IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 2 December 2022, accepted 31 December 2022, date of publication 3 January 2023, date of current version 6 January 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3233828

METHODS

Privacy-Preserving Identity Management System on Blockchain Using Zk-SNARK

DUC ANH LUONG AND JONG HWAN PARK
Department of Computer Science, Sangmyung University, Seoul 03016, South Korea

Corresponding author: Jong Hwan Park (jhpark@smu.ac.kr)

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) funded by the Korean Government (MSIT) through the Blockchain Privacy Preserving Techniques Based on Data Encryption under Grant 2021-0-00518.

ABSTRACT Privacy plays a crucial role in the internet era, where many applications allow people to communicate and use their services through the internet. Privacy-preserving Identity Management (PPIdM) system is a scheme that helps manage users' identities and protects users' privacy by enabling users to authenticate themselves without disclosing their real identities. The PPIdM system also allows users to reveal some minor identity attributes while others remain secret selectively. However, anonymity also encourages malicious users to break the system's policy and commit crimes since their real identities are anonymous. Existing PPIdM systems use the identity provider (IP) as a medium to verify users' identity attributes, record all users' real identities, and ensure that malicious users' identities are traceable. Therefore, users' identities are hidden from all entities but the IP. However, the user's privacy is vulnerable because there is nothing to guarantee that the IP is always honest and not curious about their users' activities and private information. This paper proposes a PPIdM system on the blockchain that helps users manage their identity attributes and keeps their real identities secret from all entities, including the IP. Still, the system's consensus can trace malicious users' real identities if they violate the system's policy. The PPIdM's security requirements are analyzed and proved informally using the game-based proof scheme. The main idea of this study is to combine zk-SNARK, a type of zero-knowledge proof (ZKP), Shamir's secret sharing (SSS), and several other cryptographic techniques.

INDEX TERMS Identity management, decentralized identifier, blockchain, anonymity, authentication, secret-sharing, zk-SNARK.

I. INTRODUCTION

In the internet era, using online services from a service provider (SP) or communicating with other users in an online community sometimes requires users to have specific IAs, like age, gender, and additional information. In these cases, users must prove that their IAs are valid, and an IdM system [1], [2] is necessary. The IdM system is a system that helps users manage their IAs. There are two types of IAs: primary identity attributes (PIAs) and minor identity attributes (MIAs). A PIA is a unique attribute that can identify a user. Driver licenses, citizen identity, and passport numbers are typical types of PIA. On the other hand, MIAs are common attributes that do not specify a user. They are date of birth, gender, workplace, or other additional information.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Huang.

Using the IdM system, a user must register and ask a trusted IP to verify his IAs. After verifying the IAs, the IP will issue a certificate indicating the user's IAs are valid. Afterward, the user can manage his verified IAs and use his certificate to prove the validity of the IAs when using the SP's services.

Because each user is identified by a PIA and the PIA must be shown in the certificate, the SP can track all their user's activities. Hence, this type of system lacks privacy. A PPIdM system does not include the user's PIA in the certificate. Instead, the IP will issue each user a pseudonym (PS) and have this PS in the user's certificate. Because a PS does not reveal any information about the user's PIA and only the IP and the PS's owner know to whom the PS belongs, the SP and other entities cannot track users' activities and service history. Fig. 1 describes this system.

Nevertheless, if the IP is malicious, it can collude with the SP to track all the user's activities and service history.

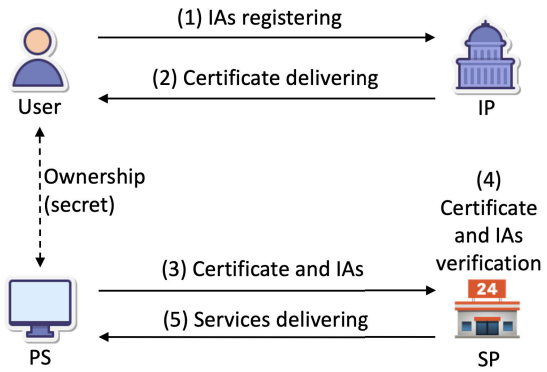

FIGURE 1. PPIdM system. The ownership is the relation between the PIA and the PS. Only the user and the IP know this relation.

Therefore, this system is only ideal with an assumption: the IP is always honest and is not curious about the user's private information and activities.

A solution to the above issue is to keep the PS away from the IP. In other words, the certificate still comprises the PS instead of the PIA, but the IP does not know who the PS owner is. Therefore, there must be a method for the IP to verify a user's IAs and issue the user a certificate without knowing the user's PS. Anonymous authentication using zk-SNARK [3] is a good solution to this problem. However, if only the user knows the PS, this PPIdM system will lack traceability. It will encourage malicious users to commit crimes because there is no way for the IP to trace their PIAs. In addition, the zk-SNARK scheme is vulnerable to collusion attacks, in which multiple users can simultaneously use a certificate as the zk-SNARK's witness. Collusion attacks enable unregistered users to join the system by reusing other users' certificates. Therefore, an ideal PPIdM system should provide anonymity for honest users and traceability for malicious users. It should also prevent a user from sharing his certificate with other users. With this idea, we introduce a novel PPIdM system with the following contributions.

- We employ the public blockchain, the ElGamal encryption, the hash-based commitment scheme, and the zk-SNARK scheme to provide IAs-selective disclosure, anonymity, and unforgeability and counter the collusion attacks. We use the Shamir Secret Sharing (SSS) scheme [4] and the blockchain's consensus to provide traceability.
- We analyze the security of the proposed system based on essential requirements for a PPIdM system.
- We calculate the time complexity and the gas cost of the proposed system.

II. RELATED WORK

To date, several blockchain-based IdM systems [5], [6], [7], [8], [9], [10], [11], [12], [13] have focused on protecting users' privacy and providing anonymity. In this section, we review the articles related to our study. Table 1 summarizes the comparison between the proposed system and others.

ElGayyar et al. [5] introduced a robust automatic blockchain-based federated IdM. Smart contracts automatically generate identities and audits for users. Users can control their identities and store them on the blockchain. However, this system's anonymity is entirely based on the anonymity of the blockchain. In addition, because this system employs smart contracts to encrypt and decrypt users' data or identities, users' identities may be visible to the public because of the transparent property of the public blockchain.

The system of Gao et al. [6] also allows users to hide their real identities from the public. First, a user authenticates himself by bringing a smartphone with a biometric authentication mechanism and his real identity to the IP. The IP generates a pair of asymmetric keys and embeds the secret key in the smartphone. It issues a certificate comprising the user's information and the public key to the user. Afterward, the user uploads this certificate to the blockchain network using his PS. Any transactions from the user must be signed by the secret key embedded in the smartphone, which is unlocked only by the user's biometric. This system is similar to ElGayyar et al. [5]. Users' identities are stored on the blockchain and visible to other users or SPs. The system's anonymity is based on the anonymous addresses of the blockchain.

Xu et al. [7] have the same approach as Gao et al. [6] for building an IdM system for mobile devices. A user must first provide his identity and public key to the IP. After the IP authenticates his identity and gives him a verifiable claim, he sends his authenticated identity and the verifiable claim to the network operator. The network operator will verify this claim and upload his identity and public key to the blockchain network. Afterward, all the blockchain network members can verify this user by querying his public key on the blockchain. Unlike Gao et al. [6], their scheme employs the consortium blockchain instead of a public blockchain.

Besides the above system, schemes [8] and [9] have the same overall architecture. However, these systems have a low privacy level as users' identities are visible to everyone because they are stored in the blockchain network. The following systems can solve these problems.

In the system of Kassem et al. [10], storing the user's IAs in a permissioned ledger can prevent unauthorized SPs from accessing the user's IAs without a user's consent. Each user has an Ethereum account that maps to their IAs, verified by specified smart contracts. The SPs must gain the user's consent to access his permissioned ledger to verify his IAs. In the system of Faber et al. [11], each user has a personal off-chain database to store their identities instead of keeping them in the blockchain. The hashes of these data are uploaded to the blockchain as the addresses of their IAs. Service providers must have the user's consent to access a user's personal database to verify the user's IAs. However, users must use the same account for consuming on-chain services and receiving IAs and data from IPs.

Nevertheless, the above researches have a common problem that decreases privacy. Although the users' real identities are hidden from the public, they must be visible to the IPs

to trace the real identities of malicious users. In other words, the privacy of these systems strongly relies on the IPs. There is nothing to guarantee that the IPs will not collude with the service providers to link their real identities with the corresponding private data from the service providers. Similarly, the scheme [12] also has this problem.

To eliminate this disadvantage, Zhuang et al. [13] introduced a blockchain-based PPIdM system that enables users to hide their IAs from both the public and the IP but still allows the IP to trace their real identities if the users violate the system's policy based on the blockchain's consensus and SSS. This system works similarly to the scheme of Xu et al. [7] to achieve this goal, except that it divides the user's PIA into k shares using SSS instead of directly storing it in the blockchain network. A valid user can use his user identity (UID) to communicate with service providers. The service providers can verify if the user is valid by querying his UID on the blockchain. Notably, this UID does not disclose information about the user's PIA. If the user violates the system policy, the system can reconstruct his PIA based on k shares using SSS. However, this system strongly assumes that the IPs must delete the user's PIA and the associated UID. Otherwise, SSS is meaningless because the IPs can always map the UID to the corresponding PIA.

Based on the above literature, we can see that these systems' privacy is low or moderate. In addition, none of them support selective disclosure. Therefore, we addressed their existing problems by designing a privacy-preserved IdM system on the blockchain that enables users to hide their IAs from the public and the IP. The proposed system ensures that no single party can trace a user's PIA. When malicious users violate the system policy, it requires the agreement of a pre-defined number of parties to open the user's PIA.

III. PRELIMINARIES

A. SHAMIR SECRET SHARING

Shamir Secret Sharing (SSS) [4] is an algorithm for protecting a secret in a distributed way based on polynomial interpolation over finite fields. Given a secret s_0, SSS is a perfect (k, n)-threshold scheme that satisfies two following properties.

- The secret s_0 can be reconstructed using at least k shares with probability 1.
- The secret s_0 is information-theoretically hidden when trying to reconstruct the share using at most $k-1$ shares.

