

Blockchain Powered Automated Crowdfunding via Smart Contracts

Prajwal Chandrakant Kamble¹, Santosh Borde² and Atharva Chandrakant Kamble³

¹⁻²JSPM's Rajarshi Shahu College of Engineering, Pune, India

Email: {prajwalkambleofficial, spraoborde}@gmail.com

³J D College of Engineering and Management, Nagpur, India

Email: atharvakamble2004@gmail.com

Abstract—Crowdfunding, a mechanism for amassing capital from numerous contributors, has faced challenges due to reduced investment and a lack of transparency. Efforts to address these issues have proven largely unsuccessful. However, blockchain technology offers a potential solution, enabling enhanced transparency in transactions. This study investigates the perception of blockchain among small and medium enterprises (SMEs), a domain where its implementation remains relatively unexplored. Through a survey of 300 SMEs in Italy, the research assesses knowledge, diffusion, and interest in blockchain. Findings indicate a moderate awareness of blockchain, coupled with limited understanding. Despite low adoption rates, there is optimism regarding future implementation, suggesting the transformative potential of blockchain in reshaping SME business models and processes.

Index Terms— Crowdfunding, Blockchain, Ethereum, smart contract, fundraiser.

I. INTRODUCTION

In the realm of online crowdfunding platforms, there exists a pressing need for an alternative methodology to address inherent shortcomings. Presently, crowdfunding websites host myriad projects soliciting funds from contributors who donate at their discretion. However, a glaring flaw lies in the lack of trust engendered by this system. Instances abound where funds are diverted for purposes other than the intended project, betraying the trust of contributors. Moreover, centralized servers pose a vulnerability to malicious activities, exacerbating concerns surrounding transparency, intellectual property, fund centralization, and reputational risks. Investors clamor for greater transparency, yearning to trace the trajectory of their funds and their tangible impact on projects. To redress these deficiencies, the incorporation of Ethereum Blockchain emerges as a pivotal solution, promising heightened transparency and security within the crowdfunding domain.

A. Types of crowdfunding

Donation-Based Crowdfunding: It entails contributions without any expectation of financial return, commonly associated with charitable causes, NGOs, disaster relief efforts, and medical assistance initiatives[1].

Rewards-Based Crowdfunding: It involves individuals contributing to projects in exchange for rewards, which may include monetary profit or tangible products. Numerous platforms leverage this model for crowdfunding campaigns.

Equity - Based Crowdfunding: In contrast, it offers investors an opportunity to acquire shares in the company.

Participants in this type of crowdfunding stand to receive returns based on the company's profitability, allowing them to become stakeholders in the venture.

B. Present-day crowdfunding

Presently, crowdfunding transactions rely heavily on diverse crowdfunding platforms, imposing substantial fees on both investors and contributors for processing requests, often yielding suboptimal outcomes. These platforms often function as gatekeepers, imposing stringent regulations that impede the freedom of both investors and contributors in fostering project success. Merely possessing a groundbreaking idea on a crowdfunding platform does not guarantee success; users must employ strategic tactics to enhance the visibility of their crowdfunding page on search engines and attract new customers, necessitating significant investments in advertising. Furthermore, many crowdfunding platforms fail to ensure the fulfillment of promises to contributors, fostering skepticism and hindering investment, thereby posing challenges for project managers. Additionally, project managers frequently encounter setbacks, witnessing the collapse of their entire enterprise before production commences, as the popularity of ideas on crowdfunding websites inspires emulation among various entrepreneurs, intensifying competition.

C. Blockchain

A blockchain is an evolving sequence of interconnected blocks, linked via cryptographic methods. Each block computes a hash utilizing cryptographic algorithms, incorporating the hash of the previous block, a timestamp of its creation, and value-based data, typically organized in a Merkle tree structure [2]. Immutable by design, a blockchain prevents unauthorized modifications to block data; altering recorded information necessitates the consensus of over half the nodes in the network, typically 51 percent. Functioning as a shared and immutable ledger, blockchain affords transparent access to its data, rendering it decentralized. Consequently, blockchain circumvents the need for centralized servers or trusted intermediaries, thus addressing issues of excessive expenditure.

