Bachelor's thesis

Information Technology

2018

Alexander Mikhaylov

SIMPLE AND SECURED ETHEREUM TRANSACTIONS

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2018 | 42

Alexander Mikhaylov

SIMPLE AND SECURED ETHEREUM TRANSACTIONS

This thesis work documents the process of developing a web application aimed to simplify the Ether (cryptocurrency) transaction process. The thesis also discusses the theoretical background for the technologies used during the development of the application.

The purpose of the thesis was to solve the problem of slow Blockchain technology integration into society due to the complexity of transaction process for ordinary people. The proposed solution is meant to simplify the process of transactions by allowing users to send ether using a phone number.

In order to implement the proposed solution, a web application was developed as a front-end point for a service that would enable users to send ether using a phone number of a receiver. The programming code of the application was written using the JavaScript React framework.

The developed solution is fully functioning and accessible. It has received positive feedback from the Ethereum community and aroused interest from companies such as Trust wallet and Branch.

The application has simplified the process of Blockchain and cryptocurrency adoption as it was aimed to. Several users stated in their feedback that this application helped them to introduce cryptocurrency to their friends and relatives.

KEYWORDS:

Cryptocurrency, Ethereum, Ether, Blockchain

CONTENTS

LIST OF ABBREVIATIONS **5**

1 CURRENT PROBLEMS OF ETHEREUM TRANSACTIONS **6**

2 ETHEREUM **8**

2.1 Blockchain 9

2.1.1 Security 9

2.1.2 Paradigm 10

2.1.3 Forks 11

2.2 Ether and Gas 13

2.3 Accounts system 13

2.3.1 Owned accounts 14

2.3.2 Smart contracts 14

2.4 Ethereum Virtual Machine 15

2.5 Merkle tree 16

3 SOLUTION **17**

3.1 Idea 17

3.2 Requirements 17

3.3 Verified proxy 18

4 IMPLEMENTATION **19**

4.1 Back-end work flow 19

Sending 19

Receiving 20

4.2 Security 22

4.2.1 Send transfer function code 23

4.3 Front-end workflow 24

Sending 24

Receiving 29

4.4 Timeline 35

4.5 Code structure 36

4.5.1 Eth2phone 36

4.5.2 Eth2phone-server 36
4.6 Testing 38

5 CURRENT PROGRESS AND PLANS 39
5.1 Current state of the project 39
5.2 Significance and validity 39
5.3 Future plans 41

REFERENCES 42

FIGURES

Figure 1. State transition. 11
Figure 2. Forks. 11
Figure 3. Merkle tree illustrates the structure of Merkle tree. 16
Figure 4. Sending transaction function code. 23

PICTURES

Picture 1. Sending illustrates the process of sending ether: 20
Picture 2. Receiving illustrates the process of receiving ether: 21
Picture 3. Send screen. 25
Picture 4. Send screen error example. 26
Picture 5. Sent transfer progress. 27
Picture 6. Sent transfer error. 28
Picture 7. Receiving screen. 29
Picture 8. Receiving SMS verification. 30
Picture 9. Receiving transfer progress. 31
Picture 10. Receiving transfer completed. 32
Picture 11. Transactions history screen. 33
Picture 12. Timeline. 35

LIST OF ABBREVIATIONS

BC	Blockchain
Eth	Ether
EVM	Ethereum Virtual Machine

1 CURRENT PROBLEMS OF ETHEREUM TRANSACTIONS

Nowadays cryptocurrency, such as Bitcoin, is gaining more and more popularity among people and we cannot deny its role in the world economy. Although cryptocurrency does not replace fiat currency in the nearest future, it will take a significant part of the market. Despite the fact that it does not have any special regulations in most countries yet, there are already ways people can use cryptocurrency. While crypto markets have been banned in China, Japan, on the contrary, officially allows people to pay for different services or goods with Bitcoin.

At the moment, there is a vast number of different cryptocurrencies but this thesis describes the service which works with Ether (Eth) on its blockchain – Ethereum. Unlike Bitcoin, Ethereum blockchain is not just a supportive tool for its cryptocurrency, it is a powerful decentralized platform that brings several opportunities for development: new markets, debt, registry and other services. This makes Ethereum an ideal replacement for known payment methods.

Nevertheless, the current method of transferring eth is still difficult for ordinary people and demands some technical background. This is one of the current problems of the Ethereum platform. The process of creating a transaction is too complicated at the moment, therefore, most people do not understand or are afraid of using it. That slows down the process of eth integration into society and makes it less attractive comparing to the ordinary payment systems.

Moreover, there is another important problem that needs to be taken into consideration – the risk of assets loss. In order to create a transaction, the sender should know the public address, a 42 characters long hexadecimal string, of the receiver. If there is a mistake in any of the characters, the money will be sent to a wrong address and cannot be retrieved. Since blockchain network does not have any centralized management system, there are no instances that can help to restore the lost funds.

After these two problems were considered by the development team, it was decided to create a service – eth2phone - which allows making ether transactions using a phone number instead of a public address. At the same time, the sent money can be retrieved if it has not been received, even if the sender makes a mistake in the phone number.

This thesis concentrates on the description of back-end and front-end processes as well as blockchain technology details. Thesis work documents theoretical background and the development flow: website requirements, implementation and testing.

2 ETHEREUM

The platform the service works on is called Ethereum [1]. It can be described as a powerful decentralized virtual machine which runs different applications – smart contracts – using custom blockchain. These applications run as programmed and cannot be censored, shut down, frauded or interfered. Created by Vitalik Buterin in 2015 this open-source platform helps to ease the process of Blockchain (BC) technology integration. Therefore, it attracts interest from different startups as well as large companies such as Microsoft and IBM.

The Ethereum platform can be used in different fields of business or finance. It guarantees security and prevents any intrusion into system. Companies and services based on Ethereum can do business with other companies and services they do not know without risk of fraud. Ethereum allows to register any type of trades with any asset without need to use judicial proceedings. This makes it preferable compared to the current methods of trade registration.

2.1 Blockchain

Blockchain (BC), firstly created as a supportive tool for Bitcoin, has become an independent technology used not only in the cryptocurrency field. The process behind BC is similar to torrent peer-to-peer networks. BC represents a distributed ledger which contains ”account balances, reputations, trust arrangements, data pertaining to information of the physical world; in short, anything that can currently be represented by a computer is admissible” – as stated in “Ethereum Yellow Paper” [2]. The ledger consists of blocks containing a copy of the information about transactions or contracts, and it is constantly updated. Each block contains a cryptographic hash linked to the previous block so they create a chain of blocks – Blockchain. Data hosted on Blockchain can be presented as a shared database which is stored not only in one location, but on the million computers at the same time. No single center makes it almost impossible to compromise information stored on BC.

There are two main types of BC: public and private. The key concepts of them are similar but the main difference is that public BCs are completely open, while in order to participate in a private BC, the user needs either to have a permission from the creator or to be validated by the set of special rules. Both of them have benefits and disadvantages as well as different areas to be used in. For instance, public BCs are more secured but need more computational power to maintain at a large scale. Private BCs are usually business-oriented and do not grant access to the information to non-participants of the network. At the same time, the smaller size of such networks makes them more vulnerable to hacker attacks. The following section further examines BC security.

2.1.1 Security

This subsection explains the main concepts that lay behind BC security. There are two of them: the unique cryptographic fingerprint for each block and the ”consensus protocol”.

Each block is signed with a unique cryptographic hash. In order to generate that hash computing power and time of the miner (a miner is a special node of the BC network which gives it computational power to the network) device is used. The miner encrypts

the block with the generated hash and adds it to the network. This hash represents a guarantee that the block is valid. Hash generation requires a large amount of computational power which is used to solve a mathematical problem. Anyone can try to generate the hash, but only the fastest miner wins the contest. After that, nodes validate the generated hash of a new block, and this block is added to the network. The validation process is, on contrary, easy comparing to the hash generation, and does not take much power and time. As soon as a new block is added, as a "proof of work", the miner that generated the hash for this block is awarded with a digital token used in the particular network (Bitcoin, Ether, etc.). The process documented above is called the consensus protocol.

When the miner encrypts the block with the hash, this hash seals the block. In order to break the seal, it is required to generate a new hash. Therefore, to change a single record it is needed to change the hash of the whole block. However, the hash also contains the links to the neighbor blocks. This means that to change the block's hash, it is needed to change the hashes of the subsequent blocks as well.

