Journal of Information Security, 2022, 13, 43-65
https://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

A Verifiable Credentials System with Privacy-Preserving Based on Blockchain

Zhiji Li

College of Information Science and Technology, Jinan University, Guangzhou, China
Email: lzjjj@stu2019.jnu.edu.cn

How to cite this paper: Li, Z.J. (2022) A Verifiable Credentials System with Privacy-Preserving Based on Blockchain. *Journal of Information Security*, **13**, 43-65.
https://doi.org/10.4236/jis.2022.132003

Received: January 28, 2022
Accepted: February 25, 2022
Published: February 28, 2022

Abstract

Decentralized identity authentication is generally based on blockchain, with the protection of user privacy as the core appeal. But traditional decentralized credential system requires users to show all the information of the entire credential to the verifier, resulting in unnecessary overexposure of personal information. From the perspective of user privacy, this paper proposed a verifiable credential scheme with selective disclosure based on BLS (Bohen-Lynn-Shacham) aggregate signature. Instead of signing the credentials, we sign the claims in the credentials. When the user needs to present the credential to verifier, the user can select a part of but not all claims to be presented. To reduce the number of signatures of claims after selective disclosure, BLS aggregate signature is achieved to aggregate signatures of claims into one signature. In addition, our scheme also supports the aggregation of credentials from different users. As a result, verifier only needs to verify one signature in the credential to achieve the purpose of batch verification of credentials. We analyze the security of our aggregate signature scheme, which can effectively resist aggregate signature forgery attack and credential theft attack. The simulation results show that our selective disclosure scheme based on BLS aggregate signature is acceptable in terms of verification efficiency, and can reduce the storage cost and communication overhead. As a result, our scheme is suitable for blockchain, which is strict on bandwidth and storage overhead.

Keywords

Verifiable Credential, Aggregate Signatures, Privacy Protection, Selective Disclosure, Blockchain

1. Introduction

Credentials are a part of our daily lives, such as driver's licenses, university degrees, government-issued passports and so on. A verifiable credential is a tamper-evident credential that has authorship that can be cryptographically verified [1]. Verifiable credential system is the core of the decentralized identity authentication system. Traditional identity authentication has the problems of data dispersion and repeated authentication. Users need to register different identity information in different Internet authentication systems. These identity information overlap each other. On the one hand, it causes a waste of storage resources. On the other hand, it also makes users need to perform repeated registration and verification, which brings inconvenience to users. The decentralized verifiable credential system cannot be separated from blockchain, and of blockchain provides credential system with decentralized feature [2]. Besides, it is the basis for user identity autonomy and a platform for managing identities, credentials, and data storage. Blockchain and identity encryption on the chain can turn centralized identity issuance and data sharing into distributed identity authentication. Users can control the identity and private key to carry out trusted identity authorization and sharing among multiple identity institutions, so as to solve the problem of duplicate authentication and center failure. Therefore, whether from the perspective of privacy or sharing, the research on decentralized identity authentication is very meaningful.

Verifiable credential is an indispensable part of decentralized identity authentication and autonomous identity. The verifiable credential system includes three roles: Issuer, User, and Verifier. **Figure 1** shows the structure and process of the decentralized identity authentication system, which includes the process of identity registration, credential issuance, and credential verification. Issuer verifies legitimacy and personal information of user, and then signs and issues credentials to user to provide user with a trust endorsement. When needed, the user will present credential to the verifier, and verifier will check whether the signature of the credential is issued by the corresponding issuer, thereby verifying the validity of the credential. After obtaining the credentials, user can choose to keep the credential locally, or put it on the blockchain for hosting. When user needs to be verified, he can show the credentials to the verifier. And the credential contains certain attributes which prove that the user has sufficient qualifications to meet the requirements for service access. Through verifying claims in credentials, verifier can confirm user's identity legality.

There are, however, some limitations of verifiable credential in comparison to established, centrally controlled authentication platforms concerning trust, privacy and usability [3]. Firstly, traditional decentralized verifiable credential system has privacy leakage defect of excessive exposure of personal information, that is, credential contains many information of user, while verifier may only need to obtain some information of the user. User does not want to expose personal information to verifier excessively, which has the problem of privacy protection [4]. Secondly, traditional research does not focus on how to aggregate different credentials into one credential, which is essential in practical application scenarios. Thirdly, verifiers often need to verify credentials from different users at the same time, which poses certain challenges to the throughput and verification efficiency of the system [5].

Our scheme provides a selective disclosure [6] scheme, so user can prove claims about their identity without revealing more information than they intend and need for performing a specific action. For example, Jane only has to share her age (without her gender) when she orders wine in an online store, as it is sufficient for her to state that she is old enough to purchase wine legally. In addition, Verifier may not only require users to show one credential, but may require different credentials issued by multiple organizations to make verifier trust the user's identity. For example, when we join a company, we often need to show credentials such as graduation credential, degree credential, ID card, etc. But we do not want to show the attribute of the household registration in the ID card to the employer, and the employer may not require this attribute. Therefore, we not only need to present attribute information from different credentials, but also need to selectively present some attribute information of these credentials. That is, from the user's point of view, the user hopes to show some but not all personal information in different credentials to verifier. Our selective disclosure and aggregation credential scheme aims to solve this problem.

In order to achieve selective disclosure, we use the method of signing claims instead of signing the entire credential, but this will result in too many signatures. Therefore, our scheme proposes an aggregate signature scheme that can aggregate the user's claims, thereby reducing the number of signatures and storage space stored in the blockchain. Selective disclosure is to aggregate the signatures of the claims that the user chooses to expose from different credentials. We use the BLS aggregate signature scheme [7], a short signature scheme to achieve our goal. Traditional verifiable credential system has the problem that credential volume is large and cannot be authenticated in batches. Also, our scheme supports the aggregation of the credentials from different users, thereby reducing the overall size of the credential, which is suitable for blockchain storage and batch authentication. Since the blockchain is only suitable for storing small-capacity data, traditional verifiable credential system is not suitable for blockchain-based decentralized identity authentication. The claims of credentials are signed with BLS signature. Due to the claims of BLS signatures, different signatures of claims can be aggregated. The verifier only needs to verify the final aggregate credentials to verify all the credentials.

This paper proposes a selective disclosure and credentials aggregation scheme based on BLS aggregate signature to solve the problems of privacy breach and credentials bulky, so as to reduce the blockchain network bandwidth and storage overhead and achieve the purpose of batch verification of credentials.

The organization of this paper is as follows: Section 2 introduces the current work related to verifiable credential; Section 3 introduces the theoretical basis of verifiable credential and BLS aggregate signature; Section 4 proposes a selective disclosure and credentials aggregation scheme based on BLS aggregate signature; Section 5 discusses the security and efficiency of the scheme; Section 6 summarizes

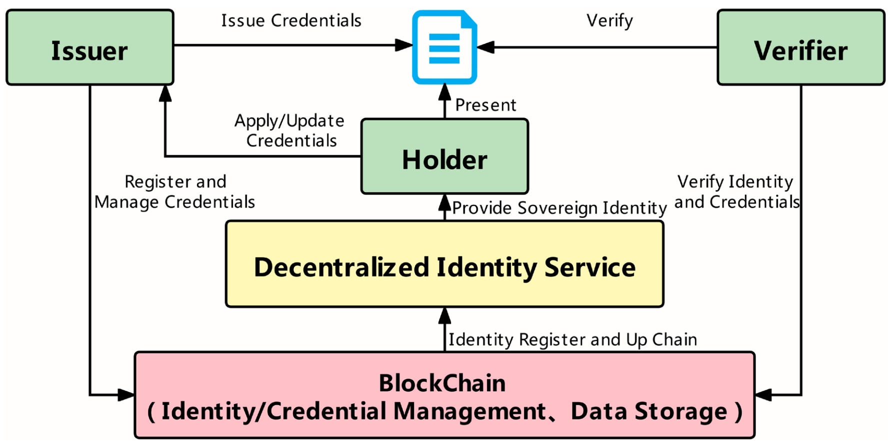
Issuer
Issue Credentials
Verify
Verifier
Present
Apply/Update Credentials
Holder
Register and Manage Credentials
Verify Identity and Credentials
Provide Sovereign Identity
Decentralized Identity Service
Identity Register and Up Chain
BlockChain
(Identity/Credential Management、Data Storage)

Figure 1. The structure of the decentralized identity authentication system.

the work of this paper and the future research plan.

2. Related Work

Since entering the digital age, identity authentication has faced the challenges of high-frequency requests, massive data, privacy and security, and emerging new digital scenarios. Traditional paper credentials are facing electronic requirements, but current electronic credentials such as bus cards, medical insurance cards, membership Cards, etc. face the problems of data isolation, easy loss, insecurity, and privacy leakage. The emergence of verifiable credentials makes up for these deficiencies. The development of digital credentials depends on and serves the development of digital identities. With the development of centralization, alliance and self-sovereign identities of digital identities, the latest evolution direction of digital credentials is verifiable credentials, which are implemented through encryption algorithms and digital signatures. The validity and portability of physical credentials are transferred to digital credentials, and the declared content, signature, and metadata can be digitally digitized within seconds or even milliseconds.

