Smart Contract based Mobile Crowd Sensing Platform Project Summary

Aaron Ogle
Missouri State University
Springfield, MO

I. Summary

This paper highlights the work done during a semester project involving blockchain, smart contracts, and blockchain based mobile crowd sensing. Attached in this github repository are several solidity files that were created or used during this graduate project based course.

II. Introduction

For my graduate project in computer science I worked with Dr. Liu and Jayanth Madupalli on topics related to Incentive Mechanisms for Mobile Crowdsensing, Behavioral Economics, Blockchain, and Smart Contracts. The overall goal of the project was to learn more about these topics and to try and create a Mobile Crowd Sensing Platform that is implemented through a Smart Contract that incorporates the Endowment Effect into the platform's bidding process. The Endowment Effect is a topic found in behavioral economics that means the value of an item to its owner is higher than the value a nonowner places on the item [10]. Throughout this paper we will refer to the mobile crowdsensing platform as MCS.

To include the endowment effect into the blockchain based mobile crowdsensing platform the idea is to have tokens be given to users in the platform who successfully complete a task along with a monetary reward. Users who have obtained tokens show reliability to complete tasks in the mobile crowdsensing platform and are given preference in the bidding process even if their bids are not higher than other users in the platform.

Note that in the MCS platform the task publishers prefer to select users that have the minimum bids as this will allow publishers to receive a higher payout for receiving completed task data than if they were to select higher bidders. This notion of selecting minimum bids in an auction involves a reverse auction framework which will be discussed in more detail later in this paper. Jayanth is writing his Thesis on this topic and was the architect for the platform process and implemented his idea as a smart contract and then created simulations for the MCS platform; throughout this semester I attempted to help as much as I could with his thesis by offering ideas and different programs and scripts which will be discussed in this paper.

This paper will be divided as follows. Section II will consist of related works in the research areas of Incentive Mechanisms for Mobile Crowdsensing, Behavioral Economics, Blockchain, and Smart Contracts. In Section III I will provide the contributions and work that I have done in attempting to create a blockchain based MCS platform. In Section IV I will discuss the MCS platform structure that was created this semester and is still being worked on. In Section V I will provide a brief comparison between the task publication functionality of the smart contract created this semester with the smart contract that is found in [1], and in Section VI I will provide a conclusion and thoughts on future work.

III. Related Work

The author's in [1] address the problem of most MCS systems being implemented through a centralized structure; the problem with this is the platform is vulnerable to single points of failure which could lead to the entire system being shutdown [1]. Centralization of a MCS platform can also lead to issues of trust between the workers and the task publishers due to workers needing to use their own resources to complete tasks which could provide publishers with sensitive information; along with this problem there is also the problem of financial security as centralized platforms could manipulate processes and workers to obtain improper gains [1].

To address the concern of a centralized system the authors propose a decentralized MCS framework that uses blockchain, the authors refer to their platform as "ChainSensing" [1]. The author's work consists of a MCS framework where mobile users use smart contracts to interact with the decentralized blockchain network to complete tasks and submit data [1]. The authors note that MCS platforms involve computationally intensive processes such as reward determination and travel paths for users; computationally intensive tasks are not ideal to have implemented via a smart contract so the authors utilized off chain resources such as computing oracles to complete these steps [1]. Smart contracts are not used for computationally expensive operations due to the concept of "gas" which is a blockchain transaction fee.

Blockchains are a decentralized ledger that are in a peer-to-peer network [1]. The ledger is a running list of records of transactions that are input into blocks in the chain; this ledger is maintained by different protocols such as Proof-of-Stake, recently implemented in the Ethereum blockchain network, and Proof-of-Work [1]. For the POS and POW consensus protocols to work there need to be network validators in the blockchain to validate the transactions so as to prevent

the "double spending problem" which is where the same cryptocurrency is used more than once for a transaction.

The gas in a blockchain is a transaction fee that is paid to the network validators. The more computational processes that are contained in a smart contract the higher the gas prices will be to run the computations so it is best to have as little computationally intensive processes as possible in a smart contract which is why the authors implemented some of their MCS platform methods off the blockchain [1].

The authors run simulations using the Ethereum blockchain network and note the amount of gas that their contract uses during different processes and are able to show that their framework allows for tasks to be published on the blockchain which will in theory motivate more participants due to its decentralized nature [1].

