Blockchain-based Zero Knowledge Proof Platform

***Abstract*— Zero knowledge proofs (ZKPs) have emerged as a pivotal solution for enhancing data privacy within blockchain technology. Among various ZKP systems, zk-SNARKs stand out for their succinct proof size and single-round communication. However, a significant limitation of zk-SNARKs is their reliance on a trusted third party to generate shared parameters, such as the Common Reference String (CRS). In this paper, we introduce a novel blockchain-based ZKP platform that enables the use of zk-SNARKs without the need for a trusted third party. Our platform, named BCZKP, replaces the traditional trusted third party with a combination of smart contracts and non-fungible tokens (NFTs). The BCZKP framework comprises three main components: a smart contract for NFT creation, a CRS NFT, and a ZKP NFT. The CRS NFT securely stores a commitment to the CRS, while the ZKP NFT holds the proving and verification keys, which are generated using the CRS detailed in the CRS NFT. To mitigate the blockchain's storage cost, only commitments are recorded on the blockchain. Furthermore, to reduce computational load on the blockchain, key generation and proof production processes are executed off-chain. This paper outlines the architecture of BCZKP and demonstrates its efficacy in enhancing blockchain data privacy without the dependence on a trusted third party.**

Keywords—Zero knowledge, zk-SNARK, blockchain, non-fungible token (NFT)

I. Introduction

The blockchain industry, initially sparked by the advent of Bitcoin [1], has rapidly evolved into a burgeoning market. Central to this industry are decentralized services [2, 3, 4] and digital assets, notably cryptocurrencies [5] and Non-Fungible Tokens (NFTs) [6]. Despite its growth, a critical challenge facing the blockchain industry is data privacy.

Inherent to blockchain technology is the principle that state changes are facilitated exclusively through transactions. These transactions, initiated by clients, are executed and subsequently recorded in blocks. To maintain the integrity and transparency of the blockchain, transactions must be verifiable by any participant in the network. This necessitates making all transaction-related data publicly accessible. However, this transparency raises significant privacy concerns, especially when transactions involve sensitive information such as social security numbers or bank account details.

To address this privacy issue, the concept of Zero Knowledge Proofs (ZKPs) has been introduced [7]. ZKPs allow for the concealment of private data while still enabling verification by anyone possessing the public parameters. In a ZKP system, a prover generates a zero knowledge proof using hidden inputs, public parameters, a specific circuit that defines the problem, and a proving key. A verifier then authenticates the proof using the public parameters, the same circuit, and a verification key. Both the proving and verification keys are derived from the Common Reference String (CRS) and the circuit.

The most widely recognized ZKP system is the Zero Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) [8, 9]. zk-SNARK is notable for its ability to produce succinct proofs that are relatively small in size, eliminate the need for repeated communication between the prover and the verifier, and effectively shield the prover's private data from the verifier.

Introducing zk-SNARK to blockchain has several problems:

- To utilize zk-SNARK, CRS must be generated in secure manner. Generating CRS makes trapdoor that must be excluded without leakage so the trusted third party takes the responsibility. The trusted third party is centralized and potential security hole regardless of security level of blockchain.
- Generating CRS, proving key, verification key, and zero knowledge proofs needs complex computations. These computations cannot be processed on-chain because of blockchain computation cost.
- CRS and the circuit size is too large to store on-chain. The original data should be stored off-chain and the data can be validated with some manner.
- The prover and verifier in zk-SNARK shares CRS and the circuit before creating and verifying the zero knowledge proof. Typical zk-SNARKs use secure channel protocol to share the data.

In this paper, we introduce a novel platform for blockchain-based zero knowledge proof systems, particularly focusing on zk-SNARK. Our platform, named BCZKP, innovatively generates the common reference string without the need for a trusted third party.

BCZKP is designed with a unique structure that includes two types of Non-Fungible Tokens (NFTs): the CRS NFT and the ZKP NFT. The CRS NFT is responsible for securely storing the CRS, while the ZKP NFT holds both the proving key and the verification key. This framework allows for efficient distribution and sharing of the CRS, proving key, and verification key among users through the selection of corresponding NFTs.

To optimize the system's performance, BCZKP incorporates an off-chain calculation module and off-chain public storage. This design choice significantly reduces the computational and storage burdens on the blockchain, thereby enhancing the overall efficiency and scalability of the platform.

The rest of this paper is organized as follows. Section II gives background information to understand the components of BCZKP. Section III shows the design of BCZKP. Section IV explains the detailed process of BCZKP. Section V shows the application using BCZKP. Section VI provides the conclusion and directions for future research.

