Journal of Computer Science

Research Article

Exploring Self-Sovereign Identity Through a Digital Transcript Proof-of-Concept

Naphat Khajohn-Udomrith and Ittipon Rassameeroj

Faculty of Information and Communication Technology, Mahidol University, Nakhon Pathom, Thailand

Article history
Received: 06-11-2025
Revised: 14-04-2026
Accepted: 16-04-2026

Corresponding Authors:
Ittipon Rassameeroj
Faculty of Information and Communication Technology, Mahidol University, Nakhon Pathom, Thailand
Email: ittipon.ras@mahidol.ac.th

Abstract: Credential forgery presents a significant threat to the integrity of academic and employment documents. While Self-Sovereign Identity (SSI) and Verifiable Credentials (VC) offer a decentralized alternative, existing solutions often struggle to balance institutional trust with holder privacy. This paper proposes a novel, dual-layered architectural framework for digital transcripts. We implement a strategic bimodal DID approach, utilizing did:web for the issuing institution to leverage established web reputation and did:key for the student holder to ensure lightweight, stateless mobility. Furthermore, we address privacy vulnerabilities in credential status checks by developing a k-anonymous Revocation List. This mechanism protects students by grouping 40,000 individual records into a single compressed bitstring (k = 40,000), ensuring that a specific student's status cannot be uniquely identified or singled out during a check. Benchmarking results confirm the system's efficiency, with core local cryptographic verifications executing in under 1 ms. We identify that system latency is primarily driven by network dependent interactions, specifically the time required for API-based credential retrieval and sending presentations to verifiers. Comprehensive functionality and security tests validate that this integrated approach successfully enhances credential integrity while maintaining superior user privacy compared to standard decentralized models.

Keywords: Self-Sovereign Identity, Verifiable Credentials, Digital Identity, JSON Web Signature, JSON Link Data (JSON-LD), Digital Transcripts

Introduction

In the century of digital transformation, the management of personal and professional documents faces serious problems caused by inefficiencies and vulnerabilities to fraud. Traditional centralized systems compromise privacy and speed in data transactions, necessitating a move to more secure and autonomous solutions. Self-Sovereign Identity (SSI), which eliminates third-party intermediaries, and Verifiable Credentials (VCs) are transformative technologies.

The former is a technology that puts the individual in control of their identity, while the latter is a standardized, interoperable framework for digital attestations that are secure, private, and verifiable.

Digital transcripts will be powerful when used in educational contexts since they will make the process easier, improve the integrity of the data, and decrease false claims. Using SSI and VCs, institutions can provide a trustworthy and more effective way for their graduates to manage and share academic records. This paper presents the development of a proof-of-concept system for SSI for VC-based digital transcript systems and the feasibility evidence of potential benefits in revolutionizing the domain of educational records.

The embedding of SSI and VC within digital transcript workflows marks a significant paradigm shift from conventional approaches to the administration and management of academic records. Centralized legacy systems, though very popular, also present serious challenges of security threats, privacy issues, inefficiency, and lack of interoperability. In contrast, SSI-based decentralized systems provide security that is superior to that of non-SSI systems, granting students with increased control over their information, and enabling students with instant and secure verification through cryptographically signed credentials. Additionally, these systems enable global interoperability, eliminate administrative costs, and most importantly, greatly reduce fraud. This decentralized model not only ensures the confidentiality of academic

transcripts but also improves the student experience by directly, and securely providing students with their credentials which, in turn, paves the way for smooth academic and career paths. The main objective of this work is to develop a proof-of-concept system for SSI and VC-based digital transcript systems for the issuance and verification of digital transcripts within educational institutions. This project attempts to solve these downsides through a decentralized, secure, and tamper-evident solution for managing digital transcripts using SSI and VC technology. The specific objectives of the paper are as follows:

- Developing the infrastructure that would support the implementation of identity management within the SSI framework and the issuance of Verifiable Credentials in systems at the academic institution. This includes setting up DIDs, decentralized key management, and integration with relevant distributed ledger technologies
- Design a Verifiable Credential schema for digital transcripts that is interoperable and compatible with existing SSI frameworks, such as W3C Verifiable Credentials, Decentralized Identifiers, and OpenID4VC
- Design a feature to facilitate the issue of digital transcripts as Verifiable Credentials on request. This follows the process of digitally signing transcripts that contain relevant academic information, such as course grades, course credits, and program completion status
- Design mechanisms to ensure digital transcripts are securely verified integrity using SSI and VC technology. The following tools or services will be given to recipients: employers or academic institutions to verify the credentials presented by students

This work marks a significant paradigm shift in the administration of academic records by moving away from centralized legacy systems. The main contributions of this paper are as follows:

- A Dual-Layered Architectural Framework: We propose a strategic bimodal DID approach that utilizes did:web for issuing institutions to leverage established web reputation and did:key for student holders to ensure lightweight, stateless mobility
- Privacy-Enhanced Revocation Mechanism: We develop a novel status-check mechanism that applies k-anonymity principles (k = 40,000) to compressed bitstring lists, ensuring individual students remain indistinguishable and cannot be uniquely identified during revocation checks
- Validated Proof-of-Concept and Benchmarking: We implement a functional system integrated with standard protocols (OpenID4VC and OpenID4VP) and provide comprehensive performance data showing core cryptographic verification executes in under 1 ms

These contributions provide a robust, decentralized, and cryptographically secure framework that addresses the critical issues of fraud and privacy found in traditional systems. However, Self-Sovereign Identity (SSI) and Verifiable Credentials (VC) are relatively new technologies in the field of digital identity management, still undergoing active development projects. There is currently limited research on these topics. As a result, their implementation and adoption face challenges in scalability, interoperability, and practical deployment across diverse systems.

Background

Self-Sovereign Identity (SSI)

Self-Sovereign Identity (SSI) refers to the critical phase in developing digital management of identity into the hands of individuals regarding their personal information against the backdrop of increased interconnection in the contemporary world. It is based on the basic principle that every individual should be in complete, sovereign control of his or her identity, independent of a central authority. This could happen by using a key feature that we called as Decentralized identifiers or DIDs to communicate in each component. Based on this very fundamental notion, SSI will be able to bring users much more privacy, security, and convenience while managing their own identity across several digital interactions. In SSI, there are four main components: Issuers, Holders, Verifiers, and Verifiable Data Registry.

- Issuer: An authoritative entity, such as a university, that creates and signs digital credentials using a predefined schema. The issuer registers its identifier in a registry and confirms the holder's identity before issuance
- Holder: The individual (e.g., student or alumni) who owns and controls their digital credentials. Credentials are kept securely in an Academic Wallet, allowing the holder to generate Decentralized Identifiers (DIDs) and present claims to verifiers without constant issuer intervention
- Verifier: An entity, such as an employer or government agency, that requires proof of specific claims. The verifier validates the authenticity and integrity of presented credentials by checking digital signatures against public keys retrieved from the registry
- Verifiable Data Registry A trustless repository, typically implemented on a blockchain or distributed ledger, used for managing DIDs and their

corresponding DID documents. It ensures that data remains immutable, tamper-evident, and accessible for public verification

Decentralized Identifiers (DIDs)

DIDs are indispensable to understanding the inner details of how SSI works. DIDs are a new identifier that empowers both verifiable and self-sovereign identities in a digital format. At the core of SSI, they are the base element that allows any entity complete control over its digital identity without a centralized registry.

DIDs are URI-compliant and designed to be globally unique, resolvable, highly available, and cryptographically verifiable. They are usually associated with cryptographic material, such as public keys and service endpoints, enabling secure peer-to-peer communication. The owner of a DID uses the associated private key to prove control over the DID; hence, their identity is authenticated.

The main advantages of DIDs are that they can achieve this decentralized operation independently of any central registry or authority and that they are typically recorded on a blockchain or any other form of distributed ledger. This helps ensure that they can be tamper-resistant and resolvable everywhere in the world. In addition, this contributes to a robust and resilient identity system without a single point of failure:

did:example:123456abcdefg

From the example above, we can separate it into three main parts. First, ”did” indicates that this URI is part of the DID specification and thus globally recognized for decentralized digital identities. The ”example” here refers to the DID method, which defines what kind of mechanism or platform underneath keeps this particular DID and how it would resolve to find a respective DID document. The ”method name” here might be a placeholder for which blockchain, distributed ledger, or other decentralized system might be used. The final segment, ”123456abcdefg” would be the actual identifier created within the namespace of the ”example” method.

At the core is the DID subject, split into two entities: The University (Issuer) and an Alumni (Holder). The University, as the Issuer, is responsible for issuing digital credentials associated with the Alumni, who is the Holder of these credentials. The DID, “did:example:abcdefg”, is a unique reference that points to a DID document. This document, which is critical in the SSI infrastructure, contains the necessary information to authenticate the DID subject’s identity and enable secure digital interactions.

