DADS: Decentralized (Mobile) Applications Deployment System Using Blockchain

*Secured Decentralized Applications Store

Faiez Altamimi *, Waqar Asif† and Muttukrishnan Rajarajan*
*Dept. of Electrical and Electronic Engineering, City University of London, UK.
{Faiez.Altamimi | r.muttukrishnan }@city.ac.uk
†Dept. of Computing and Engineering, University of West London, UK.
waqar.asif@uwl.ac.uk

***Abstract*—This paper proposes a comprehensive framework using blockchain architecture to create a decentralized applications store, in order to reduce the approval time for apps that are to be deployed on deployment systems i.e. applications stores, and specially for mobile platforms e.g. Apple and Google play stores. This work potentially can solve some of the existing challenges with the current centralized application approval process by Apple and Google. These companies currently charge a flat commission rate of 30% [1] of the total applications revenues from the applications owners. This commission is considered to be very high, especially for applications with revenues in the order of few million dollars. In addition, the length of the review period prior to deployment for any new or revised apps delays the release to the marketplace hence costing economic losses to the app owners. Currently, Apple and Google stores takes around two days and two hours respectively to review and approve the apps on their stores. However, by using decentralized technologies such as the blockchain, the time to review and publish a new or a revised app can be significantly reduced from few days to few minutes.**

In this paper a novel framework is proposed based on the blockchain architecture incorporating the Ethereum network, Interplanetary Files System (IPFS), VueJS, and a backend database. The framework is experimentally validated, and the results show the reduction in terms of the number of minutes to approve apps by the app stores.

***Index Terms*—Smart Contracts, Blockchain, IPFS, Secured and Decentralized Applications, Ethereum Ropsten and Rinkeby testnets, Vuejs, web3js, and Graphql**

I. Introduction

The number of smartphone users has increased significantly in the past couple of years. These figures have increased from 2.5 billion to approximately 3.5 billion from 2016 to 2020. This translates as an estimated 45% of world's population having smart phone devices. This increase in device sale is attributed to the proliferation of applications and the way in which the underline operating systems are built. Currently, the smart phone market realises two main operating systems (OS) namely Android and iOS. Despite the fact that both of them are built in complete isolation from one another, the underline phenomenon behind both these OS remains the same. Both of them rely heavily on providing the users the flexibility of installing applications on their devices. This as a result makes the app development community a big asset for both the OSs.

On one side, where app development is a key asset when defining the success of an OS, allowing third party applications can also pose security and privacy risk, both for the device user and the OS developer. In order to maintain control over the allowable third party applications, both OS developers have formulated a set of predefined steps that a developer has to go through before an app is made available for installation. These steps are defined using a marketplace interface where Android uses a Google Play store and iOS uses an Apple App store. The time and resources used in analysing the functionality of a third party application is charged by the OS developer in terms of a subscription fee and a 30% profit share from the app developer [1]. Alongside this, an app that is being pushed to the marketplace interface is thoroughly checked for bugs or flaws. If the reviewer finds a bug, the app is declined publishing and a feedback is provided for improvement and re-submission. This process is repeated until and unless the reviewer is satisfied. This process is also triggered every time an update or a new version is pushed for the application. This review process, though key, imposes a delay factor that might not be necessary every time. The delay factor differs from one authority to another and depends heavily on the available capacity of the reviewer. Apple used to take up to four weeks before an publishing an app and this time could be longer if the app had to go through multiple review iterations. Since then, Apple has improved its review process with apps being approved in just two days [1]. Similarly, Google play store app can be approved in a few hours. Despite this improvement, the development community is pushing for a quicker app approval process with the added feature of reduced payable charges [2] [3]. At current, the review time and app publishing charges are lower bounded by the centralised nature of the process, where all requests are handled in a queue by designated authorities.

Structure of this paper. Section II shows the motivation and contribution of this work. In sections III and IV, of this paper includes a run through current knowledge and survey over relevant works, while explaining what has been added and improved in this paper. Then in section V, the components

Identify applicable funding agency here. If none, delete this.

for system are shown and their selections are justified in each part. After that in section VI, the details of the system implementations are given and highlighting the key features of this work. Next, section VII contains the test results and their comparisions. Last and not least, section VIII will conclude this work and discuss potential future works.

II. Motivation

In this work, the limitations of the centralised approach are overcome by the introduction of a distributed solution based around Ethereum Blockchain architecture. The proposed solution exploits the well known characteristics of Blockchain such as, faster transaction time, greater security and effective cost to automate the verification and validation process in an application development environment. The Blockchain architecture is coupled with Inter-planetary File System (IPFS) architecture to overcome the limitation of handling large file sizes in a Blockchain. IPFS system is used to store all files and it allows the exchange between peers [4]. The blockchain uses smart contracts to store files hashes associated information about an app. As a result, the combined solutions reduce the burden with a centralised architecture and improves the app acceptance time.

