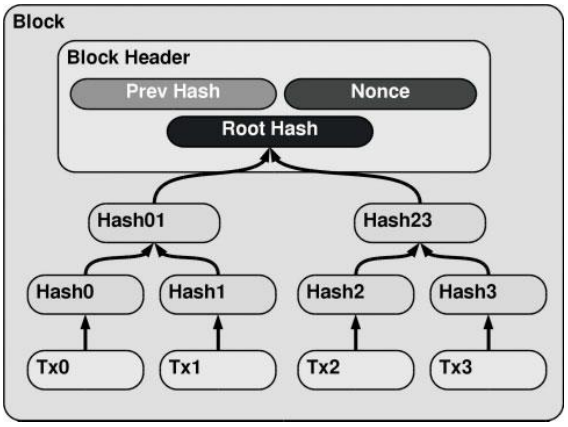
2.3.3 עצי מרקל

עד לשלב זה הנחנו כי ההאש של בלוק הוא פשוט האש על שרשור כל הטרנזקציות והאש של הבלוק הקודם. מימוש זה נאיבי ויש לו 2 חסרונות עיקריים:

- אם טרנזקציה אחת השתנתה, יש צורך לבצע את כל החישוב מההתחלה
- על מנת לאמת האם טרנזקציה מסויימת נמצא בבלוק כלשהו יש צורך בכל הבלוק

בפועל הפרוטוקול משתמש במבנה הנקרא merkle tree. מחשבים עץ בינארי שהעלים שלו הם ההאש של כל הטרנזקציות, H1, H2,…, ההאש של כל אב הוא האש של שרשור שני הבנים שלו (H01 = hash(H0 || H1) וכו עד השורש הנקרא merkle root. לאחר חישוב ה merkle root ניתן ליצור את ה block header המכיל את ההאש של הבלוק הקודם, את ה merkle root ואת ה nonce. ההאש של הבלוק יהיה ההאש של ה block header בלבד

איור 10 שנלקח מ [11] מציג את מבנה עץ מרקל של בלוק יחיד.



איור 10 - בלוק ועץ המרקל שהוא מחזיק

בהינתן המבנה הזה, ניתן למשל לוודא האם בלוק מסויים מכיל את טרנזקציה Tx3 בזמן לוגריתמי המספר הטרנזקציות מאחר והוא כבר יודע את Hash01 ואת Hash2 הוא יצטרך רק לחשב שוב את Hash3 , Hash23 ולחשב hash(Hash01 || Hash23) ולהשוות מול ה merkle root

2.4. ארנקים

הרפרנס לחלק זה הוא Understanding Bitcoin_ Cryptography, Engineering and Economics - Pedro Franco

כפי שתואר בפרקים קודמים, מטבעות ביטקוין לא "נמצאים" במחשבים של המשתמשים, הם רשומות במסד נתונים מבוזר הנקרא בלוקצ'יין ומחזיק את כל הטרנזקציות שהתבצעו ברשת. טרנזקציות אלה מספרות אילו משאבים שייכים לאיזו כתובת ביטקוין שהיא נגזרת של מפתח פומבי בהצפנת Elliptic Curve. בהינתן כתובת ביטקוין מסויימת, הגורם האוחז במפתח הפרטי המשוייך אליה, שולט אפקטיבית במשאבים המשוייכים לכתובת הביטקוין הזאת.

ארנק ביטקוין היא תוכנה המאפשרת ניהול המשאבים המשוייכים לכתובת או למספר כתובות ביטקוין. המשימות שניתן לבצע עם רוב הארנקים הן:

- תשאול הבלוקצ'יין כדי לקבל את מאזן הביטקוין הנוכחי
- יצירת כתובת חדשה (מפתח פומבי) על מנת לקבל כספים או עודף
- ממשק משתמש כדי לקבל מידע כמו כתובות חדשות או טרנזקציות חדשות באמצעים כמו QR
- שליחת ביטקוין לכתובת מסויימת. לצורך ביצוע פעולה זאת הארנק צריך להחזיק או לקבל תוך כדי התהליך את המפתח או מפתחות פרטיים איתם ניתן לחתום על טרנזקציות יוצאות שבעבר נשלחו אל המשתמש ונמצאות בבעלותו
- החזרת סטטוס לגבי עיבוד ואישור של טרנזקציות שנשלחו
- יצירת גיבוי לארנק
- שחזור ארנק מגיבוי

2.4.1 אבטחת המפתחות הפרטיים

התקנת ארנק ביטקוין ויצירת זוג מפתחות פרטי-פומבי היא המקבילה לפתיחת חשבון בנק. זוג המפתחות הזה צריך להיות קשה לניחוש. חלק מתהליך יצירת המפתחות ב EC הוא בהינתן יוצר d, בחירת מפתח פרטי A אשר ממנו מחושב המפתח הפומבי B=d*A (חשבון מודולרי). אם לדוגמא הארנק יבחר מפתח פרטי טריוויאלי d=1 אז נקבל A=B ובמצב זה תוקף יוכל לנסות התקפת brute force על מפתחות "קלים" - לחפש ב UTXO טרנזקציות החתומות עם המפתח הפומבי שהוא גלוי.
יש עדויות לכך שסריקות כאלה ודומות על מפתחות חלשים מתבצעות כל הזמן. לכן על הארנק לבחור מפתחות עם אנתרופיה גבוהה.
תוכנת הארנק מחזיקה עותק של המפתחות הפרטיים ובדרך מאפשרת למשתמש אינטראקציה איתם רק בעת בקשת גיבוי

על מנת למנוע ממשתמשים לא מורשים לגשת למשאבים אליהם נגיש הארנק, המפתחות הפרטיים המאוחסנים על הארנק בדרך כלל מוצפנים בהצפנה סימטרית עם מפתח הידוע למשתמש. כאשר הארנק מקבל פקודה לשליחת ביטקוין הוא מבקש את המפתח הסימטרי, מפענח את המפתח הפרטי ומייצר את הטרנזקציה המתאימה. שיטה זאת מספקת גם קו הגנה ראשון במקרה בו התוקף משיג עותק של הארנק אבל חוזקה תלוי בחוזק הסיסמא שנבחרה ומאחר ורוב הסיסמאות שנבחרות ע"י בני אדם לא כאלה חזקות מומלץ להחזיק רק חלק הביטקוין בארנק המחובר לאינטרנט ואת רוב הביטקוין בארנק offline

2.4.2 ארנקי offline

בדר"כ ההתקן שעליו מותקן ארנק ביטקוין מחובר לאינטרנט על מנת להסתנכרן מול רשת ביטקוין. ארנק מהסוג הזה נקראת ארנק online ומעצם היותו מחובר לאינטרנט קיימת הסכנה של חדירה וגניבת המפתחות הפרטיים.
מסיבה זאת, מומלץ לשמור את רוב הביטקוין באמצעות ארנקי offline. ארנקים אלה אינם מאחסנים את המפתחות הפרטיים במקום הנגיש לאינטרנט.
נפרט על מספר מימושים נפוצים של הקונספט.

	תיאור	יתרונות	חסרונות
מדיה חיצונית	שמירת המפתחות הפרטיים על מדיה נתיקה כמו DOK וחיבורה רק כאשר יש צורך בביצוע פעולה כמו חתימה על טרנזקציה	קל לשימוש, מוכר לרוב המשתמשים, לא דורש רכישה של ציוד מיוחד	corruption של מידע, יש לשמור עותקים של המפתחות. אם בסוף טוענים את המפתחות לארנק אונליין בעת הצורך

			אז שוב המפתחות בסיכון.
ארנקי נייר	הדפסת המפתחות על דף בפורמט שיהיה נוח לעשות לו import בשעת הצורך (לדוגמא QR)	לא קיים סיכון של corruption, אין צורך בעותקים רבים, מאפשר "לשלם" למישהו עם "כסף פיזי" כמו שטרות פשוט על ידי יצירת כתובת ביטקוין אליה יועברו ביטקוין והדפסת המפתח הפרטי	מצריך תוכנת ארנק שתומכת בפורמט. אם בסוף טוענים את המפתחות לארנק אונליין בעת הצורך אז שוב המפתחות בסיכון.
התקנים offline ייעודיים	שימוש בשני מכשירים. האחד התקן אונליין המחזיק את המפתחות הפומביים ומחובר לאינטרנט והשני התקן אופליין שלעולם אינו מתחבר ישירות לאינטרנט ומחזיק את המפתחות הפרטיים.	מאובטח מאוד, המפתחות הפרטיים לעולם אינם חשופים לאינטרנט	הרבה פחות נוח לשימוש, יש צורך בהתערבות של המשתמש על מנת להעביר את הטרנזקציה עם QR לחתימה בהתקן האופליין ואז החזרתה וכו

איור 11 שנלקח מ [11] מציג דוגמא לפלט של ארנק נייר

איור 11 - ארנק נייר

משמאל המפתח הפומבי, מימין המפתח הפרטי. ניתן לראות שהמפתחות מודפסים גם בפורמט טקסטואלי וגם ב QR

2.4.3 ארנקי web

בדומה לשירותי בנקאות ברשת שבנקים "רגילים" מספקים, גם עבור ביטקוין קיימים נותני שירות המספקים את אותה החוויה. ארנקי web הם בעצם חשבון אינטרנטי באתר של אחד מנותני השירות האלה. חשבון זה מאפשר למשתמש לנהל את כתובות הביטקוין שלו ולבצע פעולות. קיימים יתרונות ברורים מאוד כמו קלות ההתממשקות (פשוט נרשמים לאתר), ניהול המפתחות הפרטיים נעשה על ידי נותן השירות ובאופן כללי מקבלים נוחות רבה יותר וקלות שימוש במיוחד בהתחלה עבור המשתמש הלא מנוסה.
בניגוד לבנקים, לא קיימת רגולציה מספקת על נותני שירות אלה ובאופן כללי לא ניתן לסמוך עליהם ב100%. ההמלצה היא להתייחס אליהם כאל ארנק אונליין ולאחסן בהם רק כמות קטנה של ביטקוין לצורך שימוש יומיומי. מעבר לחשש שנותן השירות עצמו יגנוב את כספי הלקוח, קיים החשש שתוקף יפרוץ לאתר של נותן השירות - בשני המקרים האלה, נותן השירות לא מבטח את כספי הלקוח.

קיים סוג נוסף של ארנקי web שנקראים hybrid web wallets שם המפתחות הפרטיים נשמרים אצל הלקוח וכאשר יש צורך בחתימה על טרנזקציה, קוד צד לקוח דואג לזה. שיטה זאת טובה יותר מבחינת אבטחה מאחר והמפתחות לא יושבים על השרת אבל שוב חוזרים לנטל של ניהול המפתחות אצל הלקוח

2.4.5 ארנקי brain

כינוי לארנקים שמסתמכים על כך שהמשתמש בוחר וזוכר "במוח" שלו סיסמא. מהסיסמא הזאת יגזר המפתח הפרטי. בכל פעם שהארנק (האונליינ) יתבקש לבצע העברה ויצטרך לחתום, הוא יבקש מהמשתמש את הסיסמא, יבצע עליה את המניפולציות כדי להגיע למפתח הפרטי ואיתו הוא יחתום.

קל לראות למה שיטה זאת פגיעה להתקפת brute force. נעשו מספר ניסיונות למימוש ארנקים כאלה עם שיטות של key stretching , לדוגמא הפעלת SHA256 כ1000 פעמים על

הסיסמא כדי להגיע אל המפתח הפרטי או שימוש בפונקציות האש איטיות כמו bcrypt או scrypt אבל לאחר מספר מחקרים שנעשו נראה שקיימות המון סריקות למפתחות פומביים (כתובות) שנגזרים ממפתחות פרטיים מהסוג הזה (בסוף התוקפים גם יודעים איזה ארנקים קיימים ומה הפונקציה שהם מפעילים) ותחום התקפות ה bruteforce מפותח מאוד ויישום שיטות כמו dictionary או rainbow attack מצליחות למצוא את המפתחות הפרטיים

2.4.6 ארנקים דטרמיניסטיים

ארנק דטרמיניסטי הוא ארנק שהמפתחות הפרטיים שהוא מייצר הם תוצר של הפעלת פונקציה דטרמיניסטית כלשהי על מפתח פרטי ראשוני המשמש כ seed.
היתרונות בשיטה הזאת הם בעיקר בנוגע לגיבויים, ארנקים רגילים מצריכים גיבויים כל הזמן מאחר והארנקים מייצרים כתובות ביטקוין חדשות כמעט עבור כל טרנזקציה (לצורך שמירה על פרטיות) ולכן צריך לגבות אותם הרבה וכמו כן גודל הגיבוי הולך וגדל. עבור ארנק דטרמיניסטי יש לגבות אותו רק פעם אחת בהתחלה מאחר וכל שאר המפתחות הם נגזרת של ה seed ואין צורך גם במקום אחסון הולך וגדל.
ה seed יכול להיות סיסמא אותה בוחר המשתמש כמו ב brain wallet ואז אותם חסרונות אבטחתיים חלים גם כאן, לכן יש צורך בבחירת seed עם אנתרופיה גבוהה. לא ניכנס כאן לפרטים של ייצור המפתחות מה seed

2.4.7 פרוטוקול simplified payment verification

צומת מלא ברשת ביטקוין הוא תוכנה המכילה מיינר וארנק. צומת כזה מאמת בלוקים שמתווספים לבלוקצ׳יין, מאמת את כל הטרנזקציות בכל בלוק ותוך כדי התהליך מתחזק את מבנה הנתונים UTXO המכיל את כל הטרנזקציות שעדיין לא בוזבזו והן בעצם תיאור של הביטקוין הקיימים בכל כתובת. הארנק יכול לתשאל את ה UTXO על מנת לקבל את מאזן הביטקוין של כתובת מסויימת. אילוץ כל בנאדם שברצונו לסחור בביטקוין להתקין צומת מלא על המחשב/פלאפון שלו רק בשביל שיוכל להשתמש בארנק הוא אינו אילוץ הגיוני מ2 סיבות עיקריות

- הורדת הבלוקצ׳יין. גודלו נכון להיום 147GB
- המכשיר יצטרך לאמת את כל הטרנזקציות וזה יטחן את הסוללה של כל פלאפון

פתרון אחד יכול להיות תוכנת ארנק שמתחברת לאחד הצמתים הקיימים ברשת ומתפקדת דרכו. פתרון קיים נוסף נקרא SPV (Simplified payment protocol). ארנק המשתמש בפרוטוקול זה נקרא SPV client והוא סוג של lightweight node. SPV client שומר רק את ה header-ים של כל הבלוקים. כאשר קליינט כזה צריך לאמת טרנזקציה הוא מוריד רק את ה branch של merkle tree הנחוץ לו לאימות הטרנזקציה בבלוק (תהליך זה תואר בפרק 2.3.3).

כאשר צומת רגיל (מלא) נתקל ב fork הוא בוחר בשרשרת הארוכה ביותר, מצד שני כאשר SPV client צריך להחליט לגבי נכונות של טרנזקציה הוא יבחר בזאת שנכרו יותר בלוקים אחרי הבלוק שמכיל אותה.

חלק עיקרי בפרוטוקול SPV הוא הממשק שבין SPV client לבין צומת רגיל כלשהו. SPV client נאלץ לתשאל צומת רגיל כאשר הוא צריך לדוגמא branch של merkle tree של בלוק כלשהו או כדי לקבל טרנזקציות שנעשו וקשורות לכתובות עליהן הארנק שולט.

SPV client מייצר תקשורת עם צומת רגיל. כאשר הצומת מקבל בלוק הוא בודק את כל חלקיו (טרנזקציות, מפתחות, עץ מרקל..) מול הפילטר, מייצר filtered block עבור ה SPV client ומיידע אותו בהתאם

2.4.8 פרוטוקול התשלום

לפני שהוסדר פרוטוקול התשלום, ביצוע תשלום בין לקוח למוכר היה כרוך בשלבים הבאים:

- הזנת כתובת הביטקוין של המוכר בתוכנת הארנק
- בחירת כמות ביטקוין לשליחה
- שליחת הטרנזקציה באמצעות הארנק

תהליך זה היה פגיע ל man in the middle לדוגמא באמצעות הצגת כתובת לא נכונה של המוכר.

פרוטוקול התשלום של ביטקוין הרחיב את כתובת הביטקוין עם תוספות של שם המוכר, כמות לתשלום, והודעה נוספת שהתקבלה מהמוכר עצמו. ההודעה כולה בפרוטוקול זה חתומה באמצעות המפתח הפרטי של המוכר. שלבי טרנזקציה תחת פרוטוקול התשלום:

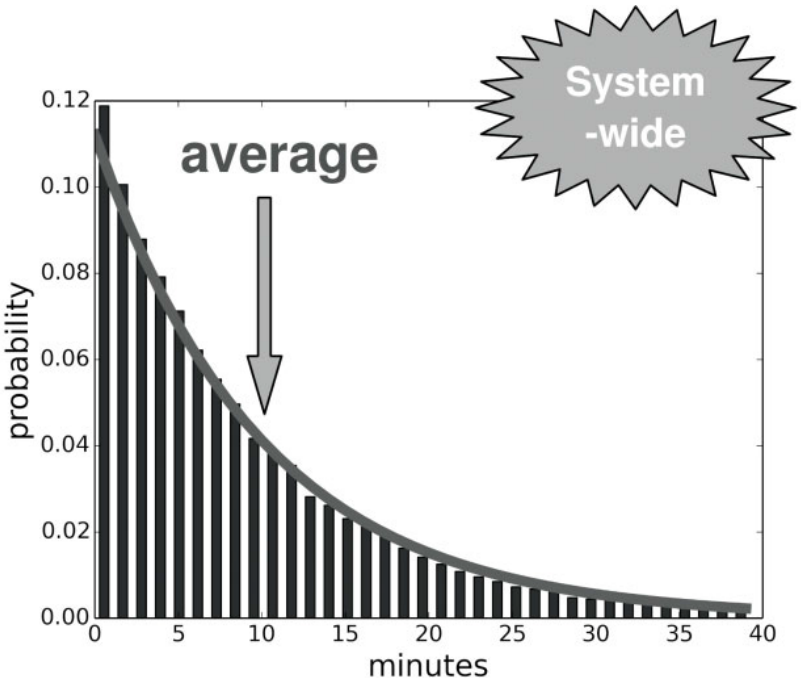
- הלקוח רוצה לשלוח תשלום, הוא לוחץ על כפתור תשלום כלשהו או יוצר טריגר באופן כלשהו
- המוכר יוצר בקשת תשלום וחותם עליה עם המפתח הפרטי של סרטיפיקט X.509 שלו. בקשת תשלום זאת מכילה את הכתובת אליה המוכר רוצה שישלח התשלום
- המוכר שולח אל הלקוח (אל תוכנת הארנק) את הבקשה החתומה, יחד עם הסרטיפיקט שלו
- הארנק בודק את תקינות החתימה ביחס לסרטיפיקט ושהסרטיפיקט חתום על ידי CA שנמצא בהתקן. לאחר מכן מציג למשתמש את פרטי העסקה
- אם המשתמש מאשר הארנק יוצר את הטרנזקציה, וחותם עליה. הטרנזקציה החתומה נשלחת בהודעה נוספת אותה שהוא שולח אל המוכר. הודעה זאת יכול להכיל פרטים נוספים כמו כתובת להחזרת עודף
- המוכר מקבל את ההודעה ומעבד אותה ומפרסם אותה לרשת ביטקוין
- המוכר שולח הודעת אישור חתומה על תשלום ללקוח

2.5. נושאים נוספים בכריית ביטקוין

2.5.1 כרייה ב pool

תחת ההנחה שהיווצרות בלוקים חדשים היא תהליך פואסוני, הזמן שבין היווצרות שני בלוקים עוקבים אמור להתפלג אקספוננציאלית. השערה זאת נבחנה אמפירית כאשר הזמנים האלה הם הפרשי ה timestamps שבין בלוקים עוקבים בבלוקצ'יין.

איור 12 שנלקח מ [11] מתאר את ההסתברות לכריית בלוק ברשת כולה כעבור מספר דקות מסויים -



איור 12 - הסתברות לכריית בלוק כתלות במספר הדקות שעברו מהבלוק הקודם

ניתן לראות כי בממוצע כל 10 דקות יכרה בלוק ברשת כולה, אולם מיינר בודד נתון לרמה גבוהה של אי וודאות לגבי הזמן שיקח לו לכרות בלוק חדש בעצמו.

לדוגמא, לפני שהחל השימוש הרחב ברכיבי ASIC המחשבים האשים בקצב גבוה מאוד וכרייה באמצעות GPU עדיין הייתה רווחית, סדר גודל של כריית בלוק ממוצעת עבור מיינר בודד המשתמש ב GPU היה כ150 ימים. כיום באמצעות שימוש ב GPU הנמצא בשימוש כמו Nvidia GTX680 הוא בערך 100 שנים.

על מנת לעזור למיינרים להתמודד עם הסיכון ולתמרץ אותם להשתתף ברשת גם אם אין בבעלותם כוח חישוב עצום, ב2010 התחילו להיווצר בריכות של מיינרים- mining pools.

בריכת מיינרים היא אגרגציה של מיינרים שתורמים את כוחם לבריכה ומתחלקים ברווחים. באמצעות יצירת בריכה, מיינרים יכולים להרוויח ביטקוין בקצב הרבה יותר צפוי מאשר כרייה בנפרד. רווחים בבריכה מחולקים באופן פרופורציונאלי לכמות כוח החישוב שהושקע בעבור הבריכה אבל כמובן מפעיל הבריכה בעצמו גובה עמלה קטנה על ההשתתפות. יתרון נוסף בכרייה כחלק מבריכה הוא שהמיינרים עצמם לא חייבים להחזיק עותק של הבלוקצ'יין

או לאמת את כל הטרנזקציות המגיעות - זה מספיק שמפעיל הבריכה יעביר להם את ה header של ה head block עליו יש לחשב proof-of-work. נכון להיום, רוב הכרייה ברשת נעשית ע״י כמות לא גדולה של בריכות איור 13 שנלקח מ[11] מציג את פילוח כוח החישוב של כל בריכה

Pool	Percentage
GHash.IO	32%
Discus Fish	14%
BTC Guild	12%
Eligius	8%
Slush	4%
Polmine	2%
BitMinter	1%
EclipseMC	1%
Unknown	26%

איור 13 - פילוח כוח החישוב ברשת בחלוקה לבריכות

קיימים קונפליקטים רבים בין מיינרים בבריכה לבין מפעיל הבריכה עצמו בנוגע ל״הבטחות״ של המיינרים לחלוק בלוקים שנמצאו ובכך את הפרסים וכמו כן ההבטחה של מפעיל הבריכה לחלק באופן שווה את הרווחים בין המיינרים והם נפתרים באמצעות פרוטוקלי כרייה בבריכה שונים אליהם לא ניכנס בעבודה זאת.

כמו כן יש להצביע על העובדה שיישויות מהסוג הזה נוגדות את העקרון הבסיסי והחשוב העומד בבסיסו של ביטקוין שהוא הימנעות מסמכויות ״שולטות״ שכן בריכה השולטת באחוז גדול מאוד מכוח החישוב ברשת מהווה איום בצורת התקפות אפשריות ופגיעה באופי המבוזר של הרשת. בפרקים הבאים נראה התקפות המסתמכות על מבנים מהסוג הזה

2.5.2 עמלות

בפרקים הקודמים הוזכרו מספר פעמים עמלות, כאשר מיינר אוסף טרנזקציות לכדי בלוק, כל טרנזקציה עשויה להעניק לו עמלה שהיא ההפרש בין הטרנזקציות הנכנסות לטרנזקציות היוצאות אבל לא הוסבר מה היא הסיבה שארנק יבחר להשאיר עמלה כזאת או אחרת, מדוע שישאיר עמלה בכלל?

