מבוא למטבעות קריפטוגרפים – תרגיל בית 4

תאריך הגשה: יום רביעי, 12 פברואר, 23:59

Bidirectional Payment Channels:

למדנו על Lightning Networ והמימוש של ערוצי תשלום. אלה נוצרים כאשר שני משתתפים מעבירים הודעות חתומות בהם מבטיחים אחד לשני חלוקה של הכספים הקיימים על הערוץ.
אם משתתף סוגר את הערוץ בהתאם למצב ישן (חלוקה ישנה של הכספים), אז המשתתף השני יכול "לערער" בעזרת מצב חתום חדש (עם serial number גבוה יותר).

בתרגיל זה אתם תממשו מערכת תשלומים 2-directional בין אליס לבוב בעזרת solidity וקוד Python (בעזרת הספרייה web3.py).
אתם תקבלו קבצי בסיס עם תיאור של ה-API שתצטרכו לממש על מנת לפתור את התרגיל, ובנוסף גם כמה טסטים פשוטים כדי להבין כיצד המערכת עובדת.
כמה נקודות נוספות על המימוש:

- הקשר שבין ערוץ תשלומים לחוזה חכם ב-solidity הוא אחד לאחד. כלומר, לכל ערוץ נעשה deployment של החוזה החכם. לדוגמה, לאליס ובוב יהיה ערוץ שיתורגם לחוזה חכם X ואליס וצ'רלי יהיה ערוץ שיתורגם לחוזה חכם Y. למידע נוסף ראו את Channel.sol ו-ChannelInterface.sol עבור ה-interface של החוזה החכם.
- קודקודים יהיו מיוצגים בעזרת מחלקה ב-Python. הם בנוסף גם יבצעו deploy לקוד solidity המקומפל בתוך Python. למידע נוסף ראו את node.py ו-lightning_node.py.
- הקבצים אינם סופיים, אתם תצטרכו להוסיף פונקציות, משתנים ואובייקטים.
- אנחנו נשתמש בספרייה web3.py ו-hardhat בתור בלוקצ'יין לוקאלי של ethereum על מנת להריץ ולבדוק את הקוד.
- אליס ובוב לא באמת רצים בתהליכים שונים. הם יהיו אובייקטים שונים וכדי לשלוח ביניהם הודעות נשתמש במחלקה Network (ראו network.py).
- אתם חייבים לשים לב שהחוזה החכם שלכם והקוד שכתבתם אכן בטוחים כך שלא אליס ולא בוב יוכלו לגנוב כספים מהצד השני או שהם ינעלו כספים לאחר גם אם הוא זדוני.
  כרגיל, אנחנו נסתכל על מתקפות שהן יעבדו כאילו האובייקטים לא חולקים את אותה סביבת Python (כלומר לא צריך לחשוש שאובייקט יכול לגשת למשתנים או פונקציות של אובייקט אחר).
  במקום זה התמקדו בהתנהגות הקודקוד למתקפות בהן הוא שולח הודעות תקולות, מסיים את התקשורת מוקדם, יוצר קשר עם הבלוקצ'יין כדי לנסות לגנוב כספים, ועוד.
  בנוסף, אובייקט הרשת עלול להיות מידי פעם תקול ולא יצליח להעביר הודעות, או שהמשתתפים האחרים ינסו לתקשר עם החוזה החכם בצורה online-ית.