The contribution and novelty of this paper consist of the following:

- The proposed solution reduces app approval time and eliminates the need of a centralised app review process.
- The proposed system reduces the hefty review fees by automating the process with minimal human intervention.
- The use of a Blockchain architecture allows the possibility of having a distributed ledger maintaining developers information across all OS platforms and thus introducing the prospect of developers ranking and ratings.
- The system allows developers and reviewers to keep a track of who developed an app and who is sending an update, thus ensuring data authenticity and provenance.

III. Background

The two major decentralized platforms used in this work are IPFS, and Ethereum blockchain using smart contracts. Interplanetary File System is built on Linked-Data and Ontologies science focusing on identifying files based on the content rather than the location. The ability to have a performant link between data is achieved by using Direct Acyclic Graphs (DAGs), or more specifically Merkle-DAGs. This approach is different than the Unified Resource Identifiers (URI), which is incorporated by HTTP web protocols. The difference is that URIs identify a resource based on its location, while Merkle-DAGs identify a resource based on its content. The Merkle-DAGS create unique hashes according to the files' contents, and those hashes are translated back using the Content Identifiers (CIDs) on nodes. CIDs are used by some Big repository-based systems e.g. Git system, that optimized the efficacy of CIDs. After that, decentralization is then enabled on IPFS environment by using Distributed Hash Tables (DHTs), to act as a distributed mapper on nodes, which leads to a peer-to-peer (P2P) files distribution. This distribution mechanism is potentially on par with the current centralized web mechanisms i.e. URIs used in web protocols, that is considered to be cumbersome with the advancement demands from users. IPFS is one of the proposed solutions to this looming issue. The history of blockchain goes way back before the name blockchain itself becomes the official name for this technology; All the efforts precedent to 2002, were focused on how to achieve fully distributed ledgers technologies (DLTs). After various attempts on making several DLTs, many different classifications were created. One of those specialized classifications laid down the ground for blockchain [5]. Thence, blockchain is an improved categorized form of DLTs, where it focuses more on creating decentralized trust and a linked timestamp between blocks to preserve data provenance. Moreover, a second generation of blockchains raised with the advent of smart contracts, which was initiated on Ethereum blockchain.

Ethereum is currently using Proof-of-Work (PoW) consensus mechanism, and having other consensus mechanisms in the testnets. In this work, we tested on two testnets. The first one is Ropsten testnet, which is the most used testnet simulating the mainnet, with PoW. And, the second testnet is Rinkeby testnet, with the Proof of Authority. There are many other consensus mechanisms such as Proof of Stake (PoS), which is based on staking rather than using resources for computational power. The blockchain from Facebook i.e. Libra is based on Proof of Stake, and Ethereum is also upgrading to PoS to enhance the system performance and scalability. Secondly, there is Delegated Proof of Stake (DPoS), and its main purpose is to keep Proof of Stake more decentralized as Proof of Work by delegating votes, and Delegated Proof of Stake is considered to be an intermediate solution between Proof of Work and Proof of Stake that contains best of those two consensus mechanisms. Last and not least, there is the Zero Knowledge Proof (ZKP), the most promising consensus mechanism, that aims to provide a mean for light weight devices to be able to participate in a blockchain network consent process by implementing a method of verification without the need for the actual knowledge of data. Hence, named Zero Knowledge Proof (ZKP). It is worth mentioning that the Ethereum community has acknowledged ZKP, and applied an implementation for it on Ethereum, by using a plug-in called Zokrates for Ethereum smart contracts. Recently, this enables a room for future work to enhance our system by implementing the Zokrates component. Smart Contracts are simply a distributed code between participants, and it is following blockchain consensus mechanisms to verify data and interactions in separated nodes. The decentralization benefits of blockchain and its characterizations of Smart Contracts opened a new field to solve many problems.

IV. Related Works

In the last few years and especially after the big escalation of Bitcoin, a lot of research were conducted on decentralization security and blockchain to synthesize new capabilities, and to

