

תמיר אלביליה 325832681



פתיחה

רקע:

באינטרנט הנוכחי קיימים מספר רב של שירותים שונים ומגוונים, אך למרות זאת, במשותף לכל השירותים הללו הוא שיש לסמוך על שרת מרכזי על מנת לעשות בהם שימוש. אך מדוע לנו לסמוך על שרת מעבר למסך שאכן ינהג בהגינות כלפי כל משתמשיו? באמצעות הקריפטוגרפיה ניתן לעקוף את מתן הסמכות והצורך בשרת מרכזי למתן שירות ובכך למגר את עריצות השרת המרכזי.

הרעיון:

מטרת הפרוייקט היא יצרת בלוקצ'יין – מסד נתונים המאוחסן ללא צורך בשרת מרכזי (Decentralized) כך שלא יהיה כוח רב לאף שרת יחיד שבו כל המחשבים המשתתפים במערכת מחוברים ברשת P2P מפוזרת. המערכת תוכל לתפקד כראוי גם כאשר חלק מן השרתים מנסים לתקוף את המערכת. בפרויקט זה מסד הנתונים יתמוך בהעברה של כסף דיגיטלי (Cryptocurrency) והפצה של הודעות ברבים.

שימוש:

המערכת שימושית עבור כל מי שמעוניין להעביר כסף דיגיטלי ללא הרצון לסמוך על שרת מרכזי כזה או אחר או לשדר מידע ברבים ללא צינזור.

הגדרת הבעיה ופתרונה

כעת באינטרנט שירותים כמו העברת כספים והפצת הודעות מחייבות מתן סמכות בשרת מרכזי. כדי שנוכל להסיר את מתן הסמכות מהמערכת ראשית יש לעבור לרשת עמית לעמית שבה כל מחשב במערכת מסוגל להתחבר לכל מחשב אחר במערכת ושאינה כוללת שרת מרכזי. כל המחשבים המחוברים במערכת יקליטו את רצף האירועים והעסקאות שקרו במערכת. אך מכך נובעת בעיה חדשה, כיצד ניתן להגיע לקונצנזוס ולהחליט מהו רצף האירועים האמיתי שקרה. על מנת לפתור בעיה זו על מסד הנתונים להיות חסין מפני שינוי ספונטני או זיוף וזהום של כל אחד מהעמיתים במערכת. נעשה זאת באמצעות הגדרת שרשרת בלוקים (בלוקצ'יין). שרשרת הבלוקים תהיה מורכבת מאוסף בלוקים שבו כל בלוק מקושר לבלוק הקודם לו באמצעות hash. כל בלוק כולל בתוכו את הhash שלו עצמו ואת הhash של הבלוק הקודם לו. מכיוון שהhash הוא בהגדרה ייחודי עבור כל בלוק, מתקבלת התופעה שעל מנת לשנות בלוק יחיד בשרשרת הבלוקים, יש לשנות את כל הבלוקים אחריו. בכדי להפוך תופעה זו לכלי הגנה מפני שינויים בשרשרת נשתמש בproof-of-work (הוכחת עבודה) ונגדיר שבשביל שhash של בלוק יהיה תקין, על הhash להיחשב כתקין, עליו לקיים ; יש לשים לב שמשום שהפלטים פונקציית הhash היא פסודו-אקראית, ההסתברות לקבל hash שעובר proof-of-work בקושי n היא . לכן כאשר מספר האפסים הנדרשים בתחילת כל hash גדול, שינוי של כל בלוק בשרשרת נעשה בלתי אפשרי עבור כל תוקף שאין לו לפחות כסך מחצי מכוח המחשוב במערכת, זאת משום שתוקף שכזה יצטרך לכרות מחדש את כל הבלוקים שאחרי הבלוק שברצונו לשנות ובו זמנית לעקוף את שאר חברי המערכת שעובדים תמידית להרחיב אותה. כאשר אין לתוקף שכזה מספיק כוח מחשוב, הוא לא יוכל לעקוף את עבודתם של שאר מחשבי הרשת. עקב כך ניתן לקבוע את השרשרת הנכונה כשרשרת המאסיבית ביותר אשר להכינה היה נדרש את כוח המחשוב הרב ביותר. מתקבל קונצנזוס משום שכל שאר המשתתפים במערכת ישתפו אחד עם השני על העדכונים ששמעו ממשתפים אחרים וישתמשו בשרשרת המאסיבית ביותר כשאר כל המשתתפים. כל אחד מהבלוקים בשרשרת מכילים העברות (transactions). כל העברה מסמלת פעולה שבה משתמש מעביר מהכסף האלקטרוני הרשום על חשבונו למספר חשבונות אחרים. על מנת לוודא שמחבר ההעברה הוא אכן בעל החשבון נשתמש בפתרון הכולל חתימות דיגיטליות ובכדי למנוע בזבוז כפול של כספים שכבר בוזבזו, נשתמש במבנה נתונים שמאפשר אכסון וארגון של כל ההעברות ההכשרות שתוצאותיהן עוד לא בוזבזו.

