An Access Control System for Verifiable Credentials with Selective Disclosure

*Note: Sub-titles are not captured in Xplore and should not be used

1st Chun-An Lin
Graduate school of Natural Science & Technology
Kanazawa University
Kanazawa, Japan
kimlin20011@gmail.com

2nd Chen-Mou Cheng
Graduate school of Natural Science & Technology
Kanazawa University
Kanazawa, Japan
cheng@se.kanazawa-u.ac.jp

3rd Masahiro Mambo
Graduate school of Natural Science & Technology
Kanazawa University
Kanazawa, Japan
mambo@ec.t.kanazawa-u.ac.jp

***Abstract*—It is important for access control mechanisms to consider both authentication and authorization components to enhance privacy and security. User-Managed Access (UMA) is an access control profile supporting (1) party-to-party sharing that allows the resource owner to authorize the resource to the third-party and (2) customization of access control policy which means resource owner can formulate the policy for accessing the protected resource. However, although the UMA profile defines the authorization process, it does not specify the detail part for authentication. To fill this gap, it is necessary to import digital credential technology to authenticate the third party. Therefore, this paper proposes VC-UMA, an access control mechanism integrating UMA with Verifiable Credentials (VC). VC is an open standard of decentralized credentials which often constructed on the blockchain that allowing user to fully control their credentials. Besides, selective disclosure mechanism is integrated into VC-UMA to address the privacy concerns raised by sharing VCs. To prove the feasibility of the VC-UMA, the proof of concept is conducted. Specifically, a prototype system is implemented and the experiments of the performance is presented.**

***Index Terms*—User-Managed Access, Verifiable Credentials, Access Control, Selective Disclosure, Blockchain**

I. Introduction

Access Control [1] is a mechanism responsible for managing the requests that want to access protected resources. Without proper access control mechanisms, internet services are prone to various privacy and security issues. For instance, invalid access to protected resources, or leakage of privacy data during the access control process.

User-Managed Access (UMA) [2] is a party-to-party right delegation profile extension for access control. In UMA, users can not only formulate the customized access control policy [3] but also realize the *Party-to-Party Sharing* scenario. Nevertheless, according to Sandhu [4], secure access control requires several important components namely, *Authorization*, *Authentication*, *Auditing*, etc. Especially, authorization and authentication play extremely important roles in the process of user access to the protected resources. However, UMA profile only defines the claim gathering concept for the authentication part which means the user needs to provide the claim to get authentication, but the trust model among all the entities is out of scope. For example, as a resource owner, the problem of *how can I trust the third party to access the resources?* cannot be solved only by employing the UMA profile.

To solve the above problems, this paper adopts Self-Sovereign Identity (SSI), a promising concept that is regarded as the next generation of digital identity [5]. This concept allows individuals fully control of their digital identities and credential. Among SSI, Verifiable Credential (VC) [6] is a core technology that is the digital credential framework of following the SSI principle and specifies by the World Wide Web Consortium (W3C). VC utilizes digital signature technology often together with the blockchain which is a distributed ledger platform with decentralized, immutable, and traceable features. With the wide acceptance of VC, privacy has become an important issue. Regarding the privacy issue, W3C recommends developers follow the data minimization principle [7] when designing VC services. This means that when presenting VCs, is it better to minimize the exposed data to prevent oversharing of the privacy credentials.

Based on the above observations, this paper aims to propose a Verifiable Credential-enabled User-Managed Access Mechanism (VC-UMA) that overcomes the lack of trust authentication model defined in the UMA profile by introducing the advantages of the VC framework. Additionally, in order to reduce the risk of private data leakage when sharing VCs, this paper adopts the selective disclosure technology so that VC holder can redact the private part of data in VC to follow the data minimization principle. To summarize, this paper claims the following contribution.

- **An access control mechanism based on decentralized credential scheme is presented.** This paper proposes a new access control mechanism: VC-UMA based on the W3C's VC model [6] and the UMA Profile [2]. Besides, the relationship and the trusted model among all entities in UMA and VC is reconsidered. Moreover, the guidelines for implementing VC-UMA are provided

Identify applicable funding agency here. If none, delete this.

in static and dynamic ways.

- **Selective disclosure method to achieve the data minimization principle for VC sharing is considered.** Considering the part of privacy-preserving VC sharing, we adopt the selective disclosure methods provided by W3C, a selective disclosure authentication flows is proposed.
- **A use case of VC-UMA has been proposed and implemented as the prototype system as proof of concept.** In order to prove the feasibility and usability of the proposed mechanism, the proof of concept research method is conducted. The implemented prototype system is presented. Furthermore, the performance of the system is analyzed.