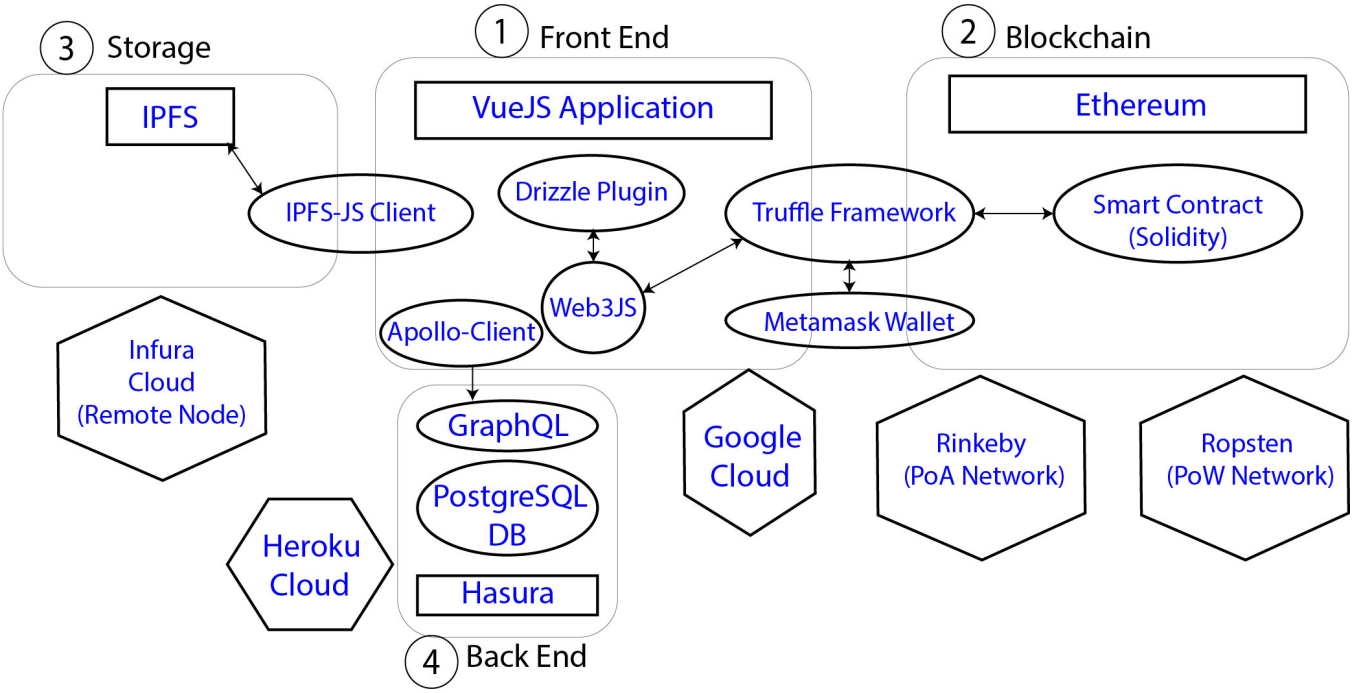

Fig. 1: System Design Overview

enhance the available structures. Some of those latest novice works contributed in founding and forming good progression for more opportunities. In the following, we will examine some of those key researches, their laid groundwork, their results and contributions.

In the first place, the authors in [6] provided a decentralized system, to handle packages management systems. Our proposed work system design corresponds a little to their system design, since development packages are close to software applications; Applications are divided between phases i.e. application development phase that includes development packages as in [6], and application deployment phase as a closed bundle. Which means that any software can mainly be divided into two sections. The programming section during development, and the production section. Thus in [6] the work is proposing to manage packages, during the development phase. While, the second phase, the deployment for production phase is used as our primary focus in this proposed work. Additionally, the system design in our proposed work is more complicated than the system design in [6]. Our smart contract is more programmatically advanced, has a deeper data structure, and miscellaneous sophisticated components are integrated in the system to enhance the structure.

Since blockchain storage is a major issue for many systems, many previous works used IPFS as a complementary component along with blockchain in their design [7], [8], and [9]. In [7], the author has used IPFS to store the transactions of the blockchain as images in order to reduce the storage size of those transactions on blockchain. The authors focused on reducing the storage size on the blockchain. However, this work is addressing the challenge of storing large files as the blockchain does not provide this capability. In [8], the authors are focused on integrating Internet of Things (IoT) with IPFS to apply a decentralized storage to address the lack of storage space in IoT. However, they focused on the issues in the storage only by using IPFS, while the performance was not addressed by using a technology like blockchain. In [9], the authors are using blockchain and IPFS to manage education documents such as certificates in a secure manner. However, the data saved on the blockchain are basically the files' hashes, which are very simple data. In the proposed system, a more complex data structure is saved on the smart contract to handle applications version, with address-based roles and access control providing additional security measures against unauthorized access. In addition to providing the functionality to create data, view data, upgrade, and delete data, a proper check is in place to allow only the authorized users for those actions. The authorities on the network with the right credentials, can approve or deny the transactions on Ethereum network, and in this work adding as well as modifying apps data to the smart contract during the approval process. This gives the end-user the necessary security at a lower cost, and also a quicker time for developers to release any new apps. Last and not least, in the mean time the idea of distributed and decntralized application store is unprecedented in other works.

V. System Design

This section will explain the detailed architecture and the associated technologies that are used in the proposed work.

Figure 1 shows the overall system that is proposed in this work. It consists of:

- Front end application
- Blockchain
- Storage
- Back end database

It is notable that the most complex sub-systems with several components, while interacting together with other components, are present in the first two sections.