David Chaum first proposed to build an anonymous electronic credential system with digital signatures, and use blind signatures to achieve payment non-traceability, aiming to protect user privacy [8]. However, Chaum did not give the specific implementation scheme of the system in this article, but proposed to implement the system with RSA digital signature and a semi-trusted third party in a later article [9].

WeIdentity [10] of WeBank implements a set of distributed multi-center identity identification protocols that conform to the w3c did specification based on the underlying platform of the fisco-bcos blockchain, enabling the real identity of entities to realize the identity identification on the chain at the same time. Give the entities the ability to directly control its own identity. WeIdentity is a complete set of decentralized identity authentication system, but WeIdentity puts most of the business logic on the centralized server to complete, the blockchain only stores data as a distributed database, and there is a problem of limited degree of decentralization.

David Bauer proposed to use the Merkle tree method to implement Verifiable Credentials to minimize information leakage [111]. While this post doesn't mention DIDs, the approach to implementing verifiable credentials using Merkle trees is instructive. This credential does not contain the user's name and other data that directly reveal the user's identity. The private part of the credential contains the user's private key and a Merkle tree whose leaves are all "micro-claims" of the user's identity. Users can "show" credentials issued by different institutions according to the requirements of the verifier, and the structure used to save these credentials is the Merkle tree. Using the verifiable credentials implemented by Merkle, the author implements a prototype system. The results show that 200 authentications per second can be achieved using this system, which is relatively fast.

W3C further standardized the standard of verifiable credentials [1] and formulated a unified specification of verifiable credentials in JSON format. This specification provides a standard specification for web transport credentials that is cryptographically secure, privacy-preserving, and machine-verifiable. The design scheme of this article refers to the credential specification of w3c, which is feasible. This specification defines that the credential should contain: the identity information of the credential subject, the information of the issuing authority, the credential type, the attributes of the credential subject, the credential export evidence, the credential expiration time, etc.

Nan Guo *et al.* proposed an anonymous credential based on BLS signature, which can aggregate different credentials into one credential [12]. The aggregate signature of credential is shorter and the verification speed is faster, suitable for small devices. The anonymous verification speed credential in the article has constant complexity about the exponent and the number of pairs, and is more efficient. However, this article just uses BLS aggregate signature to aggregate multiple credentials into one credential, and does not implement selective disclosure to achieve the purpose of privacy protection. Our scheme uses BLS aggregated signatures to implement attribute aggregation and credential aggregation after selective disclosure, which can achieve fine-grained verification and privacy protection.

3. Preliminaries

3.1. Decentralized Identifiers and Verifiable Credentials

A Decentralized Identifier (DID) [13] provides a verifiable and decentralized means for interacting with a DID Subject controlling the DID. A DID can be resolved to a DID Document, which can contain cryptographic material, verification methods, and service endpoints. An example DID is "did:did-name: WRfxPg8dantKVubE3HX8pw", where "did" tells us that it is a DID, "did-name" is the DID Method Name for Sovrin DIDs, and "WRfXPg8dantKVubE3HX8pw" identifies the DID subject.

The International Electrotechnical Commission defines "identity" as "a set of attributes associated with an entity". A digital identity is usually represented by an identity identifier and an attribute claim associated with it. A distributed digital identity includes two parts: a distributed digital identity identifier and a digital identity credential.

"Claims" refers to attribute information associated with an identity, a term that originated in claims-based digital identity, a way of asserting a digital identity independently of any particular system that needs to rely on it. Declaration information usually includes: such as name, email address, age, occupation, etc. Claims can be issued by an identity owner (such as an individual or organization) themselves or by other claims issuers, and are called verifiable claims when they are checked out by the issuer. The user submits the claim to the relevant application, the application checks it, and the application service provider can trust the verifiable claim signed by it just like the issuer. Credentials are a collection of multiple claims.

Verifiable credential provides a specification to describe certain properties of an entity to achieve evidence-based trust. DID holders can prove to other entities that certain attributes of themselves are credible through verifiable claims. At the same time, combined with cryptographic technologies such as digital signatures and zero-knowledge proofs, the statement can be made more secure and credible, and user privacy can be further protected from being violated.

Issuer is an institution that owns user data and can issue verifiable credentials based on user data, such as governments, banks, schools, and so on. The holder is user, which can apply for a verifiable credential from the issuer, and then hold and keep the credential, such as in a wallet, and show the credential to the verifier if necessary. Verifier receives the credentials presented by the user, and can provide corresponding services to the user according to the credentials. In addition, an identifier registry (Verifiable Data Registry) is also required. The identifier registry is a database that maintains DIDs, such as a blockchain or distributed ledger, which can be understood as the example field in the aforementioned DID. The Verifiable Data Registry is needed because the validator needs to validate the credentials, as well as the user.

Figure 2 shows the relationship between credential and proof. As shown in the figure, a credential consists of fields like claims, type, issuanceDate and the signature value of proof is obtained by signing all these fields of credential in traditional scheme. In addition, proof also includes some information describing the signature, such as signature algorithm, signature date, nonce and so on. The cryptographic mechanism used to prove that the information in a verifiable credential or verifiable presentation was not tampered with is called a proof. In general, when verifying proofs, implementations are expected to ensure the proof is available in the form of a known proof suite.

3.2. BLS Aggregate Signature

The BLS aggregate signature is based on the computational CDH problem and bilinear mapping. Before introducing the BLS signature, the co-GDH scheme is introduced first. Single signature in the aggregate signature is created and verified as signatures in their co-GDH scheme and the aggregate verification uses a bilinear map on G_1 and G_2. A flow chart of aggregate signature is shown in Figure 3.

1) GDH Groups and Bilinear Mapping

Computational co-Diffie-Hellman (co-CDH): Given $g_1, g_1^a \in G_1$ and $h \in G_2$, compute $h^a \in G_2$.

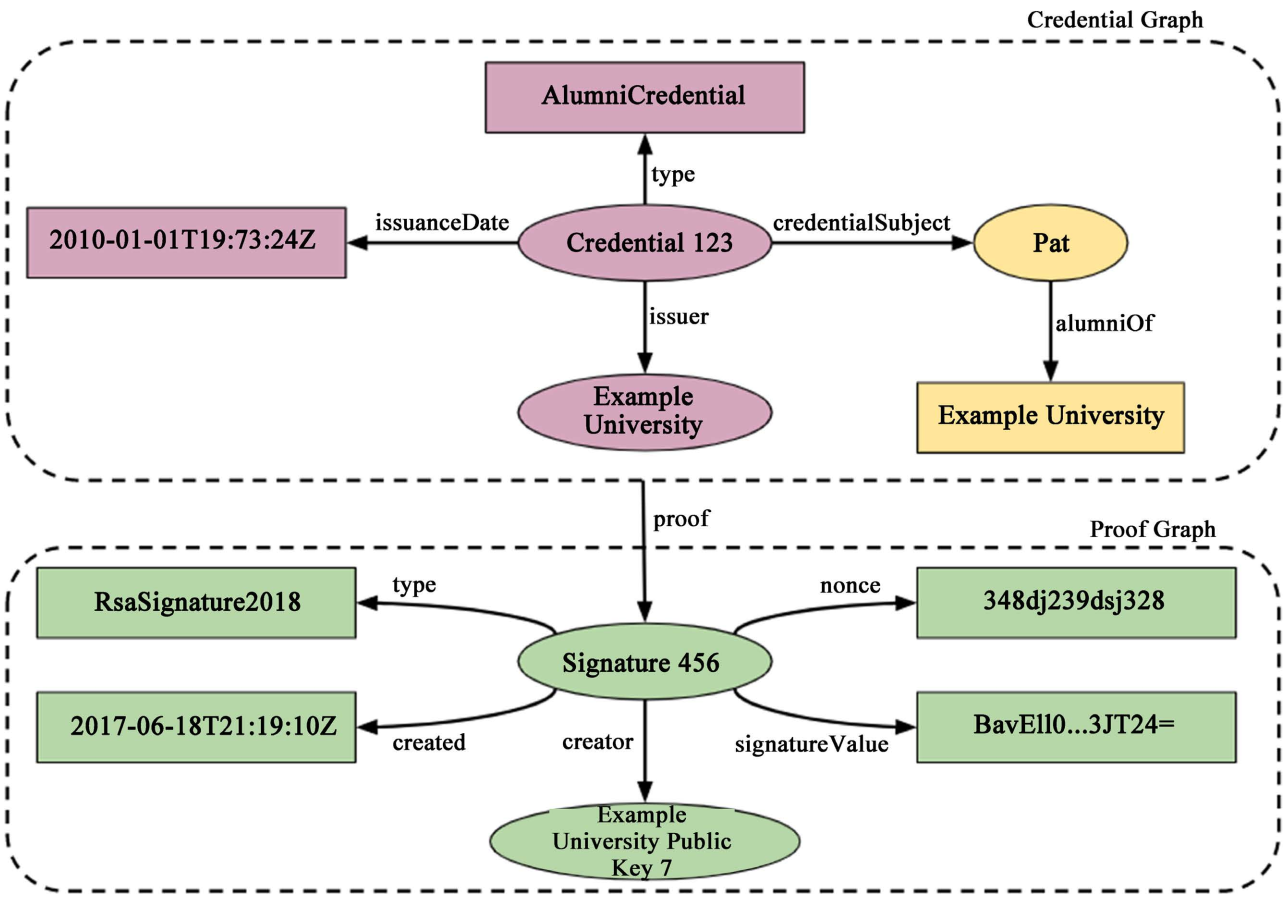

Figure 2. Credential and proof [1].