II. Background and Related Work

A. User-Managed Access

UMA is an access control profile base on OAuth2 [9], proposed by Kantara Initiative and published on Internet Engineering Task Force (IETF) [2]. OAuth2 is a widely used third-party authorization protocol, but this protocol doesn't cover the party-to-party sharing scenarios. For instance, Alice wants to use the service of photo editing software (playing the role of resource access client), so she authorizes her photos (resource) stored in the third-party cloud service to the photo editing software. However, OAuth2 doesn't support Alice to grant Bob to access her photos on the third-party cloud service. UMA fills the gap of OAuth2 that doesn't define the party-to-party authorization scenario. UMA profile is composed of several entities, the definitions of which are shown in table I.

TABLE I
Entities in User-Managed Access

UMA entities	description
Resource Owner (RO)	The owner of the protected resource.
Requesting Party (RqP)	The party who is attempting to access the protected resource.
Client	A third-party application that proxy the RqP to access protected resources.
Resource Server (RS)	The resource server stores protected resources and is capable of handling resource requests from client.
Authorization Server (AS)	The Authorization server is delegated by the RO to protect resources stored in RS and authorizes resource requests issued by RqP.

B. Blockchain and Smart Contract

Blockchain is regarded as the core technology underlying Bitcoin [10]. The underlying technology of blockchain is distributed ledgers, which is regarded as a decentralized platform with immutability, transparency, and traceability. Recently, some blockchain platforms also brought out a new idea called Smart Contract [11], which allows developers to run program logic that can be used to verify and execute transactions. Ethereum [12] is a famous blockchain platform that not only allows user to compile and deploy smart contracts but also change the account status or interact with other accounts. Furthermore, Ethereum provides Ethereum Virtual Machine (EVM) to compile and run Turing-complete smart contract languages (e.g., Solidity, Vyper, etc.) Therefore, developers can run decentralized applications (DApp) in the blockchain in the form of smart contracts. Compared with Bitcoin, Ethereum is more scalable and flexible for the development of software applications on blockchain.

C. Decentralized Identifiers and Verifiable Credentials

Decentralized identifiers (DID) is one of the important technologies of SSI. DID is a digital Identity framework build over decentralize infrastructure such as blockchain. In DID framework, user doesn't need to rely on the centralized identity service to create a personal digital identity. The composition of the DID is usually composed of a series of random numbers, for example, $DID : example : 0x2e12sfhx....$ This shows that DID is a high degree of anonymity. On the other hand, since a user can possess multiple DID at the same time, user activities can be prevented from being correlated which leads to the improvement of privacy. Besides, In the DID scheme, each DID will be connected to a corresponding DID Document, which mainly contains the public verification keys and related verification data. Specifically, since the storage location of the DID and DID Document is usually recorded in Blockchain, the DID owner can easily use the public key in the DID Document to claim ownership.

In many identity verification scenarios, *Verifiable Properties* are required. For instance, Alice needs to prove that she is graduated from university. To fill this gap, SSI also provides an important technology called *Verifiable Credentials* (VC). VC includes three important roles shown in fig. 1. *Holder* is the entity that owns and controls VC; *Issuer* is a trusted third party responsible for VC issuance; *Verifier* is the entity that wants to verify the VC. VC Framework is built on top of the Verifiable Data Registry (VDR) and DID technology. VDR is generally implemented by blockchain in order to take benefit of blockchain decentralized features. Take the school issuing graduation certificate as an example. First, both the student (Holder) and the school (Issuer) need to register their unique DIDs to VDR in advance. After that, Issuer packs and signs the student's DID, claims, and related certificate information by DID to generate VC. After obtaining VC, Verifier can find the DID Document through the issuer's DID in the VDR to obtain the public key and then verify the signature in the VC.

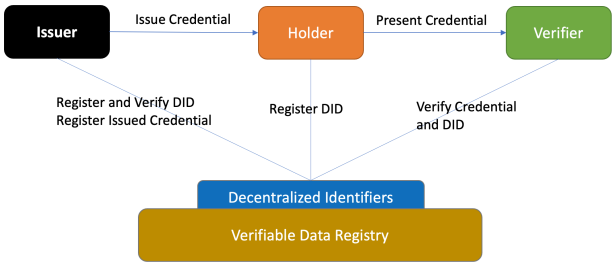

Fig. 1. The roles and information flows for Verifiable Credential

As the VC technology is still in the development stage, there are still many problems to be solved. One of the potential problems is privacy violations. For preventing privacy violations, the generally accepted best practice is data minimization. In order to achieve the data minimization principle, Selective disclosure techniques is recommended to be used in the issue or present stage of VC [6]. Selective disclosure signatures are commonly realized by the zero-knowledge proof cryptography method in order to allow the VC holder to hide the private attributes in VC. There are some developing signature schemes that can be natively applied to selective disclosure signatures systems, such as Camenisch-Lysyanskaya (CL) signature [13] and BBS+ signature [14]. Compared with CL Signature, BBS+ signature can achieve considerable security through much shorter keys and signatures [15]. Therefore, this paper not only adopts BBS+ signature technology to support the signature scheme for VC but also apply to the zero-knowledge proof for selective disclosure signatures method.