A. FrontEnd Application

As shown in Figure 1, the VueJS framework programming language was chosen for the frontend development against ReactJS due to its simple implementation. In addition, the Truffle framework was adopted due to its compatibility with the Ethereum environment. In order to manage the application's state and to keep all the data in VueJS synced, the Drizzle plugin was added as shown in the diagram. We have used IPFS-JS client library to communicate with IPFS by its protocol. To connect to the backend database server, the Apollo client library is used. The backend is based on Graphql interface and Apollo client library is designed to interact with Graphql APIs Additionally, a wallet has to be used when working with blockchain platforms to envelop the accounts, and their private/public keys. The default choice for this setup was the MetaMask wallet to interact with Truffle framework, VueJS framework, and blockchain.

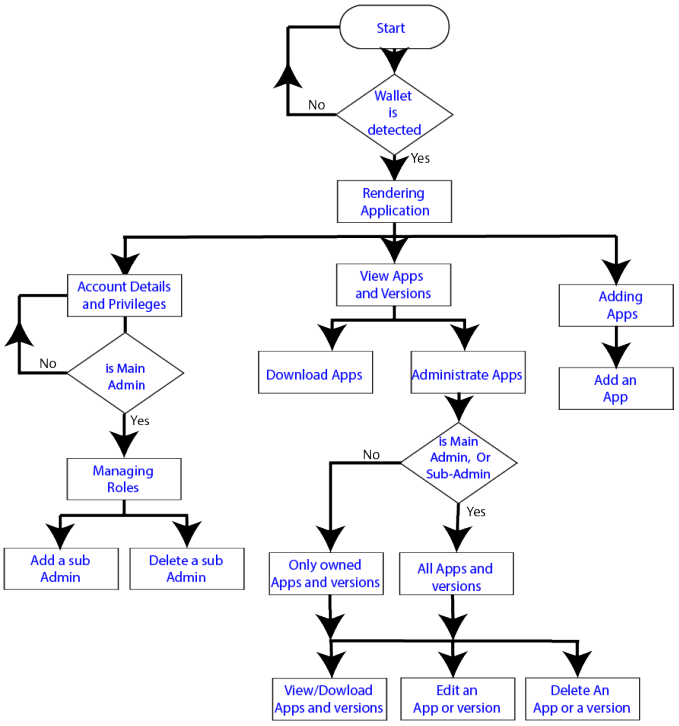

Fig. 2: Application Flowchart

B. Blockchain (Ethereum Platform)

The blockchain is the first major part to acheive a very fast secured system for applications approval. Ethereum was chosen as the blockchain platform for this work. The most significant reason of this selection is that Ethereum has blockchain developers on its network equal to at least four times any other blockchain network [10]. Thus, the majority of smart contracts are built on Ethereum using the Solidity programming language, which is now the most used language for blockchain. In addition, Ethereum is the most matured blockchain platform for smart contracts and is supported by miscellaneous communities.

C. IPFS (Decentralized Storage Platform)

One of the major shortfalls of blockchain is storing large amounts of data on its network. Therefore, a separate storage platform is required and preferably decentralized to keep the intrinsic objective of this work. Hence, the reason in choosing the IPFS, which is the second major part in the system, for this project as shown in Figure 1. The interoperability between IPFS and blockchain will be based on storing the files on IPFS while keeping the hash of the file on the blockchain as a key. IPFS is a decentralized system, and inherently uses the hash schema to point to the files on the platform [11]. There are two competitive platforms that can replace IPFS i.e. Storj, and FileCoin. However, currently, those platforms are still not fully released and yet to come into the real-world applications.

IPFS does not allow for files to be redundant on their network by generating a hash uniquely to the content of the file [11]. In addition, IPFS is a peer-to-peer network that during downloading a file from the network will divide the files into chunks so that they can be downloaded from different nodes in parallel. Once the chunks are downloaded the whole file can be accumulated.

D. Backend Database (Monitoring Performance)

In order to evaluate the performance improvements a database is created to record the time of the transactions on the used Ethereum testnets. as a performance indicator. If desirable, other performance indicators can be added to the backend database as extensions. The database used is a relational database i.e. PostgreSQL. Adding to this database, GraphQL layer, which is application programming interfaces (APIs) layer, was selected against Restful APIs layer and XML Soap layer. The graphQL is a latest promising technology emerged recently [12]. The transmitted data format is, as well, different from the XML, and JSON formats, even though it was originally derived from the JSON format.

VI. System Implementation

Figure 2 shows the flowchart for the implementation of the overall framework.

A. Frontend Application

The front end application is built using VueJS. VueJS is a single page application development framework based on JavaScript, and it focuses mainly in separating the logic into components for reusability. The division of business logic into dedicated components is known as a separation of concerns, and is considered to be a best practice. In addition, as shown

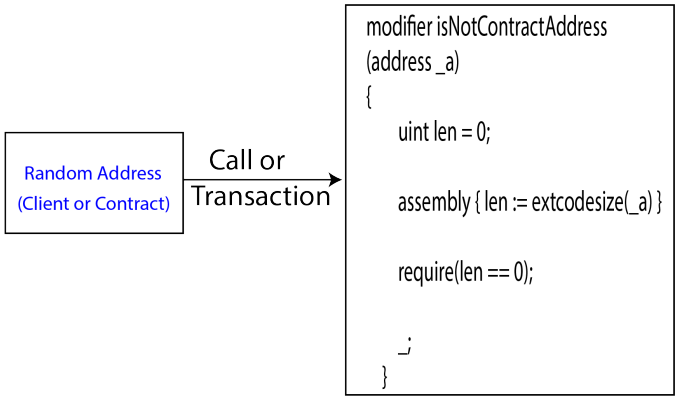

Fig. 3: Smart Contract Defensive Condition

in Figure 2, to keep the app state and memory in sync, the drizzle plugin is used, and this is achieved specifically due to its compatibility with Web3JS functionalities and VueJS state management.

