Application of Smart Contracts based on Ethereum Blockchain for the Purpose of Insurance Services

Veneta Aleksieva
Computer Science and Engineering Department
Technical University of Varna
Varna, Bulgaria
valeksieva@tu-varna.bg

Hristo Valchanov
Computer Science and Engineering Department
Technical University of Varna
Varna, Bulgaria
hristo@tu-varna.bg

Anton Huliyan
Software and Internet Technologies Department
Technical University of Varna
Varna, Bulgaria
anton.huliyan@gmail.com

***Abstract*- The insurance business is looking for solutions to minimize its operational costs and time to process claims for losses. Blockchain technology gives advantages such as transaction security, process identification, process automation and payment speed. This paper presents an experimental implementation of smart contracts based on Ethereum blockchain for insurance services. A decentralized crypto-token, based on ERC20 standard for smart contract, is implemented. A web-based interface is created for sales of these crypto-tokens. The results from the experimental tests are presented.**

Keywords—Smart contract, insurance, blockchain, Ethereum

I. INTRODUCTION

The idea of smart contracts has been known since the 1990s, but it began to shift from concept to potential reality with the advent of blockchain technology and in particular Ethereum blockchain [1] and the development of the new programming language Solidity [2]. Blockchain technology offers the ability to deliver faster and more secure transactions; it optimizes and automates the back-office operations and reduces costs through the use of cloud technologies. The impact of blockchain technology in the insurance business has been adopt by industry leaders, including the World Economic Forums in 2016, when they described the expected change in insurance processes using blockchain technology for the first time [3].

The Microsoft Blockchain as a Service platform, has been supported by Microsoft Azure since 2018. It offers crypto APIs based on Ethereum [4]. In this way, Microsoft Azure's hybrid cloud capabilities provide customers with new blockchain services that will streamline their complex business processes.

In [5] is presented a SWOT analysis of the advantages and disadvantages of such applications, which could potentially be useful in different sectors of insurance services. It is based on the prototypes of blockchain technology applications in the insurance sector. The main conclusion is that this sector has not been fully explored yet, but it is a fact that in insurance services the benefits of blockchain technology are used to:

- Improving customer satisfaction in processing of claims for redress;
- Reduction of operating costs;
- Authentication through signed transactions from a blockchain address;
- Insurance premium calculation / risk assessment / fraud prevention;
- Pay-Per-Use / Micro insurance by automatically activating / deactivating of policies, based on smart contracts;
- Peer-to-Peer insurance.

This paper introduces the experimental implementation of smart insurance contracts on Ethereum blockchain.

II. RELATED WORKS

The insurance sector of economics has begun exploring the application of blockchain technology through significant investments from large and small companies, research by consulting firms and the creation in 2016 of the first B3i-oriented blockchain insurance consortium involving over 40 companies [6]. The Cat XoL application, which is created by the B3i consortium (release will be in Q4 2019), uses Corda Distributed Ledger Technology. It has been tested at the hackathon in Zurich on 19-20.07.2019, where for 4 hours the participants (7 insurers, 12 brokers and 21 reinsurers) generated 1,033 messages during the negotiation process, which led to the signing of 41 contracts through the application.

In 2017 the German insurance company Etherisc [7] has invested in the development of an open source platform that focuses on decentralized insurance applications—the Decentralized Insurance Platform. Etherisc focuses on the use of blockchain technology to reduce inefficiencies (high document processing fees and long claims processing times). The platform has already been implemented in various areas of insurance, such as: harvesting agricultural producers against any losses due to the weather; potential crypto-currency hacks; the use of tokens for risk groups that effectively replace the role of reinsurance.

At the end of 2017, US insurance company Beenest [8] and WeTrust [9] develop blockchain-based homeowners insurance. By June 2019 it provides $15 million from token sales to help fund the development of blockchain based insurance.

The Estonian software company Guardtime [10] and logistics giant Maersk [11] developed insurance platform Insurewave, based on a blockchain and smart contracts. The platform integrates and secures streams of various data sources related to shipping insurance worldwide.