לכל בלוק יש מגבלת גודל שעומדת על 1MB. גודל טרנזקציה תלוי במספר הטרנזקציות הנכנסות והיוצאות שהיא מכילה, גודל של טרנזקציה המכילה טרנזקציה נכנסת אחת ויוצאת אחת הוא בערך 157 בתים, כל נכנסת נוספת כ 113 בתים ויוצאת כ 34 בתים.

מיינרים צריכים להחליט אילו מבין הטרנזקציות ששמעו עליהן להכניס לבלוק, זוהי בעיית אופטימיזציה על גודל הבלוק.

מיינרים בדר״כ משתמשים באלגוריתם חמדני כדי לפתור את הבעיה הזאת כאשר הטרנזקציות ממויינות בסדר יורד והמדד הוא יחס העמלה-לבתים. בנוסף מיינרים מנסים להתחשב בבעיה שבלוק גדול מידי הוא גם לא אופטימלי כי יקח לו יותר זמן לפעפע ברשת ובזמן הזה מיינר אחר עשוי למצוא גם הוא proof-of-work לבלוק אחר ולהשיג אותם בדרכם להשגת הפרס ואז עבודתם תיהיה לשווא.

כחלק מהפרוטוקול קיימת גם עמלת מינימום שתפקידה למנוע התקפות DOS על הרשת שיציפו את הרשת בטרנזקציות נטולות עמלה.

כל טרנזקציה המועמדת להיכנס לבלוק הבא מקבל עדיפות. איור 14 שנלקח מ [11] מציג את הנוסחא לחישוב העדיפות

$$Priority = \frac{\Sigma InputValue \cdot InputAge}{TxSize}$$

איור 14 - נוסחת חישוב עדיפות טרנזקציה

סכום הקלטים של הטרנזקציה כפול הזמן בו הטרנזקציה מחכה בתור (על מנת למנוע מצב של טרנזקציה שלעולם לא תיכנס לאף בלוק) , חלקי גודל הטרנזקציה.

תחת מודל זה, משתמש שמעוניין שהטרנזקציה שלו תעבור מהר יותר יכול להשאיר עמלה גבוהה יותר. בנוסף גם טרנזקציות עם עמלות קטנות ילקחו בחשבון אך לאחר זמן רב יותר וכמו כן למיינרים יש תמריץ נוסף להכליל גם טרנזקציות כאלה משום שעד שהן לא נכנסות לבלוק, הן תופסות מקום בזכרון של המיינר

3. התקפות על הרשת

בפרקים הקודמים נסקרו פרטי המימוש של רשת ביטקוין ואופן פעולתה. בפרק הבא נציג מספר התקפות אפשריות על הרשת ונסקור מאמרים אקדמאיים המתעמקים בכל אחת מהתקפות אלה

3.1. פגיעה בפרטיות

3.1.1 הצגת עולם הבעיה

רשת ביטקוין היא רשת peer to peer המהווה פלטפורמה מבוזרת למסחר במטבע הדיגיטלי ביטקוין. הרשת מתחזקת מבנה נתונים מבוזר, הבלוקצ׳יין, שבו רשומות כל הטרנזקציות שנעשו מאז הקמתה. הבלוקצ׳יין הוא פומבי וחשוף לכולם. מטבעות ביטקוין אינם משוייכים לאנשים או לכתובות ip מהן נשלחו הטרנזקציות אלא לכתובות ביטקוין שהן מפתחות פומביים כפי שהוסבר בפרקים קודמים ולכן את הטרנזקציות ואת המפתחות כולם יכולים לראות. הדעה הרווחת הקיימת היא שביטקוין באופן אינהרנטי מבטיח אנונימיות של המשתמשים ולא ניתן להפיק שום מידע לגבי המשתמש מתוך המידע הנתון בבלוקצ׳יין. מספר מחקרים מראים כי הדעה הרווחת הזאת היא שגויה. נציג שני מאמרים שמראים דרכים שונות לחשיפת מידע לגבי הישויות העומדות מאחורי הטרנזקציות

3.1.2 אלגוריתם המתבסס על web scraping - מאמר [9]

מבוא

במאמר זה מוצגת טכניקה העושה שימוש במידע הפומבי הנמצא בבלוקצ'יין וכמו כן במידע פומבי הנמצא בפורומים הקשורים לביטקוין על מנת לבצע דה-אנונימיזציה. האלגוריתם המוצג ראשית מנסה לשייך טרנזקציות פומביות ליישויות (המידע הפומבי חושף לאיזה מפתח פומבי שייכת טרנזקציה מסויימת אבל לא ידוע אילו מפתחות פומביים שייכים לאותו המשתמש) ולאחר מכן האלגוריתם מנסה להשתמש במידע מהפורומים על מנת להתאים זהות כלשהי בדמות שם משתמש לדוגמא, ליישויות שנוצרו ובכך בעצם להצליח במשימה שהיא קשירת שם "אמיתי" לטרנזקציות ביטקוין. שם "אמיתי" יכול להיות כל מזהה בלתי תלוי של אותו המשתמש. נתאר את השלבים לביצוע האלגוריתם

שלב 1 - השגת המידע

1. פרסור ה blockchain -

כותבי המאמר השתמשו בכלי שנקרא Bitcoin Armory. כלי חינמי שמתממשק לclient הביטקוין הפופולרי ביותר שנקרא satoshi client. באמצעות כלי זה הם הורידו את כל הבלוקצ'יין, כתבו מספר מחלקות פרסור המרחיבות את Armory, עברו על כל הטרנזקציות ופרסרו מכל טרנזקציה את המידע הרלוונטי לצורך האלגוריתם

2. מידע מ web scraping -

תוקף המחפש מידע שעשוי לעזור לו בזיהוי משתמשי ביטקוין יכול להיעזר בכל המידע הנמצא בפורומים, אתרי תרומות בביטקוין ורשתות חברתיות בהן עשויות להימצא כתובות ביטקוין פומביות (מפתחות פומביים) באופן מכוון או לא מכוון. המידע שסביר למצוא באופן כללי בעל אחד משני מבנים, הראשון הוא tuple מהצורה (שם "אמיתי", כתובת ביטקוין) לדוגמא כאשר מישהו חותם על פוסט עם המפתח הפומבי שלו והשני הוא (שם "אמיתי", מידע כללי לגבי טרנזקציה) לדוגמא כאשר בוב כותב לאליס "העברתי אליך 100 ביטקוין אתמול בצהריים".

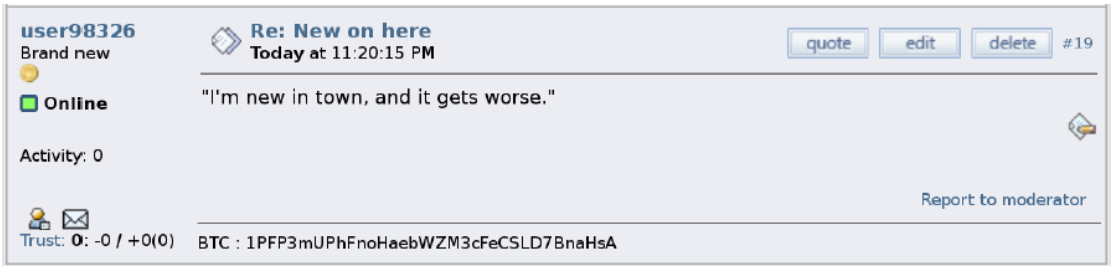
איור 15 שנלקח מ [9] מציג דוגמא להדלפה לא מכוונת:

איור 15 - הדלפת זהות לא מכוונת

כאן ניתן לראות משתמש השואל שאלה לגבי קוד שאמור לבצע טרנזקציה ובמקרה הוא חושף את המפתח הפומבי שלו בתוך ה code snippet שהוא מפרסם.

משתמשים רבים, במיוחד המשתמשים שמעוניינים לדחוף את ביטקוין קדימה, מנסים לעודד ביצוע של טרנזקציות. אחת הדרכים המקובלות לעשות זאת היא לצרף כתובת ביטקוין כחתימה לפוסטים או אימיילים. במיוחד בפורומים הקשורים לביטקוין, משתמשים רבים מפרסמים מידע או מדריכים לשימוש במטבע והם חותמים עם המפתח הפומבי שלהם כך שניתן לשלם להם "טיפ" בביטקוין. דרך פעולה זאת יצרה וקטור התקפה לדה-אנונימיזציה.

איור 16 שנלקח מ [9] מציג דוגמא לפוסט כזה:



איור 16 - הדלפת זהות מכוונת

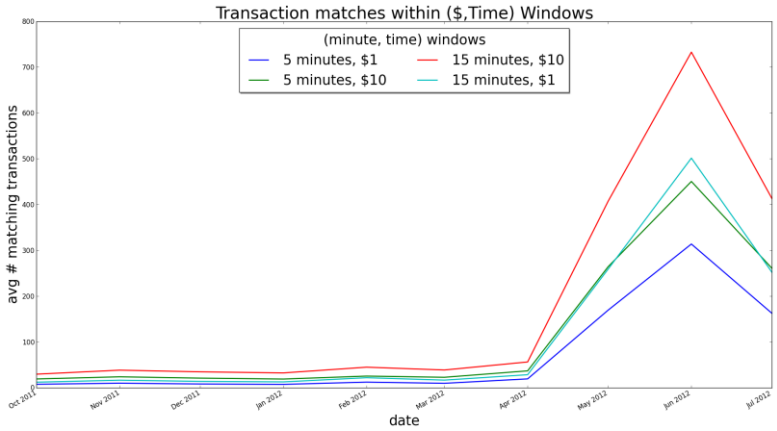
כותבי המאמר השתמשו בחבילת פייתון בשם scrapy לביצוע web scraping. הם כתבו spider שמטייל על דפי הפורום של bitcointalk.org ב BFS ומחפש כתובות ביטקוין (ביטוי רגולרי שעונה על '1.{26,33}') ומייצר זוגות (שם משתמש, כתובת ביטקוין). לאחר מכן הם בדקו שאכן באמת כתובת שנמצאה היא כתובת ביטקוין.

כעבור הרצת הקוד לכ30 שעות, נסרקו כל הפוסטים לעומק של 4 בפורום, 44086 דפים, 89088 פוסטים, 2322 משתמשים שונים ו 2404 כתובות ביטקוין שונות.

<u>שלב 2 - טביעת אצבע של טרנזקציות</u>

בשלב זה מנסים לקחת את המידע הלא מדויק שנאסף על טרנזקציות (בוב אומר לאליס שהוא העביר לה אתמול בצהריים 100 ביטקוין) ולנסות להתאים אותו לטרנזקציות ממשיות בבלוקצ'יין.

אם ניקח את המידע של אליס ובוב כדוגמא, ניתן לבחון את הקושי במציאת הטרנזקציה. נניח שערך הביטקוין חווה תנודות בסדר גודל של 1 דולר אתמול ושהזמן "אתמול בצהריים" מתכוון ל12:00 פלוס-מינוס 5 דקות אז ניתן לבדוק את כל הטרנזקציות שעונות על הטווח [99$, 101$] וגם [11:55 AM, 12:05 PM] . כותבי המאמר ביצעו מספר בדיקות עם גדלי חלונות שונים על מנת לראות באופן כללי כיצד מתנהגת הפונקציה של מספר הטרנזקציות שעונות על גדלי חלונות שונים בתאריכים שונים. איור 17 שנלקח מ [9] מציג את התוצאות



איור 17 - סיווג כתובות כתלות בתאריך הטרנזקציה

ניתן לראות באופן בולט לעין שעם הזמן ביטקוין נעשה פופולרי יותר, נעשות יותר טרנזקציות באופן כללי וכך גם טרנזקציות שעונות על החלונות.

בכל מקרה, קיים כאן מידע נוסף (שמתווסף גם ל web scraping) ומאפשר להצמיד מידע נוסף למידע הקיים בביטקוין הקשור לזהות המשתמשים.

<u>שלב 3 - יצירת Transaction Graph ו User Graph</u>

בשלב הבא האלגוריתם יוצר 2 גרפים מהמידע שנאסף, הגרף T שמכיל את המידע על הטרנזקציות שנעשו והגרף U שמכיל מידע על המשתמשים והטרנזקציות ביניהם

1. Transaction Graph -

תפקידו של גרף זה הוא לתת אינטואיציה לגבי זרימת הביטקוינים בין כתובות ביטקוין שונות לאורך זמן. את הגרף בונים מכל הטרנזקציות שהוצאו מהבלוקצ'יין בתהליך עד כה.

זהו גרף מכוון המכיל צומת עבור כל כתובת ביטקוין (מפתח פומבי). קיימת קשת בין כתובת X לכתובת Y עבור כל טרנזקציה שהעבירה ביטקוינים מX ל Y.

מאחר וגם משתמש המקור וגם משתמש היעד יכולים לייצר מפתחות חדשים עבור כל טרנזקציה, כתובות רבות עשויות להראות רק פעם אחת בגרף הטרנזקציות.

לצורך הניסוי, יצרו 2 גרפים כאלה - גרף המכיל טרנזקציות שנעשה בטווח של 24 שעות, גרף זה הכיל כ90000 טרנזקציות, 80000 כתובות ביטקוין (צמתים בגרף). בנוסף גרף המכיל טרנזקציות של 7 חודשים, המכיל כ 1,700,000 טרנזקציות, החל ממרץ 2013 ועד אוקטובר 2013 במטרה למצוא קישור למקרה ידוע ברשת ביטקוין במסגרתו ה FBI סגרו אתר שהואשם במכירת סמים באמצעות ביטקוין. ה FBI החרים את הביטקוין שהיה ברשותם וכותבי המאמר חיפשו "אזכור" כלשהו לטרנזקציות האלה.

2. User Graph -

באמצעות גרף הטרנזקציות ניתן לבנות את גרף המשתמשים. גרף המשתמשים הוא גרף מכוון שתפקידו לתת אינטואיציה לגבי זרימת הביטקוינים בין משתמשי ביטקוין שונים לאורך זמן. כל צומת בגרף מתארת משתמש (יישות "אמיתית" כלשהי), כל קשת בגרף מצומת X ל Y מתארת טרנזקציה נכנסת ויוצאת של טרנזקציה מסויימת שעברה בין

משתמש X למשתמש לY. לא ניתן ליצור את הגרף באופן טריוויאלי, המידע הנתון באופן טריוויאלי יכול לעזור על מנת להרכיב גרף בו כל צומת היא המפתח הפומבי אבל על מנת לייצר את גרף המשתמשים, יש צורך לאגד ביחד צמתים המתארים מפתחות פומביים של אותו המשתמש. הקושי בכך הוא שמפתחות פומביים היא הדרך של ביטקוין להגן על האנונימיות של המשתמשים. למעשה, זה נחשב good practice ליצור מפתחות חדשים לכל טרנזקציה.. לכן זה בלתי אפשרי ליצור גרף משתמשים מושלם אם נתון אך ורק המידע הפומבי. אולם, כפי ש satoshi עצמו תיאר במאמר שלו, מאחר וקיימות טרנזקציות עם multi-input, טרנזקציות אלה חושפות שבהכרח הבעלים של כל האינפוטים לטרנזקציה מסויימת הוא אותו הבעלים.

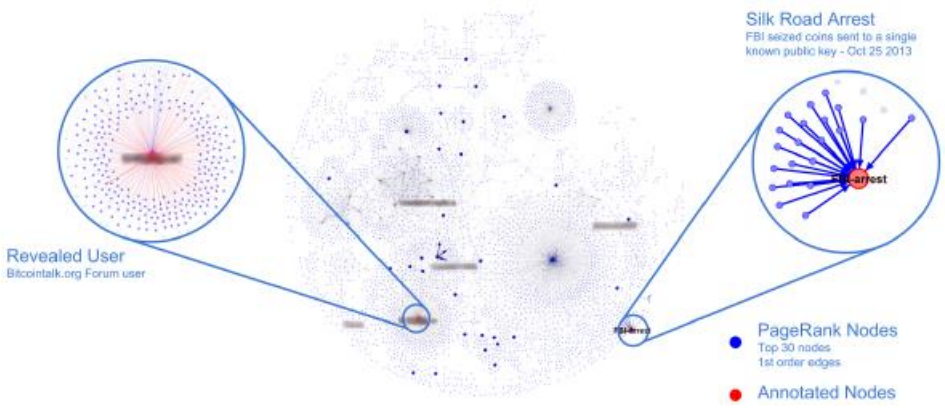
בשימוש באינטואיציה זאת תוכנן האלגוריתם הבא לבניית גרף המשתמשים:

- בנה גרף לא מכוון, עבור כל כתובת (מפתח פומבי) בנה צומת
- חבר בקשת זוג צמתים אשר המפתחות הפומביים אותם הם מייצגים היו input לאותה טרנזקציה ולכן שייכים לאותו משתמש
- מצא את הרכיבים הקשירים המקסימליים בגרף. כל אחד מהם מתאר משתמש במובן שכל הצמתים ברכיב זה הם מפתחות פומביים השייכים לאותו המשתמש

גרף המשתמשים עבור הטרנזקציות שנאספו מ24 שעות לא יצא מאוד אינדיקטיבי מאחר וקיימת סבירות גבוהה שמספר מפתחות פומביים הופיעו לפני 24 השעות האלה או אחרי ולכן מפתחות אלה לא קיימים שם על מנת לקשר בין הכתובות. גרף זה הכיל כ 54941 משתמשים וכ 90000 טרנזקציות (קשתות)

<u>שלב 4 - דירוג צמתים באמצעות Page Rank</u>

כותבי המאמר ראו קשר ישיר בין גרף המשתמשים לגרף אותו בונים מנועי חיפוש. רוב מנועי החיפוש, ביניהם גוגל, משתמשים ב PageRank כמטריקה למדידת טיב אתרים בהתבסס על החשיבות שלהם. באופן אינטואיטיבי האלגוריתם מעדיף צמתים שיותר קל להגיע אליהם, או במקרה של ביטקוין, צמתים שמראים מספיק כניסות ויציאות של טרנזקציות על מנת לסמן אותם כחשובים. כותבי המאמר השתמשו באלגוריתם זה לאחר הרצת השלבים על הטרנזקציות של 7 החודשים. הצמתים החשובים הם שרטטו כגדולים יותר בגרף. איור 18 שנלקח מ [9] מציג את התוצאה



איור 18 - דה אנונימיזציה וסיווג לקלאסטרים

כפי שניתן לראות, האלגוריתם ביצע דה-אנונימיזציה למספר משתמשים "רציניים", בין היתר נמצאו וקושרו הטרנזקציות של ה FBI במקרה מול Silk Road, זוהו משתמשים ספציפיים מהפורום bitcointalk שביצעו העברות לכתובת נוספת שזוהתה כ SatoshiDICE שהוא אתר הימורים, ואף העברות ממשתמשים שזוהו לכתובת של wikileaks

3.1.3 אלגוריתם לביצוע De-anonymization של משתמש - מאמר [10]

<u>מבוא</u>

מאמר זה מציג אלגוריתם גנרי לביצוע דה-אנונימיזציה של משתמשים ברשת ביטקוין באמצעות שיוך המפתחות הפומביים שלהם לכתובות IP.

נסביר כי שיוך מפתח פומבי ל IP יכול לשמש תוקף למספר שימושים:

1. שימוש "טיפש" - IP יכול לקשר בכל מיני דרכים לאיזור גאוגרפי. אם נדמיין שיש גורמים שמחזיקים ביטקוין בשווי מיליוני דולרים - יש סיבה טובה לחפש איפה פיזית יושב המחשב עם המפתח הפרטי שלהם
2. תקיפה בסייבר - לדוגמא, אם אני יודע את ה IP הנוכחי של משתמש בעל הרבה ביטקוין ואני גם יודע שהוא גר בישראל, אני יכול לפרוץ אתר שהוא בסבירות גבוהה גולש אליו (נגיד אתר של קהילת הביטקוין בישראל) ומשם לתקוף אותו כשהוא גולש (חולשות פומביות). ברגע שאני על המכונה המבצעת את הטרנזקציות, בסבירות גבוהה אני יכול לגנוב את המפתח הפרטי

קיימות דרכים נוספות ורבות. בכל מקרה, חשיפת מידע כזה היא חולשה בביטקוין וכל פיסת מידע מספקת לתוקף משטח עבודה רחב יותר

השיטות המוצגות במאמר זה אינן מסתמכות על מידע המגיע ממקור נוסף (כפי שראינו בדוגמא הקודמת של web scraping). במקום, השיטה מסתמכת על פרצות שנובעות מהמימוש הספציפי של של bitcoind, הקליינט הפופולרי של ביטקוין.

כותבי המאמר מראים איך ניתן לתקוף peers "רגילים" להם הם קוראים servers מאחר והם יכולים לקבל חיבורים נכנסים וגם יוצאים, peers אשר נמצאים מאחורי NAT ולכן יכולים לתחזק רק חיבורים יוצאים ולהם הם קוראים clients ואפילו peers שעוברים דרך TOR (!!).
קיימים כ 10000 שרתים ברשת וכ 100000 קליינטים.

מקורן של מרבית החולשות שהאלגוריתם מנצל היא במימוש ה peer to peer ופרוטוקול ה networking (דרך העברת ההודעות). ראשית נתאר חלקים בפרטי המימוש הנחוצים על מנת להבין את ההתקפה. לאחר מכן נציג את אלגוריתם ההתקפה, מוטיבציה לנכונותו ותוצאות

<u>פרטי מימוש של רשת ה peer to peer</u>

<u>עובדות כלליות והגדרות</u>

- צמתים ברשת מתחברים אחד לשני באמצעות חיבור TCP שאינו מוצפן ואינו מכיל שום אותנטיקציה. כל צומת מתחזק רשימת כתובות וחיבורים
- לצורך התמודדות עם DOS, כל צומת מתחזק penalty score עבור כל IP שלו. כאשר צומת מקבל 100 הודעות שאינן well formed מ IP מסויים, הוא מחרים אותו ל24 שעות
- קיימים צמתים שמתחזקים חיבורים יוצאים וגם נכנסים (servers) וקיימים צמתים מאחורי NAT בעלי רק חיבורים יוצאים (clients)
- כל צומת מנסה לתחזק 8 חיבורים יוצאים ו117 חיבורים נכנסים. אם אין צורך בשינוי בשל איבוד חיבור, אותם 8 החיבורים ישארו לנצח
- צומת מסכים לקבל מספר כלשהו של חיבורים מאותו ה IP. זה לא משנה לו

<u>אלגוריתם ליצירת החיבורים</u>

האלגוריתם מבוזר, מונחה event-ים ומשתמש ב2 הודעות, ADDR ו GETADDR כאשר ADDR היא הודעה המכילה מספר כתובות

on initialization do:

- peers ← some initial public seed peers
- send GETADDR to peers

on received GETADDR from NEIGHBOUR do:

- send ADDR with min(23% num of your known addresses, 2500) addresses to NEIGHBOUR

on received ADDR do:

foreach address in ADDR do:

- process address and decide if it should be added to peer
- if (num_of_addresses(ADDR) < 10) AND (timestamp(ADDR) is no older than 10 minutes) do:
  - schedule address forwarding

כפי שניתן לראות, הצומת לא משדר מיד כל כתובת עליה הוא שומע. אחת ל 100 מילי שניות, הוא בוחר את אחת מהכתובות שהגיעו להיות scheduled, בוחר 1 או 2 מה peers שלו (outgoing connections) ומשדר אותה אליהם.

בנוסף, לפני שליחת כתובת, הצומת בודק שלא נשלחה בעבר הכתובת הזאת דרך ה connection הזה. יש לשים לב - היסטוריית השליחות נשמרת פר connection ולא פר IP ולכן אם צומת מסויים מתחבר מחדש, ההיסטוריה עבורו "נמחקה".

