מבוא למטבעות קריפטוגרפים – תרגיל בית 1

תאריך הגשה: יום חמישי, 5 דצמבר, 23:59

מטרת התרגיל:

בתרגיל זה נבנה מערכת תשלומים מאובטחת. נתחיל עם ישות מרכזית אחד ("הבנק"). משתמשים פרטיים ייצוגו ע"י "ארנקים" שיש להם מפתח פרטי והם יוצרים טרנזקציות. הבנק יפעל לפי מודל UTxO והוא יאשר את הטרנזקציות בבלוקים.

בתרגיל המשך, אנחנו נחליף את הבנק במערכת מבוזרת בהתאם לקונצנזוס של נקמוטו.

סקירה כללית:

יש שני סוגים של ישויות במערכת שלנו: בנק וארנקים. הארנקים מייצגים משתמשי קצה שרוצים לשלוח או לקבל כסף, והבנק מייצג מסד נתונים מרכזי שמחזיק רשומות של הטרנזקציות שהתבצעו לאורך ההיסטוריה והוא מאשר טרנזקציות חדשות.

המערכת תעבוד בצורה דיסקרטית. נקרא לכל צעד "יום". בכל סוף יום, הבנק מפרסם את ההתחייבות שלו לבלוק חדש בתצורה של hash של אותו בלוק. הבלוק מייצג את החלק האחרון העדכני ביותר של המערכת, כאשר כל הבלוקים מייצרים את הבלוקצ'יין. ההתחייבות היא על המידע הבא:

1. כל הטרנזקציות שאושרו באותו היום.
2. ההתחייבות הקודמת, שמייצגת את המצב הקודם של המערכת.

הארנקים יכולים לבצע טרנזקציות ככל העולה על רוחם, והם יכולים לבקש מהבנק בלוקים שפורסמו מאז הפעם האחרונה שהם היו מחוברים למערכת. לצורך פשטות, כל הטרנזקציות מעבירות את אותו הסכום – מטבע יחיד, מכתובת מקור אחת לכתובת יעד אחת. כל הכתובות הן למעשה מפתח פומבי. ארנק יכול להעביר מטבע שהוא קיבל בטרנזקציה קודמת שאושרה ע"י הבנק.

על מנת ליצור כסף, הבנק (ורק הבנק) יכול ליצור טרנזקציה שאין לה כתובת מקור. הבנק לא צריך לחתום על הטרנזקציה, אבל כדי להבדיל בין טרנזקציות שיוצרות כסף, אוסף רנדומלי של בייטים ממוקמים בשדה החתימה של טרנזקציה מסוג יצירת כסף; תראו את קובץ הטמפלייט לפרטים נוספים.

על מנת לייצר בייטים רנדומליים השתמשו בספרייה secrets. הפונקציה secrets.token_bytes(n) יכולה ליצור n בייטים רנדומליים בצורה קריפטוגרפית בטוחה. הספרייה secrets היא ספרייה סטנדרטית כחלק מ-python ואין צורך להתקין אותה.

בתרגיל זה, הניחו כי הבנק לא משנה בלוקים קודמים בבלוקצ'יין: כל בלוק חדש משורשר לשרשרת הנוכחית.

APIs:

קבצי ה-Python שסופקו מכילות פונקציות שמגדירות את ה-APIs של בלוק, טרנזקציה, ארנק ובנק.

ההתנהגות של אלמנטים אלו מוגדרים בדוקומנטציה. תקראו אותה בזהירות!

בנוסף, יש אנוטציות לקוד עם טייפים כדי שתוכלו להבין מה הסוג של כל משתנה. אני ממליץ בחום להשתמש בספרייה mypy כדי לבצע type cheking לקוד שלכם שאתם כותבים. אם אתם לא מכירים את Python's type system, ניתן למצוא פה הסבר: https://mypy.readthedocs.io/en/stable/getting_started.html

בתרגיל מסופק ה-APIs של ארבע מחלקות: ארנק, בנק, טרנזקציה ובלוק. המשימה שלכם היא לממש את ה-APIs האלו בגירסת Python 3.8 או גבוהה יותר.

חתימות דיגיטליות: מומלץ להשתמש בספרייה pyca/cryptography (https://cryptography.io/en/latest/) על מנת להתמודד עם חתימות ווידוא של הודעות חתומות. את הספרייה ניתן להתקין בקלות בעזרת pip. על מנת לעזור לכם להשתמש בספרייה זו בקלות, אתם תראו שלוש פונקציות שמספקות את הצרכים הבסיסיים עבור חתימות דיגיטליות (חתימה, וידוא ויצירת מפתחות) בקובץ utils.py:

- sign(message: bytes, private_key: PrivateKey) -> Signature
- verify(message: bytes, sig: Signature, pub_key: PublicKey) -> bool
- gen_keys() -> Tuple[PrivateKey, PublicKey]

הטייפים PrivateKey, PublicKey ו-Signature הם טייפים מסוג bytes שגם כן מוגדרים היטב בקובץ utils.py.

