- TrustCar -

A new decentralized solution for car histories

1st Jeremy Joel Harisch
Department of Computer Science and Engineering
Pohang University of Science and Technology
Pohang, South Korea
jeremy@harisch.de

2nd Kimberly Lange
Department of Computer Science and Engineering
Pohang University of Science and Technology
Pohang, South Korea
kimberly@postech.ac.kr

3rd Yu Tongzhou
Department of Computer Science and Engineering
Pohang University of Science and Technology
Pohang, South Korea
2770526768@qq.com

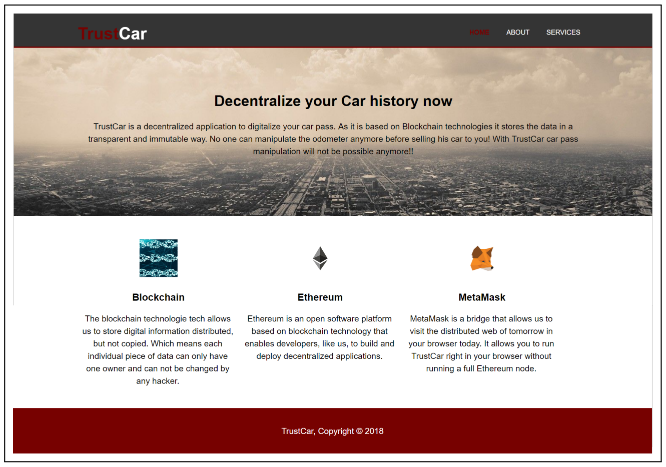
***Abstract*—This thesis introduces a decentralized web-based application called 'TrustCar', which we developed to digitize car passes. By putting car data and repair information in the blockchain, TrustCar makes them immutable and credible. The application runs on the Ethereum blockchain and users can interact with it through a web-interface. Furthermore, the smart contract was developed using 'Solidity'.**

***Index Terms*—Ethereum, decentralized application, smart contract, TrustCar, Car Pass, web-based application**

I. Introduction

Manipulated odometers or car passes are an omnipresent fear of every person buying a used car. There are many cases where someone bought a used car thinking that it is in good condition, but after some time or at the next inspection it was discovered that the car had had an accident in the past, which was not documented in the car pass. Sellers earned lots of money just by easily modifying the car pass and buyers had to suffer. The unique feature of blockchain makes it the best technology to solve this problem. Due to its characteristic of being immutable and secure, car data and repair history stored in the blockchain can never be manipulated and are always credible. So, we came up with the idea of developing a decentralized application which functions as an interface between the user, who can be a car owner or a mechanic, and the Ethereum blockchain.

TrustCar is composed of front-end part and back-end part. In front-end, TrustCar consists of an uncomplicated website that functions as a user interface. To develop such a website markup, scripting and programming languages like HTML, JavaScript and CSS are employed. In back-end, TrustCar uses the Ethereum blockchain to store all the data and to provide the front-end with the data it requires when necessary. To make use of this blockchain, a smart contract had to be developed. For this 'Solidity' was used. Furthermore, the contract was published on the blockchain with the help of the 'MIST'-browser.



Decentralize your Car history now

TrustCar is a decentralized application to digitalize your car pass. As it is based on Blockchain technologies it stores the data in a transparent and immutable way. No one can manipulate the odometer anymore before selling his car to you! With TrustCar car pass manipulation will not be possible anymore!!

Blockchain

The blockchain technologie tech allows us to store digital information distributed, but not copied. Which means each individual piece of data can only have one owner and can not be changed by any hacker.

Ethereum

Ethereum is an open software platform based on blockchain technology that enables developers, like us, to build and deploy decentralized applications.

MetaMask

MetaMask is a bridge that allows us to visit the distributed web of tomorrow in your browser today. It allows you to run TrustCar right in your browser without running a full Ethereum node.

