A Blockchain-based Method for Decentralizing the ACME Protocol to Enhance Trust in PKI

Elie F. Kfoury*, David Khoury‡, Ali AlSabeh*, Jose Gomez*, Jorge Crichigno*, Elias Bou-Harb†
*Integrated Information Technology, University of South Carolina, Columbia, U.S.A.
‡Computer Science Department, American University of Science and Technology (AUST), Lebanon.
†The Cyber Center For Security and Analytics, University of Texas at San Antonio, U.S.A.

***Abstract*—Blockchain technology is the cornerstone of digital trust and systems' decentralization. The necessity of eliminating trust in computing systems has triggered researchers to investigate the applicability of Blockchain to decentralize the conventional security models. Specifically, researchers continuously aim at minimizing trust in the well-known Public Key Infrastructure (PKI) model which currently requires a trusted Certificate Authority (CA) to sign digital certificates. Recently, the Automated Certificate Management Environment (ACME) was standardized as a certificate issuance automation protocol. It minimizes the human interaction by enabling certificates to be automatically requested, verified, and installed on servers. ACME only solved the automation issue, but the trust concerns remain as a trusted CA is required. In this paper we propose decentralizing the ACME protocol by using the Blockchain technology to enhance the current trust issues of the existing PKI model and to eliminate the need for a trusted CA. The system was implemented and tested on Ethereum Blockchain, and the results showed that the system is feasible in terms of cost, speed, and applicability on a wide range of devices including Internet of Things (IoT) devices.**

***Keywords*—ACME; Blockchain; Ethereum; Public Key Infrastructure; Trust; Certificate Authority.**

I. Introduction

In today's digital age, critical data is constantly sent across the globe through diverse technologies and protocols: Internet of Things (IoT), E-Commerce, E-Government, Instant Messaging (IM), conversational media (Voice over IP/LTE), and several others. Many applications are facing deployment issues due to the lack of proper security and privacy measures [1], including essential sectors such as healthcare [2]. IoT for instance, which includes interconnected low-powered computing devices, has not been widely adopted by organizations and consumers due to security challenges, specially client authentication [3]. Most existing systems are secured through a Public Key Infrastructure (PKI) with a trusted third-party Certificate Authority (CA). The PKI/CA infrastructure depends heavily on its trust model. Trust in this context involves having CAs issue certificates to appropriate users, and include in their "chain of trust" behaved CAs only. Unfortunately, trust remains a critical challenge: in 2011, Diginotar, a Dutch CA, got penetrated by a student who was able to issue hundreds of fraudulent certificates and infect domains pertaining to the CIA, Mossad, Google, Microsoft, Twitter and others [4]. This incident led major browsers and operating systems to remove Diginotar from their list of trusted CAs. Eventually, Diginotar declared bankruptcy the same month that the attack occurred. Furthermore, the attacker claimed to have access to four other CAs, including the reputable Comodo; an unauthorized party got a certificate issued by Comodo for the domain live.fi [5]. Other incidents related to injecting inappropriate CAs into the trusted list have also occurred [6].

A major reason for having trust problems with CAs is centralization [7] [8]. The organization's system represents a single point of failure, which can lead to Denial of Service (DoS) attacks and security breaches.

Besides the aforementioned trust issues, acquiring certificates from CAs can be cumbersome as the domain name verification is done through a collection of ad-hoc mechanisms. This typically accomplished by following interactive natural-language instructions from the CA rather than through machine-implemented published protocols. Recently, the Automated Certificate Management Environment (ACME) protocol has been proposed to automate the certificate issuance process [9]. ACME only solved the automation issue, but the trust concerns remain as ACME requires a trusted CA.

The idea of decentralizing systems has been investigated using the emergent Blockchain technology [10]. Blockchain aims at offering decentralization by leveraging consensus algorithms among participating nodes using standardized cryptographic functions and protocols. It redefines trust by providing a permissionless, censorship-resistant, immutable and transparent data store.

