International Journal of Advances in Electrical Engineering

E-ISSN: 2708-4582
P-ISSN: 2708-4574
IJAEE 2023; 4(2): 34-42
© 2023 IJAEE
www.electricaltechjournal.com
Received: 02-05-2023
Accepted: 05-06-2023

Sujeet Raosaheb Suryawanshi
Shri Jagdishprasad Jhabarmal Tibrewala University, Churu, Churela, Rajasthan, India

Dr. Prashant B Kumbharkar
JSPM's Rajarshi Shahu College of Engineering, Ashok Nagar, Tathawade, Pimpri-Chinchwad, Maharashtra, India

Dr. Shailesh Kumar
Shri Jagdishprasad Jhabarmal Tibrewala University, Churu, Churela, Rajasthan, India

Correspondence
Sujeet Raosaheb Suryawanshi
Shri Jagdishprasad Jhabarmal Tibrewala University, Churu, Churela, Rajasthan, India

Self-Sovereign identity approach in oauth 2.0

Sujeet Raosaheb Suryawanshi, Dr. Prashant B Kumbharkar and Dr. Shailesh Kumar

DOI: https://doi.org/10.22271/27084574.2023.v4.i2a.44

Abstract
Identity management, encompassing authentication and authorization within a networked environment, stands as a paramount security aspect. Over time, various identity management paradigms have evolved, progressing from the isolated silo model to the federated model and, more recently, the self-sovereign identity (SSI') approach. Notably, SSI empowers users to autonomously oversee their own data, irrespective of organizational involvement, through the utilization of emerging blockchain technology. Numerous ongoing studies are exploring the potential of SSI.
Nevertheless, SSI adoption has remained limited due to its inherent compatibility issues and user inconveniences, stemming from an unfamiliar user experience and a nascent development phase. In response, this research paper proposes a novel SSI approach rooted in blockchain technology that aligns with the widely accepted and mature OAuth 2.0 standard. This blockchain-based model ensures users' data sovereignty, allowing them to wield control over their information in a decentralized manner, free from reliance on specific monopolistic service providers.
The proposed model boasts high usability and scalability, as it can be readily embraced and implemented by users and clients familiar with existing OAuth protocols. The feasibility of the proposed model is confirmed through its implementation, accompanied by a thorough security analysis. It is anticipated that this innovative model will play a pivotal role in advancing both blockchain technology and the adoption of self-sovereign identity solutions.

Keywords: Self Sovereign identity, identity management, OAuth

1. Introduction
In the realm of internet-based identity management, models for user authentication and authorization have continuously evolved to address the shortcomings of previous approaches. In the initial identity approach, individual service providers held user information and directly carried out user authentication. However, the identity approach had limitations, primarily because authentication could only be conducted by the service provider that possessed the user data. This limitation subsequently resulted in the issue of password fatigue among users as the diversity of internet services expanded.
In order to address the shortcomings of the identity approach, the federated model was introduced. This approach aimed to resolve the issue by delegating authentication to a specific service. The federated approach was implemented in multiple variations. One of these variations was the single sign on, in which the delegated authentication server handles all authentication tasks within a single network, employing the SAML (Security Assertion Markup Language) protocol. Another approach involved OAuth, in which various third-party services delegated the responsibility of authentication and authorization to a specific service, such as Github, Apple ID, utilizing the HTTP protocol. While the federated model successfully alleviated password weariness or exhaustion that users experience due to the repeated need to remember and manage multiple passwords for various online services, it introduced new challenges. The authentication service ended up accumulating vast quantities of user data, giving rise to both management and security concerns. Such a scenario raised the potential for privacy violations as the service could potentially misuse the user data it had accumulated. Additionally, a significant issue emerged where third-party services could experience disruptions in functionality if the authentication service experienced temporary failures or faced permanent suspension.
The user-centric model emerged as a solution to provide users with greater control over their data and to address the issues encountered in previous models. An example of such a service was Open ID, but its widespread adoption faltered due to the unfamiliarity of its authentication process.

Subsequently, several authentication services with similarities to OpenID were developed. However, many of these services closely resembled the federated model, lacking significant public appeal. Their susceptibility to phishing attacks further limited their adoption.

