

RWAPSSF

Problem

1. ไม่มีใครอยากจะทำเลือกก่อน เพราะกลัวถูกอีกคนทำ front-running (การได้ประโยชน์จากการรู้ล่วงหน้าว่าคนอื่นจะเลือกอะไร)
2. ยากต่อการจะรู้ว่าเราใคร account ไหนเป็น idx ที่ 0 หรือ 1
3. เงินของ player 0 อาจถูกล็อกไว้ ถ้าไม่มี player 1 มาลงขั้นตอน
4. กรณีได้ player ทั้ง 2 แล้ว แต่มีเพียง player เดียวที่ลง choice มา แต่อีก player ไม่ยอมเรียก input function เพื่อส่ง choice มาให้ smart contract ได้ตัดสินใจแพ้ ชนะ เสมอ เช่นนี้ทำให้เงิน ETH ของทุกคนที่ลงขันมาถูกล็อกไว้โดยไม่มีใครถอนออกมาได้
5. ทำยังไงให้ contract นี้(มีการ transact กับมัน) ได้ในหลายๆ รอบโดยไม่ต้องมีการ deploy ใหม่เสมอในทุกๆ ครั้งที่ต้องการเล่น
6. เล่น

Solution

1. (Done) Use commitment to protect front-running
 - Player commit their choice by hash it by function `hashChoiceWithSalt(choice, salt)`
 - When both player commit their choice, they reveal their choice and salt and compare it with their commitment then get winner
2. (Done) Use account address instead idx 0 or 1
 - Use `msg.sender` to get player instead of using idx
 - So player not need to know their idx
3. (Done) Use timer to protect lock
 - If player 1 not play in time, player 0 can call `playerWithdraw` to get their money back
 - But if other player reveal their choice, player can't withdraw their money
4. (Done) Restart game when game finish
 - When game finish, reset all game state to initial state

Additional

1. (Done) Rock Water Air Paper Sponge Scissors Fire (7 elements)
 - Add more elements and change winner logic

Example

Player 1 win and player 2 lose

The screenshot displays the Remix IDE interface. On the left, the 'ACCOUNT' tab shows a list of addresses, with '0x787...cabaB' selected. Below it, the 'Transactions recorded' and 'Deployed Contracts' sections are visible. The 'playerCommitHashChoice' contract is highlighted, showing its 'hashChoice' value as '0x42c8a5d3347001c4e9442c8a4f50d134efc2a8cc0e39415177e09a6'. The 'playerJoin' button is highlighted in orange. The 'playerReveal' button is also visible, with a 'choice' field set to '1'. On the right, the 'RPS.sol' file is open, showing the game logic. The execution log on the right side of the IDE shows a sequence of transactions and calls, with handwritten red annotations 'P1' and 'P2' indicating the actions of Player 1 and Player 2 respectively. The log shows that Player 1 committed a hash choice, Player 2 revealed their choice, and Player 1's choice was correct, leading to a win for Player 1.

Draw

The screenshot displays the Remix IDE interface. On the left, the 'ACCOUNT' tab shows a list of addresses, with '0x787...cabaB' selected. Below it, the 'Transactions recorded' and 'Deployed Contracts' sections are visible. The 'playerCommitHashChoice' contract is highlighted, showing its 'hashChoice' value as '0x385642bdc03248d345a5d673ac0b61330db5310754d1ce085a64d714'. The 'playerJoin' button is highlighted in orange. The 'playerReveal' button is also visible, with a 'choice' field set to '0'. The 'playerWithdraw' button is also visible. On the right, the 'RPS.sol' file is open, showing the game logic. The execution log on the right side of the IDE shows a sequence of transactions and calls, with handwritten red annotations 'P1' and 'P2' indicating the actions of Player 1 and Player 2 respectively. The log shows that Player 1 committed a hash choice, Player 2 revealed their choice, and Player 1's choice was correct, leading to a win for Player 1.