Hindawi
Security and Communication Networks
Volume 2024, Article ID 9854035, 1 page
https://doi.org/10.1155/2024/9854035

Retraction

Retracted: Digital Identity Verification and Management System of Blockchain-Based Verifiable Certificate with the Privacy Protection of Identity and Behavior

Security and Communication Networks

Received 8 January 2024; Accepted 8 January 2024; Published 9 January 2024

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope

(2) Discrepancies in the description of the research reported

(3) Discrepancies between the availability of data and the research described

(4) Inappropriate citations

(5) Incoherent, meaningless and/or irrelevant content included in the article

(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] Z. Song, G. Wang, Y. Yu, and T. Chen, "Digital Identity Verification and Management System of Blockchain-Based Verifiable Certificate with the Privacy Protection of Identity and Behavior," *Security and Communication Networks*, vol. 2022, Article ID 6800938, 24 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 6800938, 24 pages
https://doi.org/10.1155/2022/6800938

Research Article

Digital Identity Verification and Management System of Blockchain-Based Verifiable Certificate with the Privacy Protection of Identity and Behavior

Zhiming Song,[1,2,3] **Guiwen Wang**,[1] **Yimin Yu**,[1,2] **and Taowei Chen**[1,4]

[1]*School of Information, Yunnan University of Finance and Economics, Kunming, China*
[2]*Institute of Intelligent Application, Yunnan University of Finance and Economics, Kunming, China*
[3]*Yunnan Key Laboratory of Smart City and Cyberspace Security, Yuxi Normal University, Yuxi, China*
[4]*Yunnan Key Laboratory of Blockchain Application Technology, Kunming, China*

Correspondence should be addressed to Zhiming Song; zz2145@ynufe.edu.cn

Received 13 March 2022; Revised 25 July 2022; Accepted 18 August 2022; Published 30 November 2022

Academic Editor: Bharat Bhushan

Due to the advantages in self-sovereignty identity management and scalability of blockchain, digital identity verification and management systems (DIVMS) of blockchain-based verifiable certificates (VC) are getting more and more attention. However, user privacy in the systems' traditional architectures cannot be guaranteed. In this paper, the zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) referred to as Groth16 are introduced in order to implement privacy protection of the user's identity and behavior of DIVMS of blockchain-based VC. In the proposed architecture, the malleability attack of Groth16 is considered, and verifications of zero-knowledge proof (ZKP) and the digital signature of an identity provider (IDP) attached to VC and the status management of VC are implemented on the smart contracts of the blockchain to overcome single point failure. Furthermore, a prototype system is designed to verify the proposed architecture's capability in privacy protection and to evaluate its performances in cost and throughput. Finally, the security of the proposed architecture is discussed, and its comparisons are conducted with those existing blockchain-based DIVMSs, especially those systems using Groth16 of zkSNARKs to improve the privacy of user. All results mentioned above have shown that the proposed system is efficient and safe, and it can improve the privacy of DIVMS of the blockchain based VC while avoiding single point failure.

1. Introduction

1.1. Background. The rapid development of Internet technology provides conveniences for people's work, study, and lives, and makes it convenient for people to access services, exchange data and information from different application service providers (SP) through various digital identities. However, while people are enjoying the advantages of all kinds of Internet services, the identities and behavior information associated with their digital identities have also been exposed and stored in the digital identity verification and management system (DIVMS) of the application SPs, resulting in network security problems such as illegal use of identity, identity forging and disclosure, extortion based on the obtained identity. For example, in 2018, the British company Cambridge Analytica illegally obtained the identity information of more than 50 million Facebook users and pushed political advertisements to them to interfere in the US election [1]. In August 2018, the information of 3,000 chain hotels of China Huazhu Group was leaked, and as a result, more than 100 million users' identity information and check-in records were illegally obtained by hackers and then sold or used for extorting [2].

Obviously, the safety of digital identity, as "virtual DNA," is of great significance to safeguarding the computer and cyberspace. For digital identity, its storage, management, verification, authorization, and mapping with the real identity are executed through DIVMS. Therefore, the

architecture of DIVMS is the precondition to ensure the safety of digital identity.

However, most of the current DIVMSs, such as the combination of "user name + password" [3], CA digital certificates based on public key infrastructure (PKI), dynamic password technology, and electronic identity (eID) [4], have a centralized architecture based on application SPs or authorities. The architecture has inherent disadvantages such as centralization in the storage and management of identity information, single point failure, poor self-sovereignty management of the user's identity, nontransparent verification and authorization of identity, and so on. The disadvantages mentioned above still exist despite the fact that some corresponding federal authorization frameworks, such as OpenID [5], UMA [6], and OAuth [7], and two-factor [8] or three-factor security [9] authentication-system based on smart cards have been proposed consecutively.

In recent years, with the development of blockchain technology, its decentralization, openness, transparency, traceability, tamper-resistance, and other features have attracted the attention of all parties. Many scholars have also applied the blockchain technology to DIVMS and put forward different design schemes. The schemes can be divided into three categories in terms of main architectures:

1.1.1. Distributed Ledger. In this architecture, users' digital identities or their hash digests are stored and managed through the multilaterally maintained distributed ledger of the blockchain (as shown in Figure 1) to avoid the issuance of malicious certificates [10–12] and implement cross-domain verification and traceability of issued certificates [13, 14].

1.1.2. Smart Contract. In this architecture, the programmable smart contract of the blockchain is used to claim, publish, and authorize user's digital identity (as shown in Figure 2) [15–19].

1.1.3. Verifiable Certificate. In this architecture, the digital identity is issued to the user in the form of a verifiable certificate (VC) to implement the self-sovereignty management of the digital identity (as shown in Figure 3). In the meantime, blockchain is not used to publicly store or publish the digital identity, and its function mainly focuses on associating the digital identity (identity identifier) with the real one, managing the revocation information of the digital identity, implementing on-chain verification of the digital identity, etc. [20–25].

In comparison with the other two architectures, the architecture of VC has more advantages in the autonomy of digital identity and the scalability of blockchain. What is more, the well-known World Wide Web Consortium (W3C) is currently promoting some corresponding standards for blockchain-based VC [20]. As a result, the architecture is likely to become the main direction of blockchain-based DIVMS [26].

Due to the fact that the plaintext of user's digital identity is recorded in the VC [27], there unfortunately exists the risk of privacy disclosure in the traditional architecture of blockchain-based VC. Furthermore, the privacy of digital identity is gaining more and more attention. For example, the European Union enacted the General Data Protection Regulation in May 2018 [28] in order to strengthen the security and privacy protection of personal digital identity. China passed the Personal Information Protection Law of the People's Republic of China in August 2021 [29] to emphasize the importance of privacy protection for digital identity and regulate the behavior of digital identity processors, etc. Therefore, the privacy-compatible architecture of blockchain-based VC is becoming increasingly significant.

In terms of the privacy of digital identity, different application environments have different notions. First of all, with respect to the target object of privacy, in some applications, the target object is defined against the public (eavesdropping attackers) rather than the server (service provider) which can know the real identity of each user [8, 9]. On the other hand, in other applications, the target object is defined against the public and the server both of which cannot know the real identity of each user [30–38]. Secondly, with respect to the meaning of privacy, it is likely to contain the anonymity of the user's identity attributes [30, 31], the unlinkability of the user's different identity attributes [32], and their combination [33–38]. In the user's identity attributes, the target object of privacy cannot know the exact identity attribute of user. For example, the target object of privacy can just know a user is greater than 33 without knowing the user's real age value, which in the rest of this paper is referred to simply as the privacy protection of the identity attribute; As to the unlinkability of the user's different identity attributes, the privacy target object cannot collect different identity attributes of a user and thus cannot track operations performed by the same user using different identity attributes. For example, a user has the attributes age = 33 and income = 3300 dollars, but the target object of privacy cannot know that the attributes belong to the same user and thus cannot track operations performed by the same user with the two attributes, which in the rest of this paper is referred to simply as the privacy protection of identity behavior.

This paper mainly focuses on the privacy-compatible architecture of blockchain-based VC and the system architecture of blockchain-based VC, which includes three roles, as shown in Figure 3. Therefore, in this paper, the target object of privacy is defined as both the public and the service provider, and the meaning of privacy is the privacy protection of identity attribute and behavior.

1.2. Motivations and Contributions of this Work. To improve the privacy of the architecture of blockchain-based VC, some schemes referred to as "anonymous certificate" are proposed. In these schemes, some cryptography algorithms, such as Pederson commitment, blind signature, ring signature, accumulator, and so on, are combined to obtain the

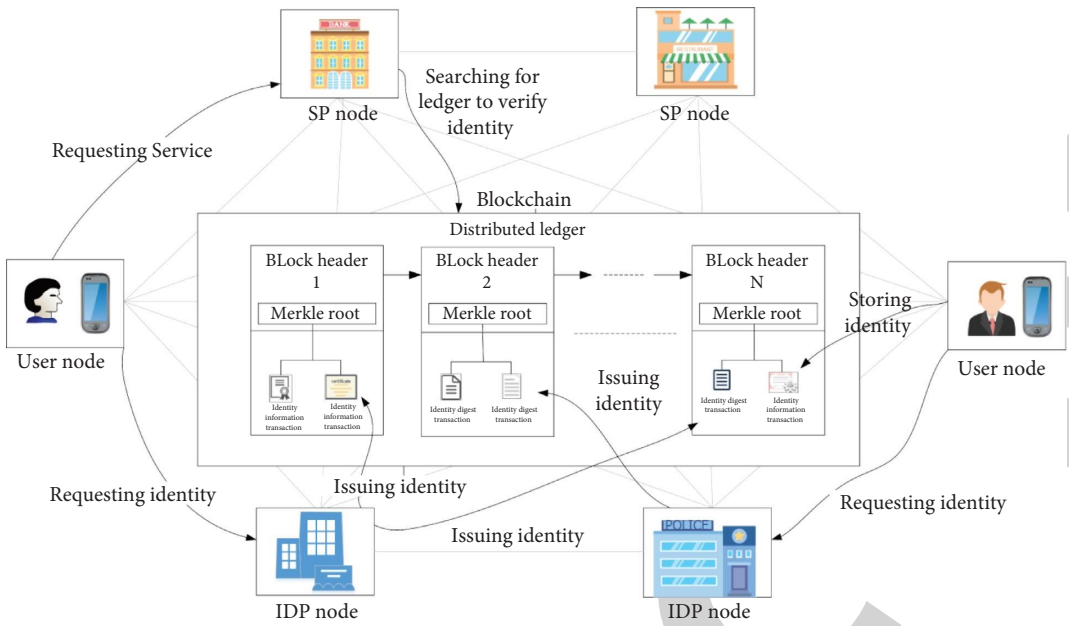

Figure 1: The architecture of DIVMS based on distributed ledger.

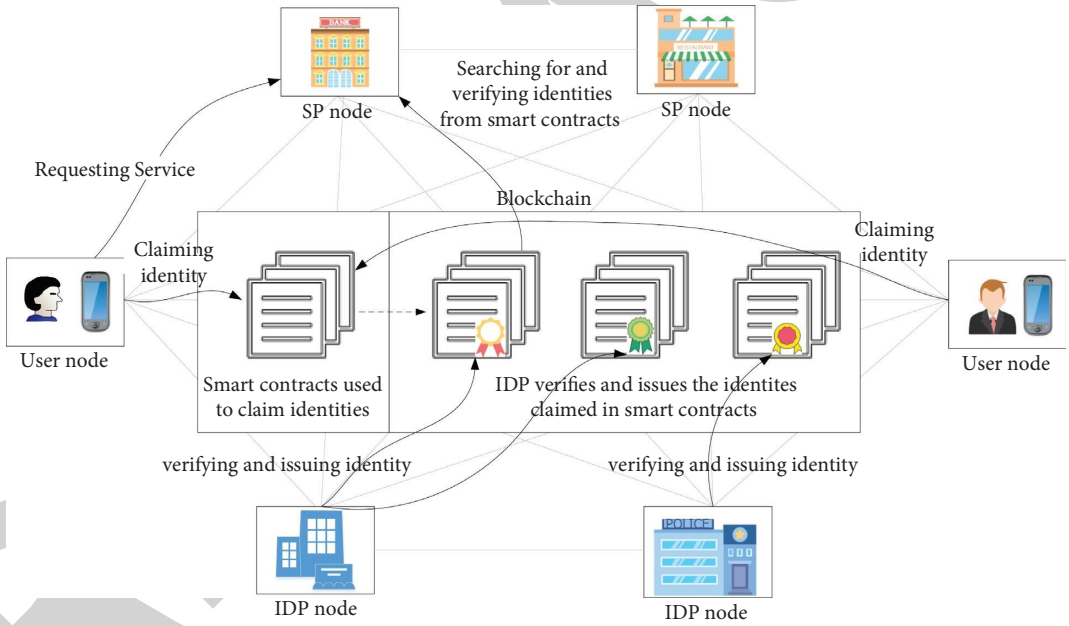

Figure 2: The architecture of DIVMS based on the smart contract.

privacy of identity attribute and behavior [33–37]. Besides, some schemes based on zero-knowledge succinct noninteractive arguments of knowledge (zkSNARKs) are also proposed [27, 30, 38]. However, compared with the schemes of anonymous certificate, some problems still exist in the schemes based on zkSNARKs. First of all, in the schemes of zkSNARKs, the algorithm Groth16 [39], is used to implement zero-knowledge proof (ZKP) of VC. However, for Groth16, there is still a malleability attack which has not been taken into consideration by the schemes. Secondly, in the schemes of [27, 30], zkSNARKs are used just to protect the privacy of user identity without protecting behaviour privacy. What is more, the scheme presented in [30] also does not guarantee that the holder of zero-knowledge proof (ZKP) must be the holder of the private key of VC. Finally, in the scheme of [27], the verifications related with ZKP are implemented outside the blockchain. However, off-chain verification in this case is nontransparent for the user, and a potential single point failure may occur.