The authors in [2] address the problem of privacy preservation when it comes to crowdsensing workers in an Internet of Vehicle (IoV) style platform where workers can use their vehicle to help with tracking road congestion and traffic accidents [2]. Noting the problems that traditional communication methods can bring with the IoV the authors propose to use satellite terrestrial systems for the vehicles to communicate with along with using blockchain to act as a channel that the publishers and workers can communicate in [2]. With using blockchain the authors use a smart contract so that the workers and publishers can communicate and since payment methods can be implemented on a blockchain the authors implement a total payment minimized auction algorithm [2].

With their total payment minimized auction algorithm the authors are able to show that it is computationally efficient, individually rational, profitable, systematically efficient, and truthful [2]. The authors are also able to show that their auction algorithm can be deployed on blockchain and that when their aggregation error is larger that their auction has lower total payment than the baseline auction [2]. By aggregation error, the authors are measuring a relationship between the number of participants and whether a task is completed or not; the smaller the aggregation error is the more workers there will need to be to complete a task [2].

The authors in [3] propose to use smart contracts to allow interactions between mobile crowdsensing providers and mobile users for spatial crowdsensing, by spatial crowdsensing we mean that workers will need to be at specific locations in order to perform the task [3]. Smart contracts are used for their MCS architecture to preserve user privacy and to make payments [3]. In their work the authors also design a truthful and cost-optimal auction that will minimize the payment that publishers must pay to the mobile workers; with their auction the authors are able to show that it outperforms other proposals in regards to cost by ten times for higher numbers of workers and tasks [3].

The author's MCS platform works by having crowdsensing service providers sending requests to ISPs (internet service providers) who use a smart contact to register the requests and then collects fees for the CSP for the request; the ISP then runs the auction using another smart contract so that mobile users (workers) can be selected [3]. Workers must pay a participation fee that they will lose if they are selected for the task and they do not submit their measurements, every user in the auction receives a temporary ID from the ISP in order to preserve privacy and the identify of the worker [3].

Finally, the ISP uses another smart contract to collect task participation proof from the workers and will then pay them, and then a fourth smart contract is implemented so that the task publishers (CSP) have access to the data that was collected [3]. With their MCS platform and cost-optimal auction the authors are able to show their method out performs others with minimizing cost [3].

In [4] the authors address the problem of designing incentive mechanisms in Vehicular Ad-hoc Networks (VANETs) due to these networks having frequent topology changes, multi-hop routing, storage space limitations, and potentially selfish users [4]. Noting that most mechanisms are designed using a perspective from traditional economics that assumes the negative and positive effects produced by equal amounts of gain and loss are equally absolute in value, the authors note that the theory of loss aversion points out that these effects are not actually equal and will lead to deviations between the decision making behavior of nodes in actual situations [4].

With this problem in previous works the authors propose a Loss-Aversion-based Incentive Mechanism (LAIM) to promote the sharing of information in VANETs; with their approach the authors are able to show through simulation results that compared with traditional incentive mechanisms their method can increase the average utility of nodes by more than 34.35% which shows the promotion of the cooperation of the nodes in the VANET [4].

Loss aversion refers to the fact that when people are faced with an equal amount of loss and gain that the pain brought on by the loss is much higher than the pleasure that is given by the gain [4]. The authors show an example of this by discussing Amazon's online bookstore that provides free shipping on books if a user pays over a certain amount, many users do not intend to purchase multiple books when arriving to the website but with free shipping being enticing they are willing to spend more to acquire other books so that they will get free shipping [4]. The authors take this idea of loss aversion and apply it to VANETs in order to promote cooperation in the network [4].

In [5] the authors address the problem of current incentive mechanisms being designed for maximizing revenues of single rounds of tasks rather than focusing on a long term incentive and that the incentive mechanisms that do consider long term results do not consider the law of diminishing marginal utility so their actual performances are not as high as expected [5]. To address these shortcomings the authors introduce concepts of capital deposit and intertemporal choice from behavioral economics and introduce an Addiction Incentive Mechanism (AIM) to allow for long-term incentives to users [5]. The authors AIM method entices users to become addicted to cooperative behavior in the crowdsensing platform and also mitigates the effect of diminishing marginal utility

[5]. Through simulations the authors show that their method improves participation rates when compared with other state of the art incentive mechanisms [5].

