Smart Contract Based Traffic Tickets

Ritika Barakol
Department of Computer Science
University of Southern California
Los Angeles, CA 90007
Email: barakol@usc.edu

Rebecca Lee
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90007
Email: leerk@usc.edu

***Abstract*—Currently, traffic rules are implemented by police. This requires citizens to trust that a central authority will fairly enforce the rules. With the advent of IoT and Smart Cities, it is possible to leverage this infrastructure to decentralize the detection and penalization of speeding violations, mitigating the need for law enforcement officials. Here we propose a concept for a smart contract pertaining to the enforcement and regulation of speeding laws.**

I. Introduction

Road safety is an important issue in urban cities. In 2016, speeding was a contributing factor in 27% of traffic fatalities in the US [1]. Increased monitoring of drivers behaviors and enforcement of traffic laws could help to decrease speeding-related deaths. Under the current system in California, police manually enforce speed limits on the road using handheld radar guns. However it is impossible for police to catch all violators. There is also a sense of public distrust in police enforcement and speeding tickets are often contested in court. In other locations traffic violations are detected using speeding cameras. However inaccuracy in these types of systems and legal issues make them unreliable for proper enforcement of traffic rules.

Recent innovations in IoT have introduced the concept of smart cities that run themselves. In the future, city-wide road sensors may be used to collect data regarding vehicle speeds, leading to automatic issuance of speeding violations and mitigating the need for law enforcement officials. Implementing automatic traffic law enforcement systems requires security and robustness in the smart city network. Blockchain technology provides a system where recorded sensor data can be stored securely.

In this paper, we explore how blockchain technology and IoT devices can be combined for efficient management of speeding tickets. We propose a system where road-side sensors send data regarding vehicle speeds to the blockchain for automatic detection of traffic law violations and generation of speeding tickets. We simulated an initial design in Ethereum for proof of concept. We also make suggestions for future improvements to the initial design and discuss potential issues.

II. Related Work

Speed cameras, such as the Gatso camera [2], have been used to enforce traffic rules in countries including the United Kingdom, Canada, and the United States. These devices detect vehicle speeds through Doppler radar or LIDAR signals. If the measured speed is over the limit, the speed camera takes a photograph of the offending vehicle and uses automatic number plate recognition to track and fine violators. These types of camera systems face public criticism due to their centralized nature and accuracy issues. Violations could be issued erroneously due to hardware faults in the sensors or with incorrect measurement methods. Speeding tickets are often contested in court because of these inconsistencies, wasting city resources. Under the current system, data regarding drivers' speeds are not available immediately. It may take days or even weeks for drivers to receive a traffic ticket in the mail. During this time, drivers may lose the ability to gather evidence that disproves (or proves) the speeding camera's decision. The lag also opens the possibility for data to be tampered with. Conflict of interest issues can also arise when private contractors are employed to install and maintain traffic enforcement cameras. Contractors may be bribed to install faulty cameras as a greedy ploy to raise money for the government, and contractors paid in commission depending on the number of issued tickets may be tempted to install inaccurate sensors in order to increase ticket count. Overall these issues make speeding sensors installed by the government untrustworthy to the public.

III. A Blockchain-Based Approach to Solving Inaccuracies in Automatic Traffic Enforcement Systems

In order to mitigate problems caused by inaccurate speeding camera readings, we propose to combine blockchain technology and smart city data for automatic, uniform detection and management of speeding violations on the road. Blockchains have several properties that could be used for this purpose:

- With a high bandwidth blockchain network, data gathered by speed sensors can be written to the blockchain quickly, making it readily available to drivers.
- The immutability property of blockchains ensures that sensor data is not tampered with before traffic rule violators receive tickets.
- By using a permissionless blockchain and making all collected speed data publicly viewable, drivers can gather evidence that can potentially be used to determine if road-side sensors are faulty.
- Smart contracts on the blockchain can be used to automate the issuance and management of traffic tickets.

- Smart contracts can be used to issue traffic tickets in a uniform and unbiased manner.

IV. System Design

For our initial design, we consider that participants in the system are either speed sensors or vehicles. Sensors measure the speeds of vehicles that pass by and send them to a smart contract. We assume that all participants are honest. This means that sensors are not malicious or faulty and they report vehicle speeds accurately. It also assumes that drivers do not contest traffic tickets that were issued fairly. The smart contract determines which vehicles' speeds are not within the legal range and records violations on the blockchain. The blockchain is used to store information about sensors (e.g. location, speed limit at that location) and vehicles (e.g. vehicle ID number, owner, registration information). The blockchain also stores information regarding violations and outstanding tickets issued to each vehicle.

