Hindawi
Mobile Information Systems
Volume 2021, Article ID 2807395, 15 pages
https://doi.org/10.1155/2021/2807395

Research Article

Designated-Verifier Anonymous Credential for Identity Management in Decentralized Systems

Xudong Deng, Chengliang Tian, Fei Chen, and Hequn Xian

College of Computer Science & Technology, Qingdao University, Qingdao 266071, China

Correspondence should be addressed to Chengliang Tian; tianchengliang@qdu.edu.cn

Received 26 May 2021; Revised 31 July 2021; Accepted 13 August 2021; Published 10 September 2021

Academic Editor: Vishal Sharma

Most of the existing identity management is the centralized architecture that has to validate, certify, and manage identity in a centralized approach by trusted authorities. Decentralized identity is causing widespread public concern because it enables to give back control of identity to clients, and the client then has the ability to control when, where, and with whom they share their credentials. A decentralized solution atop on blockchain will bypass the centralized architecture and address the single point of the failure problem. To our knowledge, blockchain is an inherited pseudonym but it cannot achieve *anonymity* and *auditability* directly. In this paper, we approach the problem of decentralized identity management starting from the *designated-verifier anonymous credential* (DVAC in short). DVAC would assist to build a new practical decentralized identity management with *anonymity* and *auditability*. Apart from the advantages of the conventional anonymous credential, the main advantage of the proposed DVAC atop blockchain is that the issued cryptographic token will be divided into shares at the issue phase and will be combined at the showing credential phase. Further, the smooth projective hash function (*SPHF* in short) is regarded as a designated-verifier zero-knowledge proof system. Thus, we introduce the *SPHF* to achieve the designated verifiability without compromising the privacy of clients. Finally, the security of the proposed DVAC is proved along with theoretical and experimental evaluations.

1. Introduction

Identity management is viewed as a tool for the protection of user identification and account privacy security, government enterprise management, and public service demand, or the security and economic needs of operators and providers. Before, the ID card system is succinct for the government to manage people's identity. With the rapid development of the Internet, a large number of identity management systems appear in our field of vision. They all have their merits for some special demand and also vulnerability for practice at the same time.

Typically, a trusted party certificate authority (CA) is used to manage and store identities for users. As far back as the late twentieth century, "Passport" is a unified identity management project based on the NET platform implemented in [1]. "Passport" provides great convenience to users by allowing them to authenticate with only one site. However, as its system architecture is centralized and coordinated, the problem follows. In practice, single points of failure are coming through a trusted party. Such as the DigiNotar incident [2], CA was held hostage by hackers, in which Google's certificate was published mistakenly. So, we need to effectively store a person's identity information and ensure its privacy and effectiveness on the Internet where threats and vulnerabilities are ubiquitous. And, how to protect the privacy of individuals and ensure their identity is verified without giving away their privacy is an important issue.

Bitcoin and blockchain had been proposed in 2008 [3]. The transaction of virtual currency built on the chain can guarantee the privacy and security of both parties. The natural decentralization and unchangeability of blockchain give us a new direction. With the rapid development of blockchain, there are more and more decentralized systems appearing based on blockchain. Furthermore, IBM and

Samsung have been working on an idea called ADEPT that uses blockchain technology to form a decentralized network of IoT devices. Simply, it is feasible to construct a relatively secure identity management system with blockchain because some security features on blockchain fully meet the requirements of the identity management system.

Recently, blockchain-based identity management has also had limited success, such as DAC [4] and DBLACR [5]. In these systems, users obtain information credentials from an authority (e.g., government) and upload their credentials to the blockchain. When an entity, such as a service provider, has requirements for its customers, users can prove those requirements for verification by the blockchain, which is used as a transparent infrastructure for authentication. Since we want to ensure the privacy of user information, we need to encrypt user information, but how to verify the encrypted information and verify it accurately and effectively is a problem (we certainly cannot provide authentication after decrypting user information, which violates the principle of privacy).

In this paper, we propose DVAC, a decentralized anonymous credential system to protect the privacy of the clients. In particular, interaction in the system is completely anonymous and the registered identity information is self-sovereign. Instead of traditional CA for the whole system level, DVAC employs a decision-making institution committee; the committee consists of an indefinite number of members. The function of the committee is the same as that of a traditional CA, which issues credentials to users, except that it requires the approval of its members when making important decisions. This can be seen as effectively diminishes the power of CA and avoiding the problem of single points of failure. Moreover, DVAC supports the change of membership in the committee. On the contrary, conflicts are inevitable between service providers and users because of the anonymity of users in the anonymous credential system. We need a reasonable and fair audit to protect the interests of both parties in the conflict.

To this end, we introduced proactive secret sharing. We use proactive secret sharing to distribute the private key of the committee to members, and no party in the committee can decide on its own. Moreover, proactive secret sharing can redistribute the secret key periodically according to the system conditions. In this way, members of the committee are prevented from being heavily bribed to ensure the correctness of the committee's decision-making. When a conflict occurs, members of the committee negotiate whether to conduct an audit, but only if the number of supporting reaches a certain threshold.

Contribution. Below, we conclude our main contribution along with the techniques' roadmap as follows:

(i) *A Neat Decentralized Anonymous Credential via Cost-Efficiency SPHF*. We construct a decentralized identity management system and describe each step of our scheme in detail. We use *SPHF* to realize anonymous authentication of the system, and we add an audit function to our system. It points us to a new way of identity management.

(ii) *A Privacy-Preserving* DVAC *Service via Our Designed* DAC. We design an application using our DVAC system for identity management.

(iii) *A Simply Prototype of* DVAC. We implement and evaluate our system and test its performance under different lengths of the secret key.

Organization. The rest of the paper is organized as follows: Section 2 shows the related work of DVAC. Section 3 presents our hardness assumptions and cryptographic building blocks. Our system model and security model are presented in Section 4. We reviewed previous scenarios for anonymous credentials in Section 5. In Section 6, we describe each step of the construction of our solution in detail and we have proved the correctness and security of our scheme. In Section 7, we briefly introduce an application of our scheme. In Section 8, we evaluate the performance of DVAC. Finally, we conclude our work in Section 9.

2. Related Work

DVAC is engaged in anonymous authentication and identity management of private data and privacy protection blockchains. DVAC uses a zero-knowledge proof scheme and proactive secret sharing to create a new decentralized anonymous identity management system, which achieved fast and provable correct queries.

2.1. Anonymous Credential (over Blockchain). In [6], a bilinear pair-based signature scheme was proposed, and based on the signature scheme, they constructed an efficient anonymous credential system. Au et al. proposed a constant-size anonymous authentication scheme [7], which use short group signature in [8]. The scheme can achieve multiple dynamic authentications, and the signature certificate length is constant, which makes the scheme have better efficiency. Additionally, Begum et al. [9] proposed another pairing-based anonymous credential system that also satisfies the constant size of the formula proof. In 2010, IBM proposed "Identity Mixer" [10], which can be used for anonymous authentication and the transfer of authentication attributes. It allows users to authenticate without collecting any other personal data. But the "Identity Mixer" has a defect, it does not provide identity tracking. In 2020, Li et al. [11] proposed a round-optimal asymmetric PAKE protocol, which could construct a new anonymous credential system. "DAC" [4] and "DBLACR" [5] are two decentralized anonymous credential systems based on blockchain. The anonymity of blockchain ensures that users' private information will not be disclosed. However, there must have a trusted party to verify the reasonableness of the user's identity information. There is still a risk of single points of failure.

2.2. Identity Management (over Blockchain). "Liberty Alliance" proposed a three-way interaction protocol based on users, identity provider, and service provider [12]. It has improved the issue of leakage of users' private information

based on "Passport." Subsequently, there comes out many identity management systems such as Tivoli Access Manager [13] and Central Authentication Service [14]. These systems provide a more secure privacy protection protocol. At the same time, these systems provide more complete basic functions and expand the application scope. But with the rapid popularization of electronic identity and the increasing demand of people, the centralized management scheme has become the most fundamental drawback [15]. CertCoin [16] was proposed by Conner Fromknecht et al. It constructs a distributed authentication system and maintains a common ledger for storing user IDs and public key information by using blockchain. After that, Blockstack decentralized PKI system [17] was proposed by Ali et al. Blockstack uses bitcoin's working proof mechanism to maintain the system's state consistency. In Coconut [18], Sonnino et al. proposed a new signature scheme based on Waters signature scheme, BGLS signature [19], and signature scheme of Pointcheval and Sanders [20]. Coconut enables general-purpose selective disclosure credentials to be efficiently used in settings with no natural single trusted third party to issue them. In addition, because of Shamir's secret sharing [21], it will waste more time when adding and removing authorities. After Coconut, Nym credentials [22] were proposed by Halpin et al., namely, Nym credentials can be viewed as an improvement or extension of Coconut.

2.3. Public Distributed Ledger (atop Blockchain). In ZkLedeger [23], Narula et al. proposed the first privacy-protected, verifiable audit distributed ledger system. All information about the transaction is uploaded to a public ledger and publicly verifiable. Unlike zk-SNARKs, Zkledger provides efficient and fast audits through noninteractive zero-knowledge proof and it does not require a trusted setup. Furthermore, ZkLedger provides much different auditing queries and real-time feedback to the auditor. In PrETP [24], Balasch et al. proposed a new way in privacy-protection ETP system. It can protect the user's privacy to the greatest extent and provide the correct audit. PrETP resolves the conflict between the user's right to privacy and the service provider's right of interest. After PrETP, Meiklejohn et al. proposed Milo [25]. Milo solves the problem of privacy leakage in the audit process in PrETP and provides a new way in preventing drivers from ganging up to cheat.

3. Preliminaries

Below, we will outline the cryptographic building blocks that will be used in the following sections.

Notation. Throughout this paper, let λ be the security parameter; then, we denote the vector (i.e., a) and the matrix using the lower letter and upper letter, respectively. In addition, let $\mathbb{G}$ be a group of prime order p with generator g.

Decisional Diffie–Hellman (DDH) Assumption is that given a group $G = \langle g \rangle$ of order q, distribution $(g^a, g^b, g^c | a, b, c \xleftarrow{\$} \mathbb{Z}_q)$, and $(g^a, g^b, g^{ab} | a, b \xleftarrow{\$} \mathbb{Z}_q)$ are indistinguishable for any polynomial time adversary.