In this paper we propose a method that decentralizes the ACME protocol by leveraging the Blockchain technology, aiming at reducing the trust concerns of existing PKI systems. This paper proposes to build on top of Blockchain a decentralized mechanism for domain ownership verification, a scheme that performs on-chain verification with minimal trust. The verification process resembles "Let's Encrypt" CA and the ACME protocol. The proposed system not only solves the automation of certificates' issuance, but also minimizes the trust concerns. Furthermore, by creating such scheme, the problem of having compelled certificates is implicitly resolved since the CA is not doing the verification manually. As a result, governments and third parties can no longer interfere in the verification process. The main contributions of this paper can

This work was supported by the U.S. National Science Foundation (NSF) (Division Of Graduate Education (DGE) #1822567.

be framed as follows:

1) Resolving the server-side trust concerns through an automated Blockchain-based Domain Control Verification (B-DCV) with a provably-verifiable approach.
2) Eliminating the need for a trusted CA in the domain verification process.
3) Resolving DDoS attacks targeting single points of failure in CAs through the usage of Blockchain.
4) Enabling certificate issuance automation by imitating the ACME protocol.

The rest of this paper is organized as follows. In the next section, we provide a background on Blockchain and ACME. In Section III, we elaborate on the proposed system and detail its architecture, components, and configuration. In Section IV, we present the implementation and analyze the obtained results. Finally, Section V provides concluding remarks discusses the future work.

II. Background

A. Blockchain background

In 2008, Satoshi Nakamoto published a paper entitled "Bitcoin: A peer-to-peer electronic cash system" [11] which introduced an innovative and novel way to transfer (send and receive) digital money (called crypto-currency) without the need of going through a trusted third party or intermediary bank. This paper is considered a breakthrough in the cryptography world, as it solved the digital cash's well-known "double spending" problem [12]. The idea is based on having an immutable digital ledger that records all transactions in a verifiable and persistent way. The ledger is replicated across several nodes, which means that no single authority owns or maintains it. The ledger's version validity is established through consensus among the participating nodes, also called miners. The transactions are stored in blocks linked using cryptography (hence the term Blockchain), specifically using hash functions: each block stores the hash of the previous block, timestamp, and transactions data. Therefore, data on a specific block cannot be altered without changing subsequent blocks, which requires the network's consensus.

B. Smart contracts

The exchange of digital currency was the main purpose of Bitcoin. Afterwards, researchers involved in creating a more general platform for decentralized applications (DApps) [13] development through" Smart Contracts". A smart contract is defined as a computer code running over Blockchain, capable of exchanging any value (money, property, etc.) without the need of a third party. They offer the following advantages over the existing computer programs: 1) Autonomous: their execution is managed by the network, 2) Trust-less: the ledger's version is validated with consensus among nodes, 3) Data safe: the application's data remain permanently in the Blockchain, 4) Transparent: smart contract's code and storage are publicly available

C. Ethereum

Ethereum is an open source decentralized Blockchain-based computing platform that executes smart contracts [14]. It includes the Ethereum Virtual Machine (EVM) as its runtime environment. Its smart contracts are written in high-level programming languages like Solidity, Serpent, Mutan, and are compiled to EVM bytecode for execution. Smart contracts are considered as accounts in Ethereum, and are controlled by their code. Once deployed, the contract becomes identified by an address, and its associated data become publicly visible. The other account type is the Externally Owned Account (EOA): It is a user type account, linked to a keypair generated upon account creation. The holder of the private key corresponding to a wallet controls it and signs transactions initiated from the wallet address. The EOA address is used to reference the account, while the private key is used for signing the transactions. Executing smart contracts functions is considered as a transaction in Ethereum. The cryptocurrency which Ethereum uses is called Ether. It is not only transferred between the users, but also to incentivize miners who run the Blockchain network.

D. Light Client

A light client is developed to enable constrained devices (mobile, IoT, etc.) interact securely with the Blockchain. The devices download only the block headers, and fetch data from the Blockchain on-demand. These clients cannot be used as miners, and do not participate or affect the consensus process. In Ethereum, the Light Ethereum Protocol (LES) [15] is adopted, whose security is based on Merkle proofs. Basically, a light client initiates a request to a light client server, which in turns, finds the data, fetches the Merkle branch (all hashes from the requested data to the tree root), and returns them to the client, which then verifies the authenticity of the requested data in logarithmic complexity.

