Lottery Smart Contract Base on the Ethereum

Solidity Project

薛劭杰 1930026143

Theoretical foundation

In order to introduce some relevant basics of our application, I will also declare the rules of the design.

1. An administrator will be included in the contract, and the administrator will be responsible for announcing the winner and withdrawing the prize. The way we determine administrators is by randomly choosing a participant, or through another consensus mechanism.
2. The price of each ticket purchased is fixed (1 ETH).
3. Each person can bet multiple times. The more times a customer bets, the greater the chance of winning. Similar to the real world, but sellers can't cheat.
4. Each person can only bet within the specified time. Within 30 minutes of the wagering time, the administrator shall draw the lottery and announce the winner, ie broadcast this information to every participant. Then, we say that the process is transparent and supervised due to the traceability of the blockchain.
5. During the lottery, the contract will perform hash operation on the modulo of the number of bets based on the mining difficulty, time, and number of bets of the current block. The result obtained is the index value of the current winner.
6. The winner will receive the Ethereum in the prize pool proportionally, and the administrator will charge a proportional fee for platform maintenance. It is also the main means of making money.

To conclude, he above rules are automatically implemented after the requirements are met, excluding human subjective judgments. All information is dynamically recovered on the blockchain. Our method of choosing to generate an index indicating the winner is based on hashing. Therefore, no one can modify this index with malicious intent.

Blockchain basics:

1. Blockchain basics: Blockchain technology stores transaction data in blocks, and then each block is connected to the previous block to form a chain-like structure. The data of the transaction includes both parties and the amount of the transaction, and the information is hashed and encrypted. It supports adding new blocks, but once added, it cannot be modified or deleted. Most of us use blockchain technology instead of other transaction data storage technologies. The main reason is that blockchain can ensure data integrity without relying on a central authority, which is based on achieving decentralization. The essence of "mining" on the blockchain is to distribute Tokens to reward or attract users, run blockchain nodes, and become a bookkeeping node of the blockchain to do transactions. block mechanism. This is also the reason why Bitcoin, Ethereum, and other virtual currencies are constantly generating chains.

2. Ethereum basics: Like all other blockchains, Ethereum requires thousands of people to run a piece of software on their personal computers to power the entire network. Each node (computer) in the network is used to run the Ethereum Virtual

Machine, which can understand and execute software written in a specific programming language on Ethereum. The software or application executed by the EVM is called a smart contract. The software remix we use is an EVM language solidity IDE. Smart contracts allow trusted transactions without third parties, which are traceable and irreversible. This is because once a contract is written, it cannot be edited or modified. Therefore, we can guarantee that no matter what the content of the contract is, it will be executed unconditionally. But unfortunately, everything we do on Ethereum must provide gas for data as well as computation. This is because Ethereum relies heavily on the hashing efficiency of miners, which means that more miners, more hashing efficiency, can have a more secure and fast system. In order to attract more miners to the system, they need to make the system profitable for the miners, and the gas fee we pay goes to the miners.

Business Application

In the traditional lottery system, people can only buy paper lottery tickets. After buying them, they need to pay attention to the time of the lottery, and match them one by one to check whether they have won the lottery. If you win the lottery, you need to take the lottery purchase record to the physical store to exchange, not to mention there is a time limit. If the time for redeeming the prize is exceeded, the lottery dealer has the right to refuse to redeem the prize. This has caused a serious problem, and people often need to spend a lot of time and energy on an entertainment-oriented aspect for a very small chance of winning the lottery.