D. Related Work

In recent years, there are many studies dedicated to utilizing SSI technology to improve privacy and security for access control mechanism. CredChain [16] provides an SSI platform that enables issuer and VC holder to create, share and verify credentials in a safe environment. In consideration of personal privacy issues, this paper also leverages redactable signature [18] method allowing users to hide the secret attribute in credential; CanDID [19] is a user-friendly decentralized identity platform that is committed to achieving Sybil-resistance, accountability, key recovery, and other goals in a privacy-preserving way; Lagutin et al. [20] proposed a DID and VC enabled OAuth-based delegation network. They attempt to establish an OAuth-based access control framework through the introduction of DID and VC technology to solve the problem of OAuth2 which does not cover the authentication profile. However, this study only made a preliminary process design for the framework but didn't consider the privacy issue of VC as well as the party-to-party sharing issue often encountered in IoT scenarios.

To overcome the issues mention in advance, this paper is committed to designing a *privacy-preserving access control mechanism* that focuses on both *Authorization* and *Authentication* domain in the access control mechanism. Besides, the proposed mechanism is based on the UMA profile to support the party-to-party sharing scenario. Moreover, by introducing promising DID and VC technologies, the proposed mechanism allows users to participate in the access control process while also protecting the privacy of the personal credentials presented by themselves.

III. System Design

In order to overcome the aforementioned issues, this paper is dedicated to designing a *Privacy-Preserving Verifiable Credential-enabled User-Managed Access* Mechanism, and named it VC-UMA. VC-UMA is a user-centralize access control mechanism, which contains two subdomains, namely VC domain and UMA domain. UMA domain refers to UMA Profile [2]; The VC domain refers to the decentralized identity technology in SSI, such as DID and VC. Besides, the notation used in this paper are listed in Table II. While designing the overall system, we also commit to achieving the following goals.

TABLE II
Notation

Notation	Description
doc	DID Document
pk	Public key
sk	Secret key
σ	Signature by sk over id and subject
π	Proof
CS	The set of claims
ctx	Context

1) **Decentralization of credential issuance system: trusted and high availability.** The centralized credential issuance system has disadvantages such as the risk of signal point of failure. Therefore, importing a decentralized credential issuance scheme is the primary goal of this paper.
2) **Privacy of Credential.** When utilizing VC technology, privacy data leakage of the credential is also an important issue. Hence, the introduction of the related technology of privacy-preserving VC is also one of the goals of this paper.
3) **Compact Size of Privacy-Preserving Cryptographic.** Privacy-preserving VC technology often requires the usage of encryption methods, such as zero-knowledge Proof. However, related technologies often require considerable computer resource(e.g., zk-SNARKs). Therefore, this paper is committed to employing the privacy-preserving cryptographic that can achieve the high-level security which only requires compact size of computing and storage resources.

A. System Architecture

As shown in fig. 2, the access control mechanism is divided into the UMA domain and VC domain. Among them, the UMA domain is responsible for handling the authorization management of protected resources and the verification of the requesting party (RqP); The VC domain defines the process for RqP to obtain VC. Besides, we assume that RO, RS, and RqP already own DID which corresponds to the private key, public key, and DID Document registered to the Verifiable Data Registry (VDR). In addition, anyone can directly obtain the DID Document corresponding to the DID through VDR.

The UMA Domain mainly refers to the entities defined in UMA Profile [2]. RO is the provider of protected resources, responsible for the management of RS, and defines the access control policy of RS through AS; RqP is a third party that wants to access protected resources;

In VC Domain, there are four roles (1)**VC issuer** is responsible for issuing and releasing VC to RqP; (2)**VC wallet**

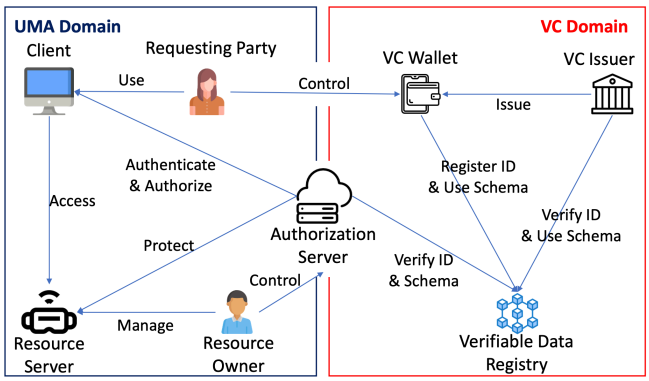

Fig. 2. System Architecture of VC-UMA

owned by RqP which supports storing and managing VC as well as DID; (3)**Verifiable Data Registry (VDR)** is a decentralized infrastructure such as blockchain that supports the registration and resolution for DID, DID Document, and VC status; (4)**Authorization Server (AS)** represents the role of VC verifier in the VC model. Besides, AS also response for verifying whether the Verifiable Presentation (VP) provided by the VC wallet complies with the access control policy or not.