In addition, each record inside the block is secured by cryptography. Every participant of the network owns a private key, which is used to sign transactions. In case the record is altered, the signature becomes invalid. When this happens, the peer network is alerted that something unusual has happened. This allows to locate the intrusion at an early stage and prevent further damage.

Since BC is decentralized and distributed across peer-to-peer network that is constantly updated, BC does not have a single point of failure. In other words, there is no central location of BC, therefore, to hack such network it is required to gain access to its every instance, or at least to a 51 percent majority of the network [3]. This means that the bigger network is, the less vulnerable it is. That is why private BCs are more vulnerable compared to the public ones.

2.1.2 Paradigm

Ethereum BC can be described as a transaction-based state machine [4]. Everything begins with the "genesis" – blank – state, which is empty and does not contain any transactions in it. Whenever transactions are added to the state, a transition to a new state happens:

new state = old state + new transactions

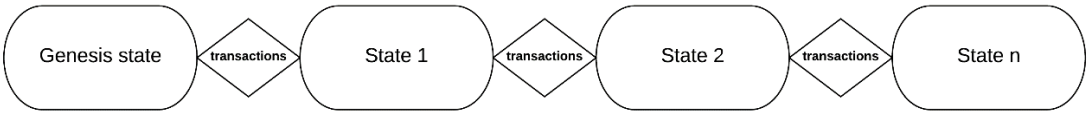


Figure 1. State transition.

The figure above illustrates the state transition process. The new state is the final state of the blockchain and the single global truth for all the participants of the network. Transactions thus represent a valid arc between two states [5].

As mentioned previously, new transactions can be added to the state only in case they are valid. The validation process (mining) is executed by the group of nodes which are using their computer resources to mathematically validate the transactions and create a new block of the chain. A valid block can be created by any miner and the fastest created block is added to the chain. On the block submission, the miner presents the mathematical “proof” which guarantees that the block is valid. This raises the question: how do miners stick to the same chain of blocks?

2.1.3 Forks

Since there is only one true state that everyone accepts, there cannot be multiple chains. Otherwise, different chains can have different states, and that leads to chaos. However, sometimes multiple states are generated and it is called “fork” (Figure 2).

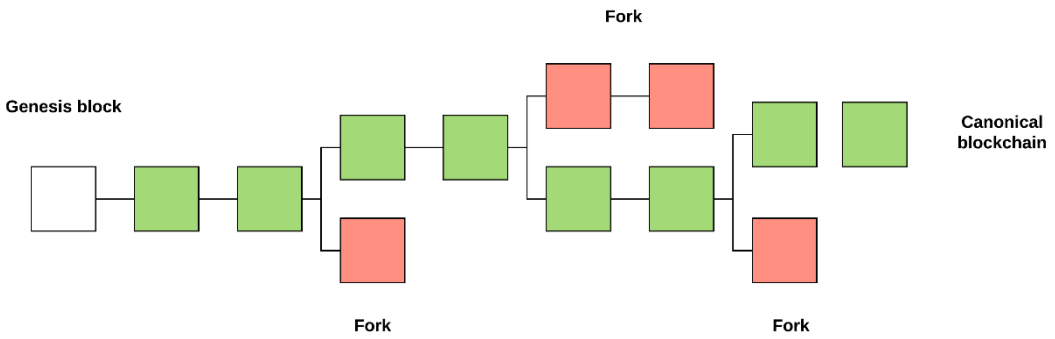


Figure 2. Forks.

When a fork occurs, the nodes of the network must decide which path is the valid one. The decision is made using the “GHOST protocol”:

“GHOST” stands for “Greedy Heaviest Observed Subtree”.

In other words, the “UNCLE” (a stale block) with the most computations made is determined as the valid one. Ethereum implements a simple version of GHOST protocol that has seven levels of validation:

- The block must contain information about its parent and children.
- It must be a direct child of its ancestor less than seven blocks below.
- It cannot be the direct ancestor of a new block.
- It must have a valid header.
- An UNCLE cannot be equal to other UNCLEs in previous blocks and the new block.
- The miner that added the UNCLE block receives an additional 3.125% for each block of the UNCLE, but receives 93.75% of a block reward.

2.2 Ether and Gas

Ether is a cryptocurrency used by Ethereum platform in order to pay for the gas. Ether is generated whenever a new valid block is added to the network and is awarded to the node that created this block. Gas is a unit of computation which is used for all Ethereum state transition such as transactions and computational services.

In order to execute a transaction, a user needs to set a gas limit and a gas price. The gas price equals to the amount of ether user wants to pay per gas unit. The price of gas is measured in Gwei, which is 1 000 000 000 Wei. Wei is the smallest unit used in Ethereum and 1 Eth equals 1 000 000 000 000 000 000 Wei.

Gas limit is the maximum amount of Wei user wants to spend on transaction. For instance, in order to execute a transaction with 21000 gas limit and 10 Gwei of gas price, user needs to pay 21 000 * 10 Gwei = 210 000 Gwei (0.00021 Eth).

2.3 Accounts system

There are two types of accounts: owned accounts and smart contracts. Both of them contain the following information:

Balance – the amount of Wei available on the account

Nonce – the amount of transactions sent from the owned account, or, if it is a smart contract, the amount of contracts created by it.

storageRoot – a 256-bit hash of the root node of a Merkle Patricia tree (this tree encodes the storage contents of the account)

codeHash – the hash of the EVM (Ethereum Virtual Machine) code of this account. This code is executed when the account is asked for a permission to be modified (for instance, receive a transaction)

2.3.1 Owned accounts

When an owned Ethereum account is created, a public-private key pair is generated. The address of the account is represented as last 20 bytes of the public key. This address is used for tokens transferring and association with the account. A private key is used to control the account and create transactions. It is always stored in encrypted form – keystore. When a transaction is created, it is signed with the private key of the owner. The private key is needed during the validation process in order to confirm that the sender owns the account.

2.3.2 Smart contracts

An Ethereum smart contract is also an account that can hold ether, but also it has a code part that is executed by the EVM (Ethereum Virtual Machine) under certain conditions. This code part can be described as a programming class containing different functions. When tokens are transferred into a smart contract, it can automatically choose an action to execute: transfer money to one person or another, to make a refund or do something else.

Smart contracts are written using different programming languages such as Solidity, Serpent, LLL, etc. Smart contracts are created by the programmers and deployed on Ethereum BC.

Since smart contracts are also located on BC, they are immutable. This means that there is no human intervention during the code execution. Moreover, as well as ordinary accounts, smart contracts are located in the blocks. Therefore, they cannot be modified without consent of all the nodes they are linked to. BC security rules are also applied to smart contracts, and, as it was mentioned in the BC Security chapter, it is almost impossible to tamper a smart contract without the owner access.

2.4 Ethereum Virtual Machine

The EVM is a runtime environment used by Ethereum blockchain for smart contract execution. It is isolated from the main blockchain, therefore, it does not have access to the network, accounts or processes, and has a limited access to smart contracts.

Whenever code is executed or transaction is created by the EVM, a particular amount of gas is used. The amount of gas and the gas price user defines on the transaction creation are needed to limit the amount of computational power EVM should use in order to process this transaction. Therefore, the more gas user uses, the faster created transaction is executed.

2.5 Merkle tree

Global Ethereum state represents the mapping between accounts states and their addresses. The mapping has a structure of a Merkle Patricia tree.