Let $\mathsf{G} = (g, q)$ be a group of order q, where g is a generator of G. The (k, n)-threshold SSS in a group G can be presented with two following functions.

- $\mathsf{SharesGen}(\mathsf{G}, \mathbb{P}_{k-1}, s_0) \rightarrow \mathbb{S}_n$, where $\mathbb{P}_{k-1}$ is a set of $k-1$ numbers from p_1 to p_{k-1} randomly chosen in $\mathbb{Z}_q$ and $s_0 \leftarrow g^w$ is the secret for a random $w \in \mathbb{Z}_q$. Then, $\mathbb{S}_n$ is a set of n points $s_i = (x_i, y_i)$ for $i = 1, \ldots, n$, which are generated as follows.

$$\begin{pmatrix} s_0 \leftarrow g^w, \\ g^{f(x)} := s_0 \prod_{i=1}^{k-1} g^{p_i x^i}, \\ s_1 = (1, g^{f(1)}), \\ s_2 = (2, g^{f(2)}), \\ \vdots \\ s_n = (n, g^{f(n)}), \\ \mathbb{S}_n = \{s_1, s_2, \ldots, s_n\}. \end{pmatrix}$$

- $\mathsf{Reconstruct}(\mathbb{S}_k) \rightarrow s_0$, where $\mathbb{S}_k$ is a set of arbitrary k shares $s_i = (x_i, y_i)$ for s_1 to s_k. The secret s_0 is reconstructed as follows.

$$s_0 = \prod_{j=1}^{k} y_j^{\Delta_j}, \quad \text{where} \quad \Delta_j = \prod_{m=1, m\neq j}^{k} \frac{x_m}{x_m - x_j} \in \mathbb{Z}_q.$$

B. ZK-SNARK

Our proposed system employs the zk-SNARK scheme of Parno et al. [14]. We briefly recall it as follows.

- $\mathsf{KeyGen}(C) \rightarrow \{PK, VK\}$: this function takes an arithmetic circuit C and creates a pair of a proving key PK, and a verification key VK.
- $\mathsf{Proof}(\vec{w}, \vec{x}, PK) \rightarrow \pi$: this function creates a proof string π (that consists of eight elliptic curve points) by taking as input a public input $\vec{x}$, a witness $\vec{w}$, and the proving key PK.
- $\mathsf{Verify}(\pi, \vec{x}, VK) \rightarrow b$: this function takes the proof string π, a public input $\vec{x}$, and the verification key VK to verify that π is valid or not, and outputs a decision bit b.

The zk-SNARK scheme of Parno et al. also satisfies the following properties.

- **Completeness:** If the prover generates a correct proof string π using a witness $\vec{w}$, the verifier always accepts π.
- **Succinctness:** The size of the proof π is short, regardless of the size of the witness $\vec{w}$ and the public input $\vec{x}$, and also π is efficiently verified.
- **Zero-knowledge:** There exists an efficient simulator that takes a proving key PK and a public input $\vec{x}$, and outputs a simulated proof that is indistinguishable from a real proof (generated with a witness $\vec{w}$).
- **Soundness:** Given a proving key PK, a verification key VK, and a public input $\vec{x}$ (but not the associated witness $\vec{w}$), it is infeasible to generate a correct proof string to be successfully verified.

C. DIGITAL SIGNATURE

We recall the Edward-curve digital signature algorithm (EdDSA) [15]. EdDSA is a variant of the Schnorr signature based on performance-optimized twisted Edwards curves. Let $\mathsf{G} = (g, q)$ be a group (over an Edwards curve) of order q, where g is a generator of G. Then, EdDSA consists of the following three algorithms.

TABLE 1. System comparison.

System	Privacy	Anonymous authentication	PIA-opening techniques	IAs storage	Selective disclosure
ElGayyar *et al.* [5]	Weak	No	Centralized IP	Blockchain (ciphertext)	No
Gao *et al.* [6]	Weak	No	Centralized IP	Blockchain (plaintext)	No
Xu *et al.* [7]	Weak	No	Centralized IP	Blockchain (plaintext)	No
Ren *et al.* [9]	Weak	No	Centralized IP	Blockchain (plaintext)	No
Mell *et al.* [8]	Weak	No	Centralized IP	Blockchain (plaintext)	No
Kassem *et al.* [10]	Moderate	No	Centralized IP	Permissioned ledger	No
Faber *et al.* [11]	Moderate	No	Centralized IP	Private storage	No
Westerkam *et al.* [12]	Moderate	No	Centralized IP	Private storage	No
Zhuang *et al.* [13]	Moderate	No	Consensus & SSS	Blockchain (ciphertext)	No
The proposed system	High	zk-SNARK	Consensus & SSS	Blockchain (Commitment)	Yes

- $\mathsf{KeyGen}_{\mathsf{sig}}(\lambda, \mathsf{G}) \rightarrow \{SK_{sig}, PK_{sig}\}$, where λ is a security parameter, $SK_{sig} \in \mathbb{Z}_q$ is the secret key, and $PK_{sig} \leftarrow g^{SK_{sig}} \in \mathsf{G}$ is the public key.
- $\mathsf{Sign}_{\mathsf{sig}}(m, SK_{sig}) \rightarrow \{R, s\}$, where m is a message to be signed. Given a hash function $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$, $\{R, s\}$ is a signature that is calculated as follows.

$$\begin{pmatrix} r \leftarrow \mathsf{H}(\mathsf{H}(SK_{sig}), m) \pmod q, \\ R \leftarrow g^r, \\ h \leftarrow \mathsf{H}(R, PK_{sig}, m) \pmod q, \\ s \leftarrow r + PK_{sig} \cdot h \pmod q. \end{pmatrix}$$

- $\mathsf{Verify}_{\mathsf{sig}}(m, PK_{sig}, \{R, s\}) \rightarrow b$, where b is the decision bit and is decided as follows.

$$\begin{pmatrix} h \leftarrow \mathsf{H}(R, PK_{sig}, m) \pmod q, \\ P_1 \leftarrow g^s \\ P_2 \leftarrow R \cdot PK_{sig}^h \end{pmatrix}$$

If $P_1 = P_2$ then $b = 1$ (valid), else $b = 0$ (invalid).

Since EdDSA is a variant of the Schnorr signature, it is easily shown (and well-known) that EdDSA is also unforgeable against chosen-message attacks based on the hardness of a discrete logarithm problem in G, assuming that H is modeled as a random oracle.

D. ElGamal ENCRYPTION

We briefly describe the ElGamal encryption scheme as follows. Let $\mathsf{G} = (g, q)$ be a group (over an Edwards curve) of order q, where g is a generator of G. The ElGamal encryption consists of the following algorithms.

- $\mathsf{KeyGen}_{\mathsf{ElGamal}}(\lambda) \rightarrow \{PK, SK\}$, where λ is a security parameter. The secret key SK is randomly chosen in $\mathbb{Z}_q$, and $PK \leftarrow g^{SK} \in \mathsf{G}$ is the corresponding public key.
- $\mathsf{ENC}(m, PK; r) \rightarrow c$, where $m \in \mathsf{G}$ is a message and r is a random number in $\mathbb{Z}_q$. The ciphertext $c = \{c_1, c_2\}$ is generated as follows:

$$\begin{pmatrix} c_1 \leftarrow g^r \in \mathsf{G}, \\ c_2 \leftarrow m \cdot PK^r \in \mathsf{G}. \end{pmatrix}$$

- $\mathsf{DEC}(SK, c) \rightarrow m$. The decryption process is as follows.

$$\begin{pmatrix} s \leftarrow c_1^{SK} \in \mathsf{G}, \\ m \leftarrow c_2 \cdot s^{-1} \in \mathsf{G}. \end{pmatrix}$$

The ElGamal encryption is known to be correct and secure against chosen-plaintext attacks, assuming the decisional Diffie-Hellman problem holds in G.

E. HASH FUNCTION

A (cryptographic) hash function $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$ has the following properties.

- **Onewayness**. It is computationally infeasible to compute a preimage from a hash value given.
- **Collision-resistance**. Finding two distinct preimages mapped into the same hash value is computationally infeasible.

F. COMMITMENT

A commitment scheme allows a committer to commit a value to keep it hidden from others and reveal the committed value later. A commitment scheme for our system is a simple and well-known commitment scheme based on a hash function.

Suppose that a committer A wants to commit a value x to B. The hash-based commitment scheme can be represented as follows.

- A sends $C \leftarrow \mathsf{H}(x, q)$ to B, where C is the commitment, x is the committed value, and q is chosen randomly from $\{0,1\}^l$ for some integer l.
- A reveals his commitment by sending $\{x, q\}$ to B. B then verifies A's commitment by checking whether $C = \mathsf{H}(x, q)$.

The above hash-based commitment scheme satisfies the following two properties.

- **Hiding**. The commitment C should give no information about the committed value x.
- **Binding**. The committer A cannot change the committed value x once the commitment C is sent to B.

G. BLOCKCHAIN

Blockchain is a decentralized data structure implemented in a peer-to-peer network. Instead of storing data in one

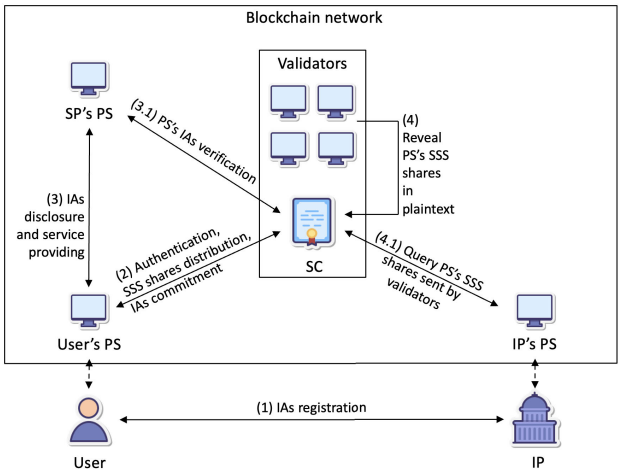

FIGURE 2. System overview and the relations of all entities. (1) The user registers his IAs to the IP. The IP issues the user a certificate. (2) The user uses his PS and the certificate to anonymously authenticate himself, distribute his encrypted SSS shares to validators, and commit his IAs to the SC. (3) The user uses his PS to send his IAs to the SP and receive services from the SP. (3.1) The SP verifies the PS's IAs by querying them in the blockchain. (4) The validators reveal the PS's SSS shares in plaintext. (4.1) The IP queries for the PS's SSS shares and uses them to reconstruct the PS's PIA.

centralized database, the system keeps the replicated data in numerous peers (nodes). Each time new data are added, all nodes in the system update their replicated data. In the blockchain, data are sent and received in transactions.