E. ACME (Automated Certificate Management Environment)

Automatic Certificate Management Environment (ACME) is a proposed standard used by "Let's Encrypt" CA to automate the issuance of domain-validated (DV) certificates [16]. Deploying an HTTPS-enabled website is complicated, expensive, and error-prone for server operators. The HTTPS difficulty resides in generation of the certificate and the interactions with the certificate authorities that should be trusted by the Web browser. The installation of a certificate in a webserver requires the server to use a key generation software and to manually follow steps to configure and validate the control of the domain name. Let's Encrypt has simplified the HTTPS protocol adoption through ACME protocol. It automates the server configuration, identity validation, certificate issuance, and server configuration, which results in providing certificates at low cost. The ACME mechanism can be summarized as follows:

1) The CA generates a random token and sends the token and list of challenges that the client (certificate's requester) can complete to prove ownership of identifier.

2) The client selects the HTTP challenge, creates a file containing a token, and hosts it at a directory on the claimed server.
3) Client informs the CA that challenge is complete.
4) The CA verifies that the file is present and that it contains the correct challenge response.
5) Client sends a Certificate Signing Request (CSR) PKCS#10.
6) CA issues the certificate.

III. Proposed system

The proposed system decentralizes the domain ownership verification used by the ACME protocol. The aim is to use Blockchain, which is a trustless decentralized network, in the domain verification process. Fig. 1 depicts the high level architecture, which has the following components:

Device. The device is the server that needs to register its domain name in the Blockchain. Typically, the device is a web server that has a domain name. The device is composed of a 1) registrar: a software module that is responsible for initiating and completing the registration process; 2) light client (LES): a software component that enables a secure interaction with the Blockchain without having to download all the blocks; 3) Wallet: the server's EOA; and 4) web server: used in the domain verification process.

Smart contract. The smart contract deployed in the Blockchain has a mapping data structure which maps the devices' domain names to their records. Each record contains: a) Domain name: represented in UTF-8 string encoding, b) Public Key: RSA 2048-bits public key, c) Address: Device's wallet address, d) Verified: Boolean variable to identify the verification result. A mapping data structure can be seen as a hash table, and has a complexity of O(1) in storage and retrieval. The smart contract additionally contains functions used for the identities' verification.

Oracle Service and Attestators. The Oracle service is a component situated between the Blockchain network and the public Internet. It helps smart contracts issue requests to the Internet via HTTP(S) POST and GET methods to gather information or post data. The main challenge with oracles is trust. Fortunately, recent substantial research attempts succeeded in solving these trust issues by providing different trusted computing techniques. They provide an "authenticity proof" which is essentially a cryptographic based evidence that the data is not tampered with, and the data integrity property is maintained. This is achieved via oracle services such as Oraclize. Oraclize provides an enhanced oracle network that uses the TLSNotary proof [17], which returns a cryptographic proof for the user showing that a certain HTTPS request returned data from the right server at a specific time Hence, Oraclize refer to their service as "provably-honest". On the other hand, companies like Town Crier are focusing on using Trusted Execution Environments (TEE) such as the Intel Software Guard Extensions (SGX), to guarantee that the returned data is not tampered with. Other hardware-based techniques include Qualcomm TEE, Android SafetyNet, Ledger Nano S attestation, Samsung Knox, etc. Each one of the aforementioned techniques is bound to a major company (the attestator).

A. Blockchain-based Domain Control Verification (B-DCV)

The B-DCV mechanism is depicted in Fig. 2. Initially, the domain owner generates a certificate and install it on the webserver. This certificate can be self-signed, or optionally signed by a trusted CA. The device sends to the smart contract the domain name (identity) and the generated public key in the certificate. The smart contract challenges the device to prove that the domain is controlled by the user. N_1 is generated with the help of an oracle service, as generating random numbers in a deterministic machine (EVM) is not possible. Hence, the smart contracts contacts Oraclize to generate a random number N_1. Oraclize uses the TLSNotary proof method, and stores the secret in an Amazon Web Services virtual machine [18].

