Design and Implementation of Smart Contract: A use case for geo-spatial data sharing

E. Leka[1], L. Lamani [2], B. Selimi[3] and E. Deçolli[4]
[1] South East European University, Tetovo, Macedonia
[2] Polytechnic University of Tirana, Tirana, Albania
[3] South East European University, Tetovo, Macedonia
[4] Polytechnic University of Tirana, Tirana, Albania
el23618@seeu.edu.mk, luis.lamani@fgjm.edu.al, b.selimi@seeu.edu.mk, eliodecolli@gmail.com,

***Abstract* - In this paper, we propose to use the blockchain technology as a mechanism to store and share geospatial projects. Blockchain helps to improve efficiency and security. Smart contracts provide a secure, distributed and shared decentralized ledger of all assets and transactions. We will discuss a way to implement a platform on which scientists can share their studies. We propose a design methodology for the mentioned smart contracts, which enables the development of different use cases using blockchain technology. A detailed design of the smart contracts, functions and processes is presented. We will provide an outline of advantages and limitations of blockchain in general, and for the proposed platform.**

Keywords - Blockchain; Ethereum; Smart contract; distrubuted data sharing

I. INTRODUCTION

Lately blockchain technologies have received considerable attention form many researchers and government institutions. In this paper, we propose to add blockchains as a mechanism to design and implement a smart contract application to share geo-spatial data. Blockchain technology [1,2] enables the creation of a decentralized environment, where the cryptographically validated transactions and data are not under the control of any third-party organization [3]. Any transaction ever completed is recorded in an immutable ledger in a verifiable, secure, transparent and permanent way, with a timestamp and other details. Each blockchain employs a consensus mechanism to resolve different states, or "forks" in the network and the choice of mechanism varies among networks. [4] details a variety of consensus mechanisms.

We have chosen Ethereum [5] as the blockchain in our research. Ethereum provides a decentralized Turing-complete platform [6] called Ethereum virtual machines to run application codes called smart contracts. Ethereum also provides a currency called ether, that is used to implement value exchange between nodes in the platform. Smart contracts are codes that reside within the Ethereum blockchain environment that executes when specific conditions are met. Smart contract can store and control ether. The functionality to control ether can be used to build applications that require deposit and payout of ethers such as games, Identity managements systems [7,8]. In the Ethereum blockchain platform, each computational step has a cost associated with it [9] called gas.

In this paper we propose a design methodology for smart contract to implement a project called 'D-GIS' [10], which enables geologists and engineers to share geospatial data and studies in the most efficient manner possible. In [11] we have discussed a way to implement a platform on which scientists can share their work/study in a secure and efficient manner using blockchain technology and artificial intelligence [12]. We use the blockchain to prevent two of the major flaws in scientific research as well as data sharing and collection: ownership-rights, and based on the model described here, equal rights and a democratic eco-system for publishing your data, the last is completely automated; by doing so we also remove the need for a third-party, thus removing bureaucratic short-comings, i.e. corruption and inefficiency [11]. Based on this, we also introduce a model detailing how such platform can be programmed to mimic an economic market in order to also produce competition between authors, leading to better studies. Artificial intelligence is used in order to support this model by examining similarities between projects, therefore detecting potential plagiarism or unauthorized redistribution.

We aim to build a ranking system based on points (tokens) won by contributing to the community, the bigger the contribute, the bigger the reputation the user has in the community. When someone votes, his vote is equivalent to his power inside the network. But also, the voter has a small penalization when he votes. This acts as a friction mechanism which limits the amount of votes a user can submit. In order to generate a median-like variable to check the ratio of a specific user's tokens with respect to the total amount of tokens in the network.

D-GIS is a decentralized application, meaning that no one owns it and everybody can benefit from it. Due to this fact, no one can manipulate the will of the community on a certain decision it makes.

Although 'D-GIS' makes use of two technologies: Deep Learning [13] for categorization and Blockchain for distribution, at this paper, we're going to focus only on the distribution part, more specifically the on-chain part. Focusing mainly on some coding approaches related to Smart Contracts, taking a look at advantages and disadvantages for each of them. Our main goal is to

identify specific methodologies used in DApp, which allow us to grab onto the strongest pillars of Ethereum and in turn develop a safer experience for the end-user.