1. Disadvantages of traditional lotteries:
   a. It is impossible to avoid collusion between insiders and notaries to modify the winning results
   b. Data may be tampered with by insiders during storage
   c. Unable to resist the behavior of denial, such as the lottery player suspects that the printed number does not match the one
   d. The award is not transparent

2. Use blockchain technology to solve such problems
   Blockchain technology can fundamentally change the existing system architecture of lottery sales, so as to achieve complete fairness and transparency.
   a. Through encryption technology, all the generated valid information and data are non-repudiation and non-modifiable, that is, the number you put down when you buy is determined, and there is no behavior or possibility of denying.
   b. The operation and management of the system are automatically carried out in accordance with the agreed rules through smart contracts, which will not be controlled by any party, and perfectly achieve decentralization, which can effectively avoid the possibility of internal personnel interfering with the winning results.
   c. All the rules for purchasing lottery tickets and their source code are transparent and open, and even the lottery company cannot change the results.

3. Why can this problem be solved with blockchain technology?

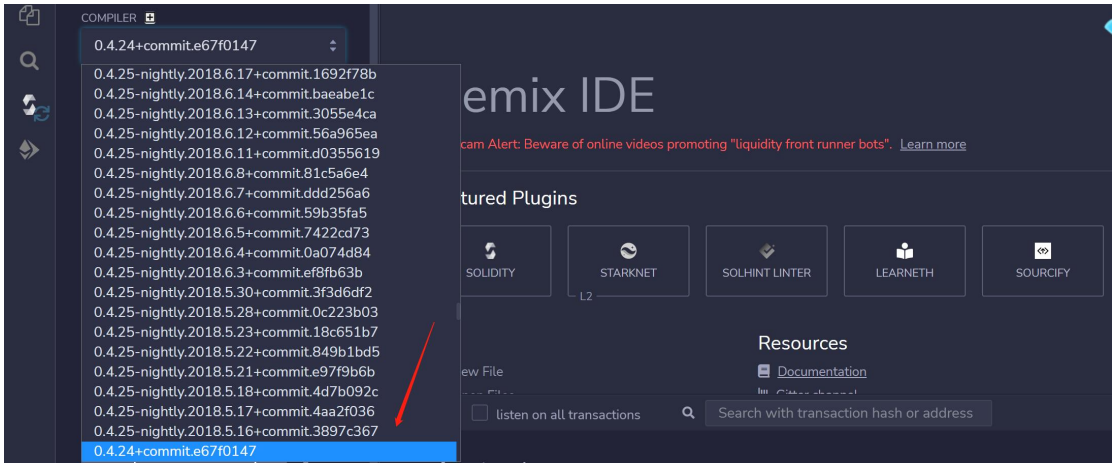
a. The lottery result is calculated according to the contract, which effectively improves the fairness of the lottery system. Because once it starts, not even the lottery company can predict the result of the lottery until it ends. Supervision and impartiality agencies can also provide better and more convenient supervision, which solves the shortcomings of traditional lotteries that are not transparent and open to the public.

b. In terms of user identification, the traditional offline or user name + password login method has been changed, which makes it more difficult to tamper and better meets the needs of users.

c. Effectively reduce operating expenses, which can effectively reduce the cost of maintaining servers.

Methodology

Environment: Since the adaptability of grammar, we set the *Solidity* to ^0.4.24.



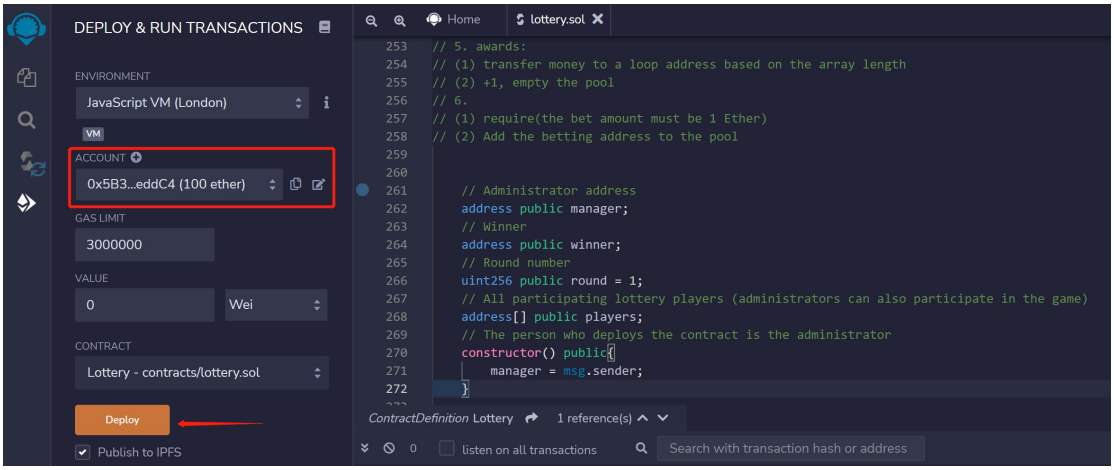
Then we can use code *pragma solidity* ^0.4.24 to set up. It is worth noting that this is very important. If you don't do much, the system will report you a lot of unknowing errors, and most of them are caused by syntax mismatches, which wastes a lot of time when debugging.