TrustCar, Copyright © 2018

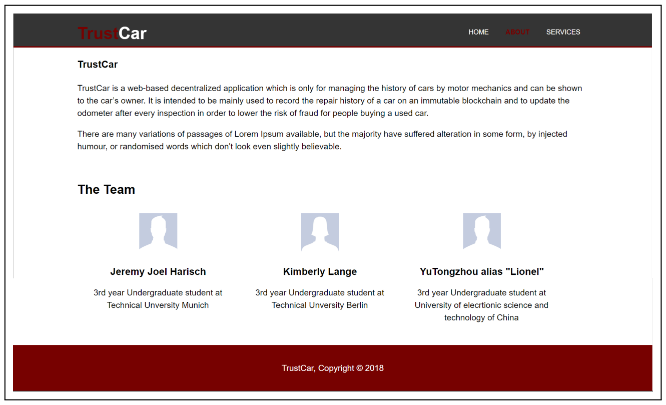
Fig. 1. Homepage of TrustCar

Finally, the 'TrustCar'-webpage uses the web3.js library to interact with the blockchain.

II. Detailed description of DApp

TrustCar is about storing car data and repair history in Ethereum blockchain to digitize the car pass. Like real car passes, TrustCar has to deal with different kinds of users, such as car owners and garages. In our application we also modeled the role of an admin, which is a garage with additional rights. By logging in on the website using their Ethereum wallet account, different users should be able to carry out different actions:

1) A car owner should be able to see the data of her car, including its model, license plate, owner, odometer, time of construction and place of construction. More importantly, she should be able to see the repair history of the car. This is useful when the owner wants to sell or rent the car to others because she can show the history



TrustCar

TrustCar is a web-based decentralized application which is only for managing the history of cars by motor mechanics and can be shown to the car's owner. It is intended to be mainly used to record the repair history of a car on an immutable blockchain and to update the odometer after every inspection in order to lower the risk of fraud for people buying a used car.

There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable.

The Team

Jeremy Joel Harisch
3rd year Undergraduate student at Technical Unversity Munich

Kimberly Lange
3rd year Undergraduate student at Technical Unversity Berlin

YuTongzhou alias "Lionel"
3rd year Undergraduate student at University of elecrtionic science and technology of China

TrustCar, Copyright © 2018

Fig. 2. About-page of TrustCar

of this car to them and let them know the real condition of the car.

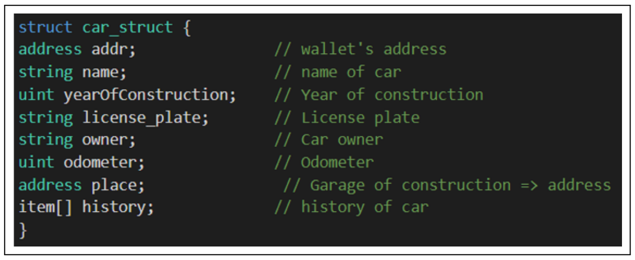
2) A garage is the place where the car gets repaired after it is damaged, where the car receives regular maintenance or where the transaction of the car is validated. A garage should also be able to see the data and repair history of a car because it needs this information to repair the car. Also, after finishing repairing the car, the garage should be able to update the repair history and sometimes add some damage information into the blockchain. Additionally, it should be possible to update the data of a car such as odometer when the car receives regular maintenance or when it gets repaired. Finally, when the car is sold from one person to another, the original owner of the car should go to a garage to let it change the identity information of the car such as car owner and the license plate. Thus, the transaction can be validated.
3) An admin is a garage with the administrative right. It has all the features of a normal garage. Besides that, it can also create new car and garage wallets. An admin would be a car manufacturer which will create a new car account for every newly-created car. In addition, when an admin has bad performance, its admin right can be taken away by an upper-level admin, or an admin can give other garages the admin rights.

III. Smart Contract

A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. It is the core when we want to use Ethereum blockchain as our back-end and database. After deploying and publishing it, the smart contract can never be modified again and is open to everyone. Thus, it is nearly impossible to modify someone's car data.

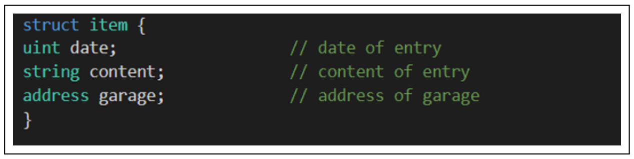
In order to realize the goal of TrustCar, a well-functioning smart contract is needed. Details of the data structure and the functions of the smart contract are explained separately for different kinds of user in the following.