Decisional Bilinear Diffie–Hellman (DBDH) Assumption is that no polynomial time algorithm can achieve nonnegligible advantage in deciding the BDH problem; in other words, no adversary has the ability to distinguish the distribution $(g, g^a, g^b, g^c, e(g,g)^{abc})$ from $(g, g^a, g^b, g^c, e(g,g)^z)$.

Smooth Projective Hash Function (SPHF) was proposed by Cramer and Shoup [26]. SPHF gives us a method to achieve zero-knowledge proof for the specified verifier [27–29]. For a NP language L, a word $x \in L$ and a complete SPHF are defined as follows:

(i) $\mathrm{hk} \longleftarrow SPHF.HashKG(L)$: takes a language L as input and generates a hashing key hk.
(ii) $\mathrm{hp} \longleftarrow SPHF.ProjKG(\mathrm{hk}, L, x)$: for a word $x \in L$, use hk and L as inputs, and output the projective hashing key hp.
(iii) $H \longleftarrow SPHF.Hash(\mathrm{hk}, L, x)$: the algorithm's inputs are same as $SPHF.ProjKG$; output a value H.
(iv) $H' \longleftarrow SPHF.Proj(\mathrm{hp}, L, x; w)$: w is a witness for that $x \in L$. This algorithm uses $(\mathrm{hp}, L, x; w)$ as inputs and outputs a value H'.
The SPHF contains two properties, one is *smoothness* and another *projective*.
(v) *Projective(Correctness)*. For all $x \in L$ and its witness w, satify $SPHF.Hash(\mathrm{hk}, L, x) = SPHF.Proj(\mathrm{hp}, L, x; w)$.
(vi) *Smoothness*. For any $x \notin L$ and any parameters, the following distributions are statistically indistinguishable:

$\{(\mathrm{hp}, H)$: $\mathrm{hk} \longleftarrow SPHF.HashKG(L), \mathrm{hp} \longleftarrow SPHF.ProjKG(\mathrm{hk}, L, x), H \longleftarrow SPHF.Hash(\mathrm{hk}, L, x)\}$,
$\{(\mathrm{hp}, H)$: $\mathrm{hk} \longleftarrow SPHF.HashKG(L), \mathrm{hp} \longleftarrow SPHF.ProjKG(\mathrm{hk}, L, x), H \longleftarrow^{\$} \Theta\}$,where Θ is the set of hash values.

3.1. Pedersen Commitment. Let $h \in \mathbb{G}$ is a generator of the group $\mathbb{G}$, and we denote $x \in \mathbb{G}$ as the message. Randomly select a commitment parameter $r \in \mathbb{G}$. The commitment scheme is proceeded as follows:

(i) $c \longleftarrow Com(x, r)$ computes and outputs $c = g^x h^r$
(ii) $0/1 \longleftarrow Open(c, x, r)$

Pedersen commitments contain two important properties, one is perfectly hiding: the commitment **c** reveals nothing about the committed value x. Another is computationally binding: an adversary of *probabilistic polynomial time* cannot find a value x' for r such that $c = g^{x'} h^r$.

Linear Encryption (LE in short) is based on *Decision Linear problem* [8]. Below, we review the original scheme proposed by Boneh et al. [8]. For the public common parameters of LE scheme (G, G_1, p, g, g_1, e), the detailed construction is as follows:

(i) $(\mathrm{lsk}, \mathrm{lpk}) \longleftarrow LE.KGen(1^\lambda)$: lsk is the private key; lpk is the public key. We select $(x_1, x_2) \longleftarrow \mathbb{Z}_p$ randomly, and let $\mathrm{lsk} = (x_1, x_2)$ and $\mathrm{lpk} = (Y_1 = g^{x_1}, Y_2 = g^{x_2})$.

(ii) $c \longleftarrow LE.Enc(m; \text{lpk})$: $m \in G$ is a message, $r = (r_1, r_2) \xleftarrow{\$} Z_p$ are random scalars, we use public key lpk, m, and r_1 and r_2 as input and output a ciphertext $c = (c_1 = Y_2^{r_1}, c_2 = Y_2^{r_2}, c_3 = g^{r_1+r_2} \cdot m)$.

(iii) $m \longleftarrow LE.Dec(c; \text{lsk})$: we use ciphertext c and private key lsk as input and output plaintext $m = c_3/(c_1^{x_1^{-1}} c_2^{x_2^{-1}})$.

Waters Signature is an efficient signature scheme with some optimizations and follow-up works [30, 31]. In our solution, we only use the original one to assist DVAC. We define a generator $h \xleftarrow{\$} G$ and a vector $\mathbf{u} = (u_1, u_2, \ldots, u_k) \xleftarrow{\$} G^{k+1}$, which defines the *Waters hash* of a message $m = (m_1, m_2, \ldots, m_k) \in (0,1)^k$ as $F(m) = u_0 \prod_{i=1}^{k} u_i^{m_i}$. Below, we review the construction of Waters signature:

(i) $(\text{wsk}, \text{wpk}) \longleftarrow Waters.KGen(1^\lambda)$: wsk is the private key used for signing, and wpk is the public key used for public verification. wsk $= h^z$ and wpk $= g^z$ where $z \xleftarrow{\$} Z_p$.

(ii) $\sigma \longleftarrow Waters.Sign(m; \text{wsk})$: we use a message m, private key wsk, and a random $s \xleftarrow{\$} Z_p$ as inputs and generate a signature $\sigma = (\sigma_1 = h^z \cdot F(m)^s, \sigma_2 = g^s)$ of m.

(iii) $(0,1) \longleftarrow Waters.Verify(m, \sigma; \text{wpk})$: we use wpk, message m, and its signature σ as inputs. The verify algorithm will output a result that showed whether the signature $e(g, \sigma) = e(g^z, h) \cdot e(F(m), \sigma_2)$ is valid.

Decentralized Anonymous Credential (DAC in short) inherits the properties of anonymous credential [32, 33] except distributing the cryptographic token into a couple of token shares. Most of state-of-art of DAC are considering to instantiate in a decentralized way. Below, as shown in Figure 1, we adopt the definition of DAC discussed by Garman et al. [4] that contains seven PPT algorithms.

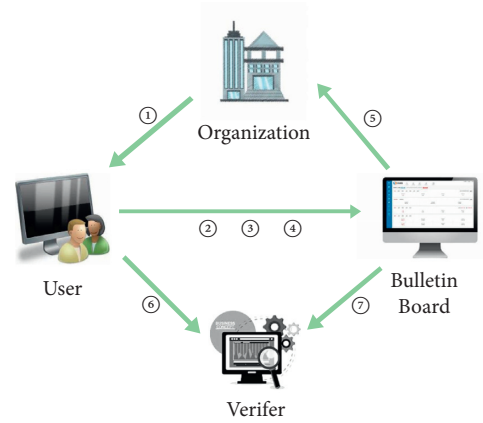

Figure 1: A brief illustration of DAC. ① user gets essential parameters of the system; ②③④ are executed offline by the user on the local side and finally uploads resulting values to the bulletin board; ⑤ the nodes of the organization verify the information from the bulletin board; ⑥ the user plays the role of a prover, and he sends proof of his credential with other auxiliary information to the verifier; ⑦ the verifier downloads the essential information from the bulletin board and validates the proof sent from the prover.

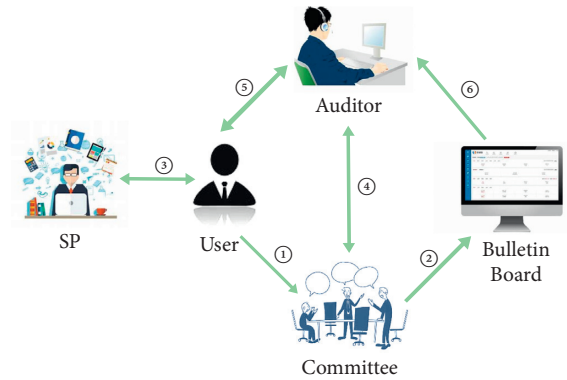

Figure 2: System model of DVAC. ① User send his information to committee. ② The committee upload user's information to blockchain and issue a credential to the user after verifying the information. ③ User shows his credential to the SP and acquires service. ④ The auditor requests the audit process by interacting with the committee. ⑤ The auditor interacts with the user to get the required information. ⑥ The auditor gets the information uploaded by the user at ① from the blockchain and returns the audit result.

4. System and Threat Models

4.1. DVAC System Model Overview

4.1.1. System Participants. During our whole scheme, we separate five entities including user, committee, bulletin board (BB), service provider (SP), and auditor as illustrated in Figure 2 and optimized system model in distribution way in Figure 1.

User is required to register their information on the bulletin board and get a credential from the committee.

Committee is a group of members who can perform the same function as a certificate authority (CA) only under certain conditions.

Bulletin board is a distributed ledger that can be instantiated by the blockchain. The data stored on the BB pseudonymously are public and unchangeable, but only the client who has the secret key could read the data.

Service provider is an organization or government that provides concrete services to clients, and SP determines whether the client has the qualification to access his services.

Auditor is an entity used to audit user information on the blockchain, which is typically acted by some of the members of the committee.

Figure 2 illustrates an example of the model of our scheme. The concept of this paper is that remove the single points of failure of CA and guarantee the privacy information of users during the certification process. We propose a decentralized self-sovereign credential management system. We use secret sharing, and blockchain provides the decentralized function and the immutability of data, and the

SPHF scheme guarantees zero-knowledge, and the blockchain could provide an environment which ensures anonymity. This means the security of users' private information is more strongly guaranteed when they obtain certifications. For simplicity, we have omitted the parameters of the entire system, and our approach proceeds as follows:

4.1.2. Initialization. First of all, the dealer generates system parameters and a pair of keys for the committee. The committee's public key is open for everyone and uses a secret share scheme to distribute the committee's secret key for people in this group. Meanwhile, the committee should set a threshold value for recovering the secret key. The committee can then perform the corresponding function (audit) only if the person who has reached this threshold agrees.

4.1.3. Registration. In registration, the user will get his unique id on the blockchain, which cannot be changed. A new client needs to register his identity information $info$ and his attribute $attr$ to the blockchain. He generates a commitment $\mathbf{c}$ for his identity information info and sends $(\mathrm{id}, \mathbf{c}, attr)$ to committee after signing by his secret key. The committee will first verify the validity of $(\mathrm{id}, \mathbf{c}, attr)$ after receiving it. If the verification passes, the committee will generate credentials for the user's attributions $attrs$ and upload the client's cryptographic information and attributions to the bulletin board.

