A Decentralised Sharing App running a Smart Contract on the Ethereum Blockchain

A Decentralised Sharing App running a Smart Contract on the Ethereum Blockchain

Andreas Bogner
ETH Zurich
Zurich, Switzerland
abogner@ethz.ch

Mathieu Chanson
ETH Zurich
Zurich, Switzerland
mchanson@ethz.ch

Arne Meeuw
University of St. Gallen
St. Gallen, Switzerland
arne.meeuw@unisg.ch

ABSTRACT

The sharing economy, the business of collectively using privately owned objects and services, has fuelled some of the fastest growing businesses of the past years. However, popular sharing platforms like Airbnb or Uber exhibit several drawbacks: a cumbersome sign up procedure, lack of participant privacy, overbearing terms and conditions, and significant fees for users. We demonstrate a Decentralised App (DAPP) for the sharing of everyday objects based on a smart contract on the Ethereum blockchain. This contract enables users to register and rent devices without involvement of a Trusted Third Party (TTP), disclosure of any personal information or prior sign up to the service. With increasing distribution of cryptocurrencies the use of smart contracts such as proposed in this paper has the potential to revolutionise the sharing economy.

ACM Classification Keywords

K.4.4 Electronic Commerce: Distributed commercial transactions

Author Keywords

Sharing; Sharing Economy; Blockchain; Ethereum; Smart Contract; Decentralised App (DAPP); privacy protection.

INTRODUCTION

The sharing economy has emerged as an important driver of growth in the last decade, creating some of the fastest growing *unicorns* like Airbnb, Uber or Lyft [4]. Common problems of these platforms are the need of a Trusted Third Party (TTP) (i.e. a platform operator), lack of privacy when using them and repetitive individual sign up for each platform. We propose that these problems could be resolved using Ethereum based smart contracts replacing the intermediary.

An Ethereum based smart contract is a cryptographic box which stores information, processes inputs, writes outputs and is only accessible to the outside if certain predefined conditions are met [1]. In practice, Ethereum allows for an easy implementation of such smart contracts [3]. These contracts contain code comprising a Turing complete set of operations. This code is executed by the Ethereum Network once a function of a smart contract is called. Such a computation may result in several outcomes: alteration in the state of the smart contract, returning a result and transferring monetary value.

IoT'16 November 07-09, 2016, Stuttgart, Germany

ACM ISBN 978-1-4503-4814-0/16/11...$15.00
DOI: http://dx.doi.org/10.1145/2991561.2998465

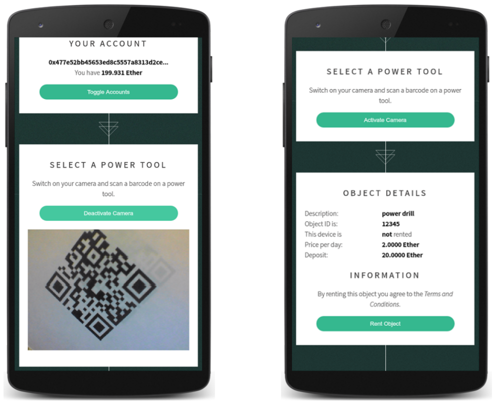
YOUR ACCOUNT
0x477e52bb45653ed8c5557a8313d2ce...
You have 199.931 Ether
Toggle Accounts
SELECT A POWER TOOL
Switch on your camera and scan a barcode on a power tool.
Deactivate Camera
SELECT A POWER TOOL
Switch on your camera and scan a barcode on a power tool.
Activate Camera
OBJECT DETAILS
Description: power drill
Object ID is: 12345
This device is not rented
Price per day: 2.0000 Ether
Deposit: 20.0000 Ether
INFORMATION
By renting this object you agree to the *Terms and Conditions*.
Rent Object

Figure 1. DAPP in use: The owner of the smart contract registers devices by scanning a barcode with a numerical id (left screen). After scanning the same barcode on a registered device, the app displays the rental conditions to the interested renter (right screen).

We demonstrate a web app (shown in Figure 1) for the sharing of objects based on a smart contract running on the Ethereum test network. Users are inherently identified by their Ethereum public key which allows for anyone in possession of such a key to participate in the app without sign up or revelation of personal financial information. All details of the rental agreement are accessible in the public Ethereum blockchain and are executed as specified, thus avoiding a TTP. The rental terms and conditions are set by the owner of an object during the registration process. Objects are identified by scanning a QR code referencing a key in the smart contract. The details of the procedure are explained in the following section.

SYSTEM ARCHITECTURE

An overview of the system architecture is shown in Figure 2. The main components are a smart contract hosted on the blockchain, the local Ethereum client, and a web app. The web app provides a graphical user interface for the (local) Ethereum client, which in turn interacts with the smart contract on the Ethereum blockchain. Objects participating in the rental scheme are labeled with a QR code that encodes a unique numerical ID for the web app.