1. Structure of contract

a. At first, we should initialize entities including participants (players), admin(manager), winner. There is only one winner and admin, so just use an address variable to store their unique address. And there are usually more than 1 player so we can use an address list to store them. We also define the number of rounds to record how many rounds there are in the lottery.

```
    // Administrator address
    address public manager;
    // Winner
    address public winner;
    // Round number
    uint256 public round = 1;
    // All participating lottery players (administrators can also participate in the game)
    address[] public players;
    // The person who deploys the contract is the administrator
    constructor() public{
        manager = msg.sender;
    }
```

Additionally, the person who deploys the contract is the manager. When the person clicks the deploy button, it will get the sender's address by $msg.sender$.



b. Then we define a $play$ function. In this function, people who use this function can participate the lottery. In order to ensure fairness and rationality, we set the amount of each bet to a fixed 1ether and append the address of the people who click the "play" bottom to the player list. And we should pay attention to the keyword $payable$ which means that the function will occur transaction otherwise it will get some error.

```
function play() payable public{
    // Requires a minimum investment of 0.05ETH
    require(msg.value ==  1 ether);
    // Add the punter to the pool
    players.push(msg.sender);
}
```

c. We define a $draw$ function which use to draw the winning number of the lottery. Firstly, check the validity before the lottery, if no one cannot participate in the lottery. Then random winner requires a random index value, we use the difficulty value, current time, the number of participants as seeds to generate a large number of generated indices, so the randomness and safety can be guaranteed. 90% of the prize pool will be transferred to the winner, and the remaining 10% will be given to the manager as compensation. Finally, you need to add 1 to the number of rounds. Also, this function is that only managers can implement.

d. In order to avoid some other occurrences, we have set up a refund function. The

administrator returns the prize and he should transfer money to the players one by one. Finally, empty the players pool. Also, this function is that only managers can implement.

```
function exit() onlyManager public{
    // Transfer money one by one
    for(uint i = 0; i < players.length; i++){
        players[i].transfer(1 ether);
    }
    round++;
    // Empty the players pool
    delete players; }
```

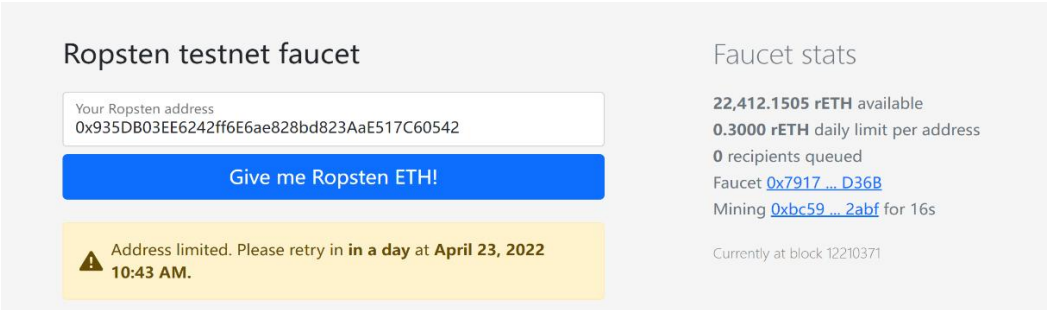
e. The two methods of withdrawal and lottery can only be used by the manager, who is the person who deploys the contract. So we define a modifier and use the require keyword to detect that the sender is the manager.

```
modifier onlyManager{
    // Restricted function, non-administrators are not allowed to call lottery and return functions
    require(msg.sender == manager);
    _;
}
```