<u>אלגוריתם לגילוי peer חדש</u>

בעת היווצרות צומת חדש, הוא מספר על כתובות ה IP שלו. כתובות המייצגות את ה network interfaces שלו אך גם את ה IP שלו כפי שהוא נראה מנקודת המבט של האינטרנט (לדוגמא יכול להיות IP של ה ISP במקרה של NAT)

on initialization do:

- send HTTP GET request to 2 hard-coded websites to get external IP
- give scores to each of ur IPS, 1 for local, 4 for external, 5 for IP that is both

on add a new outgoing connection to NEIGHBOUR do:

- send VERSION message containing your highest score IP to NEIGHBOUR

השכן שהפך להיות אחד מ8 החיבורים היוצאים של הצומת החדש יפרסם באמצעות אלגוריתם יצירת החיבורים את כתובתו בעלת הציון הגבוה ביותר של הצומת החדש.

<u>אלגוריתם להעברת טרנזקציות</u>

כאשר צומת כלשהו הוא זה שיצר את הטרנזקציה הוא פועל באופן זהה, ניתן להסתכל על זה כאילו הוא שולח אותה לעצמו.

INVERNTORY היא הודעה המכילה hash-ים של טרנזקציות עליהן צומת שמע והוא מפיץ לשכנים שלו

on received INVENTORY from NEIGHBOUR do:

- get hash of transactions from the INVENTORY message
- if hash is valid do:
  - send GETDATA to NEIGHBOUR

on received GETDATA from NEIGHBOUR do:

- send TRANSACTION with actual transaction info to NEIGHBOUR

on received TRANSACTION do:

- process transactions in TRANSACTION
- create a hash of TRANSACTION
- if hash ends with 2 zero bits do:
    - send TRANSACTION to peers
    - else:
        - schedule TRANSACTION for forwarding

אותן הערות המופיעות עבור יצירת החיבורים תקפות גם לגבי שליחת הטרנזקציות. טרנזקציות שנראו מתועדות באמצעות ה hash-ים והם נשמרים לנצח

<u>התמודדות עם TOR</u> (מאמר [10])

הטכנולוגיה של גלישה באמצעות TOR מוכרת ומחוץ ל scope של העבודה. נסביר רק באופן כללי שזאת דרך לביצוע גלישה אנונימית שמתבססת על העברת הפאקטות של המשתמש דרך סדרת שרתים ידועים, כל שרת מוסיף שכבת הצפנה ובסוף הפאקטה יוצאת דרך השרת האחרון. התהליך גורם לכך שנראה שה IP הגולש הוא שרת TOR האחרון.

במקרה ומבקשים לבצע דה-אנונימיזציה למשתמש ביטקוין שידוע שהוא גולש דרך TOR מציעים כותבי המאמר לגרום לכך שכל שרתי TOR יוחרמו על ידי צמתי ביטקוין מעניינים (המשך השיטה שלהם תבהיר מי אלה) באמצעות האלגוריתם הבא:

1. צור צומת ביטקוין
2. עבור כל צומת מעניין בצע:

- עבור כל שרת TOR (קיימים כ1000 כאלה) בצע:
- בצע התחברות בין הצומת שיצרת לבין הצומת המעניין דרך שרת ה TOR
- שלח 100 הודעות שאינן well formed, לדוגמא TRANSACTION שמכיל 0 טרנזקציות

האלגוריתם יגרום לכך שכל צמתי TOR יוחרמו על ידי הצמתים המעניינים ויבטיח שכל חיבור נוסף לא יתאפשר על ידי TOR ולכן רק חיבורים "רגילים" יתאפשרו (ל24 שעות)

<u>התמודדות עם clients</u>

כפי שהסברנו, קליינטים הם צמתים אשר נמצאים מאחורי NAT ומתחזקים רק חיבורים יוצאים. כעת נתאר כיצד ניתן ללמוד מידע על השרתים (servers) אליהם הם מתחברים.

השיטה מתבססת על העובדה שכאשר קליינט C יוצר קשר עם אחד השרתים, הוא מפרסם את הכתובת שלו Ca כפי שהיא נראית מנקודת המבט של האינטרנט. אם נניח לרגע שהתוקף

כבר מחובר לאחד השרתים אז הכתובת Ca תועבר אליו. עובדה זאת רומזת על מוטיבציה לאסטרטגיה הבאה:

1. התחבר ל W שרתים, כאשר W זה מספר הקרוב לכל השרתים
2. עבור כל Ca עליו שמעת, תחזק רשימה E של שרתים שהעבירו את Ca לצמתי התוקף. נסמן אותם ב ECa

קיימות 2 בעיות עיקריות - השרתים עשויים לשלוח את Ca לצמתים אחרים, לאו דווקא של התוקף והשנייה היא שהקליינט לא מתחבר לכל השרתים במקביל ויש הפרש זמנים. בשני המקרים Ca עשוי להגיע לשרת של התוקף דרך peers אחרים, דבר שגורם לכניסות ב ECa שהן רעש (false positives)

הכותבים מציעים אסטרטגיה להתמודדות עם הרעש -
האסטרטגיה מתבססת על ההנחה שאו שה IP של הקליינט כבר היה ידוע ברשת ביטקוין או שIP החיצוני שלו נמצא ברשימת IP-ים פומביים כלשהי, לדוגמא IP של ISP כלשהו.
אם התוקף יודע את Ca, הוא יכול להגביל את הפעפוע של Ca באמצעות העובדה שאם כתובת כבר נשלחה מ A ל B אז היא לא תישלח שוב.
לכן, תוקף יכול לשלוח את Ca לכל השרתים אליהם הוא מחובר מספר פעמים ובכך הסתברותית להפחית את הרעש.
האלגוריתם נראה כך

1. שלח את Ca למספר גדול של שרתים
2. התוקף יוצר חיבורים חדשים למספר גדול של שרתים ומקווה שחלקם יהיו בעלי חיבורים שונים
3. כאשר קליינט C מתחבר לשרת E1 אליו התוקף מחובר, התוקף לומד ש C אולי מחובר ל E1 (מוסיף את E1Ca לרשימה אותה הוא מתחזק שכעת יש בה פחות רעש)

<u>האלגוריתם הראשי לביצוע דה-אנונימיזציה</u>

1. צור רשימה S של שרתים
2. צור רשימה C, כתובות של קליינטים (או שרתים), שיש לעשות לה דה-אנונימיזציה
3. למד את השרתים אליהם קליינטים מהרשימה C מתחברים כאשר הם מתחברים לרשת
4. הקשב לתעבורה מצמתי S והצמד טרנזקציות לשרתים וקליינטים

שלב 1 הוא די ישיר וניתן לביצוע במספר דרכים כמו שליחת GETADDR ברשת באופן רקורסיבי, עבור כל כתובת P ניתן לנסות לפתוח tcp connection ובמידת ההצלחה לסווג כשרת.

שלב 2 ניתן לביצוע על ידי בחירת טווח IP כלשהו או שימוש בכתובות IP של צמתי ביטקוין שכבר היו ידועות או אפילו לבחור מתוך הרשימה S

שלב 3 מתבצע בעצם באמצעות האלגוריתם שתואר להתמודדות עם קליינטים. נקודה חשובה היא שמאחר וקיימים כ 10^5 שרתים ברשת ומספר האופציות לשרתים אליהם קליינט מתחבר הוא 8*10^5, מתקיימת המשוואה הבאה:

$$\frac{10^5 \cdot 10^5}{(8 \cdot 10^3)^3} \ll 1$$

ולכן אם דואגים לגלות 3 שרתים אליהם קליינט מתחבר אז יהיו מעט מאוד collisions וגם ניתן לזהות באופן חד ערכי קליינט באמצעות triplet של שרתים אליהם הוא מתחבר. כמו כן גם רק זוג שרתים מזהה קליינט בהסתברות לא זניחה.

שלב 4 -

שלב זה רץ במקביל לשלבים 1-3. לצורך ההסבר נסמן את הסט של השרתים אליהם צומת P מתחבר ב Ep.

התוקף מנסה בשלב זה לצמד בין טרנזקציות שהוא שומע ברשת לסטים Ep.

אלגוריתם לביצוע השלב -

1. עבור על כל הודעות ה INVENTORY מכל החיבורים בבעלות התוקף ועבור כל טרנזקציה T הנמצאת ב INVERNTORY הזה, שמור רשימה RT המכילה את q כתובות הביטקוין הראשונות מהן שמעת על ה INVENTORY הזה (המלצה q=10)
2. עבור כל Ep, התוקף מסתכל על תתי קבוצות של Ep בגודל 3, נקרא להן Ep' ומחפש RT עבורן יש לו התאמה (רואים את כל ה3 כחלק מה10 המצאים ב RT).
3. אם הוא מוצא התאמה הוא מציע התאמה (P, T)
4. אם אין התאמה, הוא מחפש התאמות כאשר Ep' הן תתי קבוצות בגודל 2 ואם שוב אין אז בגודל אחד. עשויות להתקבל מספר התאמות בשלב זה אבל הן יצטמצמו כתלות במספר הטרנזקציות שמעבדים

<u>תוצאות אמפיריות</u>

לצורך POC כותבי המאמר בדקו את ההתקפה על רשת ה test net (שהוזכרה בפרק הרקע הטכני) ולא על main net מסיבות אתיות.

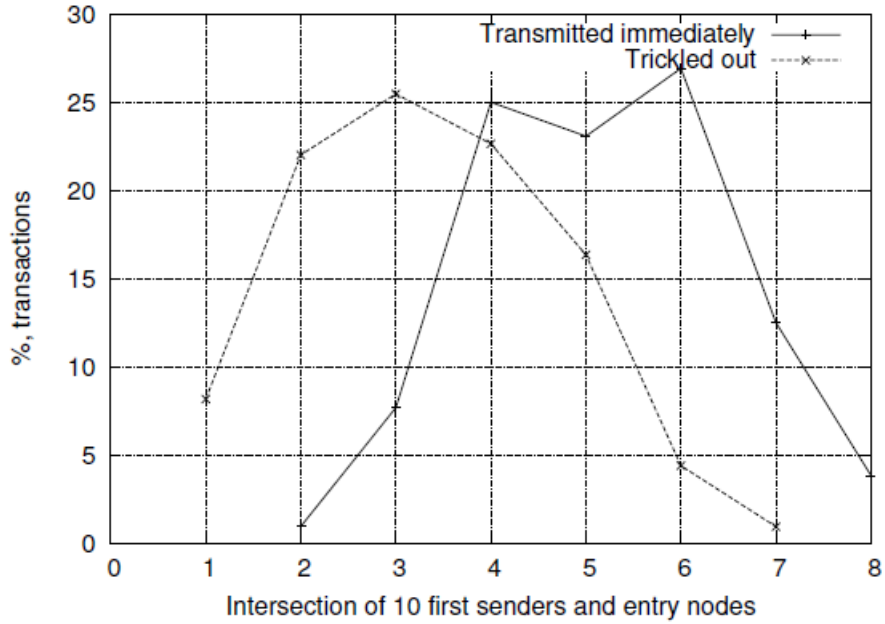
לצורך הניסוי מימשו bitcoin client משלהם שייתן יכולות ספציפיות לביצוע ההתקפה שלהם כמו שליחת הודעות מסויימות on demand או פתיחת מספר מסויים של חיבורים לאותו שרת ביטקוין. לצורך ניטור על מצב הניסוי (מי הם הצמתים שכרגע מחוברים וכו,) הם השתמשו ב open source crawler.

מספר השרתים ב test net נע בין 230 ל 250, דרגת הצמתים הממוצעת הייתה כ30. במסגרת הניסוי יצרו מספר משתמשים המתחברים לרשת מאותה ה ISP וגם מ ISP-ים שונים, בזמנים שונים. כחלק מההתקפה התווספו לכל שרת ברשת 50 חיבורים. עבור כל ניסוי בחלק הראשון של ההתקפה, נשלחו הכתובות של אותם הקליינטים שהם הוסיפו כ10 דקות לפני שהקליינטים התחילו לשלוח טרנזקציות. הקליינטים שלחו 424 טרנזקציות במהלך הניסוי.

שני ניסויים הוצגו במאמר. בניסוי הראשון אוששה ההשערה שטרנזקציות ראשית נשלחות על ידי השרתים הסמוכים (השערה עליה מתבסס האלגוריתם בשלב 4 על ידי בחירת q הראשונים). לצרוך הניסוי נבחר q=10.

הם חילקו את הטרנזקציות ל 104 ששודרו מיד ול320 ששודרו לפי האלגוריתם לפעפוע טרנזקציות (ניסו לחקות קרוב כמה שיותר את פעולתו של צומת "רגיל")

איור 19 שנלקח מ [10] מראה את מספר השרתים שהיו מבין 10 הראשונים להעביר את הטרנזקציות לצמתי התוקף



איור 19 - מספר השרתים שהיו מבין 10 הראשונים להעביר טרנזקציות לצמתי התוקף

כמצופה, אם טרנזקציה היית נשלחת מיידית לכל השרתים המחוברים לצומת, התוקף היה יכול לתפוס שלוש או יותר מהן ב 99% מהמקרים. במקרה של סט הטרנזקציות השני, אלה שנשלחות לפי האלגוריתם של כל 100 מילי שניות וכו', התוקף הצליח לתפוס 3 או יותר צמתים ב 70% מהמקרים. בנוסף ציינו כותבי המאמר כי הם שמו לב שעבור רוב הטרנזקציות, שני הצמתים הראשונים ש"סיפרו" לצומת תוקף על טרנזקציה היו שרתים מבין 8 החיבורים של הצומת המעניין.

בניסוי השני הורץ האלגוריתם במלואו. להלן התוצאות:

- כל אחד מהקליינטים אותם הם הוסיפו זוהו באופן חד ערכי על ידי השרתים אליהם הוא מחובר
- בממוצע זוהו 6 שרתים עבור כל צומת
- כ 60% מכל הטרנזקציות הוצמדו ל IP הנכון

בניסוי נוסף זהה לזה הם הוסיפו רק כ 20 צמתים והגיעו לזיהוי מוצלח של כ 41% מהטרנזקציות

<u>ניתוח הצלחת האלגוריתם</u>

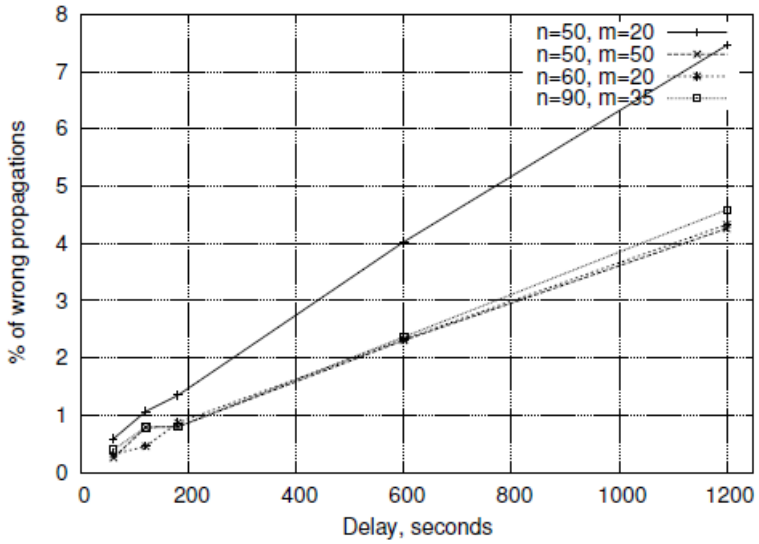
אחוז ההצלחה של האלגוריתם תלוי במספר פרמטרים. החשוב בהם הוא מספר החיבורים של התוקף מבין סך החיבורים שמתחזקים השרתים אליהם מחובר הצומת המעניין. (המאמר קורא להם entry nodes - נסביר שוב, אלה הם 8 השרתים אליהם מתחבר הקליינט כ outgoing connections שלו)

ככל של entry nodes יש פחות חיבורים פתוחים עם צמתים כלשהן ברשת, תוקף יכול לפתוח איתם יותר ויותר חיבורים משלו וכך להעלות את ההסתברות לדה-אנונימיזציה של הצומת המעניין.

בשלב זה נציג חלק מניתוח ההתקפה והסתברויות ההצלחה עבור פרמטרים שונים.

<u>מספר החיבורים לשרתים</u>

כותבי המאמר אספו הסתברויות מתצפיות עבור מספר חיבורים של התוקף וחיבורים שאינם שייכים לתוקף עבור רשתות שונות ועבור ערכי delta שונים. הם ביצעו כ 10000 הרצות של המודל ואספו תוצאות. איור 20 שנלקח מ [10] מציג את התוצאות



איור 20 - טעויות ניתוב כתלות בזמן עבור ערכי n, m, delta שונים

כאשר n הוא מספר החיבורים "הרגילים" של צומת, m הוא מספר החיבורים של תוקף.

כמצופה, ניתן לראות כי ככל שיש לצומת יותר חיבורים, הטעויות פחות סבירות.

אבחנה נוספת היא שההסתברות של צומת להעביר את Ca על גבי חיבור שאינו שייך לתוקף, תלויה במספר הכולל של חיבורים קיימים ולאו דווקא בחלק היחסי של חיבורים השייכים לתוקף.

<u>מדד הצלחה כולל לאלגוריתם</u>

המאמר מציע מטריקה למדידת ההצלחה של האלגוריתם באופן הבא

Pc - מדד הצלחה.

מדד זה תלוי במספר מאפיינים של הרשת האמיתית. ראשית, נניח כי התוקף יוצר את כל החיבורים האפשריים לשרתי ביטקוין. מהמידע שנאסף לגבי מספר החיבורים הפתוחים ניתן להעריך את הערך PAvgAddr שהוא הערך הממוצע של PAddr. במהלך הניסויים כותבי המאמר לא יצרו יותר מ50 חיבורים על מנת להימנע מהעמסה על השרתים. הערכה פסימית היא ש50 חיבורים הם המקסימום שתוקף יצליח להתחבר לצומת מסויים. תחת ההנחה הזאת מתקבל שערכו של PAvgAddr הוא בערך 0.34.

לאחר מכן נניח כי test net ו main net הם בעלי מאפיינים רשתיים דומים ולכן ההסתברות P3(L) עבור הממ L המתאר את מספר ה entry nodes שנמצאים ב top 10 הם בערך זהים בין הרשתות. מחשבים את הערך P1(R) עבור הממ R שסופר את ה entry nodes שזוהו מתוך ה8 כפונקציה של PAvgAddr. לאחר מכן מחשבים את ההסתברות הכוללת שהתוקף יגלה לפחות M3 - ממ המתאר את מספר הצמתים מבין ה entry nodes שהתגלו ב top 10.

ומקבלים את ההערכה הבא - Psuccess = 0.11.

אם מגבילים את האלגוריתם לזוגות של entry nodes במקום שלשות, מגיעים להערכה של Psuccess = 0.35.

ב test net הצליחו כותבי המאמר להגיע ל PAddrAvg של 0.86 ואחוז הצלחה של 60% עבור שלשות.

תוקף "אמיתי" יכול להגיע למספרים האלה אם הוא יבצע תהליך ראשוני אגרסיבי וארוך של הצפת השרתים בחיבורים השייכים לו אולם זה עלול להיות "רועש" ולגרום ל QOS נמוך ברשת.

לכן לסיכום, תוקף "זהיר" יכול להיצמד להנחיות הכלליות של מקסימום 50 חיבורים לכל השרתים, שימוש בשלשות, ובכך דה-אנונימיזציה נכונה של כ 11% מכלל הטרנזקציות.

בהינתן העבודה שחולפות כ 200000 טרנזקציות ביום, הוא יכול לגלות את מקורן של כ 22000 טרנזקציות כל יום. זה גם אומר שעל מנת שמשתמש יחשוף את ה IP, הוא צריך לשלוח 9 טרנזקציות בממוצע

3.2. selfish mining

3.2.1 הצגת עולם הבעיה

כריית ביטקוין היא פעילות בה לוקחים חלק המיינרים ברשת. מיינרים משלמים בכוח חישוב, חשמל, על מנת לפתור חידות קריפטוגרפיות ווידוא של טרנזקציות על מנת להבטיח את

תקינות ואת אבטחת הרשת וקיום המטבע. מיינרים מצופים לפעול על פי הפרוטוקול ומסופקים להם תמריצים שונים - פרס בלוק ועמלות על מנת שיפעלו בדרך זאת. הדעה הרווחת היא שפרוטוקול ביטקוין הוא incentive-compatible ז״א, שמנקודת המבט של תורת המשחקים, הדרך היחידה של מיינר למקסם רווחים היא לפעול לפי הפרוטוקול. מסתבר שהמצב אינו כזה, קיימות אסטרטגיות פעולה שונות שיבטיחו למיינרים רווחים גדולים יותר. מאחר וכריית ביטקוין הוא משחק סכום אפס - אם מיינר אחד מרוויח, מיינר אחר מפסיד, אסטרטגיות מהסוג הזה יגרמו להפסדים של שאר הרשת וכפועל יוצא, כל הרשת עשויה לעבור לכרות באמצעות אסטרטגיה כזאת כי זה יותר רווחי עבורה. בפרק הבא גם יוצג מימוש של התקפה זאת שתהווה את החלק המעשי של העבודה

3.2.2 אלגוריתם לביצוע selfish mining - מאמר [6]

<u>מבוא</u>

מאמר זה נכתב על ידי איתי אייל, היום פרופסור בפקולטה להנדסת חשמל בטכניון, נכתב בנובמבר 2013, מתאר התקפה גנרית ברמת הפרוטוקול לביצוע selfish mining ועד היום משמש בקהילת ביטקוין כרפרנס כמעט היחיד לביצוע התקפה מהסוג הזה. המאמר מראה כיצד באמצעות סטייה מהפרוטוקול במסגרתה מספר מיינרים שעובדים ביחד מקבלים החלטות לא טריוויאליות לגבי זמן פרסום בלוק חדש שהם מצאו, מצליחים להגיע לרווחים שעולים על כוח החישוב היחסי אותו הם משקיעים בכרייה.

באופן ״טריוויאלי״ התקפה כזאת מתועדת עוד במסמך הראשון אותו פרסם satoshi ובו הציג את הרעיון של ביטקוין עם ההבדל החשוב - ההתקפה אותה תיעד satoshi דיברה על שליטת התוקף במעל 50% בצמתי הרשת ובכך בעצם שליטה בקונצזוס. המאמר מציג אלגוריתם וכן את הניתוח ההסתברותי שמראה, תאורטית, שאין צורך באמת בשליטה כל כך רחבה ברשת על מנת לבצע את ההתקפה.

הרעיון הכללי הוא ש pool באופן מכוון ישמור את הבלוקים שהוא מצליח לכרות בסוד ובכך באופן מכוון יגרום ל fork של הבלוקצ׳יין ולכך שה pool מחזיק בבלוקצ׳יין פרטי, אלטרנטיבי.

שאר הצמתים, שמניחים במאמר זה שהם "honest" (כורים בצורה תמימה ולפי הפרוטוקול), ימשיכו לכרות בלוקים בבלוקצ׳יין הפומבי והבריכה תמשיך לכרות בבלוקצ׳יין הפרטי. אם הבריכה מוצאת עוד בלוקים היא מגדילה את הפער של הבלוקצ׳יין הפרטי. כאשר צמתי הבריכה ״מבחינים״ שה branch הפומבי מתקרב לאורך של הbranch הפרטי שלהם, הם מגלים חלק (או את כל) מהבלוקים הפרטיים שלהם. אסטרטגיה זאת תוביל את הצמתים התמימים לבזבז משאבים על כריית בלוקים שלא ישרתו שום מטרה.

המאמר מראה שמעבר לערך סף כלשהו של גודל ה pool, הרווח של ה selfish pool גדל באופן גדול מלינארי.

