International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 14, Number 2, pages 225–247, July 2024

A Blockchain-based Approach with zk-SNARKs for Secure Email Applications

Md. Biplob Hossain[⋆§], Maya Rahayu[⋆#], Md. Arshad Ali[†], Samsul Huda[‡], Yuta Kodera[⋆], Yasuyuki Nogami[⋆]

[⋆]Graduate School of Environmental, Life, Natural Science and Technology, Okayama University, Japan
[†]Faculty of CSE, Hajee Mohammad Danesh Science and Technology University, Bangladesh
[‡]Green Innovation Center, Okayama University, Japan
[§]Electrical and Electronic Engineering Department, Khwaja Yunus Ali University, Bangladesh
[#]Electrical Engineering Department, Politeknik Negeri Bandung, Indonesia
Email: {p8rg1uci, mayarahayu}@s.okayama-u.ac.jp, arshad@hstu.ac.bd, {shuda, yuta_kodera, yasuyuki.nogami}@okayama-u.ac.jp

Received: February 15, 2024
Revised: May 5, 2024
Accepted: June 4, 2024
Communicated by Toru Nakanishi

Abstract

Email serves as the primary mode of communication in today's interconnected digital world, encompassing business, education, and interpersonal relationships. However, email's reliance on shared media makes it susceptible to interception and misuse of confidential data. Pretty Good Privacy (PGP) protects the privacy of email contents to address this problem. While PGP offers encryption, its key sharing has weaknesses. Blockchain technology is characterized by its immutability feature. Once information is stored in the blockchain, altering it becomes extremely difficult. This characteristic serves as a valuable defense against weaknesses in the PGP key sharing system. Furthermore, the implementation of smart contracts eliminates the need for a Man-in-the-Middle when sharing keys, thereby improving the security of key sharing and fostering trust among individuals. Blockchain and smart contracts improve security, but privacy remains a concern. To further bolster privacy protection, in this paper we propose the integration of Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARKs) and blockchain into PGP key sharing mechanism. zk-SNARKs enable efficient verification of encrypted data without revealing sensitive information, thus preventing exposure of user privacy. Additionally, we employ Elliptic Curve Cryptography (ECC) in order to guarantee the confidentiality of the PGP key. Through this holistic integration, the security of the PGP key is enhanced, ensuring both confidentiality and integrity while safeguarding user privacy. Furthermore, gas consumption and transaction costs were evaluated with and without zk-SNARKs. The results demonstrate that the proposed mechanism minimizes gas consumption and transaction costs.

Keywords: Email security, Blockchain, Pretty Good Privacy (PGP), zk-SNARKs

1 Introduction

One of the most significant forms of communication that individuals utilize on a regular basis to communicate information globally is email [1]. Because of its convenience characteristics, there are an

increasing amount of emails sent and received worldwide every day. Traditional email systems retain the content in plain text on the server, however they only authenticate users based on their user name and password on the email server [2]. For this, although using emails in everyday conversations is quite easy, there is an important threat to people's privacy when using them. This is largely because emails often include sensitive information and regular large-scale cyber attacks on email networks have also become more common. As a consequence, both individuals and companies are increasingly using encrypted email services and are more aware of email security.

Because of this, before transmitting an email from the sender to the recipient, it must be encrypted and signed. However, email encryption using traditional IBE may not be enough. Preserving confidentiality over time, ensuring that emails encrypted in the past stay protected even if the long-term secret key is compromised, is essential for enhancing the security of encrypted email platforms. Secure/Multipurpose Internet Mail Extensions (S/MIME) and PGP [3] are two of the most widely used secure email encryption techniques available. Using both S/MIME and PGP techniques, early work concentrated on the integrity, secrecy, and authenticity of email communications [4]. Email contents are secured by encryption, but the safe transmission of the key needed to decrypt the communication remains a significant problem in this scenario.

PGP is an asymmetric key encryption system that supports data secrecy and integrity via the use of cryptography. PGP can be used for both encrypting and signing the messages. Initially, the sender generates a private key, sometimes referred to as a random key, in the public key authentication system to PGP. Subsequently, the sender uses this random key to encrypt the communication. To secure the random key during transmission, the encryption is further ensured by utilizing the recipient's public key. At that case, the recipient acquires both the message and the encrypted random key. Using his/her private key, the receiver must first decode the random key on the other end. Then the receiver may read and decode the original message using this random key. PGP provides the secrecy and integrity by signing and encrypting communications sent over the network [4]. When PGP is used, it simultaneously transmits both the encrypted message and the encrypted key, creates a risk of losing the key, if the attackers manage to get them. Therefore, sharing the encrypted key from the sender to the receiver with PGP still presents a difficulty.

In this situation, using a blockchain can lessen the weaknesses in PGP key sharing protocol. Blockchain allows the encryption key to be shared independently from sender to receiver due to its distributed and immutable properties. The absence of a central authority diminishes the possibility of a single point of failure in the encrypted key transfer process. Every time the sender saves the encrypted key on the blockchain, all participants must verify the transaction. Whenever the intended user attempts to access the encrypted key, the same verification is required. Due to the specified terms of smart contracts, attackers are unable to obtain the encryption key directly since storing and accessing it from the blockchain necessitates several verification procedures. Additionally, it eliminates any encryption key deletions or modifications made during transactions. Conversely, all users within a network may see transactions recorded on a blockchain, fostering a sense of mutual trust and holding each other equally accountable for transaction fulfillment [5].

Blockchain is a distributed ledger in which each node is free to carry out their own responsibilities as long as they cooperate to confirm every transaction and keep the ledger consistent [5]. Blockchain becomes more adaptable when a smart contract is used to write computer code that specifies the procedures and system administration [6]. It guarantees that every node abides by the same rules and automates the transactions. Due to the cryptographic hashing technique, the blocks are impenetrable and immune to manipulation. Blockchain is taken into consideration in secure email applications for this hashing cryptographic properties [7]. Piedrahita et al. [8] proposed a blockchain-based secure email solution, where the authors noted the security issues with email correspondence and suggested potential fixes based on an adaptable and unchangeable blockchain architecture. Nevertheless, the authors reviewed a number of research studies to identify the issues and provide a course of action for resolving them rather than implementing any blockchain design to address the issues. Vasantha et al. [9] suggested to use blockchain technology together with the blowfish algorithm to secure and protect email correspondence. The blowfish method is employed in this case to convert the information of the node into a hashed text structure, which is then saved in separate segments.

Blockchain technology has the ability to provide anonymity and security for a number of uses,

such as file sharing, data sharing, and email delivery. It incorporates smart contracts, a distributed file system [10], and a hybrid encryption method in addition to a consensus algorithm [11]. Francisca et al. [12] introduce a blockchain-based certified electronic mail solution. The method maintains the privacy of the correspondence between the sender and the recipient, even when the contents are in plain text. In order to offer a decentralized and secure method for verifying mail, Varghese et al. [13] suggest a system based on blockchain technology that monitors the dispatch of outgoing mail. The authors did not provide any particular answers, while having described in technical terms how the suggested system operates.

One of the works suggested using smart contracts for email protocols in [14]. By creating a decentralized system, authors suggested that the architecture may reduce typical issues with email services including spoofing, phishing, and spam. Even yet, there can be obstacles to the solution's acceptance and use of secure email communication.

This study introduces a proposal for a framework based on blockchain to effectively convey the encrypted key of the PGP system from the sender to the receiver during the transmission of the encrypted message in a manner that mirrors PGP. In our previous work [15], we already successfully transmitted PGP's encrypted key using the blockchain technology. Immutability plays a crucial role in blockchain technology by enhancing its security and resilience against data tampering. It ensures that once data is recorded on the blockchain, it cannot be altered or deleted without detection. In our study, we stored the encrypted key in the block of blockchain for secure transmission. Each block within the blockchain is comprised of a unique hash that represents its content, which also includes the hash of the preceding block. This creates a chain of blocks, with each block cryptographically linked to the one before it. Any attempt to modify the encrypted key in the block would change its hash value, making it immediately evident that the block has been tampered. Furthermore, consensus mechanisms are essential for blockchain networks as they play a crucial role in reaching a mutual understanding among nodes regarding the authenticity of transactions and the sequence in which they are incorporated into the blockchain. It is imperative for the majority of network participants to reach a consensus on the blockchain's status. Even attackers are able to compromise a single node, they would need to compromise a majority of the nodes in the decentralized network simultaneously to alter any data in blockchain, which is highly impractical. Additionally, the immutable nature of the blockchain creates a tamper-evident audit trail of all transactions. Any attempt to modify historical data would be immediately apparent, making it easy to detect and trace any attempted tampering by attackers.

However, nodes in a blockchain network must come to an agreement and for this the user's need to disclose their personal information to the network for executing the transaction verification. This poses significant privacy issues for the users. That is, the privacy of user data cannot be guaranteed by the blockchain alone. Some other techniques can be added to increase the privacy of the transaction. Several privacy-preserving methods, like homomorphic encryption and zero-knowledge proof (ZKP), have been used to blockchain applications to address the privacy issues in the blockchain [16].

Zero-knowledge proofs (ZKP) are a set of procedures created to enable interactive confirmation among two or multiple parties. Without giving any valuable details, this technique allows verifiers to be certain that a prover holds certain confidential data. This characteristic makes it possible for ZKP technology to ensure data validity without disclosing any confidential details about the data. The generation of both the proving key and verification key is essential for the successful implementation of the zero-knowledge proof protocol. The proof generated should allow for multiple verifications to be conducted using the verification key, while ensuring that no additional sensitive information is disclosed to any potential adversaries during the entire verification process. ZKP is often used to implement blockchain technology, anonymous credibility, remote verification, and asset settlement [17].