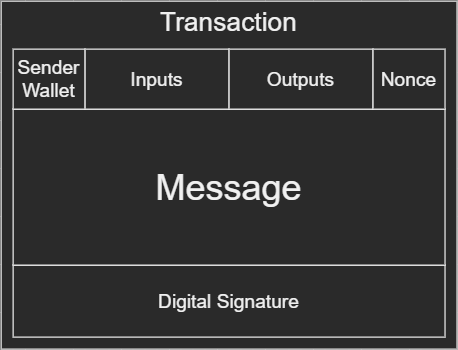
מבנה הבלוקצ'יין

העברה – Transaction

בבלוקצ'יין העברה היא המידע הבסיסי שמסד הנתונים מכיל, כל העברה היא הצהרה על מידע שיש לשמר בבלוקצ'יין. העברה מכילה הכרזה על העברת מטבעות דיגיטליים מארנק אחד לארנק אחר. ההעברות ישודרו בפומבי אל משתמשי הרשת על מנת שיועברו על להעברות מספר תנאים שיש לעמוד בהם על מנת לממש רכיב זה בבלוקצ'יין.

1. כל העברה חייבת להיות מיוצרת על ידי המקור המצוין בה בהכרח.
2. יש למנוע מכלל הלקוחות לשנות את התוכן של העברה לאחר שפורסמה ברשת
3. נדרשת היכולת להבדיל בין העברה אחת להעברה אחרת ללא תלות בתוכן ההעברה.

הינה תרשים של Transaction המתארת את מאפייני ההעברה:



* Sender wallet – הארנק של שולח ויוצר העברה והארנק של הגוף בהעברה הארנק יהווה מפתח ציבורי במערכת הצפנה א-סימטרית RSA.
* Inputs – מילון המכיל את כל מקורות ההעברה וערכם.
* Outputs – מילון המכיל את כל ארנקי מקבלי ההעברה וחלקם ממנה.
* Message – ההודעה שמכילה ההעברה, אורך מקסימלי של 222 תווים.
* Sender’s Digital Signature – החתימה הדיגיטלית של מחבר ההעברה, החתימה כוללת את כל העברה ומושפעת מערך הNonce. מטרתה להבטיח שימור של הערכים בהעברה ולהבטיח ששולח ההעברה הוא גם המחבר שלה.