4.1.4. Interactive Verification. After a client gets his credential, he is a legitimate user of the blockchain. He could interact with an SP or any other users as a prover. The process of interaction is anonymous; each presentation of a credential is anonymized. The prover proves that his identity info satisfies some requirement which comes from verifier through SPHF. Prover and verifier get the required parameters for proof and verification through an interaction. A verifier could verify the correctness of the user's credentials.

4.1.5. Secret Refresh. In our system, the group of the committee is not fixed and we should keep the number of members in a stable state. There must have members quit or join. For example, members of the committee may be bribed or some user wants to be a member of this committee. Although a few bribes do not affect the overall situation, the committee must regularly check and weed out those who have been bribed. So, we need to satisfy (1) security: let the sharing in the hands of the person weed out to be invalid, that is, he can no longer participate in the decision-making and his share cannot use to recover the secret key of the committee. (2) Fairness: provide regular rights of membership for new members.

To ensure the security and fairness of the system, we must refresh the secret key held by the members of the committee. After the committee's secret key had been shared, the $Refresh$ algorithm will be recalled after some time (periodic testing by the committee) or some special change of participants. We will describe the details in Section 6.

4.1.6. Audit. Consider that the user can provide attributes that he does not have to get the committee to sign or the member of the committee falsified. If an SP doubts the truth of a user's credential, SP could apply an audit for the user's identity. When the member in the committee who has reached the threshold agrees, the committee will generate an audit request for the client. This request will ask the user to open the commitment of his identity and compare it with the credential issued by the committee. If the user agreed, he should open the commitment of his information. At the end of the audit, the auditor returns a list of dishonest users and the committee will cancel these users' id on the blockchain. If he refused, the committee will adopt other means of restriction for him.

4.2. Threat Model. In DVAC, it is assumed that the members of the committee will be bribed. We assume that no more than a third of the members will be malicious in each period of a secret refresh. In terms of privacy, we need to protect against a malicious SP speculate the identity of a user in the course of interacting with the user, even if SPs arbitrarily collude. We assume that there are dishonest users who try to trick SP into providing services without the relevant credentials. In DVAC, auditing is only used as a solution for resolving conflicts when they occur. There must have been information leaked to the auditor by an audit process. Users' privacy might be hard to protect if frequent audits are carried out.

4.3. System Goals. As an identity management system, DVAC should achieve the following goals:

(i) *Completeness.* A prover who shows his credentials correctly will surely pass the verification of the verifier.

(ii) *Anonymity.* Private information of each user can only be read by oneself. It means that a verifier does not know the prover's other information except the validity of the credential.

(iii) *Unforgeability.* Prover can generate a valid verifier convinced proof if and only if it has a valid credential and the information contained in the certificate meets the security policy.

(iv) *Unlinkability.* In any two credential showing processes, or in the credential issuance process and credential showing process, an adversary verifier has only one negligible advantage linking them.

(v) *Decentralized Auditability* (D-Auditability). If there is a dispute between SP and user in the process of interaction between the two parties, third parties will intervene and audit. But third parties must reach a consensus (majority rule) to avoid a single point of failure on the part of the auditor.

5. Neat Decentralized Anonymous Credential from SPHF

5.1. Review Decentralized Anonymous Credential. Below, we recall the generic construction of decentralized anonymous credential (DAC) that follows the solution of Garman et al. [4]. In a nutshell, they adopt the following:

(i) $param_{\mathrm{DAC}} \longleftarrow Setup(1^{\lambda})$ takes as input the security parameter λ and then proceeds the following computations:

(1) $param_{\mathrm{ACC}} \longleftarrow AccSetup(1^{\lambda})$, where $param_{\mathrm{ACC}} = (N, u, p, q, g_0, \ldots, g_n)$ are parameters of RSA, p and q are primes such that $p = 2^w q + 1$ for $w \geq 1$, and $g_0, \ldots, g_n$ are random generators of a group $\mathbb{G}$

(2) Output $param_{\mathrm{DAC}} = param_{\mathrm{ACC}}$

(ii) $msk \longleftarrow MskGen(param_{\mathrm{DAC}})$ generates a main secret key msk for the user U, while the msk is the only one that is bound to the user, and the detailed procedures are as follows:

(1) $msk \longleftarrow Z_q$, where msk is randomly selected, and it will be used in the credential mint phase

(2) Output msk

(iii) $(\mathrm{nym}_{\mathrm{U^O}}, msk_{\mathrm{nym}_{\mathrm{U^O}}}) \longleftarrow NymGen(param_{\mathrm{DAC}}, msk, \mathrm{U}, \mathrm{O})$. The pseudonym $\mathrm{nym}_{\mathrm{U^O}}$ (from the user U to the origination O) is generated by U who proceeds under the master secret key msk as follows:

(1) $msk_{\mathrm{nym}_{\mathrm{U^O}}} \longleftarrow NymMskGen(1^{\lambda})$, picks up a randomness r_{msk} from Z_q, and sets $msk_{\mathrm{nym}_{\mathrm{U^O}}} = r_{msk}$

(2) $\mathrm{nym}_{\mathrm{U^O}} \longleftarrow BuildNym(param_{\mathrm{DAC}}, msk_{\mathrm{nym}_{\mathrm{U^O}}, msk})$ and computes $\mathrm{nym}_{\mathrm{U^O}} = g_0^{r_{msk}} g_1^{msk}$ for an organization O

(3) Output $(\mathrm{nym}_{\mathrm{U^O}}, msk_{\mathrm{nym}_{\mathrm{U^O}}})$

(iv) $(\sigma, sk_{\mathrm{U}}, \pi_{\mathrm{U}}) \longleftarrow CredMint(msk, \mathrm{nym}_{\mathrm{U^O}}, msk_{\mathrm{nym}_{\mathrm{U^O}}, \mathrm{att}, \mathrm{aux}})$ is executed by the user U to generate a credential σ which is essential to a Pedersen commitment for his corresponding attributes $\mathrm{att} = (a_0, a_1, \ldots, a_m) \in Z_q$, i.e., $\overline{\mathrm{c}} = g_0^{r_{\mathrm{ped}}} g_1^{msk} \prod_{i=0}^{m} g_{i+2}^{a_i}$ for its randomness $r_{\mathrm{ped}} \in Z_q$ and msk, along with a secret key sk_{U} for the user U and a zero-knowledge proof

(1) $\overline{\mathrm{c}} \longleftarrow Ped.Com(param, msk, \mathrm{att})$, and it takes as input a selected random number $r_{\mathrm{ped}} \in Z_q$ and sets $\mathrm{sk}_{\mathrm{U}} = r_{\mathrm{ped}}$; then, the user calculates

$$\sigma := \overline{\mathrm{c}} = g_0^{r_{\mathrm{ped}}} g_1^{msk} \prod_{i=0}^{m} g_{i+2}^{a_i}. \tag{1}$$

(2) $\pi_{\mathrm{U}} \longleftarrow ZK.Prove(msk, r_{msk}, r_{\mathrm{ped}}, \mathrm{aux})$, and the user proves that

the commitment $\overline{\mathrm{c}}$ and the pseudonym $\mathrm{nym}_{\mathrm{U^O}}$ contain the same master secret key msk and the attributes are in some allowed set

(3) Output $(\overline{\mathrm{c}}, \mathrm{sk}_{\mathrm{U}}, \pi_{\mathrm{U}})$

After generating a credential of attributes, user submits $(\overline{\mathrm{c}}, \pi_{\mathrm{U}}, \mathrm{att}, \mathrm{nym}_{\mathrm{U^O}})$ with auxiliary data aux to blockchain.

(i) $\{0, 1\} \longleftarrow MintVerify(param_{\mathrm{DAC}}, \mathrm{att}, \sigma, \mathrm{nym}_{\mathrm{U^O}}, \mathrm{aux}, \pi_{\mathrm{U}})$. This algorithm is run by nodes in the organization O. The credential's legality is accepted to the blockchain if and only if this algorithm returns 1.

(1) $\{0, 1\} \longleftarrow ZK.Verify(param_{\mathrm{DAC}}, \mathrm{att}, \sigma, \mathrm{nym}_{\mathrm{U^O}}, \mathrm{aux}, \pi_{\mathrm{U}})$

(2) Output 1 if verification is successful, otherwise 0

If user's credential through verification of organization's nodes, he could show others a proof that he is a legal user and satisfy some attributes without revealing his credential. This process is noninteractive and anonymous.

(ii) $\pi_s \longleftarrow CredShow(msk, {}^{\mathrm{UV}}_{\mathrm{nym}}, msk_{{}^{\mathrm{UV}}_{\mathrm{nym}}}, \sigma, \mathrm{sk}_{\mathrm{U}}, C_O)$. In this phase, the user shows a proof to a verifier, where a verifier is either an organization O or a third verifier.

(1) $msk_{{}^{\mathrm{UV}}_{\mathrm{nym}}} \longleftarrow NymMskGen(1^{\lambda})$ and outputs $msk_{{}^{\mathrm{UV}}_{\mathrm{nym}}} = r_{msk}'$, where $r_{msk}' \in Z_q$

(2) ${}^{\mathrm{UV}}_{\mathrm{nym}} \longleftarrow BuildNym(msk_{{}^{\mathrm{UV}}_{\mathrm{nym}}}, msk_{{}^{\mathrm{UV}}_{\mathrm{nym}}}, msk)$ and outputs ${}^{\mathrm{UV}}_{\mathrm{nym}} = g_0^{r_{msk}'} g_1^{msk}$ for a verifier V

(3) $A = Acc(param)_{\mathrm{ACC}}, C_O$, on inputs $param(N, u)$ is executed by the user U and computes $A = u^{\overline{\mathrm{c}} \cdot \overline{\mathrm{c}}_1 \cdot \overline{\mathrm{c}}_2, \cdots \overline{\mathrm{c}}_n}$ where $C_O = \{\overline{\mathrm{c}}, \overline{\mathrm{c}}_1, \overline{\mathrm{c}}_2, \ldots, \overline{\mathrm{c}}_n\}$ are a set of credentials fetched from the blockchain

(4) $w = WitGen(param, c, C_O)$, on inputs $param(N, u)$; then, the user computes a witness $w = u^{\overline{\mathrm{c}}_1 \cdot \overline{\mathrm{c}}_2, \ldots, \overline{\mathrm{c}}_n}$

(5) $\pi_s \longleftarrow ZK.Prove(msk, w, r_{msk}', \overline{\mathrm{c}}, r_{\mathrm{ped}}, \mathrm{nym}_U^V)$:

$$AccVer(param_{\mathrm{ACC}}, A, \overline{\mathrm{c}}, w) = 1 \wedge \overline{\mathrm{c}} = g_0^{r_{\mathrm{ped}}} g_1^{msk} \prod_{i=0}^{m} g_{i+2}^{a_i} \wedge \mathrm{nym}_U^V = g_0^{r_{msk}'} g_1^{msk}. \tag{2}$$

(6) Outputs a proof π_s

At first, the user should generate a new nym ${}^{\mathrm{UV}}_{\mathrm{nym}}$ and its secret key $msk_{\mathrm{nym}_U^V}$ for this verifier. Then, the user calculates a proof that

(i) He knows a credential on the blockchain from organization O

(ii) The credential opens to the same msk pseudonym, and

(iii) It has some attributes

At the end of this phase, the prover sends π_s to verifier through his nym nym_U^V.