The paper is organized as follows. In Section II is presented a background of blockchain, smart contracts and different approaches used to implement smart contract and related works. Section III presents system details, on-chain and off-chain elements. The On-chain elements of our implementation are described at section IV and the off-chain ones at section V. Smart contract design methodology, and some aspects of security are described in section VI. The remain sections discuss conclusions and future work.

II. Background and Related Work

Many people and researchers believe that blockchain applications in different vertical industries could lead to three generations of the Blockchain, namely Blockchain 1.0, Blockchain 2.0 and Blockchain 3.0 [14,15,16,17].

Analyzing the papers, we found that some of the platforms that use the concept of smart contract are: Bitcoin [18], Ethereum, Codius [19], Lisk [20], Counterparty [21], Monax [22], Stellar [23], Rootstock [24]. Two of most prominent platforms that use concept of smart contracts were Ethereum and Bitcoin.

To implement smart contract, exists three approaches [25,26,27]: (1) *Centralized*: The smart contract is deployed on a Trusted Third Party. This approach is also known as off-blockchain implementation since there is no blockchain involved; (2) *Decentralized*: The Smart contract is deployed on a blockchain platform such as Ethereum. This approach is also known as on-blockchain; (3) *Hybrid*: The contract is split and deployed partly off and partly on-blockchain. Some clauses are enforced off blockchain; others are enforced on-blockchain. The partition is based on several criteria including blockchain cost, performance, consensus latency, smart contract languages and privacy.

The paper [28] discusses the implementation of smart contracts on hybrid architectures. On-blockchain platforms suffer from scalability, performance, transaction costs and other limitations. Off-blockchain platforms are afflicted by drawbacks due to their dependence on single trusted third parties. They showed how a smart contract can be split and executed partially and an off-blockchain contract compliance checker and partially on the Rinkeby [29] Ethereum network. They argument that in several applications areas, hybrid platforms composed from the integration of off- and on- blockchain platforms are better than either alone [28].

While implementing smart contracts, it is important to take into consideration security aspect. Unfortunately, it is a challenge to create smart contracts that are free of security bugs. We can mention some cases where critical vulnerabilities in smart contracts have led to losses reaching millions of USD [30,31,32,33].

An important implementation of a geospatially-enabled blockchain is FOAM [34]. It uses a crypto-spatial coordinate (CSC) system. A FOAM blockchain does not just record an entry's specific time, but also requires and validates its associated proof of location, giving an immutable spatial context that regular blockchains lack, and allowing the accurate mapping of physical world events in a temporal sequence [35, 36, 37].

In [38], authors think that citizen engagement in the crowdsourcing of geo-tagged data can be combined with augmented reality and blockchain technology in powerful new crisis mapping and recovery scenarios, e.g., in the production and real-time updating of an augmented crisis map for navigating a disaster-stricken area.

III. System Details

For the design and deployment of Blockchain implementation we must consider the following selection criteria [39,40]: (1) Type of consensus mechanism [41]; (2) Programing language; (3) Type of cryptocurrency [42] used for mining; (4) Authorized participant in Blockchain network.

A high-level programming language called Solidity [44], is used to write smart contracts and decentralized applications (DApps). In our study we focus on smart contracts written in Solidity [43, 44] programming language, because it is adopted by the largest blockchain network that supports smart contracts.

A Decentralized Application (DApp) is an application that uses smart contracts providing a friendly user interface to smart contract [45]. The steps that are required to follow for the development of the DApp are [45]: (1) design and implementation of smart contract in high-level language; (2) Compile the contract to generate a binary file; (3) Deploy the contract on Ethereum Blockchain network using Ethereum clients; (4) Build a Web application (front-end) that interact with the smart contracts.

The basic setup of our D-GIS project consists of two components. The on-chain components which mainly consist of Ethereum Smart contracts for access control, generating and storing projects, and the off-chain modules which consist of client application module that interfaces with the smart contracts.

IV. On-chain Elements

A. Local Storage

A local storage (LS) is used to represent a work done by a user and then uploaded on the blockchain. Physically speaking it represents a study on a specific zone (inside a specific region). These elements have an ownership, rating and other details used to describe the work done. Figure 1 represents a Local Storage in its structure.

