בוחן בית בקורס מבוא לטכנולוגיית בלוקצ'יין
ולדיס מרקין 317273969

1. השאלה מתייחסת ל ECC
   a. החתימה הדיגיטלית באמצעות ECC מתבצעת על ידי מכפלות על גבי Elliptic Curve. הסבר במילים כיצד ניתן להפוך הודעה שרוצים לחתום עליה לנקודה על גבי העקומה האליפטית?
   b. הסבר במילים את תהליך החתימה ב ECC ומה מאפשר לכולם לאמת שהחתימה אכן בוצעה על ידי בעל הארנק שמעביר את המטבעות?
   c. ציין שלוש סיבות מרכזיות מדוע ב secp256k1 משתמשים בחשבון modulo:

$y^2 = x^3+7$ ולא פשוט במשוואת העקומה $y^2 \bmod p = (x^3+7) \bmod p$

תשובה:

1)

a. תהליך להפיכת הודעה שרוצים לחתום עליה לנקודה על גבי העקומה האליפטית:

1. לוקחים את ההודעה שמורכבת משדות של מחרוזות ומספרים המכילה בין היתר את כתובת המקור והיעד, timestamp, טרנזקציה וכו'.
2. משרשרים את כל השדרות אחד לשני ויוצרים מחרוזת אחת ארוכה.
3. עושים Hash למחרוזת SHA-256 או Keccak-256 לצורך הדוגמה. ← Hash (Message) ומקבלים מספר בעל 256 ביט קבוע לפי הפונקציה שאנו בוחרים להשתמש.
4. ה Hash של ההודעה (256 ביט) הוא למעשה כאורדינטת ה- x של נקודה על העקומה.

b. תהליך החתימה ב ECC:

1. הופכים הודעה שרוצים לחתום עליה לנקודה על גבי העקומה האליפטית (סעיף a).
2. מגרילים מספר רנדומאלי d המשמש למפתח פרטי. המפתח הפרטי ב ECC הוא integer בטווח של העקומה (256-bit integers בדרך כלל) .
3. מכפלת המפתח הפרטי ב Hash של ההודעה שרוצים לחתום עליה היא החתימה. כלומר, Signature = P*d והחתימה היא גם נקודה על העקומה האליפטית מכיוון שהמכפלות על העקומה האליפטית תמיד נותנות נתונה על העקומה האליפטית (נובע מההגדרות של חיבור וכפל של נקודות על העקומה).

ניתן לאמת שהחתימה אכן בוצעה ע"י בעל הארנק שמעביר את המטבעות בדרך הבאה:

1. נגדיר K מפתח ציבורי שנוצר מהמכפלה של d (המפתח הפרטי) ב-G (נקודה מסוימת על העקומה האליפטית) כאשר K ו G ידועים לכולם.
2. ה Validator מבצע פעולת Hash על ההודעה ← Hash(Message) = P ומקבל נקודה בת 256bit על העקומה האליפטית.
3. כעת עליו לבצע 2 פעולות:
   a. מכפלה של המפתח הציבורי ב Hash של ההודעה ← P*K.
   b. מכפלה של החתימה של ההודעה בנקודה G ← Signature*G.
4. 2 המכפלות צריכות להיות P*d*G מכיוון ש:
   a. P*K = P*(d*G)
   b. Signature*G = (P*d)*G
5. במידה והמכפלות שוות, ה-Validator יודע ששום דבר לא שונה והחתימה אכן בוצעה ע"י בעל הארנק. בהנחה והתוקף שינה את היעד של החשבון לצורך הדוגמה, כשה-Validator יעשה Hash על ההודעה הוא יקבל P שונה מה-P של בעל החשבון שחתם על ההודעה. ברגע ש ה-P יהיה שונה, המכפלה P*K תשנה ולא תהיה זהה למכפלה של Signature*G.

c. sekp256k1 משתמשים בחשבון mod על משוואת העקומה האליפטית מפני ש:

- לא נרצה לקבל נקודה על העקומה שהקואורדינטה שלה גדולה ממספר הביטים הנדרש (256 במקרה של bitcoin) ולכם פעולת mod מגבילה את הגרף למספר הביטים הנדרש.
- הביטים מיוצגים בבינארי ולשם נוחות נרצה לייצג מספרים שלמים (יותר נוח להשתמש בקריפטוגרפיה). החלוקה עצמה והשארית הם מספרים שלמים.
- ה mod גורם לכך שקשה מאוד לשחזר את y בפונקציה. ההצפנה חזקה וקשה לגלות את ה private key ועל מנת למצוא אותו נצטרך לבצע brute force רציני שאינו בר ביצוע.