B. Testnets

Interaction with the mainnet will cost real Ether (ETH), and the mainnet is not available for testing purposes. Therefore, the initial development was evaluated on a local test environment using Ganache application that simulates the blockchain environment and its necessary computations. Then, at a certain point, a testnet be used for proper testing, and Ethereum has many different testnets. MetaMask wallet was used in this work to allow access to those testnets.

There are four testnets that are accessible in Metamask, namely are Ropsten, Rinkeby, Kovan, and Goerli testnets. These are mainly the most popular testnets on Ethereum. The consensus mechanism in Ropsten is PoW, while in the other three is PoA.

C. Smart Contract and Solidity Language

In order to write smart contracts on Ethereum blockchain, the two main widely available programming languages are Solidity and Vyper. Solidity is the most developed language and is used mainly on other popular blockchain platforms. Hence, in this proposed work the smart contract was developed using the Solidity language.

Many of the smart contracts attacks are focused on programming flaws [13], and on the simplicity of Solidity language. We ensured appropriate mechanisms accordingly in our proposed system. In the proposed implementation, a condition is added in the smart contract to prevent other smart contracts from calling our smart contract. This condition, by using the assembly language which is an option in Solidity for advanced coding, is shown in Figure 3. This condition checks during a trasanction if the sender address has an actual code size or if the size is equal to zero. If the address has code size bigger than zero, then this the address belongs to a smart contract, and the condition will not permit such an address. By stopping other smart contracts from interacting with the smart contract, the smart contract is preserved safe from major security attacks e.g. reentrancy attack [13]. This type of protection can stop other automated machine based passive attacks on the smart contract. However, since this condition stops only other smart contract request, this means that an attack can still be automated using a client address from client sources e.g. web applications. To avoid any client application misuse or attack, in the proposed work a control was added to the smart contract to allow only for one transaction at any specific time from any client. As shown in Figure 4 , this control adds a flag for every client to indicate whether a client is still processing a transaction and if he is being locked or not. After a transaction is finalized, the flag will get reset to allow the client to perform another action. On one hand, we handled attacks from other smart contracts with the first condition. And on the other hand, we stopped clients from misusing the system with the second method.

Many of the blockachain security attacks are from the smart contract implementation and from the exploitation of the Solidity programming language.

1) Access Control : In this proposed work, some access control measures is applied to the smart contract by using features available in the Solidity programming language. Solidity is a complete Turing machine based programming language, which means programmatically it is capable of incubating any intended logic. In order to protect the access rights on the smart contracts, manually some authentication mechanisms were implemented on the code to validate the user privileges.

Figure 5 represents the access control work flow for the smart contract.

2) Code: In this work modifiers are used. This is a feature that is available in Solidity to lower the calculated gas of the deployed opcodes, i.e. operation codes.

As shown in Figure 5, three different user types are created based on their addresses. The first user type is based on the main administrator address, and in the proposed work, is the creator of the smart contract. The second type of addresses are sub-admins, that resembles authorities in the store. The final type are the regular users addresses. Simply, the data saved on