Therefore, in this paper, a modified scheme based on zkSNARKs is proposed to improve problems which exist in the schemes of [27, 30, 38]. Furthermore, the main contributions of this paper are as follows: First of all, the detailed descriptions of the malleability attack of Groth16 are presented in this paper, and a defence method for the attack is given in the verification of ZKP of the VC. Secondly, to avoid

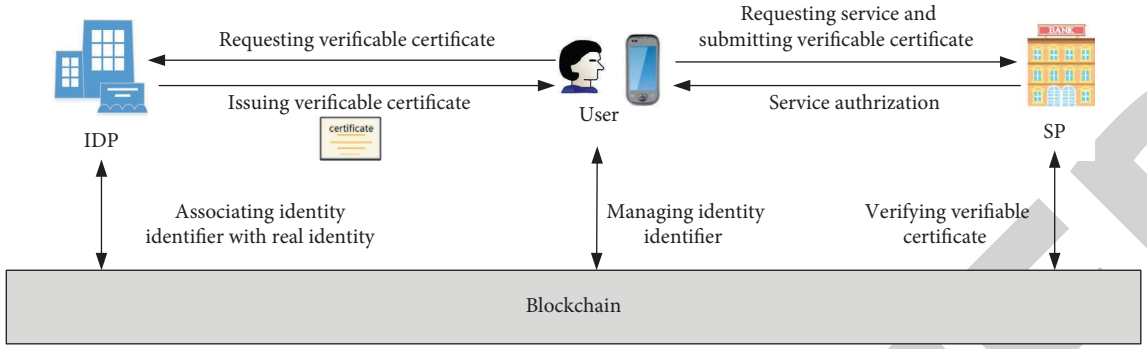

Figure 3: The architecture based on verifiable certificate.

single point failure, all operations associated with verification and management of VC are implemented on the blockchain by smart contracts. Thirdly, the proposed architecture, protocol, and algorithm of zkSNARKs are described in detail, and the actual use cases of the proposed architecture are given. Furthermore, the main programs and smart contracts associated with the use cases are shared on GitHub (https://github.com/Songzhiming123/zero-knowledge-proof-of-VC) to provide technical references for the readers who need to conduct relevant researches.

1.3. Organization. The rest of this paper is organized as follows: In Section 2, some blockchain-based DIVMSs and corresponding techniques are described. In Section 3, we introduce the preliminary steps involved in the proposed DIVMS of blockchain-based VC. In Section 4, we introduce the proposed system model and smart contracts' architecture. In Section 5, we describe the protocols and core codes of zkSNARKs contained in the proposed system architecture. The actual use cases of the proposed system architecture are described in Section 6. Next, evaluations, analyses, and comparisons of the proposed system architecture are presented in Section 7. Finally, the summary and outlook for the paper are given in Section 8.

2. Literature Review

As described in Section 1.1, blockchain-based DIVMSs mainly contain three architectures. Currently, with respect to different architectures, some corresponding systems have been proposed.

First of all, in terms of the architecture based on distributed ledgers, CertCoin, proposed by the scholars of MIT, is the first decentralized PKI system that utilizes the distributed ledger of the blockchain to manage PKI digital certificates [10]. In CertCoin, user identity is associated with their certificates' public key in a public way, and the registration, update, and revocation of certificates are realized through the transaction of the blockchain. Therefore, anyone in CertCoin can query the certificate information, which has solved the problems of certificate transparency and single point failure faced by the traditional certificate system based on PKI. With openness and transparency, Wang et al. [11] record certificate-related information using the distributed ledger of the blockchain to issue, manage, and repeal digital certificates. Everledger, a blockchain supply chain company, calculates the hash digests of features of diamonds such as height, width, thickness, texture, and so on, and then stores them into the tamper-resistant distributed ledger of the blockchain to use the hash digests as the identity certificates of diamonds [12]. Wang et al. [13] propose a DIVMS named BlockCAM. In BlockCAM, the distributed ledger of the blockchain is used to store the hash digests of cross-domain digital certificates to implement the identity verification of cross-domain entities. Shocard, a commercial blockchain digital authentication platform, calculates the hash digests of users' physical certificates and stores them into the distributed ledger of the blockchain so as to compare the hash digests stored in the blockchain with the ones obtained from users' physical certificates when identity verification of the user is required [14].

Secondly, in terms of the architecture based on smart contracts, the most typical DIVMS based on smart contract is UPort, where the address of the smart contract is taken as the unique identity identifier of the user and the index information related to the identity of the user is published in the smart contract (the real identity information is stored on the off-chain storage system) [15]. In addition, a DIVMS similar to UPort is also proposed [16] in which the address of a smart contract is also taken as the unique identity identifier of the user. The difference between this system and UPort is that in this system, the off-chain verification of identity is realized by determining whether the digital signature recorded in the smart contract is valid. A DIVMS referred to as EverSSDI based on smart contract of blockchain is proposed in [17]. In EverSSDI, the identity of the user is encrypted and published on the smart contract. Similarly, in EverSSDI, the contract address is also taken as the unique identity identifier of the user. In [18], a DIVMS named as smart contract public key infrastructure (SCPKI) is proposed where four smart contracts, namely, entity contract, attribute contract, signature contract, and revocation contract, are used to claim and manage the identities of users and implement trusted verification. In [19], taken as the user's digital identity, the public key of the account of the Ethereum blockchain is registered and published on the Ethereum smart contract by the authority for resource access control.

Thirdly, in terms of the architecture based on verifiable certificate, the W3C proposes a digital identity framework

based on the VC of blockchain [20], which has been gradually adopted by many decentralized self-sovereignty identity schemes [21, 22]. In [23], a DIVMS of blockchain-based VC is proposed to explore and implement self-sovereignty digital identity management based on blockchain. An identity authentication system called the Kiva identity protocol based on the VC of blockchain is proposed by Kiva, a nonprofit organization founded in San Francisco in [24], to apply to the query of loan credit records and to solve the problem of citizen credit difficulties caused by the lack of national credit institutions in Sierra Leone and other developing countries in Africa. Baidu, a Chinese Internet giant, also designs a DIVMS based on VC of blockchain named as CloudDID in [25] to explore the decentralized digital identity based on VC.

The systems described above belong to traditional blockchain-based DIVMSs, and although some systems use hash [12–14] or symmetric encryption [17] to preserve the privacy of digital identity from exposure, verification of digital identity needs to conduct hash comparison and decryption operations, which also result in the disclosure of digital identity privacy. To improve the privacy, some privacy-compatible systems have been proposed [27, 30–38, 40, 41]. In particular, these systems mainly focus on improving the privacy of the architecture of VC [27, 30, 31, 33–38] due to the fact that it has more advantages in the autonomy of the digital identity and the scalability of blockchain.

Anonymous certificate [31, 33–37] and zkSNARKs [27, 30, 38] are the two most frequently used schemes to improve the privacy of blockchain-based VC. First of all, in terms of the anonymous certificate scheme, the identity mixer (Idemix) developed by IBM is adopted in [31] to protect the privacy of identity attributes of IoT devices, and in the scheme, Idemix is used for credential issuance and zero-knowledge proofs for attribute verification. In addition, Sovrin is a commercialized system of anonymous certificates which can implement the privacy of identity and behavior. What is more, in Sovrin, the cryptographic accumulator is introduced to implement selective revocation of identity [33]. In the system of anonymous certificates presented on paper [34], the claims of identity property is hidden by Pederson's commitment to implement the privacy of identity. In the meantime, an effective self-blinding signature is used to implement the privacy of behavior. In the paper [35], the privacy of identity is also implemented through Pederson commitment, while the privacy of behavior is implemented through a randomizable signature. Furthermore, a dynamic accumulator is introduced in the paper [35] to implement selective revocation of identity. Similarly, the commitment mechanism is also used in paper [36, 37], but the difference is that to implement the privacy of behavior, in [36], the blind signature is used while in [37], the ring signature is used. Secondly, in terms of the zkSNARKs scheme, the corresponding systems and their problems have been described in detail in Section 1.2 of this paper. Obviously, from the descriptions, we can conclude that the systems of zkSNARKs indeed need to be improved. Therefore, this paper aims to improve the problems in the scheme of zkSNARKs and implement a privacy-compatible architecture of blockchain-based VC through zkSNARKs.

3. Preliminaries

3.1. Verifiable Certificate. Generated on the basis of real identity attribute of its owner, physical certificate is widely used in people's daily life such as driver's license, student's degree certificate, passport, and so on. In contrast to the physical certificate, a VC is the digital form of physical certificate based on a digital signature and other algorithms which are used to guarantee the authenticity of the identity attribute. As shown in Figure 3, the traditional architecture of blockchain-based VC often consists of three roles: IDP, user, and SP. IDP is the issuer of VC; SP is the service provider; and the service is authorized to the user who holds the VC issued by legal IDP. On the other hand, as shown in Table 1, VC often contains the items: the identity attribute of its holder, the unique identity identifier of its holder, the unique identity identifier of its issuer (IDP), and the digital signature of the above information generated by its issuer (IDP).

In this paper, only one identity attribute is contained in VC, and it is the item "Attribute," shown in Table 1, to guarantee the minimum privacy disclosure. Furthermore, in Table 1, User DID is the identity identifier of the holder of VC, and it is the hash digest of the holder's public key, PK_{user} (User DID = keccak256 (PK_{user})). IDP DID is the identity identifier of IDP and is the account address of IDP' Ethereum blockchain. IDP Signature is the digital signature generated by IDP with the elliptic curve digital signature algorithm, ECDSA, precompiled in the Ethereum blockchain.

In addition, Figure 4(a) shows that in the traditional architecture shown in Figure 3, when a user submits multiple VCs shown in Table 1 to SP to request corresponding services, SP can easily collect the identity attributes of the user and take advantage of the identity identifier User DID to trace the behavior of the user with different identity attributes. Therefore, as shown by the red solid line in Figure 4(a), the user's attributes and ownership of attributes are visible to SP. In contrary to Figures 4(a) and 4(b) shows the proposed architecture of this paper. In the proposed architecture, user's attributes and identity identifier, User DID, are hidden by ZKP based on zkSNARKs and invisible to SP, as shown with the red dotted line in Figure 4(b). Therefore, the privacy protection of identity and behavior of user can be achieved by the proposed architecture.

3.2. Blockchain Smart Contract. A smart contract is the on-chain code of a blockchain that is deployed and called after the participants of the blockchain have reached an agreement (Consensus). Therefore, the rules of smart contract cannot be unilaterally tampered. In the meantime, when a smart contract is deployed and called, the corresponding transaction is recorded in the distributed ledger of blockchain, and consequently, smart contracts have features such as openness, transparency, security, and credibility, which are meaningful in avoiding single point failure and constructing trusted systems. On the other hand, the participants of the blockchain can use high-level programming

Table 1: The contents and descriptions of VC in this paper.

Items	Description
Attribute	Single identity attribute of user
User DID	Globally unique identifier of user
IDP DID	Globally unique identifier of IDP
IDP signature	Digital signature of IDP

languages such as solidity, Golang, and so on to design smart contracts to meet the request of their decentralized applications. Therefore, the emergence of smart contracts is also known as the "era of blockchain 2.0."

Due to the advantages of smart contracts mentioned above, some blockchain-based DIVMSs have transplanted identity verification into smart contracts to implement on-chain verification of identity (although some systems based on smart contract are described in paragraph 3 of Section 2 of this paper, the systems scarcely implement the on-chain verification of identity). For example, in paper [40], smart contracts are used to implement transparent and anonymous secure multiparty computation of identity and to avoid single point of failure of identity verification. In paper [41], smart contracts are also used to implement on-chain registration, verification, and payment transactions of IoT devices. Similarly, this paper also uses smart contracts to achieve the on-chain verification of identity, but this paper has the following advantages compared with paper [40, 41]: (1) In comparison with paper [40], the on-chain verification of this paper is noninteractive without the need to make multiple interactive communications which will result in lower verification overhead; (2) in comparison with paper [41] which just emphasizes the privacy protection of identity, the on-chain verification of this paper can protect both the privacy of identity and behavior.

Finally, the blockchain platforms currently supporting smart contracts include Ethereum [42], Hyperledger [43], and so on. In the rest of this paper, ZKP based on zkSNARKs is implemented in the smart contract of Ethereum. Therefore, the Ethereum platform is adopted in this paper.

3.3. ZKP Based on zkSNARKs. ZKP is an encryption technique through which a prover can convince the verifier that a certain statement is correct without providing the verifier with any additional information or leaking any information about the witness. The statement could be that the prover knows the preimage of the hash value [32] or the plaintext content of the encrypted data [44].

The schemes of ZKP can be divided into two categories: interactive [45] and noninteractive [46]. Compared with the interactive, the noninteractive does not need to make multiple interactive communications in the proving process, thus avoiding the collusion attack while ensuring higher security. Particularly in the applications of blockchain, the noninteractive can avoid transaction confirmations caused by repeated on-chain interactions, thus making it possible to improve the applications' privacy without influencing the performance of blockchain.