2. משתמש זדוני מעוניין לשלם פעמיים עם אותו Ether אחד שיש לו בארנק (double spending) . הוא מבצע שתי טרנזקציות משני מחשבים שונים (המחוברים ל peers שונים) שכל אחת מהן מעבירה Ether אחד לארנק אחר. הסבר מדוע בסופו של דבר הפעולה של המשתמש הזדוני לא תצלח והוא לא יוכל לשלם פעמיים עם אותו מטבע שיש לו.

תשובה:

אנו זקוקים לדרך בה מקבל התשלום יוכל לדעת שהבעלים הקודמים לא חתם על אף טרנזקציה מוקדמת יותר. נקבע טרנזקציה המוקדמת ביותר היא הקובעת ועל מנת להשיג זאת ללא צד נאמן (בנקים וכדומה), חובה להכריז על כל הטרנזקציות באופן פומבי (peer to peer network) ונחוצה מערכת להסכמה בין משתתפים לגבי **היסטוריה יחידה** של הסדר בו הטרנזקציות התקבלו. המערכת הזאת היא למעשה ה timestamp server (הבלוקצ׳יין).

בהנחה ומשתמש זדוני בזבז פעמיים את אותה הטרנזקציה, נניח Owner 2 העביר את אותה הטרנזקציה ל Owner 3 ול Owner 4 (אחת מהן היא חוקית, שתיהן ביחד לא חוקיות) אז בפעם הראשונה ש Owner 2 יעביר ל Owner 3 הטרנזקציה תהיה חוקית ובפעם השנייה ש Owner 2 יעביר את אותה הטרנזקציה (הכסף שהוא קיבל) ל Owner 4 אז כל האחרים ידחו אותה כי היא לא חוקית מפני שכל בלוק מכיל מספר בלוק וטרנזקציות אשר **לא ניתן לשנות** (ה-data בבלוקצ׳יין הוא immutable) ולכן ניתן לראות שאותה הטרנזקציה ״הזדונית״ בוזבזה כבר באחד הבלוקים.

3. תאר במילים מדוע כמעט בלתי אפשרי לשנות טרנזקציה שכבר נחתמה בבלוק קיים. מהם המנגנונים המונעים זאת? אילו פעולות על התוקף לבצע בכדי לשנות טרנזקציית ביטקויין אחת בבלוק הלפני אחרון?

תשובה:

ראשית, בלוקים חדשים מאוחסנים תמיד באופן ליניארי וכרונולוגי. כלומר, הם תמיד מתווספים לסוף הבלוקצ׳יין ולכל בלוק Timestamp משלו. קשה עד בלתי אפשרי לשנות את תוכן הבלוק (במקרה שלנו טרנזקציה) והסיבה לכך היא שכל בלוק מכיל Hash משלו, יחד עם Hash של הבלוק שלפניו, כמו גם חותמת זמן. אם המידע בבלוק נערך בצורה כלשהי, קוד ה-Hash ישתנה גם כן.

במידה והתוקף ירצה לשנות טרנזקציה בבלוק הלפני האחרון, אז הטרנזקציה תשנה את ה Hash של כל Markel Tree מה שיוביל לשינוי ה Hash של הבלוק הלפני האחרון. התוקף יצטרך לדאוג לכך שה Hash שרשום בתוך הבלוק האחרון (Prev_Hash) יהיה זהה ל Hash של הבלוק לפני האחרון וזה קשה מאוד על בלתי אפשרי לביצוע. אם הוא לא יצליח, ה Hash של הבלוק ישתנה ואם הוא משתנה אז הוא מתנתק מהשרשרת ומה שמתנתק מהשרשרת זורקים אותו כי הוא משהו שהוא לא חוקי.