בנוסף לחתימות, תצטרכו לייצר hashes של הבלוקים ושל הטרנזקציות (transaction id – TxID יהיה למעשה hash על התוכן של הטרנזקציה). על מנת לעשות זאת, השתמשו בפונקציה hashlib.sha256().digest().

לפרטים נוספים: https://docs.python.org/3/library/hashlib.html.

מודולריות וקוד נקי: חשוב מאוד לוודא שהקוד שלכם מודולרי וכתוב היטב (אבל אל תנסו להיות כמה שיותר כלליים). זה יחסוך לכם זמן בתרגיל הבא (תוכלו להשתמש בקוד מהתרגיל הנוכחי בתור נקודת התחלה אם תרצו).

הקוד כרגע מחולק לכמה קבצים וקיים קובץ נוסף בשם "__init__.py" על מנת לחבר את כל הקבצים האלו לספרייה אחת.

Tests :

בתרגיל סופק לכם גם טסטים מועילים (ב-pytest) שבעזרתם תוכלו לבדוק את הקוד שלכם ולהבין איך למעשה האובייקטים השונים עובדים ביחד. ישנם טסטים נוספים שלא שותפו ויהיה בהם שימוש לצורך בדיקת תרגיל הבית. אני ממליץ לכם לכתוב טסטים במקביל למימוש המחלקות.
אם התקנתם את החבילה pytest (בעזרת pip), אז כדי להריץ את הטסטים יש להריץ את הפקודה "python -m pytest" בתיקייה הראשית של תרגיל הבית (בתיקייה זו יהיו תתי תיקיות בשם ex1 ו-tests).
הטסטים כתובים בתור בדיקות עקביות, במילים אחרות "מתקפות" על הבטיחות של המערכת שלכם. לכן, חשוב מאוד לוודא שכיסיתם את כל המקרים שתוקף עלול לנצל. אל תשנו את השמות של הפונקציות ב-APIs מכיוון וזה עלול לשבור טסטים.

הבהרה : בעולם האמיתי, הבנק והארנקים היו רצים על תהליכים שונים במחשבים שונים ומחוברים על גבי האינטרנט. במערכת שלנו, הם מיוצגים על ידי אובייקטים באותה מכונה. ניתן להניח שאין לתוקף גישה לשדות פרטיים ופונקציות לא פומביות של הבנק והארנקים, אבל ניתן לשנות את המידע שמועבר בין אובייקטים אלו. לדוגמה, טרנזקציה היא למעשה רק אוסף של בייטים שניתן ליצור ולשלוח לבנק ע"י כל משתמש, ביניהם משתמשים "רעים". בנוסף, מתקפה נוספת יכולה להיות העתקה של חתימה בטרנזקציה אחת אל הודעה אחרת, ועוד.
ניתן להניח בתרגיל הבית כי הבנק הוא ישות שניתן לסמוך עליה. הבנק תמיד ייצור רק בלוקים תקינים והוא לא יבצע שום מניפולציה על הבלוקצ'יין בצורה כלשהי. בנוסף, ניתן להניח שמידע שנשלח מהבנק אל הארנקים הוא מידע שלא עבר שינוי ע"י תוקף. למעשה, בבלוקים שהבנק שולח אל הארנקים אין חתימה, וניתן להניח כי המקור והתוכן אכן נכונים. המניפולציות היחידות שיכולות לקרות הן על הודעות שנשלחות ע"י הארנקים.

לכן, התוקף יכול ליצור טרנזקציות או לבצע מניפולציות על הודעות שמגיעות מארנקים אחרים. שימו לב, ה-hash של בלוק או טרנזקציה מחושב על התוכן.

"אוכל למחשבה" :

חשבו על הנקודות הבאות (אין צורך לכתוב תשובות לשאלות אלו, או לממש זאת בקוד שלכם) :

- נניח כי כל הארנקים מוודאים את הבלוקצ'יין, האם הבנק יכול להעביר כסף של ארנק מסוים בלי ההסכמה שלו?
- האם הבנק יכול לבצע revert לטרנזקציה שכבר הוכנסה לבלוקצ'יין ופובלשה?
- האם הארנקים יכולים שלא להסכים על התוכן של הבלוקים?

הגשה:

יש להגיש את הקבצים בתוך zip שנקרא ex1.zip. קובץ ה-zip יכיל את קבצי ה-python של תת התיקייה ex1 **(אל תוסיפו את קבצי ה-tests)**. בנוסף, עליכם להוסיף קובץ README (לא README.txt). קובץ ה-README צריך להכיל את השמות שלכם, תעודות זהות והסבר קצר על התרגיל.

על מנת להריץ את הפתרון שלכם, אני אייבא אותו מ-ex1 (כמו שה-tests בתיקיית ה-tests עושים).

שימו לב שאתם מגדירים רק classes רלוונטיים (כמו בקבצי הטמפלייט) ושלא רץ קוד נוסף.

בהצלחה!