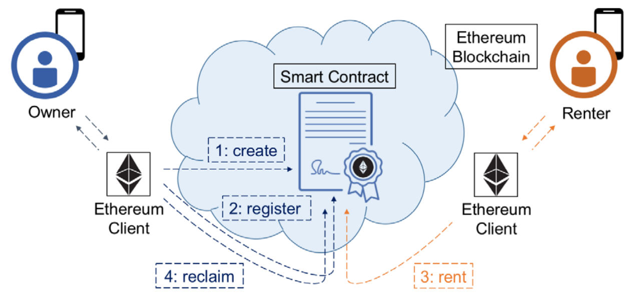

Figure 2. Schematic procedure of the DAPP with the smart contract at its core. A tool owner is allowed to register and reclaim an object while a renter is solely authorised to rent out an object of his choice.

```
function rentObject(uint _objId) returns (bool) {
  if (objectIsRented(_objId) ||
      msg.value < objects[_objId].deposit) {
    throw;
  }
  // add renter to object
  objects[_objId].renter =
    Renter({address: msg.sender, since: now});
  // send back any excess ether
  uint change = msg.value - objects[_objId].deposit;
  if (!objects[_objId].renter.address.send(change)) {
    throw;
  }
  return true;
}
```

Figure 3. The method `rentObject` in the central smart contract.

Smart Contract

The smart contract defines four core functions within the system which create the smart contract itself, register new objects, and rent devices and reclaim them (see Figure 2).

The constructor is executed by the transaction which defines the smart contract. The sender of this transaction becomes the owner of the smart contract.

The contract holds a key value store named `objects`. Calling the `registerObject` function inserts a new key value pair where the value is an object with its relevant properties (description, daily rental price, deposit due) and the key is the numerical ID contained in the QR code.

The `rentObject` function assigns a renter to an object and stores the current timestamp. This only succeeds if the corresponding transaction transfers an amount larger or equal to the object's defined deposit and whether the object is free at the time of the transaction (see Figure 3).

The `reclaimObject` function can only be triggered by the contract owner. It assigns a rented object back to its owner, calculates the rental fee (in reference to the timestamp from the previous step), and sends the rental fee to the owner and the remaining deposit (less the rental fee) back to the renter.

All functions listed above are fully deterministic and publicly available on the blockchain, eliminating the TTP. The two parties agree to these conditions when signing the respective calls to the contract. The Ethereum network guarantees that the contract is executed accordingly. In particular, the deposit is put in escrow with the smart contract and neither the owner nor the renter nor any other third party can get hold of it.

Web App

The web app written in JavaScript and HTML5 provides the user interface for the Ethereum client, which in turn interacts with the blockchain.

The registration of an object is automatically initiated when its QR code is read the first time by the DAPP. The owner is prompted to state the details of the rental agreement. On submission, the web app issues a transaction calling the `registerObject` function of the smart contract with the details provided. This contract ensures that solely the creator of the contract is authorised to register objects.

Upon scanning a registered object the user is presented with the rental terms (see Figure 1). When the user clicks "Rent object", the app triggers a transaction calling the `rentObject` function with the object ID and simultaneously transfers the required deposit to the smart contract.

The object owner analogously reclaims the object by triggering the `returnObject` function via the same web app.

The authors have built the web app using the *Truffle* development framework [2]. This framework, amongst others, generates JavaScript bindings for the smart contract and includes libraries such as *web3.js* [5] that facilitates the communication between the web app and the Ethereum client.

DEMONSTRATION SETUP

Our demonstrator shows a practical implementation of a DAPP for sharing objects. While this DAPP may be used for sharing any kind of tangible object, we implement a sharing solution for power tools. In our setup two mobile devices, used by the object owner and the renter, access the Ethereum test network.

In our smart contract demonstration, we have an owner add a cordless drill to the web app, have an additional person rent the drill, and then return the drill thereafter; thus presenting the four core functions of our smart contract.

REFERENCES

1. Vitalik Buterin. 2014. A next-generation smart contract and decentralized application platform. *white paper* (2014).
2. ConSensys. 2015. Truffle - A development framework for Ethereum. (2015). `https://github.com/ConsenSys/truffle`.
3. Kevin Delmolino, Mitchell Arnett, Ahmed E Kosba, Andrew Miller, and Elaine Shi. 2015. Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab. *IACR Cryptology ePrint Archive* 2015 (2015), 460.
4. Alberto Marchi and Ellora-Julie Parekh. 2016. How the sharing economy can make its case. *McKinsey Quarterly* 1 (2016), 112.
5. Fabian Vogelsteller. 2015. web3.js. (2015). `https://www.npmjs.com/package/web3`.