הפעולות אותן יצטרך לבצע:

1) שינוי ה Hashים ב Markel Tree (הופך את זה לעוד יותר מסובך).
2) שינוי ה Hash של הבלוק לפני האחרון שיהיה זהה ל Hash שרשום בתוך הבלוק האחרון.

המנגנונים המונעים זאת:

1) מבנה ה Markel Tree (שמירת ה Hashים בצמתים של העץ)
2) מבנה ה Header של כל בלוק שמכיל את ה Hash של הבלוק הבא.
3) הרשת תפיל בלוק לא חוקי בגלל חוקי הקונזנזוס. לא משתלם לתוקף לפעול נגד החוקים. עדיף שיהיה ישר ויקבל את שכרו.

4. הסעיפים הבאים מתייחסים ל GAS באטריום
   a. מה המטרה של Gas באטריום?
   b. מה יקרה עם הגז יהיה בחינם?
   c. מדוע כמות ה GAS היא קבועה עבור פעולות אבל מחיר הגז משתנה?

תשובה:

a. Gas מתייחס ליחידה המודדת את כמות המאמץ החישובי הנדרש לביצוע פעולות ספציפיות ברשת Ethereum. מכיוון שכל עסקת Ethereum דורשת ביצוע משאבים חישוביים, כל עסקה דורשת עמלה. המטרה של Gas:

1. לתגמל את ה Miner שמריץ את ה Smart Contract על הבלוק שלו על המשאבים החישוביים וכוח האיבוד שאפליקציית ה Smart Contract דורשת.
2. יכול להיות שישנם פעולות ב Smart Contract שיתקעו את ה Miner (לולאה אינסופית, כתיבה כבדה לזיכרון וכו'...) במקרה כזה ה Miner עדיין צריך לקבל תגמול כי ההרצה הסתיימה אך הוא בכל זאת הפעיל מאמצים חישוביים.
3. לכתוב Smart Contracts יעילים על מנת לא לשלם הרבה Gas.

b. במידה וה-Gas יהיה חינמי לא תהיה ל Minerים מוטיבציה להוסיף Smart Contracts לבלוק שלהם שכן הם דורשים כוח איבוד ומאמצים חישוביים רבים (במיוחד בתוכניות לא יעילות ולא תקינות) המתבטאים בין היתר בתשלום חשמל מצד ה Miner לדוגמה. אף Miner לא יסכים לשלם בלי תמורה וכתוצאה מכך אף Miner לא יסכים להכניס Smart Contracts לבלוקים (זה מה שמייחד את ה Second Generation).

c. Gas הוא **קבוע**. אין לבעל החשבון שיצר את הטרנזקציה שליטה על כמה Gas הטרנזקציה דורשת. ה Miner יחשב תוך כדי הרצה את ה Gas שהוא השתמש. מה שכן, בעל החשבון יכול לקבוע את המחיר שהוא מוכן לשלם לכן המחיר **משתנה**. התגמול שה Miner יקבל שווה למכפלה של ה Gas ששומש (קבוע) כפול המחיר שבעל החשבון היה מוכן לשלם (משתנה).

5. מתכנת על רשת איטריום התקין ארנק Metamask במחשב אישי שלו והעביר 2 ETH לארנק שלו. אחרי זמן מה התקלקל המחשב שלו והמפתח הבין שלא שמר את ה 12 המילים (mnemonic words) שהארנק רשם לו כשהוא יצר את הארנק (וגם לא גיבה את המחשב)
   a. הסבר מה יקרה עכשיו למטבעות בארנק שלו ומדוע.
   b. מתכנת זה כתב חוזה חכם ועשה לו deploy באמצעות הארנק . מה יקרה לחוזה החכם אחרי שאבד הארנק ו ה mnemonic words?