In addition, all functions that do not write data into the blockchain but only read and return information from it were implemented as so-called view functions. Their major advantage is that they do not need any transaction gas. Those functions can be called directly. If a user wants to trigger one of the other functions, a wallet log-in pop-up opens. Besides, if the log-in was successful but the user does not have the rights to perform the wished tasks, it will be caught by 'required'-statements in the contract.



```
struct car_struct {
address addr;               // wallet's address
string name;                // name of car
uint yearOfConstruction;    // Year of construction
string license_plate;       // License plate
string owner;               // Car owner
uint odometer;              // Odometer
address place;               // Garage of construction => address
item[] history;             // history of car
}
```

Fig. 3. Struct of cars



```
struct item {
uint date;                  // date of entry
string content;             // content of entry
address garage;             // address of garage
}
```

Fig. 4. Struct of history items

A. Data structures for cars and garages

Initially, a struct named `car_struct` (see figure 3) is created for a car's data and its repair history. Several entries are created in this struct to store the information of the car such as its odometer and license plate.

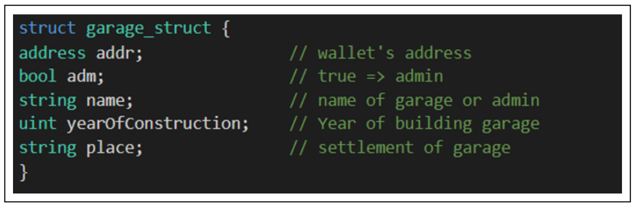
In order to store the detailed repair history of the car, each car struct has an array entry named item, which itself is also a struct (see figure 4). Each item can store one piece of repair history in the form of a string and the date when the repair history is added to the car as well as the address of the garage which adds the repair history to the car.

Similarly, a struct named `garage_struct` is created for the identity information of the garage (see figure 5). The Boolean named `adm` is used to tell if the garage is an administrator or not. The other entries are used to store the name of the garage, the year when the garage is constructed and the place where the garage is settled.

Each struct is mapped from its unique address to the data saved on the blockchain.

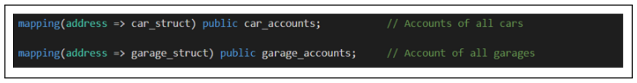
B. Car Owner functions

The owner of a car should be able to check the data and the repair history of her car as well as check the information of garages with the address of this garage. Furthermore, all these function are implemented as view functions, thus they do not cost any transaction fees. The following functions are implemented for car owners:



```
struct garage_struct {
address addr;                // wallet's address
bool adm;                    // true => admin
string name;                 // name of garage or admin
uint yearOfConstruction;     // Year of building garage
string place;                // settlement of garage
}
```

Fig. 5. Struct of garages



```
mapping(address => car_struct) public car_accounts;          // Accounts of all cars

mapping(address => garage_struct) public garage_accounts;    // Account of all garages
```

Fig. 6. Unique adress mappings

- Function `seeCarData` is implemented to check the data of a car. The input is the address of the car that the owner wants to check and the return value of the function is all kinds of data of this car such as its odometer and license plate.
- Function `getHistoryAt` is implemented to check the repair history of the car. The input is also the address of the car that the owner wants to check and an integer that indicates which piece of repair history should be accessed. The return value is the information of repair history stored in the item struct for the car with the input address.
- Function `seeGarage` is implemented to check the information of the garage. The input is the address of the garage that user wants to check and the return value is the detailed information of this garage, including if it has admin right, its name, when it is constructed and where it is located.

C. *Garage's functions*

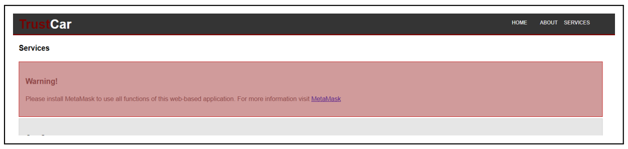
A garage should be able to do everything that the owner of a car can do. So, the above mentioned functions can be reused for garages in the front-end. In addition to that, as the core of a digitized car pass, a garage needs the option to add a new repair history to a car after finishing repairing it. Besides, as a place where car receives regular maintenance, a garage should have the possibility to update the data of the car, such as its odometer. Finally, if someone wants to sell her car, she should go to the garage to change the owner of the car and maybe also the license plate of the car. So, a garage should be able to deal with these situations. Therefore, in the smart contract, the following functions are implemented for garage:

- Function `sendHistory` is implemented for a garage to add a new repair history to a car after finishing repairing it. The functions get the address of the car as well as the repair history and then add this new repair history to the car with that address. After this repair history is added to the car, it will remain immutable and nobody has the access to change it anymore, which make it credible.
- Function `updateOdometer` is implemented for garage to update the odometer of the car when it receives regular maintenance. It will compare the new odometer with the old one to make sure that the new one is larger than the old one since the odometer of a car can never decrease.
- The functions `changeOwner` and `changePlate` are implemented to deal with the transaction of a car. These two functions will change the owner and the license plate of a car when it is sold from one person to another.

D. *Admin's functions*

An admin is a garage with administrative rights. In addition to all the functions that a normal garage has, an admin is also able to create a new car account. Since many cars already exist, an admin should also be able to register existing cars to the TrustCar system and give them unique addresses. Besides, an admin can create a new account for a garage when a new garage is build, or register an existing garage. Finally, an admin can give admin rights to a normal garage to make this garage become its down-level admin. And when an admin does not function well, its admin right can be taken away by its up-level admin. So, in the smart contract, the following functions are implemented for administrators:

- Function `createNewcar` is implemented to create a `car_struct` for a newly-created car. It takes the input to get the data of the car such as its model and owner and then use these data to initialize the value of the corresponding variables in this new `car_struct`. The value of some other variables, however, are not initialized from the input. The time when the car is created is initialized using the time provided by the system and the odometer of a new car is always initialized to 0. Also, the creation of the car is added to the repair history of the car and becomes its first repair history entry.
- Function `registerCar` is implemented to register existing cars which are created before TrustCar is released. The difference between `registerCar` and `createNewcar` is that `registerCar` needs more input data to create a `car_struct`, because an existing car has its own odometer and year of construction instead of the default values.
- Function `createNewGarage` is implemented to create a `garage_struct` for a newly-created garage. It takes all the input data and the address of the new garage to initialize the variables.
- The last two functions `changeGarageToAdmin` and `changeGarageToNoneAdmin` are implemented to give a normal garage the admin right, or take those rights away from it. It only needs the address of the garage as an input and then gives or takes the admin right to the garage with this address.



Services

Warning!

Please install MetaMask to use all functions of this web-based application. For more information visit MetaMask

Fig. 7. Warning, if web3 is not installed

IV. User Interface

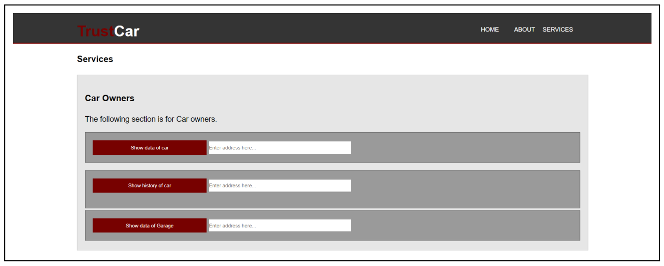
TrustCar uses an online webpage as an user interface. User can open the website with a web browser and log in with their account. But to actually use the webpage, the browser has to be able to run 'web3'. If the user did not install it before, a warning will pop up and redirect her to a webpage, on which she can install it to their browser (see figure 7).

Overall, there is one section for every different user type on the web application.

A. *Section for Owners*

The section for car owners (see figure 8) has three different components, which are 'show data of car', 'show history of car' and 'show data of garage'.

- Show data of Car:
  The car owners can type in the address of their car, click on the button and get the detailed information of the car which is stored in blockchain (no need for login).
- Show history of car:
  The car owners can type in the address of their car, click on the button and get the whole repair history of their car. Each repair history consists of the time, the detailed description and the address of the garage which added this repair history to the car (no need for login).
- Show data of garage:
  The car owner can type in the address of the garage she wants to check, click on the button and get all the detailed information of the garage, including if it has admin right, its name, when it was created and its location (no need for login).