(i) $\{0, 1\} \longleftarrow ShowVerify(param_{\mathrm{DAC}}, \mathrm{nym}_U^V, \pi_s, C_O)$. Upon receiving the proof π_s from ${}^{\mathrm{UV}}_{\mathrm{nym}}$, the verifier executes the verify algorithm.

(1) $A = Acc(param)_{\mathrm{ACC}}, C_O$, and verifier computes $A = u^{\bar{c}\cdot\bar{c}_1\cdot\bar{c}_2,\ldots,\bar{c}_n}$
(2) $\{0,1\} \longleftarrow ZK.Verify(param_{\mathrm{DAC}}, \mathrm{nym}_U^V, \pi_s, A,)$, and verify that π_s is the aforementioned proof of knowledge on $(\bar{c}, C_O)$ and nym_U^V using the known public values
(3) Output 1 if verification is successful, otherwise 0

5.2. Review (Interactive) Anonymous Credential via SPHF. Below, we recall the generic construction of *SPHF*-based anonymous credential that follows the solution of Blazy et al. [34]. In a nutshell, they adopt the

(i) $param_{\mathrm{BPV}} \longleftarrow \mathrm{BPV}.Setup(1^\lambda)$ is performed by the *credential issuer*, and it takes as input the security parameter λ and then proceeds the following computations:
  (1) $param_{\mathrm{Waters}} \longleftarrow \mathrm{Waters}.Setup(1^\lambda)$, where $param_{\mathrm{Waters}}(p, G, g, G_t, e)$ are parameters of a bilinear map, $h \in G$
  (2) $param_{\mathrm{LE}} \longleftarrow \mathrm{LE}.Setup(1^\lambda)$, where (p, G, g, G_t, e) are parameters of a bilinear map
  (3) Output $param_{\mathrm{LZ}} = (param_{\mathrm{LE}}, param_{\mathrm{Waters}})$

(ii) $((\mathrm{ek}, \mathrm{dk}), (\mathrm{sk}, \mathrm{vk})) \longleftarrow \mathrm{LZ}.KGen(param_{\mathrm{LZ}})$ generates key pair $(\mathrm{sk}, \mathrm{vk})$ of Waters signature for the *credential issuer* while issuing secret and public key pair $(\mathrm{ek}, \mathrm{dk})$ of linear encryption to the *user*, and the detailed procedures are as follows:
  (1) $(\mathrm{sk}, \mathrm{vk}) \longleftarrow Waters.KGen(param)$, where $\mathrm{sk} = h^z$ and $\mathrm{vk} = g^z, z \in Z_p$
  (2) $(\mathrm{ek}, \mathrm{dk}) \longleftarrow LE.KGen(param)$, where $\mathrm{ek} = (\mathrm{ek}_1 = g^{y_1}, \mathrm{ek}_2 = g^{y_2})$ and $\mathrm{dk} = (y_1, y_2)$, y_1 and $y_2 \in Z_p$

(iii) $\sigma \longleftarrow CredMint(\mathrm{sk}, M)$ is executed by the *credential issuer* to generate a *cryptographic credential token* that is essential to a signature of Waters $\sigma \longleftarrow Waters.Sign(\mathrm{sk}, M)$ by inputting attributes M of the user under sk. In more detail, the *credential issuer* proceeds as follows:
  (1) Takes as input a selected random number $s \xleftarrow{\$} Z_p^*$ and calculates

$$\begin{aligned} \sigma_1 &= \mathrm{sk} \cdot F(M)^s \\ \sigma_2 &= g^s, \end{aligned} \tag{3}$$

  where $F(M) = u_0 \prod_{i=1}^{k} u_i^{m_i}$ for a vector $u = (u_0, u_1, \ldots, u_k) \in \mathbb{G}^{k+1}$ and an attribute set $M = (m_1, m_2, \ldots, m_k) \in \{0,1\}^k$.
  (2) Outputs $\sigma = (\sigma_1, \sigma_2)$

Remarkably, conventional DAC eliminates the centralized origination to issue the cryptographic credential token in a noninteractive approach; however, the simple AC should depend on a centralized part to issue the credential with an associated knowledge proof. Hence, compared with the syntax of DAC, in AC, there is no legal verification after issuing the credential by the origination.

(i) $CredShow(Prover(\mathrm{vk}, \mathrm{ek}, \sigma, M), Verifier(\mathrm{vk}, M))$. This algorithm is played by the prover and the verifier, and we regard the user as a prover for simplicity of illustration, where the witness of the prover is denoted as the randomness for the linear encryption r_1 and r_2.
  (1) *Prover* proceeds as follows
    (a) $\sigma' \longleftarrow Blind(\sigma)$, selects a randomness $s' \xleftarrow{\$} Z_p$ to blind the issued credential σ from the centralized credential issuer (*a.k.a*, certification authority), where $\sigma \longleftarrow \mathrm{Waters.Sign}(\mathrm{sK}, M) = (\sigma_1 = \mathrm{sk} \cdot F(M)^s, \sigma_2 = g^s)$, and outputs $\sigma' = (\sigma_1', \sigma_2')$, i.e.,

$$\left(\sigma_1' = \sigma_1 \cdot F(M)^{s\prime}, \sigma_2' = \sigma_2 \cdot g^{s'}\right). \tag{4}$$

    (b) $ct \longleftarrow LE.Enc(\mathrm{ek}, \sigma')$, selects two different random numbers $(r_1, r_2) \xleftarrow{\$} Z_p$, encrypts σ' under ek, and outputs the ciphertext $ct = (c_1 = \mathrm{ek}_1^{r_1}, c_2 = \mathrm{ek}_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \sigma_1', \sigma_2')$
    At the end of this phase, the prover sends $ct = (c_1 = \mathrm{ek}_1^{r_1}, c_2 = \mathrm{ek}_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \sigma_1', \sigma_2')$ to verifier
  (2) Upon receiving the ciphertext, the *verifier* proceeds as follows
    (a) $\mathrm{hk} \longleftarrow SPHF.HashKG(param, L)$ and outputs the hashing key $\mathrm{hk} = (x_1, x_2, x_3) \in Z_p^3$ by picking up three randomnesses x_1, x_2, and x_3 from Z_p
    (b) $\mathrm{hp} \longleftarrow SPHF.ProjKG(param), L, \mathrm{hk}, \mathrm{ek}$ and outputs the projective hashing key $\mathrm{hp} = (\mathrm{hp}_1 = \mathrm{ek}_1^{x_1} g^{x_3}, \mathrm{hp}_2 = \mathrm{ek}_2^{x_2} g^{x_3})$
    (c) $v \longleftarrow SPHF.Hash(param, \mathrm{hk}, (L, M), ct)$ and outputs v as $c_1^{x_1} c_2^{x_2} (c_3/M)^{x_3}$; particularly,

$$e(c_1, g)^{x_1} e(c_2, g)^{x_2} \left(\frac{e(c_3, g)}{e(h, \mathrm{vk}) e(F(M), \sigma_2')}\right)^{x_3}. \tag{5}$$

    (d) Chooses a random session key K and a random challenge $r \in \{0,1\}^n$, and computes $Q = K \oplus \mathrm{KDF}(v)$ and $W = K \oplus r$.
    Finally, the *verifier* sends (hp, Q, W) to the *prover*.
  (3) Upon receiving (hp, Q, W), the *prover* proceeds as follows
    (a) $v' \longleftarrow SPHF.Proj(param, \mathrm{hp}, (L, M), ct; w)$ and computes $v' = (\mathrm{hp}_1^{r_1} \mathrm{hp}_2^{r_2}, g)$, where $w = (r_1, r_2)$ is a witness owned by user privately. Particularly,

$$e(\mathrm{hp}_1^{r_1} \mathrm{hp}_2^{r_2}, g) = e(g, g)^{(y_1 r_1 x_1 + y_2 r_2 x_2 + (r_1+r_2)x_3)}. \tag{6}$$

    (b) Computes a session $K' = Q \oplus \mathrm{KDF}(v')$ and a random challenge $r' = K' \oplus W$.

At the end of this phase, the *prover* sends r' to verifier. Finally, the *verifier* returns 1 if r' is valid, 0 otherwise.

6. Our Construction: Self-Sovereign Decentralized Anonymous Credential Management System

Below, we will specifically describe our solution that contains four phases, i.e., *Initialization*, *Registration*, *Show Credential*, and *Audit*, as illustrated in Figure 3.