תשובה:

a. המטבעות יישארו בארנק אך לבעל החשבון לא תהיה גישה לחשבון. זה קשה אם היינו צריכים לקרוא, להעתיק, או להקליד מחרוזת של 256 (המפתח הפרטי) עבור כל טרנזקציה. כדי להפוך את הדברים לקלים ובטוחים יותר עבור כל המשתמשים, מערכת סטנדרטית פותחה מתוך מחשבה על אבטחה, המכונה BIP39, כדי לספק לנו בנוחות סט של מילים, הנקראות mnemonic אשר למעשה משמש אותנו כמפתח פרטי. במידה והמתכנת איבד את המילים, הוא למעשה איבד את המפתח הפרטי ולכן לא יוכל לגשת יותר לחשבון.

b. ברגע שהתבצע deploy ל Smart Contract לא ניתן לשנות את ה Smart Contract והוא חתום בתוך הבלוקצ'יין לעולם. בעלי ארנקים אחרים יוכלו להפעיל את ה Smart Contract עם משתנים שונים.

6. השאלה מתייחסת לארנקים:
   a. מדוע משתמשים גם בארנק קר ולא ניתן להסתפק בארנק חם?
   b. מהו ה Hardware Wallet ? האם בטוח לחבר אותו למחשב המודבק בווירוס? נמק!
   c. עבור mnemonic words הבאים (לפי BIP39 בשפה האנגלית), חשב את ה seed שממנו מייצרים מפתח פרטי . תרשום הסבר כיצד חישבת את 12 הביט הראשונים

shrug thought track tissue affair race parent gravity correct aerobic sniff own

תשובה:

a. Hot Wallets הם ארנקים המחוברים לרשת ולכן פגיעים וחשופים יותר לתוקפים. ה- Cold Wallets מנותקים מרשת האינטרנט ולכן מאובטחים יותר. במידה ורמת האבטחה סופר חשובה ויש המון כסף בחשבון, כדי לשקול להשתמש ב Cold Wallets.

b. Hardware Wallet הוא סוג מיוחד של ארנק המאחסן את המפתחות הפרטיים של המשתמש בהתקן חומרה מאובטח. המפתח הפרטי לעולם לא עוזב את ההתקן והחתימה נעשית בתוכו ולכן גם אם המחשב מודבק בווירוס התוקף לא יוכל לבצע חתימות על טרנזקציות.

c.

חישוב:

1. ממירים את המילים למספרים המתאימים להם:
   1594 1799 1844 1814 34 1411 1281 816 388 33 1643 1264
2. ממירים את המספרים לבינארי (11 תווים):
   11000111010 11100000111 11100110100 11100010110 00000100010 10110000011
   10100000001 01100110000 00110000100 00000100001 11001101011 10011110000

12 הביטים הראשונים של הseed מורכבים מ11 הביטים הראשונים של המספר שמייצג את המילה הראשונה פלוס הגבוה ביותר של המספר שמייצג את המילה השניה.

7. כורה (miner) זדוני כתב כחלק מחוזה חכם פונקציה היכולה להיכנס ללולאה אינסופית אם הפרמטר המסופק לפונקציה הנו בעל ערך מאוד ספציפי. תאר דרך בה יוכל אותו כורה לבזבז משאבים של מתחריו מבלי לבזבז את המשאבים שלו.

תשובה:

הכורה הזדוני יכול ליצור לולאה אינסופית שהתנאי כניסה ללולאה היא:

If (wallet.address != me.wallet.address)

כאשר האובייקט me לצורך העניין מתאר את הכורה הזדוני. כל מי שיבצע טרנזקציה יכנס ללולאה האינסופית. הלולאה צורכת המון משאבים ולכן לכורה הזדוני יהיה יתרון על פני מתחריו מפני שהוא מבזבז להם את המשאבים דרך הלולאה.

8. השאלה מתייחסת ל forks
   a. הסבר מהם hard fork ו soft fork ?
   b. מה יקרה לבלוקצ'יין ולמשתמשים בשני סוגי ה fork אם חלק מהמשתמשים לא יממשו את השינויים?
   c. הסבר כיצד הרשת פותרת את הבעיה כאשר שני miners שונים חותמים על בלוק חדש בערך באותו הזמן.