Blockchain security relies on two primary techniques: Proof of Work (PoW) and Proof of Stake (PoS). In PoW, miners compete to generate a nonce, a time-consuming process requiring significant computational power and energy consumption. However, nonce validation is straightforward, meeting specific criteria [3]. Miners in PoW systems compete to validate transactions and receive rewards. Conversely, PoS requires miners to stake assets, earning rewards upon successful mining; failure results in loss of staked assets.

Public and private blockchains constitute the two primary variants. Public blockchains, accessible to all, enable participation, reading, and writing within the network, fostering a distributed and decentralized ecosystem. In contrast, private blockchains cater to organizations seeking distributed ledger technology implementation while maintaining data privacy. Ethereum exemplifies a public blockchain, characterized by decentralized frameworks and autonomy from external control [4]. Smart contracts, integral to blockchain functionality, automate contract execution, enforcing predefined rules and penalties. These contracts, written in the Solidity programming language, facilitate transparent, irreversible, and trackable transactions.

D. Crowdfunding using blockchain

In crowdfunding, blockchain technology enables decentralization, ensuring that smart contracts are transparent and beyond the control of any single entity or group of platforms. Operating as a peer-to-peer network, it adheres collectively to a protocol for inter-node communication and block validation. Consequently, no alteration to a block is possible without the consensus of over 50 percent of nodes, ensuring security and integrity. Projects can be created and accessed by anyone with internet connectivity, streamlining the donation process. Unlike traditional crowdfunding, contributors are relieved of concerns regarding unfulfilled promises, as smart contracts manage all transactions, securely storing funds within the blockchain. This decentralized approach grants greater autonomy to project managers and contributors, facilitating fractional contributions and fostering a sense of empowerment.

III. PROPOSED MODEL

In this study, the Ethereum blockchain has been selected due to its versatility across numerous applications. Smart contracts within Ethereum are primarily developed using the Solidity programming language. These contracts function as automated accounts governed by code rather than human intervention. They encompass various fields, including balance to monitor Ether holdings, storage for data retention, and code containing the contract's machine code.

Contracts are authored in Solidity, and upon compilation using the Solidity compiler, two distinct files are generated. The first file comprises a bytecode for deployment onto the Ethereum network, while the second file, known as an Application Binary Interface (ABI), serves as a crucial interface facilitating interaction between web applications and deployed contracts. The ABI plays a vital role in comprehending contract functionalities and testing using JavaScript frameworks, aiding in the development of web applications based on Ethereum smart contracts.

Following deployment, rigorous testing becomes imperative, particularly considering the significant financial transactions involved. Testing ensures that contracts function as intended. Deployment and testing are typically conducted on a local test network, established using the Ganache[5] library (formerly TestRPC), exclusively for contract deployment and evaluation. Conversely, the Web3 library utilizes the ABI to provide programmatic access to deployed contracts on the blockchain.

IV. IMPLEMENTATION

Implementation of a transparent system for project contributors is pivotal in our project. In traditional fundraising applications, project managers collect funds from contributors and independently allocate them, potentially leading to misuse or misappropriation. To address this issue, our project proposes an alternative approach[6].

Once the funding target is reached, the project manager must submit spending requests to contributors for approval before utilizing the funds[7]. Contributors then vote on these requests, with a majority vote determining whether the funds are disbursed directly to the vendor or service provider, bypassing the manager's control. This system ensures greater transparency, empowering contributors to actively participate in fund management.

Workflow:

The workflow of our project is illustrated in Fig 1. Initially, the manager, acting as an actor, logs into the system and creates a project. Registration occurs if it is the manager's first login. The newly created project is added to the list of existing projects. If the funding target is achieved, the project progresses; otherwise, it is terminated. Following target attainment, the manager can generate spending requests for project-related products or services. Contributors vote on these requests, and if the majority votes favorably, the funds are allocated accordingly.