The authors divide actual cases of addiction into three stages and then apply them to mobile crowdsensing, these three stages are Formation of Addiction, Cultivation of Addiction, and Maintenance of Addiction [5]. As an example, the authors compare fast-food consumption addiction with their idea for applying the concept of addiction to mobile crowdsensing. The authors note that people tend to eat fast food in order to satisfy hunger as well as cravings, this is the formation of addiction, and can be seen in the MCS platform as users gain more experience utility when they participate in tasks [5].

Fast food consumers may eat fast food multiple times and addiction will start to develop due to this increase in consumption, in a MCS platform, users will be rewarded for completing tasks multiple times and this will cause their total capital deposit to increase [5]. Due to appetite and cravings people will continue to consume fast food as opposed to stopping completely, this can be seen in a MCS light due to negative utility and utility, users will continue to participate in tasks from start until end [5]. Using these concepts of addiction the authors create a MCS incentive mechanism that improves participation rates amongst its users [5].

In [6] the authors note that with the ever increasing growth of mobile traffic that comes along with the evolving Internet of Things (IoT) that there is becoming an increasing need for more access points (APs) to address data offloading, however, the authors note that many of these access points are not willing to participate with the data offloading process [6]. To address this problem, the authors note that better incentive mechanisms must be created so that more APs are willing to help with data offloading; the authors propose an incentive mechanism that uses the concepts of loss aversion and the anchoring effect [6].

Loss aversion has been previously mentioned above, however, the anchoring effect is a bias where individual's decisions are influenced by information presented earlier to them denoted as the "initial anchor" which is used as a reference point for their later decisions [6]. The authors combine the ideas of the anchoring effect and loss aversion into a price break discount concept that incorporates the fact that when people are offered discounts then they will tend to spend more money, and if they do not participate and take advantage of the discount then they will feel regret [6]. Based off these concepts the authors build an incentive mechanism for data offloading in order to motivate access points to participate more [6]. Through simulations the authors are able to show that their incentive mechanism is able to improve the utility of access points ultimately showing that their method is effective with participation [6].

The work in [7] defines the process of a reverse auction as a tendering process where a contract that is being bid on is awarded to the bidder that submits the lowest bid [7]. From this, the authors analyze a reverse auction as a game where an extra condition is created where the bids must be unique, from this the winner of the auction will be the bidder who submits the lowest positive unique integer (LUPI) [7].

The authors study the problem to find optimal strategies for bidders and consider two extremes in their work, the first being that all of the players in the reverse auction will make random choices except for one user who they denote as "Player A", the other extreme they analyze is that all bidders will make rational choices under a given assumption that is provided in their work called the "lowest number rule" [7]. The lowest number rule is an assumption that the bidders in the auction will be indifferent about any potential strategy of other bidders and will always try and pick the lowest number [7].

Based off the two extremes mentioned above, the authors are able to prove that the best strategy for when all bidders are randomly making bids is for the bidder not making random bids, "Player A", to bid a value of one [7]. The best strategy for when the lowest number rule is taken into consideration is for a bidder to submit a bid of either one or two with an equal probability of 0.5 [7]. The author's work provides us with a view of different strategies for two extreme scenarios in a reverse auction and also provides us with a definition of a reverse auction to use.

The authors of [8] create a reverse auction based incentive mechanism for mobile crowdsensing and look into the scenario where users might drop out of participating in the sensing task after they have been selected as a winner [8]. The authors note that many of the works on mobile crowdsensing do not consider the situation where winners of the auction may quit in the middle of the sensing task and thus not actually finish and submit any useful data to the task publisher citing that most crowdsensing works assume that once a worker is selected for a task that this user will actually complete the task which is not practical with real world applications [8].

To address the problem that workers may quit in the middle of a task the authors create a task-centric winner selection based reverse auction and a payment scheme that should motivate more workers to finish a task and provides them with higher rewards [8]. The task centric winner selection algorithm is a reverse auction that follows a greedy user selection approach where the algorithm picks user groups by different iterations where the users have smaller bids but have a notion of "large task value" [8]. The algorithm employs a greedy method where the bid of each user is divided by the sum of the tasks value for that bid, once all of the bids are obtained, each bid is divided by the sum of the tasks value and are then sorted in ascending order and the winner is selected based on this [8].

For payment, each winner will be paid an amount of monetary rewards which will be the highest bid that the winner can submit to win the auction, along with their bid the worker receives the amount of that bid plus the task rewards for that bid where the task rewards is determined by the amount of time to perform the task [8]. The authors prove that their method is computationally efficient, individually rational, budget feasible, and truthful [8]. Through simulations the authors show that their method increases the average utility of

users and also increases the task coverage ratio when compared with other crowdsensing incentive mechanisms [8].