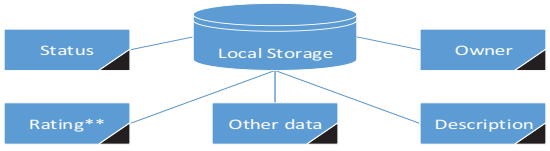

Figure 1. Local Storage

*OWNER** - This represents the owner of the storage, in Solidity this is implemented via a variable type named,

address. It points to an *Individual Account* on the network. This value is set in the initial deployment of the contract and cannot be changed later on. We assign an owner for each storage in order to avoid double-copies and to grant special *Read/Write* access to the contract. This assures that nobody else, except the author of the work will be able to access sensitive information inside the storage (i.e. status, description or if it's for sale, price).

*Rating*** - This is the reputation of the work stored in this storage across the Global Storage in which it is saved. Using of "Rating" will serve as a way to categorize any (geo) scientific based on their peer-reviews as well as an incentive for users to upload a better work. Ratings are manipulated via *user votes* and a specific *"Filtering Algorithm"* used to restrict or grant specific access on a work based on its reviews.

Description - This is a generalized description of the work or study done in that area. It is used not only for users to recognize and create a general idea on what the material uploaded consists of but also for indexing and finding data uploaded in a certain region.

Status - This parameter indicated whether this storage is available anymore. It can be either set to *true* (if available) or *false* (otherwise).

Other data - In this field we can add additional information regarding the material uploaded. On the 'DGIS' platform we use this space in order to assign a value (of Ether) which must be submitted in order to access the material.

We need to take into consideration that the material itself is not uploaded on the chain; instead we upload only its metadata. This allows us to take control over the work itself and edit it if need be.

B. Individual Accounts

Every user is registered to the network as a "Streamer", in this structure we store information regarding the user's work uploaded as well as his reputation across the chain. At Figure 2, we present the structure of an Individual Account (Streamer).

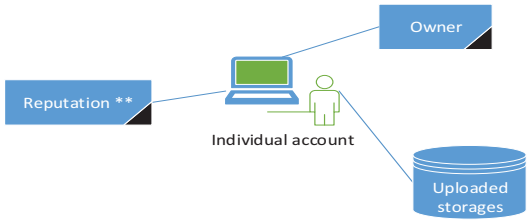

Figure 2. Individual Account

*OWNER** - In this case this address points to a wallet public address. It serves the same purpose of the field inside the LS element.

*Reputation*** - This represents the user reputation across the network, set automatically by the Filtering Algorithm based on reviews of his/her studies.

Uploaded Storages - This is used to index every LS owned by this user. It is not publicly accessible (although it might as well be), LSs can be found via a Global Storage.

The idea behind this, is that material uploaded by a single user is connected altogether and also used to create a general "image" of the user in the chain. This image represents the reputation and privileges of the user across the Global Storage. Thus, not only linking work-done to its author but also detailing and filtering any invalid or out-of-date study. Users with a negative reputation are restricted from selling (if selling is implemented) as well as publishing additional work on the network.

Global Storages (GS) are used to identify, index and store global data regarding works published on the network. These are used in order to classify materials based on their Geographical location, as such each Country (or another specific zone) will have its own GS representation on the network. Additionally, GS are used to grant access to individuals wishing to contribute with their own work done in that specific region. Thus, allowing users to register into this storage and have a limited writing access on it. Global Storages are designed in the following way as presented at figure 3:

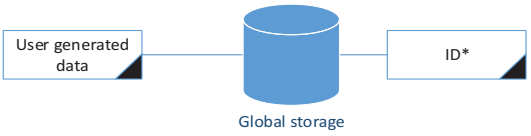

Figure 3. Global Storages

*ID** - This is used to identify each Global from one another. Information regarding its public key (address) and name (i.e. Texas, USA) is stored in the off-chain part. There's a check on the off-chain part to determine if the given ID has been taken or not so there are no GS with the same ID. However, users can freely initiate 'Globals' of their own, in which case there's no particular check to whether the given ID is already in use, this is of no concern whatsoever as we use IDs only to index 'Globals' on the off-chain source so users can search through them, but the actual connection is done by using the address of each GS. So, if user A initiates a new Global (which is not indexed and cannot be found via an implemented search engine), user B can still access it as long as A gives B the GS address.