Initialization phase is performed by *Committee* members, and at the end of this phase, all public parameters of DVAC are generated. The committee specifies a language L: $\mathrm{WLin}(wpk, \mathrm{lpk}, M)$ and proceeds the following steps with SPHF over L:

(i) Define a vector $\mathbf{u} = (u_0, \ldots, u_k) \xleftarrow{\$} G^{k+1}$

(ii) A function $F(M) = u_0 \prod_{i=1}^{k} u_i^{m_i}$ for a vector $u = (u_0, u_1, \ldots, u_k) \in \mathbb{G}^{k+1}$, where m_i is an attribute set $m_i \in \{0,1\}^k, i \in (1,k)$

(iii) A hash function $H(\cdot)$

Then, the committee proceeds as follows:

(1) $param_{\mathrm{DVAC}} \longleftarrow Setup(\lambda)$ generates global parameters $param_{\mathrm{DVAC}} = (\mathrm{G}, \mathrm{G}_t, g, g_t, p, e)$ for the whole system, where G and G_t are two circle groups of order p, g is a generator of group G, g_t is a generator of group G_t, and e: $G \times G \longrightarrow G_t$, $g_t = e(g, g)$

(2) Process of key generation

(i) $(vsk, vpk) \longleftarrow EGSign.KGen(1^\lambda)$ and generate a keypair of ElGamal signature $((vsk = x_v, vpk = g_v^{x_v}))$, where $x_v \xleftarrow{\$} G$ and g_v is a generator of group G

(ii) $(wsk, wpk) \longleftarrow Waters.KGen(1^\lambda)$, select a random generator $h_w \xleftarrow{\$} G$, and generate a keypair of Waters signature $(wsk = h_w^{x_w}, wpk = g^{x_w})$, where $x_w \xleftarrow{\$} G$

(iii) $(usk, upk) \longleftarrow EG.KGen(1^\lambda)$ generate a keypair of ElGamal $(usk = x_m, upk = g_m^{x_m})$, where $x_m \xleftarrow{\$} G$ and g_m is a generator of group G

(3) The committee divides vsk into n shares $(s_1, s_2, \ldots, s_n) \longleftarrow SS.Share(vsk)$, and the committee selects a polynomial $p(X)$ of order n:

$$p(X) = x_v + a_1 X + a_2 X^2 + \cdots + a_{t-1} X^{t-1} \wedge p(0) = x_v = \lambda_1 s_1 + \lambda_2 s_2 + \cdots + \lambda_t s_t, \quad (7)$$

where n is the number of members of committee, λ_j is publicly constructible Lagrange coefficient, and t is the threshold value, $t < n$.

(4) Outputs $param_{\mathrm{DVAC}}$, (vsk, vpk), (wsk, wpk), (usk, upk), and $(s_1, s_2, \ldots, s_n)$

Finally, $(s_1, s_2, \ldots, s_n)$ is distributed by the dealer to all members of committee randomly.

(i) *Registration.* A new client runs Request(·) algorithm to send a registration request to the committee. The committee runs an Authenticate(·) algorithm to issue credentials to a new client. The entire process we have illustrated by the first part of Figure 3. When a new client joins this system, he will get his unique id on the blockchain.

(1) *New Client* generates his public key and private key locally and proceeds as follows:

(a) $(\mathrm{esk}, \mathrm{epk}) \longleftarrow EG.KGen(1^\lambda)$, where $\mathrm{esk} = x_e$, $\mathrm{epk} = g_e^{x_e}$, $x_e \xleftarrow{\$} G$, and g_e is a random generator in group G. The keypair $(\mathrm{esk}, \mathrm{epk})$ is used to sign and encrypt message at the registration phase.

(b) $(\mathrm{lsk}, \mathrm{lpk}) \longleftarrow LE.KGen(1^\lambda)$, where $\mathrm{lsk} = (\mathrm{lsk}_1 = x_1, \mathrm{lsk}_2 = x_2), \mathrm{lpk} = (\mathrm{lpk}_1 = g^{x_1}, \mathrm{lpk}_2 = g^{x_2})$, and x_1 and $x_2 \xleftarrow{\$} G$. The key pair $(\mathrm{lsk}, \mathrm{lpk})$ will be used in the second phase.

(c) $\mathrm{ck} \longleftarrow Ped.KGen(1^\lambda)$ and output a commitment parameter $\mathrm{ck} \xleftarrow{\$} G$.

(d) $\mathrm{attrs} \longleftarrow f(\mathrm{info})$, call attribution extraction function $f(\cdot)$, extract his attributes *attrs* from his private information *info*, and output *attrs*.

(e) $I \longleftarrow EG.Enc(\mathrm{info}, vpk)$ and output a ciphertext I by calculating $I = g_v^{r_1}, vpk^{r_1} \cdot \mathrm{info}$, where $r_1 \xleftarrow{\$} G$.

(f) $C_I \longleftarrow Ped.Com(I, \mathrm{ck})$, take inputs as commitment parameter ck, and output a commitment $C_I = g_c^I h_c^{\mathrm{ck}}$ for ciphertext I, where g_c and h_c are two random generator of group G.

(g) $m \longleftarrow EG.Enc(C_I, \mathrm{attrs}, \mathrm{id}; upk)$ and output a ciphertext m by calculate $m = (g_m^{r_2}, upk^{r_2} \cdot (C_I \| \mathrm{attrs} \| \mathrm{id}))$, where $r_2 \xleftarrow{\$} G$.

(h) $s_m \longleftarrow EGSign.Sign(m, \mathrm{esk})$, sign for ciphertext m, and output a signature $s_m = (g_e^{k_e}, k_e^{-1}(H(m) - t x_e n g_e^{k_e}))$ by esk_i, where g_e is a random generator of group G and $k_e \xleftarrow{\$} G$. Finally, the *New Client* sends (m, s_m) to the committee. The request part is all done offline by the *New Client*.

(2) After receiving the request from *New Client*, the *Committee* proceeds as follows:

(a) $(0,1) \longleftarrow EGSign.Verify(m, s_m, \mathrm{epk})$ and verify the validity of m by epk at first. Calculate $g_e^{H(m)} \stackrel{?}{=} (g_e^{x_e})^{g_e^{k_e}} (g_e^{k_e})^{k_e^{-1}(H(m) - x_e g_e^{k_e})}$. Continue if the output is 1, otherwise return false to New Client.

(b) $(C_I, \mathrm{attrs}, \mathrm{id}) \longleftarrow EG.Dec(m, usk)$ and output $(C_I, \mathrm{attrs}, \mathrm{id})$ by calculating $(\mathbf{C}_I \| \mathrm{attrs} \| \mathrm{id}) = ((upk)^{r_2} \cdot (\mathbf{C}_I \| \mathrm{attrs} \| \mathrm{id})) \cdot ((g_m^{r_2})^{usk})^{-1}$.

(c) $\sigma \longleftarrow Waters.Sign(\mathrm{attrs}, wsk)$, take as input a selected random number $r_w \xleftarrow{\$} G$, and calculate and output $\sigma = (\sigma_1 = h_w^{x_w} \cdot F(\mathrm{attrs})^{r_w}, \sigma_2 = g^{r_w})$.

(d) Upload $(\mathrm{id}: C_I, \mathrm{epk})$ to bulletin board. This information will not be changed forever.

Finally, the *Committee* returns σ to *New Client*.

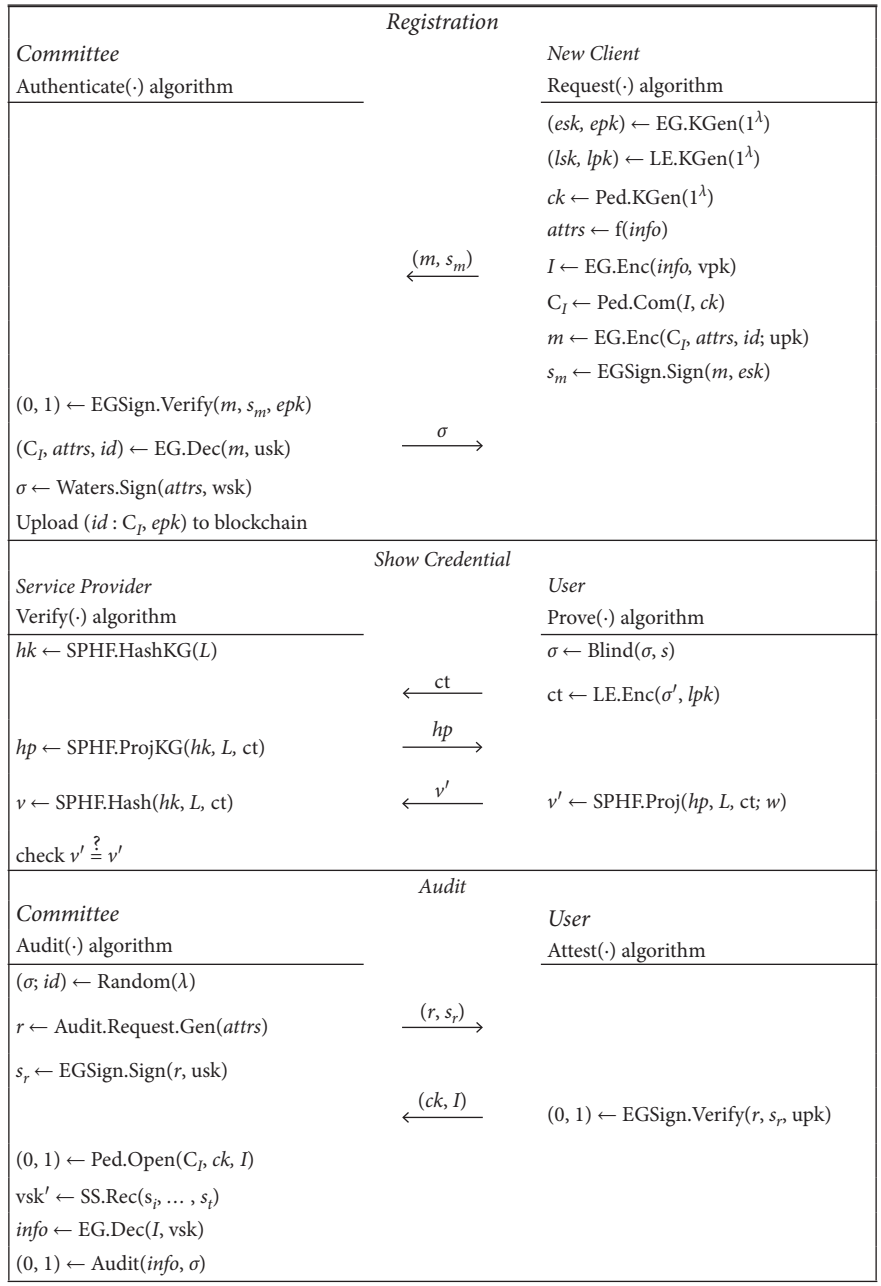

Figure 3: An illustration of DVAC.