In [9] the authors further add to the work of [8] introducing a user-interaction based incentive model which helps to address the problem of users potentially dropping out of a task before they have actually completed it [9]. The user interaction based incentive model notes that if users drop out of the task then more users will need to be recruited so that the task can be completed, this problem causes an increase in cost for the platform and could also introduce data redundancy [9].

In their work the authors propose a model that will allow users who are dropping out of a sensing task the ability to resell the unfinished task to a new user through a double auction which should improve the task completion rate without increasing the overall budget of the mobile crowdsensing platform [9]. The reselling process becomes a multi-user matching problem and is turned into a one-to-one matching problem on a bipartite graph where an edge exists between a buyer and seller if and only if a seller has bid on tasks held by a buyer, here the seller is a user who is dropping out of the task and a buyer is a new user interested in completing the task [9]. Through simulations the authors are able to show that the model improves the sensing task completion rate and also the social welfare of users with their graph based user matching algorithm [9].

The authors of [10] note that the Internet of Vehicles (IoV) consists of road side units and vehicle nodes and that road side units are unable to satisfy all of the requests from vehicle nodes due to limited bandwidth thus causing a need for additional data offloading methods, the offloading methods are typically left to spare vehicle nodes but these nodes can be selfish and have low participation so an effective incentive mechanism is needed to encourage these vehicles to participate in the data offloading process [10]. The authors note that work is being done in this area but that they fail to address loss aversion [10]. To tackle this problem the authors introduce the endowment effect from behavioral economics for increasing the participation rate of the spare vehicles in the IoV network [10].

The endowment effect is a finding in behavioral economics that states the value of an item to its owner is much higher than the value of that item to a non-owner, an example of this is an experiment known as the coffee mug experiment where participants who own a coffee mug placed more value on their mug than on Swiss sugar which they do not own when in reality the value of the two items are almost the same; from the experiment it was found that 78% of the participants would not exchange their own coffee mug for the Swiss sugar due to the influence of the endowment effect where the owner regarded their item as their endowment and preferred to keep this item instead of exchanging it for an item that had equal value [10].

Another example of the endowment effect which the authors apply to their incentive mechanism is when an organization has a product trial period where the user will not have to pay the full cost of an item for a certain period of time, these methods are typically used in marketing where organizations will allow users to use a product at a discounted rate for a certain period of time before they will be given the option to either pay the full price of the product or to stop using it [10]. The authors note a scenario where a cable news organization offers a ”golden package” to users that is a trial period for a product at a discounted rate, after experiencing the product for the trial period users tend to renew their subscription to the service at the full price due to the endowment effect [10].

To apply the endowment effect to their incentive mechanism the authors propose what they refer to as endowment compensation; the authors note that in reverse auctions there will be some users with a lower willingness to participate, by using endowment compensation the authors propose a method that lowers the range of the bid between providers with low participation rates and the winners so that the winning probability of the user is increased which will lead to more willingness of participation by the users [10]. In other words, the authors propose to implement the endowment effect in their work by lowering the range of the bids of users with low willingness to participate with those of the bid winners in order to improve the likelihood that users that are not willing to participate have a higher chance of being a winner in the reverse auction [10].

From their simulation results the authors note that due to this endowment compensation the spare vehicles with higher bid prices have a possibility to win thus increasing the payment costs of the task publishers, however, the total of the provided utility and the requester are higher and ultimately show that their method is effective [10]. The ideas formed in [10] provide us with a framework for which we can apply the endowment effect to our mobile crowdsensing platform which will be discussed later in this paper.

IV. MCS, Behavioral Economics, Blockchain, and Smart Contract Based Research Work

For this section I will provide an overview of different aspects of this project that I worked on and will conclude the section with a weekly update summary that I provided throughout the semester for this work.

The starting point for smart contract based research work was to learn as much as I could about the Solidity programming language. Solidity is an object-oriented programming language that is used for implementing smart contracts on a blockchain platform, Solidity is most commonly used on the Ethereum blockchain network since it is ran on the Ethereum Virtual Machine.

To start using Solidity, our initial IDE is the Remix Online IDE that can be accessed through an online browser. Remix allows for developing, deploying, and debugging Ethereum backed smart contracts. An additional tool that was useful is a MetaMask wallet which is a crypto wallet that is useful for Web3 applications. A MetaMask wallet was used so that we could store fake crypto coins in the wallet to use for testing the contracts. To get fake crypto coins we had to go to different crypto ”faucets” to retrieve them. These ”faucets”

were websites that allow for users to obtain test crypto coins to use for testing.