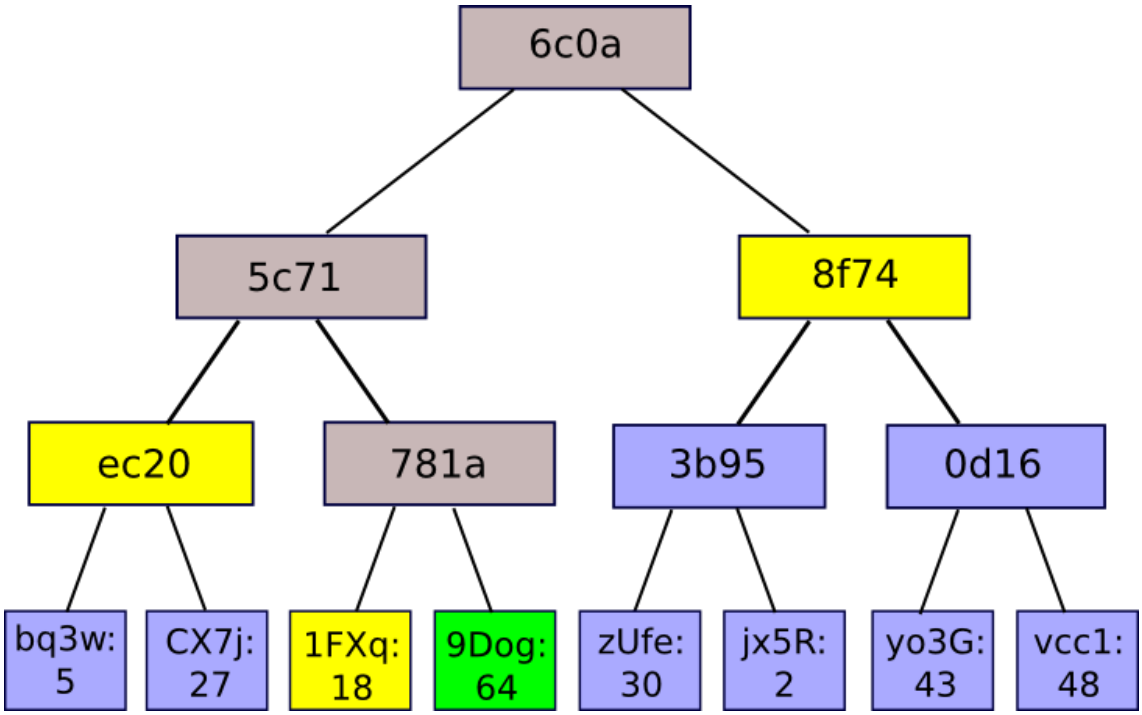


Figure 3. Merkle tree illustrates the structure of Merkle tree.

The root hash of the tree consists of three things: the chunk, the root hash and the “branch” of the tree. Branch contains all the hashes and create a path from chunk to the root. Every value stored in the tree is required to have a unique key which contains a link to the neighbor blocks. When it is needed to check information about a particular value, this key tells which path to follow to find the right block [6].

3 SOLUTION

This chapter describes the solution that has been developed in order to solve the problems mentioned in the first part. The first part explains the idea that lays behind the project. The second part of the chapter defines the technical requirements for the idea implementation. The last part contains the definition of the solution created based on the idea and requirements.

3.1 Idea

After analysis of the current Ethereum transactions problems my team has developed an idea of creating a tool which can ease the process of the transactions for the new users. At the same time making the transaction secured.

It was decided to create a web application with an opportunity to send ether directly to a phone number of a recipient with no need to use a long and user unfriendly public address of the wallet.

In addition, user gains an opportunity to cancel a transaction sent using this web application before the tokens have been received. The whole process must be secured so, in case of hack, the stolen information cannot be used to receive the transferred tokens.

3.2 Requirements

In order to implement the project three objects have been developed: a web application, a server and a smart contract.

The web application written in React, a JavaScript framework, provides a front-end for the user. It manages all the inputs of the user, generates the code for a recipient and passes information to the server.

A Node JS server is used for SMS user verification. It also communicates with smart contract in order to withdraw tokens to a receiver’s address.

Smart Contract is written in the Solidity language. This language is used for writing smart contract for the Ethereum network.

3.3 Verified proxy

The solution can be called as verified proxy. Proxy here means that a smart contract is used as an intermediate point during a transaction. The verification of a receiver is handled by the server which is also used as a transfer storage for the keystore data.

4 IMPLEMENTATION

This chapter explicitly documents the implementation process of the project. The back-end part explains the methods and processes performed in the background. While the front-end section describes and demonstrates how the application works from the user's point of view. In addition, the testing methods and code structure are provided.

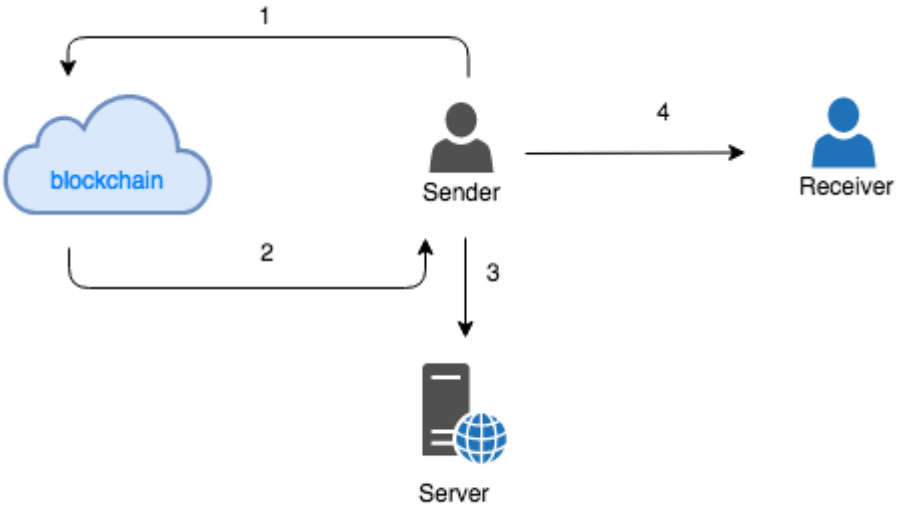
The timeline illustrates what happens on each step of the application use. The security part shows the procedures used to protect the transferred assets.

4.1 Back-end work flow

The back-end work flow can be divided into two parts: sending and receiving.

Sending

First of all, a private-public key pair is generated on the device of the sender. It is needed for signing a transaction in order to receive the tokens. After that, ether is deposited to the smart contract, and the public key of the generated pair is assigned to the deposit. Then the private key of the pair is used to create a keystore with a randomly created secret code. The secret code is passed by the sender to a recipient, the keystore – to the server.

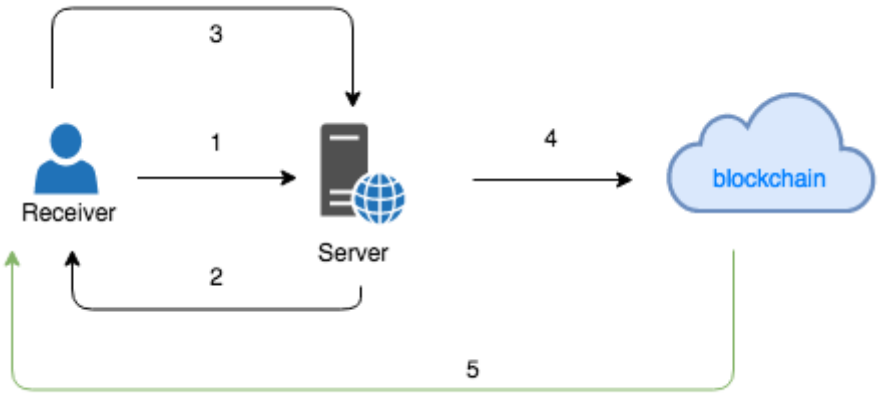


Picture 1. Sending illustrates the process of sending ether:

1. The sender generates a transit private-public key pair.
2. The sender deposits ether to the escrow smart contract and assigns the transit public key to the deposit. On withdrawal the escrow smart-contract verifies that receiver's address is signed by the transit private key.
3. The sender encrypts the transit private key with a random secret code and sends the encrypted transit private key to the verification server.
4. The sender passes the secret code to a receiver by the way he chooses (voice, SMS, e-mail, etc.)

Receiving

In order to receive the tokens a recipient needs to unlock the private key of the transit pair. User receives the keystore – the locked private key of the early generated pair – from the server after he or she was verified by SMS. The secret code to unlock the keystore is passed to the receiver by the sender. After unlocking the private key, the wallet address of the receiver is signed by that private key and sent back to the server. Then server sends a request to the smart contract to withdraw ether. If everything is correct, the ether is transferred to the receiver’s address.