III. INSURANCE SERVICES – A CLASSICAL PROCESS

In the case of classic insurance services, the process of claiming insurance when the event occurs, is shown on Fig. 1. The insuree reports the loss and claims against the insurer (and the reinsurer if it is applicable) through a broker (1). The broker may request (2) additional information from

978-1-7281-4754-3/19/$31.00 ©2019 IEEE

the insuree for the loss, then he sends the claim to the insurer and the reinsurer (3). After verifying the received information, the insurer acknowledges receipt of the claim (4).

The next step is shown in Fig. 2. The loss adjuster assesses the loss in the claim of the insuree, checks its validity using information from the insuree and from additional sources such as statistics and reports (5). If the insurer requires additional information, the broker contacts to the insuree (6) to inform him what evidence of the loss he must present.

The final stage is related to the payment of the compensation (Fig. 3). After the loss assessment, the loss adjuster draws a conclusion. He sends it to the claims agent (7). If the claim is approved (in whole or in part), the claims agent transfers the calculated amount to the insuree's bank account (8).

In this classical insurance process there are a lot of problems. For example, the insuree must complete a lot of documents and provide evidence of the value of the loss. The broker brings additional delays and costs. Sources of information are heterogeneous and fragmented, often requiring the insurer to contact to additional sources. Fraud is possible—loss assessment is done without exchanging information between insurers. The processing of the claim is subjective and not automated.

IV. Insurance Services based on Blockchain Technology

The classic claim process can be improved by using smart contracts and blockchain technology (Fig. 4). Information of loss can be sent from the insuree or directly from sensors mounted in the insured object (smart asset) to an automated claim processing application (1). For the relevant insurance policies provided by the smart contract, the customer will receive a real-time confirmation (2). The claim is automatically processed by a smart contract based on business logic, using the information provided by the insuree (3).

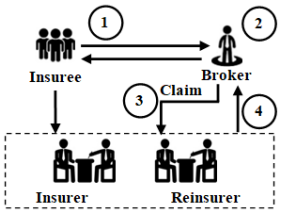

Fig. 1. Insurance claim

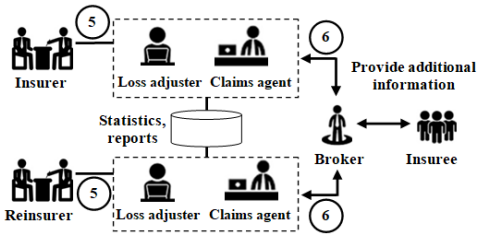

Fig. 2. Loss assessment

Fig. 3. Payment of compensation

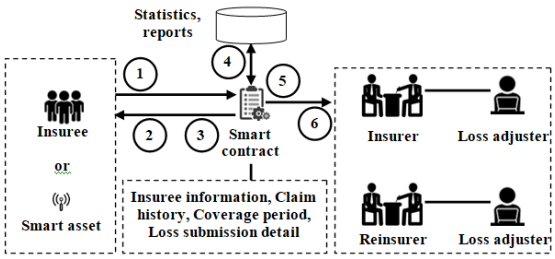

Fig. 4. Loss assessment with smart contract based on blockchain

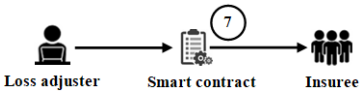

Fig. 5. Payment of compensation with smart contract

This approach automatically uses additional sources (statistics, reports) to evaluate claims and to calculate loss (4). Depending on the insurance policy, a smart contract can automatically calculate personal liability (5). In certain situations, a smart contract may activate an additional claim assessment (6). If the claim is approved (Fig. 5), payment to the insuree is initiated by smart contract (7).

The advantages of the new approach, based on smart contracts on blockchain technology, can be seen in several aspects. Claim submission is simplified and automated. Thanks to the direct exchange of loss information between insurers, this approach eliminates the need of brokers and reduces the time needed to process a claim.

The embedded business logic in blockchain smart contract eliminates the need for loss adjuster to review every claim (except in specific situations). The insurer has access to the origin of the loss, which helps him to identify potential fraud attempts. The process of payment for loss is automated by the smart contract on the blockchain, with no need of an intermediary claims agent.