Currently, zkSNARKs is taken as an efficient implementation of noninteractive zero-knowledge proof, and many excellent algorithms have been developed successively such as Groth16 [39], PGHR13 [47], and so on. Compared with other algorithms, Groth16 has the minimal calculation for verification and a succinct proof [48]. Therefore, the ZKP based on Groth16 is widely applied in the cryptocurrency systems based on blockchain such as Zcash [49], Filecoin [50], and so on. Similarly, in this paper, the algorithm Groth16 is adopted.

For Groth16, three steps are needed to implement ZKP:

(1) Setup $(R) \longrightarrow \sigma$:Take the polynomial, R as an input parameter and execute the operation setup to generate the common reference string (CRS), σ.

(2) Prove $(R, \sigma, \phi, w) \longrightarrow \pi$: Prover obtains R and σ, and takes them as input parameters along with ϕ and w to execute the step, Prove, and generate the ZKP, π where ϕ is the public parameters of prover and w is the private parameters to be proved.

(3) Verify$(\sigma, \pi) \longrightarrow 0/1$: Verifier obtains σ and π submitted by the prover and execute the step, verify to determine whether the prover really knows the private parameters, w. If the answer is OK, the result will be 1; otherwise 0.

The algorithm description of the three steps is as follows: First of all, the polynomial, R is expressed as follows:

$$R = \left\{ (\phi, w) \middle| \begin{array}{c} \phi = (z_1, z_2, \ldots, z_l) \in F^l \\ w = (z_{l+1}, z_{l+2}, \ldots, z_m) \in F^{m-l} \\ \sum_{i=0}^{m} z_i a_i(x) \cdot \sum_{i=0}^{m} z_i b_i(x) - \sum_{i=0}^{m} z_i c_i(x) = h(x)t(x) \end{array} \right\}, \quad (1)$$

where F is finite field, ϕ is the public parameters of prover, and w is the private parameters of prover which will be proved. $\sum_{i=0}^{m} z_i a_i(x) \cdot \sum_{i=0}^{m} z_i b_i(x) - \sum_{i=0}^{m} z_i c_i(x) = h(x)t(x)$ is the transformation result of a computing problem, $Z(\phi, w)$, which takes (ϕ, w) as input parameters. Furthermore, the transformation process is composed of three steps: (1) Converting $Z(\phi, w)$ into a arithmetic circuit (AC); (2) converting the AC into a Rank-1 constraint system (R1CS); (3) converting the R1CS into a quadratic arithmetic program (QAP), and Figure 5 shows an example of how to use the three steps to obtain R

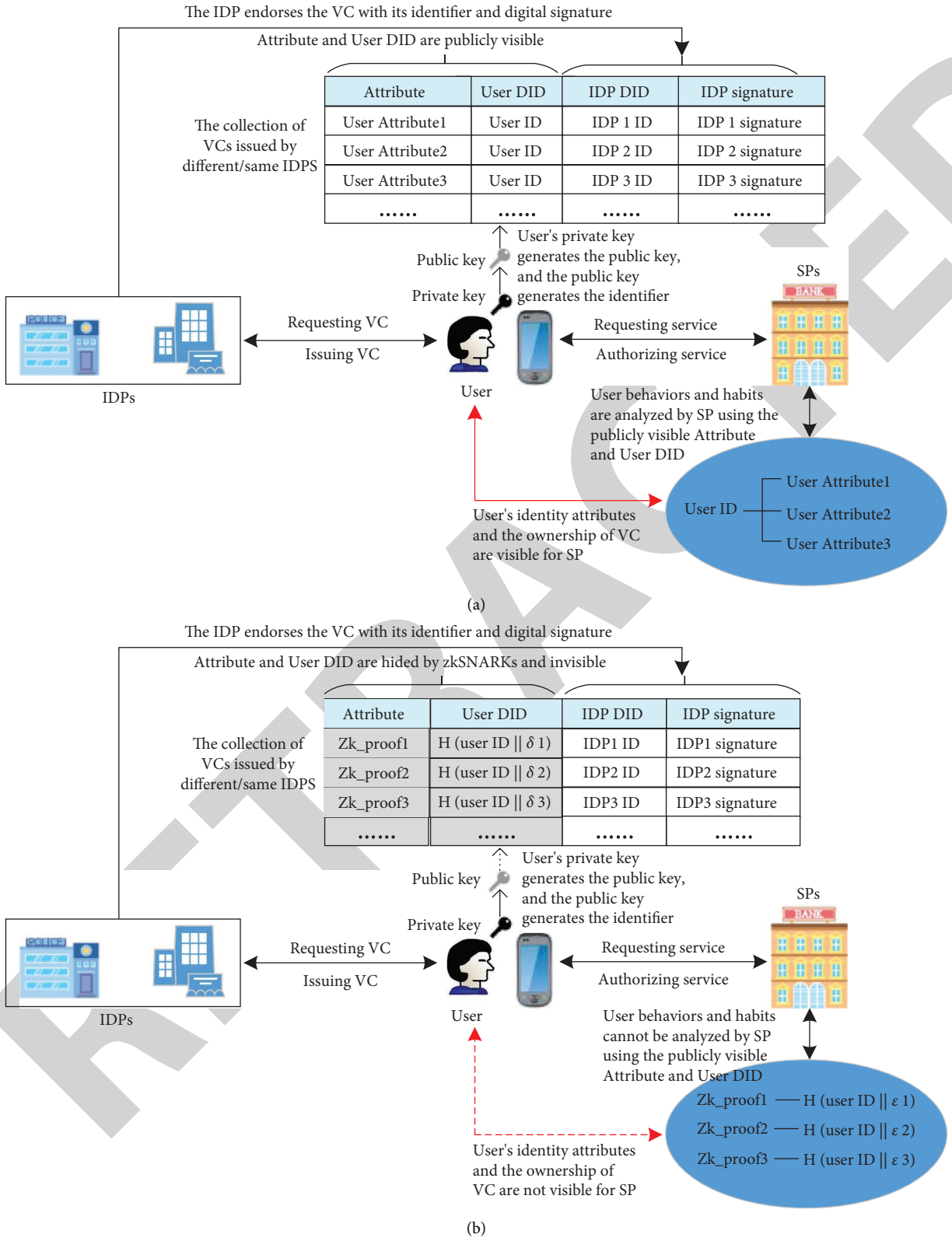

Figure 4: Comparisons in privacy protection between the traditional architecture and the proposed architecture. (a) Traditional architecture. (b) Proposed architecture.

where $Z(\phi$ and $w)$ is expressed as $T^3 + T + 5 = 35$, and ϕ and w are equivalent to (5, 35) and T, respectively.

Secondly, execute Setup $(R) \longrightarrow \sigma$, and obtain σ

$$\sigma = \left(\alpha, \beta, \gamma, \delta, \left\{x^i\right\}_{i=0}^{n-1}, \left\{\frac{\beta a_i(x) + \alpha b_i(x) + c_i(x)}{\gamma}\right\}_{i=0}^{l}, \left\{\frac{\beta a_i(x) + \alpha b_i(x) + c_i(x)}{\delta}\right\}_{i=l+1}^{m}, \left\{\frac{x^i t(x)}{\delta}\right\}_{i=0}^{n-2} \right), \tag{2}$$

where $\alpha, \beta, \gamma, \delta, x$ are the random number selected from the finite field F.

Thirdly, prover executes Prove $(R, \sigma, \phi, w) \longrightarrow \pi$ and obtains $\pi = \{A, B, C, \phi = (z_1, z_2, \ldots, z_l)\}$

$$\begin{aligned} A &= \alpha + \sum_{i=0}^{m} z_i a_i(x) + r\delta, \\ B &= \beta + \sum_{i=0}^{m} z_i b_i(x) + s\delta, \\ C &= \frac{\sum_{i=l+1}^{m} z_i \left(\beta a_i(x) + \alpha b_i(x) + c_i(x)\right) + h(x)t(x)}{\delta} \\ &\quad + (As + Br - rs\delta), \end{aligned} \tag{3}$$

where r, s is the random number selected from the finite field F.

Fourthly, verifier executes Verify $(\sigma, \pi) \longrightarrow 0/1$, to obtain the verification result, 0 or 1. The verification process is based on the following equation:

$$A \cdot B = \alpha\beta + \frac{\sum_{i=0}^{l} z_i \left(\beta a_i(x) + \alpha b_i(x) + c_i(x)\right)}{\gamma}\gamma + C\delta, \tag{4}$$

where $A, B, C, (z_1, z_2, \ldots, z_l)$ is from π submitted by prover, and $\alpha, \beta, \gamma, \delta, \{1/\gamma(\beta a_i(x) + \alpha b_i(x) + c_i(x))\}_{i=0}^{l}$ is from σ. If the (4) holds, the result is 1. Obviously, from (4), it can be seen that although prover does not leak any information about his/her the private parameters, $w = (z_{l+1}, z_{l+2}, \ldots, z_m)$, he/she still convince the verifier that he/she knows the real value of $w = (z_{l+1}, z_{l+2}, \ldots, z_m)$.

In addition, (4) can be written as follows:

$$e(G,H)^{A\cdot B} e(G,H)^{(-\alpha)\beta} e(G,H)^{\left(-\sum_{i=0}^{l} z_i \left(\beta a_i(x) + \alpha b_i(x) + c_i(x)\right)\right)/\gamma\gamma} e(G,H)^{(-C)\delta} = 1, \tag{5}$$

where $e(G_1 \cdot G_2) \longrightarrow G_T$ is the bilinear pairing of elliptic curve. G is the generator of G_1, and H is the generator of G_2. Furthermore, based on the properties of bilinear pairing, equation (5) can be expressed as follows:

$$e(A_1, B_1) e\left(-V_\alpha, V_\beta\right) \cdot e\left(-V_x, V_\gamma\right) \cdot e\left(-C_1, V_\delta\right) = 1, \tag{6}$$

where $A_1 = AG$, $B_1 = BH$, $V_\alpha = \alpha G$, $V_\beta = \beta H$, $V_x = \sum_{i=0}^{l} z_i(\beta a_i(x) + \alpha b_i(x) + c_i(x))G/\gamma$, $V_\gamma = \gamma H$, $V_\delta = \delta H$, $C_1 = CG$. Therefore, the verification of Groth16 can be conducted by the bilinear pairing expressed in (6).

3.4. The Implementation of the Adopted Architecture of ZKP. Currently, there are some libraries and development tool kits which can be used to implement the algorithm, Groth16, such as libsnark [51], Zokrates [52], and so on, as described in Section 3.3. In this paper, the development tool kit, Zokrates, is used.

As an efficient development toolkit to implement zkSNARKs, Zokrates can implement off-chain generation of ZKP and on-chain verification based on Ethereum smart contract. Zokrates mainly consists of four parts: (1) The processer of domain-specific language (DSL) which allows developers to conveniently describe the computing problem of ZKP at a high level of abstraction (Z (ϕ, w) of Section 3.3 and $T^3 + T + 5 = 35$ shown in Figure 5 are the computing problem of ZKP. In Zokrates, we can use DSL to describe them; (2) the complier which translates DSL into AC; (3) the generator of ZKP which can convert AC into R1CS, QAP, CRS, and ZKP; and (4) the constructor of Ethereum smart contract which outputs the verification smart contract of ZKP based on precompiled elliptic curve library of Ethereum (EIP196 and EIP197). Therefore, through Zokrates, the algorithm detail of Groth16 described in Section 3.3 can be masked and the developer just needs to focus on the specific problems related to privacy protection (focusing on how to construct the computing problem of ZKP with DSL), making it possible to implement the ZKP of zkSNARKs in a black-box way. In other words, in this paper, the details of Groth16 are masked, and we just focus on how to use DSL of Zokrates to construct the computing problem of ZKP which can hide the Attribute and User DID, as shown in Table 1 (Attribute and User DID shown in Table 1 will be taken as the private parameters).