f. get function: Everyone who participates in the game can check the funds of the prize pool at any time, and the number of participants can judge whether they need to continue betting. And everyone can view the number of rounds, the manager's address, winner's address and the participant's address to supervise each other.

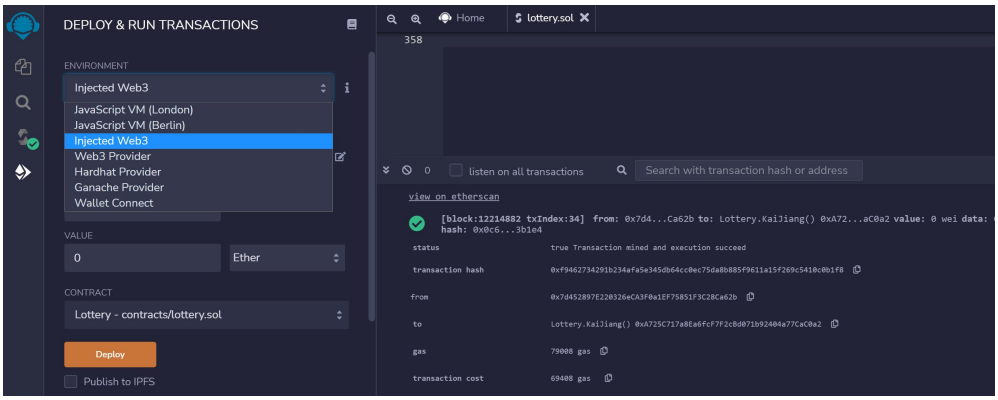
Results

1. Firstly, the blockchain our default choice is Ethereum, because it is only supporting the smart contract, but also supports we can get some basic capital (ether) every day for our test net. Here we choose the $Ropsten\ Testnet$ as our test net. For this net, we can get some ether from the faucet whose website guide is $https://faucet.metamask.io/$ or we can also get some coins from the $https://faucet.egorfine.com/$.

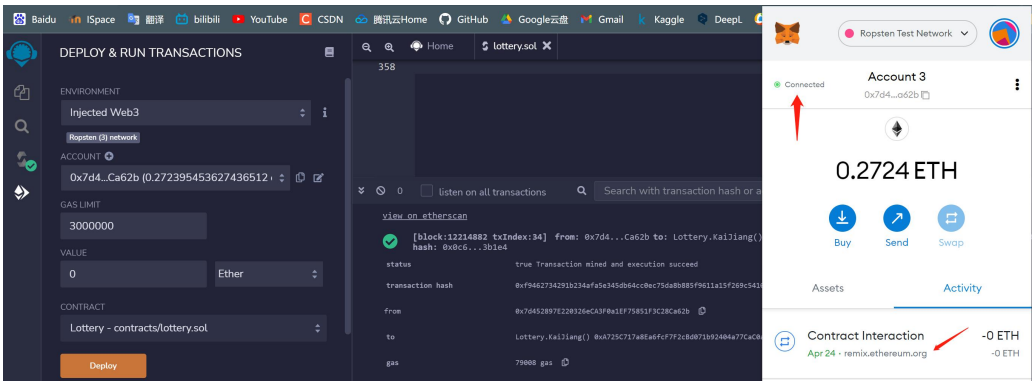


2. After we have some ether coins, we can carry out our group's experiment. The first step is compiling the $.sol$ code file and if it has no any error and bug, we can