The emergence of blockchain technology paved the way for the self-sovereign identity (SSI) approach, which effectively addressed the existing limitations while pursuing the same objectives as the user-centric approach. Blockchain's transparent and consistent nature helped resolve the data reliability issues associated with the earlier Open ID. Notable examples of blockchain-based SSI models include Veres One distributed ledger and ID union network [1]. Standardization efforts for the SSI approach are underway, with discussions on decentralized identifiers (DID) taking place within the World Wide Web Consortium (W3C).

Nonetheless, numerous challenges remain to facilitate the widespread adoption of the SSI approach. Each SSI approach employs its unique authentication and authorization process, necessitating users to learn a new set of procedures for each model.

Additionally, service developers face the burden of separately implementing this process for each SSI approach to integrate their services with them. While SSI approach have attempted to mitigate these issues through tutorial pages to aid user understanding and by offering development libraries for easier integration, these measures do not fundamentally resolve the aforementioned challenges [2].

Proposes SSI model with oAuth 2.0 but with blockchain, blockchain has got many disadvantages such as scalability and transaction speed issue. Hence, this research paper introduces an innovative open network-based self-sovereign identity (SSI) approach that effectively addresses the challenges outlined. The proposed model adheres to the core principles of the SSI model while also aligning with the well- established OAuth 2.0 framework, known for its maturity and widespread adoption. This integration with OAuth 2.0 simplifies both the development process and user experience, as users are already familiar with OAuth.

In the proposed model, the principles of user-centric authentication and authorization are upheld through a design that empowers each user to assume the role of an authorization server within OAuth, utilizing their own device. Users can securely man- age their information while offering a decentralized authentication and authorization process that is not reliant on a specific service provider, such as Github.

The proposed model has the following contributions. First, it proposes SSI approach that complies with OAuth 2.0 standard, which results in high reliability and interoperability. Second, it provides novel user-centric authentication and authorization which are controlled under a user's own device with the help of open network. Third, from the viewpoint of service developers, the proposed model can be easily applied to their service because it follows the flow of OAuth 2.0. Fourth, it enables a user to manage personal information in a both secure and high accessible way by storing the information in his own private device after encryption. The rest of this paper presents the following. Section 2 shows how OAuth 2.0 works and it examines existing studies related to SSI. Section 3 describes the structure and processes of the proposed model. Section 4 displays the results of implementing the proposed model. Section 5 provides the results of a security analysis and Section 6 contains conclusions.

2. OAuth

2.1 In brief

The OAuth 2.0 authorization framework facilitates a third-party application in acquiring restricted access to an HTTP service in one of two ways. It can do so by coordinating an approval process between the resource owner and the HTTP service, effectively acting on behalf of the resource owner. Alternatively, the framework allows the third-party application to independently obtain access in its own right.

2.2 What is solves

In the traditional client-server authentication model, when a client seeks access to a secured resource (known as a protected resource) on the server, it does so by verifying its identity with the server, utilizing the resource owner's credentials. To enable

Third-party applications to access these protected resources, the resource owner must disclose their credentials to these third parties. However, this approach presents several problems and limitations:

1. Credential Storage: Third-party applications are obligated to retain the resource owner's credentials for future use, typically storing the password in plaintext.
2. Password Dependency: Servers must support password-based authentication, even though passwords have inherent security weaknesses.
3. Broad Access: Third-party applications often acquire overly extensive access to the resource owner's protected resources, leaving resource owners without the ability to limit access duration or restrict access to specific subsets of resources.
4. Revocation Challenges: Resource owners are unable to revoke access for a specific third party without revoking access for all third parties, and they can only do so by changing the third party's password.
5. Compromised third party application: When a third-party application is compromised, it poses a significant security risk, as such a compromise can lead to the exposure of the end-user's password. Consequently, this breach could potentially grant unauthorized access to all data that is protected by that password. This underscores the importance of robust security measures and safeguards to prevent such compromises and protect user data.

These issues underline the need for more secure and flexible authentication and authorization mechanisms, which OAuth 2.0 aims to address.

OAuth effectively tackles these issues by introducing an authorization layer and separating the roles of the client and the resource owner. In OAuth, the client requests access to resources controlled by the resource owner and hosted by the resource server. Instead of utilizing the resource owner's credentials to access these protected resources, the client obtains an access token—a string that represents a specific scope, lifetime, and other access attributes. These access tokens are issued to third-party clients by an authorization server with the resource owner's approval. The client then employs this access token to access the protected resources hosted by the resource server [3] has elaborated the same as part of RFC6749.