Blockchain has the following three main properties.

- **Immutability:** Once transactions are added, no single entity can alter these transactions.
- **Decentralized:** Data are stored in numerous nodes instead of a single node.
- **Consensus:** Transactions must be verified by more than half the number of nodes to be added to the blockchain.

Basically, there are three types of blockchains as follows.

- **Public blockchain:** users are free to join and quit this system. Transactions in this system are truly transparent and visible to all nodes.
- **Private blockchain:** this system is managed by an entity. Users must have the permission of the manager to participate into. Transactions in this system are visible only to its members.
- **Hybrid blockchain:** this system combines public and private blockchain. Instead of one entity managing the system, all members manage the system and decide who can participate in the blockchain. They also determine which transactions are public or private.

There are two types of consensus algorithms, Proof of Work (PoW) [16] and Proof of Stake (PoS) [17]. PoW allows all nodes in the blockchain to generate and verify transactions. It reduces the risk of blocks generated by malicious nodes by asking block miners to solve a hash puzzle [18]. On the other hand, PoS only allows some specific nodes (validators) to create and verify transactions. PoS reduces the risk of malicious validators by asking them to delegate an amount of money as a stake before joining the validating process. Malicious validators may lose their stake if they are detected based on the system policy.

The consensus algorithm is crucial in the blockchain. It decides which block will be added to the main chain among several blocks. A bad consensus algorithm will result in an insecure blockchain that is vulnerable to 51% attacks [19] and censorship attacks [20]. The typical consensus algorithm in most blockchain networks is Byzantine Fault Tolerance (BFT) [21]. To date, there have been many other BFT versions that improve and optimize the BFT's performance [22].

One of the most-used applications of the blockchain is smart contracts [23]. A smart contract (SC) is a program stored in the blockchain that runs when the predetermined conditions are met. Because the SC's programming language is Turing-complete in most blockchain networks, the SC can be used to implement various logical operations and applications. The proposed system uses an SC as the zk-SNARK's verifier and controls all the system's processes.

IV. PROPOSED SYSTEM

A. MAIN ENTITIES

There are six main entities in the proposed system: an identity provider (IP), validators, users, pseudonyms (PSs), an SC, and service providers (SPs). Fig 2 briefly shows the relations of these entities in the proposed system. We describe these entities as follows.

IP. This entity can be a centralized or decentralized organization that issues user certificates. The IP is responsible for tracing the malicious users' PIAs.

SPs. These entities provide services for users. They can require users to have some specific IAs to use their services.

Validators. These entities are responsible for collecting, verifying, and wrapping transactions into a block and adding blocks to the blockchain. They are also responsible for keeping users' SSS shares and the PIA-opening process.

Users. The IP validates these entities and their IAs. Users use their PSs to manage their IAs, communicate anonymously in the proposed system, and use the services provided by the SPs.

PSs. They are public blockchain accounts (addresses) of users, validators, SPs, and the IP. These entities can use their PSs to sign and send transactions.

SC. The SC in the proposed system acts as a zk-SNARK verifier and helps store and query users' committed IAs.

There are two communication channels in the proposed system: the on-chain channel and the off-chain channel. We describe them as follows.

On-chain channel. This is the public blockchain network, where all messages are sent and received under the transaction form. In the proposed system, the public blockchain uses the PoS consensus algorithm. Validators collect transactions, verify them and add them to the blockchain. Communication on this channel is transparent and secure against several types of attacks.

Off-chain channel. This channel refers to all channels unrelated to the public blockchain network. In our

TABLE 2. Notations in the proposed system.

Notations	Descriptions
U_i	The i^{th} user
PS_i	The PS of U_i
v_i	The i^{th} validator
$cert_i$	U_i's certificate issued by the IP
x_i	The i^{th} attribute (PIA or MIA)
y_i	The commitment of x_i
s_i	An SSS share for v_i
c_i	The encrypted s_i
$\mathbb{X}_t$	A set of t attributes: $(x_1, x_2, ..., x_t)$
$\mathbb{Y}_t$	A set of t commitments: $(y_1, y_2, ..., y_t)$
$\mathbb{V}_n$	A set of n arbitrary validators
$\mathbb{S}_n$	A set of n SSS shares: $(s_1, s_2, ..., s_n)$
$\mathbb{C}_n$	A set of n encrypted SSS shares: $(c_1, c_2, ..., c_n)$
SK_{IP}, PK_{IP}	The IP's secret key and public key
SK_{v_i}, PK_{v_i}	The secret key and public key of v_i
$SK_{\mathbb{V}_n}, PK_{\mathbb{V}_n}$	The set of n arbitrary SK_{v_i} and PK_{v_i}
PK_C, VK_C	The zk-SNARK proving key and verification key of circuit C
PK_D, VK_D	The zk-SNARK proving key and verification key of circuit D
$\vec{w}_C, \vec{x}_C, \pi_C$	The set of private inputs, public inputs and output, and the proof string of circuit C, respectively
$\vec{w}_D, \vec{x}_D, \pi_D$	The set of private inputs, public inputs and output, and the proof string of circuit D, respectively

construction, this off-chain channel includes communications with which users initially register them to the IP or users later have access to the SP.

B. ASSUMPTIONS

We assume the following statements to ensure that this paper is within our scope and focuses on our contributions.

- The off-chain channel is secure to the extent that the channel achieves its goal. For instance, attackers cannot gain information about users from communications between users and the IP.
- The public keys PK_{IP} (for the IP), $PK_{\mathbb{V}_n}$ (for a set of validators $\mathbb{V}_n$), and the zk-SNARK's proving key PK are public and shared in advance by all entities in the system.
- All entities use the same elliptic curve for zk-SNARK and the same group $\mathsf{G} = (g, q)$ based on a certain Edwards curve for the SSS, the ElGamal encryption, and the EdDSA signature schemes. These parameters are shared in advance.
- When working with the (k, n)-threshold SSS, it is assumed that at least k number of validators are honest and $k > \frac{n}{2}$.
- The SPs are audited entities. Their public keys and addresses on the blockchain are verified.

C. CIRCUITS AND SMART CONTRACT

This section describes arithmetic circuits, the SC, and the proposed system. We present all notations used in the proposed system in Table 2 for ease of reading.

1) ARITHMETIC CIRCUIT

The arithmetic circuit is the core of zk-SNARK. It describes the relationship between the prover's witnesses $\vec{w}$ and the verifier's pre-defined constraint $\vec{x}$ (as the arithmetic circuit's public inputs). We design two arithmetic circuits C and D. Circuit C is responsible for anonymous authentication, SSS distribution, and selective disclosure processes. The previous work [24] designed an anonymous authentication technique that supports a fixed number of users in their arithmetic circuit. In our proposed system, however, we enhance their technique by using EdDSA to support an unlimited number of users in a single arithmetic circuit. The details for the circuit C are described in Algorithm 1. Algorithm 2 represents circuit D for verifying the correctness of the ElGamal encryption, which is necessary for the PIA-opening process.

In an arithmetic circuit, the private inputs correspond to zk-SNARK's witnesses $\vec{w}$. They are hidden, and the proof string π exposes no information about them. The set of public inputs $\vec{x}$ is visible to everyone. The verifier uses $\vec{x}$ and π to verify the relations between $\vec{w}$ and $\vec{x}$ declared in the arithmetic circuit.

Algorithm 1 Arithmetic Circuit C

Private inputs: $m_i, r, \mathbb{P}_{k-1}, \mathbb{Q}_t, \mathbb{X}_t, cert_i$
Public inputs: $\hat{h}_i, PK_{IP}, \mathbb{C}_n, PK_{\mathbb{V}_n}, \mathbb{Y}_t$
Output: Decision bit b_C

1. Calculate $h_i \leftarrow \mathsf{H}(m_i)$
2. Compute $root_i \leftarrow \mathsf{H}(\mathbb{X}_t, h_i)$
3. Check $\mathsf{Verify}_{\mathsf{sig}}(root_i, cert_i, PK_{IP})$
4. Check if $\hat{h}_i = \mathsf{H}(m_i + 1)$
5. Calculate $e_i \leftarrow g^{PIA}$
6. Compute $\mathbb{S}_n \leftarrow \mathsf{SharesGen}(\mathbb{P}_{k-1}, e_i)$
7. Check if $c_i = \mathsf{ENC}(s_i, PK_{v_i}; r)$ for $i = 1, \ldots, n$
8. Check if $y_i = \mathsf{H}(x_i, q_i)$ for $i = 1, \ldots, t$
9. If lines 3, 4, 7, and 8 are *true* then $b_C = 1$, else $b_C = 0$

a: CIRCUIT C

As in Algorithm 1, the circuit C has $\vec{x}_C = \{\hat{h}_i, PK_{IP}, \mathbb{C}_n, PK_{\mathbb{V}_n}, \mathbb{Y}_t, b_C\}$ and $\vec{w}_C = \{m_i, r, \mathbb{P}_{k-1}, \mathbb{Q}_t, \mathbb{X}_t, cert_i\}$, where m_i and r are two random numbers, $\mathbb{P}_{k-1}$ is a set of $k-1$ random numbers for the (k, n)-threshold SSS (line 6, Algorithm 1), and $\mathbb{Q}_t$ is another set of t random numbers for computing the commitment of $\mathbb{X}_t$ (line 8, Algorithm 1).

$$\mathbb{P}_{k-1} = \{p_1, p_2, \ldots, p_{k-1}\},$$
$$\mathbb{Q}_t = \{q_1, q_2, \ldots, q_t\}.$$

PS_i, as a prover, needs to convince the verifier that the following statements are true.

- *PS_i is the owner of a certificate $cert_i$ provided by the IP.* This statement is proven without revealing the identity of U_i. Line 1 computes h_i from m_i that U_i has selected. Because only U_i knows m_i, this line allows U_i to prove the ownership of h_i, and thus $\mathbb{X}_t$. Line 2 computes the hash value of $\{\mathbb{X}_t, h_i\}$. We call this hash value $root_i$.