The DID is recorded on a Verifiable Data Registry, which can be a distributed ledger or part of a blockchain. It guarantees that the DID and the DID document associate are tamper-evident and publicly verifiable. The registry should enable the integrity and availability of the DID documents. When the DID is resolved or queried, the process involves retrieving the DID document from the Verifiable Data Registry. This document contains cryptographic material like a public key, service endpoints for communication, and other data needed to establish trust and confirm the identity of the subject of a DID.

Decentralized Identifier Resolution (DID Resolution)

DID Resolution translates a DID into a DID document, enabling verification and interaction with the identified entity, whether a person, organization, or object.

The DID Resolution process involves taking a DID and using the specific DID method it references to look up the corresponding DID document. This document is a JSON-formatted file, like an example in Figure 1, that provides the details necessary to authenticate the identity associated with the DID and to facilitate secure communication. The DID document typically includes public keys, authentication protocols, service endpoints, and other control mechanisms.

The resolution happens via a DID resolver, software enabling interfacing, depending on the DID method, with several DID methods and underlying systems like blockchain systems or other distributed ledger technologies. Since all these systems are decentralized by nature, one resolves a DID not by looking up this identifier in some central database or authority but rather by a resolver walking through a distributed network to resolve the DID document.

DID resolution functionality is standardized to permit coexistence and interoperability across various DID methods. Interoperability of the DID ecosystem allows entities to use DIDs issued by different methods and even across different networks. The resolution itself is designed to be secure and privacy-preserving.

Fig. 1: Example of DID Document, adapted from (W3C, 2022 Verifiable Credentials Working Group)

Verifiable Credentials (VCs)

VCs are the building blocks of decentralized identity systems, which allow secure and trustworthy interactions in the digital sphere. These are digitally signed statements that represent the identifying attributes or competencies of an individual or organization in a tamper-resistant form and secured through cryptographic means.

A VC has three different parts. The first is Metadata, which embodies the context needed during presentations of the credential issuing authority, date of issuance, type of credential, and possibly its expiration date, among other description details. This Metadata is essential for the verifier so he understands the source and type of credential. The second is Claim, which forms the actual content of VC, making confident assertions about the subject; that is, the credential holder. These can be anything from identification details to qualifications or even rights. The last and most crucial element is the Proof. Usually, the issuer's digital signature guarantees that this VC is authentic and not tampered with. Any verifier can easily verify that a VC is indeed issued from an issuing entity and is not tampered with, hence establishing trustworthiness. These elements combine to deliver a secure, verifiable digital credential across different platforms and services in an SSI ecosystem.

Verifiable Presentation (VP)

VP is an important aspect that calls for a detailed explanation. It is a digital document created by the holder after issuance of VC when they have to provide evidence about their claims to any verifier. The VP contains within itself the information that will help verify that the data being presented was testified to by a legitimate issuer and remain valid.

A VP also has three different parts. The first element, ”Presentation Metadata,” holds the administrative information of the VP regarding its purpose, the context in which it will be used, and possible constraints or permissions on its actual presentation. At the heart of the VP is the element ”Verifiable Credential,” which embodies the actual data or claims about the holder previously issued and signed by an issuer worthy of trust. These would include simple identity information and complex qualifications. Finally, ”Proof” seals the VP with a cryptographic signature, guaranteeing that the holder is indeed the one making a presentation of the credential, and it has not changed since issuance. This layered structure empowers the VP to work like a secure and privacy-preserving channel of credential sharing, letting verifiers confidently trust the raised claim.

Verifiable Data Registry

A Verifiable Data Registry is one of the core infrastructural elements required in SSI. It acts as a trustless repository for storing and managing DIDs with their corresponding DID documents. The registry ensures that the data remains immutable and accessible for verification of digital identities. This type of registry is basically implemented on a blockchain or another form of distributed ledger technology, giving a decentralized approach for data management that prevents any point of failure or single point of control. In case a DID is presented for verification, any Verifier is allowed to verify the validity and authenticity of a DID document at any time. It includes public keys, authentication protocols, and service endpoints that enable secure interactions by the Verifiable Data Registry. This lets a verifier trust a DID subject's claims without contacting the issuer directly. It provides support for a tamper-resistant, verifiable record of DIDs through the Verifiable Data Registry and thus underpins the integrity and trust in the working model of SSI.

JSON Web Signature (JWS)

In the context of SSI, it is always necessary to delve into the technical details to realize how security and integrity within digital interactions come together. One of the integral elements is JWS, an ultimately compact and URL-safe means of representing digitally signed content.

A JWS is generally applied to sign a JSON object, which can later be used to assert data authenticity and integrity. Generally, the parts constituting a JWS are separated by a dot ('.'): The header, payload, and signature. The Header contains metadata about the token type and the signing algorithm used. The Payload is the signed data; under SSI, this would typically comprise claims within a Verifiable Credential. Conversely, on the other side, a Signature embeds a computed signature based on the encoded Header, the encoded Payload, and a secret, or in asymmetric crypto, a private key.

By signing a JSON object, a JWS issuer provides an equivalent cryptographic guarantee that it wrote the token and thus delivers data unchanged since its issuance; this property is fundamental in an SSI framework where entities must place trust in credentials they receive, both in terms of their validity and integrity.

Related Work

In this section, we critically examine foundational standards and guidelines proposed by the World Wide Web Consortium (W3C) and the OpenID Foundation that enable the implementation of secure, interoperable, and user-centric digital identity systems across diverse applications. Relevant research papers that align with these standards and contribute to the theoretical and practical advances in digital identity frameworks are also discussed. In the same respect, W3C's standardization of a Decentralized Identity and Verifiable Credential technologies, in a standard that includes, but is not limited

to, such as Decentralized Identifiers (DIDs) v1.0 in W3C Verifiable Credentials Working Group (2022), Verifiable Credentials Data Model v2.0 in W3C Verifiable Credentials Working Group (2024), and Verifiable Credentials Implementation Guidelines 1.0 in W3C Verifiable Credentials Working Group (2023) set a robust foundation for SSI systems. This enables secure, decentralized, and interoperable identity management through mechanisms proposed for issuance, presentation, and revocation of digital credentials. Even with the challenges of scalability, governance, and privacy, the W3C frameworks provide a useful guide on how to handle these challenges and hence provide the necessary technical measures for making digital credentialing technologies widely available and applicable. The standard outlines of OpenID are also discussed, which contain protocols and mechanisms that enable secure issuance of Verifiable Credentials. The contributions of OpenID in Verifiable Credentials focus on standardizing mechanisms for secure issuance and presentation. OpenID for Verifiable Credential Issuance 1.0 extends OpenID Connect protocols to define secure methods by which Verifiable Credentials are created, issued, and exchanged. It enables flows, such as the Authorization Code Flow and Preauthorization Code Flow, which improve user control, privacy, and security. This allows the credentials to be cryptographically signed, hence providing integrity and reducing the risks of tampering and leakage. Similarly, OpenID for Verifiable Presentations integrates Verifiable Credentials into OpenID Connect to enable secure, privacy-preserving identity assertions. It allows flows such as the Same Device Flow, which streamlines interactions within a single device, and Cross Device Flow, enabling secure credential presentation across multiple devices. Using cryptographic proofs, OpenID for Verifiable Presentations ensures that data disclosure is reduced to the bare minimum, increases interoperability, and creates trust without any need to verify directly with the issuer. These developments tackle some of the essential challenges in digital identity management and create a path toward robust user-centric solutions.

The W3C standardization has established a robust, interoperable foundation for tamper-evident digital attestations. However, while these standards provide the necessary technical measures for widespread adoption, they often lack specific implementation strategies for balancing organizational trust with lightweight holder mobility. Similarly, protocols such as OpenID4VC and OpenID4VP have successfully standardized secure issuance and presentation flows, effectively reducing the risks of data tampering and leakage. Despite these advancements, these protocols primarily focus on the communication layer and do not natively resolve the privacy risks associated with public revocation lists.

Feulner et al. (2022) examined the use of Self-Sovereign Identity (SSI) technologies in event ticketing systems, presenting a framework developed through a DSR approach and validated by a PoC. The study highlights SSI's potential to reduce fraud and control secondary markets but notes challenges like user onboarding and hardware requirements, emphasizing the need for further research. (Ferdous et al., 2023) introduced SSI4Web, a framework for passwordless, secure, user-controlled authentication via decentralized SSI wallets. The study highlights its potential to revolutionize web identity management with privacy-preserving and user-friendly alternatives to traditional password systems. Nevertheless, these studies highlight significant barriers, including complex user onboarding and a lack of inclusiveness for diverse systems.

Lux et al. (2020) examined integrating SSI with OpenID Connect, highlighting its potential to enhance user privacy, security, and autonomy in digital identity management, including IoT and PKI applications. The study emphasizes scalability and interoperability as key challenges for future research. (Giannopoulou, 2023) examined SSI's potential for user autonomy and decentralization, noting its innovative aspects but criticizing its lack of inclusiveness and individual burden. The study emphasizes the need for a comprehensive approach to address trust, accountability, and sociopolitical challenges in identity management.