(ii) *Show Credential.* After a new client gets his credentials from the committee, it means that he is a legal user of blockchain. When a user wants to obtain a service from SP, he needs to provide the corresponding credentials. And, this process is anonymous for SP. This means that the SP can only provide the appropriate service based on the credentials, and it can learn no information from the user. This phase is shown in the second part of Figure 3. User runs Prove(·) algorithm and SP run Verify(·) algorithm.

(1) User proceeds as follows:

(a) $\sigma' \longleftarrow Blind(\sigma, s)$, select a randomness $s \xleftarrow{\$} Z_p$ to blind the issued credential σ from the centralized credential issuer (*a.k.a*, certification authority), where $\sigma \longleftarrow \text{Water.Sign}(\text{wsk}, \text{attrs}) = (\sigma_1 = \text{wsk} \cdot F(\text{attrs})^{r_w}, \sigma_2 = g^{r_w})$, and output $\sigma' = (\sigma_1', \sigma_2')$, i.e., $(\sigma_1' = \sigma_1 \cdot F(M)^s, \sigma_2' = \sigma_2 \cdot g^s)$

(b) $ct \longleftarrow LE.Enc(\sigma', \text{lpk})$, select two different random numbers $(r_1, r_2) \xleftarrow{\$} Z_p$, encrypt σ' under lpk, and output the ciphertext $ct = (c_1 = \text{lpk}_1^{r_1}, c_2 = \text{lpk}_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \sigma_1', \sigma_2')$

At the end of this phase, the prover sends $ct = (c_1 = \text{lpk}_1^{r_1}, c_2 = \text{lpk}_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \sigma_1', \sigma_2')$ to SP.

(2) Upon receiving the ciphertext, the SP proceeds as follows:

(a) $\text{hk} \longleftarrow SPHF.HashKG(param_{\text{DVAC}}, L)$ and output the hashing key $\text{hk} = (x_1, x_2,$

$x_3) \in Z_p^3$ by picking up three randomnesses x_1, x_2, and x_3 from Z_p

(b) $\text{hp} \longleftarrow SPHF.ProjKG(param_{\text{DVAC}}, L, \text{hk}, \text{lpk})$ and output the projective hashing key $\text{hp} = (hp_1 = \text{lpk}_1^{x_1} g^{x_3}, \text{hp}_2 = \text{lpk}_2^{x_2} g^{x_3})$

(c) $v \longleftarrow SPHF.Hash(param_{\text{DVAC}}, \text{hk}, (L, \text{attrs}), ct)$ and output v as $c_1^{x_1} c_2^{x_2} (c_3/\text{attrs})^{x_3}$; particularly,

$$e(c_1, g)^{x_1} e(c_2, g)^{x_2} \left(\frac{e(c_3, g)}{e(h, wpk) e(F(\text{attrs}), \sigma_2')} \right)^{x_3}. \quad (8)$$

Finally, the SP sends hp to the User.

(3) $v' \longleftarrow SPHF.Proj(param_{\text{DVAC}}, \text{hp}, (L, M), ct; w)$, and upon receiving hp, the User computes $v' = (\text{hp}_1^{r_1} \text{hp}_2^{r_2}, g)$, where $w = (r_1, r_2)$ is a witness owned by user privately.

At the end of this phase, the User sends v' to SP. Finally, the *verifier* returns 1 if r' is valid, 0 otherwise.

(i) *Audit.* We denote the algorithm implemented by the committee and user, respectively, as Attest(·) and Audit(·). This phase is shown in the third part of Figure 3.

(1) The *Committee* first performs the following actions:

(a) $(\sigma; \text{id}) \longleftarrow Random(\lambda)$, and when running the Audit(·) algorithm, the committee randomly selects id on blockchain and the corresponding credentials committee had issued.

(b) $r \longleftarrow Audit.Request.Gen(\text{attrs})$, and the committee generates a request message r based on the relevant credentials.

(c) $s_r \longleftarrow EGSign.Sign(r, usk)$, and generate a signature of the request message r, and the committee calculates $s_r = (g_m^{k_m}, k_m^{-1}(H(r) - t x_m n g_m^{k_m}))$, where $k_m \xleftarrow{\$} G$.

After that, the *Committee* sends (r, s_r) to *User.* When the user receives the committee's audit request, the user should decide whether he is audited. If the user refuses, the committee will add a suspect tag to the blockchain. The user's behavior in the future will be affected. If the user accepts, he/she will do the following:

(2) $(0, 1) \longleftarrow EGSign.Verify(r, s_r, upk)$, and verify the validity of the audit request by computing $g_m^{H(r)} \stackrel{?}{=} (g_m^{x_m})^{g_m^{k_m}} (g_m^{k_m})^{k_m^{-1}(H(r) - x_m g_m^{k_m})}$.

Then, the *User* sends his opening ck and I to *Committee.*

(3) Upon reception of opening ck and I, the committee executes as follows:

(a) $(0, 1) \longleftarrow Ped.Open(C_I, \text{ck}, I)$, and the committee opens the user's commitment which uploads to blockchain in the registration phase and computes $g_c^I h_c^{ck} \stackrel{?}{=} C_I$; return true if output 1; otherwise, return false to the user.

(b) $vsk' \longleftarrow SS.Rec(s_i, \ldots, s_t)$, and vsk' is a temporary private key related to vsk, and it can only be used once for each user. The committee recovers temporary private key by calculate $vsk' = (g_v^{r_1})^{\lambda_1 s_1}, \ldots, (g_v^{r_1})^{\lambda_t s_t}$ and then output vsk'.

(c) $\text{info} \longleftarrow EG.Dec(I, vsk)$, and the committee calculates $\text{info} = (vpk)^{r_1} \cdot \text{info} \cdot (vsk\prime)^{-1}$

(d) $(0, 1) \longleftarrow Audit(\text{info}, \sigma)$, and the committee verifies whether σ is generated by info. If the audit fails, the committee will remove the user's information on the blockchain; otherwise, return true.

(ii) *Secret Refresh.* This phase usually has a fixed amount of time to run within a period, unless something special happens to trigger it (such as a sudden occurrence). Same as the *Audit* phase, there must be approval by threshold members of the committee in the current period, and the *Secret Refresh* phase will be run.

(1) $(m, k, l) \longleftarrow SS.RefSetup(\lambda)$, and all members of the committee decide the new number of members m, the new threshold k, and the new time of interval l.

(2) $u_{i,j} \longleftarrow SS.RefCompute(n, m)$, and each of them constructs a random polynomial of the form $\delta_i(z) = \delta_{i,1} z^1 + \delta_{i,2} z^2 + \cdots + \delta_{i,k} z^{k-1}$, where $\delta_{i,k}$ is a coefficient of polynomial $\delta_i(z)$. Then, they compute and send all other players $u_{i,j} = \delta_i(j)$, where $i \in (0, n)$ and $j \in \{0, 1, \ldots, m\}$.

(3) $s_i^{l+1} \longleftarrow SS.Rec(s_i, u_{1,j}, \ldots, u_{n,j})$, and each of them updates their share by $s_i^{l+1} = s_i^l + u_{1,j}^l + \cdots + u_{n,j}^l$.

$$\text{DVAC}Adv_{\mathcal{A}}^{\text{sem}} \leq 2 \cdot \text{DDH}Adv_{B_{\text{DDH}}} + 2 \cdot \text{DL}Adv_{B_{\text{DL}}}. \quad (9)$$

Theorem 1 (correct). *DVAC is a scheme which satisfies soundness.*

Proof. The soundness of *Show Credential* phase relies on the correctness of *SPHF*, $SPHF.Hash(\text{hk}, L, ct) = SPHF.Proj(\text{hp}, L, ct; w)$, and $v' = v$. □

Proof. The soundness of the Audit phase relies on the binding of Pedersen commitment and the correctness of Shamir's Secret Sharing, $g_c^I h_c^{ck} = C_I, vsk\prime == (g_v^{r_1})^{\lambda_1 s_1}, \ldots, (g_v^{r_1})^{\lambda_t s_t}$, thus $\text{info} = (vpk)^{r_1} \cdot \text{info} \cdot (vsk\prime)^{-1}$. □

Theorem 2 (semantically secure). *The DVAC is Semantically Secure if DDH holds for G, and the commitment scheme is perfectly hiding:*

Proof. We assume that an adversary $\mathcal{A}$ against the semantic security of our scheme with advantage ϵ. We start from this

initial security game. $\mathcal{G}_0$ is consistent with the situation in a real attack.

Game $\mathcal{G}_0$. Let us emulate this security game:

(1) $\mathcal{B}$ emulates the initialization of the system: it runs $EG.KGen(1^\lambda)$, generates (vpk, upk), and sends (vpk, upk) to adversary $\mathcal{A}$

(2) $\mathcal{B}$ simulates oracles of the transcripts of protocol:

(i) Runs $EG.Enc(M, vpk)$ for a message M and outputs I

(ii) Runs $Ped.Com(I, \text{ck})$ for a I and outputs C_I

(iii) Runs and outputs m

(iv) Runs $EG.Enc(C_I, \text{attrs}, \text{id}; upk) EGSign.Sign(m, \text{esk})$ for an I and outputs s_m

(3) $\mathcal{A}$ generates two inputs (M_0, M_1) and sends to $\mathcal{B}$

(4) $\mathcal{B}$ chooses a random bit $b \longleftarrow \{0, 1\}$ and simulates the protocol for M_b, and then, $\mathcal{B}$ sends m and s_m to adversary $\mathcal{A}$

(5) $\mathcal{A}$ outputs a bit b' and sends b' to $\mathcal{B}$

In this game, $\mathcal{B}$ only plays the role of a challenger; in $\mathcal{A}'s$ perspective, he/she is still interacting with the real DVAC. We then modify the challenger to obtain Games $\mathcal{G}_1$ and $\mathcal{G}_2$. In each game, b denotes the random bit chosen by the challenger $\mathcal{B}$, while b' denotes the bit output by $\mathcal{A}$. Also, for $j = 0$, 1, and 2, we define W_j to be the event $\mathcal{A}$ win this game for $b' = b$. By definition, we have

$$\text{DVAC}Adv_{\mathcal{A}}^{\text{sem}} = \left|\Pr[W_0] - \frac{1}{2}\right|. \tag{10}$$

Game $\mathcal{G}_1$. In this game, the challenger is as $\mathcal{G}_0$, except a little modifications as follows:

(i) $\mathcal{B}$ uses a randomness $\gamma \xleftarrow{\$} G$ such that $m = (g_m^{r_2}, g_m^{\gamma} \cdot (C_I \| \text{attrs} \| \text{id}))$.