We assume that an IoT speed sensor, S_i, at some location has a maximum speed limit, L_i, as illustrated in Fig. 1. Assume that a vehicle, V_j, drives by S_i with at a speed, p_{ji}, at time t_k. Also assume that speeding violations have a constant fine F. If $p_{ji}>L_i$, the smart contract will add a ticket to a list of V_j's outstanding tickets. The ticket is a structure that contains V_j, S_i, p_{ji}, and t_k. Each ticket is designed to have a unique identification number. This makes it easier to retrieve information regarding a particular violation. Drivers can also send payments to the smart contract. If the payment is greater than F, a violation is removed from their vehicle's outstanding ticket list.

From this concept, the smart contract has two main functions related to the management of speeding tickets: (1) it automatically generates speeding tickets for violating vehicles; and (2) it accepts payments for fines and removes violations from drivers' records.

V. Smart Contract Code

We implemented the smart contract in Solidity and used Ganache for testing. The contract contains three entities, i.e, Vehicle, Sensor and Ticket. These are initialized by calling the code in Listing 1.

Listing 1: Function to be called by sensor to check the speed of a passing vehicle

```
function checkSpeed(address vid, uint
    measuredSpeed) public {
        if (measuredSpeed > sensors[msg.
            sender].speedLimit) {
                giveTicket(vid, msg.sender,
                    measuredSpeed);
        }
}
```

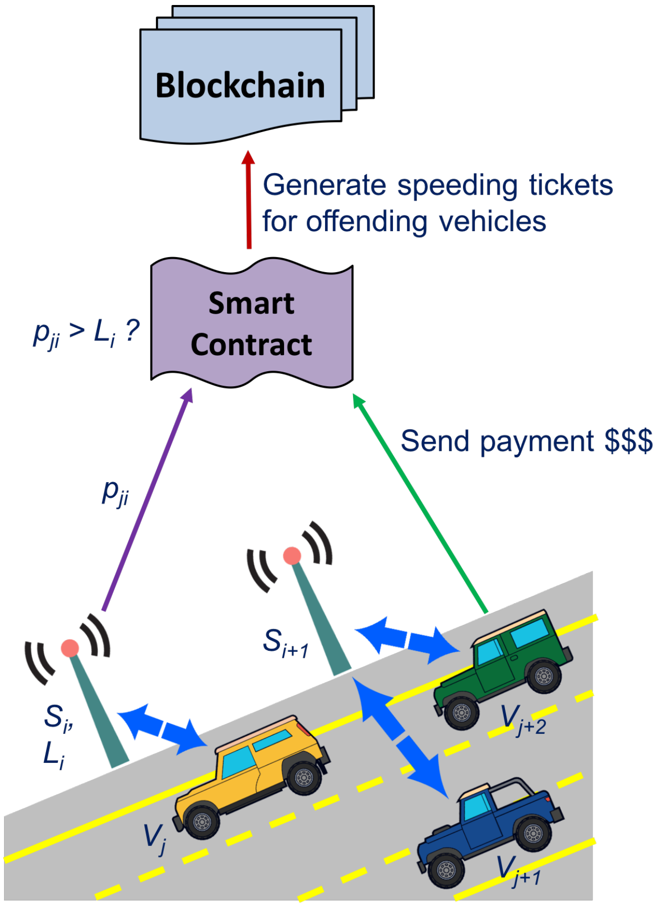


Fig. 1: Overview of the proposed system.

Listing 2: Function that gives a speeding vehicle a ticket. To be called by a sensor.

```
function giveTicket(address vid, address sid,
     uint measuredSpeed) public {
         TicketAdded(now, vid, msg.sender,
             measuredSpeed);
         registeredVehicles[vid].
             pendingTickets[ticketNumber] =
             Ticket(ticketNumber, vid, sid,
             measuredSpeed, now);
         registeredVehicles[vid].ticketIds.
             push(ticketNumber);
         ticketNumber++;
}
```

A Vehicle can pay one of its pending tickets by invoking the payTicket function shown in Listing 3. For the experiment, we have a fixed ticket price of 1 ether. If more than the required amount is transferred, then the change is returned to the sender. Once the ticket is payed off, the ticket is removed from the pending ticket list of that vehicle and a TicketPayed event is triggered.