The domain owner, generates a random number N_2, and stores $H(N_1||N_2)$ into the webserver's root directory

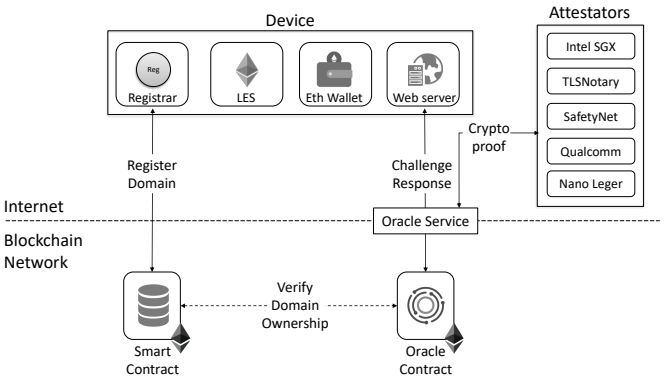

Fig. 1. Proposed System Architecture

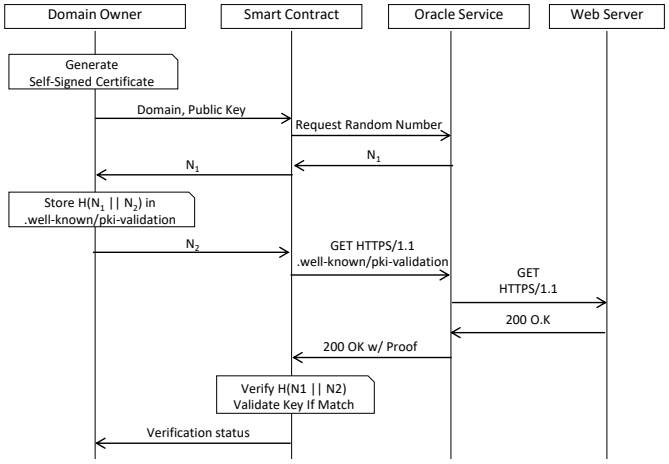

Fig. 2. Blockchain-based Domain Control Verification

(`.well-known/pki-validation`), where $H()$ is a hashing function (Keccak-256 hash function). Afterwards, it sends N_2 to the smart contract which in turns, initiates an HTTPS GET request to the server requesting the file hosted on the root directory. When the result arrives to the smart contract, it verifies the hash, the result's proof, and approves the server's record. Upon approving the server's identity, other devices can retrieve its public key and establish a secure session.

Then main idea behind this system is that the verification process is executed outband; verifying the claimed domain name is being done through a different channel than that of Blockchain. Previous approaches (e.g., [19]) considered using two-factor authentication through SMS to verify ownership of the mobile phone number (MSISDN). In a previous work [20], we proposed a system to distribute Port Control Protocol (PCP) mappings in a decentralized fashion. A verification method closely similar to the one presented in this paper was used to verify the ownership of the mappings.

B. Oracle Security

Introducing oracles in the Blockchain creates a trust problem, often referred to as the "Oracle Problem" [20]. The problem is that there is a need to trust the Oracle for the external data integrity. To overcome this concern, we propose combining the proofs of several trusted attestators, aiming at minimizing trust to a negligible level, as shown in Fig. 3. In other words, the data is considered tampered with if and only if all attestators collaborate at the same time to modify the data. This is a very unlikely scenario as the attestators correspond to separate companies. The downside of this method it that it increases the total transaction cost of the verification process as several attestators are to be contacted, and hence, several transactions should be issued.

C. Client-Server Secure Session Establishment

We demonstrate in this section how the session can be set up between a client and a server using SSL/TLS. The certificate remains in the session establishment in order to enable an easy integration with the existing systems (i.e., web server based on PKI). Once a device is properly configured and registered, it can setup a secure session by using the destination's public key from Blockchain. The certificate remains in the session establishment process in order to simplify the integration with existing PKI systems. The sequence of messages exchanged between the client and the server for the session setup is depicted in Fig. 4. After performing the standard certificate exchange mechanism, the client compares the public key stored in the Blockchain against the certificate's public key.