Adversary $\mathcal{B}$ plays attack game against challenger of DDH and plays the role of challenger to $\mathcal{A}$ as $\mathcal{G}_0$. When $\mathcal{A}$ outputs b', if $b = b'$, then $\mathcal{B}$ outputs 1; else, outputs 0. So, we have

$$\text{DDH}Adv_B^{\text{sem}} = \left|\Pr[W_0] - \Pr[W_1]\right|. \tag{11}$$

Game $\mathcal{G}_2$. In this game, the challenger is as $\mathcal{G}_1$, except a little modifications as follows:

(ii) $\mathcal{B}$ chooses a random bit $b \in \{0, 1\}$, sends to challenger of DL, and then obtains C_b'. Finally, $\mathcal{B}$ sends C_b' to adversary $\mathcal{A}$.

In more detail, adversary $\mathcal{B}$ plays attack game against challenger of DL and plays the role of challenger to $\mathcal{A}$. When $\mathcal{A}$ outputs b', if $b = b'$, then $\mathcal{B}$ outputs 1; else, outputs 0. Based on DL assumption, the adversary $\mathcal{A}$ has only 1/2 probability win this game. So, we have

$$\text{DL}Adv_B^{\text{sem}} = \left|\Pr[W_1] - (1/2)\right|. \tag{12}$$

Combining 2 and 3, yields 1, which completes the proof of the theorem. □

7. Self-Sovereign Decentralized Identity Management via DVAC

7.1. Application of Identity Management. In this section, we discuss several of the applications by DVAC. We consider the scenario where a user wants to register a long-term identity credential on the blockchain. This credential enables repeated presentation to any third parties several times without revealing extrainformation. A third party could verify the validity (whether it has the corresponding attributes) of credentials and provide related services for the user. All users on blockchain will be managed by the miner nodes which we called members of the committee. Miner nodes are not permanent, and each user could be a miner node. In addition to maintaining the system, these nodes are responsible for auditing users with a decentralized operation. This application extends the work of DAC [4] which does not consider audit and the issuance of a decentralized credential.

Our identity management system is based on blockchain and cryptocurrency. We use a public ledger to record the credentials of users with their other information, such as the bulletin board in DAC. There are three types of parties except for the public ledger: a group of credentials issuance and audit *Committee*, a set of SP, and a set of *User*.

Before the system initializes, the first batch of members of *Committee* should be specified. As shown in Figure 3, the system parameters have been setup at the initialization phase. A user sends a registration request, a commitment of encrypted privacy information, and some information needed to prove his identity attributes to *Committee*. These attributes could be age, address, or credibility. The *Committee* checks the user's information and issues a long-term credential (signature) on its sign private key *wsk*. This credential can be obtained only once and will not gonna change.

When a user wants to get service from a SP, he/she needs to show his attribute that satisfies the request of the SP. He/she should show the corresponding credential (age, sex, or property) through a zero-knowledge proof for a specified verifier. To prevent SP replay attacks, he/she blinds this credential at first. Then, he/she follows the WLin language interactive with SP. This process could not reveal any other information except the user's attributes.

Committee could doubt attributes' authenticity of users and combine with an audit algorithm to judge it periodically. At first, the *Committee* selects users on blockchain randomly. Then, the *Committee* sends an audit request to these users. It means that the right to accept the audit or not is in the hands of the users. If the user refuses, the *Committee* will mark him and his reputation will be impacted. If users accept, they send commitment openings (*ck*, *I*) to *Committee*. The *Committee* opens the commitment that is sent by

Table 1: Comparison of the identity management system.

Scheme	Passport [1]	CertCion [16]	DAC[4]	DBLACR [5]	DVAC
Unlinkability	✗	✗	✓	✓	✓
Unforgability	✗	✗	✓	✓	✓
Anonymity	✗	✓	✓	✓	✓
D-Auditability	✗	✗	✗	✗	✓

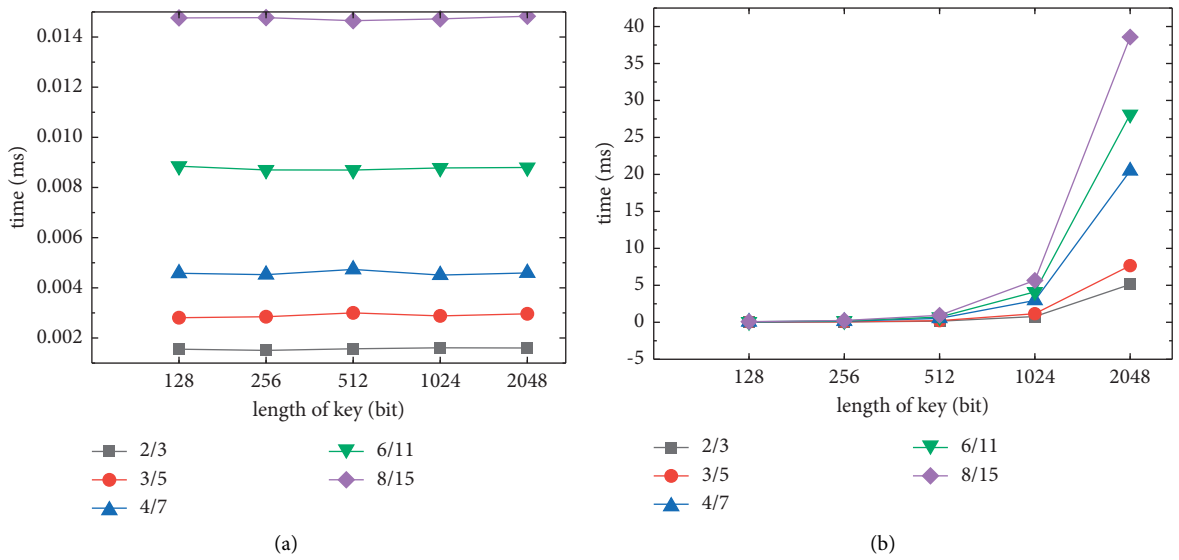

Figure 4: (a) Time-consuming of secret distribution in different thresholds (left). (b) Time-consuming of secret reconstruction (right).

the user in the registration phase. Finally, the *Committee* recovers the corresponding secret key and audits the privacy information of the user. If the result of the audit is right, return true; otherwise, the user will be removed from the blockchain and investigated relevant legal liabilities.

In Table 1, we compare our construction DVAC with other systems. Although DVAC has a slight deficiency in performance because of the secret sharing, it has a sufficient guarantee in security and privacy.

7.2. Future Work. DVAC's focus is on adding audit capability to a decentralized identity management system and addressing the single point of failure that can occur during the audit process. However, it is still an interactive proof system between the user and the SP, which will incur unnecessary waiting time loss. If in an environment with high network latency, it is likely to cause network congestion. DVAC also does not support forward-secure for audit secret key. The future work of DVAC is to provide a forward-secure audit and noninteractive proof system.

8. Evaluation

To evaluate DVAC, we implemented each step of DVAC with C++. Our essential environment is based on GMP and PCB library. We run microbenchmarks on a 4 core Intel machine with i7-8500T 2.1 GHz CPU and 8 GB of RAM, running 64-bit Windows 10. We measured the time-consuming of the key generation process, secret sharing process, and secret reconstruct process for secret keys of different bit lengths because the real interactive system is related to the speed of network transmission and is not stable. Let us assume that the network transfer does not take time and only measures the time consumption in the local calculation.

As shown in Figure 4(a), we compare the time-consuming of different key lengths. At the secret key distribution stage, there is no time consumption of secret keys of different lengths is no significant change. The timing of the distribution of the secret keys is only related to the number of participants involved in the distribution.

Figure 4 rightly depicts the time required for the auditor to recover the user information for different thresholds and different secret key lengths during the audit phase. The values of the points are very close to each other when the secret key size is less than 1024 bits. Only when the secret key size is 1024 bit and 2048 bit, the time difference required by different threshold sizes are relatively significant changes.

As shown in Figure 5(a), we can find that, with the increase of the length of the secret key, the generation time of the secret key increases exponentially. However, this increase in time is not something we should care about because the entire initialization phase is done entirely offline.

To evaluate the time-consuming during the secret reconstruction phase, as shown in Figure 5(b) and Figure 6(a), we select the threshold as 4 and evaluate the 4 out of 7 setting. The results imply that the time consumption in the

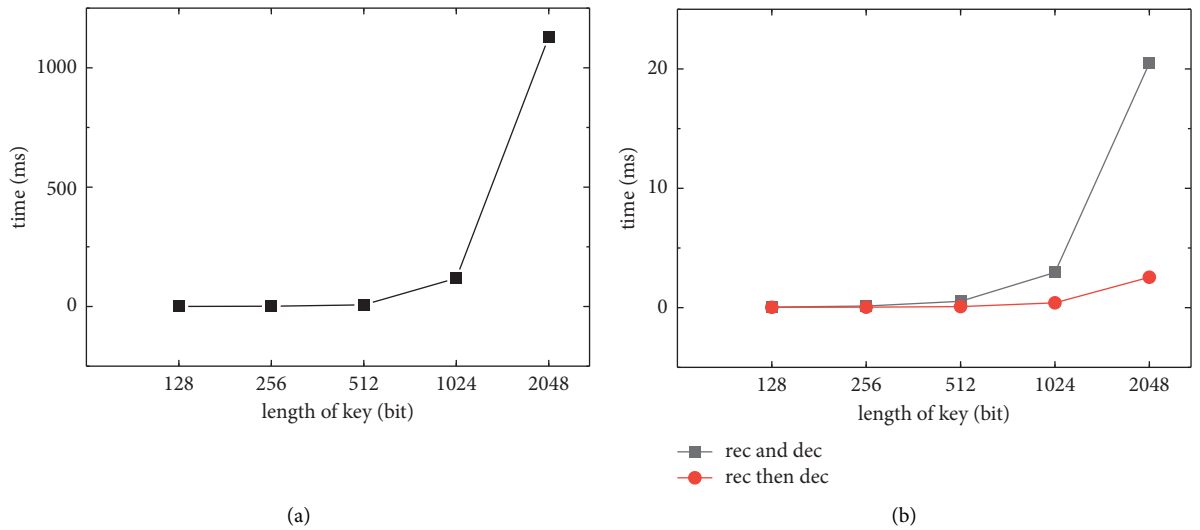

Figure 5: (a) Time-consuming of key generation (left). (b) Time-consuming of secret reconstruction by different methods (right).