Herbke and Yildiz (2022) proposed ELMO2EDS, a framework integrating digital credentials into SSI to address challenges in educational networks like EMREX. The study highlights its role in improving privacy, interoperability, and security within the educational ecosystem. (Arndt and Guercio, 2020) investigated blockchain in higher education for secure transcript management, demonstrating its potential for credential verification but highlighting challenges like scalability and interoperability, requiring further research. In the educational sector, those frameworks have been investigated to address the inefficiencies of traditional academic records. While successful in improving data integrity, these solutions often face challenges in scalability, interoperability, and the secure management of long-term credential validity across diverse educational networks.

Based on this survey of existing technologies, several critical gaps remain. First, current implementations often fail to differentiate DID methods based on the actor's role. Institutions require a high-trust, domain-linked identifier like did:web, whereas students need a stateless, infrastructure-independent method like did:key for seamless mobility. Second, traditional bitstring revocation lists or public registries can lead to student de-identification during status checks. There is a clear need for a mechanism that provides real-time verification while

maintaining a high level of k-anonymity. Lastly, there is limited evidence on how to integrate these new standards with existing university infrastructures, such as the Authorization Code Flow, to minimize the learning curve for users.

Materials and Methods

In this section, we delve into comprehensive analysis and systematic design that needs to be implemented in this paper. To ensure the robustness of the system, we first define the threat model and the scope of our functional and security testing. The detail of this section will consist of system architecture overview, the protocol of SSI that we use to develop, data flow diagram, the data model of each part like DIDs, data format, JSON schema, and proof type, also, with Verifiable Credentials verification. Lastly, we will end with Verifiable Credentials revocation.

Threat Model and Scope

The system is designed to mitigate three primary categories of threats. The first threat is credential forgery and tampering, which is the risk of unauthorized entities creating fake transcripts or altering existing ones to misrepresent academic achievements. Second, Identity impersonation is the second, which is the risk of a Man-in-the-Middle (MITM) attack or an unauthorized user claiming the identity of a legitimate student during the issuance or presentation process. The last one is privacy leaks and re-identification, which is the risk of exposing sensitive student data during revocation checks or through centralized database breaches. The scope of our testing evaluates how well the SSI framework addresses these threats across the entire lifecycle of a digital transcript—from issuance to verification and revocation. To validate the proposed SSI architecture, we define a testing scope that targets the primary vulnerabilities identified in our threat model. This scope encompasses functional Validation, security integrity, and privacy safeguards.

Importance of Decentralized Identity Management in Education

Integrating SSI and VC into digital transcript processes in educational settings marks a significant evolution from traditional transcript management systems. This new paradigm offers enhanced security, increased privacy, and superior efficiency, which starkly contrasts with the limitations of conventional methods.

Traditionally, universities and colleges manage student transcripts through centralized databases. These systems involve several steps: Collecting personal and academic information, verifying this data, creating an official record, and controlling access through centralized administrative protocols.

Despite being widely used, traditional transcript processes have several limitations. They rely on centralized systems and manual verification, which often result in delays, increased administrative workload, and inefficiencies. Additionally, verification requires direct communication with the university, making the process slow and difficult to scale. These limitations highlight the need for a more efficient and secure approach, such as decentralized identity solutions using SSI and Verifiable Credentials.

Self-Sovereign Identity Protocol

In the modern field of digital transcript management, we have oriented the development of our research toward two protocols: DIDComm and OpenID. DIDComm represents a decentralized communication protocol providing point-to-point secure communication between parties in the area of digital identity. It is a framework that allows people to share information on fully autonomous and secure terms, underpinning the SSI model upheld by our project. Conversely, OpenID is also such technology, governed by the OpenID Foundation, which complements this model of enabling users to log in to numerous internet services. Simplifying the process for end-users by providing them with single sign-on credentials to multiple platforms creates a user-friendly environment, thereby using our project's concept. The two protocols, in their functions, converge on the source integrity and usability with digital credentials like academic transcripts, providing a robust solution our project looks to capitalize on.

OpenID4VC

In this project, we decided to use the authorization code flow as the issuance model in Lodderstedt et al. (2024) to manage digital transcripts through an SSI system is strategically informed by several critical factors related to the existing infrastructure and university security requirements. Given that the university already maintains individual student accounts, the Authorization Code flow presents a seamless integration point. Students are familiar with logging into their university accounts, which typically provide access to a wide range of academic and administrative services. The familiarity of this process is important for user comfort and minimizes the learning curve associated with the adoption of new technologies. The whole process starts with the Authorization request sent by the Wallet Application to the Web Browser for authentication of the user and then redirection to the university's website for issuance of credentials. After authentication of the user and consent given, the university's website receives an authorization code from the Authorization Server, which is further redirected to the wallet through the Web Browser. This code is used by the wallet to request an access token from the Authorization Server, providing the necessary

credentials. Upon validation, the Authorization Server will issue the access token that the wallet will use to securely contact the Credential Issuer and request Verifiable Credentials. Validated, the Credential Issuer issues the requested credentials to the Wallet, completing the process of secure and authenticated issuance of credentials.

OpenID4VP

In this project, we have chosen the presenting credential model as a strategic choice of Terbu et al. (2024) designed to optimize the verification process in an SSI ecosystem. Such a decision makes much sense by virtue of its adaptability and user convenience in facilitating secure interactions across different devices. This starts the process where the wallet-in other words, the student/credential holder-initiates authorization to access a verifier's service. The verifier's website will respond with an authorization code, encapsulated in a URI, that permits resource access. Using that authorization code, the wallet may then request a "Request Object" containing specific details on the credentials the wallet intends to present. It is now on the wallet to return this structured and possibly encrypted Request Object to the verifier. Finally, the wallet is supposed to present the Verifiable Presentation-a digitally signed package of one or more Verifiable Credentials to the service endpoint at the verifier for validation.

Security Mechanism of API in SSI for VC Using TLS

In the framework of Verifiable Credentials with Self-Sovereign Identity, applying Transport Layer Security has several benefits that improve the overall security and trustworthiness of the system. First and foremost, TLS encrypts the data being passed between clients and servers, ensuring confidentiality so that sensitive information such as credentials, presentations, and revocation statuses will not get into unwanted hands. This encryption mechanism ensures users' privacy in sending their data over the transmission media.

Further, TLS ensures integrity through detection of the data during transit in case of unauthorized modification. In other words, any change in the data will be noted, hence ensuring that information sent to the client or server reaches its destination exactly the way it was sent. Protection against data tampering ensures authenticity and reliability at the point of reception.

TLS also provides robust authentication, verifying the identities of both servers and clients. This mutual authentication process ensures that clients are communicating with the legitimate server and not an imposter, thus preventing man-in-the-middle attacks. The optional client authentication adds an additional layer of security, confirming the identity of the clients to the server.

It also protects against eavesdropping, in which an attacker could intercept the communication between clients and servers to intercept and listen to the communication. Since TLS is encrypting the data exchanged in both directions, sensitive information cannot be disclosed to unauthorized parties; hence, it guarantees confidentiality for the data exchange.

Decentralized Identifiers (DIDs)

DIDs are indispensable for understanding the inner details of how SSI works by W3C Verifiable Credentials Working Group (2022). DIDs are a new identifier that empowers verifiable and self-sovereign identities in digital format. DIDs are the core of SSI, the base element that gives any entity complete control over its digital identity without a centralized registry.

DIDs are URI compliant designed to be globally unique, resolvable, highly available, and cryptographically verifiable. Usually, they are associated with cryptographic material, such as public keys and service endpoints, that enable secure, peer-to-peer communication. The owner of the DID utilizes the associated private key in proving control over the DID; hence, his identity is authenticated.

The key benefits of DIDs are that they can carry out this decentralized operation without reliance on any central registry or authority and are normally recorded on a blockchain or other form of distributed ledger. This helps ensure they can be tamper-resistant and resolvable anywhere around the world. In addition, this provides for a robust and resilient identity system with no single point of failure

In this project, we decided to advance the domain of digital identity employing two different SSI mechanisms in a strategic manner: did:web for the issuance and management of VCs and did:key for the handling of VPs. which related to the W3C Verifiable Credentials Data Model, which defines the structure and lifecycle of verifiable credentials and presentations, and the W3C Decentralized Identifiers (DID) specification, which provides the mechanism for creating and resolving decentralized identifiers. The research also reviewed DID resolution mechanisms, particularly the use of the did:web method for issuer identification and did:key for holder identities in the prototype system. This bifurcated approach is underpinned by a careful analysis of the unique advantages and compatibility of each DID method with the specific components of the SSI model.

did:web for Verifiable Credentials (VC)

'did:web' from Gribneau et al. (2024) is chosen for our VCs due to the good integration with organizational domain names and the possibility of leveraging already established web infrastructure. In this way, it would be granted that the VCs, most often issued by institutions having a large presence on the web are pinned to a fully recognizable and trusted domain of the university. This

will enable the issuing institutions to maintain control of issuance and revocation of their digital credentials within their existing web infrastructure, thereby enhancing trust while making the user experience frictionless. The format of ”did:web” is shown below:

did:web:[authority][path]