ההשלכות של זה עלולות להיות הרסניות לרשת מאחר ומגודל מסויים בו הבריכה ״תצבור תאוצה״ יותר ויותר מיינרים יצטרפו אליה עד שהיא תגדל לגודל שבו הבריכה תיהיה היישות השולטת ברשת ואופיו ה decentralized של ביטקוין יפסיק להתקיים.

בנוסף, כותבי המאמר מראים תלות חזקה ביותר בין יכולתה של קבוצת selfish miners לבצע את ההתקפה לבין הקישוריות של צמתי הבריכה לצמתים האחרים ולאו דווקא בגודל הבריכה. תוצאה זאת מצביעה על כך שגם קבוצה קטנה מאוד יכולה לבצע את ההתקפה והתרחיש האפוקליפסי יהיה זהה.
תוצאה מעניינת נוספת שמראים היא שפרוטוקול ביטקוין לעולם לא יוכל להתמודד עם התקפה של בריכה המכילה 33% ומעלה מצמתי הרשת (דרישה מחמירה אך עדיין קלה בהרבה מ50%) וכן מציעים תיקון קטן לפרוטוקול על מנת לשנות את המספר הזה לכ 25%.

מאמר זה הוא גם הבסיס התאורטי לחלק המעשי של העבודה. בחלק המעשי נציג פרויקט במסגרתו פותחה סימולציה של רשת ביטקוין וכל selfish miners המממשים את ההתקפה המתוארת

<u>הצגת האסטרטגיה</u>

<u>מידול מיינרים, בריכות והבעיה</u>

בשלב זה מציגים כמה אנוטציות. המערכת מורכבת מסט של n מיינרים 1,..,n כאשר למיינר i יש כוח חישוב mi כך ש $\sum_{i=1}^{n} m_i = 1$. כפי שתואר בחלק הראשון של העבודה, כל מיינר בוחר head block המייצגת מבחינתו את ראש הבלוקצ׳יין והוא מנסה לכרות בלוק ממשיך עבורו.
תחת הנחת מציאת בלוקים חדשים המתפלגים פואסונית, אינטרוול הזמן בין מציאת בלוקים חדשים מתפלג אקספוננציאלית עם ממוצע m_i^{-1} .
מניחים כי מיינרים הם רציונאליים במובן שהם מנסים למקסם את רווחיהם גם במחיר סטייה מהפרוטוקול.
כוח החישוב של pool הוא סכימה של כוח החישוב של המיינרים המשתתפים בה.
הרווח היחסי המצופה, או בקיצור הרווח, של בריכה הוא אחוז הבלוקים שנכרו ע״י הבריכה מתוך סך כל הבלוקים שנכרו בבלוקצ׳יין האורך ביותר.
הנחות נוספות - מיינרים מחולקים ל2 קבוצות, pool של selfish miners שהוא המיעוט ורוב של צמתים תמימים שכורים בדיוק על פי הפרוטוקול (תאורטית אין באמת הבדל אם התמימים כורים בנפרד או בבריכה גם הם)

<u>האלגוריתם</u>

המוטיבציה העיקרית היא שה selfish miners מנסים לגרום למיינרים הרגילים לעבוד על בלוקים שלבסוף לא יהיו חלק מהבלוקצ׳יין הארוך ביותר.
הם משיגים מטרה זאת באמצעות דיווח סלקטיבי על בלוקים מהבלוקצ׳יין הפרטי שלהם כך שבלוקים אלה יגרמו למיינרים הרגילים ללמוד שהבלוקים שלהם אינם חלק מהשרשרת הארוכה והם אינם רלוונטיים ובכך לזנוח אותם.

באופן לא כל כך מדוייק ניתן להגיד שהבריכה ממשיכה לכרות בלוקים פרטיים ולהאריך את ה branch הפרטי שלה בזמן שהצמתים הרגילים ממשיכים לכרות בלוקים בבלוקציין הפומבי, הקצר יותר.

מאחר ולבריכה יש כוח חישוב קטן יחסית לשאר המיינרים התמימים, השרשרת הפרטית לא תישאר ארוכה יותר מהפומבית לנצח. לכן, הבריכה מגלה בלוקים מהפרטית כדי שיכללו בפומבית ויגרמו למיינרים התמימים לזנוח את ה branch הפומבי.

זה גורם לצמתים התמימים להפסיד על בזבוז המשאבים ומזכה את ה selfish miners בפרסים על הבלוקים החדשים שהם גילו מהשרשרת הפרטית.

איור 21 שנלקח מ [6] מתאר פסואודו קוד של ההתקפה (קוד מבוזר הרץ על כל אחד מה selfish miners)

Algorithm 1: Selfish-Mine

```
1  on Init
2      public chain ← publicly known blocks
3      private chain ← publicly known blocks
4      privateBranchLen ← 0
5      Mine at the head of the private chain.

6  on My pool found a block
7      Δ_prev ← length(private chain) − length(public chain)
8      append new block to private chain
9      privateBranchLen ← privateBranchLen + 1
10     if Δ_prev = 0 and privateBranchLen = 2 then        (Was tie with branch of 1)
11         publish all of the private chain               (Pool wins due to the lead of 1)
12         privateBranchLen ← 0
13     Mine at the new head of the private chain.

14 on Others found a block
15     Δ_prev ← length(private chain) − length(public chain)
16     append new block to public chain
17     if Δ_prev = 0 then
18         private chain ← public chain                   (they win)
19         privateBranchLen ← 0
20     else if Δ_prev = 1 then
21         publish last block of the private chain        (Now same length. Try our luck)
22     else if Δ_prev = 2 then
23         publish all of the private chain               (Pool wins due to the lead of 1)
24         privateBranchLen ← 0
25     else                                               (Δ_prev > 2)
26         publish first unpublished block in private block.
27     Mine at the head of the private chain.
```

איור 21 - פסאודו קוד selfish mining

כפי שניתן לראות, הקוד מונחה event-ים וקבלת ההחלטות בו מתקבלת בעיקר על סמך ההפרש בין אורך השרשרת הפרטית לאורך השרשרת הפומבית.

אם ה branch הפומבי ארוך מהפרטי אז הבריכה נמצאת בפיגור ולכן מחליטה לקבל את השרשרת הפומבית כשרשרת הפרטית שלה מאחר ואין הרבה סיכוי שעם המשאבים העומדים לרשותה היא "תדביק את הפער".

כאשר selfish miner מצא בלוק, הבריכה נמצאת ביתרון של בלוק אחד על פני המיינרים הרגילים והשרשרת הפומבית. במקום לפרסם את הבלוק הזה באופן נאיבי ובכך בעצם לפרסם את השרשרת הפרטית ולהפוך אותה לפומבית, הבריכה שומרת את הבלוק הזה ואת היתרון לעצמה. כעת יכולות להיות שתי תוצאות:

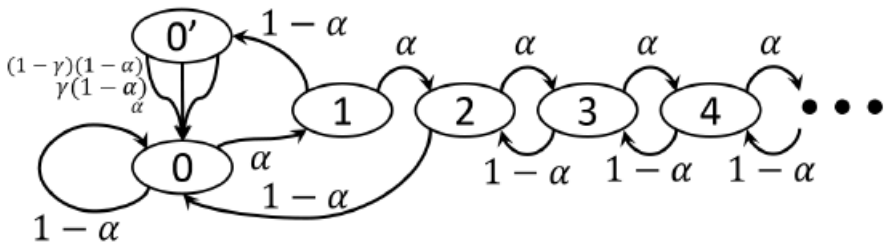
1. או שהמיינרים הרגילים מצאו בלוק נוסף לשרשרת הפומבית ובכך הם מבטלים את היתרון של הבריכה
2. או שהבריכה מצאה עוד בלוק נוסף והגדילה את היתרון שלה ל2.

במקרה הראשון, הבריכה מיד מפרסמת את הבלוק שלה ומביאה למצב של race ברשת בדיוק כפי שקורה לפעמים באופן "טבעי" כאשר שני מיינרים מוצאים בלוק אבל עם הבדל משמעותי אחד - ה race הזה מתבצע אך ורק בין הצמתים התמימים מאחר וה selfish miners יחליטו מיידית לקבל את הבלוק הפרטי.
במקרה השני, הבריכה כעת ביתרון של 2 בלוקים, הבריכה מיד מפרסמת את 2 הבלוקים וזוכה ברווחים גם של הראשון וגם של השני. אם הצמתים התמימים קיבלו את הבלוק הראשון שהבריכה יצרה (הבריכה ניצחה ב race) וכרו עוד בלוק מעליו אז הבריכה זוכה בפרס עבור הראשון והתמימים עבור השני. לבסוף, אם הצמתים התמימים כרו עוד בלוק מעל בלוק ראשון שהם מצאו אז הם זוכים בפרסים עבור שני הבלוקים.

במקרה השני, הבריכה משיגה יתרון נוח שמאפשר לה מרחב תמרון עבור תגליות עתידיות של המיינרים הרגילים. ברגע שהבריכה מצליחה להגיע למצב כזה היא ממשיכה לכרות בלוקים פרטיים. היא מפרסמת בלוק בודד עבור כל בלוק שהמיינרים התמימים מצאו. מאחר והמיינרים התמימים הם הרוב, היתרון הזה לא ישאר לנצח, הוא ידעך לכדי יתרון של בלוק אחד בהסתברות גבוהה. בנקודה זאת המיינרים התמימים קרובים מידי אז הבריכה מפרסמת את כל ה branch שלה. מאחר והשרשרת הפרטית ארוכה בבלוק אחד מהפומבית, כל המיינרים ברשת מקבלים אותה והבריכה נהנית מרווחים על כל הבלוקים שהיא כרתה. זה גם מחזיר את המערכת חזרה למצב של תחילת האלגוריתם.

<u>ניתוח האלגוריתם</u>

על מנת לבצע ניתוח של תוחלת הרווחים, המאמר ראשית מציג state machine המתאר את המצבים השונים בהם האלגוריתם מצוי ואת ההסתברויות לעבור אליהן מכל מצב.
בהינתן הסימון α עבור כוח החישוב של הבריכה, איור 22 שנלקח מ [6] מציג מכונת מצבים המתארת את ה state של הצומת



איור 22 - מכונת מצבים המתארת את מצב הבלוקציין של מיינר

מספרי המצבים מתארים את היתרון של הבריכה על פני שאר הצמתים ברשת.

המצב 0 מחולק לשניים מאחר והאלגוריתם כפי שתואר פועל באופן שונה כאשר השרשרת הפומבית והפרטית הן זהות (לדוגמא בתחילת האלגוריתם) וכאשר השרשרת הפומבית והפרטית הן באותו האורך אבל הבלוק האחרון שונה (המצב של ה race). 0′ הוא המצב המתאר את ה race.

על מנת לחשב את תוחלת הרווח יש לקחת בחשבון את ההסתברויות לקפיצות בין המצבים וחישוב הרווחים המתאימים להם. נעבור על ה event-ים ועל המצבים האפשריים.

אם לבריכה יש יתרון 1 והתמימים מוצאים בלוק, הבריכה מפרסמת את הבלוק שלה מיד, דבר שמוביל ל2 branch-ים באורך 1. מיינרים בבריכה כולם ממשיכים לכרות בשרשרת הפרטית בגלל שגילוי של בלוק נוסף יוביל ליתרון ובכך לרווח לכל הבריכה. התמימים אולם, עוקבים אחרי הפרוטוקול הרגיל של ביטקוין וכורים בשרשרת עליה ישמעו קודם.

כאן מציגים סימון נוסף - γ. סימון זה מתאר את החלק מתוך המיינרים התמימים שבחרו לכרות דווקא בהמשך לבלוק של הבריכה. כמובן ש $1-\gamma$ המיינרים התמימים הנותרים החליטו לכרות בהמשך לבלוק שהתמימים מצאו.

עבור המצבים s=0,1,2,… בהסתברות α, הבריכה כורה בלוק נוסף ומגדילה את היתרון שלה ל s+1. עבור מצבים s=3,4…. בהסתברות $1-\alpha$, הצמתים התמימים מוצאים בלוק והיתרון קטן ל s-1. אם התמימים כורים בלוק כאשר היתרון הוא 2, הבריכה מפרסמת את הbranch שלה וחוזרים למצב 0. אם התמימים כורים בלוק כאשר היתרון הוא 1, עוברים למצב 0′. ממצב 0′ ניתן להגיע דרך שלושה מעברים שונים שכולם מובילים ל0 - או שהבריכה כורה בלוק נוסף על גבי השרשרת הפרטית בהסתברות α, או התמימים כורים בלוק על גבי השרשרת הפרטית בהסתברות $\gamma(1-\alpha)$, או שהתמימים כורים בלוק על גבי השרשרת הפומבית שלהם בהסתברות $(1-\gamma)(1-\alpha)$

<u>הסתברויות המצבים</u>

המאמר מציג נספח בו מנתחים את ההסתברויות להגיע לכל מצב (מנתחים את כל אחד מהחתכים האפשריים) לא נציג כאן את המתואר בנספח. התוצאות הן להלן:

$$p_0 = \frac{\alpha - 2\alpha^2}{\alpha(2\alpha^3 - 4\alpha^2 + 1)}$$

$$p_{0'} = \frac{(1-\alpha)(\alpha - 2\alpha^2)}{1 - 4\alpha^2 + 2\alpha^3}$$

$$p_1 = \frac{\alpha - 2\alpha^2}{2\alpha^3 - 4\alpha^2 + 1}$$

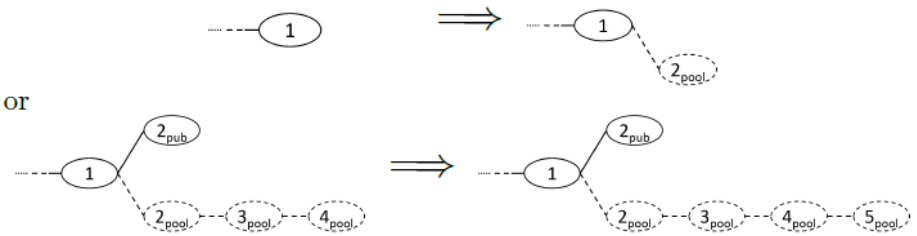
$$\forall k \geq 2 : p_k = \left(\frac{\alpha}{1-\alpha}\right)^{k-1} \frac{\alpha - 2\alpha^2}{2\alpha^3 - 4\alpha^2 + 1}$$

איור 23 - הסתברות המצאות בכל מצב

<u>חישוב רווח</u>

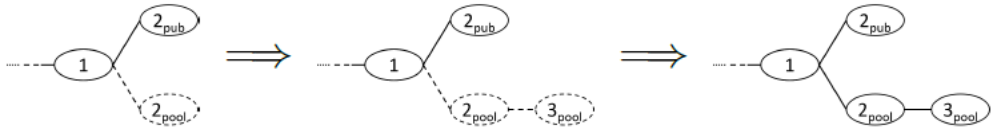
הסתברויות המצבים הם הבסיס לחישוב הרווח. על מנת לחשב ממש את תוחלת הרווחים יש שוב לחלק למקרים ולהסתכל על הרווח בכל מקרה. המקרים הקיימים הם: (ראה איורים 24- 30)

a. כל מצב מלבד מצב בו יש שתי שרשראות באורך 1 וגם הבריכה מצאה בלוק



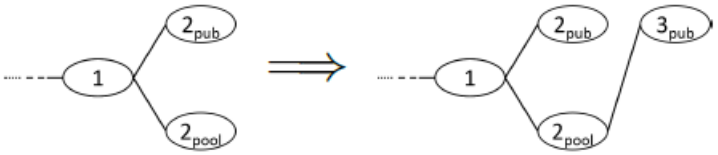
איור 24

b. היו 2 שרשראות באורך 1 וגם הבריכה מצאה בלוק



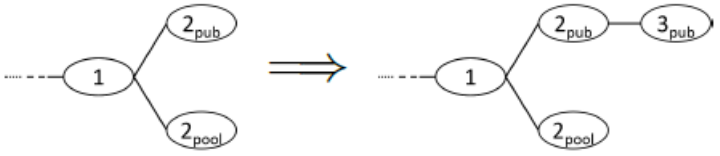
איור 25

c. היו 2 שרשראות באורך 1 וגם התמימים מצאו בלוק שממשיך את השרשרת הפרטית



איור 26

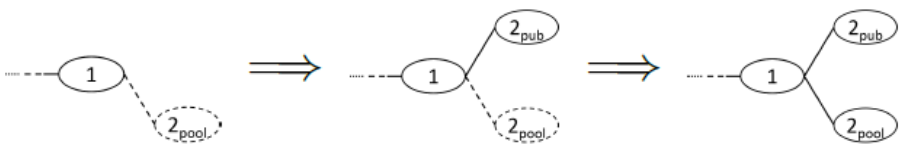
d. היו 2 שרשראות באורך 1 וגם התמימים מצאו בלוק שממשיך את השרשרת הפומבית



איור 27

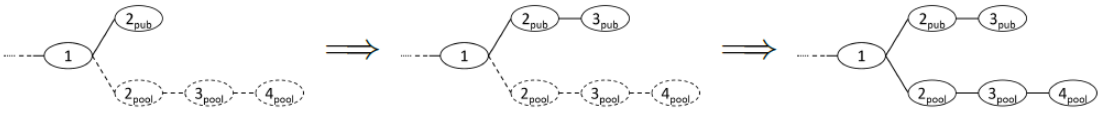
e. השרשראות זהות, התמימים מצאו בלוק

f. היה יתרון של 1 לפרטית והתמימים מצאו בלוק



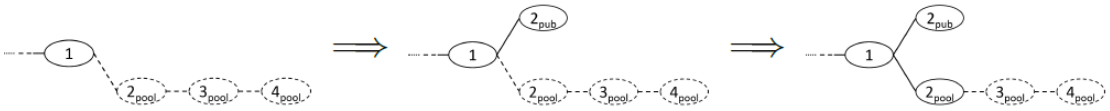
איור 28

g. היה יתרון של 2 לפרטית והתמימים מצאו בלוק



איור 29

h. היה יתרון של יותר מ2 לפרטית והתמימים מצאו בלוק



איור 30

אלו כל המקרים, לא נחזור שוב על מה האלגוריתם עושה בכל מצב לפרטים. בכל מצב או שהבריכה מרוויחה או שהתמימים מרוויחים. יש להכפיל את ההסתברויות בערך הרווח ולסכום את עבור רווח של כל גורם. המקרים בהם נסמן ב r_{others}, r_{pool} את תוחלת הרווחים של הבריכה והתמימים (others). איור 31 שנלקח מ [6] מציג את הביטויים המתקבלים

$$r_{\text{others}} = \overbrace{p_{0'} \cdot \gamma(1-\alpha) \cdot 1}^{\text{Case (c)}} + \overbrace{p_{0'} \cdot (1-\gamma)(1-\alpha) \cdot 2}^{\text{Case (d)}} + \overbrace{p_0 \cdot (1-\alpha) \cdot 1}^{\text{Case (e)}}$$

$$r_{\text{pool}} = \overbrace{p_{0'} \cdot \alpha \cdot 2}^{\text{Case (b)}} + \overbrace{p_{0'} \cdot \gamma(1-\alpha) \cdot 1}^{\text{Case (c)}} + \overbrace{p_2 \cdot (1-\alpha) \cdot 2}^{\text{Case (g)}} + \overbrace{P[i>2](1-\alpha) \cdot 1}^{\text{Case (h)}}$$

איור 31 - תוחלת הרווחים של הבריכה התמימה והבריכה הselfish

כמצופה, האסטרטגיה של ה selfish miners גורמת למיינרים התמימים לבצע עבודה לחינם, העובדה הזאת מביאה לירידה בקצב ייצור הבלוקים הכולל ומתבטא בכך ש $r_{others} + r_{pool} < 1$.

הפרוטוקול דואג לfine tuning על מנת שכל כ10 דקות ימצא בלוק חדש, לכן הקצב הממשי של רווחי המיינרים הוא החלק היחסי של הבלוקים שהם כרו מתוך כל הבלוקים בבלוקציין. באמצעות החלפה של ההסתברויות שנמצאו למצבים ברווחים עצמם שחושבו כאן כותבי המאמר מחשבים שוב את הרווח של הבריכה **עבור ערכי** α **שהם בין 0 ל0.5** ומגיעים (לאחר פישוט של הביטוי ששוב לא ניכנס אליו פה כי הוא נטו מתמטי) לביטוי עבור רווחי הבריכה. איור 32 שנלקח מ [6] מציג את הביטוי המתמטי המתקבל

$$R_{\text{pool}} = \frac{r_{\text{pool}}}{r_{\text{pool}} + r_{\text{others}}} = \cdots = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1 + (2-\alpha)\alpha)}$$

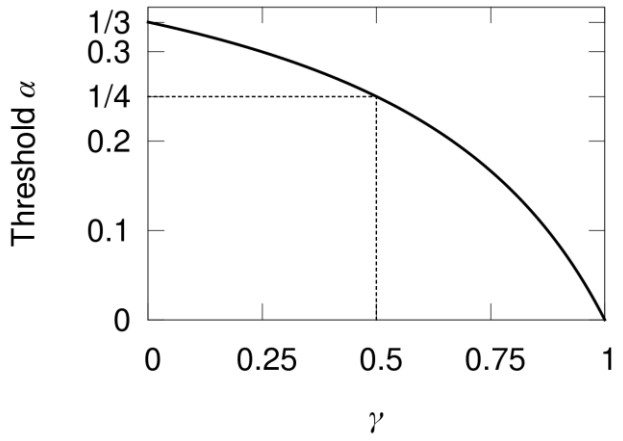
איור 32 - תוחלת רווחי הבריכה הselfish

<u>השפעתם של</u> α, γ

כאשר הרווח של הבריכה המתואר הסעיף הקודם יעלה על α, הבריכה תרוויח יותר מהחלק היחסי של כוח החישוב אותו היא משקיעה. כותבי המאמר פותרים את אי השיוויון הזה ומציגים אותו כך באמצעות האבחנה הבא:

עבור γ נתון כלשהו, בריכה בגודל α תצליח להשיג רווח הגדול מכוחה החישובי היחסי עבור α הנמצא בטווח $\frac{1-\gamma}{3-2\gamma} < \alpha < \frac{1}{2}$

תוצאה זאת מצביעה על כך שאין תלות בגודל הבריכה, נראה שזה תמיד רווחי לכרות באופן selfish בכל גודל של בריכה. אם נסתכל על המקרה הקיצוני בו $\gamma = 0$, המיינרים התמימים תמיד לוקחים את הבלוק הפומבי ולא את הפרטי של הבריכה כאשר יש race. ואז ניתן לראות שה threshold הוא 33% מכוח החישוב של הרשת להתקפה מוצלחת. עבור $\gamma = 0.5$ הthreshold הוא 25%. איור 33 שנלקח מ [6] מציג את ה threshold לביצוע התקפה מוצלחת כתלות ב γ - (ראה גם סעיף 3.3.3 על אופן קביעת ערכי אלפא וגמה)



איור 33 - ערך סף לביצוע התקפה מוצלחת

עובדה נוספת אותה ראו כותבי המאמר היא שהשיפוע של רווח הבריכה, R_{pool} כפונקציה של גודל הבריכה הוא גדול מ-1 כאשר הבריכה מעל ה threshold ודבר זה גורר את האבחנה הבאה-

עבור בריכה הפועלת לפי אלגוריתם selfish mining, הרווח של כל אחד מחברי הבריכה גדל יחד עם גודל הבריכה, עבור בריכות שגודלן עולה על ה threshold

<u>סיכום</u>

מאמר זה הראה אלגוריתם שמוכיח תאורטית כי רשת ביטקוין אינה incentive compatible. קיימת אסטרטגיה, שונה מהפרוטוקול הרשמי, אשר תבטיח לנוקטים בה רווחים הגדולים מכוח החישוב היחסי אותו הם משקיעים במערכת. המאמר מראה כי כרייה לפי האלגוריתם היא רווחית בכל גודל של בריכה.