תשובה:

a. Soft fork - הוספת חוק חדש לטובת שינוי הפרוטוקול, אך שינוי זה הינו בעל תאימות לאחור. כלומר, הצמתים בבלוקצ'יין לא חייבים לעדכן את הגרסה אותה הם מריצים, אך אם ברצונם להשתמש בחוק או בפיצ'ר החדש, הם יהיו מוכרחים לעשות זאת.

Hard fork - הסרת חוקים לטובת שינוי הפרוטוקול, וכתוצאה מכך השינוי אינו בעל תאימות לאחור. כלומר, אם כיום מגבלת תקרת גודל בלוק בביטקויין היא 1 מ"ב – שינוי פרוטוקול בהארד פורק יהיה לשנות את מגבלת תקרת גודל הבלוק ל-2 מ"ב. בלוקים של מעל 2 מ"ב גדולים מבלוק של 1 מ"ב (כמובן) ולכן אינם בעלי תאימות לאחור. הארד פורק מצריך שכול הצמתים המלאים ברשת יאמצו את החוקים החדשים. אם צמתים מסוימים מחליטים שלא לאמץ את השינוי, יש לכך השלכות על שרשרת הבלוקים.

b. Soft fork - לפחות 51% מהכורים מוכרחים לעדכן גרסה על מנת שה-Soft fork יתקיים, אחרת הוא יהפוך לשרשרת המורכבת מבלוקים יתומים. צמתים הבוחרים שלא לעדכן גרסה על אף שהרשת אימצה אותה, יהיו פחות בטוחים שכן הטרנזקציות העונות לחוקי הקונצנזוס החדשים ייראו כלא תקינות בעיניהם.

Hard fork - אם חלק מהצמתים וכוחות הכרייה אינם מסכימים אם שינויי החוקים, הם יכולים לבחור להישאר עם השרשרת הישנה. בסיטואציה כזאת, שרשרת הבלוקים מתפצלת לשני ענפים שונים בעלי אותה היסטוריה. אך זה לא נגמר בזה, שכן עם היווצרות הענף החדש – נוצר גם מטבע חדש. למעשה, הפיצול בשרשרת יוצר ענף בלוקצ'יין חדש, וענף זה "מצריך" מטבע משל עצמו.

c. אם שני צמתים משדרים גרסאות שונות של הבלוק הבא בו-זמנית (Temporary Fork), חלק מהצמתים יקבלו את האחד או השני תחילה. במקרה זה, הם יעבדו על הראשון שהם קיבלו, אבל ישמרו את הענף השני למקרה שהוא יהפוך לארוך יותר. התיקו יישבר כשהבלוק הבא ייחתם ואחד מהענפים יהפוך לארוך יותר וכל הרשת תתיישר לפיו.

9. ברשת האטריום יש שני סוגי clients – Full Node and Remote Client
   a. הסבר מה ההבדל ביניהם ומדוע יש מקום לשניהם
   b. כיצד ניתן לחבר את המחשב הביתי לרשת האיטריום?
   c. מהי רשת Peer to Peer ? אילו סוגי הודעות עיקריות עוברות ברשת האטריום ?

תשובה:

a. צומת מלא הוא צומת האוכף באופן מלא את כל כללי הבלוקצ'יין, בעוד שלקוח קל (המכונה גם Remote Client) מתייחס להעתק של צומת מלא מהימן של הבלוקצ'יין. המשמעות היא שתוכל לבצע עסקאות על הבלוקצ'יין מבלי להוריד עותק שלם של הבלוקצ'יין. חשוב לציין שמדובר בקובץ גדול מכיוון שיש להוריד את כל ההיסטוריה של הבלוקצ'יין. צומת מלא אינו אופציה לכולם מכיוון שיש הרבה אנשים שאין להם זיכרון או עיבוד מסוג זה. במקום זאת אנשים מעדיפים להשתמש ב Remote Client, מכיוון שיש להם את האפשרות לאמת טרנזקציות ללא צורך להוריד את כל הבלוקצ'יין.

b.