- did:web” is the identifier that indicates the DID is using the DID:web method
- [authority]” specifies the domain name (and optionally a port) of the entity hosting the DID document. This part follows the standard URI authority component format and might include a fully qualified domain name, such as example.com
- ”[path]” is optional and, if present, leads to a specific path where the DID document is hosted on the given domain. This part is similar to the path in a typical URL and may refer to a specific file or resource, e.g., .well-known/did.json

did:key for Verifiable Presentations (VP)

On the other hand, ”did:key” from Longley et al. (2022) was chosen for VPs because of its relative lightness and not requiring any sort of central registry or underlying infrastructure. As a matter of fact, VPs should really be very dynamic in nature and generated and presented by users on the fly; hence, the stateless lightness of did:key just makes it perfect for this application. It provides an instantaneous and effective way to build the presenter’s control over the VC being presented, so that transactions requiring minimal overhead are immediate and integral. The format of ”did:key” is shown below:

did:key:[multibase-encoded-multicodec-public-key]

- ”did:key” is the identifier indicating the use of the did:key method
- ”[multibase-encoded-multicodec-public-key]” is a single string that encodes the public key information. A method that encodes bytes in a variety of bases such as base58 and base64. This is used to adapt the binary content of public keys into textual forms that are easier to disseminate

Data Structures

Among the available data formats, there are two main types that every standard has to mention: JSON and JSON-LD. In this project, we decided to use JSON-LD. This stands out as a preferred choice because of its capability of enhancing plain JSON with the power of Linked Data. JSON-LD is designed to keep all the advantages of JSON. This includes simplicity and effectiveness for data serialization, adding features that promote linking and context awareness in data. This achieves by certain structural elements unique in JSON-LD.

JSON Schema

JSON Schema is a vocabulary that allows you to annotate and validate JSON documents. It describes the structure of data, the data types, the constraints on the data, and more. Essentially, JSON Schema defines the blueprint for what various fields in a JSON object can contain. It is similar to XML Schema for XML, but it is JSON-based. By using JSON Schema, you can ensure that data adheres to a certain format and contains the required types of information. In the context of JSON-LD, ”terms” typically refer to the keys or names that represent values within the document. These terms are defined within a given ”@context”:”https://example.jsonvc.com/schema” to ensure that they have consistent meaning across various systems. This is crucial for linked data and Semantic Web applications where data must be unambiguously understood and processed by different entities.

Verifiable Credential

Implementation of Verifiable Credentials in this work envelops all the necessary components for integrity, validity, and trust related to W3C Verifiable Credentials Working Group (2024). ”Credential metadata” conveys the contextual information of this credential: Identifier, type, issuer, and issue date. ”Credential Subject” defines whom or what this information is about, the holder of the transcript information. ”Credential Status” uses a revocation list to check whether a credential is active or revoked. ”Credential Schema”: The JSON schema to which the structure of the credential data must confirm is linked. Lastly, the ”Proof” section contains cryptographic signatures that make sure the credential was issued by an authorized issuer and hasn’t been tampered with. These all together ensure secure, verifiable, and reliable management of the credentials.

Verifiable Presentation

Verifiable Presentation in this project is implemented securely and in a trusted way with its key components: Credential Metadata provides the essential context, identifier, type, and holder details, establishing the presentation’s identity and scope that related to W3C Verifiable Credentials Working Group (2024). Verifiable Credential contains an array of credentials issued to the subject by the issuer, including metadata, subject details, status, schema, and proof. Last but not least, the Proof element provides the cryptographic guarantee that the presentation is real, unaltered, and signed by the holder, and it includes elements such as the type of cryptographic signature, creation date, purpose, verification method, and the JSON Web Signature. Together, these components deliver a robust mechanism for securing and validating Verifiable Presentations.

Verifiable Credentials Verification

Verifiable Credential (VC) is a digital statement made by an issuer about a subject. Any party with the required information may verify a VC. The process normally includes cryptographic techniques to ensure there is no forgery or tampering with the credential. One common way our project used signing credentials and verifying them was through JSON Web Signatures.

JSON Web Signatures (JWS)

JSON Web Signatures (JWS) (Jones et al., 2015) is another method for making a digital signature on a JSON object. Typical uses include signing of tokens and other pieces of data. Most JWS consists of three components: Header, payload and signature. The header contains the metadata that describes how the signature was generated. The payload merely encapsulates claims being made; in the case of VCs, this would mean statements about the subject of a credential. A digital-signing algorithm based on the header, the payload, and the secret signing key generates a signature.

Elliptic Curve Cryptography (ECC) and Ed25519

ECC is a public-key cryptography approach based on the algebraic structure of elliptic curves over finite fields. It provides the same level of cryptographic strength as RSA but with smaller key sizes, making it suitable for environments where resource efficiency is a concern in (Moriarty et al., 2016).

In this project, we have decided to employ Ed25519 in Bernstein et al. (2012) as our core cryptographic algorithm for signing VCs and VPs. The use of Ed25519 has its root in the usage of ECC, or elliptic curve cryptography-a key modern approach to encryption offering both high-level security and, at the same time, performance. This is because Ed25519 is mostly applied in new systems where security and/or performance is crucial-to be more concrete, thanks to efficiency, security features, and simpler, reliable implementation. RSA, despite being used rather widely, feels generally preferred for compatibility than for either performance or security reasons. Ed25519 was designed to operate very quickly with very high security and thus is very suitable for a system where high-security digital signature capability is important.

Verifiable Credentials Revocation

Verifiable Credentials Revocation is an important capability that allows the issuer to retract the validity of a credential before its date of expiry in case situations change. This would be for several reasons: Either the information in the credential is outdated or invalid, security concerns, or the subject of the credential requesting such revocation. In our project, we decided to use a revocation list as a strategy for the revocation process.

A Revocation List is a straightforward and widely used method for credential revocation in W3C Verifiable Credentials Working Group (2021). It involves maintaining a list (often publicly accessible) that marks credentials as revoked. The Verifiable Credential will have a part of ”credentialStatus” that provides metadata necessary to verify whether a credential has been revoked. An example of revocation list is following:

”000100000000011110000000000000000000000001000...”

Revocation lists are often implemented as simple bit strings where each bit represents a credential. A ”1” represents a revoked credential while A ”0” indicates the credential is still valid.

The above example represents the example credential status, the ‘”revocationListIndex”: ”4”’, you would look at the 4th position in this bit string to determine the credential’s status. If it is 1, the credential is revoked; if 0, it is still valid. If you compare in the example, the 4th position in this bit string is ”1”, so it means that the credential got revoked.

Security Mechanism of Bitstring Revocation List

In our project, we have increased the security and privacy level of the revocation list by contribution of k-anonymity principles; this technique not only ensures efficient credential status management but also protects privacy of a listed credential identity owner. We describe in this section the security mechanism and privacy enhancement in our bit string-based revocation list.

k-Anonymity for Privacy Protection

K-anonymity is a privacy-preserving technique used to prevent the identification of individuals within a dataset. It ensures that everyone’s data cannot be distinguished from at least k other individuals. In our system, k-anonymity is achieved by grouping data in such a way that every record is indistinguishable from at least k other records. This significantly reduces the risk of re-identification by ensuring that any single record cannot be uniquely identified. To protect the privacy of students, we implement k-anonymity in our bitstring revocation list. In our system, we maintain a revocation list with 200,000 records, which store the records of students over a span of five years. This means each year’s data consists of 40,000 records, providing k-anonymity with k = 40,000. It would mean that no single record on this list shall be uniquely identifiable because it will be indistinguishable from at least 39,999 other records.

Benefits of k-Anonymity

k-Anonymous intrinsically ensures better privacy; the data is not directly traceable. In case the attacker has access to the revocation list, it will not be possible to link a record with a particular student with high certainty. This

DOI: 10.3844/jcssp.2026.1577.1595

kind of record aggregation will reduce risks of deanonymization drastically.

System Architecture Overview

In this section, we try to analyze the security challenge of the traditional issuing transcript process. That is why we try to propose the architecture developed by implementing methodology of SSI and VC like we have shown in Figure 2. It describes a system architecture for the implementation of SSI and VC within a university context for the use case of issuing and verifying transcripts. The system is divided into three main actors: Issuer (University), Holder (Student/Alumnus), and Verifier (Employer or another educational institution). The registry, as part of the Issuer's infrastructure, is developed internally to comply with university policy.

Importance of Decentralized Identity Management in Education

Integrating SSI and VC into digital transcript processes in educational settings marks a significant evolution from traditional transcript management systems. This new paradigm offers enhanced security, increased privacy, and superior efficiency, which starkly contrasts with the limitations of conventional methods.

Traditionally, universities and colleges manage student transcripts through centralized databases. These systems involve several steps: collecting personal and academic information, verifying this data, creating an official record, and controlling access through centralized administrative protocols.

