**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 work with Ethereum to build some simple applications. The objective of this course pr um to build a Distributed Online Coin Flipping Game, which allows two or more partici e fairly and transfer funds to each other safely. Ethereum is a global, open-source platfor lications. On Ethereum, you can write code that controls digital value, runs exactly as progr ications and products that connect people everywhere. By using the services provided by the e can build a distributed payment system to support online Coin Flipping game. With it (as can also evaluate the performance (benefits and limitations) of various Blockchain implem ample, 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 project aims to develop a distributed payment system using Ethereum, which allows users to send and receive funds between or among each other, and develop a simple two-party online Coin Flipping game. As an extra credit opportunity, you can use it as an emulator to evaluate the performance of Ethereum as well as other blockchain design choices.



### A. Basic Requirements

This is a project that needs to be done individually. The basic requirement is to finish a simple two-party online Coin Flipping game in Phase I (in-class/lab) and II (partly in-class/lab, and finish the rest after class/lab). If done satisfactorily, you will receive 10 points (Phase I 3: points and Phase II: 7 points).



### B. Extra Credit Opportunities

In addition to the above basic requirements, you can receive additional credit (5 points, on top of the 10 points for the basic requirements) for Phase III of the project by implementing the required extensions:



The details about Phase I, II and III requirements of this project are described as follows:



## III. PHASE I

The first phase of the project is intended to first familiarize you with the Ethereum platform and learn the process of building user accounts. In this phase, you will create two accounts, with which users transfer Ether (ETH) to each other. The basic requirements are:

1. Once the account is created, a new User Interface will pop up to allow user to query the account ID and its balance through that user interface.
2. If the user wishes to initiate a transfer to the targeted account, she can enter the address/accountID and transfer some amount of funds to the targeted account.
3. When user wants to check the activity history, the system can provide the transaction records within a day.
4. Errors, server exceptions, and invalid user input shall produce reasonably informative error descriptions to the user.

## IV. PHASE II

Congratulations! If you have made it to this point, you should now have some working accounts and working knowledge to work with Ethereum. Let us try something new to use it! In this phase, you are expected to implement a two-party distributed 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

phase, 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 implementation 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 s (second participant) wins (if $val = 01$ or 10 (i.e., two numbers are different)).

If both of them chose the ependently, this would indeed work. But remember that we are doing this over the Interne to ensure that they all send their numbers "simultaneously". Alice might wait until she efore broadcasting hers. If she does this, you can see how it is trivial for her to make the er she wants. We cannot design the protocol to convince every party involved that none of 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 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 $3^{rd}$-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 $3^{rd}$-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 $((x+y) \mod 2)$. 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.

Requirements:

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. A separate User Interface is required to each role/particpant. 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.
5. You can build a single application that supports both banker and participants interfaces, or two separate applications may be built, one for a banker and another for a participant. The banker's ID is public to each participant by default.

## V. PHASE III

*Measurements and Analysis*

In order to compare the efficiency of your implementation, you will perform some measurements. I will leave it to you to decide what measurements should be done, for example, throughput, response time, etc.

## VI. WHAT TO HAND IN

Before you start hacking away, plot down a design document. The result should be a system level 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.

1. A short overview on **Ethereum** system: *https://www.ethereum.org/*
2. **Ethereum Developer Resources**: *https://www.ethereum.org/developers/*