B. Smart Contracts for VC domain

The VDR of VC Domain needs to be maintained by SSI service provider (a trusted party that provides SSI implementation standards. It can be a third party or VC issuer) To set up the VC Domain process, SSI service provider needs to deploy *VC Registry Contract* to VDR.

Considering the difference between *Selective Disclosure Signatures* flow and *Atomize Information* flow, this paper provides two kinds of *VC Registry Contract* support these two flows as follows.

- **Selective Disclosure Signatures Flow**
  Specifically, because the *Selective Disclosure Signatures* flow hides the claims in the VC through encryption technology to ensure that sensitive information is not disclosed, the VC issuer does not need to modify the VC generation method provided by the W3C VC document.
- **Atomize Information Flow**
  In order to protect the privacy of claims in VC, VC issuer separates n claims and then issue n VCs, denoted as $\{VC_n\}$, which one VC only include one claims. In this research, the set of $\{VC_n\}$ is defined as VC_Set.

The design of VC Registry Contract for Selective Disclosure Signatures Flow refer to fig. 3. It mainly includes *register* and *revoke* API for VC issuer to register and revoke VC, while *isRevoked* API provides verifier to query the status whether a VC has been revoked or not; The design of VC Registry Contract API for Atomize Information flow shown in fig. 4. After generating VC_Set, VC issuer assigns a VC_Set_id for VC_Set, and then register VC_Set and $\{VC_n\}$ to the smart contract by invoking $register\_set$ and $register\_VC$ API respectively. Issuer can also revoke the VC_Set through the revoke API, and verifier can check the revocation status of the VC_Set by querying *isRevoked* API.

VC Registry Contract API (Selective Disclosure Signatures Flow)

- $register(vc\_id)$: The VC issuer registers the VC.
- $revoke(vc\_id) \rightarrow true \vee false$: The VC issuer revokes the exist VC.
- $isRevoked(vc\_id, issuer) \rightarrow true \vee false$: The verifier can verify whether the VC have been revoked or not.

Fig. 3. VC Registry Smart Contract API

VC Registry Contract API (Atomize Information flow)

- $register\_set(vc\_set\_id, include\_type)$: The VC issuer registers the vc_set_id and provides the $include\_type$ which represent the type of VCs included in VC set.
- $register\_VC(vc\_id, vc\_type, vc\_set\_id)$: The VC issuer registers the VC which include in the registered VC Set to the VC registry by issuer ' s account.
- $revoke(vc\_set\_id) \rightarrow true \vee false$: The VC issuer revokes the existing VC set in the VC registry by issuer ' s account.
- $isRevoked(vc\_set\_id) \rightarrow true \vee false$: The Verifier can verify whether the VC set has already been revoked or not.

Fig. 4. VC Registry Smart Contract API (Atomize Information flow)

C. VC Domain Execution Flow

In terms of the execution flow of VC-UMA. First, in the VC domain, VC issuer should issue VC following the Selective Disclosure Signatures flow or Atomize Information flow as follows.

- **Selective Disclosure Signatures flow**

  1) The VC issuer obtains the RqP's DID_{RqP}, and then verifies the DID_{RqP} through VDR.
  2) The VC issuer prepares i claims to be issue, CS=$\{claim_i\}$, then signs the claims by DID_{issuer} corresponding public and private key. Denoted as $\sigma \leftarrow BBS\_PLUS(DID_{RqP}, DID_{issuer}, (sk, pk)_{issuer}, CS)$
  3) Generate VC, denoted as $VC = \{metadata, CS, \sigma\}$
  4) VC issuer invokes the *register* API of VC Registry to register the VC identifier.
  5) VC issuer send the VC to the VC wallet controlled by RqP.

- **Atomize Information Flow**

1) The VC issuer obtains the VC subject's (RqP) DID_{holder}, and then verifies the DID_{holder} through VDR.
2) The VC issuer prepares i claims to be issue, CS=$\{claim_i\}$, then signs the claims by DID_{issuer} corresponding public and private key. Denoted as $\sigma_i \leftarrow BBS\_PLUS(DID_{holder}, DID_{issuer}, (sk, pk)_{issuer}, claim_i)$
3) Generate VC. $VC_i = \{metadata, claim_i, \sigma_i\}$
4) VC issuer repeats the step 2 and 3 i times, then generates VC set, defined as $VC\_SET = \{VC_i\}$. After that issuer assigns the VC_Set_id as the identifier of VC_SET
5) VC issuer invokes the *register* API of VC Registry to register the VC_Set_id.
6) VC issuer send the VC_SET and VC_Set_id to the VC wallet controlled by RqP.

D. UMA Domain Execution Flow

According to the UMA Profile [2], the VC-UMA mechanism process can be divided into three phases, namely *Protecting a Resource*, *Getting Authorization*, and *Accessing a Resource*.