Listing 3: Function to pay a ticket. To be called by a vehicle.

```
function payTicket(uint ticketId, address
    receiver) payable public{
        require(msg.value>=1 ether);
         uint change=0;

        if(msg.value>1 ether){
                change = msg.value-1;
                msg.sender.transfer(change);
                ChangeReturned(now, ticketId,
                     msg.sender, change);
        }
        receiver.transfer(1 ether);
        delete registeredVehicles[msg.sender
            ].pendingTickets[ticketId];
        delete registeredVehicles[msg.sender
            ].ticketIds[ticketId];

        TicketPayed(now, ticketId, msg.sender
            , msg.value);
}
```

VI. Results

We setup a simulation consisting of two sensors with different maximum speed limits (30mph and 40mph respectively) and four vehicles. Each sensor and vehicle has an account in Ethereum. During initial setup, sensors and vehicles are registered by calling the addSensor and registerVehicle functions in the smart contract. Fig. 2 shows the output of a sample transaction where a sensor is added with a speed limit of 30mph.

Following initial setup, we ran simulations for the following scenarios:

1) Vehicle 1 moves with an average speed of 35mph from sensor 1 to 2. In this scenario, the vehicle should get a speeding ticket from Sensor 1 and no ticket will be given by Sensor 2.
2) Vehicle 2 moves with an average speed of 45mph from sensor 1 to 2. In this scenario, the vehicle should get a speeding ticket from both sensors.
3) Vehicle 3 and 4 moves with average speeds of 20 mph from sensor 1 to sensor 2. In this scenario, both the vehicles do not get any tickets from any of the sensors.
4) Vehicle 1 pays off one of its tickets using the payable function implemented in the smart contract.

Fig. 4 shows the transaction output when a speeding ticket was issued to a vehicle. Here the vehicle (owned by the account at web3.eth.accounts[3]) drives by the sensor (account at web3.eth.accounts[1]) with speed 40mph. In Fig. 2, the sensor was added to the network with a maximum speed limit of 30mph. Since the vehicle's speed is over the limit, a ticket is issued and a TicketAdded event is generated in Ethereum. A ticket with an ID number of 1 is added to the vehicle's outstanding ticket list.

The first function call in Fig. 5 shows the transaction when the vehicle adheres to the traffic rules. The same vehicle and sensor as in Fig. 4 are used, but now the vehicle drives at a speed of 20mph. Since this is under the speed limit, no TicketAdded event is issued.

The second function call in Fig. 5 demonstrates how a driver can view outstanding tickets for their vehicle. Calling the displayVehicleTickets function returns the list of outstanding tickets. In this example, the vehicle only has one ticket with an ID of 1.

Figs. 6 and 7 demonstrate the ticket payment functionality of the smart contract. The paying vehicle is at account index 3. Initially, the vehicle starts with 100.0 Ether in their wallet (Fig. 6, fourth row). The vehicle calls the smart contract payTicket function and sends an amount of Ether greater than the ticket fine. After paying the ticket, the vehicle's account balance drops to 97.32 Ether, as seen in Fig. 7.

The simulation results show that our basic smart contract functions work.

VII. Suggestions for Future Work

Our smart contract code is a first-order model of the proposed system used for proof of concept. The model works on the assumption that all nodes are honest. Networks deployed in the real world will have issues that were not modeled in this network. We discuss some of these issues here.

A. Real-World IoT Devices

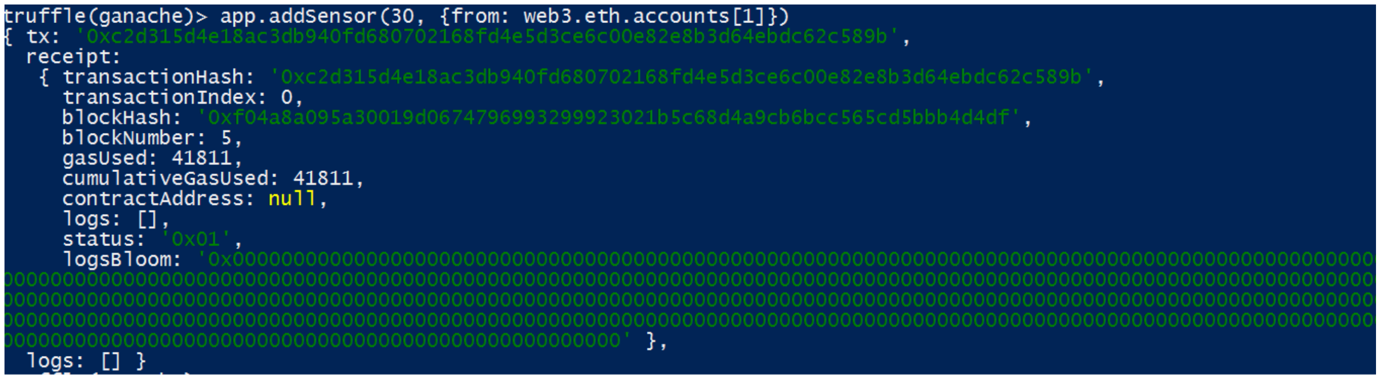
To expand the PoC we plan to employ a physical IoT sensor network to measure the speeds of vehicles. We can use this network to study the scalability of the blockchain with regards to real-time data collection and transactions.