בחלק המעשי נציג מימוש באמצעותו נבחן מעשית את יכולות ההתקפה ונציג תוצאות

3.3. התקפת eclipse

3.3.1 הצגת עולם הבעיה

ביטקוין משתמש ברשת מבוזרת של מיינרים המחשבים proofs-of-work על מנת להגיע לקונצנזוס לגבי הבלוקצ'יין. לטענתו של satoshi, יוצר ביטקוין, הרשת מאובטחת מפני תוקפים שמנסים להביא לקונצנזוס שגוי או לא עקבי של הבלוקצ'יין בתנאי ואין בידו של התוקף כ50% מכוח החישוב הכולל של הרשת. אולם, קיימת כאן הנחה סמויה, ניתוח הבטיחות מסתמך על אינפורמציה מושלמת. ז"א, מסתמך על כך שכל מיינר יקבל לידיו את ה proof-of-work שנעשו על ידי ה peers שלו.

מחקרים רבים נעשו הנוגעים לאבטחה של הפרוטוקול בעיקר בהקשרי החוזק של ה -proof of-work אבל מעט מאוד תשומת לב נתונה לאבטחה ברמת ה peer to peer.

המימוש הסטנדרטי של ביטקוין מתוכנן להיות פתוח by design, decentralized, ונטול תשתית מפתח פומבי (PKI) ולכן אין כל אותנטיקציה בין peers והמיינרים מזהים אחד השני עפ"י ה IP. כל מיינר בוחר באופן אקראי 8 מיינרים אחרים, קבועים, שהם ה outgoing connections שלו וכמו כן, כ ingoing connections הוא מקבל חיבורים מ117 כתובות IP כלשהן שמתחברות אליו וכך המיינרים מחליפים ביניהם "דעות" לגבי הקונצנזוס הנוכחי.

בהתקפת eclipse, תוקף מנצל מבנה זה על מנת להשתלט על חיבורים נכנסים ויוצאים אל צומת ובכך לבודד אותו מהרשת ובעצם לגרום לו להאמין ב"מציאות" אחרת.

כעת נסקור מאמרים שמראים שיטות למימוש eclipse attacks

3.3.2 ביצוע eclipse attack התוקף את המימוש של bitcoind - מאמר [7]

<u>מבוא</u>

המאמר מציג 2 סוגי אסטרטגיות בהן ניתן לנקוט על מנת לבצע התקפת אקליפס. 2 ההתקפות מתבססות על פתיחת חיבורי TCP רבים מול צמתי ביטקוין ולכן מתבססות על היכולת של התוקף לשלוט במספר רב של כתובות IP. ההתקפה הראשונה מתוארת כ"התקפת תשתית" מדברת על תוקף בנוסח ISP או מדינה בעלי אינטרס לפגוע ברשת ביטקוין ובבעלותם מספר סגמנטים "שלמים" של כתובות IP וההתקפה השנייה מדברת על תוקף שלא מחזיק ביכולות אלה אבל יכול להשתמש ב botnet לביצוע המשימה ואז ברשותו כתובות IP מגוונות. לאחר מכן מוצגים ניתוח הסתברותי של האלגוריתם, מדידות ותוצאות ניסויים.

על מנת להבין לעומק את המתואר כאן יש לקרוא קריאה מקדימה של האלגוריתם לביצוע דה-אנונימיזציה המוצג בפרק 3.1.3 (ניתן להסתפק בחלק המדבר על המימוש של רשת ה peer-to-peer)

<u>פרטי מימוש של רשת ה peer-to-peer שלא הוזכרו בפרק 3.1.3 של העבודה</u>

מימוש שמירת peers

צומת שומר את כתובות ה IP ב2 טבלאות, האחת נקראת tried והשנייה new.

טבלאות אלה נשמרות על הדיסק והן פרסיסטנטיות גם בפני restart של הצומת.

טבלת ה tried מכילה 64 "buckets" כאשר כל bucket כזה יכול להכיל עד 64 כתובות IP שונות איתן הצומת הצליח ליצור חיבור יוצא או נכנס. יחד עם כל כתובת, הצומת שומר timestamp של הפעם האחרונה בה התבצע חיבור מוצלח לIP זה. אופן בחירת bucket עבור כתובת מסויימת מתבצע באמצעות האלגוריתם הבא

SK = random value chosen when node is born.
IP = the peer's IP address and port number.
Group = the peer's group
i = Hash(SK, IP) % 4
Bucket = Hash(SK, Group, i) % 64
return Bucket

כאשר Group הוא 16 הבתים הראשונים של כתובת ה IPV4 של ה peer.

כך שכל כתובת ממופה לbucket אחד בלבד וכל group ממפה ל4 bucket-ים לכל היותר.

כאשר צומת מתחבר ל peer אחר, מנסים להכניס את הכתובת של ה peer ל bucket המתאים. ומבצעים

1. אם ה bucket מלא (יש שם כבר 64 כתובות) בצע (נקרא bitcoin eviction):
   - בחר 4 כתובות אקראיות מה bucket
   - מחק את הותיקה מביניהן מה bucket ושים את הכתובת החדשה במקומה
   - שים את הכתובת שנמחקה מה bucket בטבלת new
2. אחרת, אם הכתובת החדשה כבר נמצאת ב bucket בצע:
   - עדכן את ה timestamp שלה
3. אחרת הוסף את הכתובת ל bucket

נציין כי בנוסף, ה timestamp מתעדכן גם כאשר אחד ה peer שולח את אחת ההודעות VERSION, ADDR, INVENTORY, GETDATA, PING (שתוארו בפרק 3.1.3)

טבלת ה new מכילה 256 bucket-ים כל אחד מהם מחזיק עד 64 IP-ים של peers שאיתם עדיין לא נוצר חיבור מוצלח. צומת ממלא את טבלת new באמצעות מידע אותו הוא מקבל מהודעות ADDR או מ DNS seeders שהם מספר צמתים גלובליים לרשת ביטקוין המאפשרים לצומת לתשאל אותם ולקבל מספר כתובות על מנת להתחיל לעבוד.

בחירת ה bucket עבור כתובת IP מסויימת נעשית באופן הבא:

SK = random value chosen when node is born.
Group = /16 containing IP to be inserted.

```
Src_Group = /16 containing IP of peer sending IP.
i = Hash( SK, Src_Group, Group ) % 32
Bucket = Hash( SK, Src_Group, i) % 256
    return Bucket
```

כאשר הפרמטר הנוסף כאן source group הוא ה group של מי שסיפר לצומת על ה peer הזה (צומת אחר או seeder). כל זוג (group, source group) ממפה ל bucket אחר.

אם bucket מתמלא אז נקראת פונקציה בשם isTerrible, עוברת על כל 64 הכתובות ב bucket, אם אחת מהכתובות בעלת timestamp גדול מ30 ימים או נעשו אליה יותר מידי ניסיונות חיבור כושלים אז היא מסומנת כ terrible והיא מפונה בעבור כתובת חדשה, אחרת מתבצעים השלבים של bitcoin eviction כפי שתוארו עבור tried עם השינוי הקטן שהכתובת שמפונה פשוט נזרקת.

<u>מימוש בחירת peers</u>

צומת ביטקוין לעולם לא משנה את 8 החיבורים היוצאים שלו אלא ב3 מקרים - הצומת ביצע restart, חיבור התנתק, peer אליו הוא מחובר ביצע עבירות והוא מחרים אותו.

צומת בעל $\omega \in [0,7]$ חיבורים בוחר את הצומת ה $\omega + 1$ להתחבר אליו באופן הבא:

1. בחר האם להתחבר לצומת מ tried או new בהסתברות

$$\Pr[select\ \ from\ tried] = \frac{\sqrt{\rho}(9-\omega)}{(\omega+1)+\sqrt{\rho}(9-\omega)}$$

כאשר ρ הוא היחס בין מספר הכתובות ב tried למספר הכתובות ב new.

2. בחר כתובת אקראית מהטבלה אבל העניק הטייה לטובת כתובות חדשות יותר באופן הבא:
   - בחר bucket לא ריק אקראי
   - בחר slot אקראי ב bucket
   - אם יש שם כתובת, החזר אותה בהסתברות $p(r,\tau) = \min(1, \frac{1.2^r}{1+\tau})$ או שחזור לתחילת השלב. r הוא מספר הכתובות שנדחו עד עכשיו (מספר האיטרציות שנעשו כבר בשלב זה) ו τ הוא ההפרש בין ה timestamp של הכתובת והזמן הנוכחי בקפיצות של 10 דקות
3. התחבר לכתובת שנבחרה בשלב 2. אם ההתחברות נכשלה חזור לשלב 1

<u>פרטי ההתקפה</u>

ראשית נציין כי ההתקפה מיועדת לשימוש כנגד צמתים בעלי כתובת IP חיצונית בלבד.

שלבי ההתקפה ממבט על:

1. מילוי טבלת tried של הצומת הנתקף בכתובות IP בבעלות התוקף
2. מילוי טבלת new של הצומת הנתקף בכתובות IP שקריות המכונות במאמר "trash"

3. מחכים (או גורמים) שיתבצע restart לצומת הנתקף
4. התחברות לשאר 117 החיבורים הנותרים של הנתקף

<u>מילוי טבלאות tried ו new</u>

התוקף מנצל מספר עובדות על מנת למלא את הטבלאות בכתובות שלו :

1. כתובות מחיבורים נכנסים אקראיים (ללא קונטקסט) נשמרים בטבלת tried -

לכן, התוקף יכול להכניס כתובת לטבלת tried של צומת נתקף באמצעות התחברות אליו מהכתובת הזאת.

בנוסף, פינוי הכתובות כאשר bucket-ים מתמלאים נותן עדיפות לכתובות עם timestamp טרי יותר ולכן סביר שכתובות חדשות של התוקף יעיפו מהטבלה כתובות לגיטימיות, ישנות יותר

2. צומת מוכן לקבל הודעות ADDR ללא קונקסט -

הודעות אלה מכילות כתובות שנכנסות מיד לטבלת new ללא בדיקת connectivity ולכן כאשר צומת תוקף מתחבר לצומת הנתקף הוא יכול לשלוח לו המון הודעות ADDR עם כתובות IP זבל שלא קיימות עד שכל new מכילה רק כאלה

3. צמתים מסתנכרנים אקטיבית מול seeders ו peers באופן נדיר מאוד -

לכן צומת כמעט לעולם לא ינקוט בצעדים שיספקו לו מידע הסותר את המידע המגיע מהתוקף

<u>restart של צומת הקורבן</u>

קיימות מספר סיבות לכך שצומת ביטקוין יאותחל מחדש כמו תקלות של ספק האינטרנט, הפסקות חשמל, שדרוגים, עדכונים או התקפות בלתי תלויות על מערכת ההפעלה של הצומת.

מאמר עליו מסתמכים כותבי המאמר הראה כי צומת בעל IP פומבי מתאתחל מחדש לאחר כ10 שעות בהסתברות של כ 25%.

עדכוני תוכנה הם סיבה ניתנת לצפייה במיוחד בעבור תוקף מאחר ויש להודיע לכל הקהילה על שדרוג שיגרום לאיתחול מחדש ותוקף יכול לנצל הזדמנויות כאלה על מנת לבצע את ההתקפה.

בנוסף ניתן לגרום באופן מכוון לאיתחול של צומת באמצעות התקפות ידועות כמו DDOS, memory exhaustion או packet-of-death שלא יסקרו בעבודה זאת.

השורה התחתונה היא שאין לרשת אפשרות להסתמך אבטחתית על 100% uptime של הצמתים.

<u>בחירת ה outgoing connections</u>

ההתקפה מצליחה אם בשעה שהצומת הקורבן מאתחל, הוא יבחר את כל 8 החיבורים שלו להיות חיבורים אל צמתים בשליטתו של התוקף. על מנת לוודא זאת, התוקף ינצל את נטיית האלגוריתם לטובת כתובות טריות יותר.

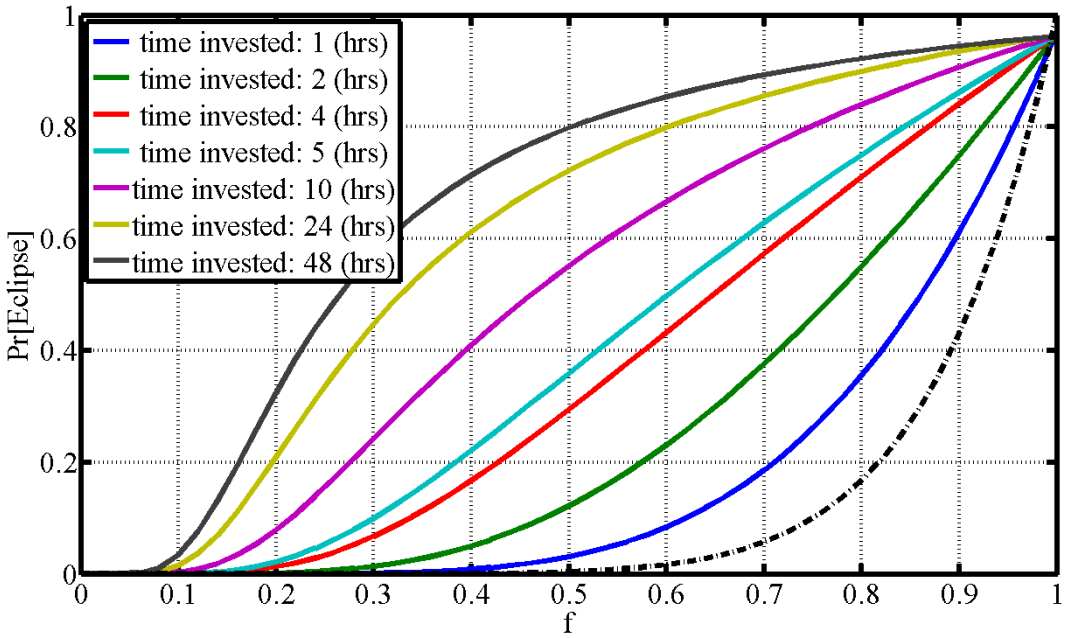
התוקף מנסה לוודא שהכתובות שלו טריות בעוד כל הכתובות הלגיטימיות הופכות לאט לאט לישנות. ננתח זאת תחת מספר הנחות (ונקודות אליהן יש לשים לב):

1. חלק מסויים, f, של הכתובות בטבלת tried של הקורבן נשלטות ע"י התוקף
2. כל הכתובות ב new הן זבל
3. ההתקפה מתבצעת בסיבובים וחוזרת על עצמה כל הזמן עד שהצומת הקורבן מתרסט. בכל סיבוב התוקף מתחבר אל הקורבן מכל הכתובות שלו. מבחינת זמן סיבוב אורך τ_a ולכן כל הכתובות התוקפות ב tried צעירות יותר מ τ_a
4. חלק כלשהו, f', מהכתובות ב tried מחוברות ממש לצומת לפני הריסוט שלו. ה timestamps של כתובות אלה מתעדכן כל 20 דקות. מניחים את המקרה הגרוע עבור תוקף - כאשר הצומת מתרסט ה timstamps שלהם מתחדשים ($\tau_a = 0$)

5. הזמן המושקע בהתקפה מסומן ב τ_l וזהו הזמן מתחילת ההתקפה ועד הריסוט של הצומת הנתקף. אם הקורבן לא השיג מידע רשתי לגיטימי חדש במהלך ההתקפה אז להוציא החלק f' מתוך ה tried, כל הכתובות הלגיטימיות ב tried ישנות יותר מ τ_l

ניתוח מתמטי מראה כי ההתקפה תצליח והקורבן יהיה eclipsed אם כל שמונת החיבורים יבחרו מצמתי התוקף, דבר הקורה בהסתברות $(q(f, f', \tau_a, \tau_l))^8$

איור 34 שנלקח מ [7] מראה שרטוטים של $(q(f, f', \tau_a, \tau_l))^8$ כפונקציה של f עבור $\tau_a = 27$ (דקות), ערכים משתנים של τ_l ו $f' = \frac{8}{64*64}$ שמשמעותו טבלת tried מלאה המכילה 8 צמתים מחוברים לפני ביצוע הריסוט של הקורבן



איור 34 - הסתברות הצלחת ההתקפה לערכים שונים

<u>השגת מונופול על החיבורים</u>

בהנחה שתוקף הצליח להשתלט על כל 8 החיבורים היוצאים, יש לדבר גם על הנכנסים. הדרישה היחידה היא שלקורבן יהיו לפחות מספר סוקטים פנויים להתחברות נכנסת. על מנת למלא את טבלת tried התוקף מבצע התחברויות מהירות לקורבן מכל הכתובות שלו באמצעות כלי low level כלשהו כמו zmap שדורש מעט מאוד משאבים ובדר״כ צוואר הבקבוק שלו הוא רוחב הפס.

כל חיבור כזה כולל tcp handshake, הודעת VERSION והתנתקות מtcp session עם הודעת RST. תהליך זה דורש העלאת 371 בתים והורדת 377 בתים. חלק מהחיבורים גם ישלחו הודעת ADDR עם 1000 כתובות והן ידרשו העלאה של 120087 בתים והורדה של 437.

אם ההתקפה מצליחה אז מלבד 8 החיבורים היוצאים שנשארים קבועים, התוקף גם לאחר זמן מסויים ישלוט בכל החיבורים הנכנסים. על מנת לשלוט בכל החיבורים באופן מוחלט ובלתי תלוי בבחירות הסתברותיות, ניתן לגרום לכתובתו של הקורבן להיות terrible עבור כל הרשת. יש לדאוג למשך 30 ימים ״להרעיל״ באופן רציף את הטבלה של הקורבן וכמו כן לא לאפשר תגובות להודעות כמו INVERNTORY דרך החיבורים היוצאים שכבר בשליטת התוקף. כפי שתואר, לאחר 30 ימים שלא נעשה חיבור, כתובתו של הקורבן תיהיה terrible עבור כל צומת שבעבר הכיר אותו, הוא ישכח מהרשת ואף אחד לא ינסה להתחבר אליו.

כותבי המאמר אימתו באמצעות ניסוי כי הקורבן יסכים לקבל את כל החיבורים הנכנסים מאותו ה IP וכי על מנת לתחזק את החיבורים הנכנסים יש צורך בשליחת פאקטות בקצב של 4 בתים לשנייה. בתנאים אלה, ניתן בקלות להשתלט על החיבורים של לא מעט קורבנות, במקביל, באמצעות מחשב בודד

<u>מספר כתובות הIP הדרוש ואסטרטגיות עבור שני סוגי התוקפים</u>

עד כה המאמר הראה כי הצלחת ההתקפה תלוי בעיקר בפרמטרים τ_l ו f .

כעת הם מבקשים לבחון את הקשר בין מספר הכתובות בשליטתו של התוקף לבין f . במילים אחרות, בכמה כתובות יש לשלוט על מנת להגיע לשליטה בחלק f מתוך טבלת tried של הקורבן. יש לזכור כי גם אם f קטן, ניתן לבצע את ההתקפה ולשלם בזמן.

בחלק זה הניתוח מתחלק לשני סוגי תוקפים כפי שתיארנו אותם במבוא - botnet ו"תוקף תשתית" שנקרא לו פה מעצמה (נזכיר, הכוונה היא לארגון שבבעלותו סגמנטים שלמים רציפים של כתובות IP).

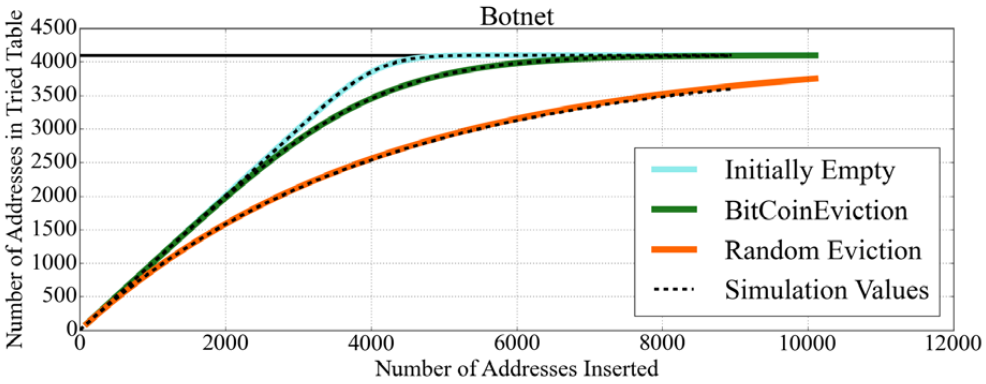
<u>botnet</u>

לתוקף מהסוג הזה יש t כתובות השייכות ל groups שונות. על מנת למדל את ה bucket אליו כל כתובת תאורטית תיכנס נתאר את התהליך על ידי התפלגות בינומית $B(t,\frac{1}{64})$.

במאמר נבחנים מספר מקרים בניסיון לענות על השאלה כמה מתוך 64*64 הכניסות של tried תוקף כזה עשוי לתפוס?

נדלג על הניתוח המתמטי המוצג במאמר..

איור 35 שנלקח מ [7] מציג את שרטוט המשוואות שהתקבלו עבור התוחלת כפונקציה של מספר הכתובות שניסו להכניס ואת תוצאות סימולצית מונטה קרלו המאמתת את התוצאות



איור 35 - מספר כתובות התוקף ב tried כפונקציה של מספר הכתובות שניסו להכניס

<u>מתקפת מעצמה</u>

במודל זה לתוקף יש כתובות IP שלקוחות מS קבוצות (groups) שונות. כל קבוצה מכילה t כתובות.

במידול שנעשה במאמר, מניחים כי התהליך דומה לכך ש4 פונקציות האש ממפות כל אחת מהקבוצות S לאחד מ64 ה buckets ואז על ידי שימוש ב lemma המובאת במאמר [7] ומוצגת באיור 36

Lemma A.1. *If k items are randomly and independently inserted into n buckets, and X is a random variable counting the number of non-empty buckets, then*

$$E[X] = n\left(1 - \left(\frac{n-1}{n}\right)^k\right) \approx n(1 - e^{-\frac{k}{n}}) \qquad (12)$$

איור 36 - למה A.1

נדלג על הניתוח המתמטי המוצג במאמר.. .

המסקנה המתקבלת היא שבאמצעות S=32 קבוצות שבכל אחת מהן יש 256 כתובות (8192 כתובות סה"כ), תוחלת הפרמטר f (החלק מתוך הטבלה tried השייך לתוקף) היא 0.86 (86% מהטבלה) כעבור מספר סיבובים מספק.

<u>סיכום botnet מול התקפת מעצמה</u>

המאמר הציג אסטרטגיה קונקרטית לביצוע eclipse attack על צומת ביטקוין. התקפה זאת מסתיימת בכך שהתוקף שולט על כל החיבורים היוצאים והנכנסים של הצומת הקורבן ובאופן ממשי שולט בו לכל צורך (הקשור לפרוטוקול ביטקוין..)

ההתקפה מסתמכת על סוג של brute force המאלץ את הצומת הקורבן למלא את טבלת הצמתים אותם הוא מכיר בצמתים של התוקף.

מאחר וההתקפה מסתמכת על כך שלתוקף יש מספר רב של כתובות IP הוצגו 2 שיטות אפשריות שתוקף עשוי להשיג משאבים כאלה - באמצעות botnet או בהיותו ארגון גדול (מעצמה) שבבעלותו סגמנטים שלמים של כתובות.

בשורה התחתונה בהשוואה ביניהם מגיעים למסקנה שהתקפת ה botnet חזקה יותר בזכות פיזור טווחי ה IP.

באמצעות botnet מגיעים לכך שכ 4600 כתובות מספיקות על מנת (בהינתן מספיק סיבובים של האלגוריתם) להגיע ל f=0.98 (98% מהטבלה היא כתובות של התוקף). באמצעות התקפה עם כתובות רציפות (מספר סגמנטים) כפי שיגיע מארגון מגיעים לכך שדרושות כ 16000 כתובות מ 63 groups שונים עם כ 256 כתובות בכל group על מנת להגיע לאותן תוצאות.