הערות נוספות על דברים שאתם לא צריכים להתמודד איתם:

- אם אליס יוצרת את הערוץ בינה לבין בוב, בוב צריך לדעת איכשהו שהערות נוצר בעזרת חוזה חכם שהוא סומך עליו. באופן אידיאלי, הוא ינסה לאמת את הקוד של הכתובת של החוזה החכם שאליס נתנה לו. אתם לא צריכים לעשות זאת. (זו משימה קשה בפני עצמה)
- אתם לא צריכים לממש "סגירה משותפת", כלומר, סגירה של הערוץ ללא appeal period. אנחנו נניח תמיד שהערוץ נסגר ע"י משתתף אחד מבלי שהוא מודיע למשתתף השני.
- אתם לא נדרשים לנטר את הבלוקצ'יין כל הזמן על מנת לבדוק האם משתתף מסוים הוסיף משתתף אחר לערוץ, או שהערוץ נסגר על ידי המשתתף השני. קודקודים יקבלו הודעות מקודקודים אחרים על ערוץ חדש, והעברות כספים. עם זאת, על סגירה של ערוץ קודקוד לא יקבל הודעה (תראו את הטסטים שהתווספו לתרגיל).

web3.py:

מצורפים הלינקים הבאים בהם תוכלו למצוא מידע נוסף ואיך להתקין:

- web3.py
- hardhat

לאחר מכן, תיצרו תיקייה חדשה ובתוכה תאתחלו פרויקט hardhat בעזרת הפקודה

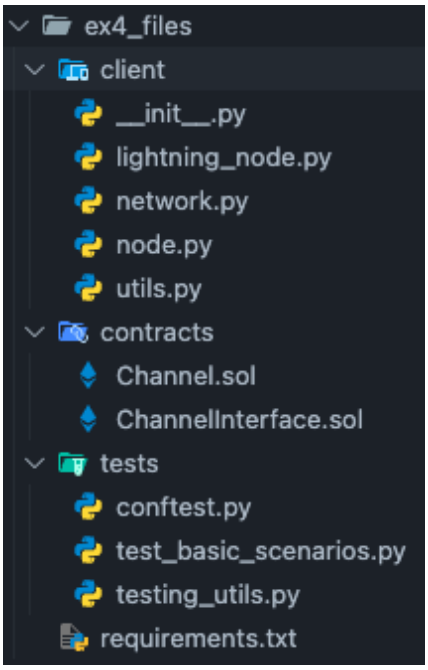
```
hardhat init
```

מתוך תיקיית הבסיס של הפרויקט, תוכלו להריץ פקודות כמו

```
hardhat node
```

רשימת הקבצים:

בתוך הקובץ ex4.zip תמצאו את הקבצים:



הקוד של החוזה החכם נמצא בתוך Channel.sol, והקודקודים נשלטים בעזרת הקוד בתוך lightning_node.py. שאר הקבצים הם קבצי עזר וטסטים. תיצרו לעצמכם סביבה וירטואלית והתקינו את הספריות הדרושות להריץ הקוד בעזרת הפקודה pip install -r requirements.txt.

כל הקוד שסופק לכם צריך להישאר תחת התיקיות בתוך התיקייה הראשית של הפרויקט. ברגע שהסביבה הוירטואלית הותקנה, אתם יכולים להריץ את הטסטים בעזרת הפקודה python -m pytest מהתיקייה הראשית של הפרויקט. הטסטים בצורה אוטומטית ירימו בלוקצ'יין לוקאלי בעזרת hardhat. אם ויש כישלון כלשהו, ראו את conftest.py וערכו את הפקודה שמריצה hardhat שתתאים להרצה אצלכם במחשב.

מומלץ להוסיף טסטים שלכם על מנת לכסות את כל ההתנהגויות הזדוניות השונות, ומקרי קצה.

הגשה:

יש להגיש את הקבצים בתוך zip שנקרא ex4.zip. קובץ ה-zip יכיל את הקבצים Channel.sol ו-lightning_node.py ביחד עם README. אל תשלחו קבצים אחרים!

הקוד שלכם ייבדק בצורה אוטומטית. אתם יכולים לשנות את הקבצים כל עוד אתם לא שוברים את ה-API שהוגדר (כלומר, אתם יכולים להוסיף עוד פונקציות, משתנים, מבנים ועוד).

שימו לב שרק שותף אחד צריך להגיש את התרגיל!

בהצלחה!