For instance, consider an end-user (resource owner) who wishes to grant a printing service (client) access to her protected photos stored on a photo-sharing service (resource server). Rather than sharing her username and password with the printing service, the end-user authenticates directly with a server trusted by the photo-sharing service (authorization server). This trusted server then issues delegation-specific credentials (access token) to the printing service, allowing it to access the specified resources without the need for the resource owner's sensitive login information.

A typical proposed flow for oAuth2.0 is as follows:

1. User (Resource Owner) connects with client and provides information on Authorization server it wishes to use to provide the required information (Protected resource)
2. Client requests for Authorization (Auth Z) to user
3. User provides information

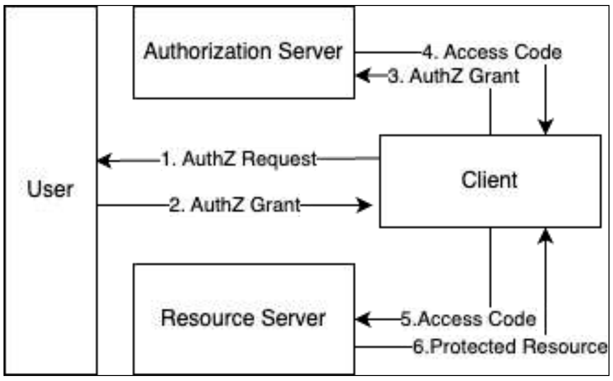

Fig 1: oAuth2.0 in Brief

1. Based on user provided information, Authorization Server checks if user is legitimate one and provides an 'Access code' that can be used by client for accessing the protected resource
2. Client uses 'Access Code' and communicates with resource server. Resource server verifies the 'Access Code' and provides protected resource to client.

3. Self-Sovereign Identity

3.1 In brief

Self-sovereign identity (SSI) offers the potential to establish a level of trust and autonomy in sharing or disseminating identity attributes in the digital realm that mirrors the control individuals have in the physical world. SSI adopts a user-centric approach, wherein the user maintains ownership of their data and isn't reliant on a central authority to verify their identity [4] has discussed the same in an elaborated way.

In an SSI framework, users exercise complete authority over the information they choose to disclose and to whom they reveal it. Through a shared identity meta system, users can verify their digital identity across diverse platforms and in various locations. Consequently, self-sovereign identity is characterized by privacy, security, and portability, delivering greater control and assurance in the digital realm.

SSI implementations depart from hierarchical certification schemas and instead operate on the basis of a peer-to-peer and distributed "web of trust," devoid of root or intermediate Certificate Authorities (CAs).

While there is an ongoing effort to establish the adoption of existing vendor- independent Self-Sovereign Identity (SSI) standards in the enterprise world and on the public internet, the integration of SSI into enterprise environments and landscapes is still a work in progress and lacks standardized practices.

In enterprise settings, where some services and applications are contained within an enterprises's internal infrastructure, delegated authentication is frequently implemented using company-owned identity providers, such as Active Directory or Red Hat Key cloak. These company-owned Single Sign-On (SSO) solutions often extend to

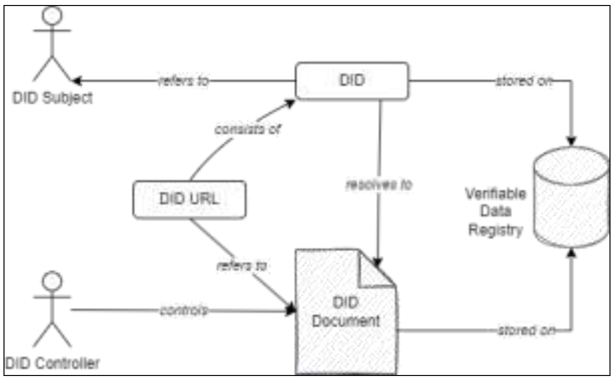

Fig 2: SSI in Brief

both web applications and" traditional" rich client applications. They utilize various protocols, including SAML, Kerberos, O Auth, and OIDC, to facilitate secure and streamlined authentication processes. However, achieving seamless integration of SSI into such enterprise environments remains a complex and evolving endeavor.