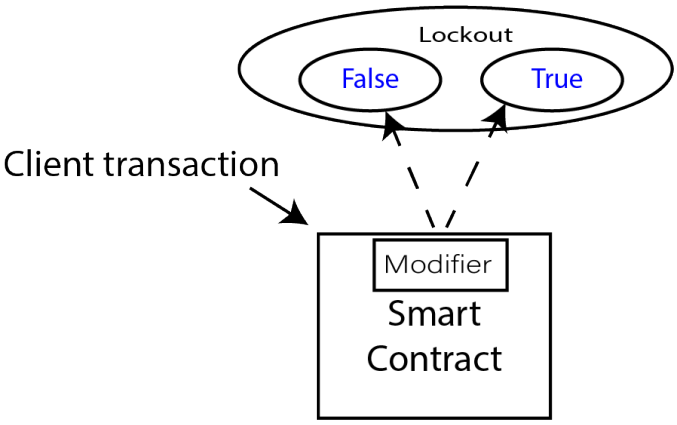

Fig. 4: Smart Contract Transactions Control

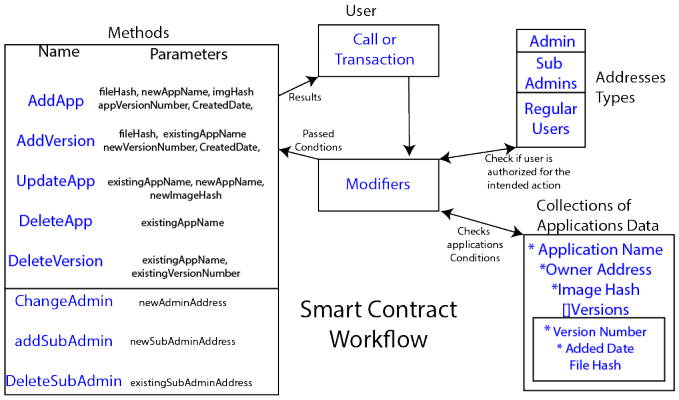

Fig. 5: Smart Contract Work Flow

the smart contracts constitutes of arrays of application details. In each application details, there are the name of the app, the address of the owner, a hash key for the app image that is saved on IPFS, and an array of versions of the app as different files.Two types of methods are used, methods that manage the users' roles, and methods that manage the application data in the smart contract. Accordingly, any call or transaction going to the smart contract undergoes a small check, through modifiers, to evaluate the request. If the request passes, the dedicated method will send the required results back.

D. Backend Recording

In order to estimate the actual time of each transaction, a backend database is created. The start and end times for each transaction will be saved in the database to calculate the transaction time to measure the performance. The average value of those records or readings will give an indication of the time that was taken for the various tasks such as the add, modify and delete actions on the smart contracts.

VII. Results

Two testnets were used, namely Ropsten and Rinkeby. We tested the speed of verifications of transactions and the cost those transaction requires i.e. gas cost, which is the internal unit used in Ethereum. There is a difference between the Rinkeby and Ropsten in the transaction verification time, since each one is using a different consensus mechanism. And the smart contract handles five kinds of transactions, which are adding a new app, adding an updated version, editing an app, deleting a version, and deleting an app.

First for the gas estimate, as shown from Figure 6, both testnets have close results. While in general, by comaprison from Table I, Ropsten is being little higher in cost than Rinkeby network. In both of them, adding new app information to the blockchain do cost the most, and following is the app editing. The least cost is when removing an app version and deleting the whole app. It is also shown that Rinkeby is not only lower in cost, but also has more stable results. Also from Figure 6. the transactions times are shown for both networks. Again by comparing the results as shown in Table II, Ropsten network was the slower one. This was due to Ropsten using the PoW mechanism which is computationally costly since the conducted trust and votings are being based on intensive computations from the participants. In addition Ropsten results are more unstable as they are based on PoW which suffers from high variance of validations performance because the contributors in those validations are in continuous competition to get the reward. From these observations, a blockchain will be more stable if based on PoA or similarily PoS.

VIII. Conclusion and Future works

A. Conclusion

Using IPFS decentralized storage as a method to allow users to download applications will give a better performance than depending on the clients and servers method. Instead of downloading the application from a dedicated server, which can be in some cases located far away from the user, downloading divided chunks from different users will optimize the network utilization and the download time will be much less for the user. Moreover, the availability and capacity will be improved with minimum costs. Secondly, implementing blockchain in the system, improves several security aspects for applications deployment systems, and allows a room for more contributions and delegations between different authorities to handle the platform in a consensus mechanism. In this use case particularly, it will be ideal to use Proof of Authority, since it has a better performance and suits the scenario to authorize partners in the validation process. Moreover, Proof of Work in Ethereum is getting really slow with growing drawbacks on the applications [14]. Overall, the system proposed in this work demonstrates the promise of the decentralized systems in enhancing the time to deploy a mobile application. The cost required in most of those platforms will be significantly decreased e.g.in App and Google Play stores. It will satisfy the ongoing demand from the users and provide the flexibility required by the community to increase the speed to deploy new apps and also reduce the commission costs the app owners have to pay to the app stores.

B. Future Works

It is observed that working on software deployment systems is considered to be relatively a rudimentary field without much experienced work involved in it. And decentralization is still a novel discipline while under-going a lot of rapid contributions and development recently. Hence, there are lot of rooms for improvements. We plan to extend this work to integrate different proprietary consensus mechanisms. The first most likely one to be Zero Knowledge Proof due to its lightweight nature. Furthermore, with the exponential increase of applications numbers in stores, a systematic decentralized way to scan any software for malicious code, as a precedent phase before deployment, will lubricate the objective of complete automation process. By the same token, checking copy rights and data provenance is also a favorable add-on for the system, and this can be a great prospect and empirical case for machine learning usability. Last and but