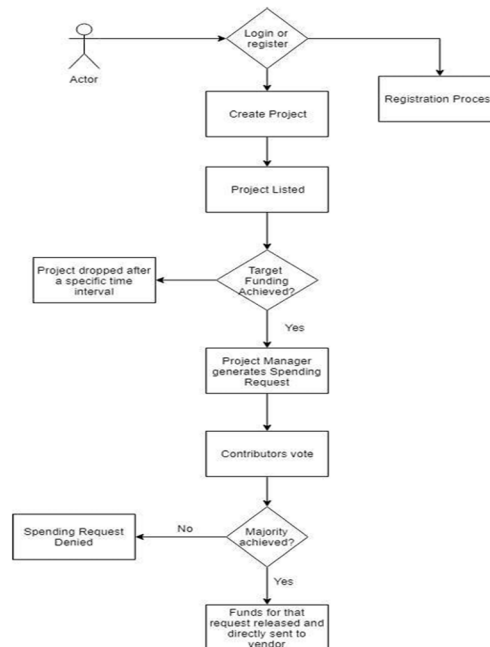


Fig. 1. Workflow of the system

Campaign figure 2 displays a code snippet of our Smart Contract, particularly the Campaign function. The manager's address is defined as a public variable to ensure transparency. Additionally, a minimum contribution requirement is set, allowing users to dynamically adjust their funding preferences.

```
function campaign(uint minimum, address creator) public{
    manager = creator;
    minimumContribution = minimum;
}
```

Fig. 2. Structure of campaign

In Fig 3, the code snippet depicts the implementation of the contribute function. Whenever an individual intends to transfer funds to our contract, the corresponding function must be designated as payable. Upon invocation, it is imperative that the contributed amount exceeds the minimum contribution threshold. In this context, 'msg' is a global variable encompassing properties such as 'sender' and 'value', where 'msg.value' represents the amount of wei (the smallest denomination of ether) transmitted by the contributor to this function[8]. If the value fails to meet the minimum contribution requirement, the transaction will be unsuccessful. Upon meeting the requirement and successful contribution, the contributor is added to the list of approvers[9].

```
function contribute() public payable{
    require(msg.value > minimumContribution);

    approvers[msg.sender] = true;
    approversCount++;
}
```

Fig. 3. Code of contribution

In Fig 4, the request struct is delineated. A struct constitutes a collection of key-value pairs accommodating different data types[10]. The request struct encompasses various fields elucidating the purpose of the request. Notably, 'description' elucidates the rationale behind the request, 'value' denotes the amount of funds the manager intends to allocate for a project, 'recipient' signifies the address to which the funds will be directed, 'complete' is a boolean variable indicating the status of the request (true if processed, false if pending), and 'approvalCount' tallies the number of contributors who have approved the request[11]. The createRequest function serves to instantiate a new struct of type Request and subsequently add it to the request list.

```
function createRequest(string description, uint value, address recipient) public restricted{
    Request memory newRequest = request({
        description: description,
        value: value,
        recipient: recipient,
        complete: false,
        approvalCount: 0
    });
    requests.push(newRequest);
}
```

Fig. 4. Code for request

Figure 5 illustrates the approveRequest function, which is invoked by a contributor to endorse or reject a request[12]. Certain requirements govern the voting system, notably the prevention of multiple votes from a single contributor, necessitating the maintenance of voting records. Moreover, the system must remain functional even with a large number of contributors. The 'index' variable is utilized to discern the request associated with each vote, accommodating multiple concurrent requests. The approveRequest function entails several checks[13]: 1) verification of the contributor's prior contribution, and 2) validation of the contributor's prior vote on the request. Approval is recorded upon successful completion of these checks.

Figure 6 introduces the finalizeRequest function, utilized by the manager to authorize expenditure on the project once a request garners sufficient approval. The 'restricted' modifier confines function access solely to the manager. The function requires an index argument specifying the request to be finalized. Upon successful verification, the boolean variable is toggled to prevent re-finalization of the request. The 'storage' keyword indicates that the 'request' variable references the existing request copy stored in the blockchain.

```

function approveRequest(uint index) public{
    Request storage Request = requests[index];

    require(approvers[msg.sender]);
    require(!request.approvals[msg.sender]);

    request.approvals[msg.sender] = true;
    request.approvalCount++;
}

```

Fig. 5. Code for approving request

```

function finalizeRequest(uint index) public restricted{
    Request storage Request = requests[index];

    require(request.approvalCount > (approversCount / 2));
    require(!request.complete);

    request.recipient.transfer(request.value);
    request.complete = true;
}

```

Fig. 6. Code for finalizing request

V. OUTPUT & RESULT

The proposed model's functionality is demonstrated through the following example. Figure 7 illustrates the creation of a campaign with a minimum contribution requirement of 1000 wei. It's noteworthy that every transaction on the Ethereum network incurs a small fee, borne by the transaction initiator. In this scenario, Account 1 is designated for the manager.