באופן כללי (מראים גם כל מיני מספרים לגבי זה במאמר) תמיד ניתן להתפשר על מספר הכתובות על חשבון הזמן הכולל, מספר סיבובי ההתקפה, במהלכם הטבלה tried מתמלאת בכתובות התוקף

3.3.3 stubborn mining - מאמר [8]

<u>מבוא</u>

על מנת להבין לעומק את פרק זה יש לבצע קריאה מקדימה של התקפת selfish mining (פרק 3.2). המאמר שיסקר בפרק זה מתאר אסטרטגיות שונות בהן ניתן לשפר את אלגוריתם ה selfish mining ולכן במובן מסויים היה מתאים לכתוב אותו תחת פרק 3.2. אולם, ההתקפה

המרכזית סביבה נכתב המאמר מתארת אסטרטגיית selfish mining המתבססת על eclipse attack על מנת לתקוף בשיעור הצלחה גבוה בהרבה. לכן טכניקה זאת מוצגת כאן, לאחר שלקורא יש כבר ידע מספק לגבי selfish mining וגם לגבי eclipse attacks.
המאמר מתחיל בהצגת מידול הבעיה, חלק גדול מהמידול נשען על המידול הנעשה בפרק 3.2. לאחר מכן מוצגות וריאציות של אלגוריתם ה selfish mining המאפשרות מצבים רבים יותר למכונת המצבים המתארת את ההפרש בין השרשרת הפרטית לפומבית ולבסוף, מוצגות מספר אסטרטגיות היברידיות המשלבות את הוריאציות יחד עם סוגים מעט שונים של eclipse attack על מנת למקסם עוד יותר את הרווחים

<u>פרטי מידול ו annotations שנוספו על המידול שהוצג עבור selfish mining</u>

- Alice : המיינרים ה selfish
- Bob : המיינרים התמימים
- בהינתן שתי אסטרטגיות X, Y, מגדירים את

$relative_gain(X,Y) = \frac{gain_X - gain_Y}{\alpha}$ כאשר gain הוא החלק של בלוקים מתוך הבלוקצ'יין אשר אותה אסטרטגיה הרוויחה. הערך מחושב ביחס לאלפא מאחר וערך זה מייצג את החלק ה"הוגן" אותו התוקף היה אמור להרוויח אם היה פועל באסטרטגיה "הוגנת"

- אסטרטגיות שיוצגו כאן יאפשרו מצבים שלא היו קיימים ב selfish mining כפי שהוגדר.

1. מצבים עם מספר שלילי שבהם Alice מחליטה להמשיך לכרות על גבי השרשרת הפרטית שלה למרות שהיא איבדה את היתרון
2. מצבים המסומנים ב 'n עבור n>0 המסמנים כי קיים fork בחלק ה"מפורסם" של הבלוקצ'יין מאחר ו Alice פרסמו שרשרת בעלת בלוקים שונים מאשר השרשרת הפומבית אבל הן באותו האורך. במצב זה Bob מתלבטים בין 2 השרשראות. המספר במצב זה, מתאר את מספר הבלוקים הפרטיים הנוספים שיש ל Alice
3. מצב "0". במצב זה אורך 2 השרשראות זהה אך למרות זאת, כל צמתי Bob כורים על אותה שרשרת פומבית

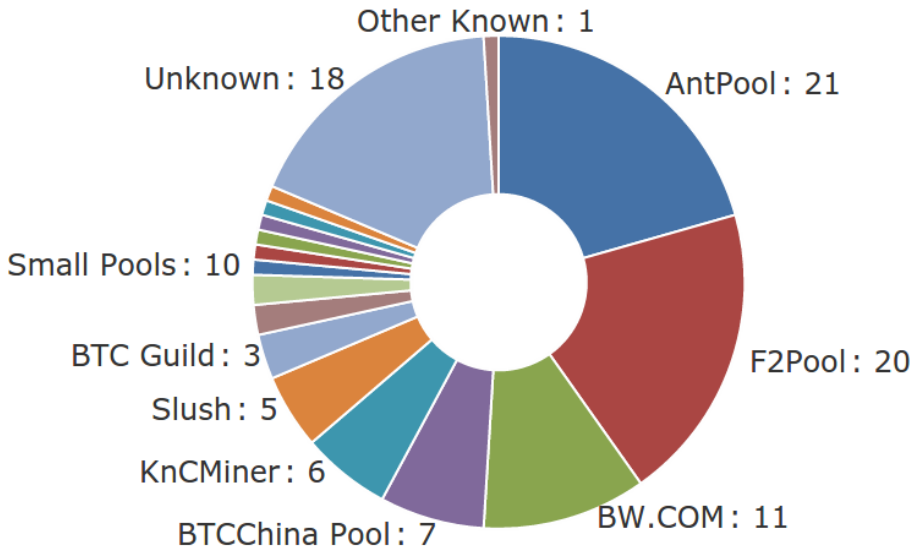
<u>ערכים סבירים עבור α, γ</u>

מהניתוח של selfish mining ניתן לראות את חשיבות שני הפרמטרים האלה עבור ה revenue של מיינר מסויים. כותבי המאמר מקדישים חלק על מנת לבחון מהו טווח הערכים ה"סביר" עבור כל אחד מהפרמטרים.

1. הערכת α :

פרמטר זה הוא החלק היחסי של כוח החישוב של Alice מתוך כוח החישוב של כלל הרשת. רוב הכרייה ברשת כיום נעשית באמצעות בריכות, 9 הבריכות המובילות מהוות כ

75% מכלל כוח החישוב. בריכות אלה עשויות לסטות מהפרוטקול על מנת למקסם רווחים. ניתן לשערך את כוח החישוב הכולל של הרשת באמצעות ה timestamp וה proofs-of-work המפורסמים ברשת והקצב שלהם. ניתן באופן דומה לשערך את כוח החישוב של כל הבריכה באמצעות יוריסטיקות שונות שמתאפשרות מאחר ובריכות חותמות בדרכים שונים על הבלוקים שהן מצאו. איור 37 נלקח מאתר blockchain.info ומראה פילוח המתבסס על שערוך שכזה



איור 37 - כוח חישוב של בריכות שונות

כפי שניתן לראות, קואליציה של 2 בריכות יכולה להגיע למעל 40% מכוח החישוב. בנוסף 18% מכוח החישוב הוא "לא ידוע" והוא עשוי להיות גם חלק מבריכות כלשהן שבכוונה מטשטשות את חלק מכוח החישוב שלהן. על כן 40% הוא ערך עליון ממשי של α.

2. הערכת γ

נניח שבנקודה מסויימת המצב בו נמצאת המערכת (בהסתכלות עליה כעל מכונת מצבים ופונקצית מעברים) הוא 2' או k' עבור $k > 1$ כלשהו. כעת Bob מוצאים בלוק ומפרסמים אותו (הם עושים זאת מיד כי הם מיינרים תמימים). Alice יכולים מיד לפרסם את אחד מהבלוקים הפרטיים הנוספים שלהם על מנת ליצור מצב של race עם הבלוק החדש ש Bob מצאו ולהעביר את המערכת למצב '(k-1).

החלק של צמתי Bob שיקבלו את השרשרת שהציעה Alice, אותו מנסים להעריך, תלוי בטיב היכולת של Alice להתחרות בהגעת הבלוק של Bob.

נסמן ב A את Alice, B את Bob, i הוא המיינר שמצא את הבלוק. התשובה לשאלה האם מיינר j יקבל את הבלוק החדש הזה, תלויה ב latency שבין שני המסלולים הרשתיים

- $i \rightarrow A \rightarrow j$

- $i \rightarrow j$

Alice יכולים להעלו את הסבירות ש j יקבל את הבלוק שלהם במספר דרכים. לדוגמא באמצעות eclipse attack מהסוג שתואר בפרק הקודם או התקפות מוכרות נוספות שתוקפות את המימוש של bitcoind. באופן תאורטי, כל דרך בה A יוכלו ליצור "קירבה רשתית" את צמתים נתקפים, יעזרו להעלות את הסבירות.
מאחר וקיימות הגנות שונות ברשת כדי להגן מפני התקפות מהסוג הזה וכמו כן ניתוח המקרה הגרוע מפיק תוצאות לא רעות בכלל, המודל כאן יהיה פשוט ככל הניתן ומנתח את המקרה הגרוע. לפי מודל זה, בגרפים אותם מנתחים אם קיים המסלול $i \rightarrow j$ אז בהכרח כל מסלול $i \rightarrow A \rightarrow j$ הוא בעל latency גבוה יותר. במילים אחרות, אם קיים מסלול מ i ל j אז j ישמע את הבלוק ש-i פירסם לפני שהוא ישמע את הבלוק ש A יפרסם בתגובה. הנחה נוספת היא שבריכות מהוות קליקה ויש לכל בריכה בגרף "קשת עצמי" שהמשמעות שלה שאפילו eclipse attack לא יכול למנוע מבריכה לכרות על גבי בלוק שהיא עצמה מצאה. מצד שני, אוסף של המון אינדיבידואליים פגיעים (כמו המיינרים התמימים) יכולים להיות ממודלים כצומת יחיד ללא קשת עצמית, ניתן אז להציג את γ באופן הבא: $\gamma = 1 - \sum_{i->j} \frac{\beta_i \beta_j}{\beta^2}$ כאשר β_i הוא כוח החישוב (hashpower) של מיינר i (מוצג בחלק היחסי מכוח החישוב של הרשת..). בנוסף הביטוי $\frac{\beta_i}{\beta}$ מתאר את ההסתברות שמיינר i ימצא את הבלוק הבא בהינתן שאת הבלוק הבא ימצא אחד מצמתי Bob.
כעת נעשה שימוש במידע לגבי התפלגות כוח החישוב בין הבריכות על מנת לעשות הנחות סבירות לגבי התקפות אפשריות.
לדוגמא, נניח כי התוקף הוא הבריכה הגדולה ביותר. זה גורר $\alpha = 21\%, \beta = 79\%$. נניח כי $\beta_o = 27\%$ שייך לכל הבריכות הקטנות וכוח החישוב הלא מסווג, ושאר 52% מחולקים בין 6 בריכות הגדולות אחרות בדומה למה שמתואר בתרשים. במקרה הגרוע נניח כי כל הצמתים ב β_o הם אינדיבידואלים פגיעים כפי שתיארנו, במצב כזה מהמשוואה $\gamma = 1 - \sum_{i->j} \frac{\beta_i \beta_j}{\beta^2}$ מקבלים $\gamma = 0.92$. במקרה הקיצון השני גם β_o וגם שאר הבריכות הן בעלות קשת עצמית ובמצב זה התוקף יכול לנצח ב race רק בין הבריכות. מתקבל ערך $\gamma = 0.45$. כמובן שניתן לקבל ערכים קטנים יותר של γ פשוט בהינתן פחות כוח חישוב עבור Alice. לכן הערכים הסבירים הם $0 < \gamma < 0.92$

איור 38 שנלקח מ [8] מציג טבלה המסכמת את הערכים וסבירותיהם לפי ניתוח זה -

TABLE 2: Hypothetical attack scenarios and parameters

α	40%	largest mining pool over most of 2014
	41%	the two largest mining pools today
	21%	largest pool today
	17%	if "unknown" mining power is a rogue miner
γ	$0 - 0.92$	depending on the attacker's influence on the overlay network (see Section 2.5)

איור 38 - המשמעות של ערכי אלפא וגאמה שונים

<u>הצגת אסטרטגיות stubborn mining</u>

כעת מוצגת משפחה חדשה של אסטרטגיות לכרייה שהן וריאציה ל selfish mining ומנסות לעשות עבודה טובה יותר. למשפחה זאת קוראים stubborn mining (כרייה עקשנית) כל אסטרטגיית selfish mining מתבססת על כרייה פרטית ופרסום בלוקים באופן סלקטיבי על מנת לגרום לשאר הרשת לבזבז משאבים. באופן ספציפי כאשר selfish miner מוביל הוא שומר לעצמו את הבלוקים שהוא כורה אבל כאשר השרשרת הפרטית מפגרת אחרי הפומבית, הוא מפרסם אותה. במקום זאת,והאינטואיציה של stubborn היא הבאה: המיינרים התוקפים לא צריכים לוותר כל כך מהר.

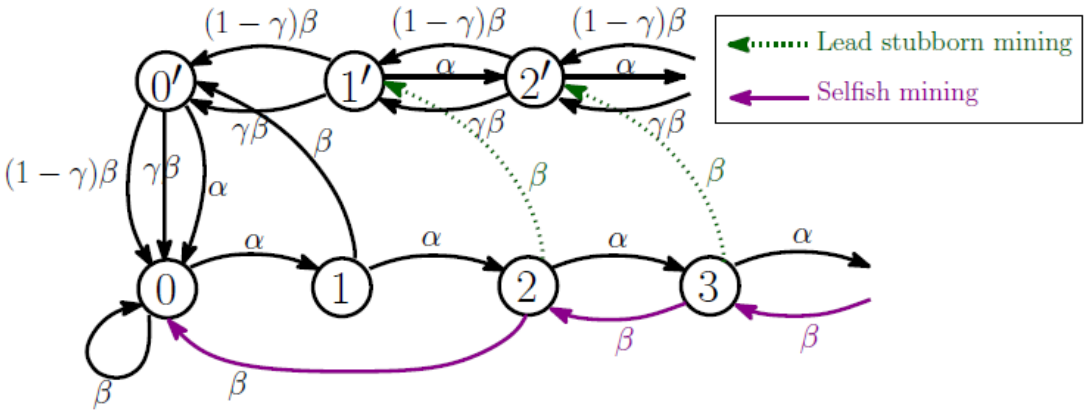
התוקף עשוי לעיתים להרוויח מכך שימשיך לכרות על גבי השרשרת הפרטית, גם אם היא בפיגור נציג שלושה וריאנטים של stubborn mining

1. Lead Stubborn שיסומן L-Stubborn
2. Equal Fork שיסומן F-Stubborn
3. Trail Stubborn שיסומן T-Stubborn

<u>L-Stubborn</u>

האלגוריתם מוצג באמצעות מכונת המצבים המתארת את המצבים האפשריים וכן את המעברים הקיימים גם עבור L-Stubborn וגם עבור Selfish mining "רגיל"

איור 39 שנלקח מ [8] מציג את מכונת המצבים



איור 39 - מכונת מצבים של L-Stubborn

עבור יתרון של 2 בהינתן שבוב מוצא את הבלוק הבא ומצמצם את הפער ל1, selfish miner רגיל מיד יפרסם את השרשרת הפרטית על מנת להבטיח שהרשת תבחר בשרשרת שלו ולכן המערכת עוברת למצב 0. ב L-Stubborn במקום לגלות את השרשרת, אליס מגלה רק בלוק יחיד מתוך השרשרת הפרטית שלו על מנת להשוות את אורך השרשרת הארוכה ביותר. במקרה הזה, γ מצמתי בוב יכרו יבחרו לכרות מעל הבלוק של אליס והמערכת תעבור למצב 1'. יכול להיות לזה יתרונות וחסרונות עבור התוקף אליס: אם γ-1 הצמתים של בוב יצליחו לקדם את השרשרת של בוב אז אליס תאבד את היתרון שלה. מצד שני, אליס עשויה להצליח לגרום ל γ-1 צמתים אלה לבצע עבודה חסרת טעם במקרה ובו γ הצמתים של בוב יצליחו לכרות בלוק מעל הבלוק הפרטי אותו אליס פירסמה.

בנוסף, כאשר היתרון הוא k>2 ובוב מוצא בלוק, selfish miner רגיל לא מגלה את השרשרת בפרטית שלו בכלל ולכן המערכת עוברת למצב k-1. אולם מיינר מסוג L-Stubborn יפעל באופן זהה למקרה הקודם, יפרסם בלוק אחד פרטי שלו, יגרום ל fork והמערכת תעבור למצב '(k-1). עפ"י סימולציות שערכו כותבי המאמר (ומתבססות על המשוואות שניתנו בחלק של הערכת γ), אסטרטגיה זאת מעניקה ל L-Stubborn miner כ 13.94% יותר רווחים, עבור הערכים הסבירים של γ ו α

<u>F-Stubborn</u>

דומה ל L-Stubborn עם הבדל אחד - כאשר המערכת נמצאת במצב 0' וצמתי התוקף אליס מוצאים בלוק, במקום לגלות מיד את הבלוק כפי ש selfish miner יעשה ולחזור למצב 0, F-Stubborn miner ישמור לעצמו את הבלוק וימשיך לכרות על גבי השרשרת הפרטית, דבר שיעביר את המערכת למצב 1.

<u>T-Stubborn</u>

זוהי בעצמה משפחה של אסטרטגיות בעלות פרמטר j. אסטרטגיה תיקרא $T_j - Stubborn$ אם:

1. במכונת המצבים שלה קיימים מצבים בעלי ערכים -1,-2,.., -k עבור k כלשהו

2. המעברים שלה זהים ל L-Stubborn למעט מעברים המובילים אל או ממצבים בעלי ערך שלילי

3. היא מקבלת את הבלוקצ'יין של Bob רק אם היא הגיעה לפיגור של j+1 בלוקים. בכל מקרה אחר, ממשיכים לנסות לכרות על גבי השרשרת הפרטית בניסיון להדביק את הפער

4. קיים מצב 0'' המסמן כי המצב שבו כל צמתי בוב כורים על השרשרת שלהם מאחר והפער הוא 0 אבל זה קרה לאחר שאליס הדביקה את הפער השלילי בו היא הייתה

איור 40 שנלקח מ [8] מציג מכונת מצבים המראה את המצבים הקיימים בכל האסטרטגיות וכמו כן את המעברים השונים שמכתיבה כל אסטרטגיה. (כאן מתואר $T_1 - Stubborn$) -

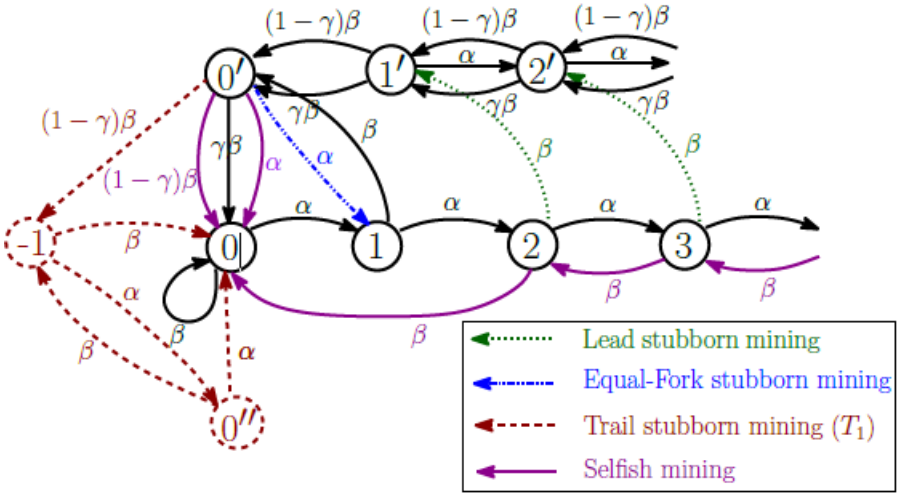


Figure 5: **Lead, Equal-Fork, and Trail Stubborn mining strategies.** Black + magenta transitions denote selfish mining [9]. Black + green transitions denote lead-stubborn mining. Black + blue transitions denote equal-fork stubborn mining. Black + brown transitions denote T_1-stubborn mining. Markov chain states are defined in Section 2.2 and Figure 1.

איור 40 - מכונת מצבים של T1-Stubborn

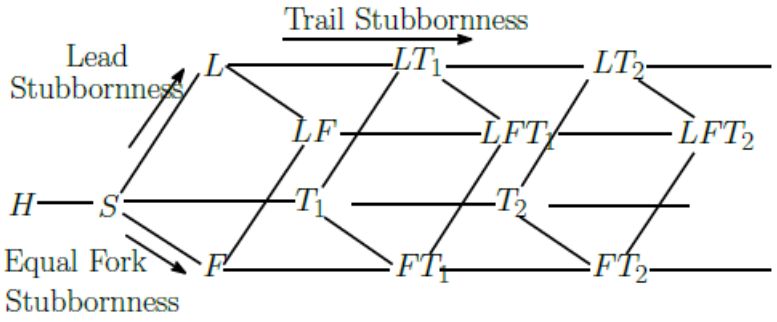
איור 41 שנלקח מ [8] מציג טבלה המדגישה באופן פשוט את ההבדלים בין האסטרטגיות -

State Transition	Action	Selfish Mining	Stubborn Strategy
lead = 2 and Alice finds a block	Reveal private chain to Bob?	Reveal	Do not reveal in L-stubborn
lead = 0′ (with a fork) and Alice finds a block	Reveal private chain to Bob?	Reveal	Do not reveal in F-stubborn
Behind by x, $x \geq 0$ and Bob finds a block	Accept Bob's block?	Accept	Do not accept till $x = j$ in T_j-stubborn

איור 41 - השוואה בין selfish ל stubborn

<u>אסטרטגיות היברידיות</u>

שלושת אסטרטגיות אלה מייצגות היוריסטיקות של כרייה שניתן להפעיל לחוד או ביחד. איור 42 שנלקח מ [8] מראה בתרשים את "יחס העקשנות" שבין סוגי האסטרטגיות -



איור 42 - אסטרטגיות היברידיות

אם קיים מסלול בגרף זה בין אסטרטגיה X לאסטרטגיה 'X זה גורר ש'X היא עקשנית יותר במובן שהיא מבצעת יותר כרייה על השרשרת הפרטית באופן הסתברותי מאשר X.

בהמשך יוצגו סימולציות עם פרמטרים שונים שעל מנת להבין אותם, עוזר להבין את ה notion הזה של "רמת עקשנות" מאחר וניתן לעשות בה שימוש לצורך בחירת האסטרטגיה המתאימה בהתאם לפרמטרי הרשת.

מיינר יכול לבחור לשנות אסטרטגיה בזמן ריצה בהתאם לפרמטרים נוכחיים שונים. לדוגמא, הוא עשוי לבחור ב $T_2 - Stubborn$ במקרה בו יש בשרשרת הפרטית שלו j>2 בלוקים פרטיים ואחרת, לבחור ב $T_0 - Stubborn$. במצב זה, אם בשרשרת הפרטית שלו יש לפחות 3 בלוקים הוא ימשיך לכרות עליה עד שהיתרון שלו ירד ל2. אחרת, הוא יקבל את השרשרת הפומבית ברגע שהוא יאבד יתרון עליה.

כמובן שניתן להרחיב לתנאים שונים נוספים. במסגרת המאמר הזה נסקרים רק צירופים המתבססים על יתרון השרשרת הפרטית אל מול הפומבית

<u>תוצאות סימולציות</u>

כעת נציג מספר תוצאות לגבי רווחיות האסטרטגיות השונות. התוצאות מתבססות על סימולציות נומריות. הרווחים של אליס ושל בוב מסומלצים כאשר אליס נוקטת בכל אחת מהאסטרטגיות המוצגות וצמתי בוב כורים באופן תמים. עבור בחירות שונות של γ ו α וכל אחת מהאסטרטגיות, סומלצו 100 מסלולים שונים של מכונת המצבים כאשר כל מסלול כזה מכיל 10^5 איטרציות. עבור כל מסלול חושבו הרווחים של אליס ובוב, ממוצע הרווחים שלהם ורווח בר-הסמך שלהם.