The first half of the semester consisted of creating various smart contracts in order to become more familiar with the languages as I was completely new to it; along with learning how to write smart contracts with the Solidity programming language I also kept up with reading work being done in the areas of Incentive Mechanisms, Behavioral Economics, Blockchain, and Smart contracts of which these research works were included in the above literature review.

The first few smart contracts that I created were a wallet based smart contract where a user could store their crypto currencies on the blockchain and withdraw them later, this is essentially the "Hello World" program of smart contract creation. The second smart contract was implementing the Euclidean Algorithm for finding the greatest common divisor of two numbers to become more familiar with functions in the Solidity language. We will explain this smart contract in further detail below.

Each smart contract must specify a SPDX-License-Identifier which is an identifier that prevents copyright infringement by noting whether the contract can be re distributed or not. The second line of a smart contract must note what version of the Solidity compiler is being used as updates are continuously being made to the environment that include new features that may not be compatible with previous compiler versions. In the below section, we show the GCD.sol program as an algorithm:

Algorithm 1 GCD Smart Contract

$contract\ GCD\{$
$uint256\ number;$
$uint256\ x;$
$uint256\ y;$

$function\ store(uint256\ num)\ public\{$
$number = num;\}$

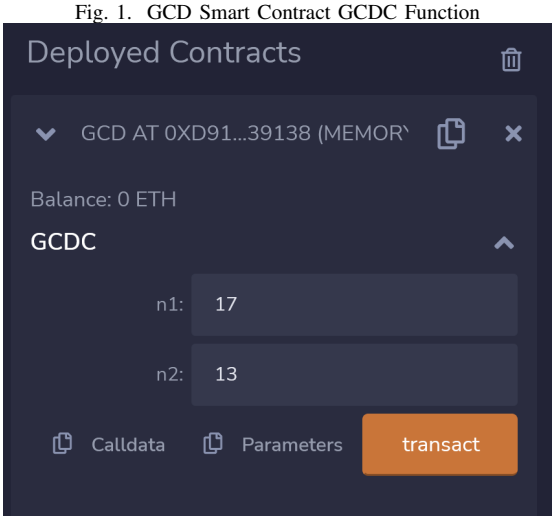
$function\ retrieve()\ public\ view\ returns\ (uint256)\{$
$return\ x;\}$

$function\ GCDC(uint256\ n1, uint256\ n2\ public\{$
$x = n1;$
$y = n2;$
$if\ (n2 == 0)\ \{$
$x = n1;\}$
$else\{$
$GCDC(n2, n1\%n2);\}$
$\}$
$\}$

It is worth noting that smart contracts cannot have functions that are the same as the contract name, hence is why in the above we name the function GCDC for Greatest Common Divisor Computation instead of GCD since this is the contracts name.

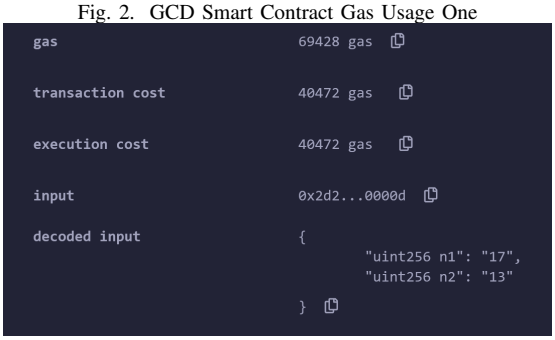
After we have compiled the above smart contract in Remix we can input two numbers and have its GCD returned. Images of this process are provided below. In the first image, we are inputting 17 and 13 into our contract to see what the GCD is; since these integers are relatively prime we expect the result to be 1.

Fig. 1. GCD Smart Contract GCDC Function



In the second image, we show that after a function has been called we can check the logs to see how much gas was used in the process of calling that function.

Fig. 2. GCD Smart Contract Gas Usage One



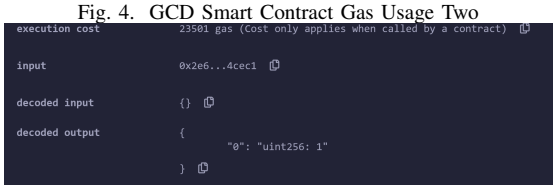
In the third image, we call the retrieve function so that the result of the GCDC function is returned.