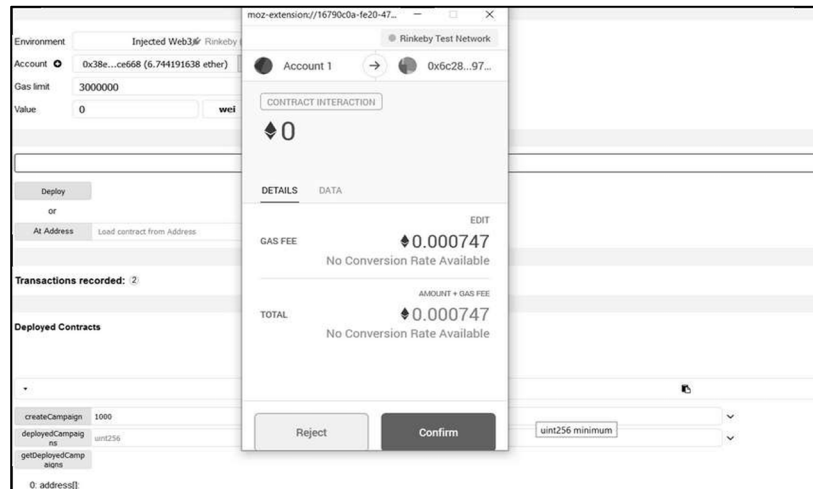


Fig. 7. Creating a Campaign

Subsequently, the manager initiates a request, specifying three parameters: a description of the fund's purpose, the amount required, and the address of the third-party recipient. In Figure 8, the manager requests funds for hiring a freelancer, amounting to 0.25 ETH.

The request undergoes a voting process, as depicted in Figure 8. With five contributors involved in the campaign, a request must secure approval from more than 50 percent of contributors to be accepted. Hence, a minimum of 3 out of 5 approvals is necessary for request acceptance. Figure 9 showcases Account 2's approval of the request.

Upon garnering 3 out of 5 approvals, the manager can proceed to finalize the request, facilitating the transfer of funds to the third-party service provider. Figure 10 depicts the finalization of the request.