B. Privacy

Another major area of concern which needs to be explored is privacy. Many people are not comfortable with having personal data regarding their driving habits visible to the world. One way in which this concern can be addressed is through the use of hybrid blockchain platforms. Speeding information can be made private and only certain users can be given permissions to access this data, while still allowing anyone to participate in the blockchain. If data regarding speeds is required to be written on the blockchain to make the system transparent, encryption techniques can be used to protect personal information and to uncouple speed data from driver identities.

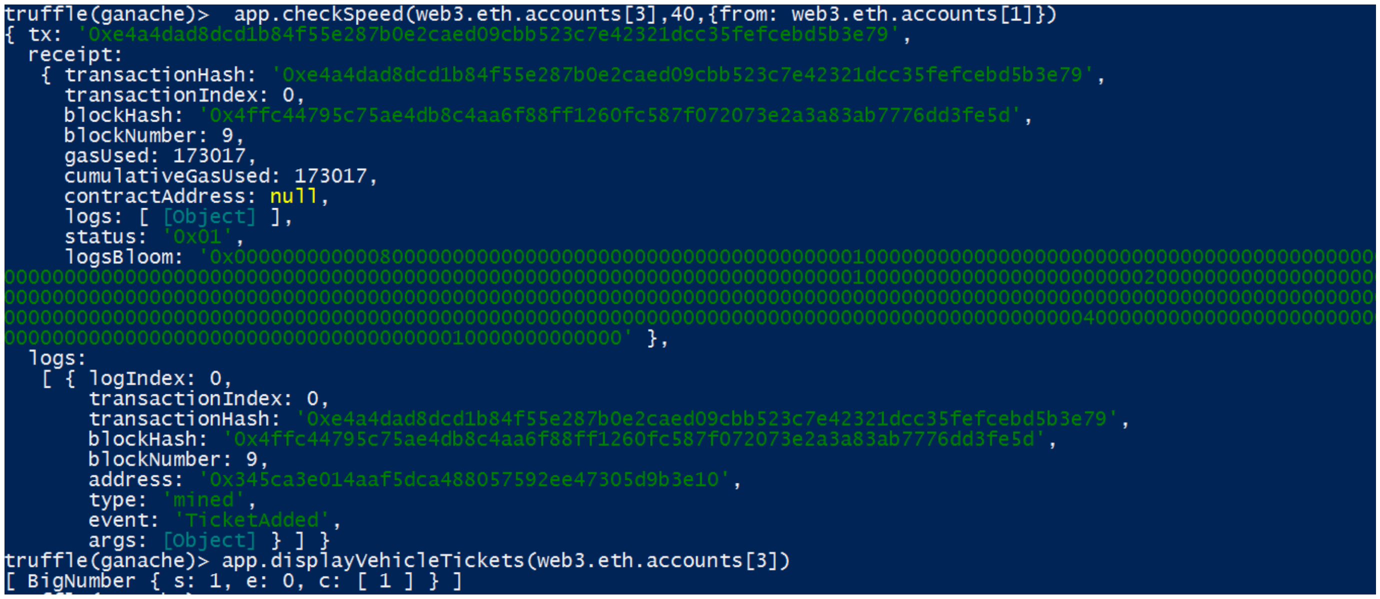
C. Sensor Malfunctions and Malicious Sensors

To address the concern of sensor malfunctioning, one possibility to be explored is the use of a consensus mechanism employed on data received from neighbouring vehicles. The idea here is for vehicles to measure and send data on the speeds of vehicles surrounding them to the smart contract, allowing them to participate in the decision-making process. Rather than solely relying on road-side sensors to provide accurate information, data from both sensors and all vehicles can be collected and a majority consensus mechanism can be applied to determine when speeding violations should be issued. If the majority of vehicles nearby report the speed of a particular



```
truffle(ganache)> app.addSensor(30, {from: web3.eth.accounts[1]})
{ tx: '0xc2d315d4e18ac3db940fd680702168fd4e5d3ce6c00e82e8b3d64ebdc62c589b',
  receipt:
   { transactionHash: '0xc2d315d4e18ac3db940fd680702168fd4e5d3ce6c00e82e8b3d64ebdc62c589b',
     transactionIndex: 0,
     blockHash: '0xf04a8a095a30019d067479699329992302lb5c68d4a9cb6bcc565cd5bbb4d4df',
     blockNumber: 5,
     gasUsed: 41811,
     cumulativeGasUsed: 41811,
     contractAddress: null,
     logs: [],
     status: '0x01',
     logsBloom: '0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000' },
  logs: [] }
```