SSI systems offer enhanced privacy features, vulnerabilities such as identity spoofing and data breaches remain concerns. Exploring cryptographic techniques, zero-knowledge proofs, and robust authentication mechanisms can contribute to mitigating these risks?].

3.2 What it solves

3.2.1 DID Subject

DID Subject can be an individual or an organization or concept or a thing.

3.2.2 DID Controller

DID Controller can be a human or non-human.

3.2.3 DID

DID consists of scheme, DID-Method and DID-Method specific identifier. It refers to a DID subject. It resolves to a DID Document. It is stored on verifiable data registry.

3.2.4 DID URL

DID URL consists of DID and additional path to resource. Resource could be public key mentioned in DID document or it could also be external resource.

3.2.5 DID Document

DID Document contains information pertaining to cryptography proofs that can be used by controller to prove its control and additional information pertaining to DID methods and services relevant for interaction with DID subject.

3.2.6 Verifiable Data Registry

It's a system that facilitates storage of DIDs and DID documents and retrieval of same.

3.2.7 DID Methods

Mechanism to create, resolve, update and deactivate DID and respective DID document

3.2.8 DID resolver and DID resolution

It accepts DID as input and produces a conforming DID document as output. This process is called DID resolution. The steps are defined by DID method specification for resolving a specific type of DID.

Naik and Jenkins [5] provides much needed insight into possibility of SSI implementation using distributed ledger technology. But In [6], use of blockchain was avoided to avoid weakness of blockchain such as scalability and transaction speed. In our approach we complete remove dependency on blockchain and make use of OAuth and SSI to achieve user centric identity management system approach - Digital Identity Secure System.

4. Digital Identity Secure System

Despite the development and proposal of Self-Sovereign Identity (SSI) models leveraging blockchain technology, widespread adoption has remained elusive. The core issue lies in the fact that each of these approach adopts its distinct authentication and authorization process. This diversity in processes presents technical challenges when existing systems attempt to integrate the SSI model. Users are confronted with unfamiliarity and inconvenience when navigating these novel processes. Additionally, the security of these unique authentication and authorization methods has not undergone thorough analysis.

In response to these challenges, this research paper introduces a novel SSI approach, which aims to address these fundamental issues. The presented approach aligns with the widely adopted OAuth framework, ensuring users have a familiar and comfortable experience through innovative authentication and authorization procedures based on OAuth principles. Leveraging open source and network technology, this model delivers decentralization and data integrity for both users and clients, thus ensuring the reliability of authentication and authorization processes.

This approach not only resolves the issues of information centralization and privacy concerns associated with existing federated identity management models controlled by major corporations but also empowers users with secure access to and control over their personal information, promoting data sovereignty.

Figure 3 provides overview of proposed approach.

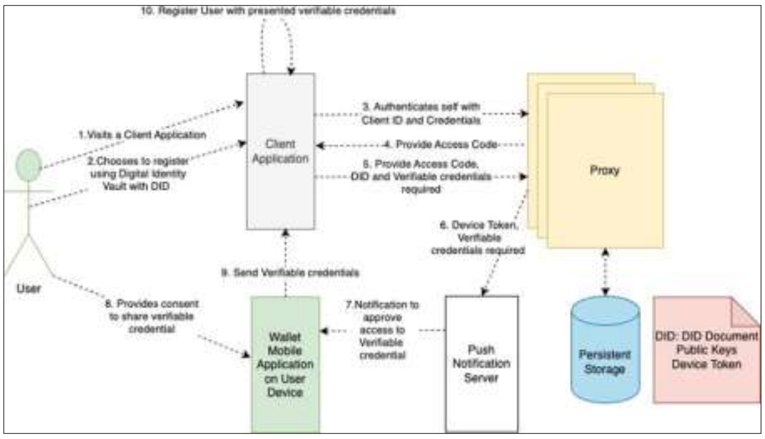
10. Register User with presented verifiable credentials
1. Visits a Client Application
2. Chooses to register using Digital Identity Vault with DID
Client Application
3. Authenticates self with Client ID and Credentials
4. Provide Access Code
5. Provide Access Code, DID and Verifiable credentials required
Proxy
6. Device Token, Verifiable credentials required
User
9. Send Verifiable credentials
8. Provides consent to share verifiable credential
7.Notification to approve access to Verifiable credential
Wallet Mobile Application on User Device
Push Notification Server
Persistent Storage
DID: DID Document Public Keys Device Token

Fig 3: User Registration

User an individual who utilizes a service offered by the client is referred to as a user. This user accesses the client via a web browser or mobile application and maintains control over the authorization process using their own device.