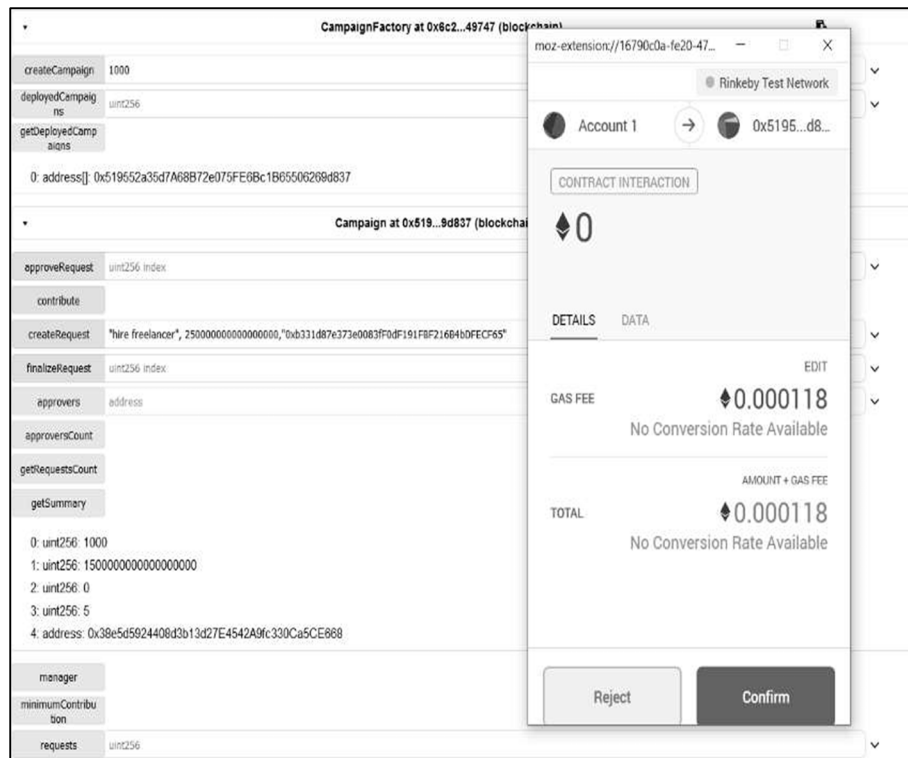


Fig. 8. Creating a request for the campaign

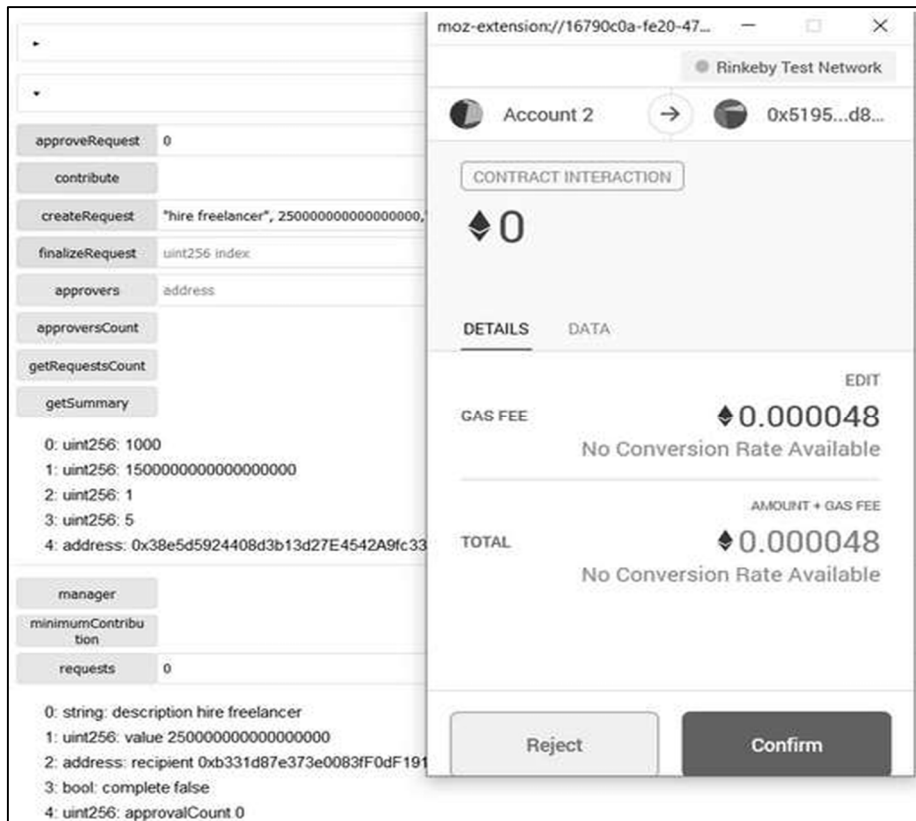


Fig. 9. Approving request for campaign

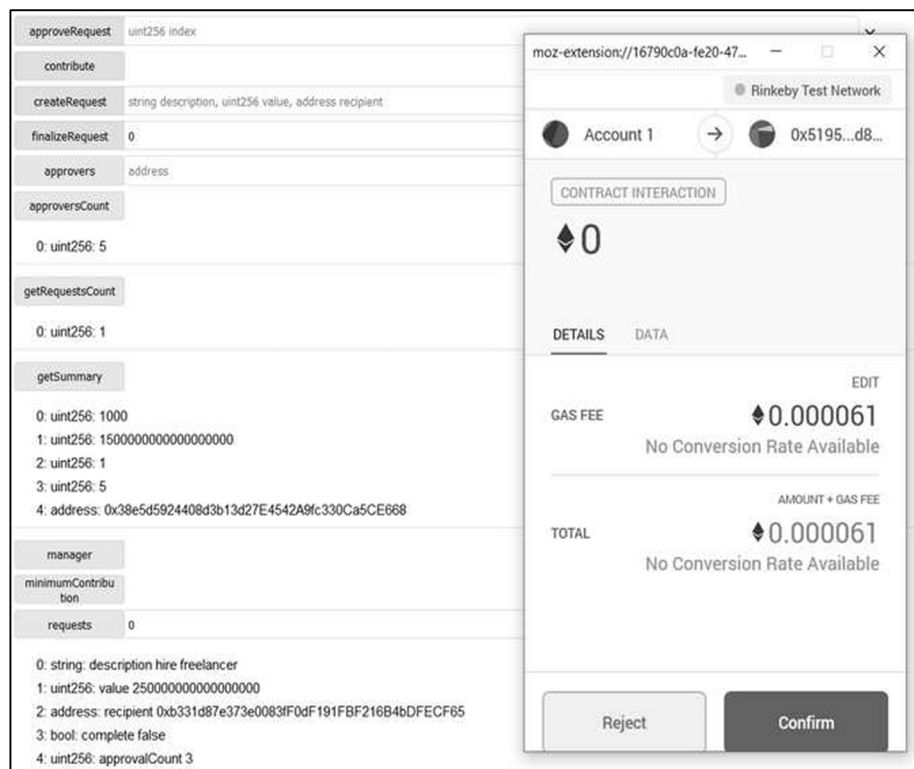


Fig. 10. Finalizing requests

VI. CONCLUSION

The integration of blockchain technology into crowdfunding presents a relatively novel concept within the ICT community. Thus far, Solidity code has been effectively written for the campaign contract and compiled using the Solidity compiler, yielding bytecode. The interface is then deployed onto the Ethereum blockchain via MetaMask. Subsequently, a decentralized web application is constructed, featuring frontend functionalities for project creation, contribution, request initiation, approval, and finalization. However, the current state of blockchain application in crowdfunding remains exploratory, necessitating the resolution of numerous legal and technical challenges.

Looking ahead, our proposed work holds promising prospects and ample room for advancement and refinement. Future research endeavors can further enhance the crowdfunding application's usability and security, facilitating the easier and safer realization of innovative ideas. Additionally, the application of technology in crowdfunding platforms enhances contributor confidentiality by leveraging the transparent nature of transactions and automated contract execution. This fosters trust among contributors, bolstering the platform's integrity. Moreover, the planned system implementation will be complemented by an acceptance study, further validating the platform's efficacy and security measures.

REFERENCES

- [1] A STUDY ON CROWD FUNDING AND ITS IMPLICATIONS IN INDIA, PRINSHA K. PARIPEX-INDIAN JOURNAL OF RESEARCH: Volume-5 | Issue-1 | January-2016.
- [2] Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 2017, pp. 557-564, doi: 10.1109/BigDataCongress.2017.85.