Issuer

The green box on the left hand side of Figure 2, illustrates the system architecture related to the 'Issuer' in a SSI framework, specifically focusing on the processes involved in managing and issuing VCs. Here is a step-by-step explanation:

- DID Controller: The University has a DID Controller that creates and manages DIDs for itself and its students. The DID is a unique identifier linked to the University domain (e.g., "did:web:university name")
- Verifiable Data Registry: This is a local MongoDB database where DIDs and their corresponding VCs are stored and managed
- DID Document: It defines the university's DID, including public keys and service endpoints, allowing others to interact securely with the university's DID
- Create Verifiable Credential (VC): The university issues a VC that represents the student's transcript. This VC is digitally signed by the university
- Authorization Server: This is a dedicated server or a software service responsible for providing security tokens. It authenticates the entity requesting access (such as a department within the university or an external party) and issues tokens that grant permission to perform specific actions like creating or revoking VCs
- Authorization Endpoint: This component of the authorization server handles interactive requests. It is where the user or service will be redirected to when they need to authenticate themselves and give consent for the issuance of a VC or any other service that requires permission
- Token Endpoint: After successful authentication and consent, this component issues access tokens, usually in the form of OAuth 2.0 tokens. These tokens are then used in HTTPs requests to the service endpoint for performing authorized operations, like accessing the registry to create or revoke VCs

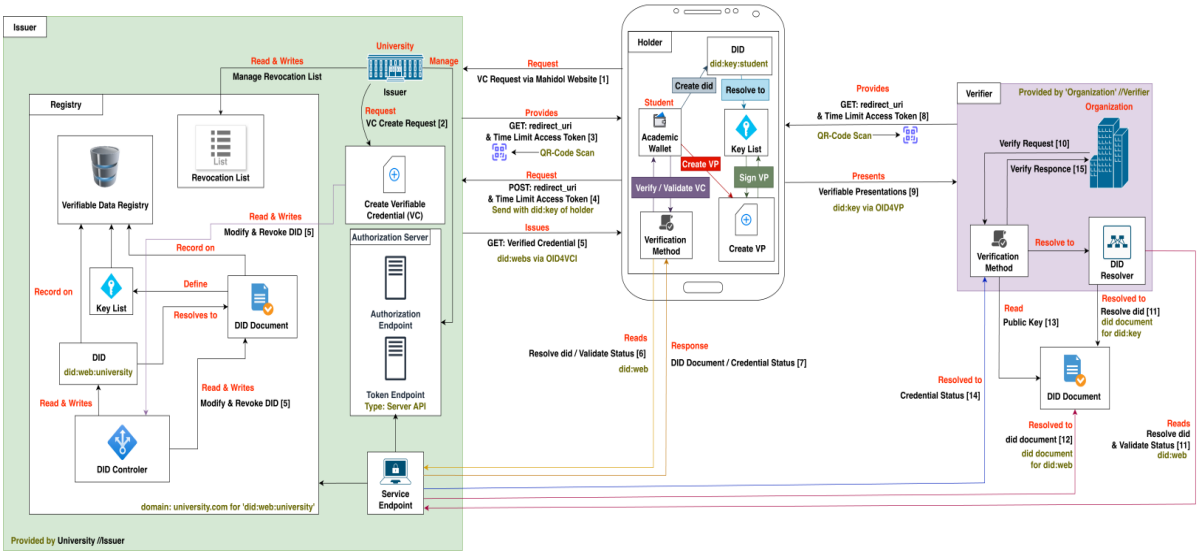

Fig. 2: The Architecture of SSI for VC-Based Digital Transcript

- Service Endpoint: This is the interface to act as an API that external verifiers can interact with to perform certain operations related to VCs. When a VC is presented to a verifier (e.g., an employer or another institution), the verifier will contact the service endpoint to validate the VC. This method is also used during VC issuance; the service endpoint would accept requests containing the access token provided by the token endpoint. It would then facilitate the VC's creation by the issuer, based on the credentials stored in the Verifiable Data Registry

Holder

At the center of Figure 2, illustrates the system architecture related to the "Holder" in an SSI framework, specifically focusing on the processes involved in managing and presenting VCs. Here is a step-by-step explanation:

- DID (Decentralized Identifier): This component serves as a unique identifier for the Holder (student or alumni), following the "did:key" method, which derives directly from a cryptographic key. The Holder generates a new DID to manage their identity within the SSI ecosystem, and the DID resolves to a key list containing public keys used for verifying digital signatures
- Academic Wallet: This is an application or digital wallet on the Holder's device that stores their VCs (such as digital transcripts) and manages their DIDs
- Key List: This refers to the set of keys associated with the Holder's DID. It is used to sign and verify information linked to their identity
- Create Verifiable Presentation (VP): Creating a Verifiable Presentation (VP) involves the Holder using their private key to sign the VP, providing cryptographic proof of ownership. This process compiles the necessary Verifiable Credentials (VCs) into a presentation format, allowing the Holder to share only the specific information they choose with a Verifier, thus ensuring privacy and security
- Verification Method: It is a tool or a process within the Academic Wallet that allows the Holder to validate their own VCs before sharing them. This step ensures that the VCs are still valid and have not been revoked or expired

Verifier

The purple box on the right hand side of Figure 2 illustrates the system architecture related to the "Verifier" in an SSI framework. The Verifier is typically an entity like an employer, a government agency, or another educational institution that needs to validate a holder's claim, such as a digital transcript. Here is a breakdown of the steps and components involved in the verification process:

- Verification Method: This method is a process, or a tool used by the Verifier to validate the presented VC or VP. This usually involves checking the digital signatures against public keys and verifying that the VC or VP has not been tampered with or revoked
- DID Resolver: The DID Resolver is a crucial component in the SSI architecture. It is used by the Verifier to retrieve the public keys or other relevant verification data from a DID Document corresponding to the DID included in the VC or VP

Data Model

This section outlines the data model employed in our project, which is fundamental to the functioning and integrity of the SSI for VC based digital transcript system.

Cryptographic and Signing Procedures

The system's security is anchored by the Ed25519 signature algorithm , which provides high-performance asymmetric encryption with smaller key sizes compared to RSA.

- Key Generation (Algorithm 1): Defines the derivation of a public key from a generated Ed25519 private key to establish unique identities for both the Issuer (did:web) and the Holder (did:key)
- Signing JSON Data (Algorithm 2): Details the process of serializing JSON documents into UTF-8 bytes and applying a private key signature, which is then output as a Base58-encoded string to ensure compact transmission
- Verify Signature (Algorithm 3): Outlines the verification mechanism where the Base58 signature and public key are decoded to validate the integrity of the JSON message; the process returns a boolean result to confirm if the data has been tampered with

Algorithm 1: Key Generation

```
function GenerateED25519KeyPair
    private_key ←
      Ed25519PrivateKey.generate()
    public_key ← private_key.public_key()
    return private_key, public_key
```

Decentralize Identifier

In this section, we delve into the implementation and significance of DIDs within the SSI for VC based digital transcript system. We begin by detailing the DID of the issuer, specifically utilizing the did method, followed by

the process of generating the DID for the issuer. Subsequently, we explore the structure and content of the DID Document associated with the issuer. Moving forward, we examine the DID of the holder, implemented through the did method, and outline the steps involved in generating the holders DID. Finally, we detail the components of the DID Document for the holder.

Algorithm 2: Signing JSON Data

```
function SignJSON(private_key, json_data)
    Input  : private_key, json_data
    Step 1: Convert JSON data to bytes
    message_bytes ← json.dumps(json_data).encode('utf − 8')
    Step 2: Sign the message bytes using the private key
    signature ← private_key.sign(message_bytes)
    Step 3: Encode the signature using Base58
    signa_base58 ← base58.b58encode(signature).decode('utf − 8')
    Step 4: Return the Base58-encoded signature
    return signa_base58
```

Generating Issuer DID (Algorithm 4): Describes the creation of a did:web identifier by binding a unique UUID to the university's domain and embedding the Base58-encoded public key into a standardized DID document.

Generating Holder DID (Algorithm 5): Outlines the generation of a did:key identifier, which prepends a multicodec prefix (0xED01) to the public key bytes to create a stateless, infrastructure-independent identity.