<u>תוצאות</u>

1. לא קיימת אסטרטגיה אחת גנרית הכי רווחית

   תוצאה מעניינת היא שעבור T-Stubborn נבדקו פרמטרים שונים של j, ולא נצפו איזורים עבורם T2 היה רווחי יותר מ T1. לכן המסקנה היא שאסטרטגיית T_j עבור j>1 אפקטיבית רק עבור איזורים קטנים מאוד של מרחב הפרמטרים

2. selfish mining אינו אופטימלי עבור חלק גדול של מרחב הפרמטרים

כפי שניתן לראות מהגרף הראשון, אל מול כרייה תמימה היא אכן אופטימלית במרחב גדול אבל מהגרף השני ניתן לראות (החלק המסומן בירוק ב R2) שהיא הדומיננטית מבין שאר האסטרטגיות רק בחלק קטן של מרחב הפרמטרים

3. אסטרטגיות stubborn עשויות להרוויח עד כ 25% יותר מselfish רגיל עבור מגוון ערכים סבירים של מרחב הפרמטרים

לדוגמא עבור $\alpha = 0.4, \gamma = 0.9$, האסטרטגיה הטובה ביותר היא בעלת צפי של 23% רווחים יותר מ selfih mining וכ 63% יותר מכרייה תמימה.

עבור $\gamma = 0.9$, selfish mining מתפקדת פחות טוב מאסטרטגיות stubborn עבור $\alpha \geq 0.15$.

4. למרות שזה נראה ההפך מהאינטואיציה - עבור $\alpha \geq 0.33$ אסטרטגיית T-Stubborn מנצחת את שאר האסטרטגיות בכ 13% רווחים.

5. הרווח היחסי של stubborn mining עולה יחד עם הערכים של γ ו α

<u>שילוב Stubborn mining עם Eclipse attack</u>

לצורך הבנת חלק זה יש ראשית לקרוא את פרטי ה eclipse attack שתוארה ב 3.3.2.

כעת נעבור לחלק בו מתוארת אסטרטגיה העושה שימוש לא טריוויאלי ב eclipse attack שבוצעה בהצלחה וב stubborn mining כדי לשפר את הביצועים. זאת אומרת, מעכשיו ננית כי התבצעה התקפת eclipse attack וכי הצמתים מחולקים ל Alice, Bob והפעם גם Lucy. לוסי הם הצמתים שנמצאים בשליטת אליס כתוצאה מהתקפת eclipse שאליס ביצעה עליהם. לוסי מעבירה הודעות דרך אליס ומשם אליס מחליטה אם להעביר אותם לבוב.

סימונים נוספים לחלק זה:

- λ - כוח החישוב של לוסי
- selfishGain(x) - הרווחים היחסיים של selfish miner המסומן x

הנחות:

- הרווחים היחידים ש"נחשבים" הם ביחס לבלוקצ'יין הפומבי, ז"א - אין ערך בכך שאליס תגרום ללוסי להסכים ביניהם על בלוקצ'יין פרטי וזהו (למרות שבמציאות זה היה מאפשר לאליס לבצע התקפת double spend נגד צמתי לוסי..)
- $\alpha + \lambda < \beta$. כוח החישוב של אליס ולוסי ביחד קטן משל בוב כדי שעדיין הם יהוו פחות מ50% מהרשת
- $\alpha < \beta$. אחרת אליס פשוט לא תעביר הודעות של לוסי בכלל ואז שוב יהיה לה אפקטיבית יותר מחצי מכוח החישוב ברשת

<u>אסטרטגיות "נאיביות"</u>

1. ללא שימוש ב eclipse

באופן לא מפתיע, תוצאות זהות לניתוח ללא התקפת אקליפס

2. Destroy the eclipsed victim - להרוס את הקורבן

התעלמות מלוסי, לא מעבירים אף הודעה שלה אל בוב. מגדיל את הרווח של אליס רק מעצם העובדה שכוחה היחסי ברשת גדל. מסומן במאמר כ Destroy Lucy (D). אם אליס מחליטה להרוס את לוסי וכמו כן כורה באמצעות selfish mining אז רווחיה יהיו

$$gain_D(\alpha,\lambda) = selfishGain(\frac{\alpha}{\alpha+\beta})$$

3. Collude with the eclipsed victim - קשירת קשר נגד הקורבן

אליס יכולה להכריח את לוסי לשתף איתה פעולה. אליס תקבל בלוקים מלוסי והם יתחזקו שרשרת פרטית ביניהן. אולם, אליס תשתף בלוקים אם בוב בהתאם לאחת האסטרטגיות ה stubborn. אסטרטגיה המסומנת Collude with Lucy (C). אם אליס מחליטה לקשור קשר עם לוסי והיא כורה באמצעות selfish mining אז רווחיה יהיו

$$gain_C(\alpha,\lambda) = \frac{\alpha}{\alpha+\lambda} selfishGain(\alpha+\lambda)$$

<u>אסטרטגיות שאינן נאיביות</u>

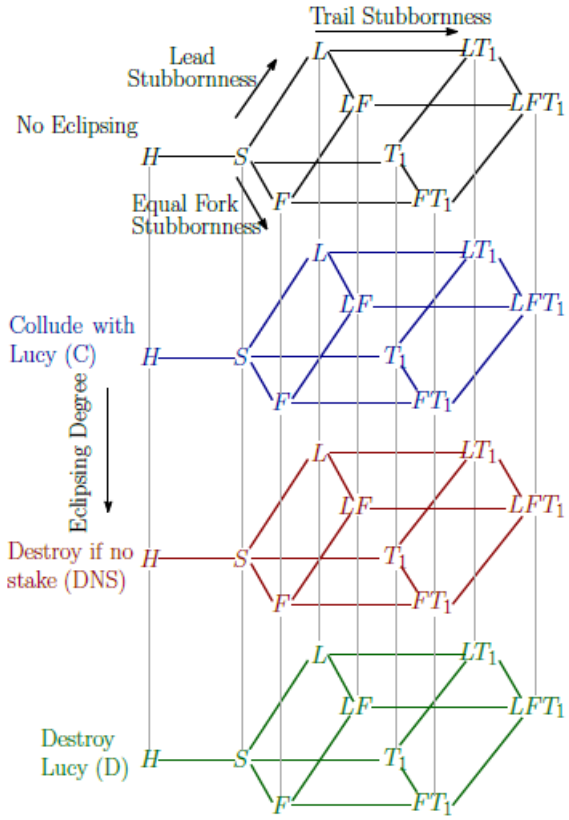
קיימות אסטרטגיות רבות בין קשירת קשר עם הקורבן להריסת הקורבן. נתאר כאן דוגמא אחת.

1. Destroy if no Stake (DNS) -

א. אליס תמיד מספרת ללוסי על ה head של השרשרת הפרטית שלה
ב. לוסי יכולה לכרות על על גבי השרשרת של אליס או על גבי שרשרת פרטית אחרת שלה
ג. כאשר לוסי מוצאת בלוק חדש אליס תפעל באופן הבא
   a. אם הבלוק נכרה על גבי השרשרת הפרטית של אליס, היא תקבל אותו
   b. אחרת, אליס לא תקבל את הבלוק של לוסי ולא תעביר אותו

תלוי במצב השרשראות של אליס ולוסי, אסטרטגיה זאת או קושרת קשר עם לוסי או הורסת את לוסי.

איור 43 שנלקח מ [8] מציג את "מרחב האסטרטגיות" המתאר את רמת ה eclipsing וכמו בחלק הקודם, את רמת העקשנות -



איור 43 - מרחב האסטרטגיות ההיברידיות

כמובן שקיימות אינסוף אסטרטגיות לא טריוויאליות. האסטרטגיות שהוצגו כאן הן אלה שיעילותן נבחנת.

<u>מידול של אסטרטגיה לא טריוויאלית</u>

עד כה נעשה שימוש במכונת מצבים די פשוטה על מנת לתאר ולנתח את האסטרטגיות השונות. על מנת לתאר אסטרטגיה לא טריוויאלית כמו DNS יש צורך בהרחבת המידול באופן פרטני עבור כל אחת מהאסטרטגיות. נציג פתרון עבור DNS-F (DNS יחד עם תהליך כרייה F-Stubborn).

האסטרטגיה מאופיינת ע״י החוקים הבאים:

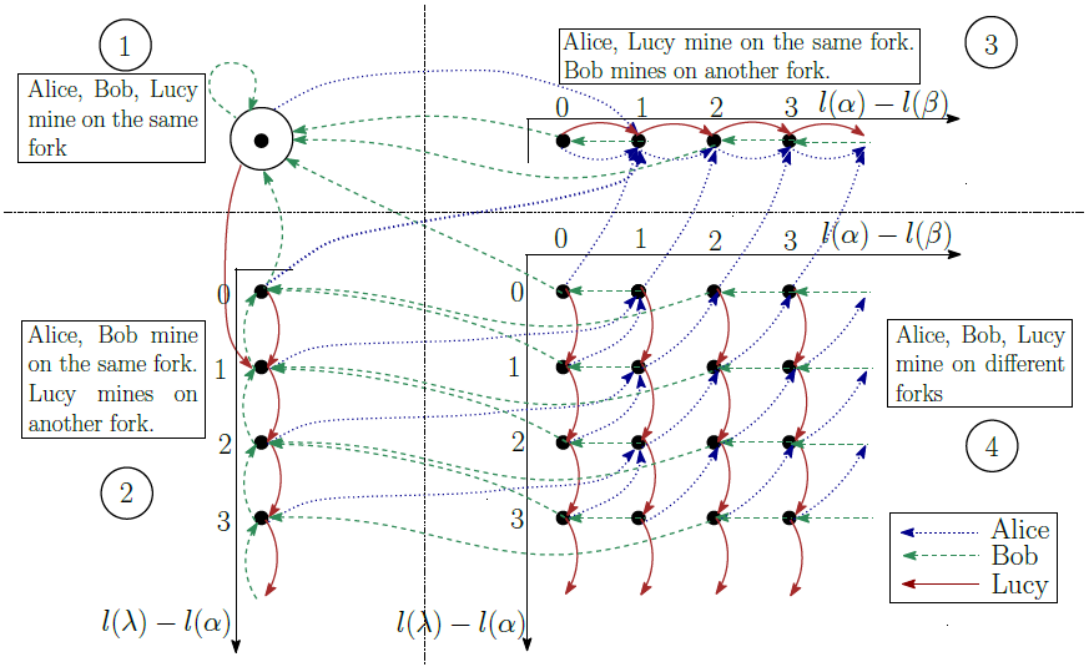
1. אליס מתחזקת שרשרת פרטית

2. מאחר ומדובר על DNS, אם אליס כורה בלוק לפני לוסי, היא משתפת אותה והן כורות ביחד עד אשר השרשרת מתפרסמת

3. אם לוסי כורה בלוק קודם, אליס מחליטה להרוס את לוסי עד אשר היא מקדימה אותה

4. מאחר ורמת העקשנות היא F, בכל פעם שאליס או לוסי מוצאות בלוק שמביא למצב של תיקו עם בוב, הבלוק לא מפורסם והן ממשיכות לכרות מעל השרשרת הפרטית

איור 44 בנשלקח מ [8] מציג מכונת מצבים המתארת זאת



איור 44 - מכונת מצבים עבור אסטרטגיה DNS-F

מכונת המצבים מחולקת ל4 איזורים.

איזור 1 - אליס וגם לוסי וגם בוב כורים על אותה השרשרת

איזור 2 - מצבים בהם אליס ובוב כורים על אותה שרשרת אבל לוסי לא. זה קורה בDNS רק אם השרשרת הזאת סטתה מהשרשרת על גביה אליס כורה ומצאה בלוקים נוספים אותם אליס לא מעבירה לבוב. אליס מחכה להדביק את הפער מלוסי אבל נכון למצבים האלה, לוסי מובילה על שניהם

איזור 3 - אליס ולוסי כורים על אותה שרשרת אבל בוב לא. השרשרת הפרטית שלהם מובילה

איזור 4 - כולם כורים על גבי שרשראות שונות. אליס מתחזקת יתרון על בוב ולוסי מתחזקת יתרון על אליס

<u>תוצאות</u>

1. גם כאן, אף אסטרטגיה לא הייתה יותר טובה מכל השאר עבור כל פרמטר

   כמו בניתוח תוצאות עבור stubborn ללא אקליפס נעשה כאן שימוש בסימולציות המסתמכות על המשוואות המתקבלות כאשר אסטרטגיה מדווחת כ״טובה ביותר״ עבור פרמטרים מסויימים אם היא קיבלה ציון גבוה הרבה יותר מאשר המתמודד שהגיע אחריה עם 95% רווח בר סמך.

   עבור ערכים קיצוניים של γ (1 או 0) האסטרטגיות הנאיביות הן הטובות ביותר בעוד עבור ערכים בינוניים (סבירים), DNS מתפקד הכי טוב.

2. בהשוואה לאסטרטגיות נאיביות, תוקף בעל יכולת eclipse המשתמש באסטרטגיה הטובה ביותר יכול להרוויח יותר באופן משמעותי.

   לדוגמא אם $\alpha = 0.4$ אז אליס יכולה להשיג כ 30% יותר רווחים בהשוואה לאסטרטגיה נאיבית כמו שילוב של אקליפס טריוויאלי עם selfish mining

התוצאות מראות שכדאי לתוקף לעשות שימוש במגוון האסטרטגיות ולעשות שימוש באסטרטגיה המתאימה כתלות בפרמטרי הרשת שכן תוקף עשוי לזכות ברווחים הרבה יותר גדולים תוך שימוש בקומבינציות שונות של שימוש ביכולת האקליפס וסוגים שונים של stubborn mining.

3. מסקנה נוספת ומפתיעה מאוד היא שעבור פרמטרים מסויימים, האסטרטגיה הטובה ביותר עבור התוקף, גורמת לכך ש Lucy מרוויחה מהתקפת ה eclipse

בהשוואה לרשת "רגילה" בה אין תוקף, הרווחיות של לוסי עולה כאשר אליס קושרת קשר איתה. באופן אינטואיטיבי, אליס ולוסי שתיהן מרוויחות מקשירת הקשר הזאת, על חשבון בוב.

מאחר ותוצאות הניסויים מצביעות על כך שהאסטרטגיה האידיאלית ברוב המקרים עבור אליס עושה שימוש בקשירת קשר עם הנתקף בהתקפת אקליפס (לוסי), זה מצביע על כך שתאורטית קיימים מקרים בהם הקורבן יעדיף להישאר eclipsed, גם אם המחיר של התנגדות לכך היא 0.

4. מימוש השוואתי של selfish miner אל מול מיינר רגיל

4.1. מטרת הפרויקט

במסגרת המאמרים שנסקרו בעבודה המסכמת הוצגו הוכחות תאורטיות לנכונות האלגוריתמים והוצגו סימולציות המתבססות על המתמטיקה אליה הגיעו. במילים פשוטות יותר, מרבית האלגוריתמים ובאופן ספציפי, selfish mining, לא נבחנו אמפירית. במקום זאת, הוצג מודל מתמטי המתאר את המערכת וניתוח הסתברותי של הצלחה.

מטרת פרויקט זה היא לממש את אלגוריתם selfish mining ולבחון את נכונותו

הפרויקט מומש במסגרת פרויקט גמר וכאן מובא תיאור תמציתי שלו

4.2. סוגיות מימוש והיקף העבודה

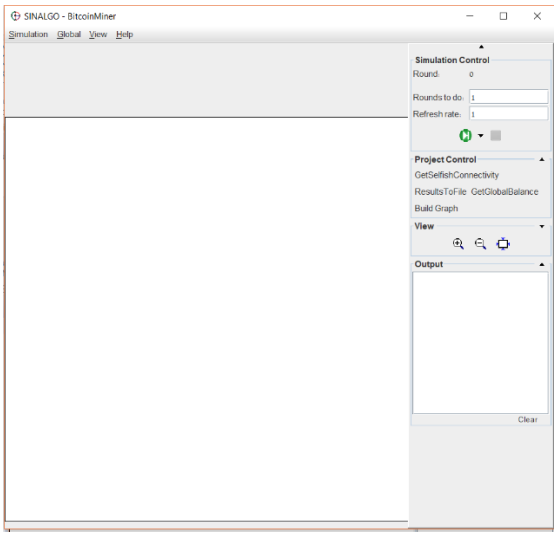
ברמת המימוש, ביטקוין הוא פרויקט open source שהוקם ע"י satoshi (ממציא ביטקוין) ומתוחזק ע"י הקהילה מאז. לפרויקט קוראים bitcoin core והוא מממש תהליך שנקרא bitcoind המתפקד כמיינר כאשר הוא מותקן על מחשב המחובר לרשת. פרויקט זה הוא מימוש המיינר המרכזי, 99% מהמיינרים מריצים אותו. מספר עובדות לגבי bitcoin core:

1. מאחר והפרויקט צמח bottom-up באופן מאוד sketchy, אין שום מקור המציג design של המערכת, תיאור הרכיבים באופן מסודר וכו. באופן כללי בקהילה כאשר מפתח מתחיל מבקש כיוון, נהוג להפנות אותו אל ה source code
2. הקוד עצמו קשה מאוד לקריאה. לדוגמא, קובץ main מכיל כ 6000 שורות קוד השייכות לאלמנטים שונים ומגוונים במערכת
3. התהליך דורש המון משאבים. על מנת להריץ תהליך אחד של bitcoind יש צורך ב2GB זכרון RAM וכ 150GB זכרון דיסק פנוי שגם הוא חייב להיות SSD. וזה עוד לפני שמדברים על תעבורה. לכן, לא כל כך ריאלי להריץ מספר תהליכים כאלה על מחשב אחד

לכן, לאחר שחקרתי את bitcoin core, הגעתי למסקנה שלמרות ששימוש בו יחסוך חלקי מימוש שונים, הוא אינו מתאים לצרכים אקדמאיים ולפרויקט הזה.
המסקנה הייתה, מימוש מיינר ביטקוין מלא מאפס המבצע את כל הפעולות הבסיסיות שמיינר מבצע. מאחר ומדובר על צומת מלא, מומש גם ארנק וכך ניתן לבצע יצירת טרנזקציות, חתימה על טרנזקציות, שליחתן ברשת, תחזוקה של קונצנזוס מבוזר באמצעות בלוקצ׳יין, ולידציה של טרנזקציות מול בלוקצ׳יין, כריית בלוקים וכו.
כמובן שבהתאם פותח selfish miner המתבסס על המיינר הרגיל ודומה לו ברוב פעולותיו מלבד תהליך הכרייה.
לצורך הרצת סימולציות נוחה, את המימוש כתבתי מעל sinalgo שהיא framework להרצת סימולציות מבוזרות הכתובה ב java. אין זה משנה את המימוש או את תוכן הפרויקט. הרעיון הוא שלאחר מימוש המיינר, הקוד נכתב בתור מחלקה היורשת ממחלקת Node של sinalgo על מנת לאפשר לנו ליצור גרפים מצמתים אלה ולאפשר להם לשלוח הודעות אחד לשני

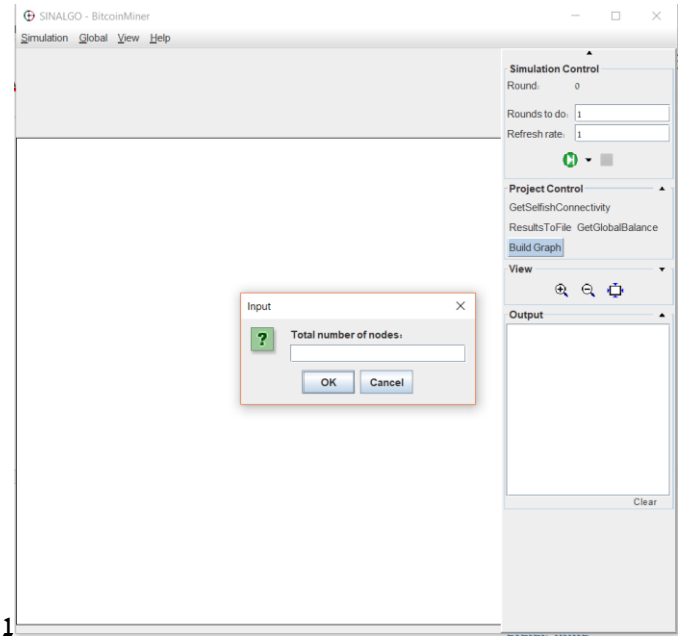
4.3. ממשק המשתמש

בעת הפעלת התוכנית יוצג מסך המשתמש. איור 45 שנלקח מהפרויקט מציג אותו



איור 45 - ממשק משתמש עמוד כניסה

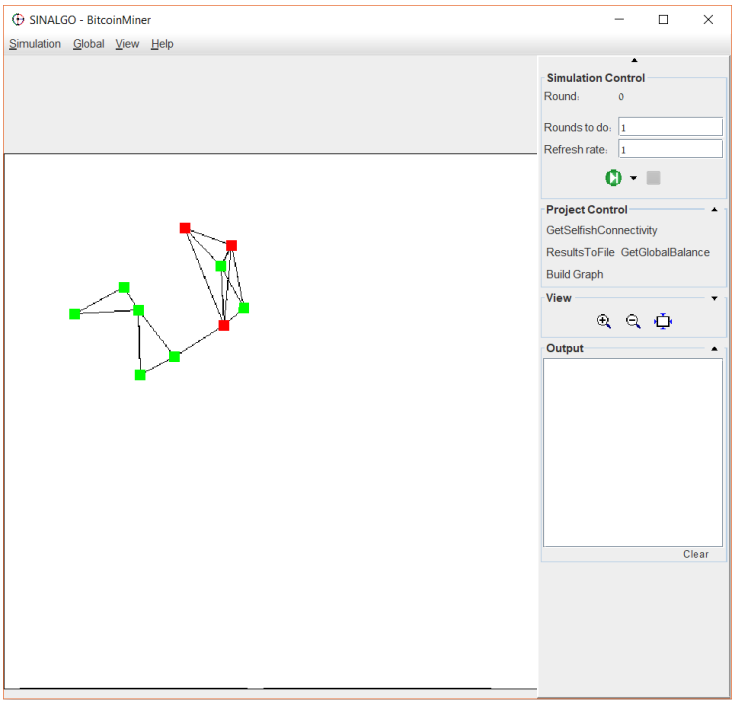
כלל הממשקים של המשתמש עם המערכת נמצאים תחת Simulation Control או Project Control. באמצעות Simulation Control ניתן להזין מספר ב text box המסומן Rounds to do וללחוץ על כפתור ה Play כדי להריץ את הרשת למספר סיבובים. במצב הראשוני, אין עדיין רשת. על מנת ליצור רשת יש ללחות על Build Graph, המשתמש יתבקש להכניס מספר צמתים בגרף ומספר צמתים שהם Selfish מתוכם



1

איור 46 - ממשק משתמש הוספת צמתים לרשת

לדוגמא אם נכניס 10 עבור מספר צמתים כולל ו3 עבור מספר צמתים selfish נקבל גרף כמו שמוצג באיור 47 שנלקח מהפרויקט



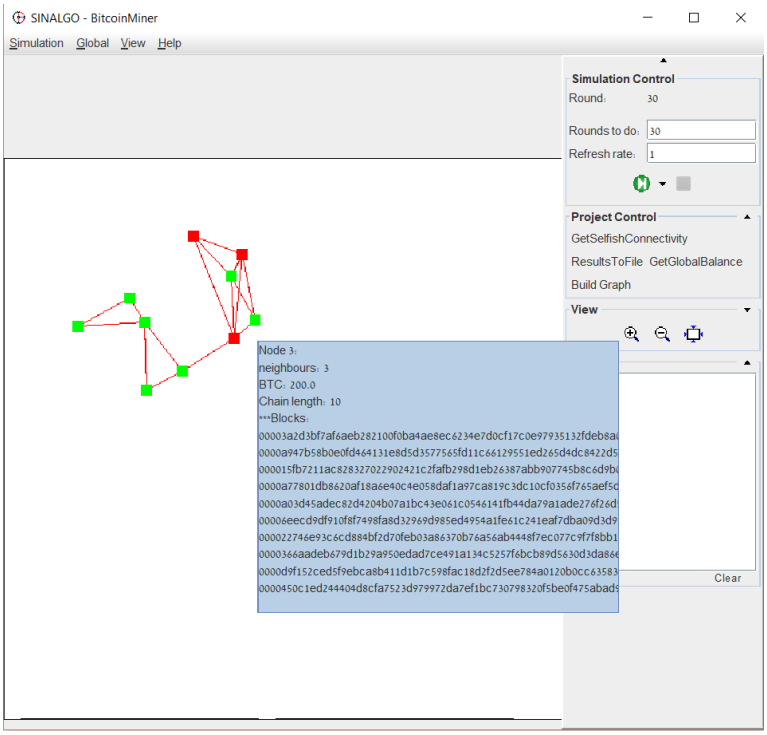
איור 47 - ממשק משתמש רשת קיימת

בכל שלב ניתן ללחוץ על GetGlobalBalance על מנת לקבל את כמות הביטקוין הכוללת שמחזיקים הצמתים ה selfish והצמתים התמימים.