Picture 2. Receiving illustrates the process of receiving ether:

1. The receiver types in his or her phone number and the secret code. The hashed phone verification request is sent to the server. (So not at any point in time the verification server has access to the verification private key.)
2. The server sends the verification code via SMS to the phone entered.
3. The receiver gets the code by SMS and types it in. If the code is correct, the server returns the encrypted keystore data to the receiver.
4. The receiver decrypts the keystore data with the secret code provided by the sender and gets the verification private key. The receiver signs the address of his or her choice with the verification private key. The receiver sends the signed address to the verification server.
5. The verification server tries to withdraw the ether through the smart-contract to the signed address. If the signature is correct, the transaction is executed and the receiver gets the ether.

4.2 Security

The escrow smart contract deployed on the Ethereum network cannot be hacked due to the high security level of blockchain (documented in Blockchain Security chapter). Therefore, it is a safe place for storing tokens during the transfer. Since all Ether transfers are performed on BC, they are trustless as well.

Despite a centralized server is used for the phone verification and keystore transferring, at any step of the transfer process the Verification Server has no control over the ether locked in the escrow smart Contract. Even if the verification server is compromised, in the worst case scenario, the receiver will not be able to receive the transfer. At the same time, the sender can cancel the transfer with a call to escrow smart contract and get the ether back.

In case the share link is stolen, ether cannot be received without phone verification: the attacker should get the transit keystore from the verification server in order to withdraw ether. In additon, the verification server does not store raw phone numbers - instead of that the hashed form generated from the phone number and salt is used:

phoneHash = hash(phone, salt)

Salt is generated on the sender device:

salt = hash(transferId, code)

transferId = hash(phone, secret code)

This allows to eliminate the association of phone numbers and Ethereum addresses.

4.2.1 Send transfer function code

The following function is used for sending transactions. Comments simplify the understanding of the processes:

```
export const sendTransfer = async ({phoneCode, phone, amountToPay, senderAddress}) => {

    // 1. generate transit keystore, with private key encrypted with random code
    const { address: transitAddress,
            keystore: transitKeystore, secretCode } = generateKeystoreWithSecret(); // verification keystore

    // 2. register transfer to Verification Server
    const transferId = generateTransferId(phoneCode, phone, secretCode);

    // salt hash in order to prevent bruteforce attack in the case if server's database is stolen
    const salt = sha3(secretCode, phone);
    const phoneHash = sha3(phone, transferId, salt);


    const result = await verificationServer.registerTransfer({
        transferId,
        phoneHash,
        transitAddress,
        transitKeystore,
        senderAddress,
        amount: amountToPay
    });

    // if server error interrupt execution and don't send deposit to smart-contract
    if (!result || !result.success) {
        const errorMsg = result.errorMsg || "Server error!";
        throw new Error(errorMsg);
    }

    // 3. send deposit to smart contract
    console.log({result, amountToPay, transitAddress, contract: escrowContract.getContractAddress()});
    const txHash = await escrowContract.deposit(transitAddress, amountToPay);
    return { txHash, secretCode, transferId, transitAddress };
}
```

Figure 4. Sending transaction function code.

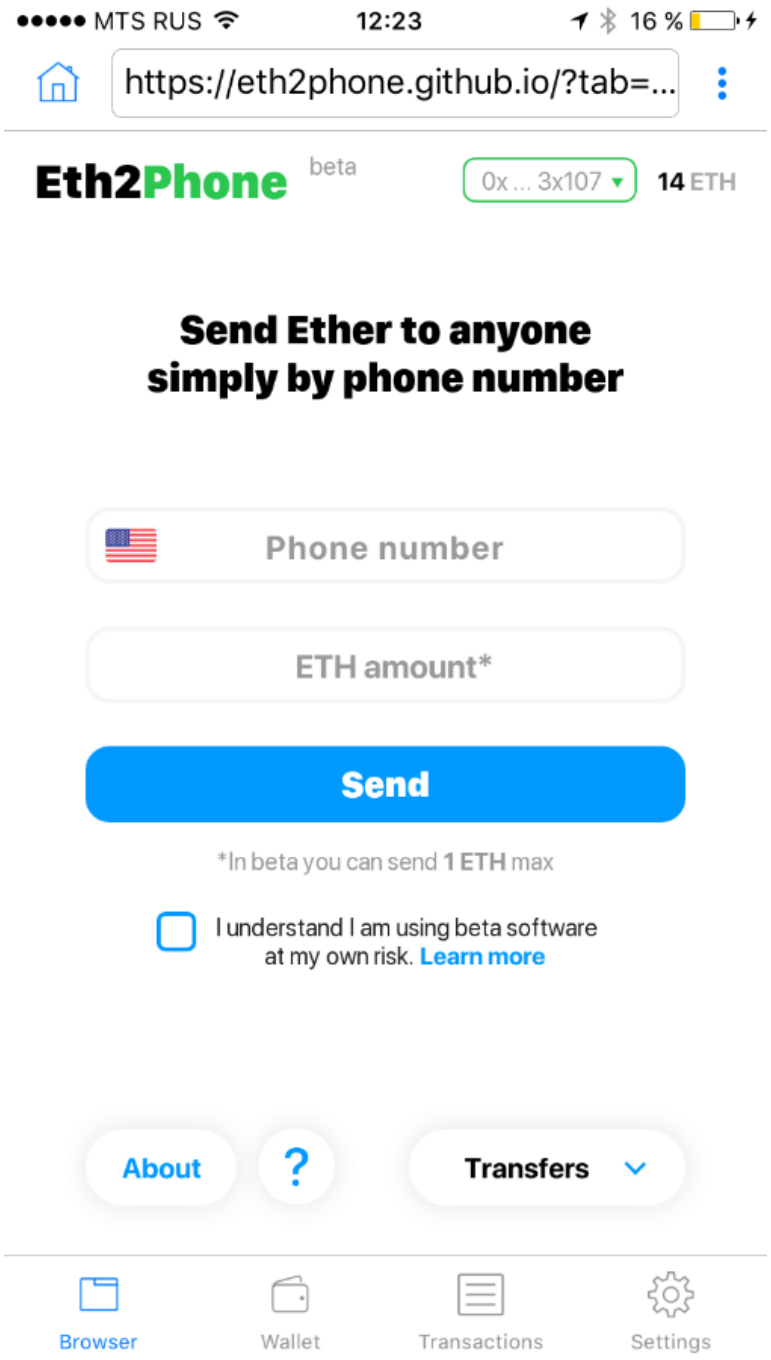
4.3 Front-end workflow

The front-end workflow can be also divided into two parts.

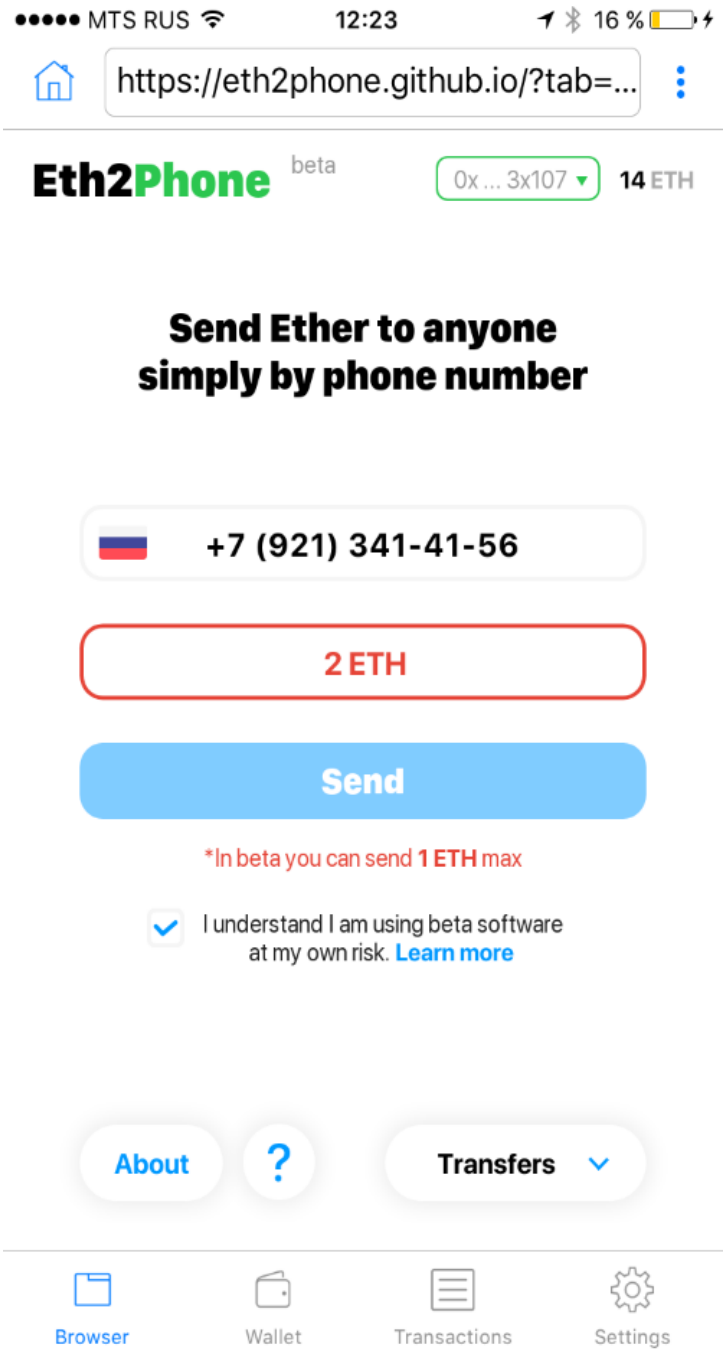
Sending

In order to use the web application user needs a web3 compatible wallet to be installed: this can be Metamask browser extension or any Ethereum wallet with a built-in web browser.