Fig. 2: Add a Sensor



```
truffle(ganache)>  app.checkSpeed(web3.eth.accounts[3],40,{from: web3.eth.accounts[1]})
{ tx: '0xe4a4dad8dcd1b84f55e287b0e2caed09cbb523c7e42321dcc35fefcebd5b3e79',
  receipt:
   { transactionHash: '0xe4a4dad8dcd1b84f55e287b0e2caed09cbb523c7e42321dcc35fefcebd5b3e79',
     transactionIndex: 0,
     blockHash: '0x4ffc44795c75ae4db8c4aa6f88ff1260fc587f072073e2a3a83ab7776dd3fe5d',
     blockNumber: 9,
     gasUsed: 173017,
     cumulativeGasUsed: 173017,
     contractAddress: null,
     logs: [ [Object] ],
     status: '0x01',
     logsBloom: '0x00000000000080000000000000000000000000000000000000100000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000001000000000000000000000002000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000400000000000000000000000
00000000000000000000000000000000000000100000000000000' },
  logs:
   [ { logIndex: 0,
       transactionIndex: 0,
       transactionHash: '0xe4a4dad8dcd1b84f55e287b0e2caed09cbb523c7e42321dcc35fefcebd5b3e79',
       blockHash: '0x4ffc44795c75ae4db8c4aa6f88ff1260fc587f072073e2a3a83ab7776dd3fe5d',
       blockNumber: 9,
       address: '0x345ca3e014aaf5dca488057592ee47305d9b3e10',
       type: 'mined',
       event: 'TicketAdded',
       args: [Object] } ] }
truffle(ganache)> app.displayVehicleTickets(web3.eth.accounts[3])
[ BigNumber { s: 1, e: 0, c: [ 1 ] } ]
```

Fig. 3: Register a Vehicle

vehicle that is much different from that detected by a sensor, it is safe to assume that the sensor is malfunctioning and is in need of service.

D. Malicious Sensors

To prevent ticket errors in the case where faulty sensors are purposely installed for malicious reasons, multiple sensors operated by different parties can be installed at each location. Instead of relying on one sensor contractor, multiple contractors can provide data and work to reach consensus on the violation status of vehicles. The peer-to-peer consensus network proposed in the previous section can also be used to mitigate malicious data by decentralizing the ticket issuing process. With a full peer-to-peer network, it is even possible that drivers can police themselves without the need for road-side speed sensors.