Algorithm 2 Arithmetic Circuit D

Private inputs: SK_{v_i}
Public inputs: c_i, s_i, PK_{v_i}
Output: Decision bit b_D
1. Check if $PK_{v_i} = g^{SK_{v_i}}$
2. Check if $s_i = \mathsf{DEC}(SK_{v_i}, c_i)$
3. If lines 1 and 2 are *true* then $b_D = 1$, else $b_D = 0$

Because $root_i$ is a hash value, it is fixed-sized, and any changes in $\mathbb{X}_t$ or h_i will result in a different $root_i$. Line 3 verifies whether $cert_i$ is the EdDSA signature of $root_i$. PS_i needs to pass this verification to prove that he is the owner of $cert_i$.

- *m_i has never been used before.* This statement aims to enhance the above anonymous authentication by removing the threat of collusion attacks where U_i shares m_i with other users. Each π_C will be identified by $\hat{h}_i \leftarrow \mathsf{H}(m_i + 1)$. The verifier records the list of $\hat{h}_i$. Any π_C with a pre-used $\hat{h}_i$ will be rejected. Line 4 implements this statement.
- *U_i correctly generates $\mathbb{S}_n$ from his PIA and then encrypts $\mathbb{S}_n$ using $PK_{\mathbb{V}_n}$.* This statement allows U_i to convince the verifier that he distributes the correct SSS shares in ciphertext generated from his PIA. These ciphertexts can be decrypted using $SK_{\mathbb{V}_n}$. Because the ElGamal encryption works with an Edwards curve, line 5 maps the PIA to an elliptic curve point e_i. Line 6 uses SharesGen to generate $\mathbb{S}_n$ from $\mathbb{P}_{k-1}$ and e_i. Line 7 encrypts s_i using the ElGamal encryption algorithm, ENC, and shows that the result is identical to c_i, which will be issued to a validator v_i for $i = 1, \ldots, n$.
- *The prover commits correct IAs which the IP verified.* PS_i must convince the verifier that $\mathbb{Y}_t$ is the set of correct commitments generated from $\mathbb{X}_t$. Later, the public $\mathbb{Y}_t$ values are used to selectively disclose some of IAs related to U_i. Line 8 computes the commitment of $\{x_i, q_i\}$ and compares it with y_i for $i = 1, \ldots, t$, which will be stored on the blockchain.

b: CIRCUIT D

Because reconstructing PIA requires $\mathbb{S}_k$ from $\mathbb{V}_k$ using the (k, n)-SSS, there must be a way to prove that v_i honestly sent its share s_i without disclosing SK_{v_i}. We design circuit D to verify that validators honestly submit their SSS shares.

As in Algorithm 2, circuit D requires v_i to enter SK_{v_i} as the private input ($\vec{w}_D = SK_{v_i}$), and $\{c_i, s_i, PK_{v_i}\}$ as public inputs ($\vec{x}_D = \{c_i, s_i, PK_{v_i}, b_D\}$). The circuit D describes the following statement: v_i owns SK_{v_i} corresponds to PK_{v_i} (line 1), and s_i is derived by decrypting c_i using SK_{v_i} (line 2). If the above statements are *true* then $b_D = 1$, else $b_D = 0$.

2) SMART CONTRACT

We design an SC to implement zk-SNARK's function Verify of circuits C and D and other functions for the proposed system. Algorithm 3 describes the details of the SC.

Algorithm 3 *SC*

Define: $Table_1, Table_2, Table_3$
1: **Function** authentication($\pi_C, \vec{x}_C$):
2: **Require** $b_C = 1$.
3: **Require** $PK_{\mathbb{V}_n}$ to be correct.
4: **Require** Verify($\pi_C, \vec{x}_C, VK_C$) to return 1.
5: **Require** $\hat{h}_i$ not to be in $Table_1$.
6: Add $\{PS_i, \hat{h}_i, \mathbb{C}_n\}$ to $Table_1$.
7: Add $\{PS_i, \mathbb{Y}_t\}$ to $Table_2$.
8: **Function** queryPS(PS_i):
9: **If** PS_i is in $Table_1$: **return** *true*.
10: **Else: return** *false*.
11: **Function** openPIA($\pi_D, \vec{x}_D$):
12: **Require** $b_D = 1$.
13: **Require** Verify($\pi_D, \vec{x}_D, VK_D$) to return 1.
14: **Require** c_i to be in $Table_1$.
15: Add s_i to $Table_3$.
16: **Function** queryPIA(PS_i):
17: **Require** PS_i to be in $Table_3$.
18: **Return** $\mathbb{S}_k$.
19: **Function** queryMIA(PS_i, y_i):
20: **Require** PS_i to be in $Table_2$.
21: **If:** PS_i has y_i: **return** *true*.
22: **Else: return** *false*.

$Table_2$				
PS	$\{y_t\}$			
	y_1	y_2	...	y_t
...	...	...	...	...

FIGURE 3. Structure of the SC's $Table_2$.

As shown in Algorithm 3, the SC has three tables described in Fig. 3 and Fig. 8. The SC's functions are as follows.

- **authentication**($\pi_C, \vec{x}_C$): this function takes as inputs π_C and $\vec{x}_C$, where $\vec{x}_C = \{\hat{h}_i, \mathbb{C}_n, PK_{\mathbb{V}_n}, PK_{IP}, \mathbb{Y}_t, b_C\}$. It requires that (1) $b_C = 1$, (2) $PK_{\mathbb{V}_n}$ are correct, (3) the zk-SNARK's function Verify outputs 1, and (4) $\hat{h}_i$ has not been used before. If all these conditions hold, this function adds $\{PS_i, \hat{h}_i, \mathbb{C}_n\}$ to $Table_1$ and $\{PS_i, \mathbb{Y}_t\}$ to $Table_2$.
- **queryPS**(PS_i): this function takes as input a PS and checks if it is authenticated. If the queried PS is in $Table_1$, it returns *true* and vice versa.
- **openPIA**($\pi_D, \vec{x}_D$): v_i calls this function when opening the PIA of a target PS. This function takes as inputs π_D and $\vec{x}_D$, where $\vec{x}_D = \{c_i, s_i, PK_{v_i}, b_D\}$. First, it requires $b_D = 1$. Afterward, it checks if the zk-SNARK's verification Verify on π_D and $\vec{x}_D$ is valid. If the proof is correct and c_i is in $Table_1$, this function appends s_i to $Table_3$ as the associated share with the target PS.
- **queryPIA**(PS_i): this function takes as input a PS and returns the corresponding $\mathbb{S}_k$ if the PS is in $Table_3$. The

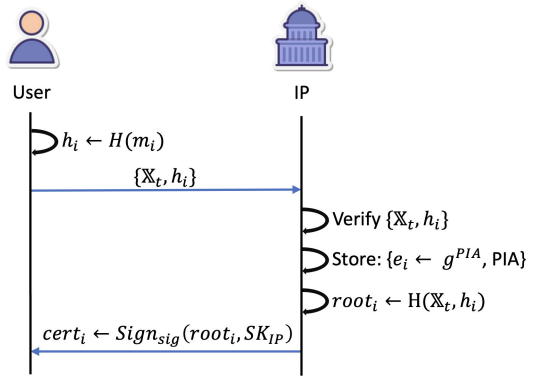

FIGURE 4. Sequence diagram of Step 1.

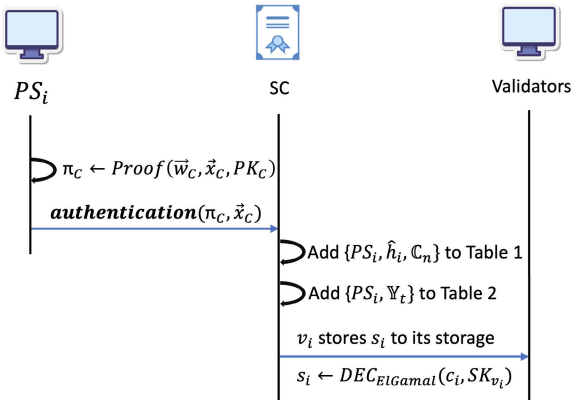

FIGURE 5. Sequence diagram of Step 2.

IP can call this function to reconstruct the PIA of the target PS.

- **queryMIA**(PS_i, y_i)**:** this function receives a PS and y_i. If the PS is in $Table_2$ and associated with y_i, it returns *true* and vice versa.

D. SYSTEM DESCRIPTION

Step 0. System initialization. The IP initializes the system as follows.

In the off-chain channel:

- Create the circuits C and D described in Algorithms 1 and 2.
- Generate zk-SNARK key pairs $\{PK_C, VK_C\}$ and $\{PK_D, VK_D\}$, where

$$\{PK_C, VK_C\} \leftarrow \text{KeyGen}(C),$$
$$\{PK_D, VK_D\} \leftarrow \text{KeyGen}(D).$$

- Distribute $\{PK_C, VK_C\}$ and $\{PK_D, VK_D\}$ to users.

In the on-chain channel:

- Construct and deploy the SC described in Algorithm 3 to the public blockchain.

Step 1. User registration. In this step, the IP verifies that U_i's IAs are valid, and the PIA identifies U_i. Afterward, the IP issues U_i a certificate $cert_i$. Fig. 4 describes the details of this step.

In the off-chain channel:

- U_i chooses a random number m_i and compute $h_i \leftarrow \text{H}(m_i)$.
- U_i prepares $\mathbb{X}_t = \{x_1, x_2, \ldots, x_t\}$, where x_1 is the PIA and $\{x_2, x_3, \ldots, x_t\}$ is the set of MIAs.
- U_i sends $\{\mathbb{X}_t, h_i\}$ to the IP.
- The IP verifies $\{\mathbb{X}_t, h_i\}$ to ensure that $\mathbb{X}_t$ is valid with respect to U_i and h_i has never been used before. Afterward, the IP computes $root_i$ and $cert_i$.

$$root_i \leftarrow \text{H}(\mathbb{X}_t, h_i),$$
$$cert_i \leftarrow \text{Sign}_{\text{sig}}(root_i, SK_{IP}).$$

- The IP computes $e_i \leftarrow g^{PIA} \in \text{G}$.
- The IP stores e_i and the PIA to its data storage.
- The IP sends $cert_i$ to U_i.