From the user point of view the process of sending and receiving is simple and intuitive. The first screen of the application has two input fields for the telephone number of the recipient and the amount of ether to be sent (Picture 3). In case of the zero input a warning error shown (Picture 4).

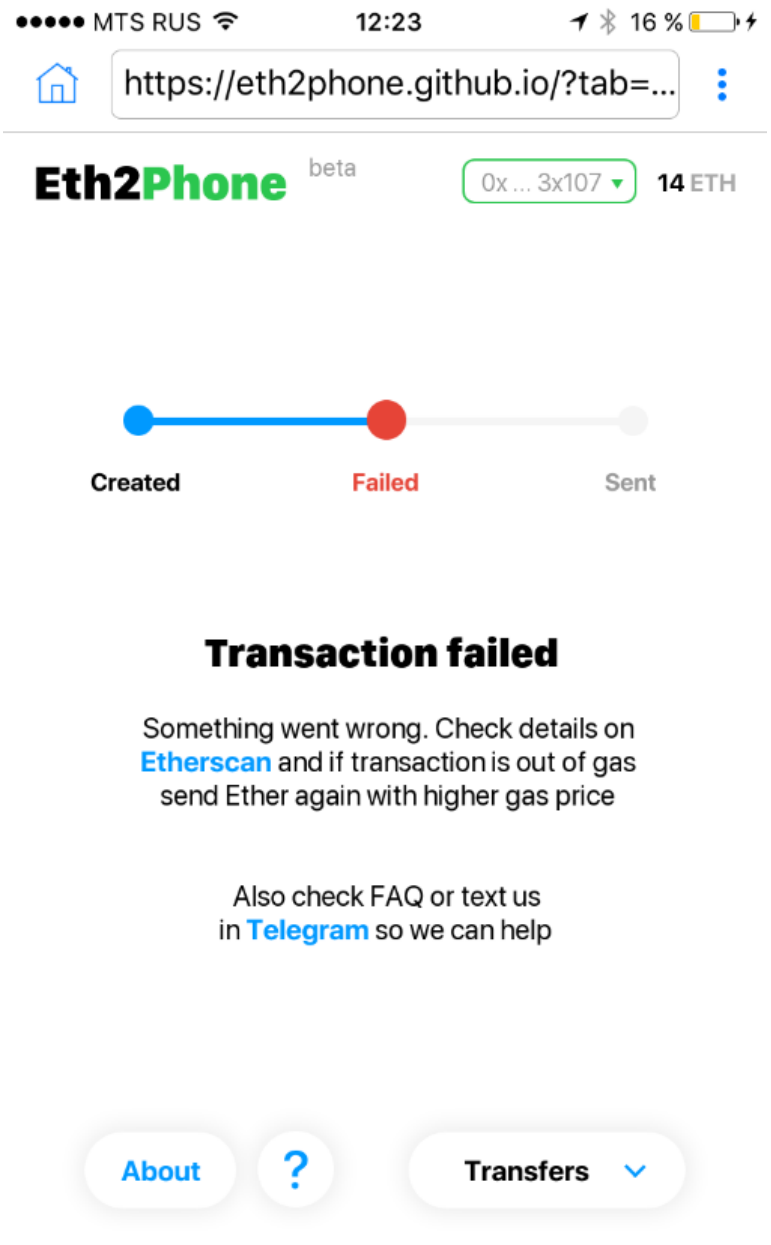


Picture 3. Send screen.

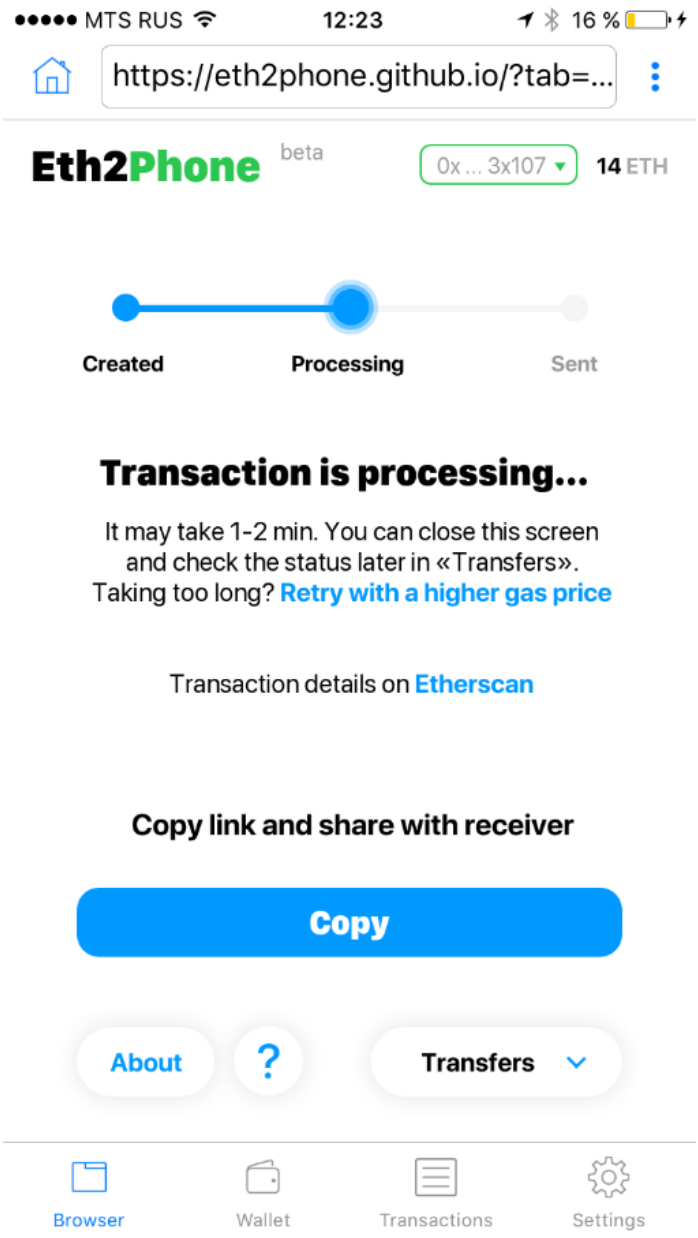


Picture 4. Send screen error example.

After clicking on the ”Send” button user needs to confirm the generated transaction in the wallet or in Metamask browser extension. While the transaction is sent to the blockchain to be confirmed, the transaction step bar screen shows the steps of this process (Picture 5). User can also check the transaction process on etherscan [7]. In case of a transaction error the step bar shows that it has not been processed (Picture 6).



Picture 5. Sent transfer progress.