Fig. 3. GCD Smart Contract Retrieve Function

Lastly, we show the gas usage for the retrieve function call. From the results we can see that the retrieve function returns one which is what we were expecting. We can also see from the results that the GCDC function requires more gas to run than the retrieve function. This makes sense due to the fact that the GCDC function is more computationally intensive than

the retrieve function and as was discussed earlier the more computationally intensive an operation is in the contract the more gas will be consumed.

Fig. 4. GCD Smart Contract Gas Usage Two



The next program that was created using the Solidity language was an implementation of Bubble Sort so that I could become familiar with using arrays. During the week that I worked on the Bubble Sort implementation I also was able to reach out to the authors of [1] who were kind enough to provide us with the source code for their smart contract that they used in their work. This provided us with a guide on how to implement a MCS platform through a smart contract which was very useful to us.

The next smart contract that I created was a simple reverse auction smart contract. The reverse auction contract allowed for me to learn more about how to use arrays in the Solidity programming language. The gist of the smart contract is to allow multiple bidders to make a bid and then the smart contract will return the bid that is the smallest value along with the bidder's wallet address and bid. This program allowed me to become familiar with how a simple auction type framework could be implemented in the Solidity programming language.

During this time Jayanth had create a framework for a mobile crowdsensing platform and Solidity smart contract program that would be built upon for the remainder of the semester. After this was constructed it became clear that we would at some point need to find a way to run numerous simulations on the smart contract and that the Remix IDE may not be the best tool to use for this. Due to this, it was decided that we should switch to using the Hardhat environment instead of the Remix IDE. The next few weeks consisted of becoming more familiar with the Hardhat environment as I found it to have a much steeper learning curve than was required for the Remix Browser based IDE.

While learning about the Hardhat environment we continued to meet to discuss the smart contract implementation of the proposed MCS system and decided that it would be a good idea to have a measure of location and distance for users in the system. Because of this, I attempted to create a smart contract method library in Solidity that could be used to calculate the euclidean distance between points. The idea behind this would be to treat the users in the MCS system as points in the euclidean plane and then be able to have a notion of distance between these points and a measure of where these points at were in space.

This became tricky because by design Solidity does not support float types so instead of being able to directly take the square root of a number (which is needed to compute euclidean distance) I had to calculate the square roots approximation. To do this, I found the Babylonian Method for computing square roots which I was able to implement in a smart contract. Learning this method was very interesting as it was not something that I had ran into before. Ultimately, we did not end up using this as Jayanth was able to find a cheaper alternative as it would be more computationally expensive to use this method. The other alternative method is to assign users a number for a specific grid that they would be in in the MCS system which is much easier to work with than trying to treat all users as points in the euclidean plane and then compute distances between them.

After attempting to help out with creating different simulation scripts in Solidity Jayanth was able to create test scripts using the Typescript language. I had never used Typescript before so I attempted to help with the simulations as best as I could. To become more familiar with the typescript language I was able to see how Jayanth was using it for his simulation and I was able to take those ideas and create test scripts for seeing how much gas was used for publishing tasks in the smart contract created this semester and then create test scripts for the smart contract found in [1] and see how much gas this smart contract used with publishing their MCS tasks. I will discuss the results of these scripts later in the Experimental Analysis section of this work.

V. MCS Framework

In the below section I will introduce the overall architecture for the mobile crowdsensing platform that was implemented as a smart contract.

The first phase of the MCS framework is to have Task Publishers and Task Workers register themselves. Modifiers have been input into the smart contract so if a user is already registered then they will not be able to register again. Other modifiers exist in the smart contract to prevent certain actions from being taken by a worker when a task publisher should be performing those actions and vice versa.

The next phase of the MCS system is to have the task publishers publish tasks. Currently, the publish task function takes as arguments the budget needed for the task, the number of workers needed for the task, the location of the task, and the deadline for when the task needs to be submitted. The tasks themselves are structures that contain a task ID, publisher user ID, worker user ID, a task budget, number of workers, deadline, location, and the current status of the task.

Once tasks have been published through the smart contract then workers can fetch the tasks that have a status of open. Once the worker views the tasks that are available they are then given the option to bid on that task if they are interested in completing it. Once the users have submitted bids then the publishers of the tasks can fetch all of the bids and then close the bidding process. The bidding process will be where the concept of the endowment effect will be used by using a concept of "tokens" where users that complete bids are given these tokens, users with tokens will be given a preference in the bidding process so even if they have a higher bid than

others then the probability that they will be selected for a task will be higher given that they have these tokens.