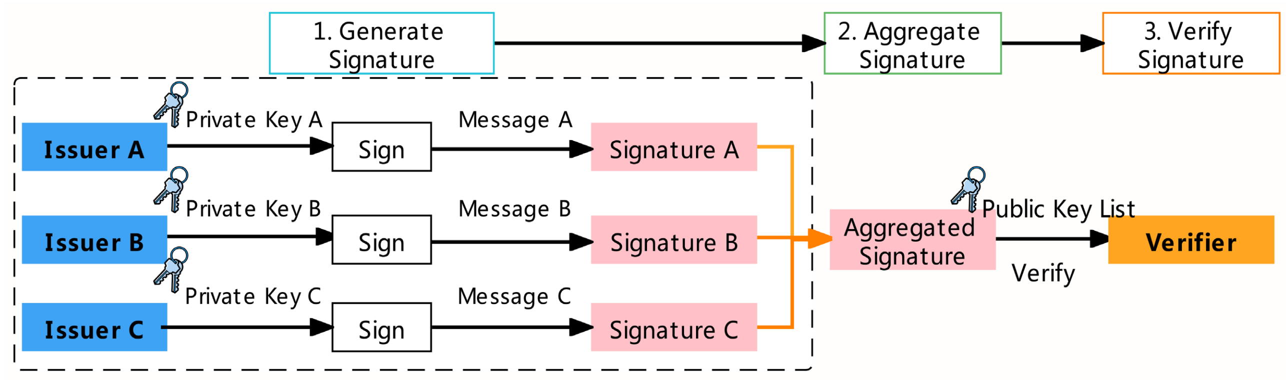

Figure 3. The process of aggregate signature.

Decision co-Diffie-Hellman (co-DDH): Given $g_1, g_1^a \in G_1$ and $h, h^b \in G_2$, if $a = b$, $\left(g_1, g_1^a, h, h^b\right)$ is a co-Diffie-Hellman tuple.

Gap co-Diffie-Hellman (co-GDH) Group Pair: Groups G_1, G_2 are co-GDH groups if they are decision groups for co-Diffie-Hellman and no algorithem breaks Computational co-Diffie-Hellman on them.

Let G_1, G_2 be two groups as above, with an additional group G_T such that $|G_1| = |G_2| = |G_T|$. A bilinear map is a map e: $G_1 \times G_2 \to G_T$ with the following properties:

a) Bilinear: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}$, $e\left(u^a, u^b\right) = e(u, v)^{ab}$.

b) Non-degenerate: $e(g_1, g_2) \neq 1$

These properties imply two more: for any $u \in G_1$, $v_1, v_2 \in G_2$, $e(u, v_1 v_2) = e(u, v_1) \cdot e(u, v_2)$; for any $u, v \in G_1$, $e(u, \psi(v)) = e(v, \psi(u))$.

Based on the above definitions, the bilinear group pair is defined as follows:

a) G_1, G_2 are bilinear groups if the group action on either can be computed in one time unit, the map ψ from G_1 to G_2 can be computed one time unit, a bilinear map $e: G_1 \times G_2 \to G_T$ exists, and e is computable in one time unit.

b) G_1, G_2 are (t, ε)-bilinear groups for co-CDH if they are bilinear groups and no algorithm (t, ε)-breaks computational co-CDH on them.

2) The co-GDH Signature Scheme

The signature scheme works on any co-GDH group pair G_1, G_2. It signs an arbitrary message $M \in \{0,1\}^*$ by using a full-domain hash function $h: \{0,1\}^* \to G_1$, viewed as a random oracle; and comprises the following algorithms:

Key Generation: Pick random $x \in_R \mathbb{Z}_p$, and compute $v = g1^x \in G_1$. The public key is $v \in G_1$. The secret key is $x \in \mathbb{Z}_p$.

Signing: Given a secret key x and a message $M \in \{0,1\}^*$, compute $h = h(M)$ where $h \in G_2$, and $\sigma = h^x$. The signature is $\sigma \in G_2$.

Verification: Given a public key v, a message $M \in \{0,1\}^*$, and a signature σ, compute $h = h(M)$ and verify that (g_1, v, h, σ) is a valid co-Diffie-Hellman tuple. If so, output valid; if not, output invalid.

4. Scheme Design

Verifiable credential scheme is based on digital signatures. Since digital signatures have the function of guaranteeing the integrity and non-repudiation of a certain message. For trust endorsement, the verification agency verifies the digitally signed credential with the secret key of the issuer. Since a credential contains multiple claims, the traditional credential is to hash all the claims and splicing them together, and then put them into the claim filed of credential's json format. Then compute the hash of the credential as specified in the credential schema and use issuer's secret key to sign the hash of the credential as signature.

In order to realize the selective disclosure of the claims of the credential, this scheme changes the traditional scheme from signing the credential to signing the hash of the splice of claim and DID. But this will cause the problem of excessive signature volume, so we adopts BLS aggregate signature to aggregate claims' signatures, so as to reduce the size of signatures.

In this section, the proposed BLS-based verifiable credentials scheme will be described in detail. To give a better understanding, the main notations will be listed in Table 1.

4.1. Scheme Overview

As shown in Figure 4, our scheme consists of the following steps. The holder applies for credentials from the issuer and the issuer signs the existing claims of the holder with its own private key, and then issues credentials to the user. After the user receives the credential, he stores the credential on the blockchain or stores it himself. When holder needs to apply for relevant services or verification, verifier will ask holder to present relevant credentials. Holder takes out one or more previously stored credentials, generates a presentation through selective disclosure and claims signature aggregation, and sends the presentation to verifier. The system will aggregate multiple presentations from different users into one presentation. Verifier finally only needs to verify the aggregate presentations with the public keys of different issuers.

4.2. Specific Application Scenarios

As shown in Figure 5, the following is a specific application scenario of verifiable aggregate credentials. Suppose an employer needs to collect resume credentials of different employees. The resume credentials must include the job applicant's school, college, major, and the position or other information, which is

Table 1. Notations.

Notation	Definition
v_i	The public key of issuer i
x_i	The secret key of issuer i
DID_i	The decentralized identifier user i
$Claim_i$	The Claim of user i
$Presentation_i$	The Presentation of user i
$Credential_i$	The Credential of user i
$Proof_i$	The Proof of user i
$Credential_{ij}$	The j-th Credential of user i
v_{ij}	The public key to verify claims in $Credential_{ij}$
x_{ij}	The secret key to sign claims $Credential_{ij}$
$Claim_{ijk}$	The k-th Claim of the j-th Credential of user i
σ_{ijk}	The signature of k-th Claim of the j-th Credential of user i