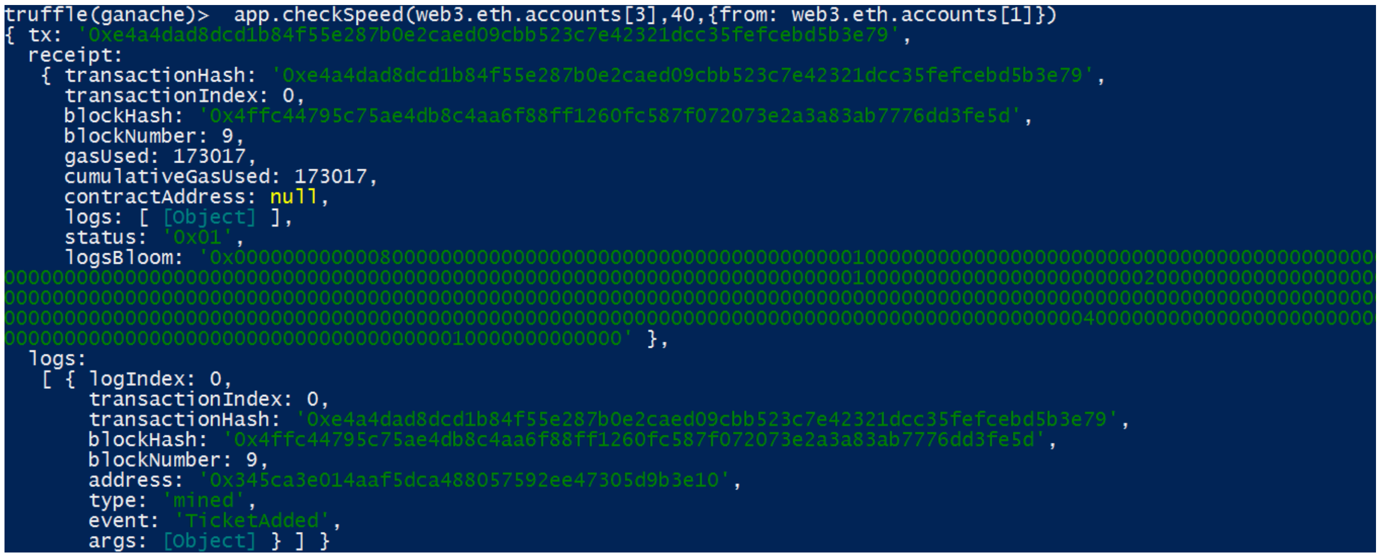
VIII. Conclusion

In this paper, we propose a smart contract based automatic implementation of traffic rules. This approach leverages Blockchain technology and IoT technology used in smart cities. The approach helps reduce the reliance on traffic police to enforce traffic laws. The use of blockchain provides immutability to the data collected from sensors, hence helping reduce disputes in traffic courts regarding tickets given by the police. The approach leverages the sensors present in smart cities. We use the Ethereum blockchain and use Solidity to write the smart contract. We use Ganache to test the contract under various scenarios. We also provide suggestions for future work.

References

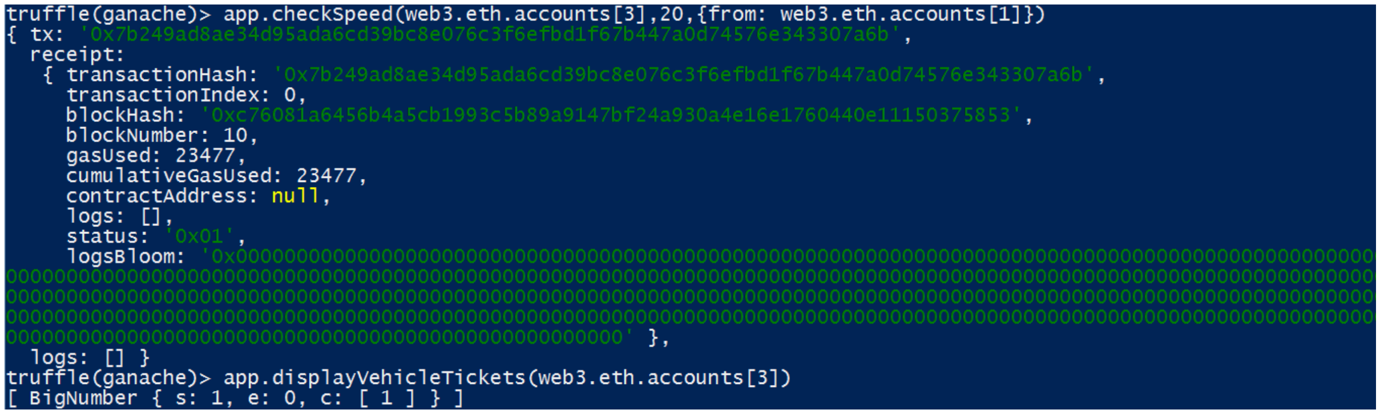
[1] N. H. T. S. Administration. (2018) Speeding. [Online]. Available: https://www.nhtsa.gov/risky-driving/speeding