II. Preliminaries

A. Zero knowledge proof system

Zero knowledge proof is the proof that does not reveal private data of the user while the proof is verifiable by the other users. Zero knowledge proof system includes two roles, a prover and a verifier. The prover creates zero knowledge proof with the private data and the verifier verifies the proof given by the prover.

Zero knowledge proof satisfies the 3 properties:

- Completeness: If the solution of the problem is true, then a prover can convince a verifier.
- Soundness: False prover cannot convince a verifier of a false solution.
- Zero knowledge: A verifier only knows that prover knows the solution of the problem is true and nothing else.

B. zk-SNARK

The Zero Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) stands out as the most prominent system within the domain of zero knowledge proofs. Traditional ZKP systems often necessitate multiple rounds of communication, wherein the verifier sends a random witness and the prover responds with the result of the computation based on the given witness. In contrast, zk-SNARKs streamline this process by enabling a single communication step, wherein a succinct proof is transmitted. This proof is not only compact in size but also requires relatively minimal verification time. A notable implementation of zk-SNARKs is Groth16 [9], which features a constant proof size and verification time, independent of the complexity of the circuit defining the problem.

However, zk-SNARKs come with the requirement of a trusted setup ceremony, such as Multi-Party Computation (MPC) [10] or Powers-of-Tau [11], during their initial configuration phase. This trusted setup ceremony traditionally involves a trusted third party, presenting a potential security vulnerability in the system.

III. Blockchain-based ZKP Platform Design

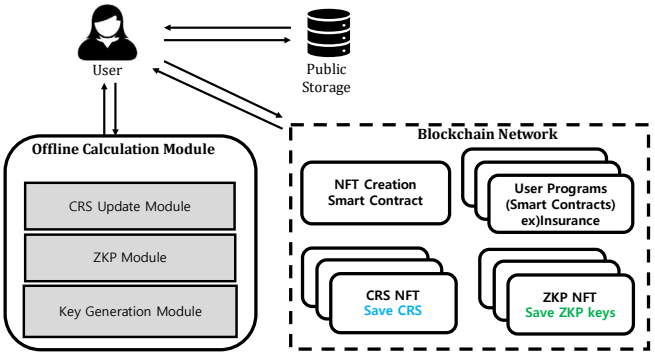

Fig. 1. Overall design of blockchain-based ZKP platform (BCZKP)

As shown in Fig. 1, our proposed blockchain-based ZKP platform consists of 3 components: Blockchain network, offline calculation module, and public storage.

In the blockchain network, the NFT creation smart contract is pivotal in managing both CRS NFTs and ZKP NFTs. The CRS NFT is responsible for storing the address and commitment of the common reference string, while the actual data is preserved in a public storage system. Meanwhile, the ZKP NFT, which references the CRS NFT, stores the commitments of a specific circuit, as well as the proving and verification keys. These keys are essential for the roles of the prover and verifier. The circuit itself is generated by compiling user programs and deploying smart contracts within the network.

Additionally, our platform features an offline calculation module, which comprises several key components. The CRS update module is responsible for refreshing the CRS, a process that involves extensive elliptic curve operations. The ZKP module facilitates the creation of zero knowledge proofs using the provided proving key, public parameters, and concealed inputs. Concurrently, the key generation module produces both the proving and verification keys, utilizing the CRS and the circuit.

Public storage is designed to be accessible to all network participants. Data stored in this public repository can be authenticated using the associated NFT. For example, the CRS is located at the address index specified in the CRS NFT. Verification of the CRS involves comparing the hash value computed by the downloader against the hash value in the CRS NFT.

In this ecosystem, users can assume various roles: a prover, who generates proofs; a verifier, who authenticates these proofs using the ZKP NFT; or a CRS updater, who updates the CRS as referenced by the CRS NFT and mints new CRS NFTs that encapsulate the previous CRS NFT.