Generally speaking, in the architecture of VC shown in Figure 3, SP verifies different identity attributes and then authorizes corresponding services, and User uses different identity attributes to obtain corresponding service authorization. Therefore, in this paper, in order to implement the ZKP of VC based on Groth16, SP firstly executes the step, setup, described in Section 3.3 to construct different computing problems of ZKP with DSL of Zokrates (the different computing problems are associated with identity requirements of different services' authorization). Next, User is taken as the prover to execute the step, Prove, described in Section

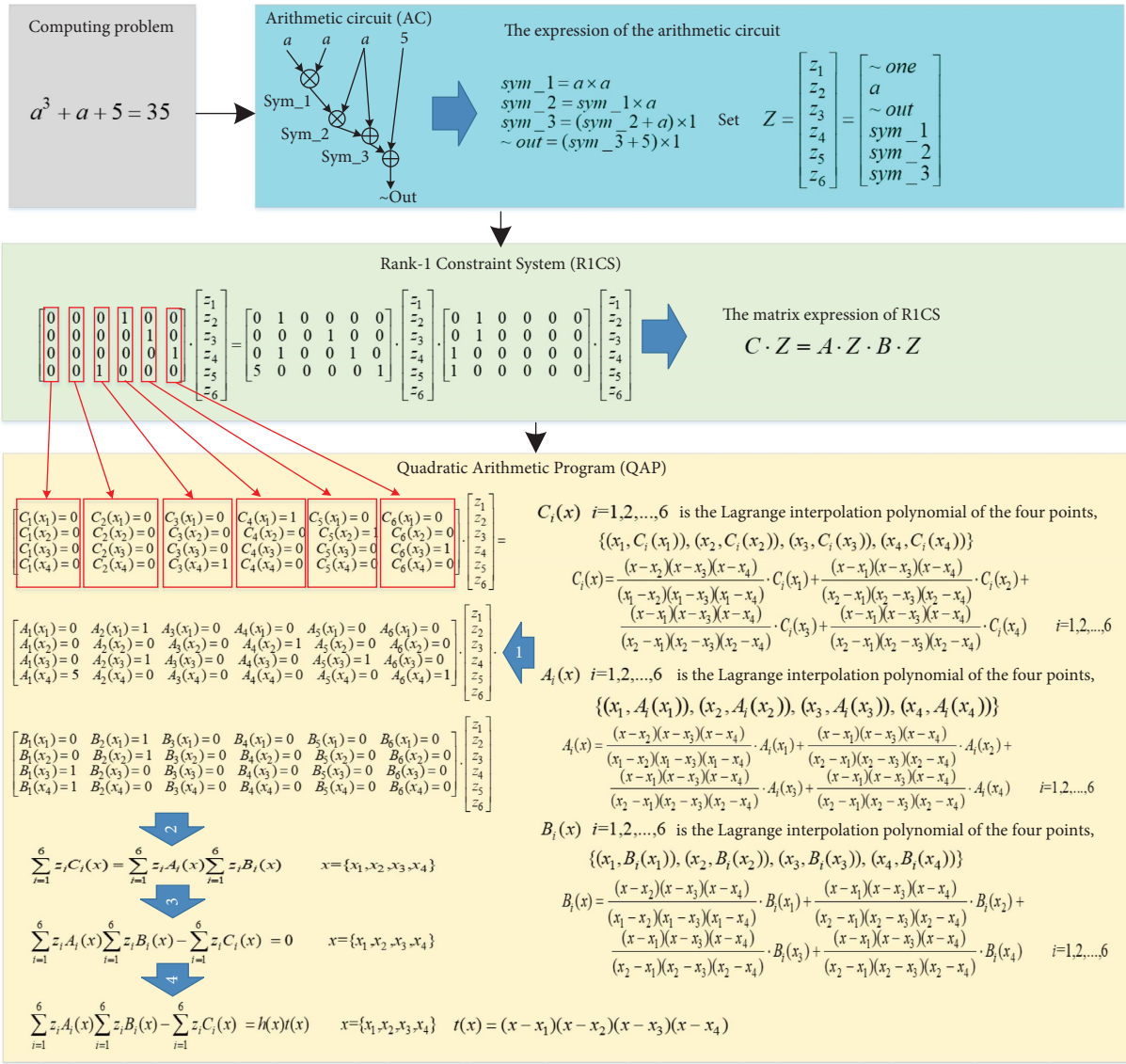

Figure 5: The process of obtaining the polynomial, R.

3.3 to generate different ZKPs based on different VCs. Finally, SP is taken as the verifier to execute the step, Verify, as described in Section 3.3 to verify ZKPs submitted by the user.

Figure 6 shows how the three steps, Setup, Prove, and Verify, as described in Section 3.3 are implemented by Zokrates. Furthermore, in Figure 6, the steps (Setup and Verify) surrounded by the dotted line are implemented by the verifier (SP), and the step (Prove) surrounded by the solid line is implemented by the prover (user). The steps will be described in the following procedure.

First of all, the step, Setup, is executed as follows:

(1) SP constructs the computing problem of ZKP with DSL as Z (Private Inputs and Public Inputs) where Private Inputs contain Attribute and User DID as shown in Table 1;

(2) SP executes the operation, compile (Z), to convert Z (Private Inputs and Public Inputs) into AC.

(3) SP executes the operation "Setup," to convert AC into QAP and obtains CRS: Verification key and Proving key where Verification key = $\{V_\alpha, V_\beta, V_\gamma, V_\delta, V_{\gamma\_abc}\}$ and $V_\alpha = \alpha G$, $V_\beta = \beta H$, $V_\gamma = \gamma H$, $V_\delta = \delta H$, $V_{\gamma\_abc} = \{1/\gamma(\beta a_i(x) + \alpha b_i(x) + c_i(x))G\}_{i=0}^{n}$. Proving key is the corresponding parameters associated with (2) of Section 3.3.

(4) SP constructs the verification smart contract of ZKP, VerifyContract, which is used to implement on-chain verification of equation (6). Furthermore, in the constructed verification smart contract, Verification key = $\{V_\alpha, V_\beta, V_\gamma, V_\delta, V_{\gamma\_abc}\}$ is used, and the operations related to bilinear pairing of Elliptic curve are implemented in the precompiled libraries of Ethereum, such as EIP196 and EIP197.

Secondly, the step, Prove, is carried out as follows:

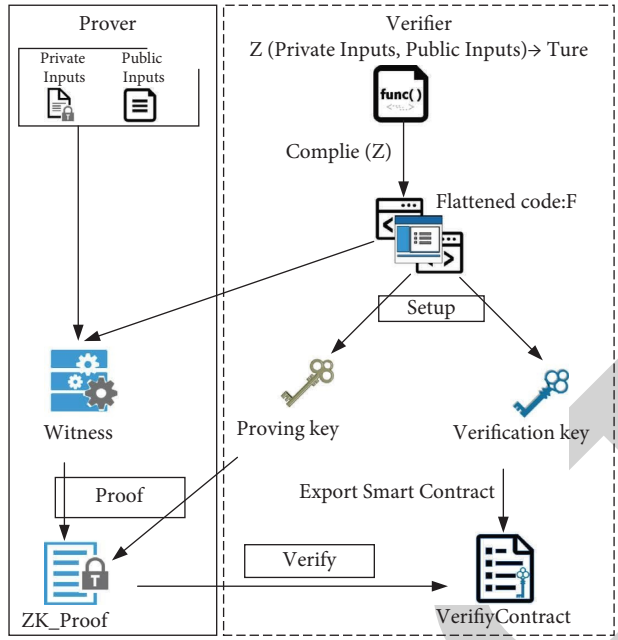

Figure 6: Operations of implementing zero-knowledge proof through Zokrates.

(1) To obtain corresponding services, User requests Z (Private Inputs, Public Inputs) and Proving key associated with the services from SP.

(2) User takes his/her Private Inputs, Public Inputs, and the obtained Proving key as input parameters and then executes the operation, Prove, to generate ZKP, $\pi = \{A_1, B_1, C_1, \text{Public Inputs}\}$ where $A_1 = AG$, $B_1 = BH$, $C_1 = CG$.

(3) User submits zero-knowledge proof, $\pi = \{A_1, B_1, C_1, \text{Public Inputs}\}$, to SP.

Finally, the step, Verify, is as follows: (1) SP calls the verification smart contract of ZKP, VerifyContract, to implement the on-chain verification of $\pi = \{A_1, B_1, C_1, \text{Public Inputs}\}$ submitted by the user.

4. The Description of Architecture and Contract

4.1. System Architecture. First of all, the proposed architecture is shown in Figure 7. In comparison with the traditional architecture shown in Figure 3, the proposed architecture has two significant features: (1) To implement the ZKP of VC, the development tool kit, Zokrates, is used; (2) to overcome single point failure, all operations associated with the verification and management of VC are implemented on the blockchain by the four Ethereum smart contracts, Cert_Status_SC, VerifySignature, Cert_is used_SC, Cert_ZK_Proof_SC. Furthermore, to facilitate the description and help reader to understand the core notion of the proposed architecture, Table 2 gives some symbols and their descriptions which will be used in the following parts.

Secondly, in the proposed architecture, three core roles are as follows:

(1) User: user is the holder of VC, and to obtain VC, user first needs to generate private key, SK_{user}, public key, PK_{user} and a Pair of session keys for RSA algorithm, SK_{RSA} and PK_{RSA}. After that, user submits the generated PK_{user}, PK_{RSA} and his/her real identity information to IDP. Next, IDP verifies whether user is legal and the received real identity information is authentic. If the verification result is correct, the VC with the format shown in Table 1 is issued to user. After receiving the issued certificate, to obtain the authorization of service from SP, user first needs to acquire the computing problem of ZKP, Z, and Proving key from SP, and then takes advantages of Zokrates to generate the ZKP, π, which can protect user's identity and behavior privacy. Finally, user sends π and the digital signature of IDP, Sign (Sign_H), attached to VC to SP to obtain the authorization of service.

(2) IDP: IDP is the issuer of VC, and the certificate with the format shown in Table 1 is issued to user after the user's public keys, PK_{user} and PK_{RSA} are received and the real identity information is verified successfully. On the other hand, after VC is issued to user, the status of the issued VC is activated by IDP through the smart contract, Cert_Status_SC.

(3) SP: SP is the provider of services and in accordance with different services authorization requirements for user's identity attributes, different computing problems of ZKP, Zs, are constructed by SP through the DSL of Zokrates. In the meantime, different Verification keys, Proving keys and the verification smart contracts, Cert_ZK_Proof_SCs, are generated. Furthermore, when user requests service, the corresponding Z and Proving key will be provided to

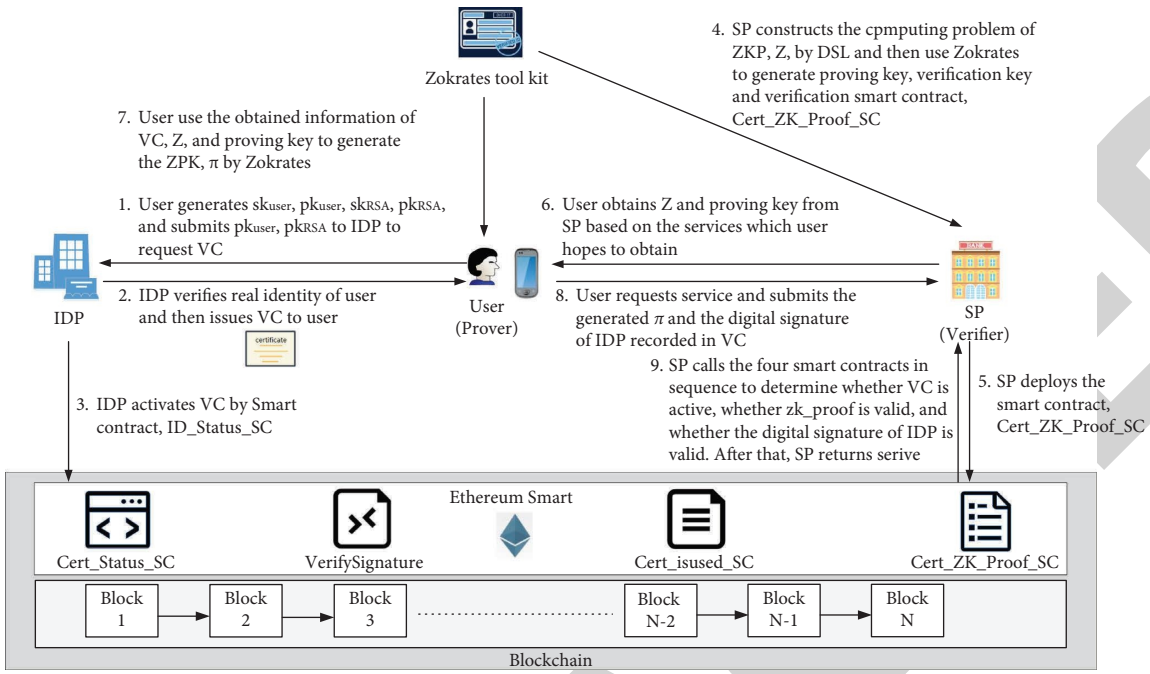

Figure 7: The system architecture of this paper.

Table 2: Symbols and their descriptions.

Symbol	Description
G	Base point of elliptic curve
(SK_{user}, PK_{user})	Private key and public key of user
δ	Random number
H_k (.)	Hash function: keccak256 (.)
$Fake\_DID_{user}$	H_k(User DID $\|\|\delta$)
Sign_H	H_k (attribute || user DID || IDP DID)
Sign (Sign_H)	Digital signature of IDP for VC
π	ZKP generated by VC
Hash_P	H_k ($Fake\_DID_{user}$ || Sign_H || IDP DID)
Z	Computing problem of ZKP
Private inputs	Private inputs of Z
Public inputs	Public inputs of Z and public inputs = ($Fake\_DID_{user}$, Sign_H, IDP DID)
Sign_EthAddress	Ethereum account address of signer

user. Finally, SP is taken as the verifier to verify the ZKPs, πs, submitted by user through the smart contracts, Cert_ZK_Proof_SCs.

Thirdly, it should be noted that, among the three roles, the determination of legitimacy is the prerequisite of system security. Therefore, this paper firstly assumes that all legitimate IDPs have an Ethereum account address known to other roles. On the basis of this assumption, the determination of legitimacy between the IDP and the user can be implemented through an operation shown in step (3) of Figure 8, namely, random number challenge, and the detailed operations will be described in step (3) of Section 5.1. Secondly, this paper assumes that the user clearly knows the SPs that he/she wants to access. Therefore, only the SP needs to unilaterally determine the legitimacy of the user during the interaction between users and SP. Furthermore, as shown in Figure 9, the determination of the legitimacy of a user is the process of the on-chain verification of the user's identity, and the detailed operations will be described in Section 5.3. Thirdly, SP needs to determine the legitimacy of IDP despite the fact that there is no direct interaction between SP and IDP. Furthermore, the determination of legitimacy of IDP is described in step (7) of Section 5.3, whose core notion is to determine whether the Ethereum account address of signer is the owned one of the legitimate IDP known to SP.