ZK-SNARKs (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) are a kind of zero-knowledge proof (ZKP) cryptography technique that is essential to protecting the confidentiality of data. Without sharing the real data, the two individuals may use zk-SNARKs to confirm their mutual accuracy in an assertion or piece of information. Because of this, zk-SNARKs are a very effective technique for protecting privacy of information [18]. Without disclosing the specifics of the

transaction, a user may demonstrate that they have sufficient balance to complete the transaction. This facilitates the accomplishment of blockchain's primary objective, which is safeguarding and disclosing information while preserving user privacy [19].

In the present study, our aim is to securely transfer the encrypted key of the PGP system from the sender to the receiver using blockchain technology without disclosing the details of the transactions.

The contributions of this paper are listed below:

- To distribute the encrypted key, we use blockchain. As blockchain is dispersed, it makes it difficult for attackers to alter blocks and eliminates the Man-in-the-Middle danger, which maintains key security.
- The blockchain contains records of every transaction that takes place between those who use the network. Network nodes use a decision-making technique that employs the majority of participants to verify transactions before introducing them to the blockchain. As a result, user's faith in the framework is increased.
- The encrypted key is kept in a chain of interconnected blocks. The option of changing an encrypted key is eliminated since it is impossible to remove or alter the chain without the entire network's consent.
- We have incorporated zk-SNARKs into the blockchain technology in order to distribute the encrypted key of the PGP system. This integration guarantees the anonymity of those involved in transactions and enhances the confidentiality of data.
- We created the necessary proofs of our system and stored them in the blockchain framework, which can significantly reduce storage requirements.

2 Literature Review

The incorporation of blockchain technology within the PGP system functions as a means to address issues like Man-in-the-Middle attacks and ineffective certificate revocation due to synchronization delays. In response to security concerns related to PGP key servers, a novel PGP management framework utilizing blockchain technology has been implemented to address these issues [3]. Users may update their own PGP certificates with ease due to the framework's quick propagation of certificate revocation. Sending both the original message and secret key simultaneously still bears resemblance to the conventional use of traditional PGP. Due to the fact that the message and the key are transmitted simultaneously, there is a chance that the attackers would attack, and we may lose both. Chuat et al. [20] introduce efficient gossip protocols for detecting inconsistencies in certificate logs, addressing trust issues in certification authorities. These procedures facilitate the identification of discrepancies in logs by utilizing simulations that are derived from actual Internet data traces, offering a strategy for implementation and specific execution instructions. Nevertheless, the limited availability of genuine data traces from real-life networks may influence the dependability of the simulation outcomes. Anada et al. [21] introduce a technique for embedding identities in decentralized public-key infrastructure, which allows for the establishment of a network of trusted entities based on public keys. This approach combines two public-key cryptosystems to embed the IDs of the public-key holder and their guarantors in the public keys. This process facilitates the creation of a decentralized Public Key Infrastructure (PKI) by utilizing both RSA encryption and elliptic curve encryption. However, the identity-embedding method proposed in the paper does not directly resolve the issue of identity retention.

A prevalent method for securely creating and asserting key pairs involves the utilization of a Public Key Infrastructure (PKI). PKI plays a crucial role in addressing authentication issues within networks and offers assurances regarding the validity of a certificate endorsed by a certification authority (CA). The application of Blockchain technology presents a potential remedy for ensuring data integrity, leveraging cryptographic techniques to establish resistance against tampering. In instances where secure communication and data integrity are paramount, Blockchain can be effectively employed. Li et al. [22] discuss the limitations of traditional centralized PKI systems and

proposes blockchain-based solutions to address security concerns. The suggested framework makes use of blockchain as a public notice board or trusted majority, showcasing its feasibility and practicality by assessing off-chain time costs and on-chain gas costs. However, decentralized PKI systems face challenges in certificate issue and storage due to the unrealistic storage of certificates on a blockchain. A Secure Blockchain Trust Management (SBTM) employs a blockchain-driven Public Key Infrastructure (PKI) to improve the security of routing protocols within Autonomous Systems (ASes) and underscores the advantages of blockchain frameworks in ensuring the security of routing protocols [23]. But the size of the blockchain in the system may grow over time, potentially posing challenges for lookup operations. Ahmed et al. [24] propose a privacy-aware blockchain-based PKI solution addressing conventional PKI flaws. The proposed privacy-aware blockchain PKI may involve a security cost due to the potential for short-term tampering with online public keys by network members.

PB-PKI is a blockchain-based Public Key Infrastructure (PKI) that prioritizes privacy considerations that does not link identity with public keys, addressing conventional PKI flaws, discussed by Axon et al. [25]. PB-PKI provides unlikable short-term key updates, user-controlled disclosure, and offline key sharing for tracing misbehavior. The security of PB-PKI relies on the honesty of most miners in the underlying blockchain, which could be undermined by a collusive majority of dishonest network members. Bassam et al. [26] highlight SCPKI, a decentralized PKI system using a web-of-trust model and Ethereum smart contracts to enhance transparency and detect rogue certificates. The system allows fine-grained attribute management for identity verification, addressing the limitations of traditional PKI systems. The system currently lacks discussion on potential future work and limitations. Yakubov et al. [27] outline a framework for managing PKI using blockchain technology, which facilitates the issuance, validation, and revocation of X.509 certificates. This system aims to improve security and dependability. Utilized Blockchain technology to eliminate single points of failure in traditional PKI systems and provide a fully traceable history log. However, the volatility of cryptocurrencies directly impacts the costs of blockchain operations, creating uncertainty in long-term and short-term certificate costs. Wilson et al. [28] suggested a PGP certificate based on Bitcoin that makes use of certain features to solve some of the drawbacks of the current PGP certificate paradigm. With each created PGP certificate, the suggested framework may conduct identity verification transactions using the Bitcoin address [29]. However, since Bitcoin infrastructure must be widely used, there may be obstacles to the suggested certificate's acceptance and deployment. Additionally, it may make the process of issuing certificates and verifying identities more difficult [30].

In the realm of blockchain and cryptocurrency applications, zk-SNARKs empower individuals to validate transactions while upholding privacy. The principal aim in safeguarding transaction privacy is to heighten the secrecy of different transaction elements such as sender and recipient addresses, transaction sums, and the connection between the transacting parties. By guaranteeing the obscurity of both sender and recipient addresses, the identities of the involved parties can be effectively obscured. Numerous innovative methodologies have been suggested to accomplish transaction privacy protection utilizing zk-SNARKs.

Miers et al. [18] presented Zerocoin as an cryptographic enhancement for Bitcoin, enabling fully anonymous currency transactions by enhancing the protocol without the need for new trusted entities or altering Bitcoin's security model. This innovation eliminates the necessity for coin issuers, allowing individual Bitcoin users to create their own coins. Nevertheless, the sender and receiver addresses, along with transaction amounts, were still visible through this method. To tackle this issue, Ben-Sasson et al. [19] proposed Zerocash as a digital currency offering robust privacy protections by concealing the origin, destination, and amount of payments, utilizing zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs). Zerocash represents a feasible implementation of a decentralized anonymous payment (DAP) system that is more efficient and confidential compared to Zerocoin, and competitive with conventional Bitcoin.

Guan et al. [31] have proposed BlockMaze, a blockchain system designed to enhance transaction privacy protection. This system is based on zk-SNARKs and aims to tackle the privacy challenges within blockchain technology. Through BlockMaze, strong privacy assurances are achieved by concealing account balances, transaction amounts, and the connections between senders and receivers.

A dual-balance framework is proposed for account-based blockchains, comprising of a transparent balance and a concealed zero-knowledge balance for every account. This novel methodology incorporates zero-knowledge transactions among participants to conceal transaction connections, alongside a reliable commitment system for transaction values and account balances. Moreover, transaction privacy was preserved by the idea of a decentralised mixing system, which was introduced by Song et al. [32]. This method concentrated on hiding the sender and recipient addresses in addition to the connection between transactions.

Xu et al. [33] have introduced a method that prioritizes privacy while utilizing zk-SNARKs to address privacy concerns. An examination of the security aspects and efficiency of this approach was conducted to showcase its efficacy in various scenarios. This method does not necessitate the involvement of any specific entity within the blockchain system and does not alter the fundamental structure of current applications. Meanwhile, Hou et al. [34] delved into the issue of privacy preservation in energy trading systems. To ensure anonymity during energy transactions, this study merges blockchain technology with a double auction mechanism, employing zk-SNARKs to validate bids without revealing personal details.

Zhang et al. [35] introduce a novel identification scheme utilizing a zero-knowledge proof (ZNP) system to enhance privacy protection. This approach enables one party in communication to persuade the other party of its legitimacy, thereby bolstering authentication within the proposed framework. While the aforementioned study bolsters security and trust in the realm of music education through the utilization of blockchain technology and smart contracts, it lacks empirical validation or case studies to substantiate the efficacy and feasibility of the proposed framework. In a similar vein, Yuan et al. [36] present a privacy-preserving mechanism for a credit-investigation system employing blockchain and ZKP technology, facilitating secure sharing of credit-investigation data across various entities. This system utilizes zero-knowledge-proof technology to guarantee fairness and confidentiality in the identity authentication process when conducting credit-investigation data inquiries. It would be beneficial to address any potential security risks or limitations associated with the use of blockchain in credit-investigation systems. Within the framework of user identity privacy preservation, scholars have concentrated on preserving the characteristics linked to specific users while guaranteeing the security of their connections. The main aim is to improve user privacy of certain identifying traits or their outward appearance within a range of acceptable values.