1) Protecting a Resource Phase: First, in the Protecting a Resources phase (fig. 5), RO needs to apply for registration for RS from AS. At the same time, RO needs to provide resource DID, resource name, and PAT which prove that the registration is issued by the RO. After receiving the registration request sent by RS, AS verifies the existence of the resource DID and verifies the integrity of the DID Document through DID Registry. If the resource DID is verified, the AS will return a unique resource ID to the RS.

The second step is *setPolicy*. At this step, RS must provide policySchema (JSON format), PAT (for authenticating RO), and resource ID (for identifying RS). Among them, *policySchema* provides the format for RqP to generate the Verifiable Presentation (VP) used in the Getting Authorization Phase; *policySchema* represents the standard for verifying VP (e.g., age over 18).

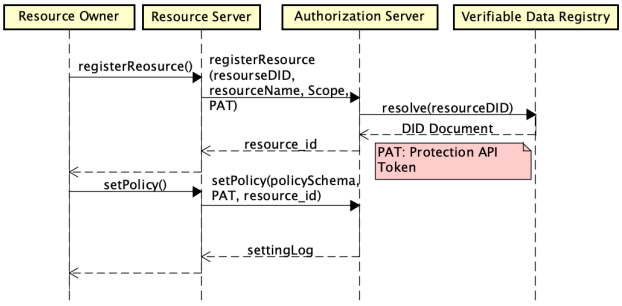

Fig. 5. Protecting a Resource Phase

2) Getting Authorization Phase: According to the UMA Profile, the Getting Authorization phase is divided into two steps. The first step is that RqP requests resource from RS through Client (fig. 6). Since Client's request has not been authorized yet, so RS will directly apply for request permission from the Permission Endpoint of AS. After that, AS will return $permissionTicket$, $policySchema$, as_uri to Client.

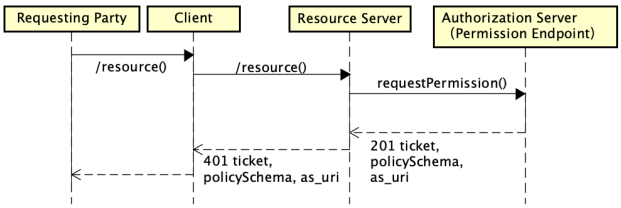

Fig. 6. Getting Authorization Phase (Permission Endpoint)

The second step of the *Getting Authorization* phase is that the Client requests an RPT from the Token Endpoint in AS by providing a VP (fig. 7). This step is divided into two stages: *Deriving Verifiable Presentation* and *Releasing RPT*.

In the *Deriving Verifiable Presentation* Stage, the Client alerts RqP that it requires using VC wallet for identity verification. Specifically, RqP can use VC wallet to obtain credential requests (in practice, it is common to use QR code). In addition, according to Selective Disclosure Flow and Atomize Information Flow, there are two separated ways of VP derivation as follows.

- **Selective Disclosure Signatures Flow**
  After accepting the credential request, VC wallet can alert RqP of the claims that need to be presented according to *policySchema* accompanied by the request. Then, RqP can choose k claims that it wants to share to the AS among the VCs with i claims, which can be expressed as $CS_k = \{claim_k\}$. After that, VC wallet derives proof from VC by *BBS+ Signature Proof Generation* method. Derived proof can be expressed as $\pi = BBS + ProofGen(pk_{issuer}, CS_{i-k}, CS_k, \sigma)$. Then VC wallet can use the derived proof to generate derived VC, expressed as $VC_{derived} = \{ctx, CS_k, \pi\}$. Finally, VC wallet package the derived VC into VP and send back to AS. VP can be expressed as $VP = \{ctx, VC_{derived}, \pi\}$.
- **Atomize Information Flow**
  In the Deriving Verifiable Presentation Stage of the Atomize Information Flow, since RqP receives VC_SET from the VC issuer, which each VC has only included one claim, RqP can select the required k VCs in the VC_SET, expressed as $VC\_SET_k = \{VC_k\}$.

At last, in the *Releasing RPT* Stage, the AS first obtains the resource request information according to the *ticket*. Then AS verifies the VP provided by VC wallet and confirms whether it meets the requirements of the policy or not. Furthermore, AS also needs to verify whether the DID of the VC issuer is legal through the VDR. When all verifications are passed, AS returns an RPT to the Client.

3) Accessing a Resource Phase: After successfully obtaining the RPT in the previous phase, the client first sends an RPT-accompanied request to the RS again. After that, the RS

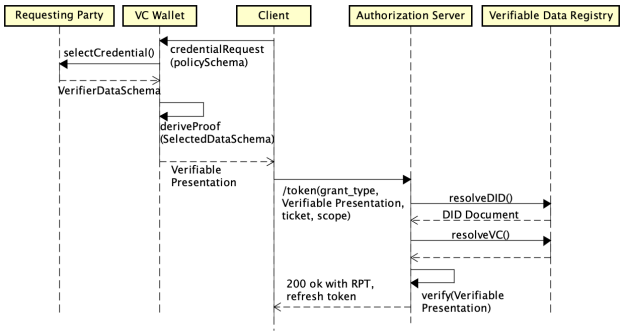

Fig. 7. Getting Authorization Phase (Token Endpoint)

redirects the RPT to the AS's token introspect endpoint for verification. Then, AS returns the verification result (token introspection object) back to the RS. Finally, if the result is successful, RS can start to allow the client to access the protected resource.