Step 2. Anonymous authentication, distribution of SSS shares, and IAs commitment. In this step, PS_i authenticates himself to the SC, distributes $\mathbb{C}_n$ to $\mathbb{V}_n$, and stores $\mathbb{Y}_t$ to the blockchain storage. Fig 5 describes the overview of this step.

In the off-chain channel:

- U_i chooses a random number r for the ElGamal encryption and two sets of random numbers $\mathbb{P}_{k-1} = \{p_1, p_2, \ldots, p_{k-1}\}$ for the (k, n)-threshold SSS and $\mathbb{Q}_t = \{q_1, q_2, \ldots, q_t\}$ for the commitment of $\mathbb{X}_t$.
- U_i computes $\hat{h}_i$, $\mathbb{S}_n$, $\mathbb{C}_n$, and $\mathbb{Y}_t$ such that:

$$\begin{aligned} \hat{h}_i &\leftarrow \text{H}(m_i + 1), \\ \mathbb{S}_n &\leftarrow \text{SharesGen}(\mathbb{P}_{k-1}, e_i), \\ \mathbb{C}_n &= \{c_1, \ldots, c_n\}, \\ \text{where} \quad c_i &= \text{ENC}(s_i, PK_{v_i}; r) \text{ for } i = 1, \ldots, n, \\ \mathbb{Y}_t &= \{y_1, \ldots, y_t\}, \\ \text{where} \quad y_i &= \text{H}(x_i, q_i) \text{ for } i = 1, \ldots, t. \end{aligned}$$

- U_i uses the circuit C and zk-SNARK's function Proof to generate π_C.

$$\pi_C \leftarrow \text{Proof}(\vec{w}_C, \vec{x}_C, PK_C),$$

where $\vec{w}_C = \{m_i, r, \mathbb{P}_{k-1}, \mathbb{Q}_t, \mathbb{X}_t, cert_i\}$ and $\vec{x}_C = \{\hat{h}_i, \mathbb{C}_n, PK_{\mathbb{V}_n}, PK_{IP}, \mathbb{Y}_t, b_C\}$.

In the on-chain channel:

- PS_i sends π_C and $\vec{x}_C$ to the SC by calling the SC's function **authentication**$(\pi_C, \vec{x}_C)$.
- After verifying π_C and $\vec{x}_C$, the SC adds $\{PS_i, \hat{h}_i, \mathbb{C}_n\}$ to $Table_1$ and $\{PS_i, \mathbb{Y}_t\}$ to $Table_2$. All the PSs in $Table_1$ are meant to be authorized users. They can selectively disclose their committed IAs stored in $Table_2$.
- v_i takes c_i from $\vec{x}_C$, and decrypts c_i to get s_i in the off-chain channel as follows.

$$s_i \leftarrow \text{DEC}(c_i, SK_{v_i}).$$

Step 3. Selective disclosure. If the SP requires an additional MIA from the PS_i, PS_i can disclose the MIA x_i from $\mathbb{X}_t$. Fig 6 describes the overview of this step.

- PS_i sends $\{x_i, q_i\}$ to the SP, where q_i is a random number in $\mathbb{Q}_t$ which is used to generate $\mathbb{Y}_t$.

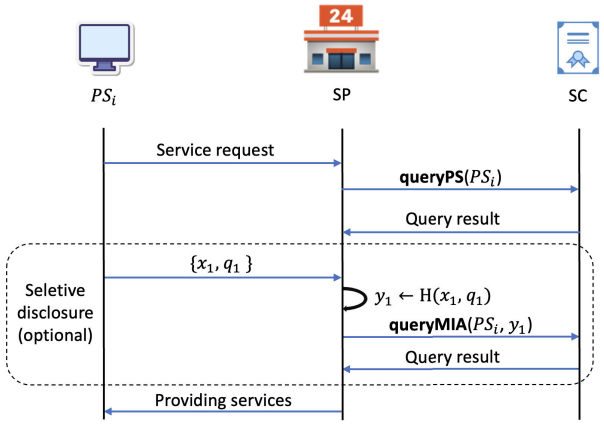

FIGURE 6. Sequence diagram of Step 3.

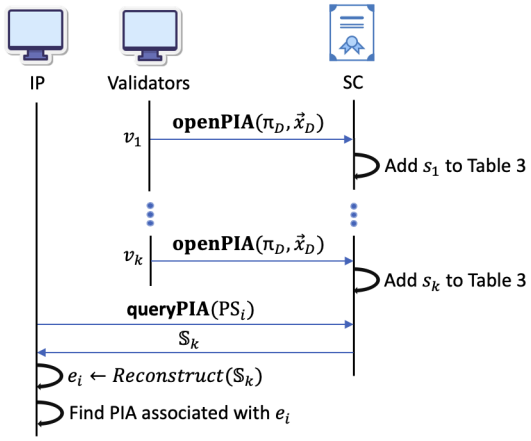

FIGURE 7. Sequence diagram of Step 4.

- The SP computes $y_i \leftarrow \mathsf{H}(x_i, q_i)$.
- In the on-chain channel, the SP calls the SC's function **queryAtt**(PS_i, y_i). If PS_i has the commitment y_i, the function will return *true*, and the SP can verify that PS_i has x_i.

Step 4. PIA-opening. In case PS_i is malicious and violates the system's policy, the IP can ask v_i to send s_i and reconstruct PS_i's e_i to find the PIA associated with PS_i. Fig. 7 describes the overview of this step, and particularly Fig. 8 describes the structure of the PIA-opening process.

In the on-chain channel:

- v_i sends $\pi_D \leftarrow \mathsf{Proof}(\vec{w}_D, \vec{x}_D, PK_D)$ and $\vec{x}_D$ to the SC by calling the function **openPIA**($\pi_D, \vec{x}_D$), where $\vec{x}_D = \{c_i, s_i, PK_{v_i}, b_D\}$ and $\vec{w}_D = SK_{v_i}$.
- The SC adds s_i to $Table_2$ if the zk-SNARK's function $\mathsf{Verify}(\pi_D, \vec{x}_D)$ returns *true*.
- After **openPIA()** is called k times by $\mathbb{V}_k$, the IP can call the function **queryPIA**(PS_i) to get $\mathbb{S}_k$.

In the off-chain channel:

- The IP reconstructs e_i using $\mathbb{S}_k$ and the SSS $\mathsf{Reconstruct}$ algorithm as

$$e_i \leftarrow \mathsf{Reconstruct}(\mathbb{S}_k).$$

- The IP finds the PIA in its data storage associated with e_i.

V. SECURITY

This section defines security requirements for PPIdM system, and proves that the proposed system satisfies them.

A. SECURITY REQUIREMENTS

Referring to security models of group signatures [25], we define security requirements for a PPIdM system as follows.

Unforgeability. Unforgeability captures that the system should prevent attacker $\mathcal{A}$ from (1) changing the registered IAs, (2) passing the authentication process by using incorrect witnesses, or (3) reusing another user's witnesses.

Anonymity. Recall that U_i's identity is associated with his m_i and PIA. Anonymity means that the proposed system should prevent $\mathcal{A}$ from finding information about m_i and the PIA associated with U_i.

Traceability. Traceability implies that $\mathcal{A}$ cannot (1) stop the PIA-opening process once triggered or (2) cheat the PIA-opening process to trace back another user instead of the target user.

B. SECURITY ANALYSIS

This section analyzes the security of the proposed system and proves that it satisfies the security requirements mentioned above.

1) UNFORGEABILITY

Theorem 1. *If the hash function* H *is collision-resistant, EdDSA is unforgeable against chosen-message attacks, zk-SNARK satisfies soundness property, the public blockchain is immutable, and the commitment scheme is binding, then the proposed system provides unforgeability.*

Proof. We use the game-based proof strategy to create a series of games and prove that the real game is indistinguishable from the final game, where the probability that adversary $\mathcal{A}$ succeeds in breaking unforgeability becomes negligible.

• Game_0 is the real game where $\mathcal{A}$ tries to join the system without registering his IAs to the IP or masquerading as other users.

• Game_1 is identical to Game_0, except that no collision on the hash function occurs. Because H is collision-resistant, Game_1 is indistinguishable from Game_0.

• Game_2 is the same as Game_1, except $\mathcal{A}$ can generate an EdDSA signature without SK_{IP}. Because EdDSA is unforgeable against chosen-message attacks, the probability of forging an EdDSA signature is negligible. This means $\mathcal{A}$ cannot use an invalid EdDSA signature to convince the SC that the IP has verified him. Therefore, Game_2 is indistinguishable from Game_1.

• Game_3 is identical to Game_2, except that $\mathcal{A}$ succeeds at generating π_C without satisfying lines 3, 4, 7, and 8 in Algorithm 1. Because zk-SNARK satisfies the soundness property, the probability of generating π_C from invalid witnesses

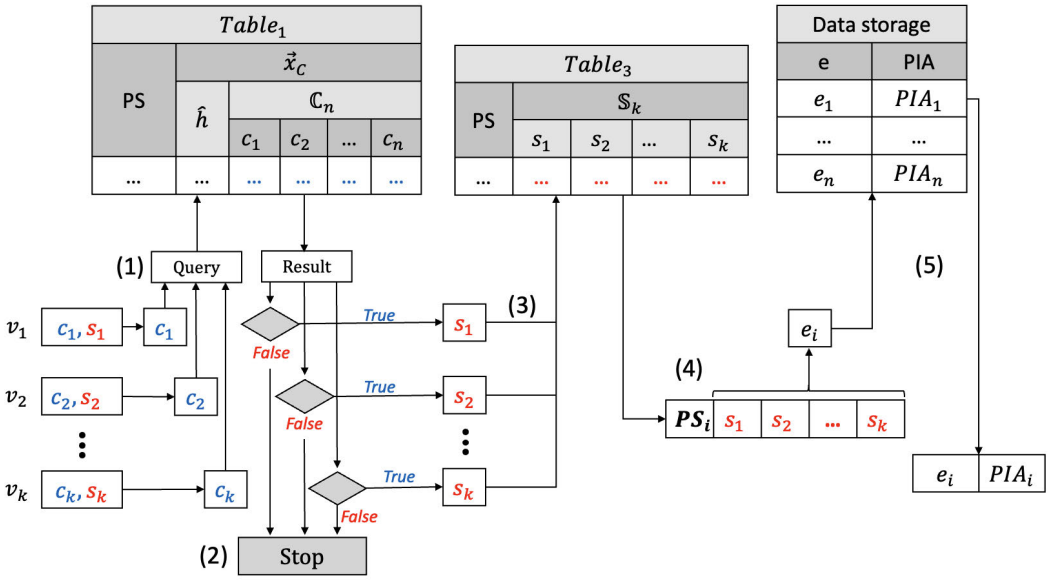

FIGURE 8. PIA-opening architecture. (1) After verifying π_D of v_i, the SC checks whether c_i is in Table 1, associated with the target PS. (2) If it is false, the process stops. (3) If it is true, s_i will be associated to the target PS in Table 3. (4) If the target PS is in Table 3 has $\mathbb{S}_k$, the IP will query for it and reconstruct e_i from $\mathbb{S}_k$. (5) Afterward, it finds the corresponding PIA in its data storage.

is negligible. Therefore Game_3 is indistinguishable from Game_2.