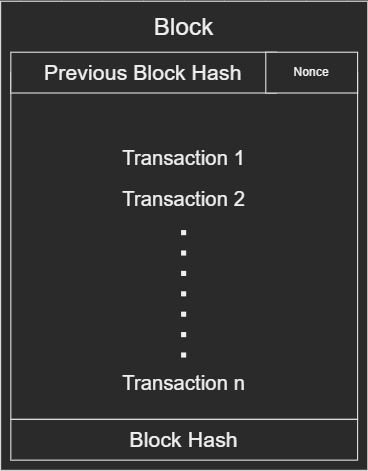
אתגרים במימוש ופתרונות

* כיצד ניתן למנוע מגוף זר לבצע העברה עבור לקוח אחר? על מנת לפתור בעיה זו נפנה למושג בקריפטוגרפיה הנקרא חתימה דיגטלית. כל לקוח ייצר צמד מפתחות אחד פרטי ואחד ציבורי, כאשר לקוח מעוניין לחבר העברה, הוא יצמיד אליה את המפתח הציבורי שלו (הוא גלוי לכולם). בסוף ההעברה ייצר המחבר חתימה דיגטלית המושפעת מגוף ההעברה ויצרף אותו אל סוף ההעברה. בסופו של דבר תתקבל מערכת שבה ניתן לוודא שהעברה מסוימת אכן חוברה על ידי השולח. כל לקוח כעת יוכל לאמת זאת באמצעות המפתח הציבורי המצורף בהודעה, והחתימה הדיגיטלית בסוף ההודעה.
* החתימה הדיגטלית גם פותרת בעיה נוספת, היא מונעת מכול זר לשנות את התוכן הקיים בה, זאת משום שכל שינוי של התוכן אמור לשנות את ערך החתימה הדיגטלית ולא ניתן לשנות את ערך החתימה משום שהמפתח הפרטי נדרש על מנת לייצר אותה, אך אין לאף זר מלבד השולח את אותו מפתח הפרטי. כל שינוי של התוכן יבטל את ההתאמה של המפתח הציבורי שצירף השולח לחתימה הדיגיטלית של ההעברה.
* כיצד נוכל לדעת אם העברה כלשהי היא חוקית ויש למחבר מספיק יתרה לבצע אותה? בכדי שההעברה תיחשב חוקית עליה לקיים את חוק שימור הכסף בinputs וoutputs שלה, כלומר sum(inputs) – sum(outputs) = 0.
* כיצד נדע איך כל אחד מהמקורות אכן על השרשרת ואינו בוזבז? שרשרת הבלוקים היא גדולה מאוד ולעבור על כולה עבור כל מקור זה אינו פתרון ממשי. על מנת לאכסן את מצב ההעברות שאינן בוזבזו נשתמש במבנה נתונים יעיל AVL שבו נאכסן את מצב כל הoutputs של כל העברה שהוכנסה לשרשרת באופן ממוין לפי החתימה הדיגיטלית של ההעברה אליהם הם משתייכים. בעץ AVL הסיבוכיות להוסיף או למצוא איבר במבנה הנתונים היא log(n) כאשר n זה מספר ההעברות בשרשרת. עקב כך נגדיר שמקור הוא חוקי רק כאשר מופיע בעץ ומחבר ההעברה הוא בעל המקור.

בלוק – (Block)

בלוק מכיל אוסף של העברות ומטרתו לשמר את סדר האירועים שבו העברות התקבלו. כל בלוק מכיל hash שלו עצמו וhash של הבלוק הקודם לו. על מנת שבלוק יהיה תקין אוסף ההעברות שלו חייב להיות תקין. על מנת שבלוק יתקבל על ידי משתתפי הרשת על hash הבלוק לעבור proof-of-work.

הינה תרשים של Block המתאר את מאפייני הבלוק:



* Previous Block Hash – הhash של הבלוק הקודם
* Nonce – ערך טקסט אקראי שמטרתו לשנות ולהשפיע על hash הבלוק, נקרא גם פתרון הבלוק
* transactions – רשימה של כל ההעברות שהבלוק מכיל.
* Block Hash – הhash של הבלוק הנוכחי.

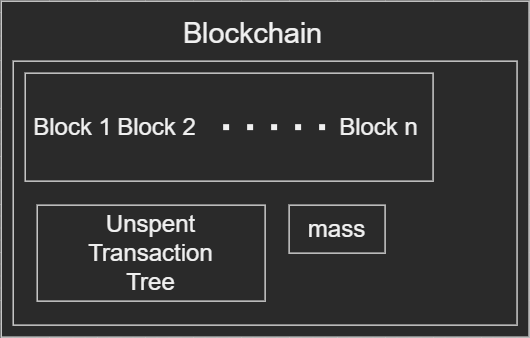
מאפיינים ועקרונות

* מכיוון שהhash של הבלוק מכיל את הhash של הבלוק הקודם לו כאשר משנים בלוק במקום הn, פעולה זו מבטלת את כל הבלוקים אחריו משום שהPrevious Block Hash שלהם אמור להשתנות בהתאמה ולכן גם הhash שלהם. כאשר על הhash.
* כאשר מייצרים בלוק אין זה סביר שהhash שיתקבל יתאים לproof-of-work הרצוי על ידי הרשת. לכן על מנת לשנות ולהשפיע על הhash, נשנה את ערך זה עד אשר יתקבל hash מתאים. תהליך זה נקרא כרייה והוא לוקח עוצמת מחשוב וזמן רבים ביחס לקושי הproof-of-work הרצוי.
* על מנת לעודד כורים להשקיע זמן וכוח מחשוב בכריית בלוקים, כאשר בלוק נכרה, לכורה הבלוק היכולת להוסיף העברה מיוחדת שמקורה אינה העברה קיימת ובכך מקבל הכורה פרס על עבודתו. גודל הפרס נקבע לפי מקום הבלוק בשרשרת. הגודל התחילי של הפרס שווה ל727 וכל 52560 בלוקים נחתך בחצי. מכיוון שמקור כל המטבעות בשרשרת מכריית בלוקים, ניתן לחשב את מספר המטבעות הכולל

שרשרת בלוקים (בלוקצ'יין) – Blockchain

שרשרת הבלוקים היא מבנה הנתונים העליון והיא מכילה אוסף של בלוקים המסודרים לפי ההhashes שלהם כך שעבור block n הPrevious Block Hash שלו שווה לHash של הבלוק הקודם לו בשרשרת והPrevious block hash של הבלוק העוקב לו שווה לBlock Hash שלו. לשרשרת הבלוקים גם עץ AVL שבו היא שומרת על איזה העברות כבר בוזבזו.