After the bidding process has been closed and winners have been selected then the workers move into the task phase of the MCS platform. When the workers have finished their task then they submit the task and a hashed value - these will be used later by the task publisher to validate the data. Once tasks have been submitted then the MCS publisher can fetch the tasks and set tasks to completed. The task publishers will then validate the data and the hash, once validated then the task publisher will transfer the reward to the worker from their wallet to the workers wallet through the contract and the task is marked as completed.

VI. Experimental Analysis

Now that we have discussed the architecture of the MCS system that has been implemented this semester we will compare the gas useage of publishing tasks through this smart contract with the gas usage of publishing a task using the smart contract created in [1]. During the semester I reached out to the authors of [1] who were kind enough to give us their .sol file for their smart contract. One thing to note before performing the analysis, their smart contract has fewer methods than the smart contract created this semester so their smart contract is expected to have less gas usage; however, it still seems beneficial to investigate this from a learning and analysis perspective.

To publish a task with the contract in [1], the following parameters are needed; longitude, latitude, reward, time, interval, and number where time is the expected amount of time that it will take for a worker to complete a task and interval is a time window that is acceptable to the task publisher for the worker to complete the task [1]. In the smart contract created this semester, to publish a task the following parameters are needed; the budget that the task publisher has for the MCS platform, the number of workers needed to complete the task, the deadline for when the task should be completed, and the location of the task. For all of our simulations done below for testing gas consumption these parameters will be randomly generated.

To perform this analysis, we must first setup the Hardhat Ethereum environment. To do this, I use WSL (Windows Subsystem for Linux) and write the code/scripts using Visual Studio. To run tests, the Hardhat Ethereum environment allows for the use of local network nodes for development so we can make the call - npx hardhat node - and are then given a local network to test on. This gives us fake accounts with fake Ethereum coins assigned to them to work with, an image of this is provided below.

Now that we have some test accounts to work with, the next thing that we must do is use a Typescript script to deploy the contracts themselves to the test network. Once these contracts are deployed then we will be ready to run test scripts on them.

To measure gas usage, we will have each smart contract publish 10, 20, and 30 tasks respectively. To do this, we again need to use the Typescript language to write a script to call and run the functions with simulated data. Running these simulations with these number of tasks we obtain the below results. Note that in the table we denote CS for the contract found in [1] and CSI for the contract that was created during the project for this semester.

Fig. 5. Test Accounts

```
Account #98: 0xFE0f143FcAD5B561b1eD2AC960278A2F23559Ef9
 (10000 ETH)
Private Key: 0xc1b5e6b1cd081956fa11c35329eeb84d31bceaf7
253e84e0f90323d55065aa1f

Account #99: 0x98D08079928FcCB30598c6C6382ABfd7dbFaA1cD
 (10000 ETH)
Private Key: 0xa3f5fbad1692c5b72802300aefb5b76036401801
8ddb5fe7589a2203d0d10e60

WARNING: These accounts, and their private keys, are pu
blicly known.
Any funds sent to them on Mainnet or any other live net
work WILL BE LOST.
```

Fig. 6. Contract Deployment

```
aogle@DESKTOP-0EPMK2E:~/ethereum_dev/CrowdSensing/csi
/scripts$ npx hardhat run deploy_chainsense.ts
0x5FbDB2315678afecb367f032d93F642f64180aa3
ChainSensing Deployed
aogle@DESKTOP-0EPMK2E:~/ethereum_dev/CrowdSensing/csi
/scripts$ npx hardhat run deploy.ts
0x5FbDB2315678afecb367f032d93F642f64180aa3
CSI Deployed
```

Fig. 7. Gas Usage for Publishing 10 Tasks

```
aogle@DESKTOP-0EPMK2E:~/ethereum_dev/CrowdSensing/csi/
scripts$ npx hardhat run publish_gas_cs.ts
Accounts found: 100
--------------------------------------
Testing Gas Use in ChainSensing Contract...

Gas Consumed Publishing Tasks:  600580
--------------------------------------
aogle@DESKTOP-0EPMK2E:~/ethereum_dev/CrowdSensing/csi/
scripts$ npx hardhat run publish_gas.ts
Accounts found: 100
--------------------------------------
Testing Gas Use in CSI Contract...

Gas Consumed Publishing Tasks:  3822822
--------------------------------------
```

The above image shows the results obtained when publishing ten tasks and the table below shows the results for each of the simulations.