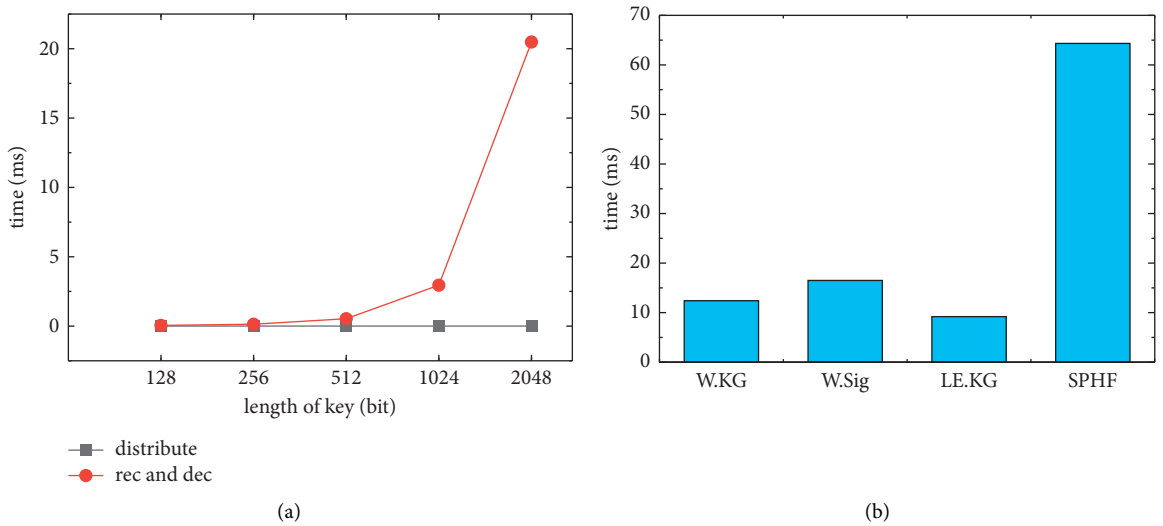

Figure 6: (a) Time-consuming of secret distribution and secret reconstruction (left). (b) Time-consuming in Waters key-generation (W.KG), Waters signature (W.Sig), linear encryption key-generation (LE.KG), and entire interaction process of SPHF (right).

audit stage is much greater than that in the secret key distribution stage when the length of the secret key is greater than 1024 bits. Further, comparing with traditional "reconstruction then decryption," the approach of "reconstruction and decryption" is more time-consuming. This is due to the multiple modular exponentiations. When the length of the secret key is longer, its time grows faster. This is the main disadvantage in realization.

As shown in Figure 6(b), we compare the time-consuming verification algorithms including key-generation and signature of Waters scheme, key-generation of linear encryption, and total verification of SPHF.

9. Conclusion

The existing anonymous credential identity management system usually exposes two shortcomings when it is implemented. One is that the correctness of identity information cannot be guaranteed when the privacy of the user's identity information is guaranteed. Second, its centralized management will lead to a single point of the failure problem. In this paper, we propose DVAC, a designated-verifier anonymous credential in self-sovereign decentralized identity management. We added the audit function to solve the problem that the correctness of user identity information could not be guaranteed. We also solved the problem of single-point failure of the centralized management system to some extent through secret sharing and realized decentralized identity management. We provide the detailed step of the DVAC system and the cryptographic primitives underlying DVAC. We also implement and evaluate DVAC and application of identity management. DVAC provides a new way for designing an identity management system.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61702294), Applied Basic Research Project of Qingdao City (17-1-1-10-jch).

References

[1] R. O. Microsoft, "Net passport: a security analysis," *Computer*, vol. 36, no. 7, pp. 29–35, 2003.

[2] D. Fisher, "Final report on diginotar hack shows total compromise of ca servers," 2012.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[4] C. Garman, M. Green, and I. Miers, "Decentralized anonymous credentials," in *Proceedings of the 21th NDSS 2014*, February 2014.

[5] R. Yang, M. H. Au, Q. Xu, and Z. Yu, "Decentralized blacklistable anonymous credentials with reputation," *Computers & Security*, vol. 85, pp. 353–371, 2019.

[6] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Proceedings of the 24th CRYPTO 2004*, vol. 3152, pp. 56–72, Springer, Santa Barbara, CA, USA, August 2004.

[7] M. H. Au, W. Susilo, Y. Mu, and S. S. M. Chow, "Constant-size dynamic k-times anonymous authentication," *IEEE Systems Journal*, vol. 7, no. 2, pp. 249–261, 2013.

[8] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures,"vol. 3152, pp. 41–55, in *Proceedings of the 24th CRYPTO 2004*, vol. 3152, Springer, Santa Barbara, CA, USA, August 2004.

[9] N. Begum, T. Nakanishi, and N. Funabiki, "Efficient proofs for cnf formulas on attributes in pairing-based anonymous credential system," *ICE Transactions on Fundamentals of Electronics Communications and Computer ences*, vol. E96-A, no. 12, pp. 495–509, 2012.

[10] J. Camenisch, S. Mödersheim, and D. Sommer, "A formal model of identity mixer," in *Proceedings of the 15th FMICS 2010*, vol. 6371, pp. 198–214, Springer, Antwerp, Belgium, September 2010.

[11] Z. Li, D. Wang, and E. Morais, "Quantum-safe round-optimal password authentication for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[12] C. Buchholz, "Liberty alliance project-gemeinschaftliche identitätsverwaltung," *Datenschutz und Datensicherheit*, vol. 27, no. 9, 2003.

[13] G. Karjoth, "Access control with IBM Tivoli access manager," *ACM Transactions on Information and System Security*, vol. 6, no. 2, pp. 232–257, 2003.

[14] W. Stackpole, "Centralized authentication services," in *Encyclopedia of Information Assurance*Taylor & Francis, Boca Raton, FL, USA, 2011.

[15] Z. Li, C. Ma, and H.-S. Zhou, "Multi-key FHE for multi-bit messages," *Science China Information Sciences*, vol. 61, no. 2, pp. 029101:1–029101:3, 2018.

[16] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention. IACR Cryptol," 2014.

[17] M. Ali, J. C. Nelson, R. Shea, and M. J. F. Blockstack, "A global naming and storage system secured by blockchains," in *Proceedings of the USENIX ATC 2016*, pp. 181–194, USENIX Association, Berkeley, CA, USA, June 2016.

[18] A. Sonnino, M. Al-Bassam, S. Bano, S. Meiklejohn, and G. Danezis, "Coconut: threshold issuance selective disclosure credentials with applications to distributed ledgers," 2019.

[19] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proceedings of the EUROCRYPT 2003*, vol. 2656, pp. 416–432, Springer, Warsaw, Poland, May 2003.

[20] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Proceedings of the CT-RSA 2016*, vol. 9610, pp. 111–126, Springer, San Francisco, CA, USA, February 2016.

[21] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[22] H. Halpin, "Nym credentials: privacy-preserving decentralized identity with blockchains," in *Proceedings of the CVCBT 2020*, pp. 56–67, IEEE, Rotkreuz, Switzerland, June 2020.

[23] N. Narula, W. Vasquez, and M. Virza, "Zkledger: privacy-preserving auditing for distributed ledgers," in *Proceedings of the 15th NSDI 2018*, pp. 65–80, USENIX Association, Renton, WA, USA, April 2018.

[24] J. Balasch, A. Rial, C. Troncoso, B. Preneel, I. Verbauwhede, and C. Geuens, "Pretp: privacy-preserving electronic toll pricing," in *Proceedings of the 19th USENIX 2010*, pp. 63–78, USENIX Association, Washington, DC, USA, August 2010.

[25] S. Meiklejohn, K. Mowery, S. Checkoway, and H. Shacham, "The phantom tollbooth: privacy-preserving electronic toll collection in the presence of driver collusion," in *Proceedings of the 20th USENIX 2011*, USENIX Association, Berkeley, CA, USA, August 2011.

[26] R. Cramer and V. Shoup, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption," in *Proceedings of the EUROCRYPT 2002*, vol. 2332, pp. 45–64, Springer, Amsterdam, The Netherlands, April 2002.

[27] Z. Li and D. Wang, "Two-round PAKE protocol over lattices without NIZK," in *Information Security and Cryptology - 14th International Conference, Inscrypt 2018, Fuzhou, China, December 14-17, 2018, Revised Selected Papers, Volume 11449 of Lecture Notes in Computer Science*, pp. 138–159, Springer, Berlin, Germany, 2018.

[28] Z. Li and D. Wang, "Achieving one-round password-based authenticated key exchange over lattices," *IEEE Transactions on Services Computing*, 2019.

[29] Z. Li, Z. Yang, P. Szalachowski, and J. Zhou, "Building low-interactivity multi-factor authenticated key exchange for industrial internet-of-things," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 844–859, 2021.

[30] O. Blazy, G. Fuchsbauer, D. Pointcheval, and D. Vergnaud, "Signatures on randomizable ciphertexts," in *Proceedings of the 14th PKC 2011*, vol. 6571, pp. 403–422, Springer, Taormina, Italy, March 2011.

[31] B. Waters, "Efficient identity-based encryption without random oracles," in *Proceedings of the 24th EUROCRYPT 2005*,

vol. 3494, pp. 114–127, Springer, Aarhus, Denmark, May 2005.

[32] M. Blanton, "Online subscriptions with anonymous access," in *Proceedings of the ACM ASIACCS 2008*, pp. 217–227, ACM, Tokyo, Japan, March 2008.

[33] M. Blanton and W. M. P. Hudelson, "Biometric-based non-transferable anonymous credentials," in *Proceedings of the 11th ICICS 2009*, vol. 5927, pp. 165–180, Springer, Beijing, China, December 2009.

[34] O. Blazy, D. Pointcheval, and D. Vergnaud, "Round-optimal privacy-preserving protocols with smooth projective hash functions," in *Proceedings of the 9th TCC 2012*, vol. 7194, pp. 94–111, Springer, Sicily, Italy, March 2012.