• Game_4 is identical to Game_3, except that $\mathcal{A}$ succeeds at modifying the blockchain data. Because the public blockchain is immutable, $\mathcal{A}$ cannot change his registered IAs stored in the blockchain. Therefore, Game_4 is indistinguishable from Game_3.

• Game_F is identical to Game_4, except that $\mathcal{A}$ succeeds at changing his committed value when verifying his commitment. Because the commitment scheme is binding, $\mathcal{A}$ cannot change his committed IAs when revealing them to the SP. Therefore, Game_F is indistinguishable from Game_4.

We can observe that, in Game_F, $\mathcal{A}$ cannot use invalid witnesses or collude with other users to reuse their witnesses. Also, once $\mathcal{A}$'s IAs are on the blockchain, $\mathcal{A}$ cannot change them due to the blockchain's immutable property. Finally, the binding property of the commitment scheme does not allow $\mathcal{A}$ to change his committed IAs in the commitment-revealing process. Hence, under these assumptions of Theorem 1, the proposed system provides the unforgeability property. □

2) ANONYMITY

Theorem 2. *If the hash function* H *is oneway, the public blockchain is anonymous, zk-SNARK satisfies the zero-knowledge property, the ElGamal encryption is secure against chosen-plaintext attacks, the commitment scheme is hiding, and the* (k, n)*-threshold SSS satisfies the correctness property, then the proposed system provides anonymity.*

Proof. We create a series of games from Game_0 to Game_F as follows.

• Game_0 is a real game where $\mathcal{A}$ tries to find the PIA associated with PS_i.

• Game_1 is identical to Game_0, except that $\mathcal{A}$ succeeds at finding the preimage of $\hat{h}_i \leftarrow \mathsf{H}(m_i + 1)$, which is public on the blockchain. Because H satisfies the onewayness property, it is infeasible to discover the preimage of a given hash value. Hence, Game_1 is indistinguishable from Game_0.

• Game_2 is identical to Game_1, except that $\mathcal{A}$ succeeds at finding the owner of a PS by analyzing the PS on the public blockchain. Because the public blockchain is anonymous, generating a PS does not require information about the owner's identity. Therefore, $\mathcal{A}$ cannot gain information about the PS's owner, and thus Game_2 is indistinguishable from Game_1.

• Game_3 is the same as Game_2, except π_C is simulated without the correct witnesses. Because zk-SNARK is zero-knowledge, π_C only shows that it is associated with a PS without revealing information about its witnesses. Therefore, Game_3 is indistinguishable from Game_2.

• Game_4 is identical to Game_3, except that $\mathcal{A}$ succeeds at decrypting $\mathbb{C}_k = \{c_1, \ldots, c_k\}$ without the corresponding $SK_{\mathbb{V}_k} = \{SK_{v_1}, \ldots, SK_{v_k}\}$, where $c_i \leftarrow \mathsf{ENC}(s_i, PK_{v_i})$ for $i = 1, \ldots, k$. However, because the ElGamal encryption is secure against chosen-ciphertext attacks, it is infeasible for $\mathcal{A}$ to decrypt $\mathbb{C}_n$ without corresponding $SK_{\mathbb{V}_n}$. Therefore, Game_4 is indistinguishable from Game_3.

• Game_5 is identical to Game_4, except that $\mathcal{A}$ succeeds at finding the committed value from the commitment. Because the commitment scheme is hiding, Game_5 is indistinguishable from Game_4. In this case, $\mathcal{A}$ cannot find the committed x_i from commitment y_i, which is public in the blockchain and associated with the target PS.

• Game_F is identical to Game_5, except that $\mathcal{A}$ succeeds at reconstructing the secret generated by the (k, n)-threshold SSS using less than k shares. Because the (k, n)-threshold

SSS satisfies the correctness property, $\mathcal{A}$ cannot reconstruct the secret $e_i \leftarrow g^{PIA}$ using $\mathbb{S}_l$, where $l < k$. Therefore, $\mathsf{Game_F}$ is indistinguishable from $\mathsf{Game_5}$.

We can see that $\mathsf{Game_F}$ is the final game where $\mathcal{A}$ cannot gain information about the PS's PIA or m_i even when $\mathcal{A}$ can obtain the PS's $\pi_C, \hat{h}_i, \mathbb{C}_n, \mathbb{Y}_t$, and $\mathbb{S}_l$, where $l < k$. Therefore, the proposed system provides the anonymity property. □

3) TRACEABILITY

Theorem 3. *If the system satisfies the unforgeability property, zk-SNARK is sound, the ElGamal encryption is correct, the public blockchain is immutable, and the (k, n)-threshold SSS is correct, the proposed system provides traceability.*

Proof. We also create a series of games from $\mathsf{Game_0}$ to $\mathsf{Game_F}$.

- $\mathsf{Game_0}$ is a real game where $\mathcal{A}$ tries to prevent the IP and validators from finding the PS's PIA.
- $\mathsf{Game_1}$ is identical to $\mathsf{Game_0}$, except that $\mathcal{A}$ succeeds at masquerading as another user or joining the system without registering $\mathcal{A}$'s IAs to the IP. Because the proposed system satisfies the unforgeability property, $\mathsf{Game_1}$ is indistinguishable from $\mathsf{Game_0}$.
- $\mathsf{Game_2}$ is the same as $\mathsf{Game_1}$, except that $\mathcal{A}$ succeeds at generating π_C without correct witnesses. Because zk-SNARK satisfies the soundness property, $\mathsf{Game_2}$ is indistinguishable from $\mathsf{Game_1}$. This means that $\mathcal{A}$ cannot omit $\mathbb{C}_n$ when generating π_C or calling the SC's function **authentication**$(\pi_C, \vec{x}_C)$ in Step 2.
- $\mathsf{Game_3}$ is identical to $\mathsf{Game_2}$, except that $\mathcal{A}$ succeeds at breaking the correctness property of the ElGamal encryption. Because the ElGamal encryption is correct, $\mathsf{DEC}(c_i, SK_{v_i})$ always returns s_i if c_i is generated using $\mathsf{ENC}(s_i, PK_{v_i}; r)$. Therefore, $\mathsf{Game_3}$ is indistinguishable from $\mathsf{Game_2}$.
- $\mathsf{Game_4}$ is identical to $\mathsf{Game_3}$, except that $\mathcal{A}$ succeeds at deleting data on the blockchain. Because blockchain is immutable, $\mathsf{Game_4}$ is indistinguishable from $\mathsf{Game_3}$. Therefore, $\mathcal{A}$ cannot delete its $\mathbb{C}_n$ or $\mathbb{Y}_t$ once they are stored in the blockchain.
- $\mathsf{Game_5}$ is identical to $\mathsf{Game_4}$, except that $\mathcal{A}$ succeeds at generating π_D without s_i and SK_{v_i} in terms of validator v_i. As before, because zk-SNARK satisfies the soundness property, $\mathsf{Game_5}$ is indistinguishable from $\mathsf{Game_4}$. $\mathcal{A}$ cannot upload incorrect s_i' to the SC because $\mathcal{A}$ cannot generate π_D using s_i'.
- $\mathsf{Game_F}$ is identical to $\mathsf{Game_5}$, except $\mathcal{A}$ succeeds at breaking the correctness property of the (k, n)-threshold SSS. Because the SSS is correct, $\mathsf{Game_F}$ is indistinguishable from $\mathsf{Game_5}$.

We can see that, in $\mathsf{Game_F}$, $\mathcal{A}$ cannot break the PIA-opening process. As a validator, $\mathcal{A}$ cannot send invalid s_i' to the SC to break the PIA-opening process. As an user, $\mathcal{A}$ cannot masquerade as another user or join the system with invalid witnesses because of the system's unforgeable property. The correctness property of the SSS and the ElGamal encryption ensures that $\mathcal{A}$ can neither generate nor distribute invalid SSS shares to validators in the authentication process. Hence, the only way for $\mathcal{A}$ to join the system is honestly using his PIA to generate and distribute SSS shares. In addition, because the SC keeps his $\mathbb{C}_n$ and $\mathbb{Y}_t$ in the blockchain storage, $\mathcal{A}$ cannot delete them. Hence, $\mathcal{A}$'s secret is always reconstructable in $\mathsf{Game_F}$, and both the requirements (1) and (2) of the traceability property are satisfied. Therefore, the proposed system provides traceability. □

TABLE 3. Parameters for off-chain simulation's environment.

Parameters	Descriptions
CPU	Intel(R) Core(TM) i5-4210M @ 2.60 GHz
RAM	4 Gb
OS	Ubuntu 21.10
Proving scheme	PGHR13 [14]
Hash function	MiMC [27], SHA256
DSA	EdDSA
Elliptic curve	ALT-BN128 [28], Baby Jubjub [29]
Number of validators	10
SSS threshold	6 of 10

VI. PERFORMANCE EVALUATION

This section shows the simulation results of the proposed system in terms of performance times and transaction costs. We simulate the proposed system according to the off-chain and on-chain tasks that each entity has to perform. Table 3 presents the parameters for the off-chain simulation, and Table 6 shows the parameters for the on-chain simulation.

