

Machine Problem: Distributed Coin Flipping Game Using Ethereum

I. OVERVIEW

At this time point, you have learnt the basics about various types of blockchain systems, and in particular, via the hands-on lab, you should now be able to work with Ethereum to write some simple smart contracts. The objective of this course assignment is to use Ethereum to write a smart contract for a Distributed Online Coin Flipping Game, which allows two or more participants to play the game fairly and transfer funds to each other safely. Ethereum is a global, open-source platform for decentralized applications. On Ethereum, you can write code that controls digital value, runs exactly as programmed, and build applications and products that connect people everywhere. By using the services provided by the Ethereum platform, we can build a distributed payment system to support online Coin Flipping game. With it (as an emulator tool), we can also evaluate the performance (benefits and limitations) of various Blockchain implementation choices, for example, system throughput, response time, robustness, security, and privacy, and gain a deeper and insightful understanding about these algorithmic and protocol design choices of blockchain technologies.

II. PROJECT REQUIREMENT

This course assignment aims to develop a simple two-party online Coin Flipping game.

We start by a simplified “coin flipping” game: Alice and Bob want to bet ten dollars. They both agree to the bet ahead of time and the method for determining the winner. Bob will flip a coin in the air, and while it is rotating Alice calls out “Head” or “Tail”. When the coin lands, they both immediately have a clear understanding of who won the bet, and they both have assurance that the outcome was random and that neither of them was able to influence the outcome.

One shortcoming is that the sequence of steps in this ceremony requires that both parties have to be present at the same place at the same time. Also, both parties still have to trust that whoever loses will pay up. In this assignment, we would like to be able to have an on-line coin flipping game that is not only just as “fair”, but also the problem of making sure that the loser actually pays, with the support of **Ethereum**.

The first challenge is replacing the “coin flip” mechanism with some online equivalent. Let’s say we now have two parties, Alice and Bob, who all want to select a number with equal probability. Here’s one attempt at such a protocol. Each of them picks a random number, e.g. Alice chooses 0, Bob chooses 1. They tell each other their numbers, and they combine the output as $val = 00$, 01, 10 or 11. Alice (first participant) wins if $val = 00$ or 11 (i.e., two numbers are the same). Otherwise, Bob (second participant) wins (if $val = 01$ or 10 (i.e., two numbers are different)).

If both of them chose their random numbers independently, this would indeed work. But remember that we are doing this over the Internet, and there is no way to ensure that they all send their numbers “simultaneously”. Alice might wait until she hears Bob’s numbers before broadcasting hers. If she does this, you can see how it is trivial for her to make the final output be whatever she wants. We cannot design the protocol to convince every party involved that none of the other parties can cheat.

To solve this problem, one possible solution is that we can use hash commitments by following the two-round protocol below:

Round 1: Each participant chooses a random number by calling appropriate random number generation function (most program languages like Python, C and Java provide such functions). For example, Alice first picks x and Bob picks y , independently. Then each participant feeds her/his number to a hash function and gets the corresponding hash value, *e.g.*, $H(x)$ and $H(y)$. After that, the hash value and her bet (*e.g.*, 20 lumens) should be sent to an independent 3rd-party banker. The banker will collect and keep the hash values and the bets deposited from each participant as well as the transaction time.

Round 2: The two parties reveal their values, x and y , to each other and the 3rd-party banker. If both x and y revealed by the participants are consistent with the committed $H(x)$ and $H(y)$ in

Round 1, respectively, the two parties and the banker can each decide who is the winner of the game by calculating $((x+y) \bmod 2)$. If the result is 0, Alice is the winner. Otherwise, Bob is the winner. Doing so will guarantee no one can cheat, assuming we use a reliable hash function.

At the end of the game, 95% of the deposits from both parties will be sent to the winner from the banker, the remaining 5% will be retained by the banker as a service fee. Also, the banker should notify all the participants about that the game is over and who is the winner. Such game can be repeated many many rounds.

III. TASKS

1. Please explain/prove whether this protocol actually works or not, in term of fairness, and what are the possible attacks (cheating) among the three parties. In your implementation, how have you solved them? You need to answer this question in your design document.
2. Implement the banker program that can store and update the game transactions. When the banker wants to query the activity history, the system should provide the game transaction records within one day. If some of the participants want to query the activity history, the banker needs to send the transaction record from previous round to each participant and allow them to verify the results.
3. You need to implement the hashing process in **Round 1** yourself using any hash function like Double-SHA256, SHA256, MD5, SHA1, *etc.* and use appropriate random number generator function.
4. There are two user roles: participants and banker. Only one active game at a time needs to be supported. At any given time, there may be at most one banker and at least two participants active with respect to a given round of game. You can write a single smart contract that supports both banker and participants, or two separate smart contracts may be built, one for a banker and another for participants. The banker's account is public to each participant by default.

IV. WHAT TO HAND IN

The result should be a design document, which you hand in along with the source code. Do not get carried away with it, but make sure it convinces the reader that you know how to attack the problem. You can choose your own language to complete this project. The source code and reports should be submitted online.

V. USEFUL LINKS

Ethereum Developer Resources: <https://www.ethereum.org/developers/>