- להתקין על המחשב Client של איטריום. ישנם מגוון Clientים.
- להתחבר דרך HTTP provider (כגון infura.io) שמשמש כ full node ומספק API שדרכו ניתן לדבר עם רשת האיטריום.

c. peer to peer היא רשת תקשורת מבוזרת, ללא צורך בתיווך או תלות בגורם מרכזי כלשהו בה כל אחד מהקצוות מתפקד הן כלקוח והן כשרת, וכל אחד מהקצוות מסוגל ליזום או לסיים התקשרות ולספק או לדרוש שירותים.

סוגי ההודעות העיקריות העוברות ברשת: העברות של מטבע וירטואלי או הפעלת חוזים (Smart Contracts).

10. ברשת האטריום יש משמעות ל"חשבון" ואילו בביטקוין אין (באטריום מעבירים מטבעות לחשבון ובעל החשבון יכול להעביר בטרנזקציה כל חלק מהחשבון בניגוד לביטקויין שחייב להשתמש בכל ה output).
תאר מה שונה באימפלמנטציה של איטריום ביחס לביטקויין המאפשרת החזקת "חשבון" . מדוע לדעתך החליטו מפתחי הביטקויין (למעשה סטושי נאקאמוטו) לא לאפשר זאת?

תשובה:

השוני במימוש של איטריום ביחס לביטקויין שמאפשר החזקת "חשבון" הוא שאיטריום גם שומר מצב ולכן חוץ מהבלוקצ׳יין ישנם עוד מבני נתונים שמטרתם לשמור את ה state. ה State trie הוא מבנה נתונים בצורת עץ שהעלים שלו הם בעצם ה Account State שמורכב בין היתר מ nonce ו balance.

לדעתי המפתחים/מפתחת/מפתח לא אפשרו זאת מכיוון שמנגנון של שמירת מצב הוא מורכב יותר, מצריך עבודה עם הרבה מבני נתונים לשמירת המידע ומצריך יותר עבודות פיתוח מאשר המנגנון הקיים.

11. השאלה מתייחסת לטרנזקציות ביטקויין
   a. הסבר מהו ה input ו output בטרנזקציות של ביטקויין.
   b. מדוע input גדול במקצת מ output?
   c. הסבר במילים כיצד רק בעל הארנק שאליו הועברו מטבעות יכול להשתמש ב output של טרנזקציה.

תשובה:

a. ישנם שלושה אלמנטים המעורבים בעסקת בביטקויין: קלט עסקה, פלט העסקה וסכום. קלט העסקה הוא כתובת הביטקויין ממנה נשלח הכסף, ופלט העסקה היא כתובת בביטקויין אליה נשלח הכסף.

b. ה input גדול במקצת מה output מהסיבה שעבור כל טרנזקציה נהוג לשלם fee (תגמול) לכורה הבלוק. ה fee מחושב כך: input-output=fee. כלומר ההפרש הקטן בין הקלט לפלט זה התגמול.

c. מכיוון שאף אחד מלבד בעל הארנק שמחזיק במפתח הפרטי לא יוכל לחתום על output של טרנזקציה.

12. השאלה מתייחסת ליצירת מטבעות ביטקויין:
   a. תאר כיצד נוצרים מטבעות ביטקויין חדשים
   b. איך זה שקצב יצירת מטבעות ביטקוין חדשים הוא ידוע ומדוע כורים (miners) זדוניים לא יכולים לייצר יותר מטבעות מהמוגדר?
   c. הסבר במילים מדוע צריכים מספר יחידות חישוב/מחשבים גדול מאוד בשביל להצליח בתחרות הכרייה (Mining) ? כיצד מספר גדול של יחידות חישוב עוזר בכריה?

תשובה:

a.

1) הכורים (Miners) מאזינים לעסקאות המשודרות ב peet to peer ברשת הביטקוין.

2) אוספים את העסקאות לבלוק עד שמגיעים לגודל של עסקאות שהוגדר מראש.