Ren et al. [37] present a proposal for a privacy-preserving system designed to safeguard user identity in a lightweight manner. This system enables users to access data and services while keeping their true identity concealed. The procedure includes a zero-knowledge proof protocol that is interactive, used for verifying the identity of the user by the cloud service provider (CSP). This protocol involves a trusted third-party that oversees user information and sets up a secure communication channel between the CSP and the user, thereby ensuring privacy preservation. In order to address the challenges related to sharing medical data in the healthcare sector, Al-Aswad et al. [38] introduced a blockchain-based solution. This model is designed to enable secure sharing of data across different services and devices while ensuring the confidentiality and privacy of the data are not compromised. Through the utilization of blockchain technology, the system strengthens the security of data exchange and protects the privacy of patients in the healthcare sector. As a result, the implementation of zero-knowledge proof ensures the confidentiality of critical medical data.

Tomaz et al. [39] put forward a method for ensuring privacy in the authentication process of mobile health systems. Addressing the privacy risks in mobile health systems, this scheme is lightweight and suitable for resource-limited devices. This paper also presents a solution for storing, managing, sharing data using blockchain and protect the health data transmission with Attribute-Based Encryption (ABE). Addressed privacy issues with sharing of customer energy data presented by Pop et al. [40]. The article discusses a decentralized approach to running demand response initiatives on the public blockchain. This method guarantees the confidentiality of consumer energy information by employing zero-knowledge proofs (ZKPs) and confirming consumer actions through smart contracts. With this method, secrecy was preserved throughout the recording, storing, and sharing of energy data. Li et al. [41] examine the issue of privacy in various blockchain-based traffic control systems. Their study introduces the idea of a gateway positioned between two neighboring blockchain-driven traffic management platforms, facilitating the transfer of vehicles from one blockchain to another.

The gateway design incorporates a non-interactive zero-knowledge range proof (ZKRP) approach to authenticate vehicle data while safeguarding confidential information.

Blockchain was used by the authors to improve the confidentiality and integrity of data in real estate contract systems introduced Jeong et al. [42]. The utilization of blockchain technology facilitates the management of online contracts and the detection of contract forgery. The confidentiality and prevention of fraud are ensured through the implementation of the zero-knowledge proof algorithm until the contract is finalized and terminated. In their work, Gai et al. [43] introduce a system for identity verification based on blockchain, which safeguards user identities and attributes by utilizing zk-SNARKs. They elaborate on the deployment of the chosen architecture of Zero-Knowledge Proofs (ZKP) with the help of the development toolkit Zokrates, enabling the off-chain generation of ZKP and on-chain verification utilizing Ethereum smart contracts. The authors made sure that identity indications and identity traits were protected by using zk-SNARKs, allowing for secure authentication without compromising privacy.

3 Background and Preliminaries

This section thoroughly explores the fundamental principles underlying the secure transmission of cryptographic keys for email communication within decentralized blockchain networks, incorporating advanced techniques such as zk-SNARKs, which play a crucial role in enhancing privacy and security measures. The discussion also covers the integration of smart contracts, various mechanisms for achieving agreement among network participants (consensus mechanisms), and the programming language used to develop decentralized applications on the Ethereum blockchain (solidity).

3.1 Blockchain Technology

In the realm of blockchain technology, every individual block is linked to another through the utilization of the preceding block's hash value. Due to its distributed data management and encryption by design, blockchain differs from all other databases currently in use.

Cryptography, through deliberate design, serves to safeguard the confidentiality of user information and ensure the accuracy of data [4]. Nevertheless, it is essential to construct the cryptographic algorithm outside the blockchain platform, in what is known as an off-chain setting, if we prefer to add further cryptographic capabilities to the blockchain network. Next, this platform communicates with the blockchain system and on-chain environment. Large key sizes and substantial processing overhead are features of cryptography algorithms like ECC. Consequently, applications for cryptography algorithms exist outside of the blockchain. The capacity of the blockchain to provide a decentralised architecture for private transactions is known as distributed data management. Furthermore, smart contracts increase the flexibility and automation of transactions [5].

These days, decentralised blockchain frameworks like Ethereum, Hyperledger Fabric, and others are available. Ethereum is the blockchain-based development platform that we have selected for our suggested solution in this research as it is well-suited for decentralized applications (dApps) and scenarios where transparency and openness are crucial. Moreover, the integration of zero knowledge proof concept with blockchain makes the privacy protection further bolster. This characteristic makes it possible to ensure data validity without disclosing any confidential details about the data. For example, in a healthcare system, a patient's confidential medical records need to be securely shared with a specialist for consultation without revealing the identity of the patient. Through the integration of blockchain with zk-SNARKs, a hospital can send the encrypted medical record of patient first and later can share the encrypted key with the specialist without disclosing the patient's identity. This ensures that sensitive medical information remains private and confidential, while still enabling collaboration and expert consultation among healthcare professionals.

3.2 Smart Contracts

These written agreements, which all parties are obligated to adhere to, are referred to as smart contracts [6]. The three phases of smart contracts are outlined below.

1) Contract Generation: It covers the parties needs, the contract's details, and clarifies between the contract's participants, among other things. A contract code will be created after the precise definition and fixing of the contract terms.

2) Contract Release: The signed contract is sent to the nodes in peer-to-peer modes after it has been formed, and it is kept there until a consensus is established.

3) Contract Execution: A contract cannot be altered after it is published and the nodes have come to an agreement. The contract automatically comes into action if the predetermined requirements are met [7].

3.3 Consensus Mechanisms

Blockchain offers decentralized privacy and security characteristics via the use of consensus techniques in a peer-to-peer network. The security, scalability, and accessibility of the blockchain are all significantly impacted by the consensus method [8]. It is regarded as an essential component of a distributed ledger that is decentralised. It establishes how the blockchain network's blocks, transactions, and contract executions are validated. It is a procedure for managing abusive conduct and achieving a consensus among users [10]. In order to guarantee that only authorised transactions are recorded in the blockchain, it is helpful to verify and review the data before it is put to the block. It contributes to the stability and integrity of a blockchain network by using consensus to make sure that all modifications are approved uniformly by all nodes [11].

3.4 Solidity

High-level programming languages like Solidity were developed especially for Ethereum blockchain smart contract development. Operates within the Ethereum Virtual Machine (EVM), this is a language that follows the principles of object-oriented programming [14]. The solidity code is created and then compiled to make sure it is accurate, effective, and prepared for deployment. Upon compilation, Solidity code undergoes a transformation into Application Binary Interface (ABI) and bytecode artefacts [24]. ABI provides a listing of the functions present in the smart contract, including information on the specific data types that are used as inputs and outputs by these functions. This ABI document essentially functions as a guide for the information exchanged with contracts, covering aspects like encoding and decoding. Meanwhile, the actual operational logic of the contract is encoded in bytecode, which is designed to be comprehensible and executable by the Ethereum Virtual Machine (EVM).

3.5 zk-SNARKs

zk-SNARKs are a class of cryptographic proof systems [28]. By not revealing specific information about the statement, individuals enable a prover to convince a verifier of the truth of a claim. The concept of "zero-knowledge" describes the situation when the evidence reveals nothing that might compromise the veracity of the claim. The constituents of zk-SNARKs are decomposed as follows:

- Zero-Knowledge: With the use of a proof, one may show that a statement is true without disclosing any information about the assertion itself.
- Succinct: The proof is brief and simple to check. Scalability depends on this, particularly for blockchain applications [29].
- Non-Interactive: There is no need for back-and-forth correspondence between the prover and verifier since the proof is produced in a single step.
- Arguments of Knowledge: Without disclosing the information itself, the proof shows that the prover is in possession of a certain piece of knowledge [35].

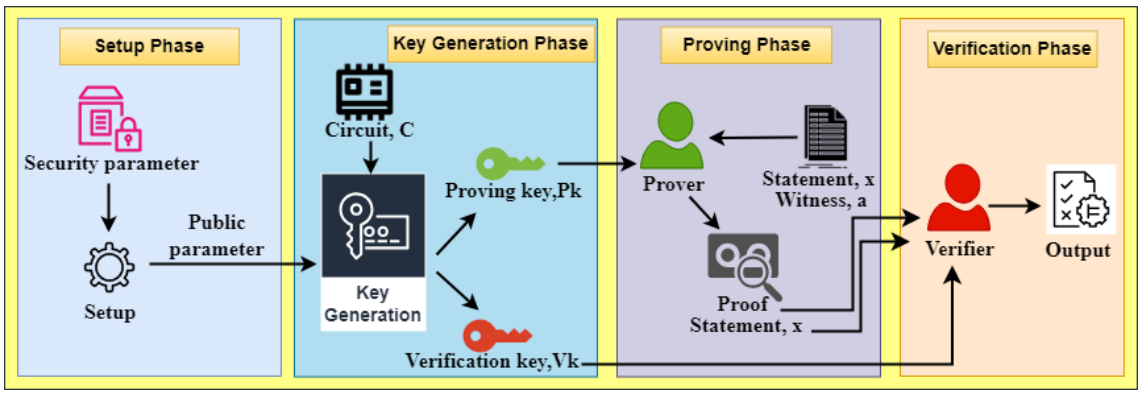

Figure 1: The framework of zk-SNARKs.

Zk-SNARKs are useful in many areas, but they are most prominently used in blockchain technology to facilitate private transactions. A user may demonstrate the legitimacy of a transaction using zk-SNARKs without disclosing the sender, recipient, or transaction amount.

Steps to implement zk-SNARKs on blockchain technology is illustrated in Figure 1 and described as follows [33]:

1. Setup Phase: It needs a one-time trustworthy setup. In this stage, a security parameter is used to build a collection of public parameters. These parameters are obtained from secret parameters during a trusted setup procedure, and together they form the Common Reference String (CRS).

2. Key Generation Phase: This stage generates a pair of keys known as the proving key and the verification key.

3. Proving Phase: Without disclosing any particulars about the statement, the prover creates the proof of a statement using the proving key from the setup step, a public statement, and a private witness.