בכל שלב ניתן ללחוץ על GetSelfishConnectivity שיחזיר את ערך ה selfish connectivity הממוצע בגרף - מדד זה הוא ממד שבחרתי לבחון ואסביר אותו בפרק ביצוע המדידות.

בכל שלב ניתן ללחוץ ResultsToFile כדי לכתוב את התוצאות הנוכחיות של הסימולציה לקבצים. המשתמש יתבקש לספק תיקייה (רצוי לספק path שנמצא תחת c:\users\[username] על מנת להימנע מבעיות של הרשאות כתיבה לדיסק). כל לחיצה כזאת תוביל לכתיבה של המידע ל3 קבצים שונים שהמבנה שלהם יוסבר גם הוא בפרק ביצוע מדידות.

כמובן ניתן להריץ מספר סיבובים מסויים, לעצור, לבחון את הערכים. ניתן גם "לעמוד" מעל צומת (hover) ולראות את מצב הביטקוין הנוכחי שלו, מצב הבלוקצ'יין שהוא מתחזק.
דוגמא לכך ניתן לראות באיור 48 שנלקח מהפרויקט



איור 48 - ממשק משתמש הצגת מצבו של מיינר

ניתן לראות כי הצומת שעמדנו מעליו עם העכבר הוא צומת עם מזהה 3, יש לו 200 ביטקוין (הרצנו 30 סיבובים כפי שגם ניתן לראות), אורך הבלוקצ'יין שלו הוא 10 בלוקים וניתן לראות את ההאשים שלהם

4.4. מיקום קבצי העבודה והוראות הפעלה

הפרויקט מוגש בצורת פרויקט ג'אווה שנכתב בסביבת אקליפס. לצורך הרצת הסימולציה קיימים גם jar-ים שנמצאים תחת התיקייה binaries. את קבצי הפרויקט עצמם שנכתבו לצורך הפרויקט ניתן למצוא תחת src\projects\BitcoinMiner\. שמות הקבצים והמחלקות הם אותם השמות כפי שהוצגו במסמך זה.

כל הקבצים מוגשים בתוך תיקייה שנקראת BitcoinSelfishSimulation. לצורך הרצת הסימולציה יש לפתוח cmd ולעשות dir אל התיקייה BitcoinSelfishSimulation ושם להריץ את הפקודה

java -cp binaries\bin;binaries\jdom.jar;binaries\bcprov-jdk15on-159.jar;binaries\gson-2.6.2.jar sinalgo.Run -project BitcoinMiner

לצורך נוחיות שורה זאת נמצאת בקובץ טקסט שנקרא "שורת הרצה.txt" בתיקייה BitcoinSelfishSimulation

4.5. ערכים הנמדדים

במסגרת המדידות נכתבים לקבצים מספר ערכים. כמעט כל הערכים שהגדרתי הם טריוויאליים ויתוארו החלק הבא שמסביר את פורמט הקבצים אך ראשית נסביר בקצרה על 2 מהערכים

selfishLeadRevenueRatio

מדד זה נותן תיאור כלשהו ליחס בו התגברו ה selfish miners על המיינרים התמימים.
נגדיר את

$$expectedGreedyRevenueRatio = \frac{Number\ of\ \ greedy\ nodes}{Number\ of\ \ honest\ nodes}$$

expectedGreedyRevenueRatio הוא היחס המצופה בין רווחי הצמתים ה greedy לבין רווחי הצמתים התמימים מאחר ועפ"י פרוטוקול ביטקוין הרווחים אמורים להתקבל ביחס לכוח החישוב.
נגדיר את

$$actualGreedyRevenueRatio = \frac{Sum\ of\ \ greedy\ nodes\ bitcoins}{Sum\ of\ \ honest\ nodes\ bitcoins}$$

actualGreedyRevenueRatio הוא היחס בפועל בין רווחי שני הצדדים.
לבסוף נגדיר את היחס

$$selfishLeadRevenueRatio = \frac{\text{actualGreedyRevenueRatio}}{\text{expectedGreedyRevenueRatio}}$$

כאשר selfishLeadRevenueRatio גדול מ-1, ה selfish miners מנצחים. בנוסף, יחס זה נותן לנו תחושה פי כמה גדל הפער בין הצמתים הרעים לצמתים הטובים. לדוגמא, אם selfishLeadRevenueRatio=2, ניתן להגיד כי עבור פרמטרי הסימולציה הנוכחית, כוח החישוב של selfish miner שווה לכוח החישוב של שני honest miners

selfishConnectivityRatio

בניתוח התקפת selfish mining עשו כותבי המאמר בפרמטר שהם המציאו וקראו לו γ . נזכיר כי פרמטר זה מוגדר כך:

בהינתן מצב של race (פורסמו שרשרת פרטית ושרשרת פומבית באותו אורך, במקביל) γ יהיה החלק מתוך הצמתים התמימים שיבחרו דווקא בבלוק של השרשרת הפרטית (השייכים למיינרים ה selfish)

המשמעות הקונספטואלית של γ ברורה, אולם היא תלויה מאוד בגורמים משתנים והסתברותיים כמו ה latency בין צמתים ברשת וכו.

ניסיתי לחשוב ולהגדיר פרמטר המשפיע על γ אך הוא קבוע עבור גרף מסויים כך שניתן יהיה לנסות למצוא קשר בינו לבין הצלחת ההתקפה.

הפרמטר selfishConnectivityRatio יוגדר עבור צומת בודד באופן הבא:

$$selfishConnectivityRatio(nodeId) = \frac{num\ of\ greedy\ neighbours}{total\ num\ of\ neighbours}$$

החלק היחסי של צמתים תוקפים מבין השכנים של צומת מסויים. התלות בין γ לבין selfishConnectivityRatio של כל אחד מהצמתים היא די ברורה - ככל שלצומת יש יותר שכנים שהם selfish, כך עולה הסיכוי שבמצב של race הוא ישמע קודם את הבלוק אותו פרסמה הבריכה ה selfish.

בנוסף ננסה למצוא קשר לפרמטר חזק יותר שהוא הממוצע של פרמטר זה על גבי הצמתים התמימים ברשת:

$$averageSelfishConnectivityRatio = \frac{\sum_{i \in honest\ node\ ids} selfishConnectivityRatio_i}{number\ of\ honest\ nodes}$$

בתקווה למצוא קשר בין ערך ממוצע זה לבין סבירות הצלחת ההתקפה

4.7. מוטיבציה לבחירת הערכים והמדידות

באמצעות הסימולציות נרצה ליצור דגימות של ערכים שונים של תכונות הרשת בזמנים שונים ולנסות למצוא קורלציה כלשהי בין תכונות אלה לבין מידת ההצלחה של ההתקפה.

באופן ספציפי, ננסה לענות על השאלות

1. באיזו סבירות ההתקפה מצליחה?
2. כתלות באילו פרמטרים ההתקפה מצליחה (הצלחה במובן בינארי)?
3. כמה הפרמטרים האלה משפיעים על מידת ההצלחה (הצלחה במובן לא בינארי אלא השוואה של גובה הרווחים)

4.8. תוצאות

סומלצו הרצות עבור 20 גרפים אקראיים שונים בגדלים 10-7 עם 4-2 צמתים selfish המהווים תמיד פחות מ50% מכוח החישוב. החלוקה נעשתה באופן הבא:

- 3 גרפים בגודל 7 עם 2 צמתים selfish, 3 selfish ו3 selfish
- 3 גרפים בגודל 8 עם 2 צמתים selfish, 3 selfish ו3 selfish
- 4 גרפים בגודל 9 עם 2-4 צמתים selfish, 4 selfish
- 10 גרפים בגודל 10, 2 מהם עם 2 צמתים selfish, 4 מהם עם 3 צמתים selfish ו4 מהם עם 4 צמתים selfish

נלקחו בחשבון רק גרפים קשירים שנוצרו מאחר וגרף שאינו קשיר לא מבטא נכון את רשת ביטקוין וגם לא "שכונה" ברשת ביטקוין מאחר ועדכונים בקונצזוס חייבים להגיע לכל חלקי הרשת לצורך תפקוד תקין.

עבור כל גרף בוצע:

1. הרצת 200 סיבובי סימולציה
2. לקיחת דגימה כל 10 סיבובים

סה"כ נלקחו 400 דגימות (מצורפות כקבצי אקסל יחד עם הגשת פרויקט זה). לאחר יצירת הדגימות נותחו התוצאות. הגרפים ועיקרי התוצאות יוצגו כעת.

4.8.1. מספרים כלליים

מתוך 20 הרשתות השונות שסומלצו, הדגימה האחרונה מראה כי המיינרים ה selfish הרוויחו יותר מערכם היחסי ב13 מהרשתות. תוצאה זאת נראית חלשה יחסית למספרים המצופים. אולם, אם בוחנים את המקרים בהם averageSelfishConnectivityRatio > 0.5 (בממוצע לפחות חצי מהחיבורים של צומת תמים הם לצומת selfish) אז ברשתות אלה, המיינרים ה selfish ניצחו ב100% מההרצות. דבר המצביע על תלות חזקה מאוד בפרמטר זה. פה שווה להזכיר את התקפת ה eclipse שתוארה בעבודה המסכמת שהראתה שהקושי ליצור מספר חיבורים גדול של צומת תוקף לצמתים הוא לא גדול מאחר וניתן לחבר את אותו הצומת התוקף באמצעות הרבה חיבורים לאותו צומת תמים אחר וכן לצמתים אחרים ועלות חישובית ורשתית נמוכה.

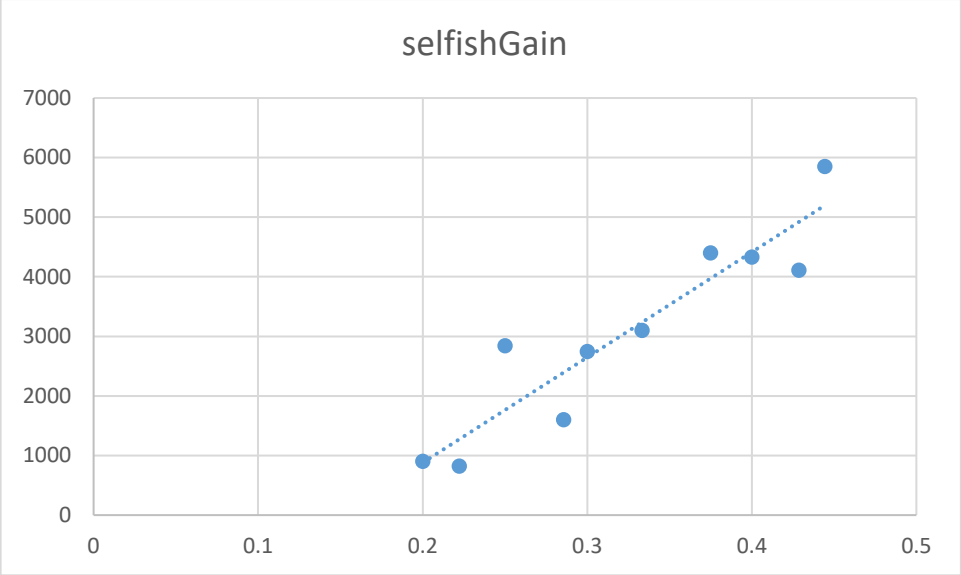
ברוב המוחלט של ההרצות ברגע שה selfish miners הצליחו לצבור פער כלשהו, הם גם הצליחו להחזיק בו ולהגדיל אותו, כצפוי מהתקפה זאת.

4.8.3. תוצאות סיכומיות

לצורך חלק זה איחדתי את המידע בכל קבצי ה Global של ההרצות השונות כדי לנסות לקבל תחושה לגבי תוצאות כלליות לרשתות מהסוג הזה ולהתקפה המתוארת ולהציגן.

נתייחס בניתוח לדגימה האחרונה שנלקחה עבור כל גרף כדגימה המייצגת את "תוצאת ההרצה".

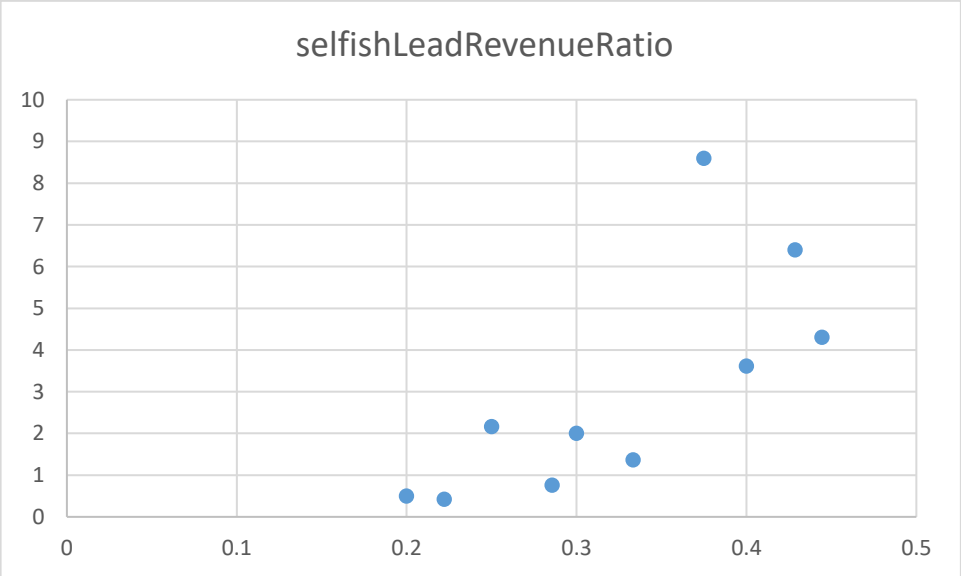
איור 49 שנלקח מהפרויקט מציג גרף של רווחי הselfish miners כתלות בחלקם היחסי מתוך כלל הצמתים ברשת (selfishRatio):



איור 49 - רווחי ה selfish miners היחסיים

מאחר ולא ביצענו ניסויים על המון גדלים שונים אז יש לנו יחסית הרבה דגימות השייכות למעט יחסים כאלה (יש גם הגבלה שחלקם היחסי של selfish miners יהיה תמיד פחות מ 0.5) אבל בגדול ניתן לראות את קו המגמה הממוצע שמראה את הקורלציה הצפויה בין חלקם היחסי של המיינרים ה selfish מתוך כלל האוכלוסיה לבין רווחיהם.

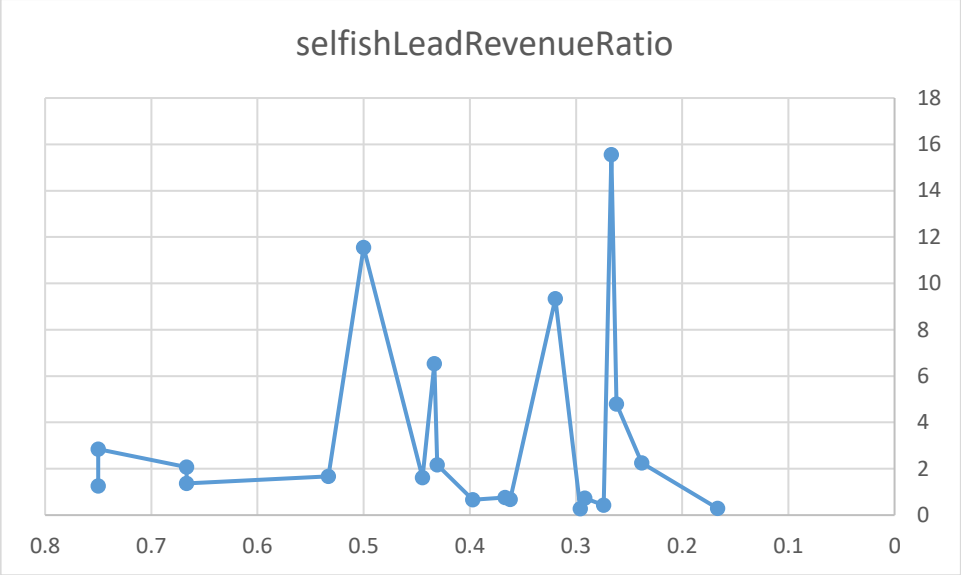
איור 50 שנלקח מהפרויקט מציג זאת



איור 50 - selfishLeadRevenueRatio כתלותב selfishRatio

פיזור גדול יותר אבל תוצאה דומה. ניתן לראות גם שכאשר חלקם היחסי של ה selfish miners מגיע לאיזור ה 0.3 אז ההתקפה נעשית יעילה (כבר בטווח היחסית קצר שסימולציה זו בוחנת) מאחר וערכו של selfishLeadRevenueRatio גדול מ1.

ננסה לבחון את הערכים האלה ביחס לקשירות של ה selfish miner באמצעות פרמטר selfishConnectivityRatio.



איור 51 - selfishLeadRevenuwRatio כתלות ב selfishConnectivityRatio

נראה שלא ניתן להסיק קשר כל כך ישיר כפי שקיוויתי בין השניים. בכל זאת כן ניתן להסיק שמעל selfishConnectivity של 0.5 זה game over והמיינרים ה selfish מנצחים ב100% .
תוצאה זאת כשלעצמה היא לא טריוויאלית מאחר וזה תנאי הרבה יותר חלש משליטה ממשית בחלק גדול מהרשת (כפי שהוסבר בעבודה תחת נושא התקפת eclipse) ומתוצאות הניסויים נראה שזה מבטיח בהסתברות מאוד גבוהה רווחים גבוהים בכל שלב.

תוצאות נוספות של ניסויים ניתן לראות בדו״ח הפרויקט

5. סיכום ומסקנות

ביטקוין הינו מטבע אלקטרוני בעל שימוש רחב בעולם. אנשים רבים משקיעים את הונם בקניית המטבע או בציוד כרייה. עבודה זאת הציגה מספר התקפות קיימות על הרשת. התקפות אלו עלולות לפגוע בכל משתמשי הרשת בין אם זה בפרטיותם באמצעות התקפות דה-אנונימיזציה שהוצגו, בניצול התקפת eclipse לצורך double spending או dos למשתמשים או באופן יותר קשה לזיהוי - בבזבוז משאביהם על ציוד כרייה לשווא מאחר ותוקפים ישתמשו ב selfish mining על ידי לכרות וירוויחו על חשבון, בזול.
קיימות סיבות רבות בגינן לא ניתן לסגור בקלות את הפרצות שהעבודה סוקרת. חלקן קשור בתמיכה לאחור של צמתי הרשת עליהן מותקן הקליינט, חלקן בכלל נובעות מהמבנה הלוגי של הפרוטוקול והן אף קשות יותר לטיפול. לכן נכון להיום מטרתה של העבודה זאת היא בעיקר להעלות את המודעות עד אשר יתוקנו הפגמים.
בחלק הממשי מומשה סימולציה של רשת ונבחנה התקפת selfish mining רגילה. כיוון טבעי לעבודת המשך הוא לבחון באופן דומה את כלל האלגוריתמים ההיברידיים שנסקרו בעבודה בפרק 3.3.3. - Stubborn mining. אולם לצורך בחינת אלגוריתמים אלה יש צורך בגרפים גדולים יותר, על כן יהיה צורך ככל הנראה במימוש מנגנון paging אל הדיסק של הבלוקצ׳יין של מיינרים על מנת לעמוד במגבלת הזכרון

6. רשימת מקורות

1. JA Garay, A Kiayias, N Leonardos (June 23, 2017). The Bitcoin Backbone Protocol: Analysis and Applications..
2. F Tschorsch, B Scheuermann. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. IEEE Communications Surveys & Tutorials (Volume: 18, Issue: 3, 16)
3. S Nakamoto (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Bitcoin contributers
4. Nicolas T. Courtois, Lear Bahack. On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency. arXiv: 1402.1718
5. Ayelet Sapirshtein, Yonatan Sompolinsky, Aviv Zohar. Optimal Selfish Mining Strategies in Bitcoin. FC 2016: Financial Cryptography and Data Security pp 515-532 (23 July 2016)
6. Ittay Eyal, Emin G¨un Sirer. Majority is not Enough: Bitcoin Mining is Vulnerable. FC 2014: Financial Cryptography and Data Security pp 436-454 (09 November 2014)
7. Ethan Heilman and Alison Kendler, Boston University; Aviv Zohar, Eclipse Attacks on Bitcoin's Peer-to-Peer Network. in the Proceedings of the 24th USENIX Security Symposium August 12–14, 2015 • Washington, D.C.
8. Kartik Nayak, Srijan Kumar, Andrew Miller, Elaine Shi. Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack. 2016 IEEE European Symposium on Security and Privacy (EuroS&P), (24 March 2016)
9. Michael Fleder, Michael S. Kester, Sudeep Pillai, Bitcoin Transaction Graph Analysis, arXiv:1502.01657 (5 February 2015)
10. Alex Biryukov, Dmitry Khovratovich, Ivan Pustogarov, Deanonymisation of clients in Bitcoin P2P network. Proceedings of CCS '14 the 2014 ACM SIGSAC Conference on Computer and Communications Security (7 November 2014)
11. Pedro Franco, Understanding Bitcoin: Cryptography, Engineering, and Economics, Wiley Publishing (24 October 2014)

Abstract

In this paper we will review the structure of the crypto-currency Bitcoin and the main attacks this network is vulnerable to.
The paper will deeply review the algorithms implemented in the network that allow the crypto-currency's existence and provide various features such as: currency transactions between users, bitcoin mining and network privacy.
Different, widely used, bitcoin client implementations will also be reviewed and variants of them will be presented as means to execute attacks on the network that can result in unfair maximization of profits (selfish mining), performing double spending, denial of service of another user in the network and de-anonymization of another user.
The reviewed topics are as followed: General Bitcoin network information [3, 2, 1, 11] Privacy and De-anonymization attacks [9, 10] Attacks that target Bitcoin's security such as Eclipse and Selfish mining attacks [4, 5, 6, 7, 8]

The practical part of this work includes implementation of a bitcoin network simulation, done as the final project. the simulation includes a regular bitcoin client (miner) that acts as a fully functioning bitcoin node that handles transactions and mines bitcoins, a miner that does all these things but also implements the selfish mining attack and mines according to a strategy that attacks the network and is described in one of the papers and a simulation that allows us to examine the algorithm on different bitcoin network configurations.
Finally, results that show the selfish mining's efficiency are presented

The Open University of Israel
Department of Mathematics and Computer Science

Attacks on Bitcoin Network

Thesis (Final Paper) submitted as partial fulfillment of the requirements
towards an M.Sc. degree in Computer Science
The Open University of Israel
Department of Mathematics and Computer Science

By
Roei Dimintshtein

Prepared under the supervision of Prof. (Dr.) Ehud Gudes

July 2018