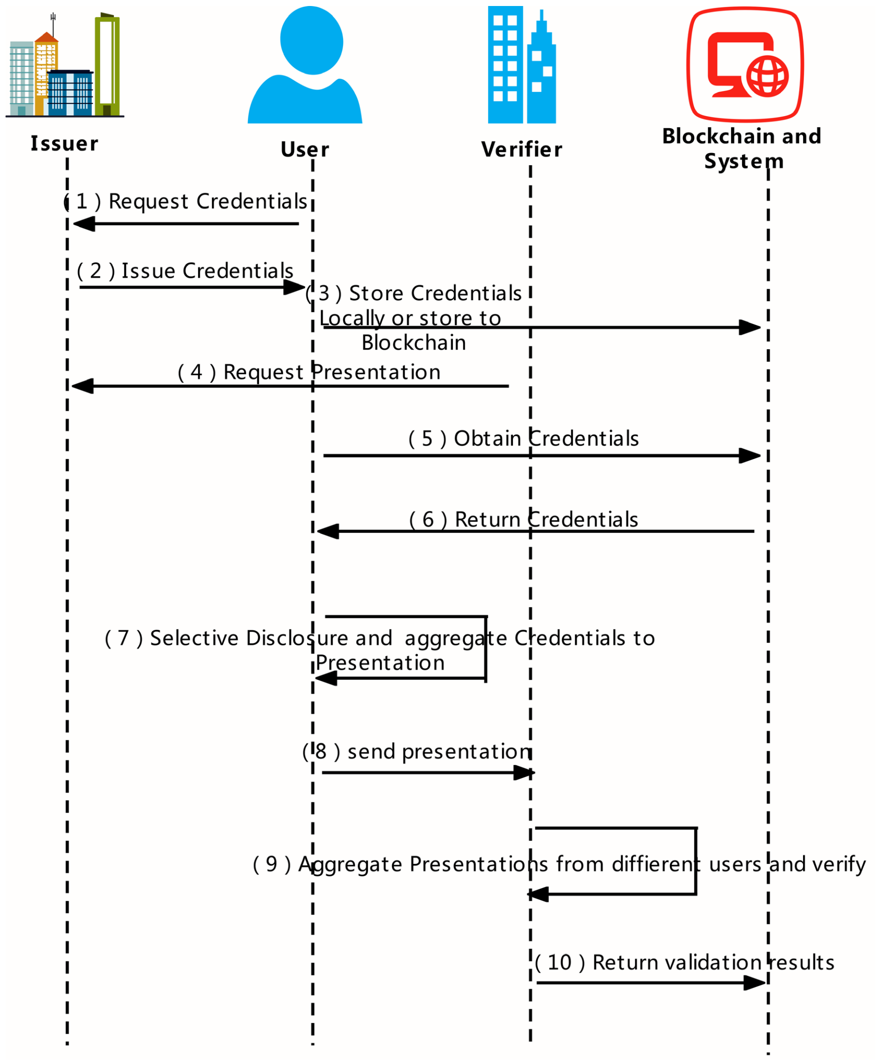
Issuer
User
Verifier
Blockchain and System
(1) Request Credentials
(2) Issue Credentials
(3) Store Credentials Locally or store to Blockchain
(4) Request Presentation
(5) Obtain Credentials
(6) Return Credentials
(7) Selective Disclosure and aggregate Credentials to Presentation
(8) send presentation
(9) Aggregate Presentations from diffierent users and verify
(10) Return validation results

Figure 4. Process of Issue and verify the credentials.

called claims. Employees need to obtain academic credentials from schools, work proof credentials from the previous company, and ID credentials from government departments. But there are many claims in the credentials issued by these agencies that the employers do not need, and the employees do not want to show all the claims to the recruiter. Our scheme allows employers to choose a part of claims of multiple credentials and aggregate their signatures into one credential, which is called presentation. Therefore, employees can use our system to aggregate the academic credentials, ID credentials, and work proof credentials into resume credentials, so as to achieve the purpose of multiple credential aggregation and selective disclosure. In addition to aggregating multiple credentials of an employer into a presentation, our solution also supports aggregating the presentations of multiple employees into a single presentation. After an employer receives presentations from multiple employees, he can aggregate them into a single presentation in the verifiable credential system. In our example, the employer (verifier) only needs to verify the aggregate presentation to verify the resume credentials from

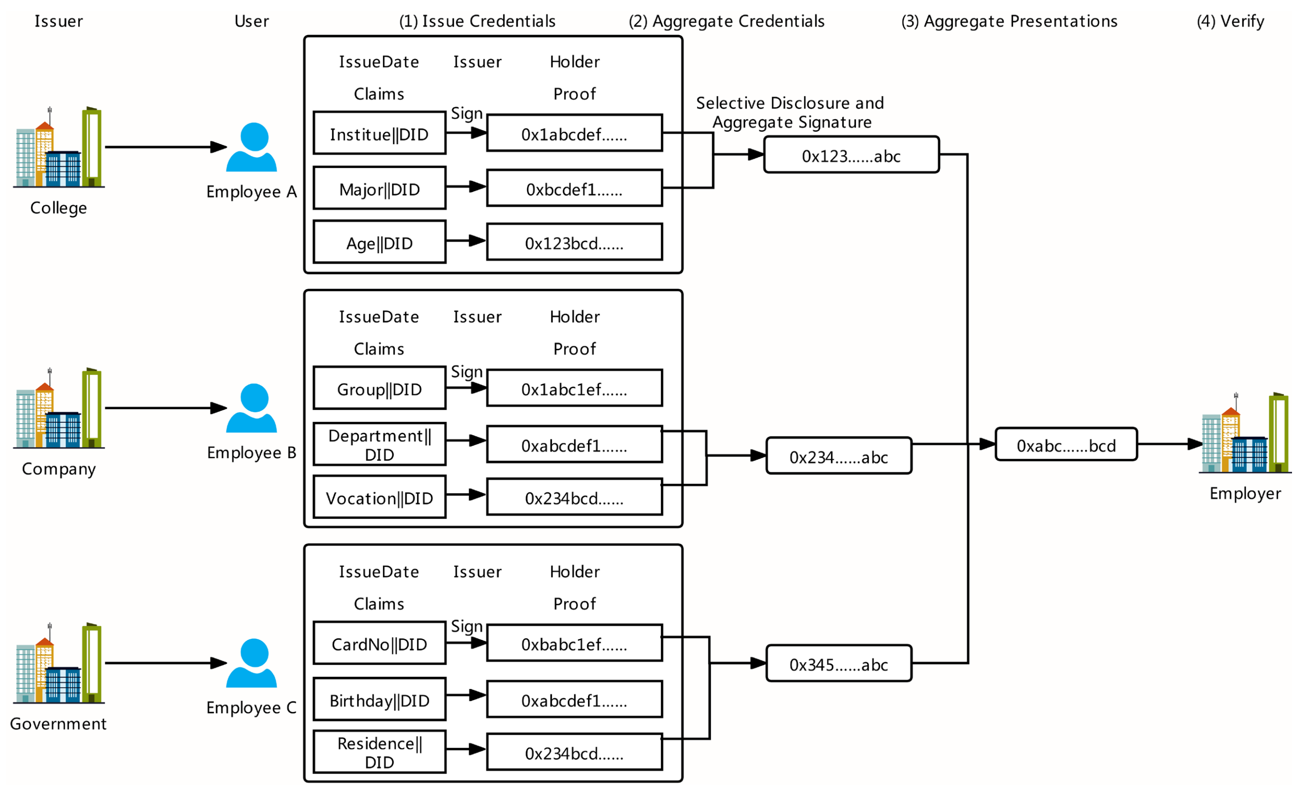

Figure 5. Overall architecture.

multiple employees, so as to achieve the purpose of batch verification of credentials.

4.3. Selective Disclosure and Credentials Aggregation

The issued credential is stored on the blockchain. When the user needs to show it, he selects some claims of the required credential, and then puts claims of the credential into the presentation after the aggregate signature, and then only needs to selectively disclose to the verifier. When presenting a credential to a verifier, we often do not need to present all the claims in the credential. For example, a bartender may only require the holder to provide claims that meet the age requirement of less than 18 years, and not other claims. The user can select the claims that need to be presented from the obtained credentials, and aggregate these claims into a signature, which becomes a presentation. The presentation contains only one signature, which is the signature that the verifier needs to verify. By verifying the aggregate signature, all the signatures of claims can be verified. As shown in Figure 6, the user uses the "school" and "institute" claims of credential A, and the "occupation" claim of credential B respectively. These information are spliced with the user's DID, hashed, and then generated by BLS signature, namely $Sig_{Sk}\left[Hash(Claim\|DID)\right]$. The reason why the DID is added after the claim is that if the DID is not added, the attacker can steal the claim signature of the victim's credential, so as to put the signature of the claim in his own credential, and state that the claim is owned by him. Therefore, DID is used

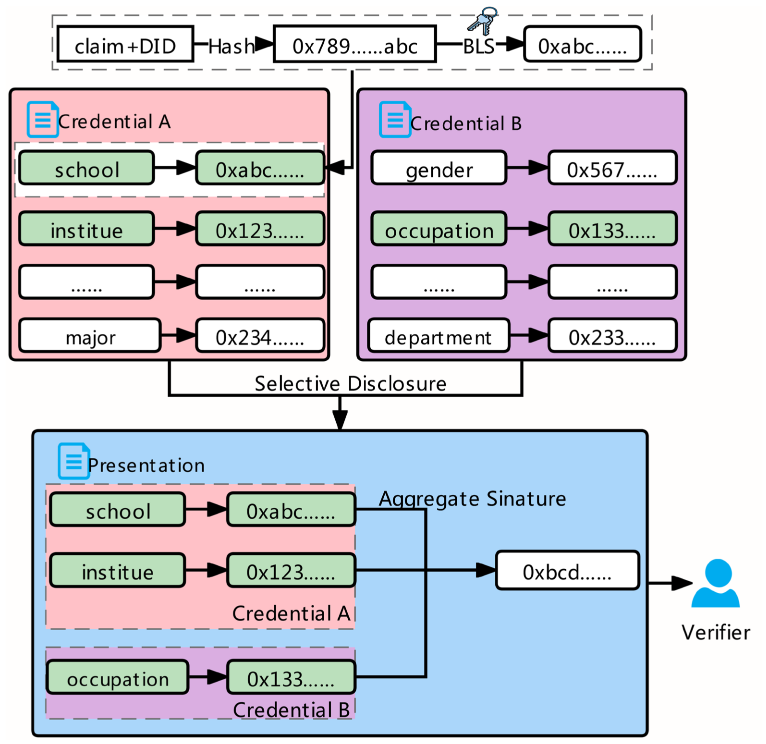

Figure 6. Selective disclosure.

to bind with user claim to prove that the user has a certain claim.