Services

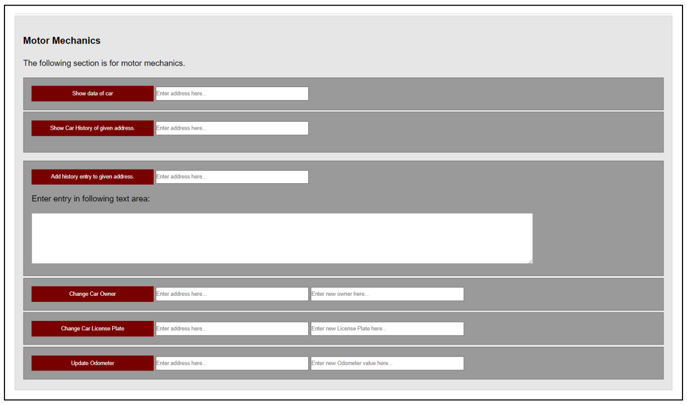
Car Owners

The following section is for Car owners.

Fig. 8. Section of Owners

B. *Section for Garages*

The section for garages (see figure 9) has six different components. The first two components are the same as in the section for car owners, which are to check the data and the repair history of the cars. The third component is to add a new repair history to a car. The remaining three components are to update the owner, the license plate and the odometer of the car separately. In order to carry out any function in the garage section the garage first needs to log in with its account address and password. This validates that the user has the correct right to perform the requested transaction.


Motor Mechanics

The following section is for motor mechanics.

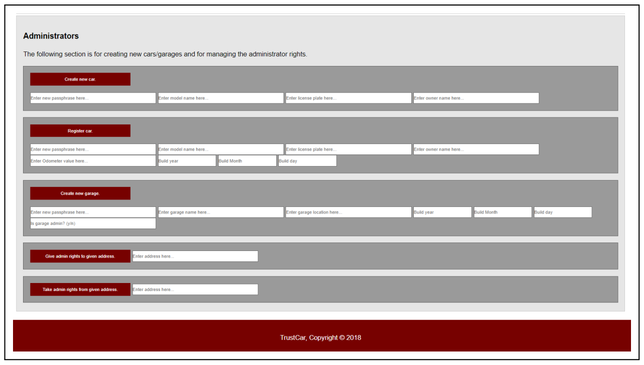
Enter entry in following text area:

Fig. 9. Section of Garages

C. *Section for Admins*

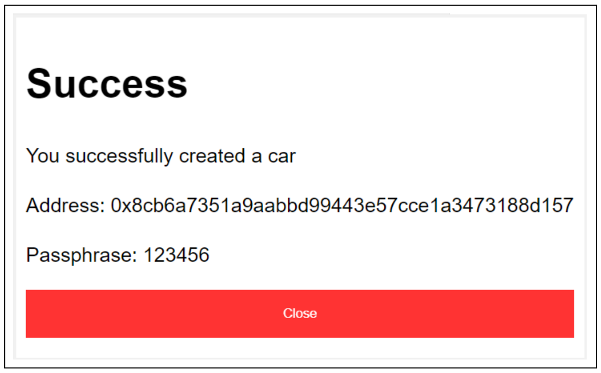
The section for admins (see figure 10) has five different components. Like a garage, an admin also needs to log in to perform all his tasks, which are not implemented as view functions.

- Create new car:
  To create a car the admin has to enter a passphrase for the new wallet . Furthermore, she has to enter the model name, license plate and the owner's name. And afterwards she needs to log in, to actually send the transaction.
- Register car:
  To register an existing car the admin has to enter the same values as for creating a new car. In addition, she needs to enter the build date and the actual odometer value.
- Create garage:
  Furthermore, the admin needs to create new garages. Therefore, she needs to give the new wallet a passphrase, a garage name and she needs to enter when it was build. Besides, the creator needs to choose if the new garage should have admin rights or not.
- Give and take admin rights:
  To perform these transactions, the user only has to enter the wallet address she would like to change and afterwards log in to make sure she has the rights to perform this action.