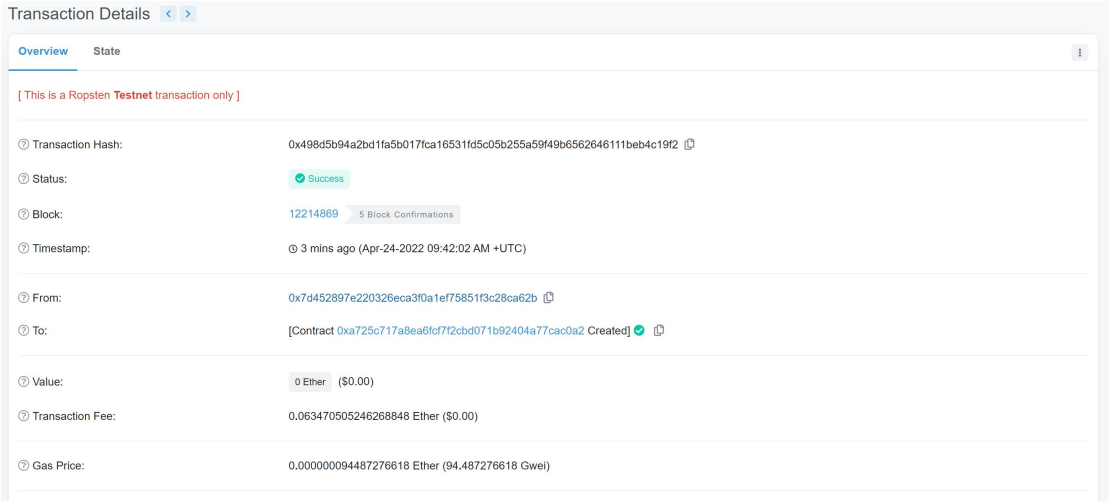
deploy it. In order to connect the account in the *metamask* Chrome plug-in, we should set the environment to *Injected Web3*. Then if we click the deploy bottom, it will automatically open *metamask* and ask you to pay a certain gas fee to create this contract. Also, we can check the connection status in the top left corner of the plugin.



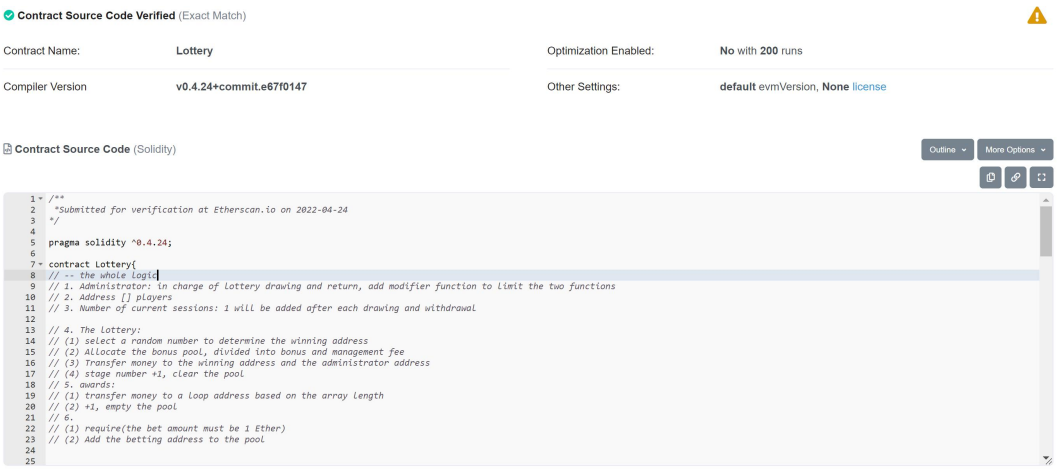
Then just wait about 20 seconds, it will show the contract interaction which means that we deploy this contract to the blockchain successfully.