4.2. Architecture of Smart Contract. The four smart contracts shown in Figure 7 are as follows:

(1) Cert_Status_SC: This contract is used to manage the status of VC which has been issued, and when the status of this contract is active, the corresponding VC is valid, otherwise it is invalid.

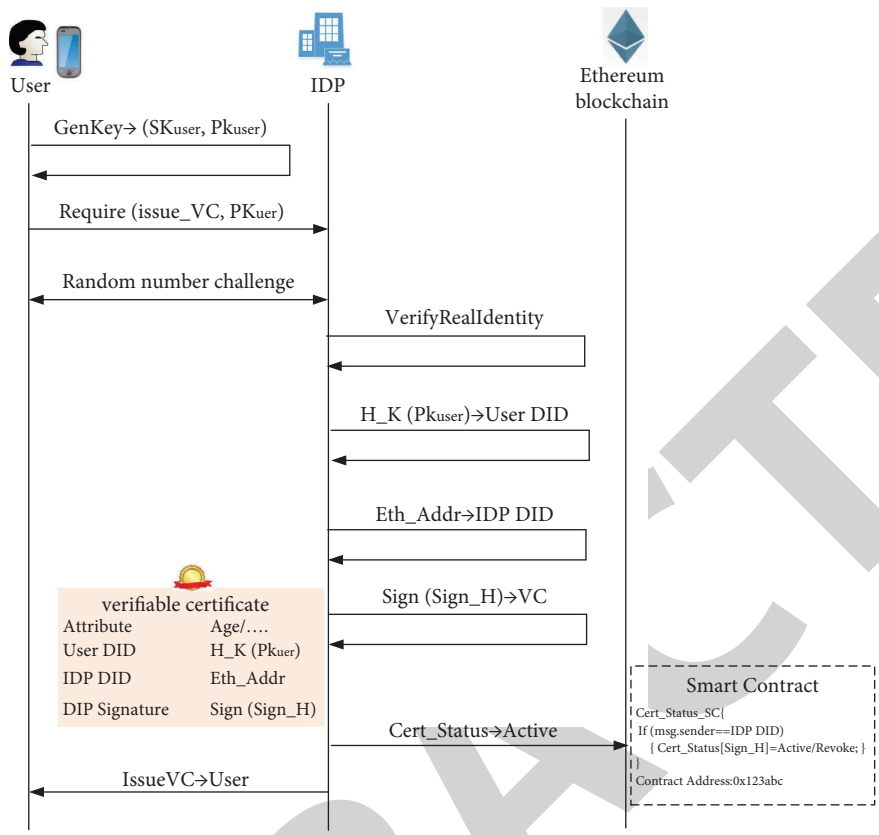

Figure 8: Issue of verifiable certificate.

(2) Cert_isused_SC: This contract is used to avoid replay attack and malleability attack of ZKP, π [48, 53]. On the other hand, the replay attack is that the valid π which has been used may be again used by the attacker of seeing it, and the malleability attack is that an attacker, seeing a valid π, can generate a different but still valid $\pi\prime$ (In this paper, the detail descriptions of malleability attack and its defence are given in the Appendix).

(3) Cert_ZK_Proof_SC: This contract is used to implement the on-chain verification of the π submitted by user based on the equation (6).

(4) VerifySignature: This contract is used to implement on-chain verification of the digital signature of IDP, Sign (Sign_H), attached to VC to ensure that VC used to generate π is issued by legal IDP.

5. Protocol and Algorithm of Proposed Architecture

5.1. Issue of Verifiable Certificate. Figure 8 shows the process of issuing VC and it is the refinement of steps 1 through 3 shown in Figure 7. The details are as follows:

(1) GenKey⟶ (SK_{user}, PK_{user}): User generates private key, SK_{user}, and public key, PK_{user}, based on the ecliptic curve algorithm, EdDSA.

(2) Require (issue_VC, PKuser): In accordance with service authorization requirement, user requests IDP to issue the corresponding VC. In the meantime, user's public key, PK_{user}, and real identity information are submitted to the IDP. On the other hand, user also sends a public key of RSA algorithm, PK_{RSA}, to IDP to complete the Step (3), Random number Challenge.

(3) Random number Challenge: After IDP receives user's public keys, PK_{user} and PK_{RSA}, he/she generates a random R. Next, IDP takes PK_{RSA} as the encryption key and uses the RSA algorithm to encrypt R, Encrpt (R, PK_{RSA}). In the meantime, IDP signs the random number R, Sign (H_K (R)), through ECDSA signature algorithm precompiled in Ethereum blockchain where IDP uses his/her Ethereum account address to complete the signature. After that, IDP sends Encrpt (R, PK_{RSA}) and Sign (H_K (R)) to user. After receiving Encrpt (R, PK_{RSA}) and Sign (R), user firstly decrypts Encrpt (R, PK_{RSA}) through the private key of PK_{RSA}, SK_{RSA}, and then obtains R. Next, on the basis of R and Sign (H_K (R)), user uses the signature verification algorithm of ECDSA, ecrecover (H_K (R), Sign(H_K (R)), precompiled in Ethereum blockchain, to obtain the Ethereum account address of signer. After that, user determines whether the obtained Ethereum account

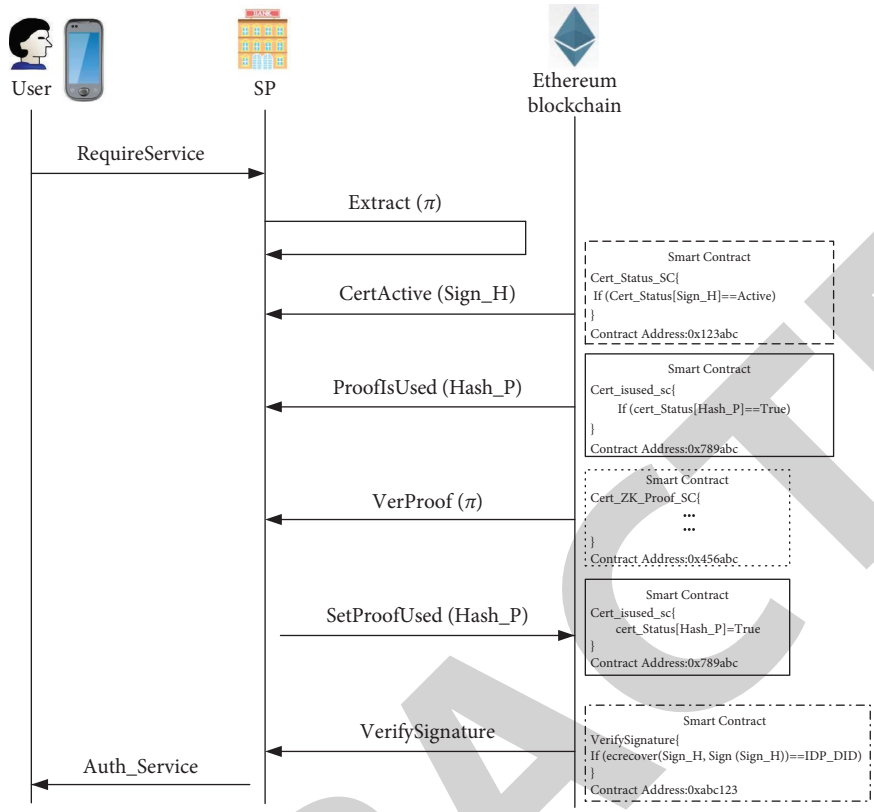

Figure 9: On-chain verification of ZKP and service authorization.

address is equal to the one owned by the IDP which he/she is accessing. If answer is ok, it indicates that IDP is legal. Next, user takes the private key, SK_{user} as the signature key and uses the EdDSA algorithm to sign R, sign (R, SK_{user}). After that, user sends sign (R, SK_{user}) to IDP. After receiving sign (R, SK_{user}), IDP determine whether sign (R, SK_{user}) is valid through the received PK_{user}. If the answer is ok, it indicates user is indeed the holder of PK_{user} and is legal.

(4) VerifyRealIdentity (User Identity): IDP verifies whether the real identity information of user is authentic.

(5) H_K (PK_{user})⟶User DID: If the real identity information of user is authentic, IDP will take advantages of PK_{user} to generate user's unique identity identifier, User DID = H_k(PK_{user}).

(6) Eth_Addr⟶IDP DID: The Ethereum account address of IDP is taken as the unique identity identifier of IDP, IDP DID.

(7) Sign (Sign_H)⟶VC: IDP concatenates user's identity attribute, Attribute, User DID and IDP DID, and then calculates their hash value, Sign_H = H_k (Attribute||User DID||IDP DID). After that, IDP calculates the digital signature of Sign_H, Sign (Sign_H), through the precompiled signature algorithm of Ethereum, ECDSA.

(8) Cert_Status⟶Active: IDP takes Sign_H as the input parameter of the smart contract, Cert_Status_SC, to activate the VC which will be issued to user.

(9) IssueVC⟶User: IDP generates the VC through the obtained Attribute, User DID, IDP DID and Sign (Sign_H), and then issues it to user.

5.2. The Generation of ZKP and Construction of Verification Smart Contract. Figure 10 shows the process which generates ZKP and constructs verification smart contract. Furthermore, the process is the refinement of steps 4 through 7, as shown in Figure 7. The details are as follows:

(1) GenZok⟶Z: SP constructs the computing problem of ZKP through using DSL of Zokrates, Z. Algorithm 1 shows the pseudocode of Z and its functions are as follows: (a) Helping user to prove that he/she is the holder of VC without leaking any information about identity attribute, Attribute, private key, SK_{user}, and unique identity identifier, User_DID. For example, from Algorithm 1, it can be seen that Attribute, SK_{user} and User_DID are all invisible Private Inputs. Furthermore, from steps 1 through 13 of Algorithm 1, it can be seen that User_DID has to be generated by PK_{user}, and PK_{user} has to be generated by SK_{user}. Therefore, the owner of User_DID must be the holder of SK_{user}; (b) Protecting user's behavior privacy. For example, in Algorithm 1, User_DID is contained in the Private Inputs, and SP just can see the forged version of User_DID contained in the Public Inputs, namely, Fake_DID_{user} = H_k(User DID||δ). Furthermore, δ is a random

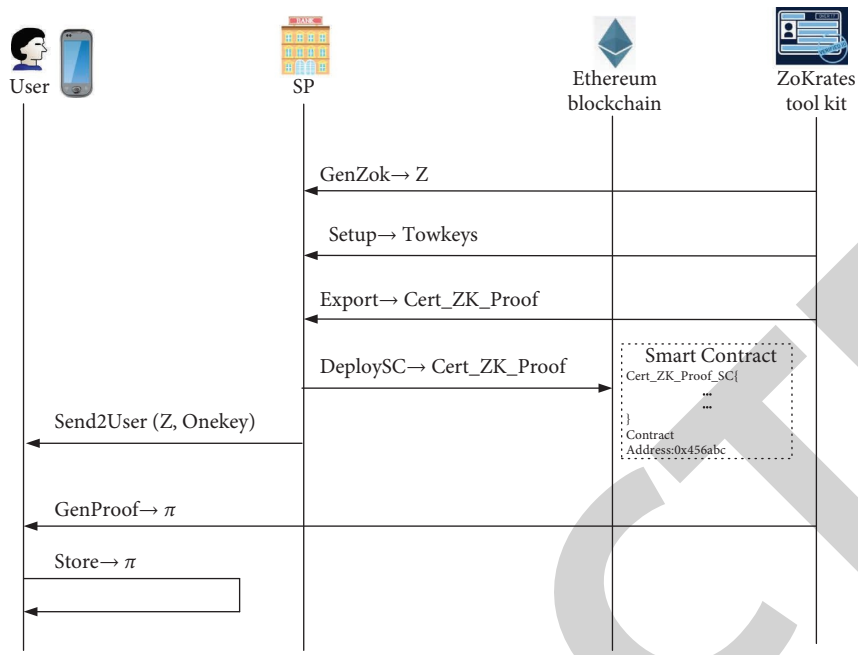

Figure 10: ZKP's generation and verification smart contract's construction.

number contained in the Private Inputs. Therefore, user can generate different Fake_DID$_{user}$s through different δs in order to hide his/her User_DID and implement behavior privacy (As shown in Figure 4(b), different Fake_DID$_{user}$s are generated by different δs to prevent SP from tracing the behavior of user through using the unique identity identifier, User_DID); (c) Helping user to proving that Private Inputs of Algorithm 1 are not fake and indeed are from user's VC. For example, in steps 20 through 25 of Algorithm 1, it can be seen that the variable contained in Public Inputs, Sign_H, has to be generated by H_K (Attribute||User DID||IDP DID). Furthermore, IDP DID has be verified in the smart contract, VerifySignature, and H_K(Attribute||User DID||IDP DID) has to be stored into the smart contract, Cert_Status_SC, to help SP to determine whether corresponding VC is active; (d) Helping user to prove that his/her privacy identity attribute, Attribute, meets the requirement of service authorization. For example, in steps 26 through 31 of the Algorithm 1, it can be seen that user's identity attribute, Attribute, contained in the Private Inputs of Algorithm 1 has to be within the range of the service's authorization requirement set by SP.

(2) Setup⟶Towkeys: SP executes the step, Setup, shown in Figure 6 to generate the CRS: Proving key and Verification key.