A. OFF-CHAIN SIMULATION

1) SIMULATION TOOLS FOR ZK-SNARK

In this simulation, we employ Zokrates [26], a tool that supports implementing zk-SNARK. Essentially, three algorithms of zk-SNARK described in Section III can be represented by five Zokrates tasks.

- $\mathsf{Compile}(C) \rightarrow P$, where C is the arithmetic circuit C, and P is a set of polynomials. This function uses the quadratic arithmetic program (QAP) to transform the circuit C into polynomials P.
- $\mathsf{Setup}(P) \rightarrow (PK, VK)$, where PK and VK are a proving key and a verification key, respectively. This function creates a pair of keys (PK, VK) by implementing zk-SNARK's KeyGen.
- $\mathsf{Compute-witness}(P, a) \rightarrow \vec{w}$, where a is a set of inputs to the circuit C (both private and public), and $\vec{w}$ is the witness.
- $\mathsf{Generate-proof}(\vec{w}, PK) \rightarrow \{\pi, \vec{x}\}$, where π is the proof string and $\vec{x}$ is the set of public inputs. This function implements zk-SNARK's Proof.
- $\mathsf{Verify}(\pi, \vec{x}) \rightarrow b$, where b is a decision bit which is decided after running zk-SNARK's Verify with π and $\vec{x}$ as inputs.

2) SIMULATION RESULT

We divide the off-chain simulation into two parts. The first part is the simulation of circuit C, in which anonymous authentication, SSS distribution, and selective disclosure

processes are implemented. The second part is the simulation of circuit D, which implements the PIA-opening process. Table 4 presents the time complexity, space complexity and the frequency of the two parts' processes. We summarize the total results in Table 5.

a: CIRCUIT C AND THE ANONYMOUS AUTHENTICATION, SSS DISTRIBUTION, AND SELECTIVE DISCLOSURE PROCESSES

First, we create circuit C according to Algorithm 1. In Step 0, we initialize the zk-SNARK protocol using circuit C and run two tasks Compile and Setup to generate a pair of keys $\{PK_C, VK_C\}$ in terms of the IP. Afterward, we compute $\{root_i, cert_i, e_i\}$ for $i = 1, .., m$, where m is the number of users. The total time complexity of these processes for the IP is $time_{IP} = 106.24 + 0.44m$, where 106.24 is the time for running Compile and Setup, 0.44 is the time for generating $\{root_i, cert_i, e_i\}$. The size of PK_C is 1,536,047,032 bits (192 Mb), and VK_C is 51,976 bits (6.5 Mb). The size of $cert_i$ is 1,274 bits. Other parameters have the same 254-bit size.

In Steps 1 and 2, we create $\{h_i, \hat{h}_i, \mathbb{Y}_t, \mathbb{S}_n, \mathbb{C}_n\}$, where $t = 9$ and $n = 10$ in terms of U_i. Afterward, we run two Zokrates tasks Compute − witness and Generate − proof to generate π_C. The total time complexity of these processes is $time_{U_i} = 38.04$. Because v_i only needs to decrypt c_i to get s_i in Step 2, the time complexity for v_i is $time_{v_i} = 0.032m$. Because generating π_C requires PK_C, the total space for generating π_C is 1,536,047,032 + 4,528 = 1,536,051,560 bits, where 4,528 is π_C's size in the JSON format. Other parameters have the same 254-bit size. Notably, $\mathbb{S}_n$ and $\mathbb{C}_n$ require 2,540 bits because $n = 10$.

In Step 3, we generate $y_i \leftarrow \mathsf{H}(x_i, q_i)$ to verify x_i and q_i in terms of the SP. The time complexity for the SP is $time_{SP} = 0.01j$, where 0.01 is the time for generating y_i and j is the number of times time the SP verifies an MIA. The size of a single commitment is 254 bits.

b: CIRCUIT D AND PIA-OPENING PROCESS

In this simulation, we create circuit D according to Algorithm 2. In terms of the IP, we execute two tasks Compile and Setup in Step 0 to generate PK_D and VK_D. Afterward, we reconstruct a secret share using function Reconstruct. Because opening a PS's PIA requires the IP to run function Reconstruct once, the time complexity for the IP is $time_{IP} = 2.83 + 0.55z$, where 2.83 is the time for running Compile and Setup, 0.55 is the time for running function Reconstruct, and z is the number of times the IP implements the PIA-opening process. In terms of v_i, we decrypt c_i to get s_i in Step 2. Afterward, we run Compute − witness and Generate − proof to generate π_D and $\vec{x}_D$ as in Step 4. Because each time opening a PS's PIA also requires v_i to send s_i once, the time complexity for v_i is $time_{v_i} = 1.58z$. The size of PK_D is 31,169,120 bits (38.9 Mb) and VK_D is 27,048 bits (3.4 Mb). Because generating π_D requires PK_D, the total space for generating π_D is 31,169,120 + 4,528 = 31,173,648 bits, where 4,528 is π_D's size in the JSON format. The size of c_i is 254 bits.

TABLE 4. The time complexity, space complexity, and the frequency of processes in the off-chain simulation, where *m* is the total number of users, *z* is the number of times the IP implements the PIA-opening process, and *j* is the number of times the SP verifies an MIA.

Entities	Steps	Tasks	Time (s)	Size (bit)	Freq.
IP	Step 0	Generate $\{PK_C, VK_C\}$	106.24	1,536,099,008	1
		Generate $\{PK_D, VK_D\}$	2.83	31,196,168	1
	Step 1	Generate $root_i$	0.13	254	m
		Generate $cert_i$	0.17	1,274	m
		Generate e_i	0.14	254	m
	Step 4	Reconstruct e_i	0.55	254	z
U_i	Step 1	Generate h_i	0.12	254	1
	Step 2	Generate $\hat{h}_i$	0.13	254	1
		Generate $\mathbb{Y}_t$	0.18	2,286	1
		Generate $\mathbb{S}_n$	0.48	2,540	1
		Generate $\mathbb{C}_n$	0.56	2,540	1
		Generate π_C	36.62	1,536,054,906	1
v_i	Step 2	Decrypt c_i	0.03	254	m
	Step 4	Generate π_D	1.58	31,173,648	z
SP	Step 3	Generate y_i	0.01	254	j

TABLE 5. The total time complexity and space complexity of the off-chain simulation, where *m* is the total number of users, *z* is the number of times the IP implements the PIA-opening process, and *j* is the number of times the SP verifies an MIA. (unit: s).

Entities	Time complexity (s)	Space complexity (bit)
IP	109.07+0.44m+0.55z	1,578,397,312
SP	0.01j	254
U_i	38.04	1,536,062,780
v_i	0.03m + 1.58z	31,173,902

Although the time complexity and space complexity are large, most resources are spent for generating $\{PK_C, VK_C\}$, $\{PK_D, VK_D\}$, π_C, and π_D. Because these processes are implemented once (excluding π_D), the computational burden for the IP and users is acceptable. Generating π_D is a computational burden (1.58 seconds and 38.9 Mb) for validators because they must do this process m times, where m is the number of malicious users. However, this burden can be migrated to malicious users after their PIA is opened, and they must pay for their PIA-opening fees.

B. ON-CHAIN SIMULATION

In this simulation, we evaluate the SC performance by interacting with its functions and showing the transaction cost. Our target is to decrease the transaction cost as much as possible.

1) SIMULATION RESULTS

Under the parameters given in Table 6, we construct the SC according to Algorithm 3, using the Solidity programming language. We deploy the SC to the Ropsten Testnet and interact with the SC using Remix IDE and MetaMask. Each function's interaction is implemented in the form of a

TABLE 6. On-chain simulation environment parameters.

Parameters	Descriptions
Network	Roptsten Testnet
SC's Language	Solidity [30]
Wallet	MetaMask v10.14.6
IDE	Remix
SC's address	0x7e2573C6C1062F32C2eC49F76bfB26E39D829a80

TABLE 7. SC's transaction cost.

Functions	Transaction hashes	Cost (gas)
Deployment	0x6fa4e98e96ec80ef62efb5a48f8675ec5a2622580703a133114a445d91b21b26	3,977,797
authentication()	0x80cf490b976881d26141feda36192f2ae07c47f65b0bd9711a40945575596696d	1,852,354
openPIA()	0xe3eea77a6349dfa1ba304bd7d5ffe1235b7fbe958c0596644a7f8f20c5f0b425	961,035
queryPs()	0x4e083a4e04f7b2d4e706604458a586608bf94b40506f7bd373fedef95727a6fb	26,008
queryPIA()	0xd27235c80ea80055cdbbdf21aa1e0f0f2cb2f542729fcde12e0050ac1da8217e	32,696
queryMIA()	0xf50a15ff5b1cb71be15c43507486a86e9771dd27c09e10617485f0c5934c03b3	51,941

transaction. The cost of calling SC's functions is summarized in Table 7. The details of the simulation are as follows.

The IP is responsible for creating and deploying the SC to the blockchain network. This deployment is implemented once. In addition, when a user violates the system's policy, the IP needs to query the user's SSS shares and reconstruct the user's PIA. Therefore, the cost for the IP in this process is $cost_{IP} = 3{,}977{,}797 + 32{,}696d$, where 3,977,797 is the SC's deployment cost, 32,696 is the cost of calling function **queryPIA()**, and d is the number of times the PIA-opening process is implemented.

U_i needs to run function **authentication()** to authenticate its PS_i, distribute $\mathbb{C}_n$ to $\mathbb{V}_n$, and add $\mathbb{Y}_t$ to the blockchain. Because the function is implemented once, the total cost for U_i is $cost_{U_i} = 1{,}852{,}354$.

v_i needs to call function **openPIA()** to send s_i to the SC. The total cost for v_i is $cost_{v_i} = 961{,}035d$, where 961,035 is function **openPIA()**'s cost and d is the number of times the PIA-opening process is implemented. Because we use the (k, n)-threshold SSS, where $k = 6$ and $n = 10$ in this simulation, the total cost for opening a PIA is $6 \times 961{,}035 = 5{,}766{,}210$.