4. Verification Phase: Anyone may easily verify the correctness of the proof by using the verification key from the setup phase, a public statement, and the proof produced from the proving step. The transaction will be successful if the verification is successful; if not, the transaction request will be rejected.

3.6 Adopted Open Tools

The authors have adopted several open tools. A brief introduction about these open tools are as follows:

- ZoKrates: Zero-knowledge proofs (ZKPs) refer to a set of probabilistic protocols. One particular category of ZKPs is identified as zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs), which have gained significant prominence recently. zk-SNARKs are widely utilized within various applications, such as the anonymous cryptocurrency Zcash and the smart-contract platform Ethereum, which were early adopters of this technology [38]. However, Ethereum's method of running computations on all network nodes leads to increased expenses, limitations in complexity, and reduced privacy. zk-SNARKs, while powerful, present challenges in terms of comprehension and manipulation, as they are primarily utilized for validating on-chain computations and calculating associated costs. To address these complexities, ZoKrates serves as a solution by facilitating the creation of off-chain programs that can be linked to the Ethereum blockchain, thereby enhancing the capabilities of decentralized applications (DApps). In our solidity

code, we implemented the zk-SNARKs idea using the ZoKrates tool. It is necessary to create a separate file first called ZoKrates (.zok) in order to integrate zk-SNARKs into the solidity code using ZoKrates. The zero-knowledge proof's reasoning is explained in this file. After that ZoKrates generates proving key to create the proof. Subsequently, the proof is confirmed using the verification key generated by the ZoKrates tool.

- Truffle: Truffle is a top-tier development environment, testing framework, and asset pipeline designed for blockchains that utilize the Ethereum Virtual Machine (EVM). Its primary objective is to simplify the tasks of users [31]. Truffle offers integrated features such as smart contract compilation, linking, deployment, and binary management. Additionally, it facilitates advanced debugging through functionalities like breakpoints, variable analysis, and step functionality. Truffle is particularly useful for automating contract testing to expedite the development process. In our proposed system, we employed Truffle as the primary development framework for Ethereum. The encrypted key generated in the off-chain environment is initially accessed through the truffle platform. We uploaded and accessed the encrypted key from truffle and connected in to the ganache for visualizing the transaction.
- Remix: Remix Ethereum IDE, also known as Remix IDE, serves as an internet-based platform for developing applications on the Ethereum network. Designed to accommodate users of varying levels of expertise, Remix IDE streamlines the process of creating smart contracts without the need for any initial configuration [40]. Offering a rapid development cycle and a diverse range of plugins featuring user-friendly graphical interfaces, Remix IDE stands as a robust open-source utility enabling the direct composition of solidity contracts within a web browser environment. The tool includes components for both testing and debugging smart contracts, as well as facilitating their deployment, among other functionalities. It can be deployed on various servers, including Node.js, and even in non-Node.js settings at the network periphery. We used remix for executing the solidity code of zk-SNARKs.

4 Proposed System

This section introduces proposed system of encrypted key sharing mechanism using blockchain with the integration of zk-SNARKs. In the previous paper as mentioned in [15], we only considered the blockchain technology for sharing the encrypted key. Now, we also added zk-SNARKs with the blockchain technology to share the encrypted key. For this, the proof of the zk-SNARKs have to be generated first as discussed in section 4.1. After that, the encrypted key have to be shared according to the system described in section 4.2.

4.1 zk-SNARKs Integration for Transaction Verification

Security, efficiency, and privacy are the main goals of Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARKs) for the purpose of sharing identities in blockchain-based systems. Ensuring privacy is crucial since it allows users to authenticate themselves on the blockchain without disclosing private information. For real-time interactions, the system has to be computationally efficient to guarantee quick verification and proof creation. Moreover, conciseness, smaller proofs, and less storage to improve the system's scalability. Figure 2 shows a summary of how zk-SNARKs are integrated with blockchain for transaction verification. It contains the following steps which are delineated as follows:

Step 1. Setup algorithm

Initially, a security parameter λ is used to construct a set of public parameters pp. These values, which come from secret values during a trusted setup procedure, make up the common reference string (CRS). Information about the user's identification are indicated by the notation $I = attribute1, attribute2...attributeN$.

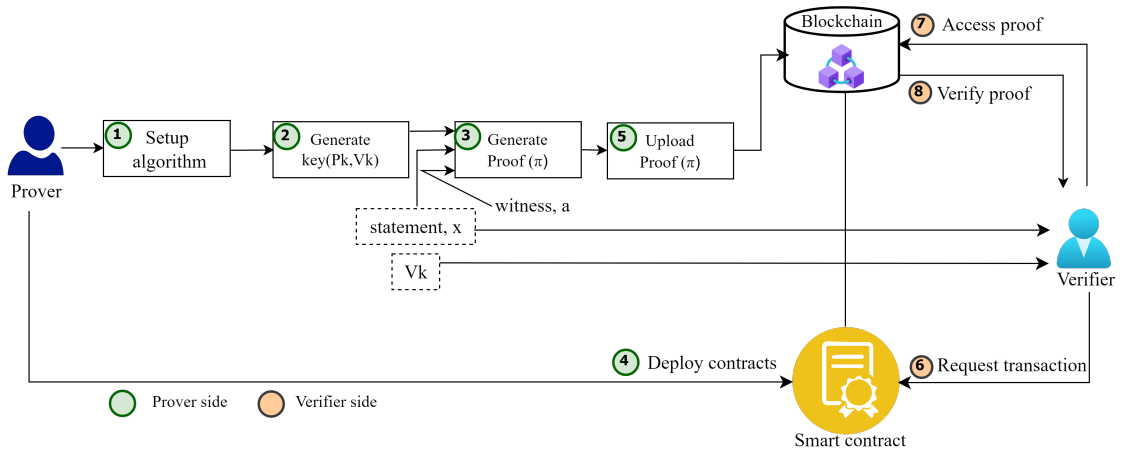

Figure 2: zk-SNARKs integration to the system for transaction verification.

Step 2. Generate Key (pk, vk)

Next, based on the data provided by the user and the specific requirements related to blockchain technology, a set of constraints representing $Constraints = constraint1, constraint2, ..., constraintM$, the necessary characteristics must be created and designated. The key generator method then produces a verification key (vk) and a proving key (pk) given a circuit C. The prover utilizes the proving key to produce proofs and the verifier uses the verification key to confirm the proofs.

Step 3. Generate proof (π)

The prover utilizes the zk-SNARKs proof generation algorithm that has been modified for blockchain applications. It takes a proving key (pk), a public statement (x), and a private witness (a) as inputs to generate a non-interactive proof (π) for the given statement. This proof showcases the correlation defined by the circuit C among the variables x and a. This evidence verifies the accuracy of the identity-related data without disclosing the true values. To do this, one must solve the zk-SNARKs equation in the context of the blockchain: $blockchain - proof = prove(pk, x, a)$.

Step 4. Deploy contracts

Before the system is initiated, the sender and recipient explicitly declare the agreements and terms of the transactions. Smart contracts are used to write these agreements. Upon creation of the contract, the sender immediately deploys it to start the system from beginning. The remaining smart contract operations are then carried out in accordance with the sender's and receiver's specifications.

Step 5. Upload proof

Next stage to upload the generated blockchain specific proof on the blockchain, linking it to the user's blockchain address or identification is a common practice.

Step 6. Request transaction

The verifier now wants to verify the proof, as it is stored successfully. For this the verifier need to make a transaction request first to access the proof stored in blockchain.

Step 7. Access proof

Following a transaction that was completed successfully, the verifier now can access the blockchain proof to verifity it. The zk-SNARKs verification parameters like public statement (x), verification key (vk) also needed as a input to verify the proof.

Step 8. Verify proof

Once the user's blockchain address or identification has been associated with the blockchain, the verifier accesses the distinct identity evidence stored on the blockchain. In assessing the credibility of the data, the verifier employs the proof verification algorithm with inputs such as a verification key (vk), a public statement (x), and a zk-SNARK proof (π). The following equation is used for verification: $Verify(vk, x, \pi) = true\ or\ false?$

In the realm of blockchain verification, it entails evaluating the unique restrictions within the proof specific to blockchain and ensuring their integrity while safeguarding the concealment of gen-

uine identity attributes. Upon achieving precise verification, the blockchain-specific proof is validated effectively, enabling the verifier to trust the authenticity of identity traits within the blockchain framework without the need to access the user's private information.

We will get the solidity file *(verifier.sol)* based on the proof generated in this section. This proof will be used as an input parameter for the transaction verification in the next section (section 4.2). We also have to import this solidity file (*verifier.sol*) into the solidity file of key exchange scenario (section 4.2) to integrate the zk-SNARKs concept to the proposed system.

4.2 Encrypted Key Sharing using Blockchain

An illustration depicting the key exchange process of the proposed system can be observed in Figure 3. Prior to utilizing the blockchain network, both the sender and receiver are required to complete the registration process. The solidity file (verifier.sol) that already generated (section 4.1) using ZoKrates tool have to import first to the proposed system's solidity file for transaction verification. The document outlines a series of 8 steps, each of which is detailed as follows:

Step 1. Generate random key

Initially, the sender initiates the process by creating a random key which is then utilized to encrypt the content of the email. Within our suggested framework, this random key is produced through the utilization of a random key generator. The generation of random bytes to match the specified key length was achieved by employing the **randomBytes()** function. Notably, PGP typically adopts a random key length ranging from 128 to 256 bits. In our study, we opted for key lengths of 128-bits (16 bytes), 192-bits (24 bytes), and 256-bits (32 bytes) respectively. Following this, the generated random bytes are converted into a hexadecimal string, thereby enhancing readability. Following this, the hexadecimal sequence is adjusted in order to match the specified key length, producing a modified hexadecimal sequence identified as the random key, denoted as k.