(3) Export⟶Cert_ZK_Proof: On the basis of Verification key, SP generates the smart contract, Cert_ZK_Proof, able to implement on-chain verification of ZKP submitted by user. Furthermore, the algorithm description of the smart contract, Cert_ZK_Proof, is given in Algorithm 2 and its core notion is to implement the bilinear pairing given in (6). In Algorithm 2, EIP196 stands for the precompiled library of Ethereum able to implement elliptic curve addition and Scalar multiplication, and EIP197 stands for the precompiled library able to implement elliptic curve bilinear pairing. In addition, in Algorithm 2, the variable, Public_in, contained in π corresponds to the one contained in Algorithm 1, Public Inputs = {Fake_DID$_{user}$, IDP DID, Sign_H}.

(4) DeploySC⟶Cert_ZK_Proof:SP deploys the generated smart contract, Cert_ZK_Proof, in Ethereum blockchain.

(5) Send2User (Z, Onekey): When user wants to obtain the corresponding service, he/she can obtain the computing problem, Z, and Proving key from SP.

(6) GenProof⟶ π: After obtaining the computing program, Z, and Proving key, user takes the four parameters, Attribute, SK$_{user}$, PK$_{user}$, User DID, δ, as Private Inputs of Z, and the three parameters, Fake_DID$_{user}$ = H_k (User DID ||δ), IDP DID and Sign_H = H_k (Attribute||User DID||IDP DID) as Public Inputs of Z. in the meantime, user executes the step, Prove, shown in Figure 6 to generate π.

(7) Store⟶ π:User stores π.

5.3. Identity Verification and Service Authorization. Figure 9 shows the process that SP calls the four smart contracts shown in Figure 7 to verify whether π and Sign (Sign_H) submitted by user is valid, and it is the refinement of steps 8 through 9, as shown in Figure 7. The details of the process are as follows:

(1) RequireService (π, sign(sign_H)): User submits the generated π, and the digital signature of IDP, Sign (sign_H), attached to the VC to SP in order to request the authorization of service.

(2) Extract (π) ⟶public_in:SP extracts Public Inputs contained in π, namely $\text{Fake\_DID}_{\text{user}}$, IDP DID and Sign_H.

(3) CertActive (Sign_H): SP takes Sign_H as the input parameter of the smart contract, ID_Status_SC, to determine whether the VC whose hash digest, H_k (Attribute||User DID||IDP DID), is equal to Sign_H is active.

(4) ProofIsUsed (Hash_P): If VC is active, SP takes Hash_P = H_k ($\text{Fake\_DID}_{\text{user}}$||Sign_H||IDP DID), as the input parameter of the smart contract, Cert_Isused_SC, to determine whether π has already been used, to avoid replay attack and malleability attack (As described in the Appendix, the malleability attack can be avoided by determining whether Public Inputs contained in π has already been used).

(5) VerProof (π): If π still has not been used, SP takes π as the input parameter of the smart contract, Cert_ZK_Proof_SC, to verify whether user is the holder of the ZKP and whether user' identity meets service authorization requirement.

(6) SetProofUsed (Hash_P) : If the verification of π is valid, SP takes Hash_P = H_K($\text{Fake\_DID}_{\text{user}}$||Sign_H||IDP DID) as the input parameter of the smart contract, Cert_Isused_SC, to mark that π has already been used.

(7) VerifySignature (Sign_H, Sign(Sign_H)):SP takes Sign_H and Sign (Sign_H) as the input parameters of the smart contract, VerifySignature, to determine whether the VC used to generate π is issued by legal IDP. In the smart contract, VerifySignature, the digital signature verification algorithm of ECDSA, ecrecover (Sign_H, Sign(Sign_H)), which has been precompiled in the Ethereum is used. Furthermore, the return value of ecrecover (Sign_H, Sign (Sign_H)) is the Ethereum account address of signer, Sign_Eth_address. Therefore, if Sign_Eth_address is equal to IDP DID extracted from the Public inputs of π, the VC used to generate π is from legal IDP.

(8) Auth_Service: If the results of the steps 3 through 7 described above are correct, SP will authorize the corresponding service to user.

6. Use Cases

To verify the proposed architecture and evaluate its performance when applied to the real scenery, this paper designs a prototype system in which user utilizes the ZKP mentioned above to prove to the SP that his/her VCs of age and income meet the requirements of service authorization without leaking any information about identity attributes (age and income) or behavior. For the designed prototype system, the three tools— ZoKrates, Remix, and Solidity— are used to generate the four smart contracts shown in Figure 7. In the meantime, the contracts are deployed on the private Ethereum blockchain, and then the three tools Html, Javascript, and Web3.js are used to develop the decentralized application (DApp) of the prototype system. Furthermore, Figure 11 shows the DApp of SP. In Figure 11(a), the verification of the ZKP for age's VC is given, and the verification for income is given in Figure 11(b).

For the prototype system, the user follows steps 1 through 9, as shown in Section 5.1 to obtain age's VC shown in Table 3, and income's VC as shown in Table 4 from two different IDPs. Obviously, from Tables 3 and 4, it can be seen that the user's identity attribute is publicly visible and the unique identity identifier, User DID, is equal. Therefore, as shown in Figure 4(a), it is easy for SP to obtain the identity the user and track his/her behavior. To avoid the exposure of behaviors, user generates two random numbers, $\delta 1$ and $\delta 2$, to forge two fake identity identifiers, $\text{Fake\_DID1}_{\text{user}}$ = H_k(User DID ||$\delta 1$) and $\text{Fake\_DID2}_{\text{user}}$ = H_k(User DID ||$\delta 2$). Next, user follows the steps 5 through 7, as shown in Section 5.2 to generate two ZKPs, π_1 and π_2 where user's private key, SK_{user}, public key, PK_{user}, age, income, User DID, and the two random numbers, $\delta 1$ and $\delta 2$, are taken as Private Inputs of Algorithm 1 (age, income, and User DID are parameters recorded in the two VCs of age and income), and the two forged identity identifiers, $\text{Fake\_DID1}_{\text{user}}$ and $\text{Fake\_DID2}_{\text{user}}$, and the two hash values, Sign_H1 = H_k (age || User DID || IDP DID1) and Sign_H2 = H_k(income || User DID || IDP DID2) are taken as Public Inputs of Algorithm 1. Finally, user submits the two generated ZKPs, π_1, and π_2, and the two digital signatures, IDP Signature1 and IDP Signature2, attached to the two VCs of age and income to SP. Next, SP calls the four smart contracts, as shown in Figure 7 to determine whether user's age and income meet the requirement of service authorization.

On the other hand, from Figure 11(a) and 11(b), it can be seen that after receiving π_1, π_2, IDP Signature1, and IDP Signature2 (The variables contained in the red rectangle of Figure 11), SP extracts the Public Inputs contained in π_1 and π_2 (The variables contained in the blue rectangle of Figure 11), and then calls the four smart contracts to obtain the verification results. Furthermore, in Figure 11(a) and 11(b), it is clear that although in VCs of age and income shown in Tables 3 and 4, User DID is equal, SP just can obtain two forged identity identifiers such as e71272eea656dc4295633c3b1249-f68e8053e0954019b019e88f6268518fe4fa shown in Figure 11(a) and 829096bb3c7dcec3570c96 d5d55f2aa444b-b69ecb65940df93f9e0f86c2abe15 shown in Figure 11(b). In the meantime, SP cannot obtain any information related to age and income. Therefore, as shown in Figure 4(b), the proposed system can realize the privacy protection of identity and behavior.

Although the prototype system mentioned above takes age and income as examples, it is also suitable for other identity attributes, and the difference lies in that instead of determining whether a user's attributes of age and income are in the range set by SP in the steps 26 through 30 of Algorithm 1, other attributes and their range should be determined. For example, in the prototype system mentioned above, the age requirement for service authorization is $18 < \text{age} < 60$, and the income requirement is

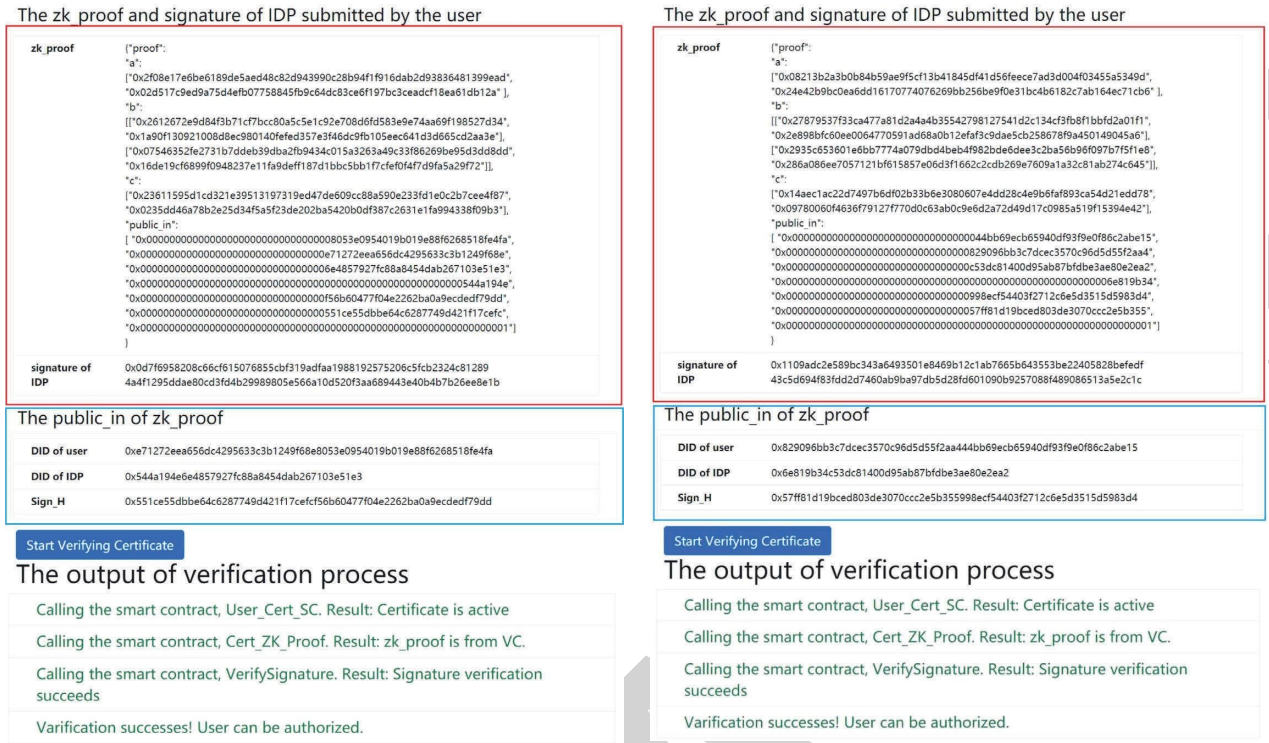

Figure 11: Interaction interface of DApp of SP. (a) Verification of age. (b) Verification of income.

income > 3000 dollars. Therefore, for the ZKPs of VCs of age and income, the steps 26 through 30 of Algorithm 1 should be Attribute ∈ [18, 60] and Attribute ∈ [3000,+∞], respectively, and for other attributes, the steps 26 to 30 of Algorithm 1 should be Attribute ∈ [corresponding attributes' ranges].

Finally, it should be emphasized that although, for different identity attributes, the steps 26 through 30 of Algorithm 1 are different, the finally generated π, Sign(-Sign_H) and smart contracts related to verification have almost the same size. Therefore, the results of analysis and evaluation of the prototype system are also suitable for other VCs.

7. Analysis and Evaluation

7.1. Cost Analysis. The average values of the total deployment and execution consumptions of Tables 5 and 6 are calculated and then taken as the total deployment and execution consumptions of the proposed architecture. Furthermore, the following equation is used to convert the average values into the number of ether to evaluate the cost of the proposed architecture:

$$\text{num}_{\text{ether}} = \text{Gas}_{\text{used}} \times \text{Gas}_{\text{price}}, \quad (7)$$

where $\text{num}_{\text{ether}}$ is the number of ether, Gas_{used} is the consumption of gas, and $\text{Gas}_{\text{price}}$ is the price of gas. For deployment and execution, Gas_{used} is 2963917 and 487361, respectively. For $\text{Gas}_{\text{price}}$, the price of the public Ethereum blockchain is used, and when this evaluation is conducted (on 26 September 2021), the price was approximately 71.25 Gwei. Therefore, for the deployment and execution, $\text{num}_{\text{ether}}$ is 0.21117908625 ether and 0.03472447125 ether, which are equivalent to 647.44 USD and 106.46 USD, respectively (on 26 September 2021, 1 ether ≈ 3065.84 USD). Obviously, with respect to $\text{num}_{\text{ether}}$, the cost of proposed system is not high. However, due to that the price of ether of the public Ethereum blockchain is too high, it is not suitable to deploy the proposed system to the public Ethereum blockchain. In other words, the proposed system architecture can be deployed to the private or allied Ethereum blockchain because this can not only implement the identity and behavior privacy but also reduce the cost.

7.2. Throughput Analysis. In real application scenarios, multiple users may simultaneously request services from the same SP. To respond to those service requests as quickly as possible, SP needs to create the same number of Ethereum accounts as the number of user requests and then use these accounts to concurrently call the four smart contracts, as shown in Figure 6 to implement verification and authorization based on ZKP. Therefore, to test the throughput of the proposed system in this situation, this paper varies the number of users requiring services from the same SP from 1 to 100, and then tests the response time in this situation. The test result is shown in Figure 12 where the horizontal coordinate axis is the number of users requiring services from the same SP, and the vertical coordinate axis is the average response time of verification and authorization for different

Table 3: The VC of user' age issued by IDP.