A credential consists of a hash value, a public key list, etc., which are spliced with multiple claims. The information of the credential must include the holder did, proof information (attribute signature information), issue date, issuer information, credential type, etc.

First, issuer will issue a credential for the user, which contains the signature (stored in the proof field), the DID of credential, creation date of credential, signature algorithm and other information, in which the information of the credential is all stored in json format. During the presentation stage, the user aggregates the claims in the signatures which include signatures issued by multiple issuers. The BLS aggregate signature is used for the aggregation here. BLS aggregate signature can not only aggregate the signature information of the claims in different credentials into a new credential, which is called presentation, as shown in Figure 7. When verifying, the verifier only needs to verify the aggregate signature in the presentation. BLS aggregate signatures can not only aggregate signatures of different claims in different credentials of the same user, but also aggregate signatures in presentations from different users. Verifier only needs to verify the aggregate signature to verify the presentation of multiple users. It is not necessary to sign all the messages of the credential (such as issuer's DID, credential type, etc.), signing claims is enough to verify.

4.4. Presentations Aggregation

In addition, our system supports aggregation of presentations (aggregate credential) from different users. As shown in Figure 8, if a verifier needs to verify presentations presented by multiple users, he does not need to verify all the resentations of each user one by one. Our solution also implements the aggregation of the presentations of different users. The user displays his presentations, and the system will automatically aggregate these presentations to one presentation, which will be verified by the verifier at last. That is, take out signature of the proof field in each user's presentations, aggregate these signatures into one signature, and put them into the proof field of a new presentation. Verifier only needs to verify the signature in this new presentation.

4.5. BLS Aggregate Signature for Credential

Both of the above two aggregation methods require the use of aggregated signatures. For the aggregate signature algorithm, we choose the BLS aggregate signature,

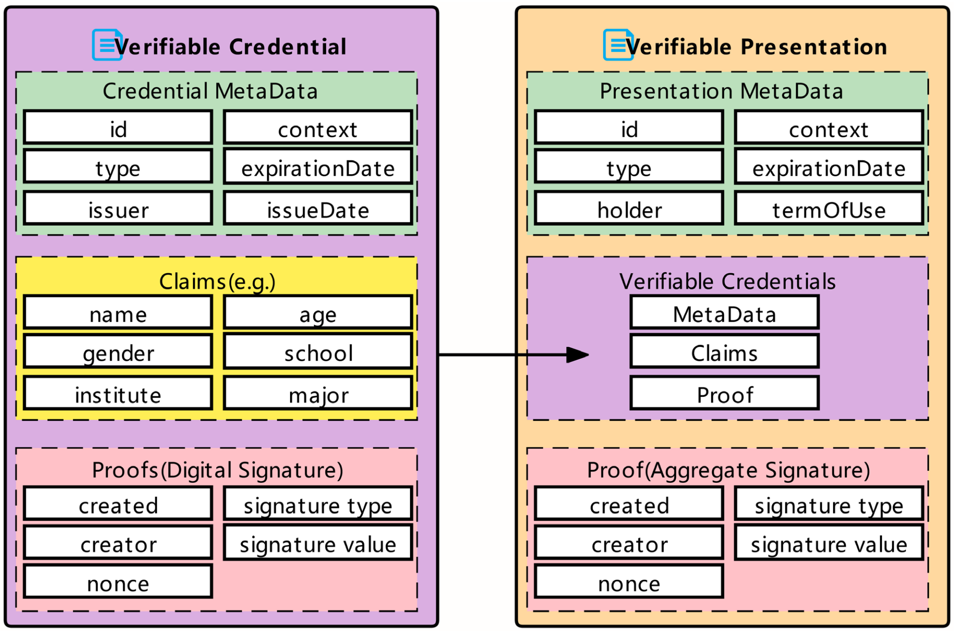

Figure 7. How credential become presentation.

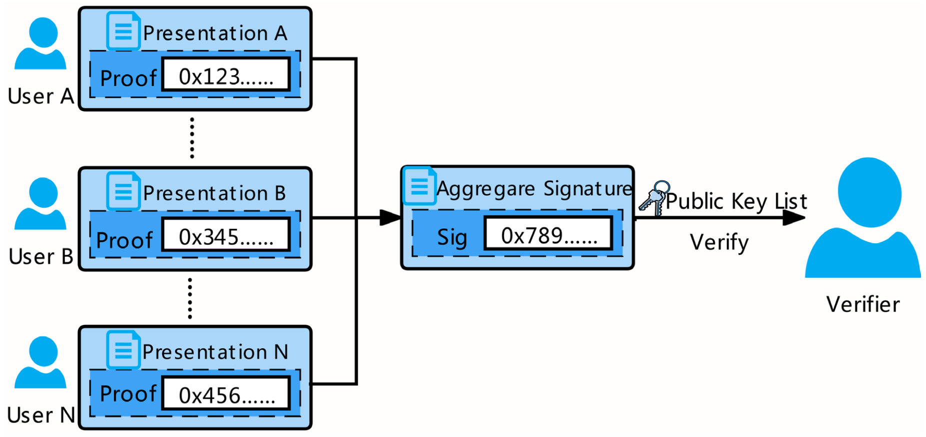

Figure 8. Presentations aggregation.

which can aggregate multiple signatures into one signature, reduce the size of the signature, and facilitate verification operations.

(1) The BLS signature algorithm is used to sign the Hash value of the credential claim plus the DID. The signing process is divided into the following steps:

System initialization: Choose multiplicative cyclic groups G_1, G_2, G_T with a prime *p* generated by g_1, g_2, g_T respectively. Define a bilinear map $e: G_1 \times G_2 \to G_T$, and hash functions $H: \{0,1\}^* \to G_1$.

Issuer Setup: For each claim managed by the $Issuer_i$, choose a random number $x_i \in \mathbb{Z}_p$ as the secret key of $Issuer_i$. The public parameter is set as $(G_1, G_2, G_T, e, p, g_1, v)$, the secret key x_i is stored by $Issuer_i$.

Sign Claims: This is an important step in issuing credentials to users. Assuming that the private key of the issuer is *x*, the issuer needs to declare *Claim* for a user's claim during the process of issuing the credential, the user's decentralized identifier is *DID*. Compute $h = H(Claim \| DID)$, $h \in G_2$, the signature is $\sigma = h^x$, $\sigma \in G_2$, the signature σ is stored in the proof field of credential.

Aggregate Cliams's signatures into Presentation: The most general assumption is made here, and other situations are similar to this assumption. Suppose $User_i$ has credentials issued by different issuers $Issuer_j$ (whose secret key is x_{ij}), marked as $Credential_{ij}$, the *k*-th claim contained in $Credential_{ij}$ is marked as $Claim_{ijk}$. The decentralized identifier of $User_i$ is DID_i. Therefore, sign the claim $Claim_{ijk}$ in the $Credential_{ij}$ (issued by $Issuer_j$) of $User_i$ (with the decentralized identifier DID_i), compute $h_{ijk} = H(Claim_{ijk} \| DID_i)$, $h_{ijk} \in G_2$, the signature is $\sigma_{ijk} = h_{ijk}^{x_{ij}}$, $\sigma_{ijk} \in G_2$, the signature σ_{ijk} is stored in the proof field of $User_i$'s credential $Credential_{ij}$. Suppose $User_i$ selects claims $Claim_{ijk}$ in $Credential_{ij}$ and wants to disclose the selected claims to Verifier, we need to aggregate the claims' signatures σ_{ijk} of $User_i$ from different Issuer $Issuer_j$. The aggregate signature is $\sigma_i = \prod_j \prod_k \sigma_{ijk}$, aggregate signature σ_i is stored in the proof field of $Presentation_i$, and then we store $Presentation_i$ into Blockchain.

Aggregate presentations from different user: After the previous step, the signatures of claims from different credentials of a user have been aggregated in one Presentation. In this step, we want to aggregate the signatures from presentations of different users, that is, to aggregate the claim aggregation signatures from different users placed in the proof field of the presentation. Aggregate the signatures of presentations, $\sigma = \prod_i \sigma_i = \prod_i \prod_j \prod_k \sigma_{ijk}$, marked $proof = \sigma$ and put $proof$ into the aggregate presentation, which is later stored in blockchain.