Step 2. Execute encryption (ECC algorithm)

Once the key is produced, the ECC algorithm is employed to encode the key at the sender's end, resulting in the encrypted key, k_e. Subsequently, this k_e is conveyed from the sender to the recipient through the blockchain framework we have put forward. Upon receiving the k_e, the recipient proceeds to decrypt it using the identical ECC algorithm, thereby obtaining k.

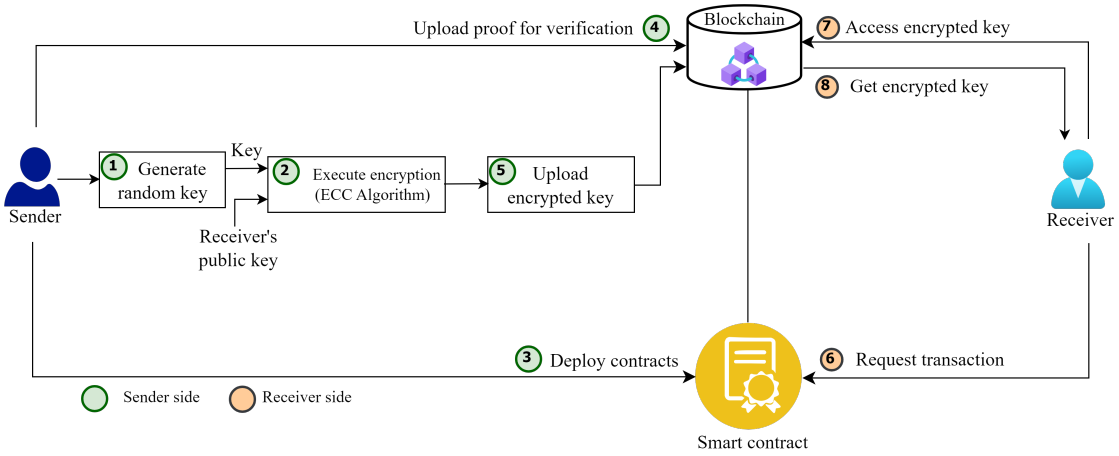

Figure 3: Proposed system to share the encrypted key.

Encryption and Decryption:

Let's examine an elliptic curve $E_p(a, b)$, where the curve equation is defined by parameters a and b. The curve is located in a finite field denoted as $\mathbb{F}_p$, where p represents a prime number. The base point G and its order n are also established. The sender initially generates a private key P_{rA} $(0 < P_{rA} < n)$ in order to calculate the public key $P_A = P_{rA} \cdot G$. Similarly, the recipient formulates a private key P_{rB} $(0 < P_{rB} < n)$ to determine the corresponding public key $P_B = P_{rB} \cdot G$.

To initiate the encryption process, the sender encrypts a randomly generated key k. Initially, a random number r ($0 < r < n$) is selected as a temporary private key, and the temporary public key $R = r \cdot G$ is computed. Subsequently, the shared secret point $S = r \cdot P_B$ is determined, and the S_x-coordinates are used to derive a symmetric key S_k for the encryption of the random key k, leading to the generation of the encrypted key $k_e = \{S_k, k\}$.

For decryption, the recipient first acquires the encrypted key k_e from the blockchain and the temporary public key R from the key server typically utilized in standard PGP procedures. Subsequently, the shared secret point S is computed as the product of R and P_{rB}, resulting in the derivation of the shared secret S_x-coordinates serving as a symmetric key S_k. Following this, the decryption of k_e is performed to recover the random key $k = \{S_k, k_e\}$.

Step 3. Deploy contracts

The terms and conditions are clearly defined by both the sender and recipient before initiating the system. These terms are formalized through the use of Solidity programming language, known as smart contracts. Upon the establishment of the contract, the sender promptly activates it at the beginning to commence the system. Subsequently, the remaining functions of the smart contracts are carried out in accordance with the specifications of the sender and receiver.

Step 4. Upload proof for verification

When the originator of a transaction requests for the inclusion of the encrypted key k_e (generated in step 2) in a blockchain block, the transaction undergoes initial verification. In order to achieve this, the sender is required to provide the proof for verification, which the recipient will assess in conjunction with the statement and verification key. If the verification of the proof is validated for the transaction, it will proceed successfully; otherwise, the transaction request will be disregarded.

Step 5. Upload encrypted key

Following the completion of a successful transaction, the individual initiating the transaction transmits the encrypted key k_e to be stored on the blockchain, where it is subsequently recorded as transaction data.

Step 6. Request transaction

The recipient is currently seeking to retrieve the encrypted key k_e, which has been securely stored. To achieve this, the recipient initiates a transaction request within the operational smart contract to obtain k_e. The success of this request hinges on the validity of the transaction proof; failure to validate the proof will result in the transaction request being disregarded.

Step 7. Access encrypted key

Following the completion of the transaction, the recipient is now able to retrieve the secured key that is stored within a block of the blockchain.

Step 8. Get encrypted key

Upon obtaining access to the block, the recipient is able to retrieve the encrypted key k_e from the blockchain. The recipient then proceeds to employ the ECC algorithm in order to decrypt k_e and obtain the key k.

4.2.1 Smart Contract Execution

The suggested system is described in this part, along with the features that are needed. The construction, implementation, and execution of smart contracts are also explained.

1. Smart Contract Creation

According to the proposed approach, three functions were created in the intelligent contract to enable the sender to store the encrypted key k_e on the blockchain and for the recipient to retrieve it. These functions include: **RegisterUser()**, **UploadData()**, and **AccessData()** functions. Using the **RegisterUser()** function, all users must first register in the system before proceeding. We took into consideration a case study situation in order to construct smart contracts with the following characteristics in our suggested solution. Let's say that in a private company, the "Manager" and the "Chief Executive Officer (CEO)" want to exchange the encrypted key k_e. According to this scenario, users must fill out registration forms with input fields like "USERNAME", "USEREMAIL", "ORGANIZATION", and "POSITION". In order to register a user, we made "USEREMAIL" a

distinct parameter and an email address that is only used once. k_e is stored in the block and retrieved from the block using the **UploadData()** and **AccessData()** functions, respectively.

2. Smart Contract Deployment

The smart contract must establish a connection with the Ethereum testing platform prior to being deployed. We validated the contract on the Ganache platform using our suggested method. To begin with, the contract is linked to the Remote Procedure Call (RPC) server, which enables communication between external apps and the Ethereum blockchain. The contract then needs to be deployed in order for it to be accessible on the Ethereum network. The contract's bytecode is immutable once it is deployed and is kept on the blockchain as transaction data. The logic and state of the contract are guaranteed to be consistent and safe due to its immutability.

4.2.2 Access Control Transactions

Smart contracts are used to build negotiation and evaluation policies for the access control transactions in the proposed framework. The individuals capable of uploading and retrieving the encrypted key k_e on the blockchain are identified through access control within the proposed system. The process of the transaction **UploadData()** is outlined in Figure 4. The sender's delineation of the steps involved in the **UploadData()** transaction is as follows:

I) After the smart contract is embedded and k_e is generated, the sender deploys it onto the blockchain.

II) The sender submits an **UploadData()** transaction to save k_e in the blockchain.

III) The transaction is executed by the smart contract within the blockchain network.

IV) After every user on the blockchain network has confirmed the transaction, a new block will be created for that transaction. Lastly, the newly formed block contains the k_e.

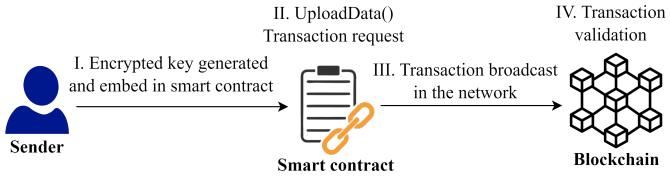

Figure 4: UploadData() transaction: Sender upload data.

The user is requesting access to the k_e by utilizing the **AccessData()** transaction depicted in Figure 5. The specific procedures for a recipient requesting access to k_e are outlined below:

I) The recipient commences an **AccessData()** transaction in order to seek authorization to access k_e from the block.

II) The blockchain network receives the transaction revealed by the smart contract.

III) A new block is added in accordance with the confirmed transaction when all users of the blockchain network have validated it. The k_e is now available to the receiver via the blockchain network.

5 Implementation and Discussion

This section highlights the simulation environments, result analysis of the system and the comparison of proposed work with previous work in details. The necessary tools for the off-chain and on-chain environment are summarized in section 5.1. After creating the proof, and transmiting the encrypted

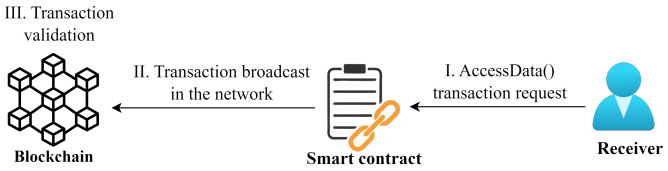

Figure 5: AccessData() transaction: Receiver asking to access data.

key through blockchain, the necessary results are highlighted in section 5.2. In section 5.3, we described the benefits to integrate zk-SNARKs with blockchain in our proposed system. We also explain the weakness of our smart contracts in details in this section.

5.1 Implementation

In order to assess the practicality of the suggested system, the specifics of the assessment are delineated in this section. The deployment settings, framework, essential technical resources, and software components are outlined in Table 1.