Items	Description
Age	33
User DID	0xb8d7659cb8e81248b58642f1d75014d1604603d45388342dc41502f63e8d5f9
IDP DID	0x6e819b34c53dc81400d95ab87bfdbe3ae80e2ea2
IDP signature	0x1109adc2e589bc343a6493501e8469b12c1ab7665b643553be22405828befedf43c5d694f83fdd2d7460ab9ba97db5d28fd601090b9257088f489086513a5e2c1c

Table 4: The VC of user' income issued by IDP.

Items	Value
Income	3500
User DID	0xb8d7659cb8e81248b58642f1d75014d1604603d45388342dc41502f63e8d5f9
IDP DID	0x544a194e6e4857927fc88a8454dab267103e51e3
IDP signature	0x65e4e978c9937bb0fb483b21491cfe6ff06591c2f7ef22279821ae36a95f739c298dcd9b4beb95847d87e2bbb807893ef3fd629b07fc4b99077022afadab3d461b

Inputs: Private Inputs = {Attribute, SK_{user}, PK_{user}, User DID, δ}, Public Inputs = {$Fake\_DID_{user}$, IDP DID, Sign_H}
Output: True or False
(1) $G = [G_x, G_y]$
(2) $pk = SK_{user} \cdot G$
(3) **IF** $pk == PK_{user}$
(4) **Goto step** 8
(5) **Else**
(6) **Return** False
(7) **EndIF**
(8) $User\_DID' = H\_K(PK_{user})$;
(9) **IF** $User\_DID' == User\_DID$
(10) **Goto step** 13
(11) **Else**
(12) **Return** False
(13) **EndIF**
(14) $Fake\_DID_{user}' = H\_K(User\_DID \,||\delta)$
(15) **IF** $Fake\_DID_{user} == Fake\_DID_{user}$
(16) **Goto step** 19
(17) **Else**
(18) **Return** False
(19) **EndIF**
(20) Sign_H' = H_K(Attribute || User DID || IDP DID);
(21) **IF** Sign_H' == Sign_H
(22) **Goto step** 26
(23) **Else**
(24) **Return** False
(25) **EndIF**
(26) **Set** Attribute_Range
(27) **IF** Attribute ∈ Attribute_Range
(28) **Return** True;
(29) **Else**
(30) **Return** False;
(31) **EndIF**
(32) Algorithm1 End

Algorithm 1: Computing problem of ZPK of verifiable certificate, Z.

Inputs: $\pi = \{A_1, B_1, C_1, Public\_in\}$
Outputs: True or False
(1) **Contract** Cert_ZK_Proof
(2) $[V_\alpha, V_\beta, V_\gamma, V_\delta, V_{\gamma\_abc}]$ = Verification key
(4) **For** (**unit** $i = 0$; $i < \pi.Public\_in.length$; i++)
(5) V_x = **EIP196.**addition($V_{\gamma\_abc}[0]$, **EIP196.**scalar_mul($\pi.Public\_in[i]$, $V_{\gamma\_abc}[i+1]$))
(6) **EndFor**
(7) **IF** (**EIP197.**Pairing ($\pi.A_1$, $\pi.B_1$,
(i) V_α, V_β,
(ii) V_x, V_γ,
(iii) $\pi.C_1$, V_δ)) == 1)
(8) **Return** True
(9) **Else**
(10) **Return** False;
(11) **EndIF**
(12) **End Contract**

Algorithm 2: Verification smart contract, Cert_ZK_Proof.

Table 5: Gas consumption of the four smart contracts for the VC of age.

Contracts	Deployment cost	Execution cost
Cert_Status_SC	413223	70863
cert_isused_sc	348845	73099
Cert_ZK_Proof	1790279	300724
VerifySignature	411558	42615
Total cost	2963905	487301

Table 6: Gas consumption of the four smart contracts for the VC of income.

Contracts	Deployment cost	Execution cost
Cert_Status_SC	413223	70851
cert_isused_sc	348845	73123
Cert_ZK_Proof	1790303	300748
VerifySignature	411558	42639
Total cost	2963929	487361

numbers of users requiring services from the same SP. From Figure 12, it can be seen that along with the increase in the number of users requiring services from the same SP, the average response time increases gradually, and when the number of users requiring services from the same SP becomes 100, the average response time reaches 2.91 seconds. It is clear that when the number of users requiring services from the same SP becomes larger, the throughput of the proposed system will not remain high due to the fact that in the proposed system, all operations associated with VC and ZKP are implemented on the smart contracts on the blockchain. As mentioned in Section 3.2, the system based on smart contracts is credible and can avoid single-point failure. Therefore, for the proposed system, it is actually a trade-off between throughput and security.

7.3. Safety Analysis. The analysis of the security of the proposed system consists of two parts which are the security of the bottom layer and the security of application layer, respectively:

The security of the bottom layer:

(1) The security of system: In this paper, the proposed system is based on the VC of blockchain, and for VC, nothing related to its plaintext is stored in the blockchain. In the blockchain, just some smart contracts are used to store the hash digests of VC and ZKP and to verify the ZKP and the digital signature of IDP. Furthermore, for VC, the holder of its private key must be the holder of ZKP.

(2) The security of ZKP: In this paper, the algorithm of ZKP, Groth16, is adopted and implemented in the tool kit, Zokrates. For Groth16, it meets the three characteristics of ZKP, namely, completeness, reliability, and zero knowledge [39], and although the malleability attack exists in Groth16, this paper considers the attack (the Appendix).

(3) The security of verification: In this paper, all verifications are implemented in the smart contracts of blockchain to avoid the single point failure of the proposed architecture.

The security of the application layer:

(1) The minimum disclosure of privacy of VC: As shown in Table 1, the VC of this paper is defined as the format containing only a single identity attribute and for the user who wants to obtain different services, multiple VCs with different identity attributes should be used.

(2) Privacy protection of identity and behavior: this paper uses ZKP to hide attribute and unique identity identifier of user' VC thus implementing the privacy protection of identity and behavior.

7.4. Comparisons with Existing Systems. As shown in Table 7, in comparison with the traditional systems described in Section 2 such as traditional distributed ledger, smart contract and VC, the proposed system improves the privacy of user through the algorithm, Groth16, of zkSNARKs. On the other hand, in comparison with other systems which adopt Groth16, the proposed system considers the malleability attack of Groth16. Furthermore, for the proposed system, all operations associated with verification are implemented on the blockchain through smart contracts, thus guaranteeing the transparency of verification while avoiding single point failure. Finally, in addition to the advantages, as shown in Table 7, the proposed system guarantees that the holder of ZKP must be the holder of private key of VC, which is absent in the system presented in [30].

In addition, from Figure 12, as can be seen that when the number of users requiring services from the same SP is 1, the average response time of the proposed system is 0.113 seconds, which is close to the response time of the scheme of anonymous certificate presented in [36] (its response time over Ethereum blockchain is 0.12 seconds [34]). Therefore, the proposed system not only solves the problems which exist in schemes of zkSNARKs but also has similar performance with the scheme of anonymous certificate based on Ethereum blockchain.

8. Summary and Outlook

While playing an important role in computer and cyberspace security, the traditional digital identity verification and management system (DIVMS) currently has many problems such as centralization in storage and management of identity information, single point failure, and poor self-sovereignty management on user's identity, nontransparent verification and authorization of identity, and so on. In recent years, many based-blockchain DIVMSs are proposed with the development of blockchain and the progress of DIVMS. In the proposed blockchain-based DIVMSs, the DIVMS of blockchain-based VC has many advantages in self-sovereignty identity management and scalability of blockchain. However, traditional DIVMSs of the blockchain-based VC lack privacy protection. To resolve the problem of privacy

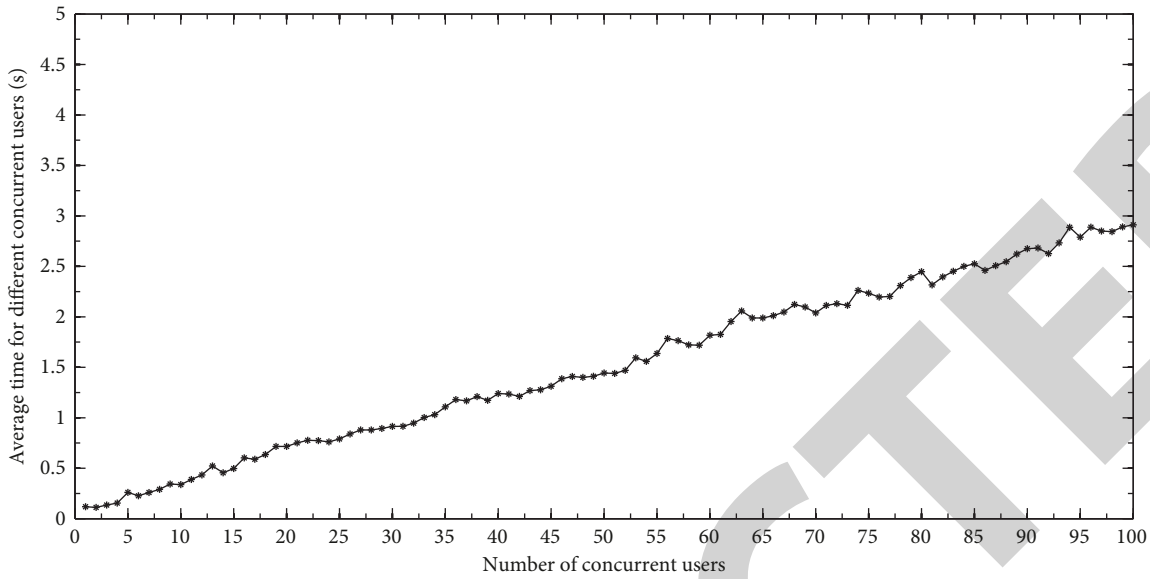

Figure 12: Test result for the throughput of system.

Table 7: Comparisons of the partial architectures described in Section 2.

System	Architecture	The scheme of privacy protection	On-chain verification based smart contract	Privacy protection of identity and behavior	Considering malleability attack of Groth16
Certcoin [10]	Distributed ledger	×	×	×	×
BlockCAM [13]	Distributed ledger	×	×	×	×
UPort [15]	Smart contract	×	×	×	×
EverSSDI [17]	Smart contract	×	×	×	×
SCPKI [18]	Smart contract	×	×	×	×
TBDID [23]	Traditional VC	×	×	×	×
Paper [24]	Traditional VC	×	√	×	×
CloudDID [25]	Traditional VC	×	×	×	×
Sovrin [33]	VC with privacy protection	Anonymous certificate	×	√	×
Paper [36]	VC with privacy protection	Anonymous certificate	√	√	×
Paper [27]	VC with privacy protection	zkSNARKs	×	×	×
Paper [30]	VC with privacy protection	zkSNARKs	√	×	×
Paper [38]	VC with privacy protection	zkSNARKs	√	√	×
This paper	VC with privacy protection	zkSNARKs	√	√	√

protection, scholars have proposed two main schemes: anonymous certificates and zkSNARKs. However, in comparison with the scheme of an anonymous certificate, the publicly available schemes of zkSNARKs based on the algorithm, Groth16, still have problems in defending against the malleability attack, single point failure avoiding, and so on. Therefore, this paper proposes a modified scheme based on zkSNARKs in order to solve the problems. In the proposed scheme, the behavior privacy of the user is protected, and the malleability attack of the algorithm, Groth16, of zkSNARKs is considered. In the meantime, all operations associated with verification and management of VC is implemented on the smart contract of blockchain to avoid single point failure. Finally, a prototype system of the proposed scheme is designed to verify the capability of the proposed system in privacy protection and evaluate its performances in cost and throughput. All results show that the proposed scheme is significant.

Although the proposed system is reasonable, limitations still exist in user identity revocation management.

This is mainly because, in this paper, the revocation management is realized through the smart contract shown in Figure 7, Cert_Status_SC. However, the revocation information of the user's identity stored in the smart contract can only be added and not deleted as a result of the tamper-proof feature of the smart contract. Therefore, as the number of users increase, the storage space that the smart contract takes up will gradually increase. The cryptographic accumulator is considered as an effective scheme of revocation management and has been used in some systems of the anonymous certificates [33, 35] to implement selective revocation of the user's identity on the premise of saving storage space. In particular, combining a cryptographic accumulator with non-interactive ZKP can implement selective revocation of user identity without leaking any information about the identity and behavior of the user [35]. Therefore, in the future, we will consider replacing the current smart contract-based method for revocation management of user identity with the combination method mentioned above to realize efficient and storage-friendly revocation of user identity.

Appendix

As described in Section 3.3, the verification of Groth16 can be implemented through the bilinear pairing presented in (6). Furthermore, (6) is equivalent to the following equation:

$$A_1 \cdot B_1 - V_\alpha \cdot V_\beta - \frac{\sum_{i=0}^{l} z_i \left(\beta a_i(x) + \alpha b_i(x) + c_i(x)\right) G}{\gamma} V_\gamma - C_1 V_\delta = 0, \tag{A.1}$$

where $A_1 = AG$, $B_1 = BH$, $V_\alpha = \alpha G$, $V_\beta = \beta H$, $V_\gamma = \gamma H$, $V_\delta = \delta H$, and $C_1 = CG$. Verification key$\{V_\alpha, V_\beta, V_\gamma, V_\delta, \sum_{i=0}^{l} (\beta a_i(x) + \alpha b_i(x) + c_i(x)))G/\gamma\}$ is from the verifier (SP), and $\pi = \{A_1, B_1, C_1, \text{Public Inputs} = (z_1, z_2, \ldots, z_l)\}$ is from the prover (User).