Tasks	CS Gas	CSI Gas
10	600580	3822822
20	1092960	9814097
30	1582528	18841746

From the results we can see the CSI contract burning more gas than the CS contract which is what was expected given that the CSI contract contains more in it than the one found in [1].

VII. Conclusion and Future Work

From doing this graduate project in computer science I learned a lot about Incentive Mechanisms for Mobile Crowd-

sensing, Behavioral Economics, Blockchain, and Smart Contracts. Before working on this graduate project I had no experience with Solidity or any of the other tools that we used for this project such as the Remix and Hardhat environments or the Typescript language so it was a great learning experience being able to work with these tools and languages. I was also not very knowledgeable about Incentive Mechanisms, Behavioral Economics, Blockchain, or Smart contracts so I was able to learn about these topics as well by doing this project.

I also learned a great deal from Jayanth and Dr. Liu by working on this project and also read through a lot of works that I might not have come across had it not been for Dr. Liu's guidance. Work is still being done on the Smart Contract and plans for my future work, if needed, are to work on the theoretical analysis of the MCS platform. From [9] we can see that other works show that their methods are computationally efficient, individually rational, budget feasible, and truthful. My goal is to prove that these factors hold true for the MCS platform that was created this semester as the first of the three show that the mechanism is viable [9].

REFERENCES

[1] X. Tao and A. S. Hafid, "ChainSensing: A Novel Mobile Crowdsensing Framework With Blockchain," in IEEE Internet of Things Journal, vol. 9, no. 4, pp. 2999-3010, 15 Feb.15, 2022, doi: 10.1109/JIOT.2021.3094670.

[2] Z. Ma, Y. Wang, J. Li and Y. Liu, "A Blockchain Based Privacy-Preserving Incentive Mechanism for Internet of Vehicles in Satellite-Terrestrial Crowdsensing," 2021 7th International Conference on Computer and Communications (ICCC), 2021, pp. 2062-2067, doi: 10.1109/ICCC54389.2021.9674460.

[3] D. Chatzopoulos, S. Gujar, B. Faltings and P. Hui, "Privacy Preserving and Cost Optimal Mobile Crowdsensing Using Smart Contracts on Blockchain," 2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), 2018, pp. 442-450, doi: 10.1109/MASS.2018.00068.

[4] Liu J, Huang S, Xu H, Li D, Zhong N, Liu H. Cooperation Promotion from the Perspective of Behavioral Economics: An Incentive Mechanism Based on Loss Aversion in Vehicular Ad-Hoc Networks. Electronics. 2021; 10(3):225. https://doi.org/10.3390/electronics10030225

[5] J. Liu, S. Huang, D. Li, S. Wen and H. Liu, "Addictive Incentive Mechanism in Crowdsensing From the Perspective of Behavioral Economics," in IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 5, pp. 1109-1127, 1 May 2022, doi: 10.1109/TPDS.2021.3104247.

[6] J. Liu, W. Gao, D. Li, S. Huang and H. Liu, "An Incentive Mechanism Combined With Anchoring Effect and Loss Aversion to Stimulate Data Offloading in IoT," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4491-4511, June 2019, doi: 10.1109/JIOT.2018.2883452.

[7] Zeng, Qi & Davis, Bruce & Abbott, Derek. (2007). Reverse auction: The lowest unique positive integer game. Fluctuation and Noise Letters. 14. 0-0. 10.1142/S0219477507004069.

[8] Y. Liu, H. Li, G. Zhao and J. Duan, "Reverse Auction Based Incentive Mechanism for Location-Aware Sensing in Mobile Crowd Sensing," 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1-6, doi: 10.1109/ICC.2018.8423009.

[9] Y. Liu, X. Xu, J. Pan, J. Zhang and G. Zhao, "A Truthful Auction Mechanism for Mobile Crowd Sensing With Budget Constraint," in IEEE Access, vol. 7, pp. 43933-43947, 2019, doi: 10.1109/ACCESS.2019.2902882.

[10] J. Liu, W. Wang, D. Li, S. Wan and H. Liu, "Role of Gifts in Decision Making: An Endowment Effect Incentive Mechanism for Offloading in the IoV," in IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6933-6951, Aug. 2019, doi: 10.1109/JIOT.2019.2913000.