IV. Detailed Process of BCZKP

A. CRS NFT and ZKP NFT

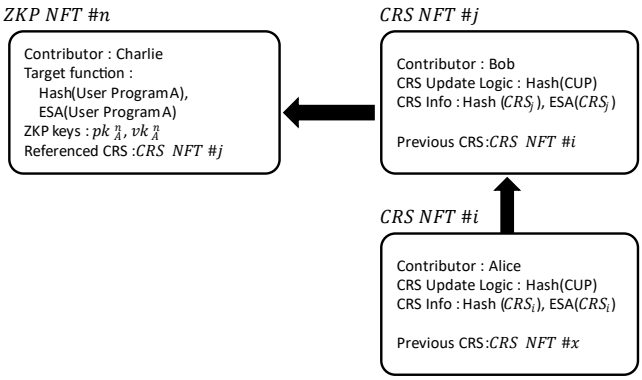

Fig. 2. Relation between CRS NFT and ZKP NFT

CRS NFT comprises several key indices: the contributor, CRS update logic, CRS information, and the previous CRS NFT. The 'contributor' field identifies the name of the individual responsible for updating the CRS. The 'CRS update logic' records the hash value of the circuit used in the validation process. 'CRS info' encompasses the hash value of the original CRS data alongside the External Storage Address (ESA) where this data is stored. The 'previous CRS' index refers to the earlier version of the CRS prior to its update by the named contributor. As illustrated in Fig. 2, CRS NFT #i is updated to CRS NFT #j by a contributor named Bob.

Similarly, the ZKP NFT includes several indices: the contributor, the target function, ZKP keys, and the referenced

CRS. The 'target function' pertains to the circuit compiled from user program A, encapsulating both the hash value and the address of the circuit. The 'ZKP keys' consist of a proving key and a verification key, which are generated using the CRS referenced in the CRS NFT and the target function. Fig. 2 demonstrates how ZKP NFT #n references CRS NFT #j.

All these NFTs are managed by the NFT creation smart contract, adhering to the ERC-721 standard for Non-Fungible Tokens [6].

B. Prover and Verifier in BCZKP

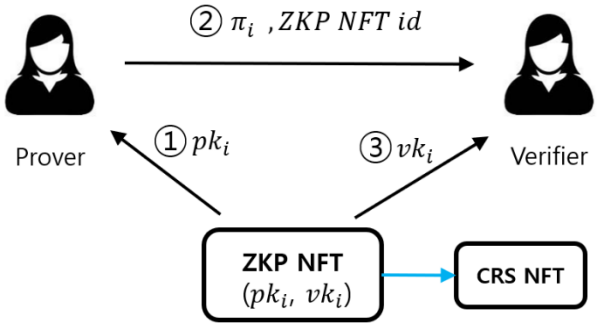

Fig. 3. Role of CRS NFT and ZKP NFT in BCZKP

In the zk-SNARK framework, it is imperative for the prover and verifier to share the CRS, public parameters, and the circuit. However, in our BCZKP platform, establishing a secure channel between the prover and verifier is unnecessary. As depicted in Fig. 3, the prover selects a ZKP NFT that corresponds to the target function they wish to prove. The prover then generates a zero knowledge proof using the proving key provided in the chosen ZKP NFT and their private input. After proof generation, the prover transmits both the proof and the ID of the selected ZKP NFT to the verifier. This ZKP NFT ID encapsulates all the information shared between the prover and verifier, including the CRS and the circuit. The verifier then authenticates the proof using the verification key contained in the selected ZKP NFT.

C. Public Storage

Initially, our design for the CRS NFT and ZKP NFT involved storing all CRS and circuit information directly within the NFTs. However, this approach proved impractical due to the high cost of blockchain storage, particularly for data amounting to megabytes in size. To address this, BCZKP utilizes public storage solutions, such as IPFS [12]. To ensure data consistency within this public storage, BCZKP records the data's commitment within the NFTs, along with the methodology used to create this commitment.

D. Offline Calculation Module

The initial design of BCZKP entailed performing all computations within the blockchain network. However, many blockchain networks implement transaction fees to prevent the misuse of resources, such as infinite loops [13, 14]. This fee structure often discourages the deployment of computationally intensive smart contracts. To circumvent high transaction costs associated with proof generation and CRS updates, BCZKP employs an offline calculation module, thereby optimizing efficiency and cost-effectiveness.

V. Application

Using BCZKP, we implemented insurance application. Insurance application and data exchange. As shown in Fig. 4, insurance application has two entities: a client and an insurance company.