- [3] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun, "A Study on Crowd Funding and its Implications in India Paripex," In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Association for Computing Machinery, New York, NY, USA, 2016.
- [4] D. Vujičić, D. Jagodić and S. Randić, "Blockchain technology, bitcoin, and Ethereum: A brief overview," 2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 2018, pp. 1-6, doi: 10.1109/INFOTEH.2018.8345547.

- [5] Truffle Suite, "Ganache," Truffle Suite. [Online]. Available: <https://truffleframework.com/ganache/>.
- [6] Suporn Pongnumkul, Chaiyaphum Siripanpornchana, Suttipong Thajachayapong, "Performance Analysis of Private Blockchain", 26th International Conference on Computer Communication and Networks (ICCCN), 2017.
- [7] E. Maaten, "Towards remote e-voting: Estonian case", *Electronic Voting in Europe-Technology, Law, Politics and Society*, vol. 47, pp. 83-100, 2004.
- [8] etherscan.io, "Ethereum GAS Tracker," Ethereum Transactions Information. [Online]. Available: <https://etherscan.io/gastracker>. [Accessed: 13-Aug-2018].
- [9] Y. Takabatake, D. Kotani, and Y. Okabe, "An anonymous distributed electronic voting system using Zerocoin", *IEICE Technical Report*, pp. 127-131, 2016.
- [10] "Final report: study on eGovernment and the reduction of administrative burden (SMART 2012/0061)", 2014, [Online]. Available: <https://ec.europa.eu/digital-single-market/en/news/finalreport-study-egovernment-and-reduction-administrative-burdensmart-20120061>.
- [11] F. Hao and P.Y.A. Ryan, *Real-World Electronic Voting: Design, Analysis and Deployment*, CRC Press, pp. 143-170, 2017.
- [12] N. Braun, S. F. Chancellery, and B. West. "E-Voting: Switzerland's projects and their legal framework-In a European context", *Electronic Voting in Europe: Technology, Law, Politics and Society*. Gesellschaft für Informatik, Bonn, pp.43-52, 2004.
- [13] "A next-generation smart contract and decentralized application platform", Ethereum white paper, [online] Available:<https://github.com/ethereum/wiki/wiki/White-Paper>.