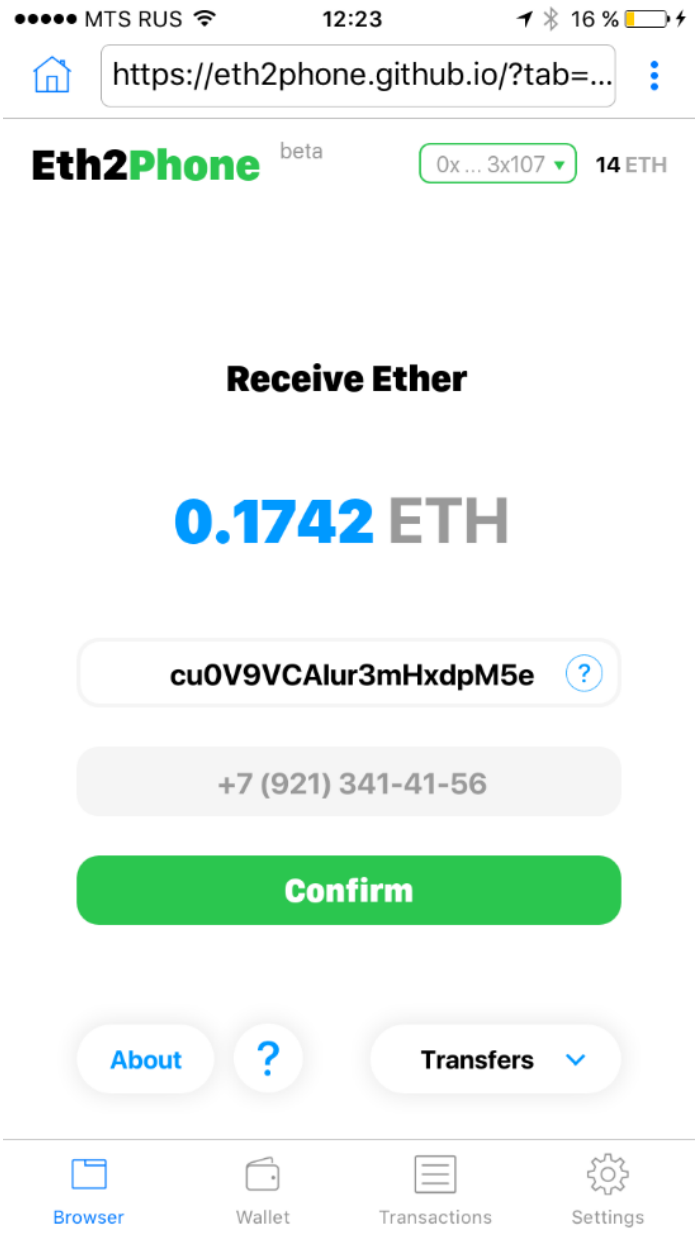


Picture 6. Sent transfer error.

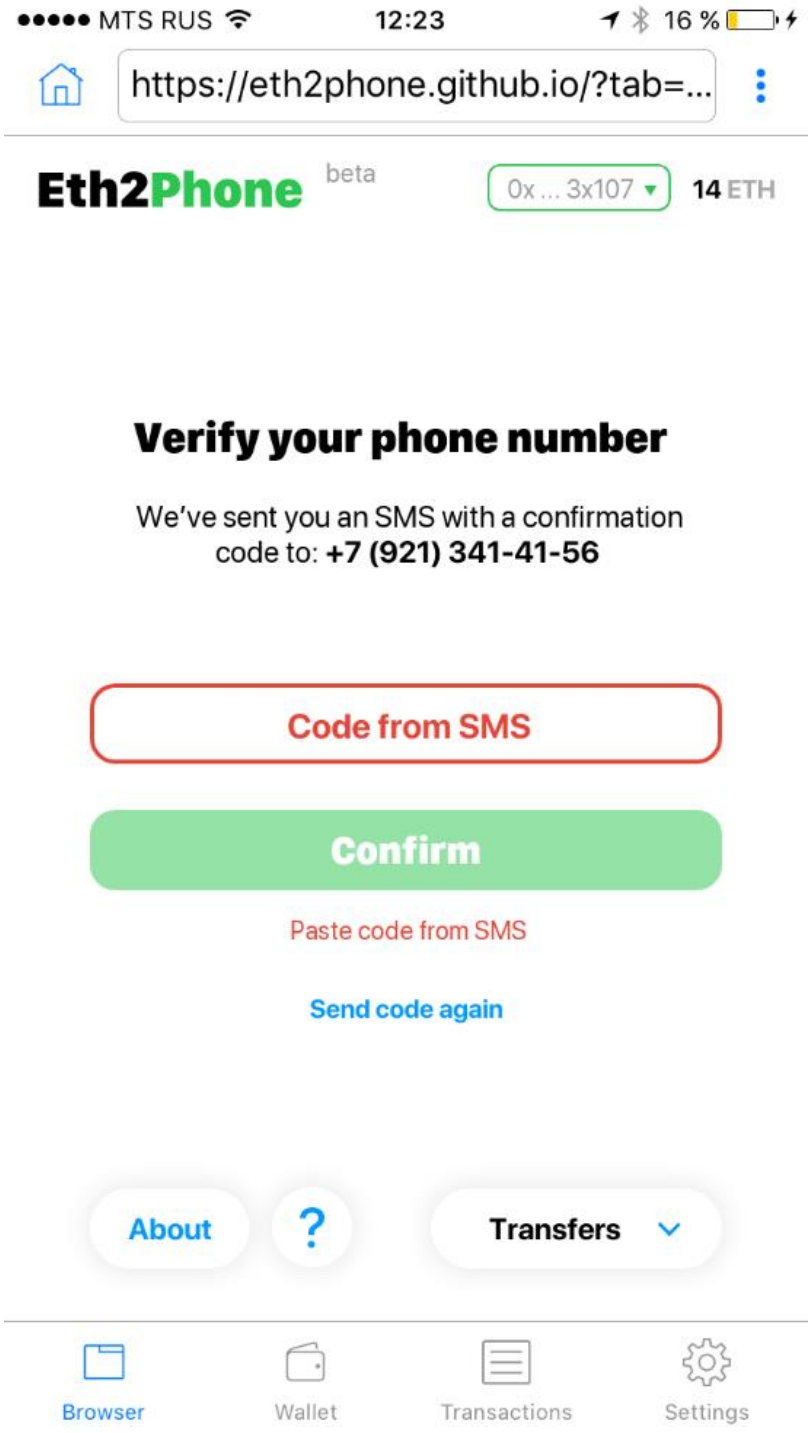
Receiving

When the transaction is confirmed a receiving link is generated and shown on the screen. The user needs to send this link to a person that he or she wants to receive the tokens.

After clicking on that link, a receiver is redirected to the receiving screen (Picture 7). The user needs to paste a received SMS confirmation code into the input field (Picture 8).