User-generated Data - This is a collection of ratings and accounts registered on the Global in order to generate enough information to be used within the filtering algorithm, creating by us at [11]. Such data consist of a list of registered accounts (Streamers), a list of Local Storages uploaded by the user and so forth.

Each GS has its own data, therefore its own set of rules which either grants or restricts certain privileges for users. Therefore, in a hypothetical situation we have: User A, Global G, Global S.

User A can publish material on G, but cannot do it on S, as it doesn't meet the necessary requirements to do so (user-generated data on S is different, therefore leading to different rules). Although, this is not always the case it is of a certain importance to mention it, so you can create a better understanding of how every element has its own

attributes and is free from a centralized controller. Special cases can arise in which A can publish in both G and S.

Before moving on to the off-chain elements it is of vital importance that you are able to realize the similarities that this system and an economic market has. We allow users to upload content wherever they want as long as they're capable of meeting the specific requirements that "market" has. By taking into account that some of these studies or works can be sold, each GS is regulated by the Filtering Algorithm which prevents monopoles and sets competitive statuses for everyone joining it.

Following this logic, if User A wants to gain revenue from a study then it must ensure that the work being uploaded is valid and correct, therefore it will be cut out of the network and his/hers privileges on that Global will lower. Moreover, if User A's reputation is being decreased this will affect his capability to interact with other 'Globals', thus cutting down A's personal "market share". So, at this moment we have a networking made out of three-on chain elements like as they are presented at Figure 4:

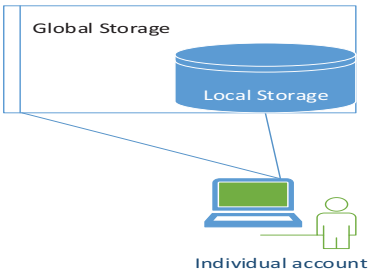

Figure 4. Global Storages

At this point the Streamer has access to each Local Storage inside the Global Storage, however it cannot edit or access sensitive data as long as it does not have ownership of that Local Storage.

V. Off-chain Elements

A. Ethereum Client-Server Service

We're going to name this element as "GoServer", since our work so far has been done on Geth and Go-Ethereum [46] (both Go implementations of the Ethereum protocol). GoServer is an application-layer which allows users to connect to the Ethereum blockchain (or even a private blockchain) thus interacting with storages and accounts there. In our implementation we've chosen to use a client-server approach, allowing multiple users to access the network from multiple sources. An example of this would be User A accessing the network from both its PC or from its Smartphone to check for pending requests.

Such architectures not only favor a more complex yet strangely elegant approach to access the network but it also reduces the weight of each user's application on its device. Another way to see this is like a proxy service, where users send requests to a single instance which in turns runs as a proxy connecting both users and the blockchain yet still remaining in the middle of the action. At figure 5, we illustrated the whole client-server implementation:

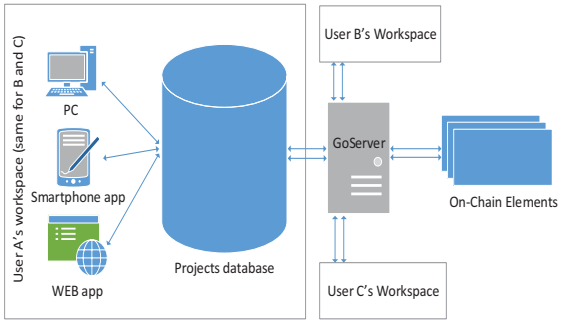

Figure 5. Client-Server Implementation.

As you can see from the diagram multiple users are able to send requests to the network via the GoServer at once. Each request is processed and queued for sending. In order to enable a safer experience for users, the GoServer registers each user's device and allows only known devices to operate. For instance, if a user wants to use his/her PC to edit and submit studies said user must "register" on the GoServer in order to do so. Registrations are either manual or automatic, it can be configured once on the server startup. An example of this would be the following scenario:

User A wants to register its smartphone as a device in order to access the network. Whether *User A* has any other devices registered (like a PC or a WebApp) is of no concern. Assuming that the server has been configured in order to accessible by everyone (aka automatic registration) and *User A* has been already registered on the server, *User A* has to follow these steps in order to add its smartphone:

1. Send a *NEW_DEVICE* request to the server from the unknown device.
2. Server checks whether this user is registered and the device is not listed on its database.
3. Server adds the newly registered device to its database and links it to the personal virtual database of that user.
4. Server sends a unique key to the user in order to for the user to access the server.