Verify: Given the final aggregate signature $\sigma \in G_2$, the public key of the issuer $v_{ij} \in G_1$, the hash value h of the set of claims $Claim_{ijk}$ that need to be verified. To verify the aggregate signatrue σ, compute $h_{ijk} = H(Claim_{ijk} \| DID_i)$, if Equation (1) is true, then the signature σ is accepted and credentials are correctly verified.

$$e(g_1, \sigma) = \prod_i \prod_j \prod_k e(v_{ij}, h_{ijk}) \tag{1}$$

The calculation process of aggregate verification is as follows:

For different issuers' secret key x_{ij} and public key $v_{ij} = g_1^{x_{ij}}$, signature $\sigma_{ijk} = h_{ijk}^{x_{ij}} = H\left(Claim_{ijk} \| DID_i\right)^{x_{ij}}$, aggregate signature $\sigma = \prod_i \sigma_i = \prod_i \prod_j \prod_k \sigma_{ijk} = \prod_i \prod_j \prod_k h_{ijk}^{x_{ij}}$. Using the properties of the bilinear map, the verification equation is shown in formula (2).

$$\begin{aligned} e\left(g_1,\sigma\right) &= e\left(g_1,\prod_i\prod_j\prod_k h_{ijk}^{x_{ij}}\right) = \prod_i\prod_j\prod_k e\left(g_1,h_{ijk}\right)^{x_{ij}} \\ &= \prod_i\prod_j\prod_k e\left(g_1^{x_{ij}},h_{ijk}\right) = \prod_i\prod_j\prod_k e\left(v_{ij},h_{ijk}\right) \end{aligned} \tag{2}$$

5. Security Analysis

In this section, we will analyze the security related to our scheme, which focuses on aggregate signature forgery attack and identity theft attack.

5.1. Aggregate Signature Forgery Attack

In order to analyze and define security, we generally believe that the security of the aggregate signature scheme is equivalent to the fact that in a certain game range, there is no adversary who can forge aggregate signatures, that is, there is no adversary trying to forge on the message of his choice. This security problem is defined as the chosen-key security model of aggregate signatures, in which, given a single public key of Adversary $\mathcal{A}$, the goal of Adversary $\mathcal{A}$ is to forge an aggregate signature. Adversary A also gives access to the challenge keys of other signing oracles. Adversary $\mathcal{A}$'s advantage, $AdvAggSig_{\mathcal{A}}$, is defined as his probability of winning the following challenge [13].

Setup: The Adversary $\mathcal{A}$ is provided with a public key $PK_{\mathcal{A}}$ which is generated at random.

Queries: Proceeding adaptively, Adversary $\mathcal{A}$ requests signatures with $PK_{\mathcal{A}}$ on messages of his choice.

Response: Finally, $\mathcal{A}$ outputs $k-1$ additional public keys $PK_2,\cdots,PK_k$. Here k is at most *N*, a game parameter. These keys, along with the initial key $PK_{\mathcal{A}}$, will be included in $\mathcal{A}$'s forged aggregate. $\mathcal{A}$ also outputs messages $M_1,\cdots,M_k$; and, finally, an aggregate signature σ by the k users, each on his coressponding message.

The forger wins if the aggregate signature σ is a valid aggregate on messages $M_1,\cdots,M_k$ under keys $PK_{\mathcal{A}},PK_2,\cdots,PK_k$, and σ is nontrivial, *i.e.*, $\mathcal{A}$ did not request a signature on M_1 under $PK_{\mathcal{A}}$. The probability is over the coin tosses of the key-generation algorithm and of $\mathcal{A}$.

The adversary $\mathcal{A}$'s ability in the chosen-key model to generate keys suggests the following attack, previously considered in the context of multi-signatures [14] [15]. In the context of aggregate signature, we can defend against the attack above by simply requiring that an aggregate signature is valid only if it is an aggregation of signatures on distinct messages. The signature of our scheme will add the users' DID after the claims. Because the user's DID is unique, the content of the signature is also distinct. Therefore, our scheme is to chosen-key security.

5.2. Identity Theft Attack

Suppose that there is a phishing attack verifier, after obtaining the signature of a certain attribute of the user, it is used for verification to the real verifier, thereby stealing a user's verifiable credential [16]. Assuming that the attribute is *Claim* and the user's identity identifier is *DID*, the claim and identity identifier are signed, that is, $h_1 = H(Claim \| DID)$, $\sigma = h_1^x$. The signature obtained by the verifier of the phishing attack is either a single signature, that is, a σ, or an aggregate signature. If it is an aggregate signature, the phishing attacker can only perform verification, and cannot obtain the specific value of different individual signatures. If it is a single signature, then according to the CDH difficulty problem, the phishing attacker cannot obtain the private key according to the public key of the Issuer, and the only information the phishing attacker can obtain is the public key of the Claim, DID, and Issuer. The previous step proved that the attacker cannot get the specific value through these forged signatures. Then a phishing attacker can only steal the credential, and says that the credential is owned by him. Then, when signing the claim, our scheme splices the claim and the DID and then signs it. During the verification process, the user's DID is spliced behind the claim. Suppose a claim is owned by the user whose DID is $DID_{\mathcal{A}}$, and the phishing attacker whose DID is $DID_{\mathcal{A}}$ obtains the claim signature through some means, and then uses it to show the claim signature to a verifier, claiming that he owns the attribute *Claim*, and verifying When verifying the signature of this attribute, the attacker will splice the attribute Claim with the identity identifier $DID_{\mathcal{A}}$ of the phishing attacker, and then hash and sign, that is, $h_{\mathcal{A}} = (Claim \| DID_{\mathcal{A}})$, $\sigma_{\mathcal{A}} = h_2^x$, According to our verification algorithm, it cannot be verified. Therefore, our scheme is safe against credential stealing attacks.

6. Implementation and Evaluation

We have a proof-of-concept implementation of the System, and we have identified essential libraries for realizing the VC-based PKI. Our system mainly includes two modules: DID system and Credential system. **Figure 9** shows the structure and credentials process of our system. For implementing the decentralized verifiable credentials system, we must first implement a DID system. DID system is a new type of identity system that enables verifiable, decentralized digital identity. Our system is implemented based on smart contract of Ethereum [17]. The technologies and components involved in writing our smart contract on Ethereum include: 1) Solidity, Ethereum's Turing complete programming language, used to write smart contracts [18]; 2) Ethereum Virtual Machine (EVM), that is, the environment for smart contract operation; 3) Web3 Java SDK [19],

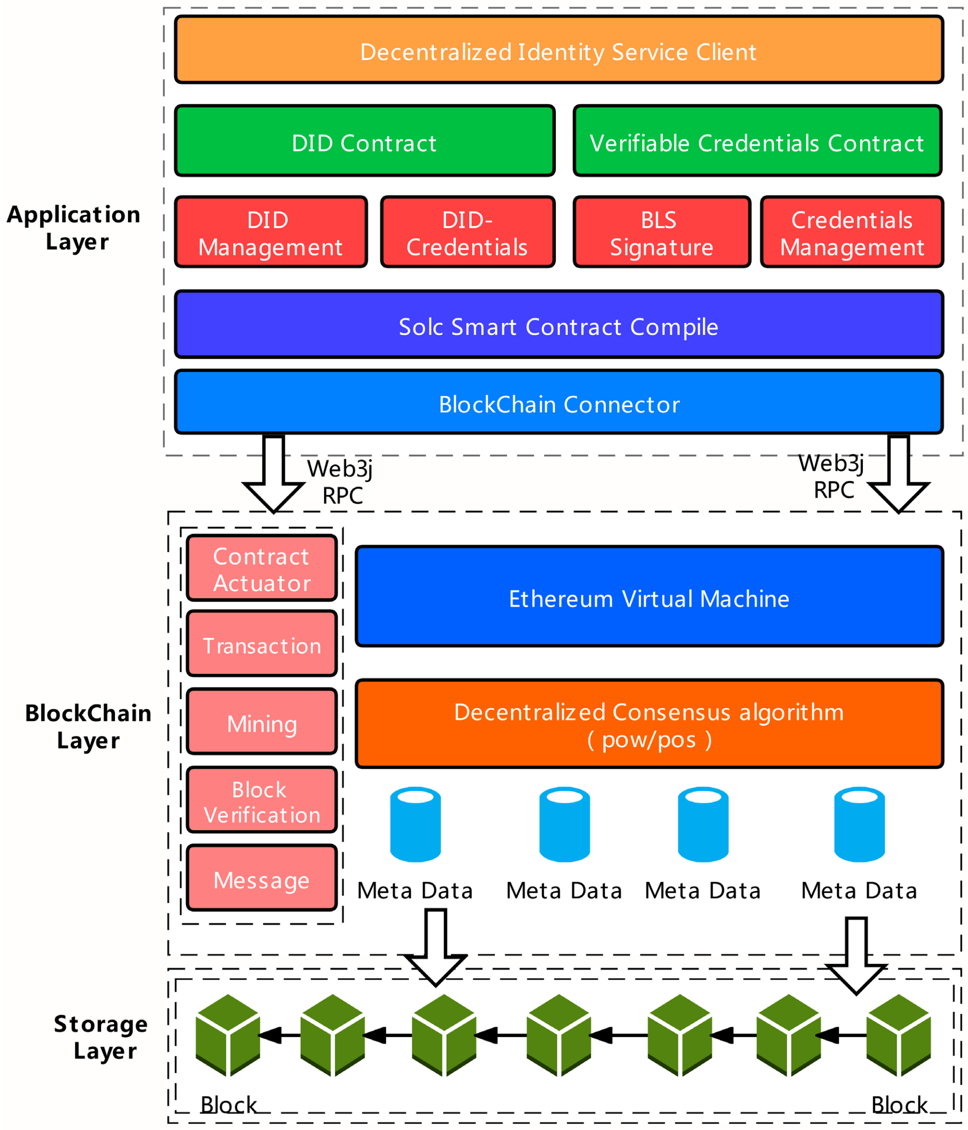

Figure 9. Implementation architecture.