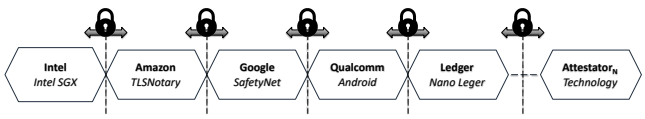

Fig. 3. Attestators chain of trust

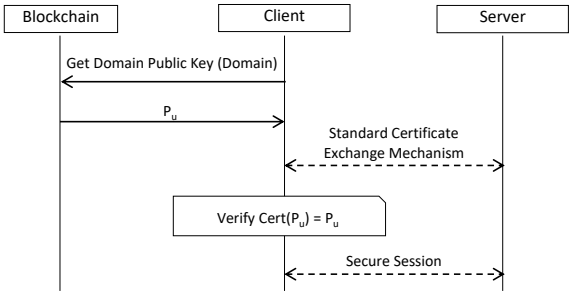

Fig. 4. Secure Session Establishment

D. Revocation mechanism

Conventional X.509 certificates are revoked whenever they can no longer be trustable. This can be due to the certificate's expired date, or leakage of the private key. The proposed system takes into consideration these concerns, and provides a way to update the public key, only if the owner authorizes this update. As explained previously, each record in the smart contract contains the device's wallet address. To update a record fo revocation purposes, a client is restricted to initiate an update request from the wallet used initially for creating its identity. If this assertion fails, the smart contracts denies updating the public key.

IV. Implementation and Results

In this section, we provide the technical implementation details of the system. Specifically, we list the technologies used for development, and analyze the costs required to call smart contracts functions and to execute them in the EVM. Moreover, we analyze the security of the smart contract against various security attacks by using a symbolic execution tool. The technologies used in the development process are: 1) Ropsten Testnet [21]: A public network that simulates Ethereum and EVM; 2) Solidity [22]: Smart contract programming language; 3) Web3j: Lightweight Java application for interfacing the Ethereum Blockchain; and 4) LES: Light client running on the devices. The time used for registering devices in the smart contract is related to the Ethereum block mining time, therefore we do not evaluate the required storage

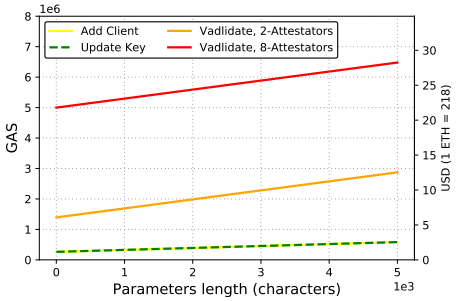

Fig. 5. Smart Contract's function GAS/USD requirements vs. total parameters length.

TABLE I. SMART CONTRACT'S GAS USAGE AND FEES (ETHER AND USD)

Task (and Subtasks)		Gas	Ether Value for Oracle	Ether Total	USD
Create a Record	Register Domain	361762	N/A	0.00723524	1.58
	Generate Random Number (+TLSNotary)	200000	0.000228749	0.004228749	0.92
	Oracle's Callback				
	Validate Record	355847	N/A	0.00711694	1.55
	Send HTTPS (+TLSNotary) in DCV	200000	0.000228749	0.004228749	0.92
	Oracle's Callback				
	Total	1117609	0.000457498	0.022809678	4.98
Update Device's Record	Update Device's Public Key (for Revocation)	246410	N/A	0.004928200	1.07

time. Evaluating a smart contract involves estimating the GAS required for executing its transactions. Table I represents the estimated costs in both Ether and USD for the smart contract's tasks execution. As of March 2020, 1 Ether $\approx$ \$218. Retrieving the public key of a device from the smart contract is free since the function is not modifying or storing data in the Blockchain. As shown in the table, creating a domain record requires 0.02281 ETH, which is equal to \$4.98. Note that this cost considers a single authenticity proof, TLSNotary. Updating a domain record requires $\approx$ 0.004 ETH, which is equal to \$1.07.

Note that the gas usage depends on the complexity of the smart contract code, and the length of the parameters passed to the functions. The presented values represent the average costs required for typical domain names lengths (less than 30 characters). Additional transaction costs are required as the length of the domain name gets larger, as shown in Fig. 5.