3) מבצעים עבודה חישובית במטרה למצוא מספר מיוחד שגורם לכך שה Hash של הבלוק יהיה קטן יותר מה Difficulty (מספר שמשתנה עם הזמן ותלוי במהירות של כריית הבלוקים)

4) ברגע שהם מוצאים את המספר הזה, הם משדרים את הבלוק שהם מצאו.

5) כדי לתגמל את הכורים על העבודה החישובית, כשהם מרכיבים בלוק, הרשת מאפשרת להם לכלול עסקה מיוחדת בחלק העליון של הבלוק אשר בה הם מקבלים כמות מסוימת של ביטקויין. העסקה הזאת נקראת "תגמול הבלוק"

6) העסקה הזאת מיוחדת מפני שהיא לא נחתמה על ידי אף אחד וזה גם אומר שכמות הביטקוין הכללית גדלה עם התוספת של התגמול על הבלוק הנוכחי.

7) ולכן כל כריה של בלוק חוקי יוצרת עוד מטבעות ביטקויין.

b. קצב יצירת מטבעות ביטקויין חדשים הוא ידוע מכיוון שיוצר (יוצרי, יוצרת) הביטקויין הגבילו מלחתחילה את מספר המטבעות הסופי (אלה אם כן יבצעו hard fork) שעומד על 21 מיליון מטבעות. כמו כן, קיים חוק הנקרא Halving שאומר שכל כמה שנים בממוצע יחתך הגמול לכורים בחצי.

כורים זדוניים לא יכולים לייצר יותר מטבעות מהמותר מהסיבה שכל המערכת הזאת עובדת על אמון. במידה וכורה יעביר לעצמו יותר מהמותר, אף אחד ברשת לא יתייחס לבלוק הזה ויפילו אותו. עדיף לכורה לשחק לפי החוקים ולקבל תגמול מסויים על הכריה של הבלוק מאשר לבזבז משאבים לכריה של בלוק ובסופו של דבר אף אחד לא יתייחס אליו.

c. לאחר בניית הבלוק מתחילה תחרות חישובית מורכבת אשר מטרתה פיצוח הבלוק והפיכת המידע המאוחסן בו לשמיש, פיצוח החידה היא למעשה הוכחת העבודה (Proof of Work). קושי החידה נוצר ע״י פונקציית SHA-256, קושי זה ניתן להתאמה ביחס לכוח החישוב הזמין ברשת המטבע בכדי לשמור על זמן פיצוח ממוצע אותו קבע סטושי. נניח ונרצה מספר בעל קושי המתחיל ב 10 אפסים בכדי לפתור זאת כך נצטרך 1.1 טריליון ניסיונות בקרוב! **המצריכים כוח חישוב אדיר.**

מספר גדול יותר של יחי׳ חישוב מגדיל את הסיכויים לפיצוח החידה מפני שמדובר בסופו של דבר בחישובים מתמטיים שהמחשב מבצע. ככל שיהיו יותר יחי׳ חישוב, כך החישוב יתבצע מהר יותר.

13. השאלה מתייחסת לרשת Ethereum:

a. מי מריץ את החוזים החכמים ברשת ה Ethereum ומי שומר את הערכים של משתנים בבלוקצ'יין?
b. הסבר מהי בעיית הסקלבליות (scalability) ברשת האיטריום וציין לפחות סיבה אחת לכך
c. ציין שני כיוונים אפשריים לפתרון הבעיה. הסבר במילים כיצד הם יכולים לעזור בפתרון הבעיה .

תשובה:

a. כורה הבלוק מריץ את החוזה החכם על מנת להכניס לצרף אותו לבלוק ומקבל על זה תגמול. כל מי שמעוניין להריץ את ה Smart Contract יכול לעשות זאת, זה עובד למעשה כמו טרנזקציה, הוא מפעיל פונקציות של ה Smart Contract עם פרמטרים מסוימים והערכים של המשתנים הגלובאליים נשמרים בתוך שטח אחסון וזיכרון של ה Smart Contract (state) והוא נמצא ב Merkle Tree בתוך הבלוקצ׳יין.

b. טרילמת המדרגיות (scalability), שהוזכרה לראשונה ע״י ויטאליק בוטרין, היא מושג בבלוקצ׳יין בנוגע ליכולתו להתמודד עם מדרגיות, ביזור וביטחון, מבלי לפגוע באף אחד מהם. הטרילמה טוענת שכמעט בלתי אפשרי להשיג את כל שלושת המאפיינים במערכת בלוקצ׳יין.