V. Smart Contracts for Insurance Services on Ethereum Blockchain

The use of blockchain technology in the insurance sector can be implemented with both private and public blockchains. Private blockchains are only readable / writeable by their owner, so they have the advantage of tracking the sender of a transaction and all previous transactions, which reduces the risk of data tampering. With smart contracts, they can be used to increase the automation of existing tasks. They guarantee lower costs and shorter transaction validation time. They reduce the risk of attacks (as the nodes validating transactions are known) and they provide increased confidentiality (since read permissions can only be granted to selected nodes). In the event of errors or bugs in smart contracts, the private blockchains could be modified or restored to previous transactions.

However, public blockchains, such as Ethereum, are useful for managing automatic payments with existing crypto-currencies or when there is a need to secure trust between the parties.

Combined solutions can be used—a private blockchain to automate back-office operations and a public blockchain to manage automatic payments with existing crypto-currencies or when it is necessary to ensure the trust.

VI. Implementation of Proposed Smart Contract

The development of a smart contract (no matter what economic area it is intended for) takes place in 3 phases:

1) Analyze: What problem a smart contract need to solve. The main objects, as well as the links between them, are identified.

2) Design: Based on the information from the first phase it must develop a scheme of the objects that will serve as variables and the interactions with other objects that will be realized as functions. The information gathered also serves to determine the blockchain technology at which a smart contract will be realized.

3) Implementation: The components of the smart contract are implemented and broadcasted on the network.

Architecture of application is presented on Fig. 6. The implementation of the ERC20 standard guarantees the smart contract free movement among the participants in the Ethereum network. The proposed solution was developed with Truffle and Ganache under the MacOS High Sierra operating system. Ganache creates a local blockchain based on Ethereum, which can directly execute commands as well as perform tests.

Metamask is used, as there is no need to download a local copy of the entire blockchain, and it is making a connection to a site that makes a connection with Ethereum. Metamask takes care of all requests from and to the blockchain network. Metamask can perform the Ethereum wallet function and support sending and receiving Ethers and ERC20 tokens. The private keys of the users is stored on the local machine, so Metamask's servers have no information about them. An application interface is shown on Fig. 7. It is based on JavaScript library web3.js, which allows communication with the Ethereum network and perform smart contracts.

Truffle is used for the implementation of the smart contract. It is an integrated system for compilation of the written smart contracts, and it uploads them on the Ethereum network. In addition to the functions implementing the ERC20 standard, to ensure the proper operation of the smart contract, the functions for token sale (Fig. 8), termination of sale (Fig. 9) and payment of funds to the contract owner are needed.

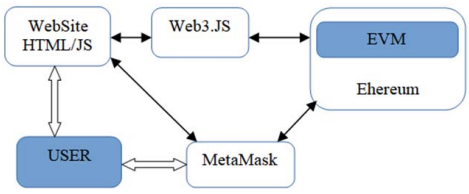

Fig. 6. Architecture of application

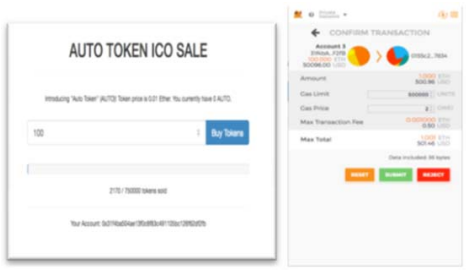

Fig. 7. Interface of application

```
function multiply(uint x, uint y) internal pure returns (uint z) {
    require(y == 0 || (z = x * y) / y == x);
}

function buyTokens(uint256 _numberOfTokens) public payable {
    require(msg.value == multiply(_numberOfTokens, tokenPrice));
    require(tokenContract.balanceOf(this) >= _numberOfTokens);
    require(tokenContract.transfer(msg.sender, _numberOfTokens));

    tokensSold += _numberOfTokens;

    Sell(msg.sender, _numberOfTokens);
}
```

Fig. 8. Purchase of tokens

```
function endSale() public {
    require(msg.sender == admin);
    require(tokenContract.transfer(admin, tokenContract.balanceOf(this)));

    admin.transfer(address(this).balance);
}
```

Fig. 9. Termination of sale

When calling the function „endSale“, it is checked who call it, because only the account administrator has the rights over it. After that, all remaining unsold tokens are transferred to the account of the administrator on its address. As a final step, all collected Ethers are transferred to the administrator. This way, if someone tries to make a purchase after calling "endSale" the check for available tokens will fail and the tokens will not be sold.