Table 1: Implementation Environments

Software	**Configuration/Version**
OS	Windows 11 pro
CPU	3.20 GHz Intel Core i7
RAM	16 GB memory
Truffle	5.11.0
Ganache	2.7.1
Node.js	16.16.0
ZoKrates	0.8.4
Remix	0.8.22

In order to implement our proposed system, we take into account two distinct environments. The first one being the off-chain environment, which operates independently from the blockchain, while the second is the on-chain environment, functioning within the parameters of the blockchain. Our proposed system specifies that the processes of proof generation, key generation, key encryption, and key decryption are to be carried out outside of the blockchain.

We used ZoKrates tool to execute the zk-SNARKs concept in our solidity code. To integrate zk-SNARKs into the solidity code using ZoKrates involves creating a seperate file **ZoKrates (.zok)** first. This file describes the logic of the zero-knowledge proof. The proof is generated based on the statement of the transaction, appropriate witness and the corresponding proving key. In our research, it was found that within a private institution, a scenario may arise where the "Chief Executive Officer (CEO)" intends to provide the encrypted key to the "Manager". For doing this, we have a function in the solidity file called **UploadData()**, using this the "CEO" can upload the encrypted key into the blockchain. The input paramaters of the **UploadData()** are "USERNAME", "USEREMAIL" and "POSITION". So we want to create a zk-SNARKs proof that indicating the transaction comes from "CEO" of the organization without revealing any input information about the user uploading the key. To generate the proof, we have to follow several steps. Firstly, have to create a file **ZoKrates (.zok)** for the **UploadData()** to demonstrate that the user has the authority to upload data. The statement we want to prove is that the user uploading the key is the "CEO" of the organization. Secondly, we have to specify the witness of the proof. The witness is a set of private inputs that satisfy the constraints of ZoKrates file. In this case, the witness includes information that proves the "CEO" status without revealing the actual CEO's identity. The witnesses for this case are as follows:

- a (User is registered): Witness for $a : 1$ (indicating the user is registered).

- b (User's company is ABC): Witness for b : 1 (indicating the user is associated with the company ABC).
- c (User's position is CEO): Witness for c : 1 (indicating the user holds the position of CEO).

Thirdly, based on the statements and the witnesses, ZoKrates generate a set of keys called proving key and verification key to generate and verify the proof, respectively. Finally, the proof is generated based on the above statement, witnesses and corresponding proving key. After that, the proof is verified based on the verification key produced by ZoKrates tool. Based on this proof, it will be decided whether the transaction will be valid or not. The proof generated based on our proposed system in shown in Figure 6.

After generating the proof, the verifier key is used by the aggregator associated smart contracts. This file **(Verifier.sol)** contains the Solidity code for the ZoKrates verifier contract. Now, we have to import this verifier contract **(Verifier.sol)** to the main contract **(UserRegistration.sol)** of our proposed system. To do so, we need to include the content of the **"Verifier.sol"** file directly into the **"UserRegistration.sol"** file. According to the the Proof struct in the **"Verifier.sol"** file, we need to modify the main contract **(UserRegistration.sol)** file to replace Pairing.G1Point and Pairing.G2Point with the actual values from the proof generated by the ZoKrates. We need to extract the individual components of proof and construct the Pairing.G1Point and Pairing.G2Point objects accordingly. Based on this proof, now we can modify the **UploadData()** of the main contract.

Figure 6: The generated proof using zk-SNARKs.

In the context of key exchange, the initial step involves the generation of a random key k by the user. Subsequently, the sender employs the receiver's public key to encrypt the random key, resulting in the encrypted key k_e. For this specific scenario, a random key of size $k = 32$ bytes (equivalent to 256-bits) was selected. The encryption of a specific value k was implemented through the utilization of ECC-256-bit key pair encryption, resulting in an encrypted key size of $k_e = 48$ bytes (equivalent to 384 bits).

In the realm of on-chain activities, Truffle was employed as a development framework, with ganache being utilized as the testing ground for Ethereum. The encrypted key k_e, which stems from an off-chain environment, is first retrieved within the solidity code. It is then required to be conveyed through the smart contract in order to be displayed on the ganache platform. The function **UploadData()** accepts the encrypted key k_e as input and forwards it as a parameter in

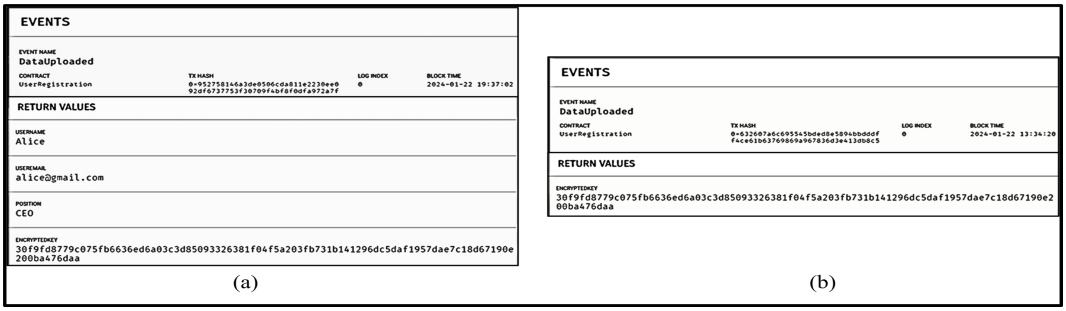

Figure 7: Transaction output for UploadData() (a)without zk-SNARKs and (b) with zk-SNARKs.

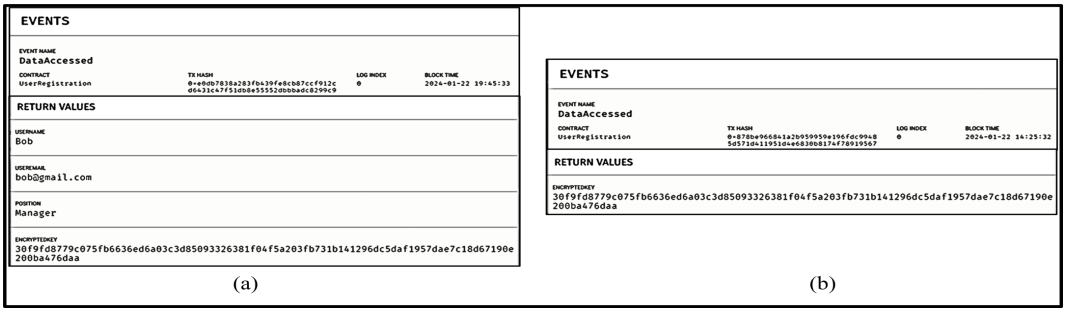

Figure 8: Transaction output for AccessData() (a)without zk-SNARKs and (b) with zk-SNARKs.

the process. The function is subsequently called upon as a transaction for the secure storage of k_e in the blockchain. Once the transaction is successfully completed, the bytecode of the transaction becomes visible as transaction data on the ganache platform. In order to access the original data from the blockchain, events were produced within the function to convey information from the smart contract to the user. A recipient then initiates an **AccessData()** transaction to retrieve k_e from the blockchain. The outcomes of the **UploadData()** transaction and the **AccessData()** transaction, both with and without zk-SNARKs principles, are depicted in figure 7 and figure 8 correspondingly.

In this illustration, we designate "Alice" as the sender and "Bob" as the recipient. Furthermore, we utilize the email addresses "alice@gmail.com" and "bob@gmail.com" to establish the sender and recipient within the system. After that, we consider two environments, one is without zk-SNARKs and another is with zk-SNARKs to upload and access the encrypted key in blockchain and to observe the transaction output. In case of without zk-SNARKs, we need to put "USERNAME", "USEREMAIL" and "POSITION" as an input parameter to execute the transaction verification. We also make the "POSITION" of the sender as an unique parameter to valid the transaction.

On the other hand, when we integrated zk-SNARKs, at that case, we do not need to put any input parameter to validate the transaction. Only proof is required to verify the transaction. As the proof is generated based on the input statements and appropriate witness. As a result, the transaction verification process becomes more simpler. Moreover, it does not need to show any identity of the prover to verify the transaction that increases the privacy of the transaction.

After a transaction that was deemed successful, the sender deposits the value k_e into the block, enabling the receiver to subsequently retrieve k_e from the block. Upon obtaining k_e, the receiver employs their individual key to decipher k_e and obtain the generated key k.

5.2 Results

A novel system for secure key-sharing in email applications was developed based on blockchain technology, incorporating zk-SNARKs for enhancing transaction privacy. The smart contracts within this innovative system were meticulously compiled and deployed on the Ethereum blockchain plat-

form, where gas serves as the standard unit for quantifying the computational work carried out. An informative comparison detailing the gas consumption and computational cost associated with the execution of smart contract operations, with and without the integration of zk-SNARKs, can be found in Table 2. The exchange rate between Ethereum and the US dollar as of October 12, 2023, was established at 1 Ether equaling 1529.41 USD, indicating the prevailing market valuation. The minimum gas price required for executing a trade stands at 1 GWEI, which is equivalent to 10^{-9} Ether. It is important to note that the gas values presented are based on the test Ethereum network and do not involve real cryptocurrency.