IV. Evaluation

This section illustrates the feasibility of the VC-UMA by proof of concept. The prototype system is introduced. In addition, the experiments and the analyzation of prototype system is reported.

A. Feasibility

The prototype system is constructed based on the selective disclosure signatures flow of *VC-UMA*. It is divided into *VC Setup Side* and *Access Control Side*. *VC Setup Side* includes VC issuer, which mainly implements VC domain execution flow; *Access Control Side* includes Alice's (requesting party) device, resources, and authorization server owned by Bob (Resource Owner), mainly implement the UMA domain execution flow. In order to communicate with smart contracts, VC issuer, authorization server, and Alice's device are required to deploy blockchain node.

The backend of the prototype system is developed in TypeScript language. Furthermore, Node.js is used as the execution environment and Docker's MySQL image is deployed as the database. The Koa2 is adopted as a backend framework to encapsulate the system service modules through the RESTful API. Frontend or other applications can invoke related services through the RESTful API provided by Koa2. Besides, Ethereum blockchain is adopted to implement as VDR infrastructure. The Ethereum client, Go-Ethereum [21], is deployed to play the role of bridging the backend system and blockchain. In addition, Solidity is the smart contract language chose to implement the VC Registry. On the other hand, Alice's device, authorization server, resources, and VC issuer services in the prototype system is divided into four processes implemented in the device with *macOS Monterey 12.0.1* OS, *Apple M1* CPU and 16 GB RAM.

To meet the needs of signature and derive proof for VC, the BBS+ Signature Module [22] provided by MATTR is used, which is designed based on the pairing-friendly curve, *BLS12-381*. BBS+ Signature module not only provides VC issuer to implement multi-message VC signature but also provides VC wallet to realize the selective disclosure for deriving VC proofs.

According to the prototype, the operation flow for each entity is listed as follows. First, Alice can provide claim data to the VC issuer to apply for the VC(fig. 8a). After that, the VC issuer can generate VC by signing the received data and then return it to Alice; As the resource owner, Bob can register resources to be rented into the backend database of AS. Then, Bob can customize the access policy for resources by generating the *policySchema* so that AS can conduct the access control process based on the defined policy; After finishing the above setup process, Alice can start the flow to request the protected resource.

First, Alice can access the protected resource web page9 and push the button of the resource she wants to access. Then the page will output the file of *policySchema*. After that, Alice can use her VC wallet to receive the *policySchema*. After receiving the *policySchema*, Alice's VC wallet will redirect Alice to the page to select the reveal credential attributes. Here, Alice can select the VCs or claims she wants to present (fig. 8b). After Alice confirms, VC wallet will derive *Derived Proof* and package it into VP. Finally, the permission ticket and the VP are sent to the token endpoint of AS. When the AS verified the metadata sent by VC wallet, Alice can be authorized to access the resource.

Fig. 8. The prototype system. (a) requesting party apply for VC; (b)requesting party select reveal claims in VC

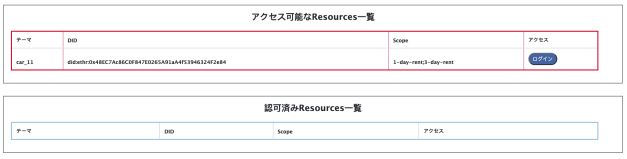

Fig. 9. Protected resource web page for the prototype system

B. Experiments

In order to help developers evaluate the appropriate authentication flow and the performance of using blockchain when applying the *VC-UMA*, this paper presents three different experiments listed in Table III.

TABLE III
EXPERIMENTS FOR VC-UMA

Experiment name	Description
Verification Time for Verifiable Presentation	The response time for verifying the verifiable presentation which reveal n claims (n = 1, 10, 20, 30, 40, 50)
Response Time of Invoking VC Registry API	The response time of invoking the API of VC Registry Contract
Gas Cost of Invoking VC Registry API	The gas cost of invoking the API of VC Registry Contract

1) Verification Time for Verifiable Presentation: According to the results (fig. 10), it can be found that although *Selective Disclosure Signatures* flow takes more time to verify VP_{SD} with $n = 1$, the overall response time does not significantly increase as the increase of n. Besides, the verification time of *Atomize Information* flow will increase significantly with the increase of the number of VP_{AI}. Therefore, it can be inferred that *Atomize Information* flow can be used when only a few claims need to be verified, and *Selective Disclosure Signatures* flow is more appropriate in cases where VP included many numbers of claims.