Since the GoServer is responsible for connecting the user to the network, it also passes on request from other sources to the user, such as material reviews, requests and so on. Then the server stores that information in a personal database for each user registered on it, allowing users to access their history but also avoiding data loss if a sudden server breakdown occurs. However, if the user is not registered on the server there are two more scenarios to cover: (1) The server has been configured in order to accept automatic registrations; (2) The server has been configured not to accept automatic registrations.

In the second case the server owner must register the user manually from the machine running the server, then the user can proceed to register its devices. Otherwise, automatic registrations follow these steps: (1) User sends a registration request to the server, including any info regarding itself (username, password and so on); (2) Server checks whether a user with the same name has

been registered; (3) Server registers the user and sends back its ID.

VI. Methodology Implementation

In this section we aim to present a model, or methodology to use while creating a DApp. D-GIS is a decentralized data sharing platform built on top of Ethereum's blockchain. Smart contracts are public, hosted on a huge decentralized network, therefore accessible by millions of users, so we must write our code keeping in mind that we shall inspect any kind of security worm-hole. Another thing to take in consideration is code efficiency, in order to reduce gas fees [47, 48]. These fees are based on what users are willing to pay in order to keep the network up as well as how much miners are willing to invest in the forward-propagation of the blockchain. Thus, every time you deploy a smart contract, call a specific function or change its state you have to pay a specific fee in order for your transactions to be pushed on top of the latest block being mined. Fees are known to lower once the Ether price rises, however this is not in a 1:1 ratio.

Last but not least, we must remember that once a contract is deployed its code cannot be changed anymore. So, once we reach production stage, we have to be sure that our code is completely "bug free". However, this is very impractical and, in most cases not really possible. Additionally, if we want to implement new features, we must re-deploy the whole network of smart contracts, and this is a real trouble. To get around this, we present a work-around idea, which adds a bit more of work but it might save some time during the future releases.

A. Security

Contracts ownership is now a standard, but here we have to make sure that the caller which wants to access a published data, or even change its rating, must be of the right type. More specifically we allow only *TxtStreamers* to vote on a storage reputation. To do this, we first register each streamer's address as a "friendly address". And to take it one step further, we also check during this registration that the address we're marking as friendly represents a *TxtStreamer*. For this we use the soon-to-be implemented opcode, EXTCODEHASH [49] and check its hash code with that of a *"dummy" TxtStreamer* deployed by the Global storage during its initialization. The following code can be found inside the Global's constructor:

```
TxtStreamer mDummy_streamer = new
TxtStreamer(address(this));
address m_addr = address(mDummy_streamer);
assembly{
       dummy_streamer := extcodehash(m_addr)
}
```

dummy_streamer is of type bytes32, which is a bytes representation of a Keccak256 hash code [50] derived from the deployed dummy's byte-code. From this we use mappings to assign contracts as friendly.

```
function IsStreamer(address addr) public view
returns(bool o_streamer){
bytes32 o_data;
        assembly{
            o_data := extcodehash(addr)
        }
        o_streamer = (o_data == dummy_streamer);
    }
    function RegisterStreamer() public {
        require(IsStreamer(msg.sender), "Invalid
source calling function RegisterStreamer().");
        require(!IsAccountPresent(msg.sender),
"Streamer already registered.");
        accounts_present[msg.sender] = true;
    }
```

accounts_present is a boolean type of mapping address, and if *accounts_present[X] == false* (that's what *IsAccountPresent(<address>) checks)*, then the calling contract *(which address is X)* is not "friendly", therefore it cannot vote on storages reputation. However, hash-checks consume gas on their own, that is one of the reasons we consider using mappings to query "friendly" addresses instead of checking the hash of each storage on every call they perform inside a Global.

B. Proxy contracts

When you deploy a contract, its address cannot be modified, therefore if you want to roll out updates you have to re-deploy every other contract. Sometimes this is not a problem, however there are times when users are not able to switch to the new contracts unless they migrate all their data there, which is impractical.