Java API to interact with Ethereum blockchain nodes.

We use solidity to write smart contract, our smart contract are mainly used to store DID data and credentials. We use Java as our backend development language. There are two smart contracts in our system implementation: DID contract and Verifiable Credentials contract. DID contract is used to manage DID, includes reading, resolving, validating and updating DID. Credential-related operations are credential creation, credential signing, aggregate signing (credentials aggregation and presentations aggregation), credential verification, and aggregate verification.

For implementing the BLS Aggregrated Signature, we use JPBC [20] library of Java. JPBC is the Java Pairing-Based Cryptography Library that provides a part of the Pairing-Based Cryptography Library, library developed by Ben Lynn, to perform the mathematical operations underlying pairing-based cryptosystems directly in Java. It is a Wrapper that enables the delegation of the pairing computation to the PBC library to gain in performance. We implement the process of BLS signature: initialization, signature, aggregate signature, verification, aggregate verification.

6.1. Experimental Analysis

In this section, we used computational cost as a metric to analyze the performance of our scheme. We have implemented BLS Signature into an open-source library JPBC. We used the JPBC library Ver. 2.0.0 as an implementation of cryptographic operations. JPBC provides several implementations of elliptic curves, we use the elliptic curve of $Type_A$, $Type_A$ parings are constructed on the curve $y^2 = x^3 + x$ over the Field $\mathbb{F}_q$ for prime *q*. Both G_1 and G_2 are the group of points $E\left(\mathbb{F}_q\right)$, so this paring is symmetric. The experimental test runs in the Windows 10 environment, and the specific configuration is Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz 2.40 GHz, 8 GB memory.

6.1.1. Efficiency

In order to verify the practical value of our scheme, the experiment mainly tests the performance of BLS aggregate signatures and the performance of ECDSA [21].

ECDSA elliptic curve digital signature algorithm is currently the most mainstream digital signature algorithm, the combination of ECC and DSA, the entire signature process is similar to DSA, the difference is that the algorithm adopted in the signature is ECC (Elliptic Curves Cryptography), and the final signed value is also divided into *r*, *s*. ECDSA digital signature algorithm is used in WeIdentity, a well-known decentralized identity authentication project developed by WeBank.

We compare the BLS aggregate signature algorithm and the ECDSA signature from the aspects of signing efficiency, verification efficiency. In our experiments, The BLS aggregation signature algorithm and ECDSA signature algorithm are both implemented in Java.

We compare the signing speed and verification speed of ECDSA signature and BLS aggregate signature when the number of signatures is 500, 1000, 1500, 2000, and 2500. In addition, we compared the verification speed of the two under the same and different key pairs.

Figure 10 shows the signature speed comparison between BLS aggregate signature and ECDSA signature. The ordinate indicates the number of signatures, and the abscissa represents the signature speed (ms). It can be seen intuitively that in terms of signing speed, the BLS aggregate signature speed (with the addition of aggregation time, but it can be ignored) is slower than ECDSA's signature speed. The signing time of BLS aggregate signature increases linearly, which has a lot to do with the exponentiation operation required in the signature process of BLS aggregate signature.

Figure 11 is the comparison of the verification speed of BLS aggregate signature

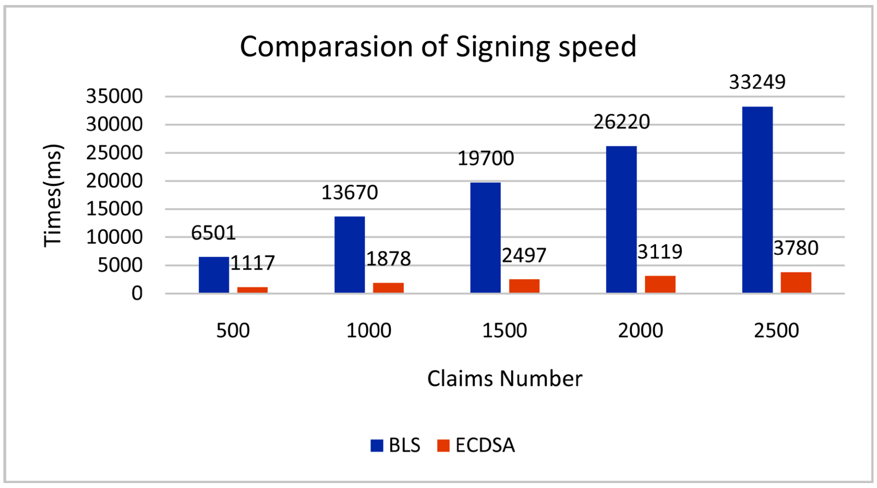

Figure 10. Comparison of signing speed between ECDSA and BLS.

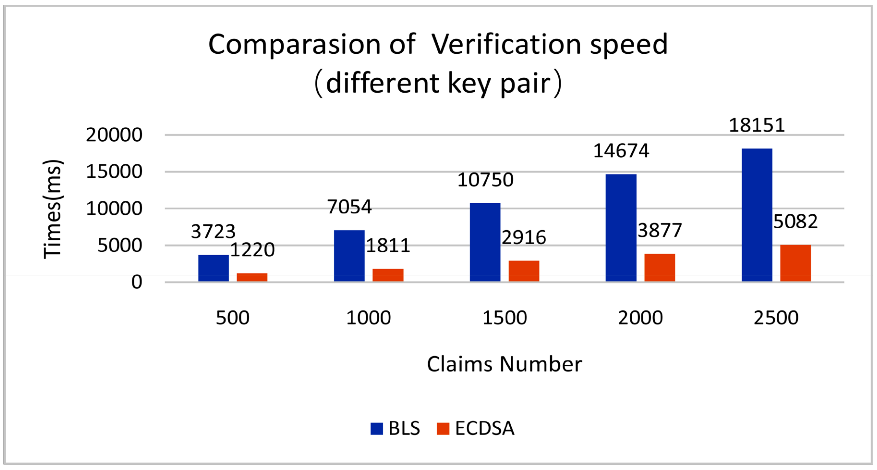

Figure 11. Comparison of verification (different signers) speed between ECDSA and BLS.

and ECDSA under different key pairs. The verification time of BLS aggregate signature is also greater than that of ECDSA. This is because the BLS aggregate signature needs to use a very time-consuming pairing when verifying. The more signatures to be aggregated, the more pairing operations need to be used, but the BLS aggregation signature is still acceptable in terms of verification speed. It takes about 7000 milliseconds to verify the signatures of 1000 signature aggregations, equivalent to only 7 ms verification time for a signature, which is acceptable in performance.

Figure 11 is the comparison of the verification speed of BLS aggregate signature and ECDSA under same key pairs. In the case of signing and verifying different messages with the same key pair, the verification speed of BLS aggregate signature is much faster than that of ECDSA signature. This is because different messages are signed with the same private key. When verifying, only one public key is needed, let σ be an aggregate of the n signature $\sigma_1,\cdots,\sigma_n$. The time to verify the aggregate signature σ is linear in n. In the special case when all n signatures are issued by the same public key v, aggregate verification is faster.

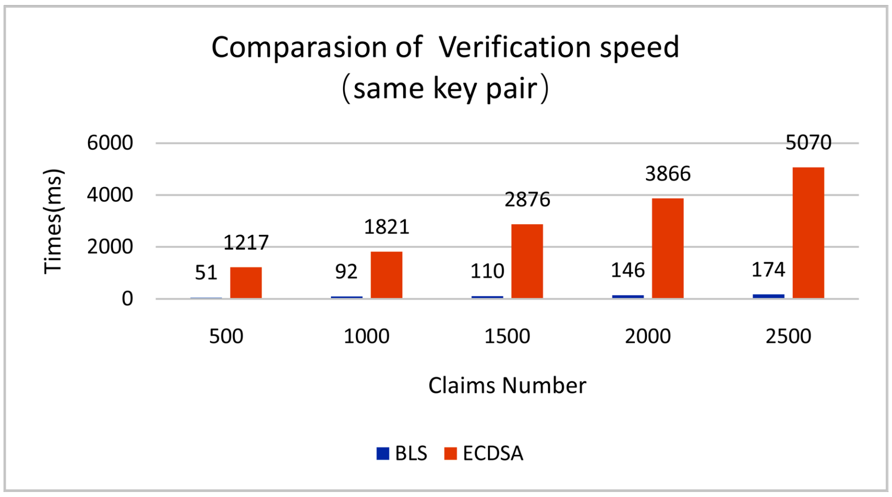

Figure 12. Comparison of verification (same signer) speed between ECDSA and BLS.