Table 2: The gas and cost of performing several operations with and without zk-SNARKs

Operation	**Gas used (GWEI)**		**Tx cost (Ether)**		**Tx cost (USD)**	
	Without zk-SNARKs	With zk-SNARKs	Without zk-SNARKs	With zk-SNARKs	Without zk-SNARKs	With zk-SNARKs
Deployment	1743924	1089729	0.001744	0.001090	2.6672	1.6671
RegisterUser()	134916	134916	0.000135	0.000135	0.2064	0.2064
UploadData()	57144	26221	0.000057	0.000026	0.0872	0.0398
AccessData()	36083	24866	0.000036	0.000025	0.0551	0.0382

Table 3: The percentage of gas reduction due to the integration of zk-SNARKs

Operation	**Gas (GWEI) reduction**
Deployment	37.5%
RegisterUser()	-
UploadData()	54.1%
AccessData()	31.1%

In our proposed system, there were four main operations of smart contracts. Every time, we calculated the used gas and the corresponding execution cost of operations considering with and without zk-SNARKs concept. Initially, the cost associated with the implementation of smart contracts is computed. This initial cost is determined upfront to establish the system. Subsequently, the cost of executing various operations within smart contracts is calculated. From this table, we can observe that for the operation of **Deployment**, **UploadData()**, and **AccessData()** the required gas is lower while integrating zk-SNARKs concept. As a result, the corresponding transaction cost also becomes lower as shown in Figure 9, considering the transaction with zk-SNARKs than the transaction without zk-SNARKs. This figure illustrates that the cost for the **Deployment**, **UploadData()**, and **AccessData()** becomes lower due to the integration of zk-SNARKs. While the cost for the **RegisterUser()** function is still same for both with and without zk-SNARKs cases. This is due to register a user we considered the same input parameters for with and without zk-SNARKs environments. The total cost required for the system without zk-SNARKs and with zk-SNARKs are \$3.0159 and \$1.9515, respectively. So integrating zk-SNARKs, it reduces the system total cost approximately 35.3%. This analysis illustrates the viability of the suggested approach in practical applications, specifically in relation to the necessary gas values and computational cost [30].

Due to the integration of zk-SNARKs, the amount of gas reduction is investigated in Table 3. From this table, we can observe that integrating zk-SNARKs, the required gas reduce for deployment, **UploadData()** and **AccessData()** while for **RegisterUser()** the gas was same as previous. It shows 37.5%, 54.1% and 31.1% gas reduction for the deployment, **UploadData()** and **AccessData()** functions respectively due to the integration of zk-SNARKs. It makes the system more feasible.

Smart contracts have the capability to be utilized within an email platform for the secure transmission of encrypted keys through the enforcement of predetermined rules governing the transaction. The integration of smart contracts within our system guarantees the resolution of the following concerns:

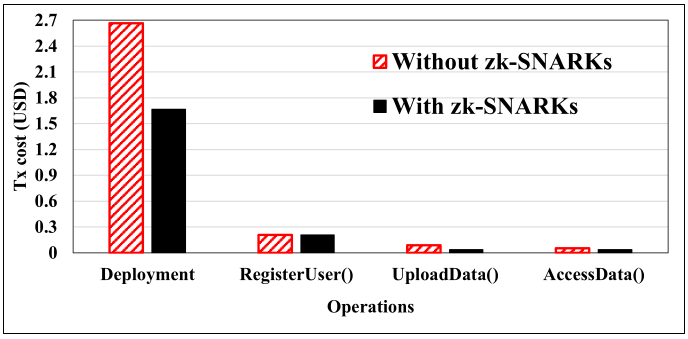

Figure 9: Transaction cost of the system with and without zk-SNARKs.

Firstly, the action of storing the secured key within the blockchain, the sender must issue a transaction request under our suggested approach. Nevertheless, for the receiver to be able to retrieve the encrypted key stored on the blockchain, a request for a transaction has to be sent. In accordance with the stipulations specified in the smart contract agreements, every network user confirms the transaction. A transaction is only regarded as legitimate if it satisfies the agreements. For instance, let's say that in a private company, the "Manager" and the "Chief Executive Officer (CEO)" would want to share the encryption key. According to this scenario, the encrypted key will be kept stored if the transaction request is sent by the "Chief Executive Officer (CEO)" alone; if not, it will not. Likewise, access to the encrypted key will only be permitted if the transaction request originates from the "Manager"; otherwise, it will not be accepted. Therefore, the blockchain network may be used to specify the terms for the exchange of encrypted keys thanks to smart contracts. In PGP, however, the encrypted key is sent straight to the recipient by the sender, with no further requirements. The amount of unauthorised access to the encryption key may be decreased by these extra requirements.

Secondly, the details of the contract are made available to all blockchain network users. Before introducing a transaction to the blockchain network, all users check over and confirm each one. Transparency is therefore encouraged. Furthermore, a smart contract cannot be modified after it has been implemented. It eliminates the potential for code modifications to the contract.

Thirdly, in order to complete the transaction successfully within the suggested framework, both the sender and the recipient must follow a number of procedures. It does not need the assistance of Man-in-the-Middle to carry out the steps, despite the fact that passing them is necessary. A smart contract contains the predetermined terms of the transaction. These terms are automatically evaluated and validated throughout the transaction. The system becomes automated by smart contracts.

5.3 Discussion

The integration of zk-SNARKs in our framework ensure the following issues:

Data Minimization and Privacy Preservation: By enabling users to provide just the information that is required without disclosing extra details, zk-SNARKs promote data reduction and privacy protection. zk-SNARKs, which are part of our proposed system, allow the verification of a certain characteristic (like the user's position) without revealing additional personal information (such the user's name or email). By doing this, the likelihood of data breaches is decreased and sensitive data exposure is avoided. Moreover, it also improves the data privacy.

Non-Interactivity: Because zk-SNARKs are non-interactive, there is no requirement for bidirectional communication to occur between the entity providing information and the entity verifying it during the process of creating and verifying the proof. This makes the verification procedure easier and is especially beneficial for blockchain transactions.

Trustless Verification: By enabling trustless verification, zk-SNARKs do away with the need that parties have faith in one another. Without depending on the prover's integrity, the verifier may

independently confirm the reliability of the evidence.

Besides, smart contract faces some weakness described by Chen et al. [44]. We used Ethereum platform and solidity code for smart contracts that may faces the following challenges:

- Access Control Policy: We used access control policy which is one of the key parameters in smart contract security. Certain operations that require a high level of sensitivity are limited to particular users for this reason. In our scenario, we also used the check statements/conditions for the authorization of sender and receiver to exchange encrypted key securely. If the check is missing or invalid, attackers may be able to access the system and carry out risky actions which could ultimately result in vulnerabilities. To mitigate this issue, careful consideration is needed while writing the conditions.

- Use of *require*() function: In our proposed system, we used *require*() function for input validation. We created some specific conditions such as "USEREMAIL" should be unique and "POSITION" of the user should be fixed to execute the transaction. Prior to the execution of the remaining contract code, certain requirements need to be satisfied. Failure to meet these conditions will result in the transaction being reverted and an exception being raised.

Table 4: Comparison between Yakubov et al. [3] and proposed work

Characteristics	**Yakubov et al. [3]**	**Proposed**
Blockchain development platform used	Ethereum	Ethereum
PGP key generation	Conventional (RSA-3072)	ECC-256
Block size to store the encrypted key	Large memory required (encrypted key size=384 bytes)	Less memory required (encrypted key size=48 bytes)
New block should be approved by more than 50% of nodes for authorization	Yes	Yes
Gas and cost analysis of smart contracts (both ether and USD)	No	Yes
MITM risks solve by the default characteristics of blockchain	Yes	Yes
Zero Knowledge proof concept for privacy protection	Not considered	Considered

In their study, Yakubov et al. [3] presented a framework for managing PGP that leveraged blockchain technology to quickly disseminate the process of revocation certificates on PGP key servers. In contrast, our research aims to the secure transmission of the encrypted key used in the PGP from the sender to the receiver using blockchain technology while maintaining the confidentiality of the transactions. A detailed comparison between the work of Yakubov et al. [3] and our proposed approach is provided in Table 4.

Both the work consider Ethereum as the development platform of blockchain. However, we used ECC algorithm to generate the PGP key rather than the conventional RSA algorithm. Using ECC, we produced smaller sizes encrypted key which are feasible to store in the block considering the storage capability of blockchain. We calculated the gas and cost required for every functions of smart contract to show the practicality of our proposed system. Moreover, we consider the user privacy protection by using zk-SNARKs.

6 Conclusion

In the present study, the suggestion was put forth to combine Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARKs) and blockchain technology with PGP's key sharing system in order to improve the degree of safeguarding personal information. This ensures data

verification without revealing sensitive details, strengthening privacy protection. We also employed Elliptic Curve Cryptography (ECC) to keep the PGP key confidential. This combined approach enhances the security of the PGP key, ensuring confidentiality, integrity, and user privacy. Additionally, we assessed gas consumption and transaction costs with and without zk-SNARKs. The results showed that our proposal consume less gas consumption and transaction costs compared to scenarios without zk-SNARKs. In future work, we will measure the time taken for generating and verifying proofs in the zk-SNARKs scheme, considering varying lengths of the encrypted key.

References

[1] Esra Altulaihan, Abrar Alismail, M. M. Hafizur Rahman and Adamu A. Ibrahim. Email Security Issues, Tools, and Techniques Used in Investigation. *Sustainability*, 15(13):1-28, 2023.

[2] Gurpal S. Chhabra and Dilpreet S. Bajwa. Review of the e-mail system, security protocols, and email forensics. *International Journal of Computer Science and Communication Networks*, 5(3):201-211, 2015.

[3] Alexander Yakubov, Wazen Shbair, Nida Khan, and Radu State. BlockPGP: A Blockchain-based framework for PGP Key Servers. *International Journal of Networking and Computing*, 10(1):1-24, 2020.

[4] Mandrita Banerjee, Junghee Lee, and Kim-Kwang R. Choo. A blockchain future for internet of things security: a position paper. *Digital Communications and Networks*, 4(3):149-160, 2018.

[5] Mohamed Fartitchou, Khalid E. Makkaoui, Nabil Kannouf, and Zakaria E. Allali. Security on Blockchain Technology. *In Proceedings of 3rd International Conference on Advanced Communication Technologies and Networking (CommNet)*, volume 1, pages 1-7, 2020.