The malleability attack of Groth16 is that an attacker seeing a valid $\pi = \{A_1, B_1, C_1, \text{Public Inputs} = (z_1, z_2, \ldots, z_l)\}$ can very easily generate a different but still valid π'. In other words, an attacker can pretend to be a real user passing the verification of (6). In the following, the attack process will be described.

An attacker seeing a valid $\pi = \{A_1, B_1, C_1, \text{Public Inputs} = (z_1, z_2, \ldots, z_l)\}$ can select a random number, $k > 1$, from the finite file F, and then generates a new $\pi' = \{A_1' = kA_1, \ B_1' = k^{-1}B_1, C_1' = C_1, \text{Public Inputs} = (z_1, z_2, \ldots, z_l)\}$. For π', it still can pass the verification of equation (A.1) as a result of $A_1' \cdot B_1' = A_1 B_1$.

On the other hand, π' also can be expressed as $\pi\prime = \{A_1' = A_1, B_1' = B_1 + \eta V_\delta, C_1' = C_1 + \eta A_1, \text{Public Inputs} = (z_1, z_2, \ldots, z_l)\}$ where η is a random number, $\eta \geq 1$, from the finite file F. At this time, we substitute A_1, B_1, C_1 of equation (A.1) into A_1', B_1', C_1', and express equation (A.1) as follows:

$$A_1 \cdot \left(B_1 + \eta V_\delta\right) - V_\alpha \cdot V_\beta - \frac{\sum_{i=0}^{l} z_i \left(\beta a_i(x) + \alpha b_i(x) + c_i(x)\right) G}{\gamma} V_\gamma - \left(C_1 + \eta A_1\right) V_\delta = 0. \tag{A.2}$$

Obviously, equation (A.2) is still equivalent to equation (A.1).

Finally, from $\pi = \{A_1, B_1, C_1, \text{Public Inputs} = (z_1, z_2, \ldots, z_l)\}$ and $\pi' = \{A_1', B_1', C_1', \text{Public Inputs} = (z_1, z_2, \ldots, z_l)\}$, it can be seen that the two Public Inputs of π and π' are constant. Therefore, in this paper, the hash digest of Public Inputs (The item, Hash_P, shown in Table 2 of this paper) will be used to indicate that the real π has been used, which can avoid the malleability attack mentioned above.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 71964037), Yunnan Key Laboratory of Blockchain Application Technology (No. 202105AG070005, YNB202108), Yunnan Provincial Key Laboratory of Forensic Science (No. YJXK005), Scientific Research Foundation of Yunnan Education Department (No. 2022J0473), Scientific Research Foundation of Yunnan University of Finance and Economics (No. 20220002), Yunnan Basic Research Program (No. 202101AU070132),

Research on Key Technologies of Cross-Border Trade Blockchain for RCEP (No. 202202AD080011), Yunnan Cross-Border Trade and Financial Blockchain International Joint Research and Development Center (No. 202203AP140010), and Yunnan Key Laboratory of Smart City and Cyberspace Security (No. 202105AG070010-SG-07).

References

[1] D. Ingram, *Facebook says data leak hits 87 million users, widening privacy scandal*, Reuters, London, U.K, 2018.

[2] A. Hashim, "China's Huazhu Group data breach exposed 500 million customer records," 2018, https://latesthackingnews.com/2018/09/03/chinas-huazhu-group-data-breach-exposed-500-million-customer-records/.

[3] D. Wang, N. Wang, P. Wang, and S. Qing, "Preserving privacy for free: efficient and provably secure two-factor authentication scheme with user anonymity," *Information Sciences*, vol. 321, pp. 162–178, 2015.

[4] A. B. Sideridis, L. Protopappas, S. Tsiafoulis, and E. Pimenidis, "Smart cross-border e-Gov systems and applications," in *Proceedings of the International Conference on e-Democracy*, pp. 151–165, Springer, Cham, Berlin, Germany, December 2015.

[5] D. Recordon and D. Reed, "OpenID 2.0: a platform for user-centric identity management," in *Proceedings of the second ACM workshop on Digital identity management*, pp. 11–16, Alexandria, VA, USA, November 2006.

[6] R. D. Dhungana, A. Mohammad, A. Sharma, and I. Schoen, "Identity management framework for cloud networking infrastructure," in *Proceedings of the 2013 9th International Conference on Innovations in Information Technology (IIT)*, pp. 13–17, IEEE, Al Ain, UAE, March 2013.

[7] E. Y. Chen, Y. Pei, S. Chen, and P. Tague, "Oauth demystified for mobile application developers," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 892–903, Scottsdale, AZ, USA, November 2014.

[8] D. Wang and P. Wang, "Two birds with one stone: two-factor Authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, 2016.

[9] Q. Jiang, N. Zhang, J. Ni, J. Ma, X. Ma, and K. K. R Choo, "Unified biometric privacy preserving three-factor Authentication and key agreement for cloud-assisted autonomous vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 9390–9401, 2020.

[10] C. Fromknecht, D. Velicanu, and S. Yakoubov, "CertCoin: A Namecoin Based Decentralized Authentication System 6.857 Class Project 2014," 2021, http://courses.csail.mit.edu.

[11] Z. Wang, J. Lin, Q. Cai, Q. Wang, D. Zha, and J. Jing, "Blockchain-based certificate transparency and revocation transparency," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 681–697, 2022.

[12] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: beyond bitcoin," *Applied Innovation*, vol. 2, no. 6-10, p. 71, 2016.

[13] W. Wang, N. Hu, and X. Liu, "BlockCAM: a blockchain-based cross-domain authentication model," in *Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 896–901, IEEE, China, June 2018.

[14] S. Haddouti and M. D. E. C. Kettani, "Analysis of identity management systems using blockchain technology," in *Proceedings of the 2019 International Conference on Advanced Communication Technologies and Networking (CommNet)*, pp. 1–7, IEEE, Morocco, April 2019.

[15] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena, "Uport: A Platform for Self-Sovereign Identity," 2016, https://blockchainlab.com/pdf/uPort_whitepaper_DRAFT 20161020.pdf.

[16] Z. Diebold, *Self-Sovereign Identity Using Smart Contracts on the Ethereum Blockchain. Master in Computer Science*, University of Dublin, Trinity College, 2017.

[17] H. Zhao, T. Zhou, and X. Li, "EverSSDI: blockchain-based framework for verification, authorisation and recovery of self-sovereign identity using smart contracts," *International Journal of Computer Applications in Technology*, vol. 60, no. 3, pp. 281–295, 2019.

[18] M. A. Bassam, "SCPKI: a smart contract-based PKI and identity system," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pp. 35–40, Abu Dhabi, UAE, April 2017.

[19] P. Kamboj, S. Khare, and S. Pal, "User authentication using Blockchain based smart contract in role-based access control," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2961–2976, 2021.

[20] S. Manu, L. Dave, and C. David, "Verifiable Certificates Data Model 1.0 - Expressing Verifiable Information on the Web," 2019, https://www.w3.org/TR/vc-data-model.

[21] X. Fan, Q. Chai, L. Xu, and Q. Guo, "DIAM-IoT: a decentralized identity and access management framework for internet of things," in *in Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, pp. 186–191, October 2020.

[22] Y. Kortesniemi, D. Lagutin, T. Elo, and N. Fotiou, "Improving the privacy of iot with decentralised identifiers (dids)," *Journal of Computer Networks and Communications*, vol. 2019, Article ID 8706760, 10 pages, 2019.

[23] M. Aydar, S. Ayvaz, and S. C. Cetin, "Towards a Blockchain Based Digital Identity Verification, Record Attestation and Record Sharing System," 2019, https://arxiv.org/abs/1906.09791.

[24] F. Wang and P. De Filippi, "Self-Sovereign identity in a globalized World: credentials-based identity systems as a driver for economic inclusion," *Frontiers in Blockchain*, vol. 2, p. 28, 2020.

[25] Baidu, "Baidu Cloud DID Method," 2020, http://did.baidu.com/.

[26] Q. Stokkink, G. Ishmaev, D. Epema, and J. Pouwelse, "A truly self-sovereign identity system," in *Proceedings of the 2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pp. 1–8, IEEE, Edmonton, Canada, October 2021.

[27] J. Lee, J. Choi, H. Oh, and J. Kim, "Privacy-preserving identity management system," *Cryptology ePrint Archive*, 2021.

[28] P. Voigt and A. V. D. Bussche, "The eu general data protection regulation (gdpr)," in *A Practical Guide*, 1, Ed., vol. 10, Berlin, Germany, Springer International Publishing, Article ID 3152676, 2017.

[29] Stanford University, "Translation: personal information protection Law of the people's Republic of China – effective nov," *DigiChina Project*, vol. 20, 2021, https://digichina.stanford.edu/work/translation-personal-information-protection-law-of-the-peoples-republic-of-china-effective-nov-1-2021/.

[30] Q. Li and Z. Xue, "A privacy-protecting authorization system based on blockchain and zk-SNARK," in *Proceedings of the 2020 International Conference on Cyberspace Innovation of*

Advanced Technologies, pp. 439–444, Guangzhou, China, December 2020.

[31] J. L. C. Sanchez, J. Bernal Bernabe, and A. F. Skarmeta, "Integration of anonymous credential systems in IoT constrained environments," *IEEE Access*, vol. 6, pp. 4767–4778, 2018.

[32] X. Yang and W. Li, "A zero-knowledge-proof-based digital identity management scheme in blockchain," *Computers & Security*, vol. 99, Article ID 102050, 2020.

[33] D. Khovratovich and J. Law, "Sovrin: digital identities in the blockchain era," *Github Commit by jasonalaw October*, vol. 17, pp. 38–99, 2017.

[34] K. Singh, O. Dib, C. Huyart, and K. Toumi, "A novel credential protocol for protecting personal attributes in blockchain," *Computers & Electrical Engineering*, vol. 83, Article ID 106586, 2020.

[35] Y. Yu, Y. Zhao, Y. Li, X. Du, L. Wang, and M. Guizani, "Blockchain-based anonymous authentication with selective revocation for smart industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3290–3300, 2020.

[36] A. Sonnino, M. A. Bassam, S. Bano, S. Meiklejohn, and G. Danezis, "Coconut: threshold issuance selective disclosure certificates with applications to distributed ledgers," 2018, https://arxiv.org/pdf/1802.07344.pdf.

[37] Y. Cao, J. Chen, and Y. Cao, "Blockchain-based privacy-preserving vaccine passport system," *Security and Communication Networks*, vol. 2022, Article ID 4769187, 16 pages, 2022.

[38] G. Ra, T. Kim, and I. Lee, "VAIM: verifiable anonymous identity management for human-centric security and privacy in the internet of things," *IEEE Access*, vol. 9, Article ID 75945, 2021.

[39] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proceedings of the Annual international conference on the theory and applications of cryptographic techniques*, pp. 305–326, Springer, Berlin, Germany, April 2016.

[40] E. Samir, H. Wu, M. Azab, C. Xin, and Q Zhang, "DT-SSIM: a decentralized trustworthy self-sovereign identity management framework," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 7972–7988, 2022.

[41] B. D. Deebak, F. H. Memon, K. Dev, S. A. Khowaja, W. Wang, and N. M. F. Qureshi, "TAB-SAPP: a trust-aware blockchain-based seamless authentication for massive IoT-enabled industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 243–250, 2023.

[42] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[43] E. Androulaki, A. Barger, V. Bortnikov et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15, Porto, Portugal, April 2018.

[44] Z. Cao and L. Zhao, "A design of key distribution mechanism in decentralized digital rights management based on blockchain and zero-knowledge proof," in *Proceedings of the 2021 The 3rd International Conference on Blockchain Technology*, pp. 53–59, Porto, Portugal, April 2021.

[45] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of Cryptology*, vol. 1, no. 2, pp. 77–94, 1988.

[46] L. Lesavre, P. Varin, P. Mell, M. Davidson, and J. Shook, "A taxonomic approach to understanding emerging blockchain identity management systems," 2019, https://arxiv.org/abs/1908.00929.

[47] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: nearly practical verifiable computation," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, pp. 238–252, IEEE, Berkeley, CA, U.S.A, May 2013.

[48] Z. Guan, Z. Wan, Y. Yang, Y. Zhou, and B. Huang, "Blockmaze: an efficient privacy-preserving account-model blockchain based on zk-SNARKs," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1446–1463, 2022.

[49] E. B. Sasson, A. Chiesa, C. Garman et al., "Zerocash: decentralized anonymous payments from bitcoin," in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, pp. 459–474, IEEE, Berkeley, CA, USA, May 2014.

[50] N. Gailly, M. Maller, and A. Nitulescu, "SnarkPack: practical SNARK aggregation," *IACR Cryptol. ePrint Arch.*vol. 2021, p. 529, 2021.

[51] S lab, "Libsnark: A C++ Library for Zksnark Proofs," 2018, https://github.com/scipr-lab/libsnark.

[52] J. Eberhardt and S. Tai, "Zokrates-scalable privacy-preserving off-chain computations," in *Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1084–1091, IEEE, Halifax, Canada, August 2018.

[53] ZoKrates, "G16 Malleability," 2021, https://zokrates.github.io/toolbox/proving_schemes.html#g16-malleability.