Fig. 6: Ropsten & Rinkeby Transactions Comparision

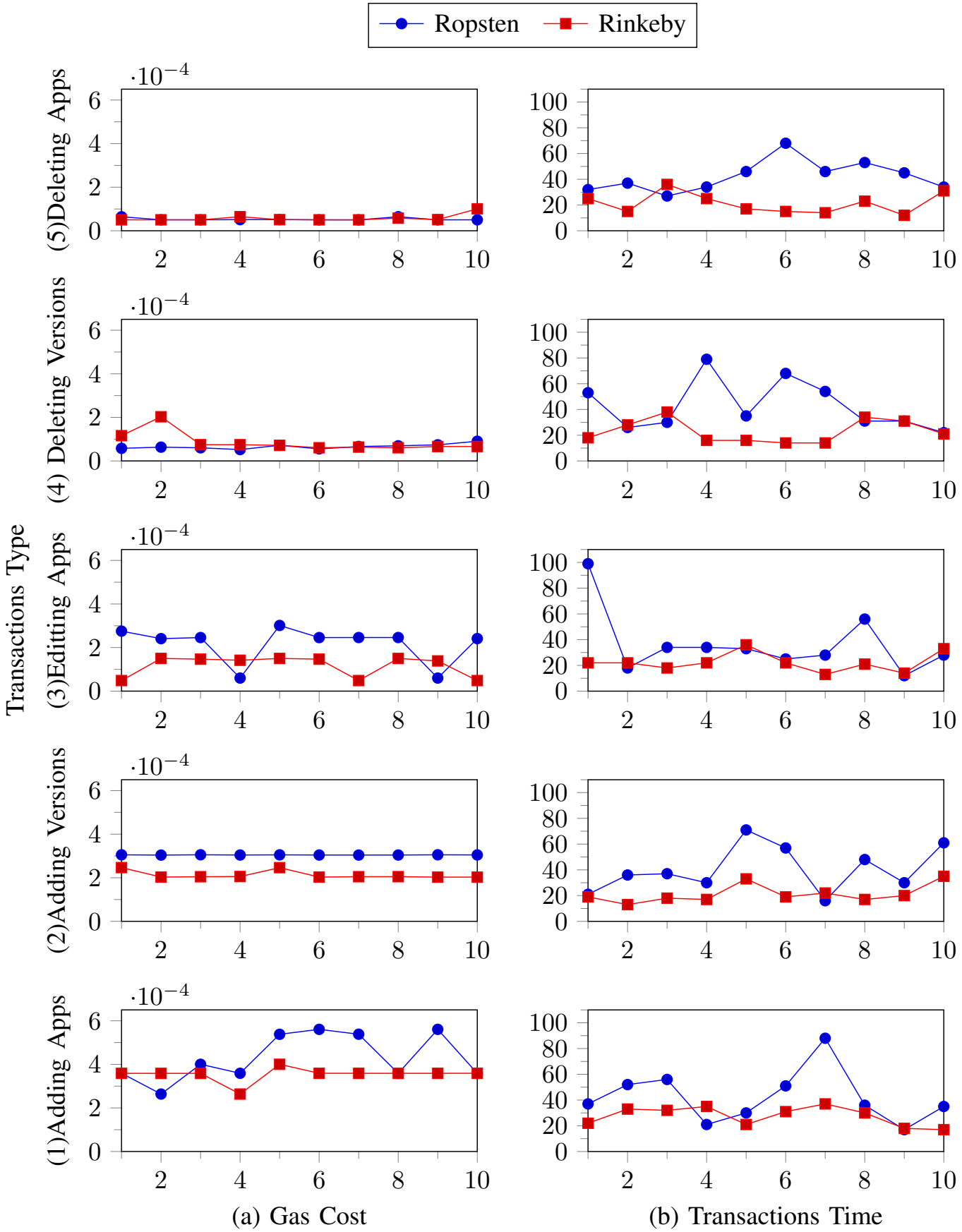

TABLE I: Transactions Gas Costs Comparison

Transaction	Network	1	2	3	4	5	6	7	8	9	10
Adding an App	ropsten	3.590	2.636	4.003	3.590	5.379	5.606	5.382	3.587	5.606	3.585
	Rinkeby	3.588	3.586	3.585	2.6365	4.002	3.590	3.589	3.588	3.588	3.590
Adding a version	ropsten	3.053	3.040	3.054	3.043	3.050	3.053	3.040	3.054	3.043	3.050
	Rinkeby	2.465	2.031	2.0465	2.055	2.463	2.031	2.047	2.049	2.029	2.029
Editting an App	ropsten	2.75	2.409	2.459	0.599	3.014	2.458	2.460	2.460	0.600	2.411
	Rinkeby	0.484	1.501	1.467	1.417	1.501	1.468	0.484	1.500	1.386	0.0484
Deleting a Version	ropsten	0.0579	0.663	0.603	0.522	0.719	0553	0.663	0.695	0.740	0.904
	Rinkeby	1.160	2.028	0.746	0.745	0.716	0.605	0.603	0.605	0.661	0.661
Deleting an App	ropsten	0.643	0.502	0.502	0.511	0.0502	0.501	0.500	0.064	0.501	0.502
	Rinkeby	0.501	0.502	0.502	0.646	0.511	0.502	0.501	0.508	0.512	1.005