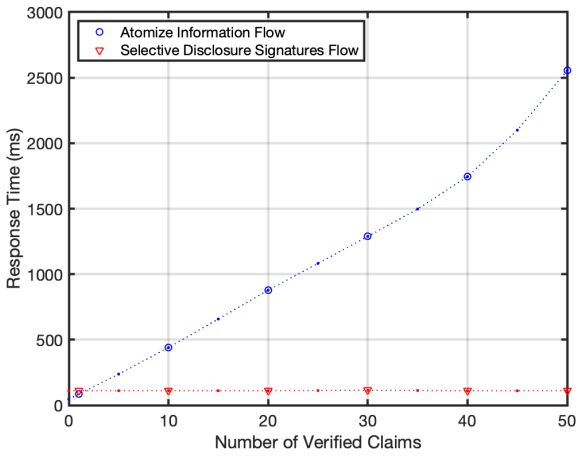

Fig. 10. Verifiable Presentation Response Time for Each Authentication flow

2) Response Time of Invoking VC Registry API: As the experimental results, it can be observed that since the *deploy contract*, *register*, and *revoke* API need to wait for the new block to be mined, a considerable time is taken. On the contrary, since *isRevoked* API, does not need to change the block state, so the response time is relatively fast.

3) Gas Cost of Invoking API in VC Registry: The result is as shown in fig. 11. It can be found that most of the cost are consumed when deploying of the VC Registry Contract, while

TABLE IV
THE RESPONSE TIME OF INVOKING VC REGISTRY API

API Name	deploy contract	register	revoke	isRevoked
Response Time(ms)	4580.2	6461.6	5172.2	1.8
Standard Deviation(ms)	2732.86	1729.52	1809.99	0.84

Register VC and *Revoke VC* API are the next. The *isRevoked* API does not cost the gas since it does not need to modify the block status. Developers can focus on reducing the amount of data size that needs to be stored on the blockchain when deploying and invoking the VC Registry contract to reduce the cost of gas.

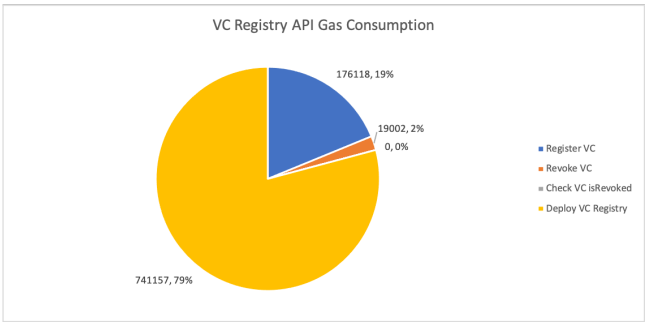

Fig. 11. The Gas Consumption of Each VC Registry Contract's API

C. Analysis

1) Security: When deploying VC-UMA, some attacks may cause the system to face security risks. The following will analyze how VC-UMA can resist against related attacks.

- **Spoofing Attack Resistance**
  The adversary in spoofing attack can access the protected resources by disguising the identity of a legitimate user. According to the method proposed in this research, users need to hold a legal VC to pass the authentication of authorization sever. The trusted model of VC relies on the trusted VC issuer and underlying blockchain technology. As long as the VC issuer can be fully trusted, is it impossible for the adversary to pass the authentication by modifying the blockchain status or generating fake VCs.
- **Man-in-the-middle Attack（MITM）Resistance**
  In the process of authenticating the requesting party in VC-UMA, the adversary may intercept the data among transmission and insert malicious content to disrupt the authentication process of VC-UMA. To solve this problem, first, the *HTTPs* protocol with *resist MITM attack* feature should be leveraged to prevent the adversary from participating in the access control process; Second, before presenting the VC, the requesting party must sign the VC in the VC Wallet. In this way, the authorization server can

verify VC is signed by the DID owned by RqP; at last, in UMA Profile, Authorization Access Token (AAT) is provided to bind the client and RqP. By providing AAT, malicious clients from impersonating the requesting party can be avoided.

2) Comparison: Table V lists the comprehensive comparison between the VC-UMA and other access control mechanisms. *Party-to-Party Authorization* means if the access control mechanism allows resource owner to define party-to-party sharing scenario; *Decentralization* illustrates if the common implementation of access control system relies on centralized services for authentication component; *Privacy-preserving Authentication Flow* indicates whether the user can fully control their credential in the access control process or not; In a nutshell, VC-UMA not only provides an access control mechanism enhancing security and privacy but also provides a more decentralized and more privacy-preserving solution for authentication component in access control compared with other traditional mechanisms.