הינה תרשים של Blockchain המתארת את מאפייני שרשרת הבלוקים:

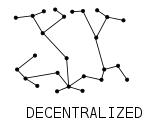


* chain – רשימה של הבלוקים שהשרשרת מכילה
* Unspent Transaction Tree – עץ AVL העוקב אחר אלו מקורות חוקיים עוד לא בוזבזו.
* mass – סכום הproof-of-work של כל הבלוקים בשרשרת.

מבנה התקשורת

התקשורת בפרויקט נעשית באמצעות רשת עמית לעמית (P2P) שבה כל עמית (peer) יוצר חיבורים ומקבל חיבורים במקביל. מטרת הרשת היא להיות מפוזרת ככל שאפשר ולכן אל לנו להשתמש בשרת מרכזי ורצוי שהרשת תהיה מפוזרת ככל שאפשר.

להלן ציור מופשט של מודל של רשת מפוזרת:



שרת - Server

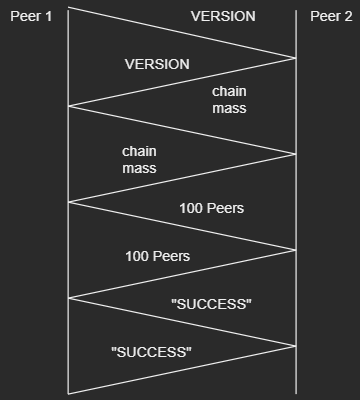
השרת מטרתו לנהל את כל החיבורים עם שאר העמיתים שאליהם עמית כלשהו מחובר. כל עמית פותח במחשבו שרת שביכולתו ליצור ולקבל חיבורים במקביל. השרת מתחיל להאזין לחיבורים לפי כתובת הIP והפורט שצוינו בקובץ הconfig של הפרויקט (פורט ברירת המחדל הוא 22222). לשרת כמות מוגבלת של חיבורים שלהם מוכן להסכים, פרמטר זה מוגדר גם כן על ידי קובץ הconfig וברירת המחדל שלו היא 150. כאשר מתבצע חיבור עם עמית אחר השרת מוסיף אותו למילון העמיתים שאליהם מחובר ביחד עם אובייקט הConnection שיפתח עבור אותו עמית. בעת פתיחת הConnection שני העמיתים עוברים תהליך של תסדיר.

חיבור – Connection

אובייקט החיבור מטרתו לנהל כל חיבור בין עמית יחיד לעמית יחיד בלבד ואת התקשורת ביניהם. בתחילת החיבור נפתח Connection run thread שבו מאזין החיבור להודעות מהעמית המחובר. לאחר מכן מתקיים תהליך של תסדיר setup שבו שני העמיתים המחוברים מחליפים ביניהם מידע לגבי עצמם. לאחר התסדיר החיבור יכול לשמש לצרכיהם של כל אחד מהעמיתים אשר יכולים לשלוח פקודות לעמית המחובר בהתאם לצרכיהם ומצב השרשרת שלהם.

תסדיר

התסדיר מתקיים בעת פתיחת חיבור. והוא תהליך של העברת מידע על כל אחד מהעמיתים. יש לשים לב שמכיוון שהתוכנה היא סימטרית, על מנת לשמור על סדר דיבור הגיוני, נגדיר שהעמית הראשון לדבר הוא האחד בעל כתובות הIP הגדולה יותר. להלן תרשים של שיחת התסדיר:



ראשית מחליפים שני העמיתים את גרסת האפליקציה שלהם. העמיתים ממשיכים בשיחה רק אם הגרסה שלהם זהה, אחרת החיבור נסגר. הגרסה מוגדרת בקובץ הconfig של הפרויקט.

לאחר מכן הם מחליפים את מסת השרשראות שלהם, כל עמית שומר את מסת השרשרת של העמית שאליו הוא מחובר באובייקט הConnection.

לאחר מכן כל אחד מהם שולח עד 100 עמיתים בצורה של כתובות IP מברכת העמיתים המוכרים שלו אשר נבחרו אקראית. כל עמית יוסיף לברכת העמיתים המוכרים שלו את העמיתים ששותפו אתו כעת. שיתוף זה מרחיב את כמות המחשבים שעמית מסוים מכיר ברשת ובכך תורם לפיזור שלה.

לסיום כל עמית שולח "SUCCESS" להראות שהתהליך הסתיים בהצלחה.