סיבה אחת לכך היא העליה בפופולריות השרת ובעקבותיה עליה במספר הטרנזקציות היומי. את כל הטרנזקציות האלו ה Minerים צריכים להכניס לבלוקים, למצוא Hash כדי לכרות בלוק, להריץ את ה Smart Contracts ולבדוק שהכל תקין. כולם במקביל עושים כמות עבודה עצומה והעומס גדל ולכן הגבילו את מספר הטרנזקציות שניתן לעשות לשניה.

c. כיוונים לפתרון הבעיה:

- מעבר מ PoW ל PoS - ב PoW כולם במקביל עושים עבודה קשה הדורשת המון משאבים וכוח חישוב. ב PoS קובעים כי אדם יכול לכרות בלוקים בהתאם לכמה מטבעות שהוא מחזיק. פירוש הדבר שככל שיש לו יותר איטרים, כך יש לו יותר כוח כרייה. בכל פעם נבחר אדם אחד בעל כוח כלכלי והוא נבחר לחתום על הבלוק בלי לעשות את העבודה הקשה. כך נחסכת העבודה הקשה (משאבים) ו״המלחמה״ על כריית בלוקים.
- Sharding – לכל Shard יש את ה state שלו. ואז אם חשבון שייך ל Shard מסוים אז הוא יחיה רק ב Shard הזה.

בוחן בית בקורס מבוא לטכנולוגיית בלוקצ'יין
ולדיס מרקין 317273969

14. השאלה מתייחסת לחוזה חכם הכתוב בשפת Solidity

a. השורה הראשונה בחוזה חכם מציינת "pragma solidity 0.x.y;" (X ו Y לדוגמה 5 ו 2) . הסבר מדוע מוסיפים שורה זו, כאשר בשפות אחרות (כמו ג'אווה או פייטון) לא רושמים שורה מסוג זה.

b. מהו modifier ומדוע כדאי להשתמש בו (כשיש הגיון) בכתיבת חוזים חכמים?

c. מה מיוחד ב state variables ומדוע משתמשים בהם בכתיבת חוזים חכמים? הסבר!

d. מהו המנגנון שמונע מחוזה חכם להמשיך לרוץ בלולאה אינסופית (בהנחה שחוזה חכם נכתב בצורה כזאת שמאפשר לולאות אינסופיות)?

תשובה:

a. השורה pragma solidity מתייחסת לגרסה של הקומפיילר. הסינטקס משתנה לעיתים מגרסה לגרסה ומשתמש שרוצה להפעיל את החוזה החכם מעוניין לדעת באיזה גרסה הוא נכתב מפני שיכול להיווצר מצב שאותו קוד נותן פלט שונה בגרסאות שונות וכשמדובר בכסף העניין קריטי. בשפות כמו Python, Java וכו' אין סיבה לכתוב את השורה הזאת מכיוון שרוב האפליקציות הן לא להעברת כספים.

b. ה modifier בודק תנאים מסוימים שכותב החוזה החכם מעוניין לדעת לפני הרצה של פונקציה מסוימת. מכיוון שיש מחיר לכל Opcode, כלומר עלות על כל הרצה, כותב החוזה החכם מעוניין לחסוך ולדעת כמה שיותר מוקדם אם לצורך הדוגמה יש לו תנאי מסוים שלא עובד. ה-modifier נותן את האפשרות הזאת.

c. ה State Variables הם משתנים שלא נעלמים בהרצה והם מתעדכנים בכל הרצה של ה Smart Contract. משתנים אלה שמורים ב State trie.

d. מנגנון ה Gas. אם נגמר ה Gas לתדלוק הרצת החוזה החכם, ההרצה תיפסק.