[2] Gatso. (2018). [Online]. Available: http://www.gatso-usa.com/



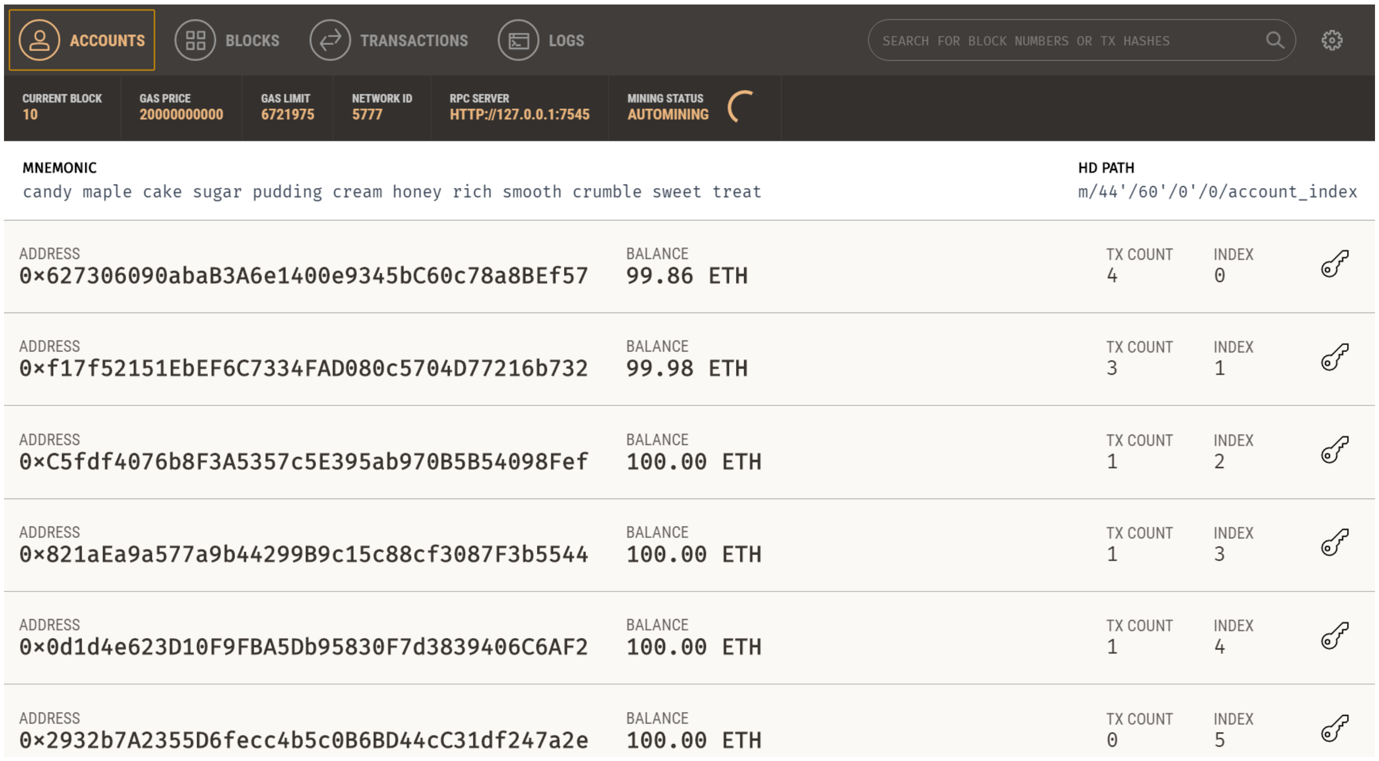
```
truffle(ganache)>  app.checkSpeed(web3.eth.accounts[3],40,{from: web3.eth.accounts[1]})
{ tx: '0xe4a4dad8dcd1b84f55e287b0e2caed09cbb523c7e42321dcc35fefcebd5b3e79',
  receipt:
   { transactionHash: '0xe4a4dad8dcd1b84f55e287b0e2caed09cbb523c7e42321dcc35fefcebd5b3e79',
     transactionIndex: 0,
     blockHash: '0x4ffc44795c75ae4db8c4aa6f88ff1260fc587f072073e2a3a83ab7776dd3fe5d',
     blockNumber: 9,
     gasUsed: 173017,
     cumulativeGasUsed: 173017,
     contractAddress: null,
     logs: [ [Object] ],
     status: '0x01',
     logsBloom: '0x00000000000080000000000000000000000000000000000000100000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000100000000000000000000002000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000400000000000000000000
00000000000000000000000000000000000010000000000000' },
  logs:
   [ { logIndex: 0,
       transactionIndex: 0,
       transactionHash: '0xe4a4dad8dcd1b84f55e287b0e2caed09cbb523c7e42321dcc35fefcebd5b3e79',
       blockHash: '0x4ffc44795c75ae4db8c4aa6f88ff1260fc587f072073e2a3a83ab7776dd3fe5d',
       blockNumber: 9,
       address: '0x345ca3e014aaf5dca488057592ee47305d9b3e10',
       type: 'mined',
       event: 'TicketAdded',
       args: [Object] } ] }
```

Fig. 4: Give Ticket to a Speeding Vehicle



```
truffle(ganache)> app.checkSpeed(web3.eth.accounts[3],20,{from: web3.eth.accounts[1]})
{ tx: '0x7b249ad8ae34d95ada6cd39bc8e076c3f6efbd1f67b447a0d74576e343307a6b',
  receipt:
   { transactionHash: '0x7b249ad8ae34d95ada6cd39bc8e076c3f6efbd1f67b447a0d74576e343307a6b',
     transactionIndex: 0,
     blockHash: '0xc76081a6456b4a5cb1993c5b89a9147bf24a930a4e16e1760440e11150375853',
     blockNumber: 10,
     gasUsed: 23477,
     cumulativeGasUsed: 23477,
     contractAddress: null,
     logs: [],
     status: '0x01',
     logsBloom: '0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000' },
  logs: [] }
truffle(ganache)> app.displayVehicleTickets(web3.eth.accounts[3])
[ BigNumber { s: 1, e: 0, c: [ 1 ] } ]
```