TABLE II: Transactions times Comparison

Transaction	Network	1	2	3	4	5	6	7	8	9	10
Adding an App	ropsten	37	52	56	21	30	51	88	36	17	35
	Rinkeby	22	33	32	35	21	31	37	30	18	17
Adding a version	ropsten	21	36	37	30	71	57	16	48	30	61
	Rinkeby	19	13	18	17	33	19	22	17	20	35
Editting an App	ropsten	99	18	34	34	33	25	28	56	12	28
	Rinkeby	22	22	18	22	36	22	13	21	14	33
Deleting a Version	ropsten	53	26	30	79	35	68	54	31	31	22
	Rinkeby	18	28	38	16	16	14	14	34	31	21
Deleting an App	ropsten	32	37	27	34	46	68	46	53	45	34
	Rinkeby	25	15	36	25	17	15	14	23	12	31

not least, integrating a robust authentication module for users, with a second component, will supply users with authorization capabilities in smart contracts, and will wrap up the security of blockchain communications with web applications. One of the best authentications is JWT based authentication, and there is a plausible sense to decompose JWT based authentication methodology to employ it in a decentralized platform and fine-tune its advantages. Overall, decentralization direction is evolving with several future possibilities.

REFERENCES

[1] P. Roma and D. Ragaglia, "Revenue models, in-app purchase, and the app performance: Evidence from Apple's App Store and Google Play," *Electronic Commerce Research and Applications*, vol. 17, pp. 173–190, may 2016.

[2] N. Statt, "Tinder is now bypassing the Play Store on Android to avoid Google's 30 percent cut - The Verge," jul 2019. [Online]. Available: https://www.theverge.com/2019/7/19/20701256/tinder-google-play-store-android-bypass-30-percent-cut-avoid-self-install

[3] J. Padhiyar, "Why Apple's App Store is Charging 30% Fees and How is it Justified?" jul 2018. [Online]. Available: https://www.igeeksblog.com/why-app-store-is-charging-30-percent-commission/

[4] Y. Xu, "Section-blockchain: A storage reduced blockchain protocol, the foundation of an autotrophic decentralized storage architecture," in *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, vol. 2018-Decem. Institute of Electrical and Electronics Engineers Inc., dec 2018, pp. 115–125.

[5] Serokell, "Blockchain vs. Distributed Ledger Technology - What's the difference?" jan 2020. [Online]. Available: https://medium.com/better-programming/blockchain-vs-distributed-ledger-technology-whats-the-difference-2587d9780c99

[6] G. D'mello and H. González-Vélez, "Distributed Software Dependency Management Using Blockchain," in *Proceedings - 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2019*. Institute of Electrical and Electronics Engineers Inc., mar 2019, pp. 132–139.

[7] R. Kumar and R. Tripathi, "Implementation of Distributed File Storage and Access Framework using IPFS and Blockchain," in *2019 Fifth International Conference on Image Information Processing (ICIIP)*. IEEE, nov 2019, pp. 246–251. [Online]. Available: https://ieeexplore.ieee.org/document/8985677/

[8] S. Muralidharan and H. Ko, "An InterPlanetary File System (IPFS) based IoT framework," in *2019 IEEE International Conference on Consumer Electronics, ICCE 2019*. Institute of Electrical and Electronics Engineers Inc., mar 2019.

[9] A. K. Shrivastava, C. Vashistth, A. Rajak, and A. K. Tripathi, "A Decentralized Way to Store and Authenticate Educational Documents on Private Blockchain," *2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pp. 1–6, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8977633/

[10] ConsenSys, "Ethereum Has 4x More Developers Than Any Other Crypto Ecosystem," 8. [Online]. Available: https://consensys.net/blog/blockchain-development/ethereum-has-4x-more-developers-than-any-other-crypto-ecosystem/

[11] J. Benet, "IPFS - Content Addressed, Versioned, P2P File System," jul 2014. [Online]. Available: http://arxiv.org/abs/1407.3561

[12] G. Brito, T. Mombach, and M. T. Valente, "Migrating to GraphQL: A Practical Assessment," in *SANER 2019 - Proceedings of the 2019 IEEE 26th International Conference on Software Analysis, Evolution, and Reengineering*. Institute of Electrical and Electronics Engineers Inc., mar 2019, pp. 140–150.

[13] A. Dika and M. Nowostawski, "Security Vulnerabilities in Ethereum Smart Contracts," in *Proceedings - IEEE 2018 International Congress on Cybermatics: 2018 IEEE Conferences on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, iThings/Gree*, 2018.

[14] X. Liu, G. Zhao, X. Wang, Y. Lin, Z. Zhou, H. Tang, and B. Chen, "MDP-based quantitative analysis framework for proof of authority," in *Proceedings - 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2019*. Institute of Electrical and Electronics Engineers Inc., oct 2019, pp. 227–236.