Administrators

The following section is for creating new cars/garages and for managing the administrator rights.
Create new car.
Enter new passphrase here...
Enter model name here...
Enter license plate here...
Enter owner name here...
Register car.
Enter new passphrase here...
Enter model name here...
Enter license plate here...
Enter owner name here...
Enter Odometer value here...
Build year
Build Month
Build day
Create new garage.
Enter new passphrase here...
Enter garage name here...
Enter garage location here...
Build year
Build Month
Build day
Is garage admin? (y/n)
Give admin rights to given address.
Enter address here...
Take admin rights from given address.
Enter address here...
TrustCar, Copyright © 2018

Fig. 10. Section of Admins



Success

You successfully created a car

Address: 0x8cb6a7351a9aabbd99443e57cce1a3473188d157

Passphrase: 123456

Fig. 11. Example of user feedback after transaction

D. General

Another interface feature which is prepared for users, is the error handling when putting in the addresses. First of all the address which was input into an address field, will be checked if it is a valid Ethereum address. If it is, then it will be checked if the address is registered in the TrustCar system or not. When some of the given circumstances are not given, the user will get a notice in the form of a small pop-up above the input line.

Moreover, after the user triggered a transaction she will get a feedback via another pop-up window (see figure 11). It will show her if the transaction was successful or not, and it will show all the main details of the transaction, for instance which task was performed for which car or garage.

V. Discussion

Although TrustCar meets all the requirements that were formulated at the beginning of the semester and is able to execute all the tasks we wished, it also has its weaknesses.

First of all, the contract was only deployed on a private blockchain because deploying it on the real Ethereum blockchain would cost real Ether and consequently real money. Moreover, every transaction would then cost our users real money. This is a disadvantage and would avoid car owners to use our application. Other blockchain technologies, such as Hyperledger, offer the possibility to create DApps without transaction fees and would therefore be a better choice for our application. However, TrustCar is the first decentralized app we ever developed and we learned how to create DApps with Ethereum in our course at the university. So, we decided to use a private Ethereum blockchain and develop a prototype of our idea using the knowledge we gained in class.

Another reason why we only created the prototype is that many companies worldwide are already working on a digital car pass. After we had our idea and started to do the first research, we found some companies on the Internet that are working on the exact same idea at the moment and will launch their application soon. For instance, Car Vertical created a blockchain based application to digitize the car pass [1] and had many similar ideas we had. Moreover, Volkswagen cooperates with IOTA and plans to already launch its digital car pass by the beginning of next year [2]. IOTA, however, uses its so-called Tangle technology instead of blockchain technologies.

Similar to our application these companies also aim to detect odometer fraud by immutably storing a car's history. However, they also consider to use the car's sensors and internal computers to collect and analyze a car's data. This would also make it possible to update the odometer value in real time. This is another weakness of our implementation. The odometer is only updated when the car is at a garage for an inspection or repair. Consequently, it is still possible to manipulate the odometer to a certain amount. As we do not have access to real time car data, there unfortunately was no possibility for us to solve this problem.

Furthermore, it might be a little cumbersome to always be required to enter the long Ethereum address. While it is needed for logging in and executing a transaction, other identification methods could be used for the cars and garages. A unique identification number could be assigned to every car and garage and then connected to the Ethereum address. However, as the database is probably very large, this ID would not be very short either. Thus, it is uncertain if this problem can be solved efficiently.

VI. Conclusion

All in all, we are really satisfied with our decentralized application TrustCar. Although it will never be used for storing real car pass data, it shows that it is a really good idea to digitize car passes and use blockchain technology to make it safe and avoid manipulation. This is proven by the fact that many large car companies are already working on it. So, we are looking forward to using a digital car pass for our own car soon.

In addition, this project aroused our interest in blockchain technologies and we believe that they can significantly change the way data is stored and processed in the future. As discussed in the previous section, Ethereum is not the best choice for many decentralized applications due to its gas fee for every transaction. Therefore, future work for our application could be to transfer the smart contract to another blockchain, such as Hyperledger.

Acknowledgment

We would like to show out greatest gratitude to Prof. James Won-Ki Hong, the instructor of the Blockchain & Cryptocurrency course, who has provided us with the valuable guidance and excellent instruction in the class. He helped us to learn a lot about blockchain and cryptocurrency, which allowed us to successfully finish our term project and design our first decentralized application.

References

[1] (2018, Dec.) Homepage of Car Vertical [Online]. Available: *https://www.carvertical.com/*

[2] Invest in Blockchain, (2018, Sep.) "IOTA and Volkswagen Expected to Launch Digital CarPass in Q1 2019". [Online]. Available: *https://www.investinblockchain.com/iota-volkswagen-digital-carpass/*