V. CONCLUSION

In this paper, we have proposed a Blockchain-based method that decentralizes the ACME protocol by combining elements of the PKI/CA model with Blockchain technology. It aims at resolving the trust concerns of the existing PKI/CA infrastructure. The method eliminates the need for a trusted CA in the domain verification process, and resolves DDoS attacks targeting single points of failures. It also automates the certificate issuance process by imitating the ACME protocol. Our implementation was tested on Testnet, an Ethereum Blockchain public simulator. Results showed that the solution is efficient in terms of transaction costs. For future work, we intend to develop the session establishment software module as a plug-in to be integrated in major browsers. Additionally, we aim at solving the client authentication problem.

REFERENCES

[1] M. Galluscio, N. Neshenko, E. Bou-Harb, Y. Huang, N. Ghani, J. Crichigno, and G. Kaddoum, "A First Empirical Look on Internet-scale Exploitations of IoT Devices," in *IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications*, IEEE, 2017.

[2] G. Srivastava, J. Crichigno, and S. Dhar, "A Light and Secure Healthcare Blockchain for IoT Medical Devices," in *IEEE Canadian Conference of Electrical and Computer Engineering*, IEEE, 2019.

[3] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things Security: A Survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017.

[4] D. Fisher, "Final Report on DigiNotar Hack Shows Total Compromise of CA Servers." https://threatpost.com/final-report-diginotar-hack-shows-total-compromise-ca-servers-103112/77170/.

[5] N. Leavitt, "Internet Security Under Attack: The Undermining of Digital Certificates," *Computer*, vol. 44, no. 12, pp. 17–20, 2011.

[6] S. Gangan, "A Review of Man-in-the-Middle Attacks," *arXiv preprint arXiv:1504.02115*, 2015.

[7] E. Kfoury and D. Khoury, "Distributed Public Key Infrastructure and PSK Exchange Based on Blockchain Technology," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1116–1120, IEEE, 2018.

[8] E. F. Kfoury and D. J. Khoury, "Secure End-to-End VoIP System based on Ethereum Blockchain," *Journal of Communications*, vol. 13, no. 8, pp. 450–455, 2018.

[9] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten, "Automatic Certificate Management Environment (ACME)," RFC 8555, RFC Editor, March 2019.

[10] M. Pilkington, "Blockchain Technology: Principles and Applications," in *Research handbook on digital transformations*, Edward Elgar Publishing, 2016.

[11] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," tech. rep., Manubot, 2019.

[12] U. W. Chohan, "The Double Spending Problem and Cryptocurrencies," *SSRN 3090174*, 2017.

[13] S. Raval, *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*. O'Reilly Media, Inc., 2016.

[14] G. Wood *et al.*, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[15] V. Buterin, "Ethereum Light Client Protocol," *Accessed: Jul*, vol. 9, p. 2019, 2016.

[16] J. Aas, R. Barnes, B. Case, Z. Durumeric, P. Eckersley, A. Flores-López, J. A. Halderman, J. Hoffman-Andrews, J. Kasten, E. Rescorla, *et al.*, "Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2473–2487, 2019.

[17] TLSnotary.org, "TLSnotary - a Mechanism for Independently Audited HTTPS Sessions." https://tlsnotary.org/TLSNotary.pdf.

[18] J. Heiss, J. Eberhardt, and S. Tai, "From Oracles to Trustworthy Data On-Chaining Systems," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 496–503, IEEE, 2019.

[19] D. Khoury, E. F. Kfoury, A. Kassem, and H. Harb, "Decentralized Voting Platform based on Ethereum Blockchain," in *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, pp. 1–6, IEEE, 2018.

[20] E. F. Kfoury, J. Gomez, J. Crichigno, E. Bou-Harb, and D. Khoury, "Decentralized Distribution of PCP Mappings Over Blockchain for End-to-End Secure Direct Communications," *IEEE Access*, vol. 7, pp. 110159–110173, 2019.

[21] D. Mohanty, "Deploying Smart Contracts," in *Ethereum for Architects and Developers*, pp. 105–138, Springer, 2018.

[22] C. Dannen, *Introducing Ethereum and Solidity*, vol. 1. Springer, 2017.