VII. Experimental Study and Results

The first part of the tests is related to the proper work of smart contracts—balance of account, transfer of tokens etc. There are a handful of tools for automated smart contract security vulnerability testing based on code-level analysis. In [12] is given a synopsis of the four most related tools that is possible to use in experiments: Oyente, Mythril, Securify and SmartCheck. However, the evaluate level of rigor, ranging from syntactic, heuristic, analytic to fully formal, refers to underlying security testing technique of the given tool and up to this moment the researchers trusted on the implemented in the Solidity test tools. Truffle (and Solidity) has a built-in smart-contract-testing mechanism that is written in JavaScript, which here is used.

On Fig. 10 a token transfer test code is provided. For direct transfer testing, 250 000 tokens are transferred from the administrator's address to the recipient's address. Once the transfer has taken place, the event is captured and checked for "Transfer" type. If this test is successful, the balance of the recipient address is checked for the presence of transferred tokens. On Fig. 11 a test code for crypto-token disposal is provided. The delegated transfer check is similar to the direct transfer check. First, 100 tokens are transferred from the administrator's address to the address from which a delegated transfer will be allowed—address 1. It is allowed 10 tokens to be spent from address 3, which sends them to address 4. After performing these actions, it is expected that address 1 have 90 tokens, address 2—0, and address 3—10. The result of their implementation with correct parameters is shown on Fig. 12, and with wrong parameters—on Fig. 13. The transaction of these tokens in real Ethereum blockchain is presented on Fig. 14.

```
it('initializes the contract with the correct values', function(){
    return AutoCoin.deployed().then(function(instance){
        tokenInstance = instance;
        return tokenInstance.name();
    }).then(function(name){
        assert.equal(name, 'AutoCoin', 'has the correct name');
        return tokenInstance.symbol();
    }).then(function(symbol){
        assert.equal(symbol, 'AUTO','has the correct symbol');
        return tokenInstance.standard();
    }).then(function(standard){
        assert.equal(standard, 'AutoCoin v1.0', 'has the correct standard');
    });
})
```