TABLE V
COMPARISON OF ACCESS CONTROL MECHANISMS IN EACH ASPECT

Mechanism Name/ Feature	UMA	CanDID [19]	Control-Chain [17]	Cred-Chain [16]	Lagutin et al. [20]	VC-UMA
Authorization Component	+	−	+	−	+	+
Authentication Component	+	+	−	+	+	+
SSI-supported	−	+	−	+	+	+
Party-to-Party Resource Sharing	+	−	−	−	−	+
Anonymous Credentials	−	+	−	+	−	+

+: solved
−: unsolved

V. CONCLUSION

This paper proposes a new access control mechanism, VC-UMA, by combining VC technologies with UMA to fill the gap with not defining authentication component in the UMA profile. We demonstrated the VC-UMA mechanism through dynamic and static structures. By virtue of the decentralized nature of VC, the risks of single node failure and privacy leakage are reduced compared to the common practice which relies on centralized services when implementing UMA. Besides, since the import of VC needs to be built on a decentralized blockchain infrastructure, this paper is also committed to designing a smart contract API suitable for VC-UMA in order to let authorization server communicate with the blockchain. On the other hand, considering the privacy issue in VC, this paper also introduces a selective disclosure method so that the requesting party can hide private information when presenting the VC. Furthermore, to reduce storage consumption and verification time, we choose the BBS+ signature method that has compact features while remaining the high-security level. Finally, to prove the feasibility of the proposed VC-UMA, the prototype system and the result of the experiments are demonstrated. As the future work, the complexity of blockchain technology and selective disclosure technologies for VC are expected to be improved to reduce the cost for developers implementing VC-UMA mechanism.

REFERENCES

[1] P. Samarati and S. C. de Vimercati, "Access control: Policies, models, and mechanisms," *in International School on Foundations of Security Analysis and Design*, pp. 137–196, 2000.

[2] E. Maler, M. Machulak, J. Richer, and T. Hardjono, "User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization," *Internet Engineering Task Force*, https://datatracker.ietf.org/doc/draft-maler-oauth-umagrant (accessed Oct. 06, 2021).

[3] M. P. Machulak, E. L. Maler, D. Catalano, and A. Van Moorsel, "User-managed access to web resources," *in Proceedings of the 6th ACM workshop on Digital identity management*, pp. 35–44, 2010.

[4] R. S. Sandhu and P. Samarati, "Access control: principle and practice," *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 40–48, 1994.

[5] C. Allen, "The Path to Self-Sovereign Identity," http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html (accessed Nov. 17, 2021).

[6] World Wide Web Consortium, "Verifiable Credentials Data Model v1.1," https://www.w3.org/TR/vc-data-model/ (accessed Nov. 22, 2021).

[7] D. Chadwick, D. Longley, M. Sporny, O. Terbu, and D. Zagidulin, "Verifiable Credentials Implementation Guidelines 1.0," W3C Work. Group Note Sep, 2019.

[8] A. Preukschat and D. Reed, "Self-sovereign identity," *Manning Publications*, 2021.

[9] D. Hardt et al., "The OAuth 2.0 authorization framework," RFC 6749, October, 2012.

[10] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *IEEE Decentralized Bus. Rev.*, p. 21260, 2008.

[11] X. Xu et al., "A taxonomy of blockchain-based systems for architecture design," *in 2017 IEEE international conference on software architecture (ICSA)*, pp. 243–252, 2017.

[12] V. Buterin, "A next-generation smart contract and decentralized application platform," *in White Pap.*, vol. 3, no. 37, 2014.

[13] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," *in Annual international cryptology conference*, pp. 56–72, 2004.

[14] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," *in Annual international cryptology conference*, pp. 41–55, 2004.

[15] W.Shih, "Comparing CL Signatures with BBS+ Signatures," https://gist.github.com/wayne-shih/46c3d57608d9dcf8e6722d86084e710c (accessed Nov. 18, 2021).

[16] R. Mukta et al., "Blockchain-based Verifiable Credential Sharing with Selective Disclosure," *in 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 959–966, 2020.

[17] O. J. A. Pinno, A. R. A. Gregio, and L. C. De Bona, "Controlchain: Blockchain as a central enabler for access control authorizations in the iot", in GLOBECOM 2017-2017 IEEE Global Communications Conference, pp. 1–6, 2017.

[18] R. Johnson et al., "Homomorphic signature schemes," *in Cryptographers' track at the RSA conference*, pp. 244–262, 2002.

[19] D. Maram et al., "CanDID: Can-do decentralized identity with legacy compatibility, Sybil-resistance, and accountability," *in 2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1348–1366, 2021.

[20] D. Lagutin et al., "Enabling decentralised identifiers and verifiable credentials for constrained IoT devices using OAuth-based delegation," *in Proceedings of the Workshop on Decentralized IoT Systems and Security (DISS 2019), in Conjunction with the NDSS Symposium, San Diego, CA, USA*, vol. 24, 2019.

[21] Ethereum Foundation, "Go Ethereum," https://geth.ethereum.org/ (accessed Nov. 08, 2021).

[22] MATTR, "A solution for privacy-preserving verifiable credentials," https://github.com/mattrglobal/bbs-signatures (accessed Nov. 16, 2021).

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.