Algorithm 3: Verify Signature

```
function VerifySignature(public_key, json_data, signature)
    Input  : public_key, json_data, signature
    Step 1: Decode the base58-encoded public key.
    public_key_bytes ← DecodeBase58(public_key_base58)
    Step 2: Initialize the verification key using the decoded public key.
    verify_key ← VerifyKey(public_key_bytes)
    Step 3: Convert the JSON data to bytes.
    message_bytes ← json.dumps(json_data).encode('utf − 8')
    Step 4: Decode the base58-encoded signature to bytes.
    signature_bytes ← DecodeBase58(signature_base58)
    Step 5: Attempt to verify the signature using the verification key.
    if verify_key.verify(message_bytes, signature_bytes) succeeds then
        return true
    else
        return false
```

Algorithm 4: Generating DID of Issuer

```
function generate-did-issuer
    Step 1: Generate Ed25519 key pair
    (private_key, public_key) ← generateEd25519KeyPair()
    Step 2: Generate a UUID for the user
    user_uuid ← str(uuid.uuid4())
    Step 3: Create the DID String
    did ← "did:web:university:user:{user_uuid}"
    Step 4: Encode the public key to Base58 format
    public_key_base58 ← Base58Encode(publicBytes(public_key))
    Step 5: Create the DID Document
    did_document ← createDIDDocument(did, public_key_base58)
    Step 6: Return issuer DID as a DID web document
    return did_document
```

Algorithm 5: Generating DID of Holder

```
function generate-did-holder
    Step 1: Generate Ed25519 key pair
    (private_key, public_key) ← generateEd25519KeyPair()
    Step 2: Decode the public key from Base58 format
    public_key_bytes ← decodeBase58(public_key_base58)
    Step 3: Concatenate Multicodec prefix for Ed25519 public key
    multicodec_ed25519 ← b'0xED01' + public_key_bytes
    Step 4: Encode the multicodec Ed25519 into Base58 format
    did_key ← "did:key:z" + encodeBase58(multicodec_ed25519)
    Step 5: Return holder DID as a DID key
    return did_key
```

Revocation and Privacy Mechanisms

To address privacy during status checks, the system utilizes a k-anonymous bitstring revocation list.

- Initiate Revocation List (Algorithm 6): Handles the initialization of a 200,000-record dataset, shuffling and grouping student records by graduation year to ensure that each entry is indistinguishable from at least 39,999 others (k = 40,000)
- Compress and Encode (Algorithm 7): Optimizes the bitstring for transmission by applying GZIP compression and Base64 URL encoding, significantly reducing storage overhead while maintaining data integrity

- Decompress Bitstring (Algorithm 8): Reverses the encoding and compression to restore the original bit list for local status verification by the Holder or Verifier
- Check Revocation List (Algorithm 9): Integrates the full verification flow: extracting the credentialStatus ID, retrieving the list via a GET request, and validating the specific status index (0 for valid, 1 for revoked)

Testing and Results

In this section, we present the functional testing conducted to validate the core components of the SSI for VC systems. The tests are designed to ensure that each component (Issuer, Holder, and Verifier) operates as intended, providing the necessary functionality to support secure and reliable identity management.

Algorithm 6: Initiate Revocation List

```
function InitiateRevokeList
    Step 1: Initialize Parameters
    total_students ← 200,000
    groups ← 5
    s_per_group ← total_students ÷ groups
      //total students per group
    start_year ← 2019
    Step 2: Generate and Shuffle Student List
    for i in range(total_students) do
        students ← f'Student_{i+1}
        shuffle(students)
    Step 3: Group Students by Graduation Year
    grouped_students ← {}
    for i in range(groups) do
        group_name
          ← f'Year_Graduate{start_year+
          i+1}'
        start_idx ← i × s_per_group
        end_idx ← (i+1) × s_per_group
        student_chunk ← students[start_idx :
          end_idx]
        grouped_students[group_name] ←
          student_chunk
    Step 4: Save Grouped Students to JSON File
    'revocationList.json' ←
      grouped_students
```

Algorithm 7: Compress and Encode Bitstring

```
function CompressEncodeBitstring(bitstring)
    Input  :bitstring //The bitstring as
      a list of bits
    Step 1: Convert bitstring to Byte Array
    bitstring_bytes ← bytearray(bitstring)
    Step 2: Compress bitstring_bytes Using GZIP
    compressed ← gzip.compress(bitstring_bytes)
    Step 3: Encode Compressed Data Using
      Base64 URL
    encoded ←
      base64.urlsafe_b64encode(compressed).rstrip(b'
      =')
    encoded_string ← encoded.decode('utf-8')
    return encoded_string
```

Algorithm 8: Decompress Bitstring

```
function DecompressBitstring(encoded_bits,
  size_kb)
    Input  :encoded_bits, size_kb
    Step 1: Decode the base64url-encoded
      bitstring
    decoded ←
      base64.urlsafe_b64decode(encoded_bits +
      '==')
    Step 2: Decompress the decoded bitstring
      using GZIP
    decompressed ← gzip.decompress(decoded)
    Step 3: Convert the decompressed data to a list
      of bits up to size_kb
    bitstring ← list(decompressed[:size_kb])
    Step 4: Return the decompressed bitstring as a
      list of bits
    return bitstring
```

Testing Setup

To evaluate the implementation of the Self-Sovereign Identity (SSI) framework, we designed and deployed a controlled virtual environment that simulated the interaction between the primary components: Issuer, Holder, and Verifier. This setup aimed to replicate a real-world SSI deployment from our proof of concept designed while detailing test case and analysis of system functionality and security.

We utilized a virtual machine (VM) environment to create isolated instances for each component of the SSI framework. Three separate VMs were provisioned to represent the Issuer, Holder, and Verifier roles. This architecture ensured a distributed and independent execution of each module, simulating the decentralized nature of an SSI system. The VMs were configured using VirtualBox, each with tailored resource allocations to optimize their individual processes.

The Issuer VM was configured as the central entity responsible for creating and issuing Verifiable Credentials (VCs). The Issuer implemented cryptographic signing using the Ed25519 algorithm and maintained a secure local database for managing credential schemas and revocation lists. Additionally, the Issuer provided APIs for credential issuance, validation, and revocation, ensuring seamless integration with the Holder and Verifier components.

The Holder VM hosted a digital wallet application designed to manage Decentralized Identifiers (DIDs) and Verifiable Credentials. This wallet allowed users to

generate DIDs, receive credentials from the Issuer, and compile Verifiable Presentations (VPs) for verification. The wallet application leveraged JSON Web Signatures.

Algorithm 9: Check Revocation List

```
function CheckRevocationList(vc_credential)
    Input  : vc_credential
    Step 1: Extract relevant parts from the
      credentialStatus ID
    id_revoke_list ←
      vc_credential["credentialStatus"]["id"].split("/")

    base_url ← "https:universityLink.com"
    url_path ← join(id, "/")
    url ← base_url + url_path

    Step 2: Send a GET request to the revocation
      list URL
    response ← requests.get(url)
    revoke_list ← response.json()
    index ← vc_credential["credentialStatus"]
      ["statusListIndex"]

    Step 3: Decompress and decode the bitstring
      from the revocation list
    bitstring ←
      decompress_bitstring(revoke_list["compressed_bitstr"],
      200000)

    Step 4: Verify the revocation status using the
      decoded bitstring
    if verify_status(bitstring, index, 1) == 0 then
        return true
    else
        return false
```

(JWS) to ensure the cryptographic integrity and authenticity of the VPs. The Holder's interface was further tested for usability and compliance with privacy-preserving principles.

The Verifier's VM was configured to validate the VPs presented by the Holder. The Verifier implemented a DID resolver to fetch public keys associated with the Issuer's DID and verify the digital signatures of the VPs. A secure API endpoint was established to process VP requests, ensuring data integrity and compliance with credential revocation policies.

To simulate a secure and realistic environment, we established a private network connecting the Issuer, Holder, and Verifier VMs. All communications were conducted over HTTPS, secured with TLS encryption, to prevent interception or tampering during data exchange. The network design ensured low latency and high reliability for seamless interactions between components.

Testing Workflow

Figure 3 details the functional tests of the Issuer, which involve verifying the processes of credential issuance and management.

Next, we examine the functional tests of the Holder, focusing on the receipt, storage, and presentation of credentials. Finally, we explore the functional tests of the Verifier, ensuring that the verification of credentials is performed accurately and efficiently. The details of each process will be explained this section.

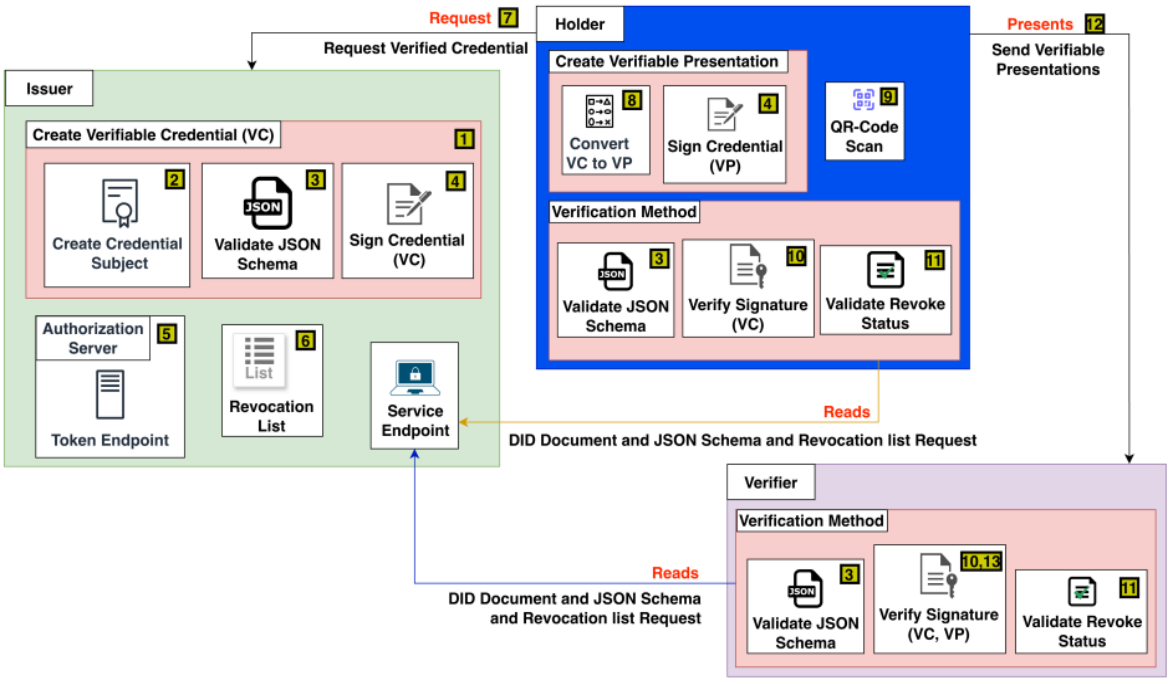

Fig. 3: Overview of Testing Workflow

Issuer Test Cases

In this subsection, we examined potential scenarios specific to the Issuer component within the SSI system. The detail of each test case has explained as following.