We prefer to call a form of work around for this, a proxy contract. Just like a real proxy this type of contract will be used as a bridge to connect users to the source they want to call. *Contract 1* can call a specific function inside *Contract 2*, by simply knowing the address of *Contract 2*. However, if we push an update, the newly deployed contract will have a different address, so *Contract 1* cannot access it anymore. A proxy contract works by storing the address of the currently updated contract and simply passing it to the caller. Moreover, the proxy contract can also store previous versions of the source contract, and let the caller decide which version they want to use. At figure 6, presented below, we can see the structure of a proxy contract.

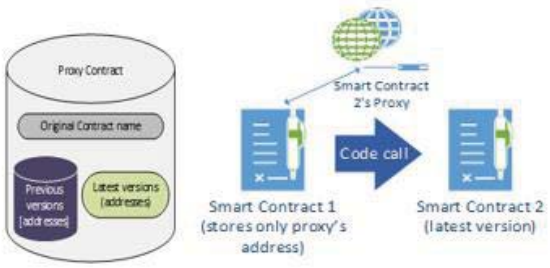

Figure 6. Structure of proxy contract.

So, a caller contract can simply store the address of the proxy and request the newly released version to be sent, or otherwise if possible, a previous working version.

VII. Conclusion

Even though Ethereum is still in its infancy, it shows all the signs of a successful future. Developers looking to invest their time and resources in this new technology must always be prudent while deploying their projects.

Analyzing D-GIS as a use case we try to highlight some of the advantages and possibilities that can be derived through the blockchain. On this paper, we mainly focused on some coding approach relating smart contracts. Our main goal was to identify specific methodologies used in DApp, which allow us to grab onto the strongest pillars of Ethereum and in turn develop a safer experience for end-user.

REFERENCES

[1] S. T. Aras, V. Kulkarni, "Blockchain and its application – a detailed surevey", in Internrational Journal of Computer Applications, Vol 180-No.3, December 2017

[2] Swan, Melanie. Blockchain: Blueprint for a new economy. " O'Reilly Media, Inc.", 2015.

[3] C. Holotescu, "Understanding blockchain technology and how to get involved," in the 14th International Scientific Conference eLearning and Software for Education Bucharest, April 2018

[4] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," IEEE Communications Surveys and Tutorials, vol. 18, no. 3, 2016, pp. 2084–2123.

[5] "Ethereum," https://www.ethereum.org/, 2019.

[6] V. Buterin, "A next-generation smart contract and decentralized application platform," Ethereum Project, 2014 https://github.com/ethereum/wiki/wiki/White-Paper.

[7] P. Dunphy and F.A.P. Petitcolas, "A first look at identity management schemes on the Blockchain", in IEEE Security and Privacy Magazine", January 2018.

[8] Sh. Y. Lim, P. T. Fotsin, A. Almasri, O. Musa, M. L. M. Kiah, T. F. Ang, R. Ismail, "Blockchain Technology and Identity Management and Authentication Service Disruptor: A Survey", in International Journal on Advanced Science Engeenering Information Technology, Vol.7, No. 4-2, 2018.

[9] G. Wood, "ETHEREUM: A secure decentralized generalized transaction ledger", 2017 . http://gavwood.com/paper.pdf

[10] D-GIS Project, https://github.com/D-GIS/D-GIS

[11] E. Deçolli, K. Lleshi, E. Leka, L. Arapi "Using Blockchain Technology and Artificial Intelligence in geospatial data sharing" in EGU General Assembly 2019, in press.

[12] T. Marwala and E. Hurwitz, "Introduction to man and machines," in Artificial intelligence and economic theory: skynet in the market", Springer International Publishing AG, pp. 1-14, 2017.

[13] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," APSIPA Transactions on Signal and Information Processing, 2015.

[14] D. Efanov and P. Roschin, "The All-Pervasiveness of the Blockchain Technology", in Procedia Computer Science, 2018, pp. 116-121.

[15] N. Dimitri, "The blockchain technology - some theory and applications" Maastricht, The Netherlands: Maastricht School of Management, 2017

[16] M. Swan, "Blockchain: Blueprint for a New Economy", United States of America, O'Reilly Media Inc, 2015.

[17] I. Karamitsos, I, M. Papadaki and N. Barghuthi, "Design of the Blockchain Smart Contract: A Use Case for Real Estate" in *Journal of Information Security* pp. 177-190, 2018.

[18] Bitcoin, https://www.bitcoin.com.