Fig. 10. Code of test for transfer of tokens

Fig. 11. Code of test for disposing of tokens

Fig. 12. Results of test with proper parameters

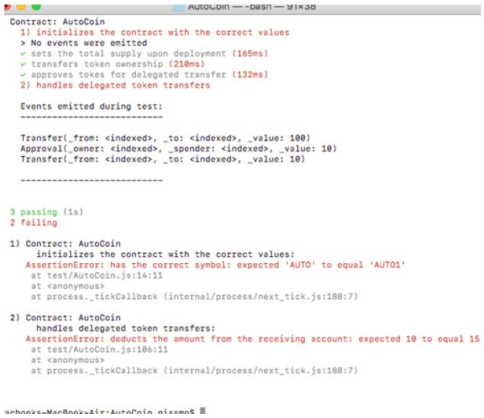

Fig. 13. Results of test with wrong parameters

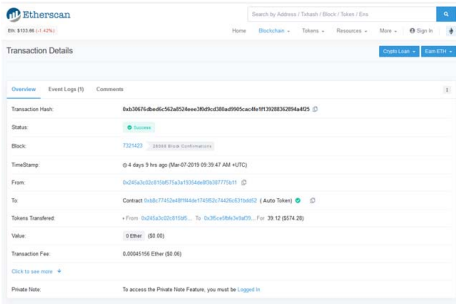

Fig. 14. Transaction of the proposed token in the Ethereum blockchain

Network communication with the Metamask server was also tested when sending Ethers to buy a token for smart contract development proposals. The tests were carried out in the local network with flat topology. The client connects to the Metamask server. The parameters of computers are Apple Mac Book Pro Late 2011 Specs, Core i5 (I5-2435M) 2.4GHz 2/4 Cores/Threads, 4GB DDR3 1333Mhz RAM. In the Metamask when is sending an Ether to buy a token, the protocol TLSv1.2 is used. In Fig. 15 is presented network communication between client and Metamask server during successful transaction of tokens. The client sends Ether to the Metamask server to buy tokens. The client PC has IP address 192.168.100.2 in the local network in Varna, Bulgaria. Data transfer between the client and Metamask server is encrypted. The server is located in Ashburn, United States with an IP address 34.202.251.153 and communication is over TLSv1.2 protocol. The Metamask server does not allow invalid transactions. In this case, the block time is set so that when a transaction is created, a new block will be created and entered immediately into.

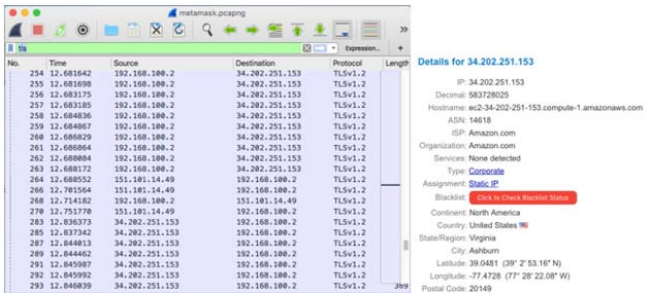

Fig. 15. Network communication from client to Metatask during transaction of the proposed token

VIII. Conclusion

This paper analyzes the possibilities of using blockchain technologies in the field of insurance services. The advantages of such solutions over the traditional way of insurance are presented. This new concept can be implemented with both private and public blockchains. Combined solutions can also be used—to automate claims settlement operations using a private blockchain, and to manage automatic payments with existing crypto-currencies with a public blockchain.

The proposed solution with smart contracts for insurance is based on the ERC20 standard. It has been implemented experimentally on Ethereum blockchain. The results of the experiments show that the proposed solution is fully operational in terms of managing automatic payments on approved claims for loss.

The goal of future research is to create a private blockchain to handle automatic claims handling operations, both on request from the insuree and directly from sensors installed in the insured facility. In this way, private blockchain will receive real-time back-end confirmations and based on statistics and reports stored in the blockchain, an automated claim assessment and loss calculation will be performed.

References

[1] Ethereum Homestead Documentation, http://www.ethdocs.org/en/latest/ (last visited 20.09.2019).

[2] Solidity Documentation, http://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.html (last visited 20.09.2019).

[3] JesseMcWaters R., G.Bruno, "The future of financial infrastructure. An ambitious look at how blockchain can reshape financial services." World Economic Forum, p.130, 2016.

[4] Blockchain as a Service, https://cryptoapis.io/products/baas/ (last visited 20.09.2019).

[5] Gatteschi V., F.Lamberti, etc., "Blockchain and Smart Contracts for Insurance: Is the Technology Mature Enough?", MDPI, Basel, Switzerland, Future Internet 2018, p.16, 2018.

[6] B3i - The Blockchain Insurance Industry Initiative, https://b3i.tech/home.html (last visited 20.09.2019)

[7] Etherisc, https://etherisc.com/ last visited 20.09.2019)

[8] Beenest, https://www.crunchbase.com/organization/the-bee-token/ (last visited 20.09.2019)

[9] WeTrust, https://www.wetrust.io/(last visited 20.09.2019)

[10] Guardtime, https://guardtime.com/(last visited 20.09.2019)

[11] MAERSK, https://www.maersk.com/(last visited 20.09.2019)

[12] Reza M., PariziAli Dehghantanha, Kim-Kwang Raymond Choo and Amritraj Singh, "Empirical Vulnerability Analysis of Automated Smart Contracts Security Testing on Blockchains", Proceedings of CASCON'18, Canada, October 2018.