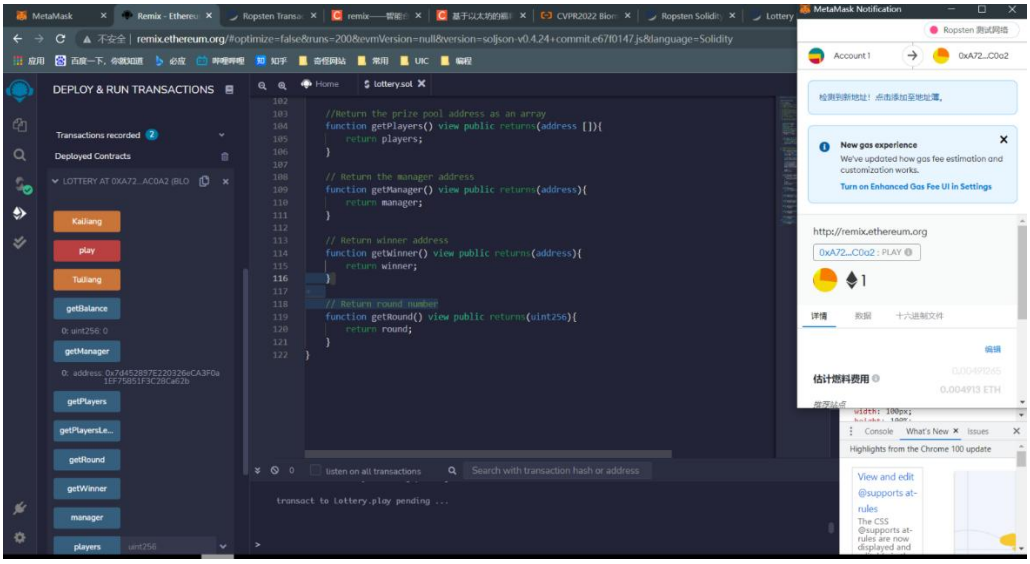
3. Then we can check the information and some details about the contract from the *etherscan* net: *https://ropsten.etherscan.io/*. Actually, it is important because we can make sure others can participate our contract. Here we can see the deployer of the contract who is also the manager of the lottery project, as well as the address of the contract. I send the address to my team mates so that they can call and participate the contract.



When they search the address in this website ($https://ropsten.etherscan.io/$), it will show more detail about it. Before this, I should verify the contract including the code, language and environment vision too in order to ensure the transparency and openness of the contract. Anyone can see the code, version and other details which means that if they want to use our contract, just copy can just compile in the remix or other IDE and just call the address of the contract, and they can use the player function.

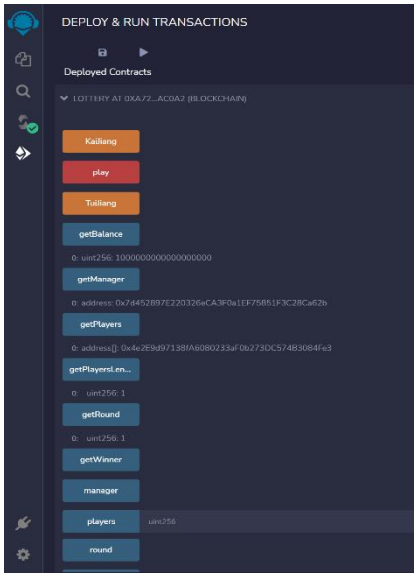


4. Initially, the manager is me and there is no player in the pool so we can see the money is 0 and the address list of players is null. You may be wondering why the button colors are different for different functions, but actually it is a very user-friendly design. The blue one means that anyone can use and no need to any other operations. The orange buttons means that the function is meant to be modified by modifiers, here the draw ($kaijiang$) and $refund(tuijiang)$ are only can be implemented by manager. And the red one can be a bit complicated, users need to input how many coins they want to pay. In other word, this function will happen "$transfer$".
5. Then I call my team mate to call the lottery contract, and make user he does not need to deploy the code again and just only need to copy the contract address and click the at address button. After that, they can use the visual functions. As follow:

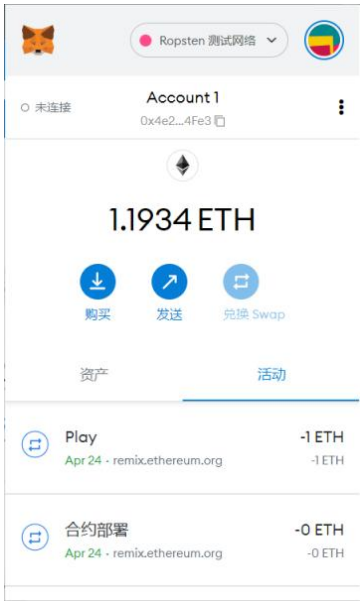


He can see the manager address which is the deployer of the contract which means that he calls the contract successfully, and he can participate the contract by clicking the red play button. Then he will automatically open $metamask$ and he needs to enter the amount to pay 1 ether (other amount will raise an error) because the keyword $require$ which is mentioned earlier.

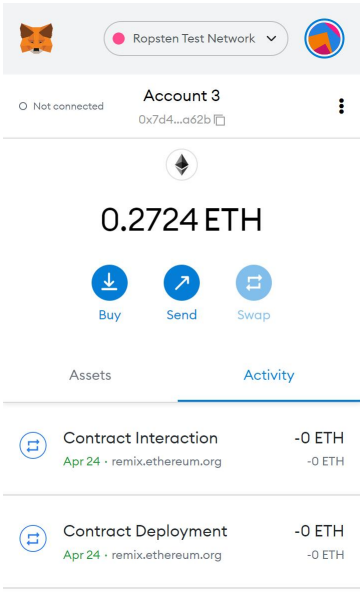
After a few seconds, the system will complete the transaction and we call the $getPlayer()$ and $getBalance()$ functions, we can find that the number of the players become to 1, and the prize pool becomes $1 * 10^{18}$. Then the manager implements the $Kaijiang()$ function, we can get the winner which is the player because there is only one player.



5. After drawing the winning numbers of a lottery, Managers get 10% of the prize pool, and winners get 90%. Then just wait a few minutes, the balance of the player is from 1.12978 ether to become 1.1934 ether. The balance of the player is from 1.12628 ether to become 0.2724 ether. It will cost a certain amount of gas fee.

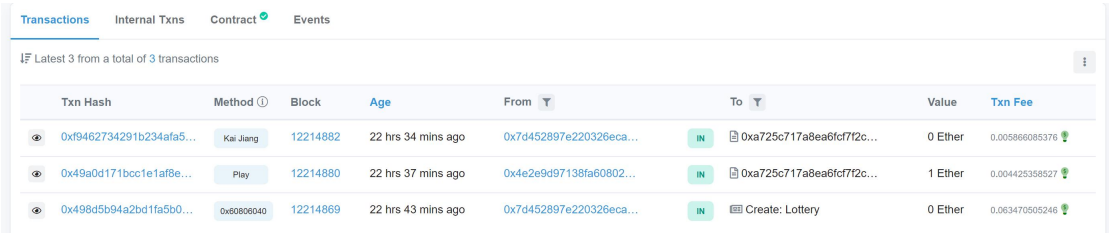


player



manager

6. Finally, we can get all the transaction details in the https://ropsten.etherscan.io/address/0xa725c717a8ea6fcf7f2cbd071b92404a77cac0a2. We can know who did what operations, and transaction happen on which block and transfer value in each time.



Improvement

I think our function can be considered more difficult and challenging than the token contract, because it only involves the transfer and acceptance of currency, and we replace the "minters" with managers. This means we have more functionality to implement. And for the given ICO source code, it is also carried out under the ERC20 protocol, so it is essentially the same. Therefore, our task is to award lottery projects similar to "voting contracts". The deployers of the contracts can earn ETH to a certain extent, and can also ensure the fairness of the lottery, because the generation of winners is completely random and I think this point very attractive.