[6] Zeli Wang, Hai Jin, Weiqi Dai, Kim-Kwang R. Choo, and Deqing Zou. Ethereum smart contract security research: survey and future research opportunities. *Frontiers of Computer Science*, 15(2):1-18, 2020.

[7] Shafaq N. Khan, Faiza Loukil, Chirine Ghedira-Guegan, Elhadj Benkhelifa, and Anoud Bani-Hani. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-Peer Network Application*, 14(5):2901-2905, 2021.

[8] Diego Piedrahita, Javier Bermejo, and Francisco Machío. A Secure Email Solution Based on Blockchain. *Blockchain and Applications*, 320(1):355-358, 2021.

[9] R.Vasantha and R. Satya Prasad. Secured Email Data Based on Blowfish with Blockchain Technology. *Science, Technology and Development*, 8(1):456-464, 2019.

[10] Vasileios Dimitriadis, Leandros Maglaras, Nineta Polemi, Ioanna Kantzavelou, and Nick Ayres. Uncuffed: A Blockchain-based Secure Messaging System. *In Proceedings of the Pan-Hellenic Conference on Informatics (PCI)*, volume 25, pages 340-345, 2021.

[11] Hsiao-Shan Huang, Tian-Sheuan Chang, and Jhih-Yi Wu. A Secure File Sharing System Based on IPFS and Blockchain. *In Proceedings of the 2nd International Electronics Communication Conferenc*, volume 20, pages 96-100, 2020.

[12] M. Francisca Hinarejos, and Josep-Lluis Ferrer-Gomila. A Solution for Secure Multi-Party Certified Electronic Mail Using Blockchain. *IEEE Access*, 8(1):102997-103006, 2020.

[13] Arun Varghese. Email Verification Service using Blockchain. *Technical Disclosure Commons*, 2468(1):1-8, 2019.

[14] Jose C. Gonzalez, Vicente G. Diaz, Edward R. N. Valdez, Alberto Gomez, and Ruben G. Crespo. Replacing email protocols with blockchain-based smart contracts. *Cluster Computing*, 23(1):1795-1801, 2020.

[15] Md. Biplob Hossain, Maya Rahayu, Md. Arshad Ali, Samsul Huda, Yuta Kodera and Yasuyuki Nogami. A Smart Contract Based Blockchain Approach Integrated with Elliptic Curve Cryptography for Secure Email Application. *Eleventh International Symposium on Computing and Networking Workshops (CANDARW)*, pages 195-201, 2023.

[16] Jiahui Huang, Teng Huang, Huanchun Wei, Jiehua Zhang, Hongyang Yan, Duncan S. Wong, and Haibo Hu. zkChain: A privacy-preserving model based on zk-SNARKs and hash chain for efficient transfer of assets. *Transactions on Emerging Telecommunications Technologies*, 2022(1):1-11, 2022.

[17] Xiaoqiang Sun, F. Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. A Survey on Zero-Knowledge Proof in Blockchain. *IEEE Network*, 35(4):198-205, 2021.

[18] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin. Anonymous distributed Ecash from bitcoin. *In Proceedings 2013 IEEE symposium on security and privacy*, volume 1, pages 397-411, 2013.

[19] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. *In Proceedings 2014 IEEE symposium on security and privacy*, volume 1, pages 459-474, 2014.

[20] Laurent Chuat, Pawel Szalachowsky, Adrian Perrig, Ben Laurie, and Eran Messeri. Efficient gossip protocols for verifying the consistency of certificate logs. *In 2015 IEEE Conference on Communications and Network Security (CNS)*, volume 1, pages 1-9, 2015.

[21] Hiroaki Ananda, Junpei Kawamoto, Jian weng, and Kouichi sakurai. Identity-embedding method for decentralized public-key infrastructure. *In Proceedings of International Conference on Trusted Systems*, volume 9473, pages 1-14, 2014.

[22] Yannan Li, Yong Yu, Chunwei Lou, Nadra Guizani, and Lianhai Wang. Decentralized Public Key Infrastructures atop Blockchain. *IEEE Network*, 99(1):1-7, 2020.

[23] Alfonso d. L. R. Gómez-Arevalillo, and Panos Papadimitratos. Blockchain-based Public Key Infrastructure for Inter-Domain Secure Routing. *International Workshop on Open Problems in Network Security (iNetSec)*, volume 1, pages 20-38, 2017.

[24] Abu S. Ahmed, and Tuomas Aura. Turning Trust Around: Smart Contract-Assisted Public Key Infrastructure. *In 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications*, volume 1, pages 104-111, 2018.

[25] Louise Axon and Michael Goldsmith. PB-PKI: a Privacy-Aware Blockchain-Based PKI. *In Proceedings of the 14th International Joint Conference on e-Business and Telecommunications* , volume OICETE, pages 311-318, 2017.

[26] Mustafa A. Bassam. SCPKI: A Smart Contract-based PKI and Identity System. *In Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, volume 1, pages 35-40, 2017.

[27] Alexander Yakubov, Wazen M. Shbair, Anders Wallbom, David Sanda, and Radu State. A Blockchain-Based PKI Management Framework. *In: NOMS 2018–2018 IEEE/IFIP Network Operations and Management Symposium*, volume 1, pages 1-6, 2018.

[28] Duane Wilson, and Giuseppe Ateniese. From Pretty Good To Great: Enhancing PGP using Bitcoin and the Blockchain. *In Proceedings of International Conference on Network and System Security*, volume 1, pages 368-375, 2015.

[29] Silas Nzuva. Smart Contracts Implementation, Applications, Benefits, and Limitations. *Journal of Information Engineering and Applications*, 9(5):63-75, 2019.

[30] Akanksha Saini, Qingyi Zhu, Yong Xiang, Longxiang Gao, and Yushu Zhang. Smart- Contract-Based Access Control Framework for Cloud Smart Healthcare System. *IEEE Internet of Things Journal*, 8(7):5914-5925, 2021.

[31] Zhangshuang Guan, Zhiguo Wan, Yang Yang, Yan Zhou, and Butian Huang. BlockMaze: An efficient privacy-preserving account-model blockchain based on zk-SNARKs. *IIEEE Transactions on Dependable and Secure Computing*, 19(3):1446-1463, 2022.

[32] Zhiming Song, Guiwen Wang, Yimin Yu, and Taowei Chen. Digital identity verification and management system of blockchain-based verifiable certificate with the privacy protection of identity and behavior. *Security and Communication Networks*, 2022(1):1-24, 2022.

[33] Lei Xu, Nolan Shah, Lin Chen, Nour Diallo, Zhimin Gao, Yang Lu and Weidong Shi. Enabling the sharing economy: Privacy respecting contract based on public blockchain. *In Proceedings of the ACM workshop on blockchain, cryptocurrencies and contracts*, volume 1, pages 15-21, 2017.

[34] Dongkun Hou, Jie Zhang, Sida Huang, Zitian Peng, Jieming Ma, and Xiaohui Zhu. Privacy-preserving energy trading using blockchain and zero knowledge proof. *In Proceedings 2022 IEEE international conference on blockchain (blockchain)*, volume 1, pages 408-412, 2022.

[35] Ying Zhang. Increasing cyber defense in the music education sector using blockchain zero-knowledge proof identification. *Computational Intelligence and Neuroscience*, 2022(1):1-7, 2022.

[36] Ke Yuan, Yingjie Yan, Tong Xiao, Wenchao Zhang, Sufang Zhou, and Chunfu Jia. Privacy-protection scheme of a credit-investigation system based on blockchain. *Entropy*, 23(12):1-15, 2021.

[37] Zhengwei Ren, Xianye Zha, Kai Zhang, Jing Liu, and Heng Zhao. Lightweight protection of user identity privacy based on zero-knowledge proof. *In Proceedings 2019 IEEE International conference on system, man and cybernetics (SMC)*, volume 1, pages 2549-2554, 2019.

[38] Hasan Al-Aswad, Hesham Hasan, Wael Elmedany, Mazen Ali, and Chitra Balakrishna. Towards a blockchain based zero-knowledge model for secure data sharing and access. *In Proceedings 2019 7th International conference on future internet of things and cloud workshops (FiCloudW)*, volume 1, pages 76-81, 2019.

[39] Antoni E. B. Tomaz, Jose C. D. Nascimento, Abdelhakim S. Hafid, and Jose N. D. Souza. Preserving privacy in mobile health systems using non-interactive zero-knowledge proof and blockchain. *IEEE Access*, 8(1):204441-204458, 2020.

[40] Claudia D. Pop , Marcel Antal, Tudor Cioara, Ionut Anghel, and Ioan Salomie. Blockchain and demand response: Zero-knowledge proofs for energy transactions privacy. *Sensors*, 20(19):1-21, 2020.

[41] Wanxin Li, Hao Guo, Mark Nejad, and Chien-Chung Shen. Privacy-preserving traffic management: A blockchain and zero-knowledge proof inspired approach. *IEEE Access*, 8(1):181733–181743, 2020.

[42] SoonHyeong Jeong and Byeongtae Ahn. Implementation of real estate contract system using zero knowledge proof algorithm based blockchain. *The Journal of Supercomputing*, 77(10):11881–11893, 2021.

[43] Keke Gai, Haokun Tang, Guangshun Li, Tianxiu Xie, Shuo Wang, Liehuang Zhu, and Kim-Kwang R. Choo. Blockchain-based privacy preserving positioning data sharing for IoT-enabled maritime transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 24(2):2344–2358, 2022.

[44] Jiachi Chen, Mingyuan Huang, Zewei Lin, Peilin Zheng, and Zibin Zheng. To Healthier Ethereum: A Comprehensive and Iterative Smart Contract Weakness Enumeration. pages 1-12, 2023. arXiv:2308.10227 [cs.SE].