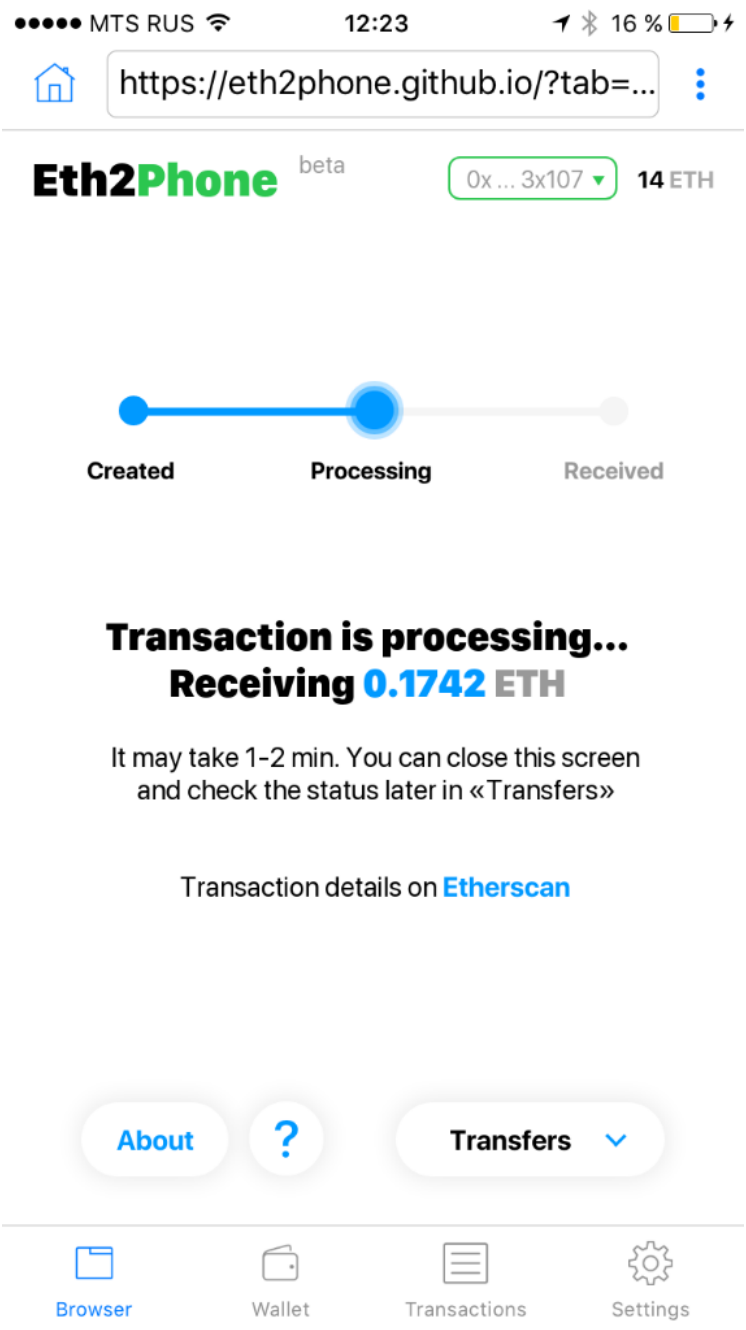


Picture 7. Receiving screen.

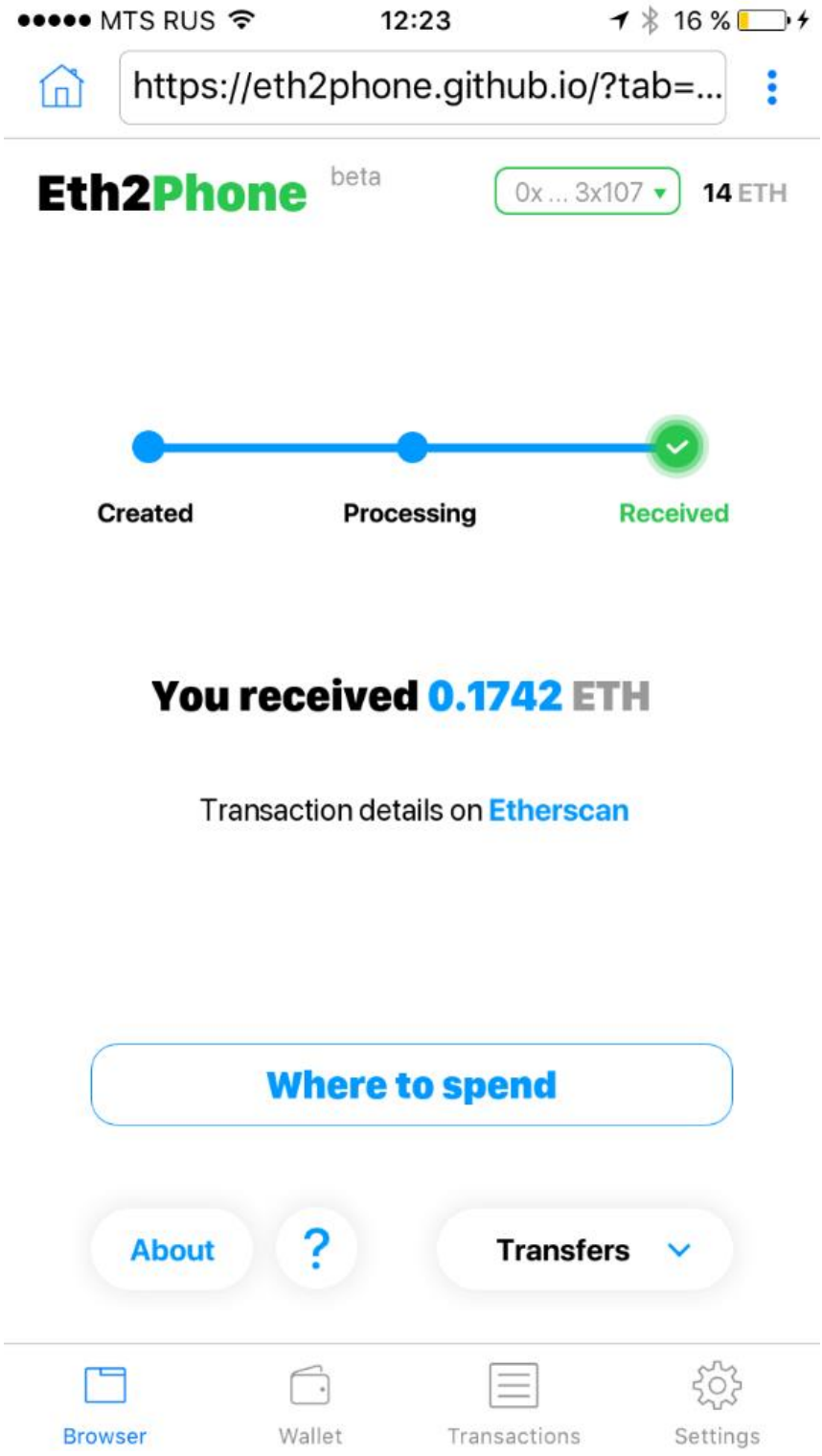


Picture 8. Receiving SMS verification.

After the user's phone number has been verified a transfer steps bar screen shows the process of receiving (Pictures 9 and 10). The transaction process can also been checked on etherscan.

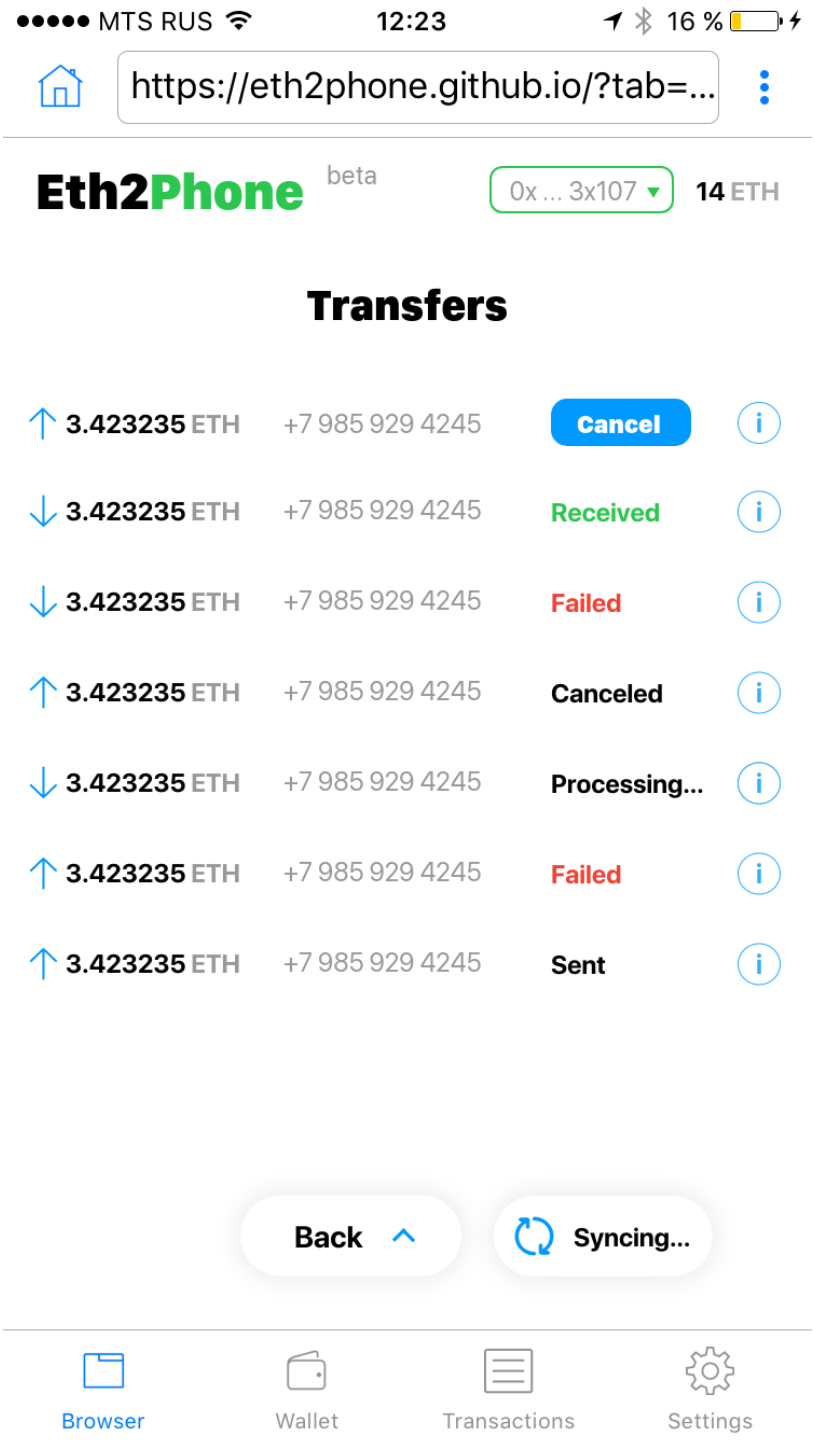


Picture 9. Receiving transfer progress.