Before starting the insurance application, the initialization

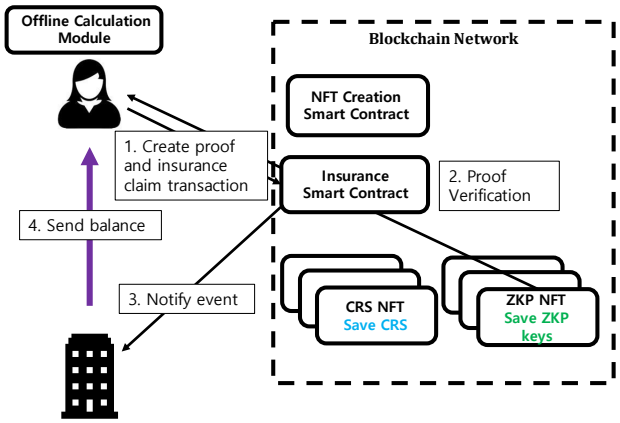

Fig. 4. Insurance application process

including deploy smart contracts, mint new CRS NFT by updating CRS, mint new ZKP NFT that saves proving key and verification key generated with new CRS NFT and the circuit that complies insurance smart contract as inputs. The ZKP NFT is linked with insurance smart contract so if a client wants to claim, the client must select linked ZKP NFT notified by the insurance smart contract. We assume that the input data for a client is given from the hospital department.

After initialization, a client creates a zero knowledge proof that proves the insurance claim is legitimate. Then the client sends insurance claim transaction to the insurance smart contract. Insurance smart contract verifies the proof by calling verify function in the contract. When the verification result is true, insurance smart contract emits the event to the insurance company. Finally, after notifying the valid insurance claim event, the insurance company sends balance to the client whose amount is determined in insurance contract.

In this scenario, we use ZKP NFT to notify (1) which problem would be proved (insurance claim), (2) which CRS and circuit are used to generate the proving key and the verification key. A client does not need to generate both keys personally since the keys are already written in the ZKP NFT. If the client wants to verify the validity of keys in ZKP NFT, the client can generate ZKP keys with CRS referenced in CRS NFT and the circuit saved in external storage address written in ZKP NFT.

VI. Conclusion

In this paper, we solve the data privacy problem in blockchain by adopting zero knowledge proof system, zk-SNARK. To avoid trusted third party that is necessary in zk-SNARK initialization, we proposes a new blockchain-based platform the zero knowledge proof system in blockchains. Our solution, BCZKP keeps two different NFTs, one saves CRS and the other saves ZKP keys and circuits. These NFTs simplify the various information that must be shared between provers and verifiers into just one ZKP NFT. To deal with the blockchain storage cost and computation cost, BCZKP introduces offline calculation module and public storage.

Future research will concentrate on expanding BCZKP applications with various smart contracts such as cryptocurrency exchange and digital data exchange. A client of these applications cannot create input with no limitation.

We will design a input validation protocol in BCZKP to solve the input consistency of the hidden inputs.

ACKNOWLEDGMENT

REFERENCES

[1] S. Nakamoto. “Bitcoin: a peer-to-peer electronic cash system.” 2009.

[2] https://steemit.com

[3] A. Zen. “CryptoKitties.” 2017.

[4] H. Teng et al. "Applications of the Decentralized Finance (DeFi) on the Ethereum." 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), pp. 573-578, 2022.

[5] Cash, Bitcoin. "Bitcoin cash." Development 2 (2019).

[6] S. Casale-Brunetet al. "Networks of Ethereum Non-Fungible Tokens: A graph-based analysis of the ERC-721 ecosystem." 2021 IEEE International Conference on Blockchain (Blockchain), pp. 188-195, 2021.

[7] Fiege, Uriel, Amos Fiat, and Adi Shamir. "Zero knowledge proofs of identity." Proceedings of the nineteenth annual ACM symposium on Theory of computing. 1987.

[8] Gabizon, Ariel, Zachary J. Williamson, and Oana Ciobotaru. "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge." Cryptology ePrint Archive (2019).

[9] Jens Groth. “ On the Size of Pairing-Based Non-interactive Arguments.” EUROCRYPT 2016. Lecture Notes in Computer Science, Vol. 9666. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-49896-5_11

[10] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza, “Secure sampling of public parameters for succinct zero knowledge proofs,” In IEEE Symposium on Security and Privacy, 2015.

[11] Sean Bowe, Ariel Gabizon, and Ian Miers, “Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model,” Cryptology ePrint Archive, Paper 2017/1050. Retrieved from https://eprint.iacr.org/2017/1050

[12] J. Benet, “IPFS - Content addressed, versioned, p2p file system (draft 3),” https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf, 2014.

[13] Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum project yellow paper 151.2014 (2014): 1-32.

[14] M. Cebe, E. Erdin, K. Akkaya, H. Aksu, and S. Uluagac, ‘‘Block4Forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles,’’ IEEE Commun. Mag., vol. 56, no. 10, pp. 50–57, Oct. 2018.