[19] Codious: https://codious.org/, latest accessed Jan 2019.

[20] Lisk. https://lisk.io/, latest accessed on January 2019.

[21] Counterparty: https://counterpaty.io/ latest accessed January 2019.

[22] Monax. https://www.monax.io/, latest accessed on January 2019.

[23] Stellar. https://www.stellar.org/, latest accessed on January 2019.

[24] Rootstock. www.rsk.co/, latest accessed on January 2019.

[25] C. Molina-Jimenez, E. Solaiman, I. Sfyrakis, I. Ng, and J. Crowcroft, "On and off-blockchain enforcement of smart contracts," in Proc. Int'l Workshop on Future Perspective of Decentralized Applications (FPDAPP), 2018.

[26] J. Eberhardt and S. Tai, "On or off the blockchain? insights on offchaining computation and data," in Proc. i6th European Conference on Service-Oriented and Cloud Computing (ESOCC'17), 2017.

[27] G. Zyskind, O. Nathan, and A. S. Pentland, "Enigma: Decentralized computation platform with guaranteed privacy," arXiv:1506.03471v1 [cs.CR]..03471, Jan 2015,

[28] C. Molina-Jimenez, I. Sfyrakis, E. Solaiman, I. Ng, M. W. Wong, A. Chun, J. Crowcroft, "Implementation of Smart Contracts Using Hybrid Architectures with On- and Off-Blockchain Components", in IEEE 8th International Symposium on Cloud and Service Computing (SC2), 2018.

[29] Rinkeby Ethereum network, https://www.rinkeby.io/#stats, latest accesed January 2019

[30] Multichain homepage, https://www.multichain.com, last accessed December 2018

[31] Cnbc homepage, https://www.cnbc.com, last accessed January 2019

[32] Security Alert, available from: https://paritytech.io/blog/ security-alert.html, 2017

[33] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (sok)," in POST. Springer, 2017, pp. 164–186.

[34] FOAM. 2018, https://www.foam.space/, last accessed 14 March 2019.

[35] A. Dasgupta, "The Game Changer of Geospatial Systems-Blockchain", in Geospatial World (online), 2017, last accessed 1 March 2019

[36] J. Anderson, "FOAM: The future of Geospatial Data, on Ethereum Blockchain", Steemit (online), last accessed 1 March 2019

[37] W. Tewelow, "Bitcoin, blockchain and GIS could change the world. Geospatial Solutions (online), 2018, last accessed 1 March 2019

[38] N. Fierro, "How Augmented Reality can change how we navigate a natural disaster". MIMIR Blockchian Publication on Medium (online), 2017

[39] R. Beck, J. Stenum Czepluch, N. Lollike, and S. Malone, "Blockchain the Gateway to Trust-Free Cryptographic Transactions" in 24th European Conference on Information Systems, Istanbul, Turkey, 2016, pp.1-14.

[40] F. Glaser and L. Bezzenberger, "Beyond Cryptocurrencies—A Taxonomy of Decentralized Consensus Systems" in 23rd European Conference on Information Systems, Munster, 2015, pp. 1-18.

[41] Nguyen, G.-T., Kim, K.: A Survey about Consensus Algorithms Used in Blockchain. 28 (2018).

[42] R. Houben, A. Snyers, "Cryptocurrencies and blockchain. Legal context and implications for financial crime, money laundering and tax evasion", white paper, July 2018.

[43] C. Dannen, "Introducing Ethereum and Solidity", Apress, Berkeley, CA, 2017.

[44] Solidity, https://solidity.readthedocs.io/en/v0.5.3/

[45] I. Karamitsos, I, M. Papadaki and N. Barghuthi, "Design of the blockchain smart sontract: A use case for Real Estate" in *Journal of Information Security, 2018,* pp. 177-190.

[46] Go Ethereum, https://github.com/ethereum/go-ethereum

[47] T. Chen, X. Li, X. Luo, X. Zhang, "Under-Optimized Smart Contracts Devour Your Money", in IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), February 2017.

[48] Hackernoon, https://hackernoon.com/ether-purchase-power-df40a38c5a2f, last access on December 2018.

[49] EIP-1052, https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1052.md

[50] Keccak256 hash function, https://ethereum.stackexchange.com/questions/11572/how-does-the-keccak256-hash-function-work, latest access on Dec. 2018.