- Testing of creating Verifiable Credentials (VCs): To validate the creation of Verifiable Credentials, we create a test process on Figure 3: Box number 1, which focuses on the accurate population of essential fields such as ”@context,” ”id,” ”type,” ”issuer,” and ”issuance date.” The VC creation method ensures these fields adhere to the specified standards, resulting in reliably formatted and trustworthy verifiable Credentials. The process involves generating a credential and thoroughly examining its fields for correctness and compliance with the expected structure
- Testing of creating credential subject: The test process on Figure 3: box number 2, that aims to ensure the proper creation of the ’credential subject’ section in a Verifiable Credential, which contains critical details about the credential holder, such as transcript information. Simulated data is used to verify that all fields are accurately populated and align with data from the university database. The process involves creating the ’credential subject’ and meticulously examining its fields for integrity and completeness
- Testing of validating JSON schema: This test ensures the accuracy of the schema validation method for verifiable credentials, verifying that credentials conform to the specified schema. We presented it in box number 3 in Figure 3. It evaluates whether all necessary fields are included and identifies missing or incorrect data. The process involves validating a Verifiable Credential against the schema and examining the results for any validation errors, ensuring the consistency and reliability of issued credentials
- Testing of signing VCs and VPs: In Figure 3, box number 4 focuses on verifying the accuracy of the digital signature process for JSON documents. It ensures that digital signatures are generated accurately, safeguarding the authenticity and integrity of the credentials. The procedure involves applying a digital signature to a JSON document and confirming its validity using the associated public key, ensuring proper implementation and reliable security measures.
- Testing of the Access Token: In box number 5 in Figure 3, we tried to evaluate the server’s ability to authorize access securely using access tokens. By examining responses to both valid and invalid tokens, it ensures that only authorized users can access server resources, thereby safeguarding the system’s security and integrity. The process involves sending requests with various token statuses to confirm the proper enforcement of access control.
- Testing of creating a revocation list: Box number 6 in Figure 3 examines the method for generating the revocation status index of credentials, ensuring its accuracy and reliability. The process supports holders and verifiers in effectively identifying revoked credentials. By comparing generated indexes to expected outcomes, the test provides the integrity and trustworthiness of the revocation mechanism.

Holder Test Cases

In this subsection, we explore potential scenarios related to the Holder component within the SSI system. The detail of each test case is explained below.

- Testing of VC to VP conversion: Box number 8 is a process to validate the transformation form of Verifiable Credentials (VC) into a Verifiable Presentation (VP). It evaluates the method’s ability to identify the correct credential type from the holder’s wallet and convert it accurately into a VP format suitable for verification. A sample VC is used to ensure that the resulting VP meets expectations, confirming the accuracy and reliability of the transformation process
- Testing of QR code collection: In box number 9 in Figure 3, we tried to examine the process of extracting and interpreting an access token from a QR code displayed on the verifier’s web page within the OpenID for Verifiable Presentations (OpenID4VP) framework. Ensure that the method accurately decodes the QR code to retrieve the token required for the holder to initiate a presentation request. The process involves simulating a QR code scan and verifying that the extracted token conforms to the expected format and content
- Testing of verifying the signature of a VC: Box number 10 in Figure 3 verifies the correctness of the signature verification process for a Verifiable Credential (VC). Calling an API to request a DID Document obtains the public key necessary for verifying the digital signature within the VC. In the test method, a sample VC is used together with proof to see if this actually can correctly validate a signature that ensures an authentic, untampered credential. It tests for valid signature verification on a sample VC. It verifies the correct behavior of a verification mechanism when it comes to the recognition of a proper or invalid signature
- Testing of verifying the revocation status: This test case is represented in box number 11 in Figure 3 to evaluate the accuracy of the process used to verify the revocation status of a Verifiable Credential (VC). The method determines whether a given VC appears in a sample revocation list. If the VC is active, the method returns ”true”; if revoked, it returns ”false.” The process involves comparing the sample VC against the revocation list to ensure the status is correctly identified and accurately reflects the credential’s validity

Verifier Test Cases

In this subsection, we explored potential scenarios related to the Verifier component within the SSI system. The details of each test case will be explained.

- Testing process of verifying the signature of verifiable presentation (VP): This test case is represented in box number 13 in Figure 3, which examines the accuracy of verifying the digital signature embedded within a Verifiable Presentation (VP). A sample VP would need to be taken as input to determine whether the signature is valid, returning a boolean result of ”True” for valid signatures and ”False” for invalid signatures. This step is crucial because it will ensure the verifier can validate VPs. Provide a sample VP as an input for the ”Verify VP” method and check whether the output is exactly like the expected validation result

Issuing and Presenting Test Cases

This test aims to validate the process of receiving a Verifiable Credential (VC) from the issuer by calling the issuer’s API endpoint (/vc/token). This test ensures that the method responsible for retrieving VCs from the issuer functions correctly. A sample VC data compares the expected output when calling the endpoint. If the process encounters any error, it should respond with ”None” to indicate the unsuccessful operation. The methodology involves API calls to the issuer and verifying that the retrieved VC matches the expected sample data. The objective of this test is to validate the process of presenting a VP to the verifier by calling the appropriate API endpoint. This test ensures that the method for sending a VP to the verifier works correctly. The method calls the API endpoint specified in the VP definition and redirects to the endpoint for sending VPs. If the process is successful, it returns the verifier’s response; otherwise, it returns ”None.” The methodology involves sending a sample VP to the verifier’s API and verifying that the response indicates successful receipt and validation of the VP.

Results

The testing done on the Issuer, Holder, and Verifier proved that the SSI implemented works as expected. All the essential functionalities, like issuance, presentation, validation, and revocation checking without errors. Based on these test results, it is confirmed that the system developed is reliable and secure. Summary of testing results is shown in Table 1.

In addition to security tests, performance benchmarking was conducted using the pytest-benchmark framework to evaluate the efficiency of the prototype across the three primary actors. The results indicate that lightweight local operations, such as Verifiable Presentation (VP) verification (0.81 ms) and token verification (0.046 ms), execute with high efficiency. Cryptographic tasks involving Ed25519 and JWS, including JSON signing (5.32 ms) and revocation list checking (2.70 ms), demonstrate moderate processing times. In contrast, higher latency was observed in schema validation (48.32 ms) and Verifiable Credential (VC) verification (33.76 ms) due to complex structural validation and cryptographic overhead. The most resource-intensive operations were those involving network communication, specifically retrieving a VC (315.31 ms) and sending a VP to the verifier (209.71 ms). Overall, the benchmark data confirms that while local verification is nearly instantaneous, system latency is primarily driven by network-dependent interactions and multi-step credential generation. Comprehensive details regarding the execution time and throughput for the issuer, holder, and verifier are presented in Table 2.

In summary, this POC reflects that the SSI for a VC-based digital transcript system can ensure secure management of digital identities and verifiable credentials. Functional testing of the issuer, holder, and verifier components concluded that the system is robust enough to meet the necessary standards in terms of integrity and authenticity.

Only comprehensive validation mechanisms, strong access controls, and rigorous verification protocols can raise the correctness and reliability of the SSI for VC-based digital transcript systems while offering a secure solution for managing digital identities and credentials. Completing this POC lays a strong foundation for further development and real-world implementation of SSI technology.

Such detailed testing results in this chapter underline the SSI for the VC-based digital transcript system’s robustness and dependability. The functional tests were successfully passed, proving that the system meets the necessary standards of security, integrity, and performance. The system is secure in managing digital identity and Verifiable Credentials, which provides a good stepping stone for system development and actual deployment.