Fig. 5: No Ticket Given to Vehicle Following the Speed Limit



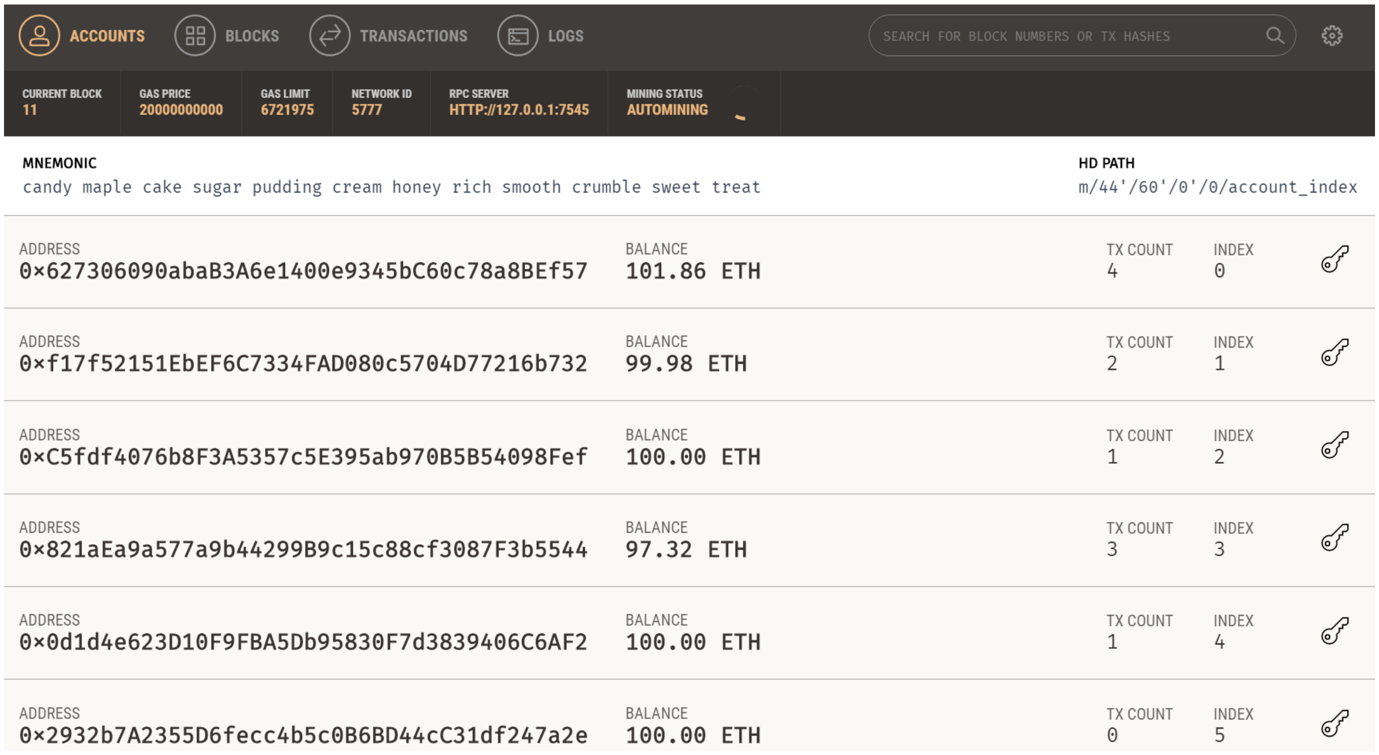
CURRENT BLOCK 10 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING

MNEMONIC
candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0x627306090abaB3A6e1400e9345bC60c78a8BEf57	99.86 ETH	4	0
0xf17f52151EbEF6C7334FAD080c5704D77216b732	99.98 ETH	3	1
0xC5fdf4076b8F3A5357c5E395ab970B5B54098Fef	100.00 ETH	1	2
0x821aEa9a577a9b44299B9c15c88cf3087F3b5544	100.00 ETH	1	3
0x0d1d4e623D10F9FBA5Db95830F7d3839406C6AF2	100.00 ETH	1	4
0x2932b7A2355D6fecc4b5c0B6BD44cC31df247a2e	100.00 ETH	0	5

Fig. 6: Before Vehicle Pays Ticket



CURRENT BLOCK 11 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING

MNEMONIC
candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0x627306090abaB3A6e1400e9345bC60c78a8BEf57	101.86 ETH	4	0
0xf17f52151EbEF6C7334FAD080c5704D77216b732	99.98 ETH	2	1
0xC5fdf4076b8F3A5357c5E395ab970B5B54098Fef	100.00 ETH	1	2
0x821aEa9a577a9b44299B9c15c88cf3087F3b5544	97.32 ETH	3	3
0x0d1d4e623D10F9FBA5Db95830F7d3839406C6AF2	100.00 ETH	1	4
0x2932b7A2355D6fecc4b5c0B6BD44cC31df247a2e	100.00 ETH	0	5

Fig. 7: Vehicle Pays Ticket