One need only verify that $e(g_2,\sigma)=e\left(\prod_i h(M_i),v\right)$, where $M_1,\cdots,M_n$ are the signed message. From this, it can be seen that using a public key to verify multiple signatures only requires two pairing operations. In fact, the verification speed of BLS depends on the number of key pairs used for different signatures when aggregating signatures. It is assumed that n signatures are aggregated into one signature, but these n signatures are generated by $k(k \le n)$ private keys. Then the verification speed is only related to k, the more pairing operation is needed, then the speed of verification is slower.

Therefore, in our solution, the BLS aggregation signature is more efficient in verifying the aggregation of credentials issued by the same issuer, and we generally do not pay too much attention to the efficiency of the signing, because it has little impact on actual projects. As shown, although slower than ECDSA, the performance of BLS signature in our scheme is still acceptable for practical use.

6.1.2. Storage Cost and Bandwidth Overhead

The verification speed of BLS aggregate signature is only one aspect, and BLS also has a full advantage in reducing the size of the signature and the public key. BLS aggregate signature can not only aggregate signatures and reduce the size of signatures, but also aggregate keys, so in the implementation, it can also reduce the size of the public key that needs to be stored. In our experiment, the single signature size of BLS aggregate signature is 128 bytes, and the key size is also 128 bytes. BLS can aggregate different signatures into one signature, and can aggregate different keys into one key, which is also 128 bytes. In our experiment, the signature size of ECDSA is 72 bytes and the key length is 72 bytes, which cannot be aggregated. Therefore, in the processing of batch signatures, the storage required by ECDSA will increase with the increase of the number of signatures, while BLS aggregate signatures do not have this concern. As a result, BLS aggregate signature is especially suitable for storage of blockchains, which can greatly reduce the storage overhead and bandwidth consumption of blockchain.

In conclusion, the ECDSA signature algorithm is good enough for its job. But

ECDSA cannot provide signature aggregation or key aggregation, so it can only verify signatures one by one. When verifying multi-signature transactions, this is too cumbersome. We need to verify all signatures and their corresponding public keys one by one, which consumes a lot of block space and transaction fees. The BLS signature algorithm can solve the above problems. It can aggregate all the signatures into one, easily implement multiple signatures verification, and avoid redundant communication between signers. In addition to that, BLS signatures are shorter in length (signatures are one point on an elliptic curve instead of two) in practice.

7. Conclusion

In this paper, we proposed a secure BLS-based verifiable credentials aggregation scheme, which is used for selective disclosure and privacy preservation on decentralized identity verification system. In our scheme, users can choose part of claims from different credentials to disclose as they want, such that verifiers cannot obtain user information excessively. BLS aggregate signature provides aggregation for signatures of claims, which reduces storage cost and network loan overhead in blockchain. In addition, our aggregation verification scheme is also used to aggregate the credentials of different users, which can effectively reduce the size of the credential signature further, and facilitate the verifier to perform batch verification operations on the credentials of multiple users. Next, we analyzed the performance of the proposed protocol to show the satisfying features in both security and efficiency. In addition, experiments show that the smaller the number of signers, the faster the verification speed. In future work, we will focus on reducing the verification speed of aggregate signatures in the case of multiple signers. Meanwhile, we will introduce zero-knowledge range proofs to strengthen privacy preservation further on the basis of selective disclosure.

Acknowledgements

This work was in part supported by National Natural Science Foundation of China (Grant No. 61932011), Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019B1515120010), Guangdong KeyR&D Plan2020 (No. 2020B0101090002), National KeyR&D Plan2020 (No. 2020YFB1005600).

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] W3C-VC (2021) Verifiable Credentials Data Model 1.0. Technical Report. https://www.w3.org/TR/vc-data-model

[2] Takemiya, M. and Vanieiev, B. (2018) Sora Identity: Secure, Digital Identity on the Blockchain. 2018 *IEEE 42nd Annual Computer Software and Applications Conference*, Volume 2, 582-587. https://doi.org/10.1109/COMPSAC.2018.10299

[3] Brunner, C., Gallersdörfer, U., Knirsch, F., Engel, D. and Matthes, F. (2020) DID and VC: Untangling Decentralized Identifiers and Verifiable Credentials for the Web of Trust. 2020 *the* 3*rd International Conference on Blockchain Technology and Applications*, Xi'an, 14-16 December 2020, 61-66. https://doi.org/10.1145/3446983.3446992

[4] García-Rodríguez, J., Moreno, R.T., Bernabé, J.B. and Skarmeta, A. (2021) Towards a Standardized Model for Privacy-Preserving Verifiable Credentials. *The* 16*th International Conference on Availability, Reliability and Security*, Vienna, 17-20 August 2021, 1-6. https://doi.org/10.1145/3465481.3469204

[5] Chen, Y.-C., Tso, R., Mambo, M., Huang, K. and Horng, G. (2015) Certificateless Aggregate Signature with Efficient Verification. *Security and Communication Networks*, **8**, 2232-2243. https://doi.org/10.1002/sec.1166

[6] Lux, Z.A., Thatmann, D., Zickau, S. and Beierle, F. (2020) Distributed Ledger-Based Authentication with Decentralized Identifiers and Verifiable Credentials. 2020 2*nd Conference on Blockchain Research & Applications for Innovative Networks and Services* (*BRAINS*), Paris, 28-30 September 2020, 71-78. https://doi.org/10.1109/BRAINS49436.2020.9223292

[7] Boneh, D., Gentry, C., Lynn, B. and Shacham, H. (2003) Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In: *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, Berlin, 416-432. https://doi.org/10.1007/3-540-39200-9_26

[8] David, C. (1985) Security without Identification: Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM*, **28**, 1030-1044. https://doi.org/10.1145/4372.4373

[9] Chaum, D. and Evertse, J.-H. (1987) A Secure and Privacy-Protecting Protocol for Transmitting Personal Information between Organizations. In: *Conference on the Theory and Application of Cryptographic Techniques*, Springer, Berlin, 118-167. https://doi.org/10.1007/3-540-47721-7_10

[10] WeBank (2021) Weidentity. https://weidentity.readthedocs.io/zh_CN/latest/docs/one-stop-experience.html

[11] Bauer, D., Blough, D.M. and Cash, D. (2008) Minimal Information Disclosure with Efficiently Verifiable Credentials. *Proceedings of the* 4*th ACM Workshop on Digital Identity Management*, New York, October 2008, 15-24. https://doi.org/10.1145/1456424.1456428

[12] Guo, N., Cheng, J.J., Zhang, B. and Yi, K.B. (2013) Aggregate Signature-Based Efficient Attributes Proof with Pairing-Based Anonymous Credential. *IEEE* 16*th International Conference on Network-Based Information Systems*, Gwangju, 4-6 September 2013, 276-281. https://doi.org/10.1109/NBiS.2013.42

[13] W3C-DID (2021) Decentralized Identifiers (dids) v1.0. Technical Report. https://weidentity.readthedocs.io/zh_CN/latest/docs/one-stop-experience.html

[14] Boldyreva, A. (2003) Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In: *International Workshop on Public Key Cryptography*, Springer, Berlin, 31-46. https://doi.org/10.1007/3-540-36288-6_3

[15] Micali, S., Ohta, K. and Reyzin, L. (2001) Accountable-Subgroup Multisignatures. *Proceedings of the* 8*th ACM Conference on Computer and Communications Security*, Philadelphia, 5-8 November 2001, 245-254. https://doi.org/10.1145/501983.502017

[16] He, B.-Z., Chen, C.-M., Su, Y.-P. and Sun, H.-M. (2014) A Defence Scheme against

Identity Theft Attack Based on Multiple Social Networks. *Expert Systems with Applications*, **41**, 2345-2352. https://doi.org/10.1016/j.eswa.2013.09.032

[17] Wood, G. (2014) Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum Project Yellow Paper 151, 1-32.

[18] Buterin, V., *et al*. (2014) A Next-Generation Smart Contract and Decentralized Application Platform. White Paper, 3(37).

[19] Web3 (2021) Web3/web3j-docs. http://docs.web3j.io/4.8.7

[20] Gas Lab (2021) Java Pairing-Based Cryptography Library. http://gas.dia.unisa.it/projects/jpbc

[21] Johnson, D., Menezes, A. and Vanstone, S. (2001) The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, **1**, 36-63. https://doi.org/10.1007/s102070100002