In terms of the SP, we run the function **queryPs()** and **queryMIA()** to check whether PS_i is authenticated by the IP. The cost for the SP is $cost_{SP} = 26{,}008a + 51{,}941b$, where 26,008 is the cost of function **QueryPS()**, a is the number of queries the SP issues to check a PS, 51,941 is the cost of function **queryMIA()**, and b is the number of queries the SP issues to check a PS's MIA. Putting them all together, Table 8 shows the total cost of our on-chain simulation.

TABLE 8. Total cost of the on-chain simulation, where *d* is the number of times the PIA-opening process is implemented, *a* and *b* are the number of times functions queryMIA() and queryPS() are invoked, respectively.

Tasks	IP	SP	U_i	v_i
Deployment	3,977,797	0	0	0
Authentication	0	0	1,852,354	0
PIA opening	0	0	0	$961{,}035d$
PS querying	0	$26{,}008b$	0	0
PIA querying	$32{,}696d$	0	0	0
MIA querying	0	$51{,}941a$	0	0
Total cost (gas)	$3{,}977{,}797 + 32{,}696d$	$51{,}941a + 26{,}008b$	1,852,354	$961{,}035d$

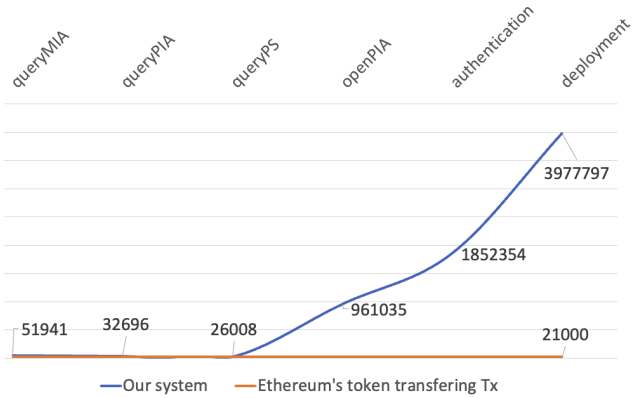

FIGURE 9. Comparison of transaction fees between the proposed system and Ethereum's token transferring transaction (unit: gas).

Compared with Ethereum's basic and most used transaction (token-transferring transaction), which costs 21000 gas, Fig. 9 illustrates how high the proposed system's transaction fees are.

VII. LIMITATIONS AND FUTURE WORKS

The first limitation is the computational power required for working with zk-SNARK. Our experiments show that resource-constrained devices (less than 4 Gb of RAM) cannot run zk-SNARK for the (k, n)-threshold SSS, especially when n is bigger than 15. The second limitation is that the decreasing number of validators can prevent the (k, n)-threshold SSS scheme from working normally. The increasing number of validators does not affect the (k, n)-threshold SSS with respect to previously joined users. However, if the number of validators that left the system is too large so that $n < k$, the (k, n)-threshold SSS and thus the PIA-opening process cannot work. The third limitation is from the fact that at least k malicious validators can collude to do the PIA-opening process in off-chain actions. Indeed, such malicious off-chain actions cannot be detected.

Based on the above limitations, future works can follow the direction of designing an efficient system using zk-SNARK-friendly hash functions and operations in the arithmetic circuit. Another direction would be to consider a novel mechanism for detecting off-chain actions of malicious validators using zk-SNARK and blockchain techniques.

VIII. CONCLUSION

This paper combines several cryptographic techniques to introduce a novel PPIdM system based on blockchain. Users' activities and service history are entirely hidden from all external entities. The proposed scheme provides anonymity by allowing users to authenticate themselves using zk-SNARK anonymously. The system's identity traceability utilizes the blockchain's consensus and the SSS algorithm. Selective disclosure is provided by using zk-SNARK and the hash-based commitment scheme. We calculated the performance of the proposed system by measuring the time complexity and space complexity in the off-chain channel and the computational power (gas cost) in the on-chain channel to show that the proposed system is efficient and realistic.

REFERENCES

[1] V. Kumar and A. Bhardwaj, "Identity management systems: A comparative analysis," *Int. J. Strategic Decis. Sci.*, vol. 9, no. 1, pp. 63–78, 2018.
[2] Y. Liu, D. He, M. S. Obaidat, N. Kumar, M. K. Khan, and K.-K. R. Choo, "Blockchain-based identity management systems: A review," *J. Netw. Comput. Appl.*, vol. 166, Sep. 2020, Art. no. 102731.
[3] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proc. 3rd Innov. Theor. Comput. Sci. Conf. (ITCS)*, 2012, pp. 326–349.
[4] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[5] M. M. ElGayyar, H. F. ElYamany, K. Grolinger, M. A. Capretz, and S. Mir, "Blockchain-based federated identity and auditing," *Int. J. Blockchains Cryptocurrencies*, vol. 1, no. 2, pp. 179–205, 2020.
[6] S. Gao, Q. Su, R. Zhang, J. Zhu, Z. Sui, and J. Wang, "A privacy-preserving identity authentication scheme based on the blockchain," *Secur. Commun. Netw.*, vol. 2021, pp. 1–10, Jun. 2021.
[7] J. Xu, K. Xue, H. Tian, J. Hong, D. S. L. Wei, and P. Hong, "An identity management and authentication scheme based on redactable blockchain for mobile networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6688–6698, Jun. 2020.
[8] P. Mell, J. Dray, and J. Shook, "Smart contract federated identity management without third party authentication services," 2019, *arXiv:1906.11057*.
[9] Y. Ren, F. Zhu, J. Qi, J. Wang, and A. K. Sangaiah, "Identity management and access control based on blockchain under edge computing for the industrial Internet of Things," *Appl. Sci.*, vol. 9, no. 10, p. 2058, May 2019.
[10] J. Alsayed Kassem, S. Sayeed, H. Marco-Gisbert, Z. Pervez, and K. Dahal, "DNS-IdM: A blockchain identity management system to secure personal data sharing in a network," *Appl. Sci.*, vol. 9, no. 15, p. 2953, Jul. 2019.
[11] B. Faber, G. C. Michelet, N. Weidmann, R. R. Mukkamala, and R. Vatrapu, "BPDIMS: A blockchain-based personal data and identity management system," in *Proc. 52nd Annu. Hawaii Int. Conf. Syst. Sci.*, 2019, pp. 1–10.
[12] M. Westerkamp, S. Gondor, and A. Kupper, "Tawki: Towards self-sovereign social communication," in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastructures (DAPPCON)*, Apr. 2019, pp. 29–38.
[13] C. Zhuang, Q. Dai, and Y. Zhang, "BCPPT: A blockchain-based privacy-preserving and traceability identity management scheme for intellectual property," *Peer Peer Netw. Appl.*, vol. 15, pp. 724–738, Jan. 2022.
[14] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 238–252.
[15] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Y. Yang, "High-speed high-security signatures," *J. Cryptogr. Eng.*, vol. 2, no. 2, pp. 77–89, 2012.
[16] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," in *Secure Information Networks: Communications and Multimedia Security, IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS)*, vol. 152, B. Preneel, Ed. Alphen aan den Rijn, The Netherlands: Kluwer, 1999, pp. 258–272.
[17] F. Saleh, "Blockchain without waste: Proof-of-stake," *Rev. Financial Stud.*, vol. 34, no. 3, pp. 1156–1190, 2021.
[18] A. Jules and J. Brainard, "Client-puzzles: A cryptographic defense against connection depletion," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 1999, pp. 151–165.
[19] C. Ye, G. Li, H. Cai, Y. Gu, and A. Fukuda, "Analysis of security in blockchain: Case study in 51%-attack detecting," in *Proc. 5th Int. Conf. Dependable Syst. Their Appl. (DSA)*, Sep. 2018, pp. 15–24.
[20] W. Y. Maung Maung Thin, N. Dong, G. Bai, and J. S. Dong, "Formal analysis of a proof-of-stake blockchain," in *Proc. 23rd Int. Conf. Eng. Complex Comput. Syst. (ICECCS)*, Dec. 2018, pp. 197–200.
[21] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, Feb. 1999, pp. 173–186.
[22] S. Gao, T. Yu, J. Zhu, and W. Cai, "T-PBFT: An eigentrust-based practical Byzantine fault tolerance consensus algorithm," *China Commun.*, vol. 16, no. 12, pp. 111–123, Dec. 2019.
[23] A. Savelyev, "Contract law 2.0: 'Smart' contracts as the beginning of the end of classic contract law," *Inf. Commun. Technol. Law*, vol. 26, no. 2, pp. 116–134, May 2017.
[24] D. A. Luong and J. H. Park, "Privacy-preserving blockchain-based healthcare system for IoT devices using zk-SNARK," *IEEE Access*, vol. 10, pp. 55739–55752, 2022.
[25] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *Topics in Cryptology* (Lecture Notes in Computer Science), vol. 3376, A. Menezes, Ed. San Francisco, CA, USA: Springer, 2005, pp. 136–153.
[26] J. Eberhardt and S. Tai, "ZoKrates–scalable privacy-preserving off-chain computations," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1084–1091.
[27] M. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen, "MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2016, pp. 191–219.
[28] C. Reitwiessner, "EIP-196: Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128," *Ethereum Improvement Proposals*, no. 196, Feb. 2017. [Online]. Available: https://eips.ethereum.org/EIPS/eip-196
[29] B. WhiteHat, M. Bellés, and J. Baylina, "EIP-2494: Baby jubjub elliptic curve," *Ethereum Improvement Proposals*, vol. 2494, pp. 1–7, Jan. 2020.
[30] C. Dannen, *Introducing Ethereum and Solidity*, vol. 1. Berkeley, CA, USA: Springer, 2017.

DUC ANH LUONG received the bachelor's degree in computer science from Sangmyung University, Seoul, South Korea, in 2020, where he is currently pursuing the M.S. degree in computer science. His research interests include blockchain technology, cryptography, the IoT, and information security.

JONG HWAN PARK received the B.S. degree from the Department of Mathematics, Korea University, Seoul, South Korea, in 1999, and the M.S. and Ph.D. degrees from the Graduate School of Information Security, Korea University, in 2004 and 2008, respectively.

He was a Research Professor at Kyung Hee University, from 2009 to 2011, and a Research Professor at Korea University, from 2011 to 2013. Since 2013, he has been an Assistant Professor with the Department of Computer Science, Sangmyung University, Seoul. His research interests include functional encryption, broadcast encryption, authenticated encryption, and various cryptographic protocols.