Picture 10. Receiving transfer completed.

Transactions history screen contains the list of sent and received transactions (Picture 11). The transactions that have not been received yet can be canceled on this screen too.

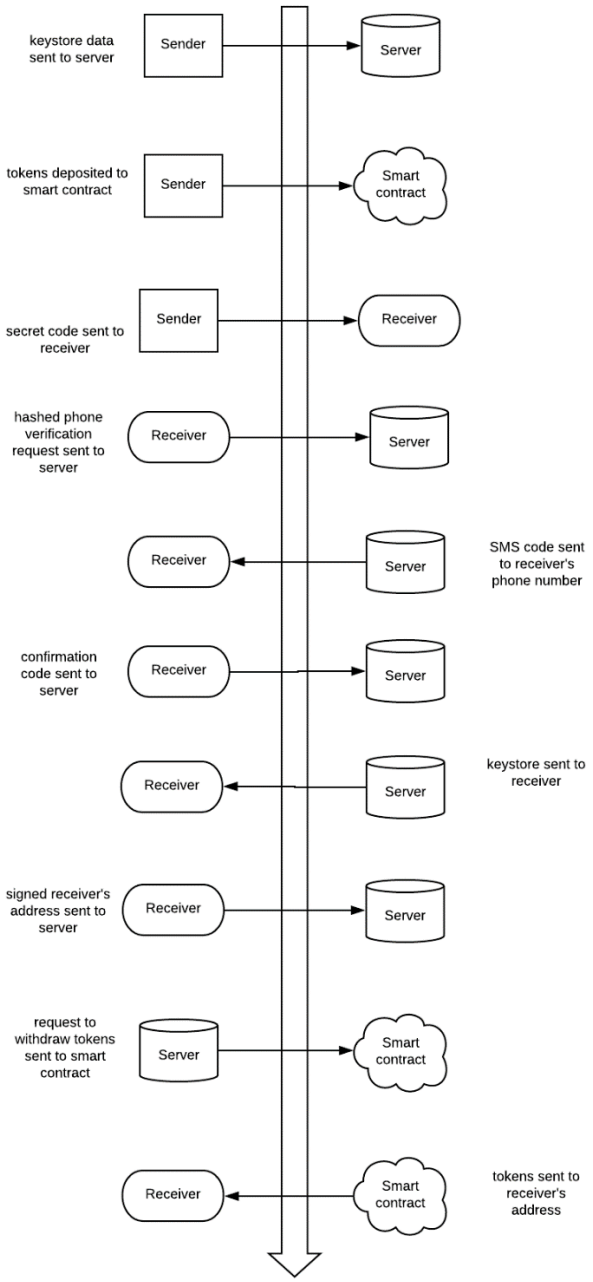


Picture 11. Transactions history screen.

In addition, the web application contains pages for landing, FAQ, Terms of Use and Privacy Policy.

4.4 Timeline

The picture below (Picture 12) illustrates the timeline of the processes happening in the background during sending and receiving operations.



Picture 12. Timeline.

4.5 Code structure

Code of the application is completely open and hosted on Git hub. There are two repositories: eth2phone and eth2phone-server.

4.5.1 Eth2phone

The repository contains smart contract code, tests and utilities for interaction with the server. The contents of the repository include:

./src - all JavaScript/React code is located in this folder.

./src/services/eth2phone/index - all interaction of the web application with the escrow smart contract and the verification server is defined in this file.

./contracts/e2pEscrow - the escrow smart contract, which locks ether from the sender and withdraws ether on the request of the owner to address signed by the transit private key. The verification server deploys the contract and controls the smart contract's owner account.

./test - tests for the VerificationProxy smart contract

4.5.2 Eth2phone-server

The repository contains the code of the verification server and the front-end for the web application. The contents of the repository:

./src/controllers/SenderController - the controller for handling sender's requests

./src/controllers/ReceiverController - the controller for handling receiver's requests

./src/services/TwilioService - the service for SMS authentication via Twilio

./src/services/TransferService - the service for handling interaction with Transfer models stored in MongoDB

./src/services/EscrowContratService – the service for handling interaction with Ethereum blockchain via e2pEscrow Contract.

4.6 Testing

The testing phase was performed on the Ropsten test network. It simulates the main Ethereum network but there is no need to pay the fee for the gas. This allows developers to test their smart contracts and create transactions for free. In addition, test cases were written in order to test the code execution.

5 CURRENT PROGRESS AND PLANS

The initial goal of the thesis was to create an application aimed to simplify the process of ether transfer for ordinary people, thus increase the popularity of BC technology and Ethereum platform. After that, it was required to document the process of development and provide information about the technologies used in the project.

At the current stage the practical part of the project can be considered as successfully implemented. It has already received positive feedback and improvement ideas from the Ethereum community. The written part of the thesis has all the required information: detailed theoretical background and documented the implementation phase.

5.1 Current state of the project

At the current moment (June 2018) web application is fully functional and can be reached using https://eth2.io address. The recent redesign has been successfully implemented. The service works in two modes: main (Ethereum main network) and test (Ropsten network).

In order to use the service, the user must have a web3 compatible wallet (the list of supported wallets can be found on the website). Trust wallet [8] is recommended: this wallet has collaborated with the eth2phone service to make the process of tokens receiving easier for new users.

5.2 Significance and validity

This project brings a small UX fix into the current transaction process but the significance of that fix for Ethereum adoption is very valuable. With the help of eth2phone application, users can send and receive ether using just a phone number. In addition, they have an opportunity to cancel the transaction, which has not been implemented by any other services yet.

The project has already attracted interest from Trust wallet and Branch [9]. They believe that eth2phone may become a tool for making crypto onboarding less difficult. In addition, service can be used for sending distributed ICO tokens and items in games based on

blockchain. Such solutions can help to promote blockchain technology and accelerate the tokenization of the society.

5.3 Future plans

The next step of the project is to create different options for the service which will allow to send eth by email address, twitter and other social network accounts. In addition to ether, next version of the application will include ERC20 tokens transferring by the phone and other methods.

It is also planned to create an independent mobile application for the service. Steps forward this direction have already been made: another team's project is an Ethereum wallet. At the current stage, a watch wallet version has been released (Quid wallet [10]) for Android and iOS. It has not been decided yet whether eth2phone should be a part of this wallet or a standalone application.

REFERENCES

[1] Ethereum platform website, https://www.ethereum.org, June 2018

[2] “The ‘Yellow Paper’: Ethereum's formal specification”,

https://github.com/ethereum/yellowpaper, cited June 2018

[3] Curtis Miles, “Blockchain security: What keeps your transaction data safe?”, published on IBM (https://www.ibm.com/blogs/blockchain/2017/12/blockchain-security-what-keeps-your-transaction-data-safe/), December 2017

[4] Preethi Kasireddy, “How does Ethereum work, anyway?”, published on Medium

(https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369), September 2017

[5] Preethi Kasireddy, “How does Ethereum work, anyway?”, published on Medium

(https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369), September 2017

[6] Vitalik Buterin, “Merkling in Ethereum”, published on Ethereum blog

(https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/), November 2015

[7] Etherscan website, https://etherscan.io, June 2018

[8] Trust wallet website, https://trustwalletapp.com, June 2018

[9] Branch website, https://branch.io, June 2018

[10] Quid wallet website, http://quidwallet.com, June 2018