Table 1: Test Results Summary

Test Name	Security Test	Passed
Create VCs	Fields are correctly	✓
Create credential subj.	Subject are correctly	✓
Validate JSON schema	Schema compliance	✓
Sign VCs and VPs	Valid digital signatures	✓
Access token	unauthorized access	✓
Create revocation list	Accurate revoke status	✓
Convert VCs to VPs	VC to VP Integrity	✓
QR Code collection	Secure token extraction	✓
Verify VC signatures	Credential authenticity	✓
Verify revoke status	Revocation check	✓
Verify VP signatures	Valid VP signature	✓
Request VC	Secure VC retrieval	✓
Send VP to verifier	Correct VP delivery	✓

Table 2: Performance Evaluation

Operations	Mean Time (ms)	Throughput (ops/sec)
Issuers:		
Token verification	0.046	21,763
Credential subject creation	0.140	7,123
JSON signing	5.32	188
DID document loading	10.53	95
Revocation list indexing	11.41	88
Credential creation	71.5	14
Schema validation	50.83	20
Holders:		
Present VC to VP	1.13	884
Sign JSON	1.27	788
Revocation list check	2.7	371
Verify VC	33.86	29
Generate QR content	68.02	14.7
Request VP definition	119.37	8.38
Send VP to verifier	209.71	4.77
Retrieve VC	315.31	3.17
Verifiers:		
Verify VP	0.81	1,232
Check revocation list	2.72	368
Verify VC	33.76	29.6
Verify schema	48.32	20.7

Discussion

The successful functional testing of the proof-of-concept (POC) system conclusively demonstrates the technical feasibility and robust reliability of implementing an SSI and VC framework for digital academic transcripts. The proposed architecture, which strictly adheres to W3C and OpenID standards, effectively addresses the core challenges of integrity, privacy, and interoperability inherent in traditional educational record management.

Interpretation of Core Results

The successful issuance and verification of VCs confirm that the system reliably guarantees data integrity and authenticity. The use of JSON Web Signatures (JWS) with the high-security Ed25519 algorithm provides a non-repudiable proof of issuance, directly combating credential forgery. Furthermore, the robust verification, including a real-time check against the compressed, k-anonymous revocation list, ensures reliance on invalid or expired credentials is technically impossible.

Strategic Design Choices and Implications

Our design choices reflect a strategic balancing act between institutional trust and holder autonomy (Giannopoulou, 2023).

- DID Method and Interoperability: The dual use of did:web for the Issuer (linking to an established domain for trust) and the lightweight did:key for the Holder is a critical architectural finding. This bimodal strategy upholds the SSI principle of sovereign control while ensuring broad acceptance. By leveraging OpenID4VC and OpenID4VP, the system is standardized for integration with external Verifiers (Lux et al., 2020), addressing scalability and proprietary adaptor concerns (Arndt and Guercio, 2020)
- Privacy-Preserving Revocation: The deployment of a k-anonymous bitstring revocation list offers a practical and secure solution. By grouping students over a five-year period (k = 40,000), any single revocation entry is made indistinguishable from thousands of others, significantly minimizing the risk of deanonymization during the status check process

Limitations and Future Directions

While the functional validity of the system is confirmed, the current implementation is a proof-of-concept confined to a simulated virtual environment. This limits the scope of the findings in terms of large-scale deployment.

- Scalability Testing: The current system uses a local MongoDB for the Verifiable Data Registry. Future research must focus on testing performance, latency, and throughput under real-world traffic and a significantly larger credential issuance volume to evaluate the scalability of this architecture
- User Adoption and Onboarding: The user experience, particularly the complexity of key management and wallet interaction for non-technical users, remains a practical challenge. Future work should include extensive usability testing and the development of simplified onboarding flows to facilitate widespread adoption within an academic setting
- Regulatory and Governance Frameworks: A successful real-world pilot would require integrating regulatory compliance (e.g., GDPR, FERPA) directly into the schema and governance model. Research into aligning the decentralized control model with existing legal frameworks is necessary for full implementation

Conclusion

This study successfully developed and validated a proof-of-concept system for managing digital academic transcripts using Self-Sovereign Identity (SSI) and Verifiable Credentials (VC). Our work provides a robust, decentralized, and cryptographically secure framework that directly addresses critical issues of fraud, privacy, and

poor interoperability found in traditional centralized systems.

The core achievement is the establishment of a fully functional and secure SSI infrastructure. This infrastructure is defined by W3C and OpenID standards, featuring a strategically differentiated use of did:web for the issuing institution and did:key for the student holder. The comprehensive functional testing demonstrated the system's reliability across all phases: secure on-demand issuance, cryptographic signing (Ed25519/JWS), privacy-enhanced credential presentation (OpenID4VP), and robust revocation verification via a k-anonymous bitstring. This technical validation confirms that SSI and VC technologies are not only suitable but highly effective for revolutionizing educational record management.

While this POC lays a strong foundation, future work must focus on the practical deployment challenges to move toward real-world adoption. This includes rigorous scalability testing under high load, optimizing user onboarding procedures, and conducting pilot programs within educational institutions to refine the system and address emerging technical and governance challenges.

Acknowledgment

The authors wish to acknowledge the valuable collaboration and support from Dr. Chakan Pramkaew from NDID Co., Ltd. and the Verifiable Credentials Working Group under Electronic Transaction Development Agency (ETDA).

Funding Information

This research did not receive any funding.

Author's Contributions

Naphat Khajohn-udomrith: Designed the system architecture, studied methodology, implemented the PoC system, and performed all functional and security testing.

Ittipon Rassameeroj: Conceived the original research concept, studied design and methodology, supervised the overall research direction.

Ethics

This study is a technical proof-of-concept involving system architecture and cryptographic protocols. It did not involve human subjects, animal subjects, or the use of identifiable personal data, as all testing utilized simulated data. Therefore, institutional ethical approval was not required. Conflicts of Interest and Data Availability: The authors declare no conflicts of interest. The synthetic data models and system logs are available from the corresponding author upon reasonable request.

References

Arndt, T., & Guercio, A. (2020). Blockchain-Based Transcripts for Mobile Higher-Education. International Journal of Information and Education Technology, 10(2), 84–89. https://doi.org/10.18178/ijiet.2020.10.2.1344

Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., & Yang, B.-Y. (2012). High speed high-security signatures. Journal of Cryptographic Engineering, 2(2), 77–89. https://doi.org/10.1007/s13389-012-0027-1

Ferdous, M. S., Ionita, A., & Prinz, W. (2023). SSI4Web: A Self-sovereign Identity (SSI) Framework for the Web. Blockchain and Applications, 4th International Congress, 595, 366–379. https://doi.org/10.1007/978-3-031-21229-1_34

Feulner, S., Sedlmeir, J., Schlatt, V., & Urbach, N. (2022). Exploring the use of self-sovereign identity for event ticketing systems. Electronic Markets, 32(3), 1759–1777. https://doi.org/10.1007/s12525-022-00573-9

Giannopoulou, A. (2023). Digital Identity Infrastructures: a Critical Approach of Self-Sovereign Identity. Digital Society, 2(2), 18. https://doi.org/10.1007/s44206-023-00049-z

Gribneau, C., Prorock, M., Steele, O., Terbu, O., Xu, M., & Zagidulin, D. (2024). did:web Method Specification.

Herbke, P., & Yildiz, H. (2022). ELMO2EDS: Transforming Educational Credentials into Self-Sovereign Identity Paradigm. 2022 20th International Conference on Information Technology Based Higher Education and Training (ITHET), 1–7. https://doi.org/10.1109/ithet56107.2022.10031276

Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Signature (JWS). https://doi.org/10.17487/rfc7515

Lodderstedt, T., Yasuda, K., & Looker, T. (2024). OpenID for Verifiable Credential Issuance - draft 15.

Longley, D., Zagidulin, D., & Sporny, M. (2022). The did:key Method v0.7.

Lux, Z. A., Thatmann, D., Zickau, S., & Beierle, F. (2020). Distributed-Ledger-based Authentication with Decentralized Identifiers and Verifiable Credentials. 2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), 71–78. https://doi.org/10.1109/brains49436.2020.9223292

Moriarty, K. Jonsson, J., Rusch, A., & Kaliski, B., (2016). PKCS #1: RSA Cryptography Specifications Version 2.2. https://doi.org/10.17487/rfc8017

Terbu, O., Lodderstedt, T., Yasuda, K., & Looker, T. (2024). OpenID for Verifiable Presentations - draft 23.

W3C Verifiable Credentials Working Group. (2021). Revocation List 2020. A Privacy-Preserving Mechanism for Revoking Verifiable Credentials. https://w3c-ccg.github.io/vc-status-rl-2020/

W3C Verifiable Credentials Working Group. (2022). Decentralized Identifiers (DIDs) v1.0. Implementation Guidance for Verifiable Credentials. https://www.w3.org/TR/vc-imp-guide/

W3C Verifiable Credentials Working Group. (2023). Verifiable Credentials Implementation Guidelines 1.0. Implementation Guidance for Verifiable Credentials. https://www.w3.org/TR/vc-imp-guide/

W3C Verifiable Credentials Working Group. (2024). Verifiable Credentials Data Model v2.0. Core Architecture, Data Model, and Representations. https://www.w3.org/TR/did-1.0/