Client The entity that offers a service to the user is the client. The client expresses its willingness to be granted permission to access the user's information by utilizing Digital Identity Secure.

Wallet Mobile Application on User Device a device, such as a mobile phone, where an application responsible for managing the user's identity is installed is referred to as the user device. In the context of OAuth, this user device takes on the dual roles of both an authorization server and a resource server. It represents a departure from traditional centralized identity management models, empowering users to take control of their own identity and personal data. This user device is responsible for validating authorization requests and securely providing encrypted user information to the client.

Proxy a component responsible for routing requests to the appropriate user device is referred to as the proxy. The proxy handles the response generated by the user device and subsequently forwards it to the client. In this setup, each client shares a client secret with the proxy, and as a result, the proxy assumes the responsibility of validating the client.

Push Notification Server The proxy employs Push Notification Server (PNS) as a messaging system for mobile devices to deliver requests to the devices. To ensure

The accurate delivery of a push message, a device token is required. This token can be obtained through the notification contract, facilitating the proper routing of messages to the intended devices.

4.1 Open APIs

In Digital Identity Secure, Open APIs are being used for communication across open network. This shall enable ease of integration for applications across enterprises. This is most preferable way of integration.

4.1.1 Registration/Subscription

Client Application Subscription

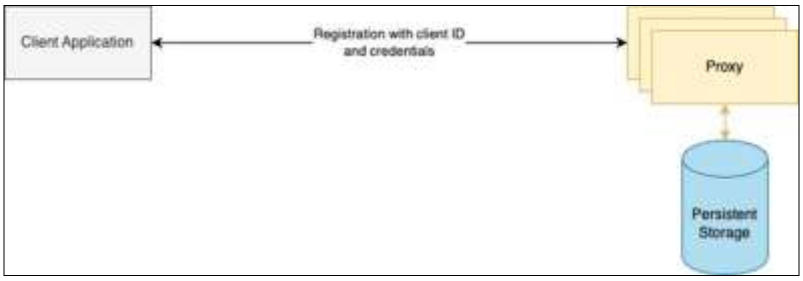

Fig 4: Client App Registration

1. Base of client application identity that is considered is its domain name. Sub- scribe rid becomes domain name.
2. Client application should also have a valid TLS certificate
3. Host an api for checking encryption key. It can be named as /on Subscribe. https://¡Client App Domain¿/¡Client App Call Back URL¿/on Subscribe
4. Generate Signing Key Pair – signing Public Key and signing Private Key
5. Generate Encryption Key Pair – encryption Public Key and encryption Private Key
6. Generate Unique Request ID (request id) in a UUID format
7. Generate SIGNED-UNIQUE-REQ-ID =¿ (Sign request ID using signing Private Key generated.
8. Create digitalID-site-verification.html and place it at subscriber ID by adding SIGNED-UNIQUE-REQ-ID generated. Proxy shall check existence of digital ID-site-verification.html at
9. https://¡subscriber Id¿/digitalID-site-verification.html
10. Configure developed /on Subscribe implementation to use encryption Private Key and Proxy public key to decrypt the challenge String.

- Subscriber Id= YOUR SUBSCRIBER ID
- Callback Url= Relative path to o Subscribe implementation
- Signing Public Key= ¡value of sign Public Key generated ¿
- Encryption Public Key= ¡value of enc Public Key generated¿
- proxy public key =”MCowBQYDK2VuAyEAvVEyZY91.”
- Unique Key Id= ¡generate a unique number for tracking key pairs¿
- Other fields include name, address, and contact details.

11. Call api/subscribe with all the details as mentioned above. Proxy shall return cre- dentials and client ID encrypted with public key of client application as a HTTP response to successful registration/subscription.

4.1.2 User Subscription

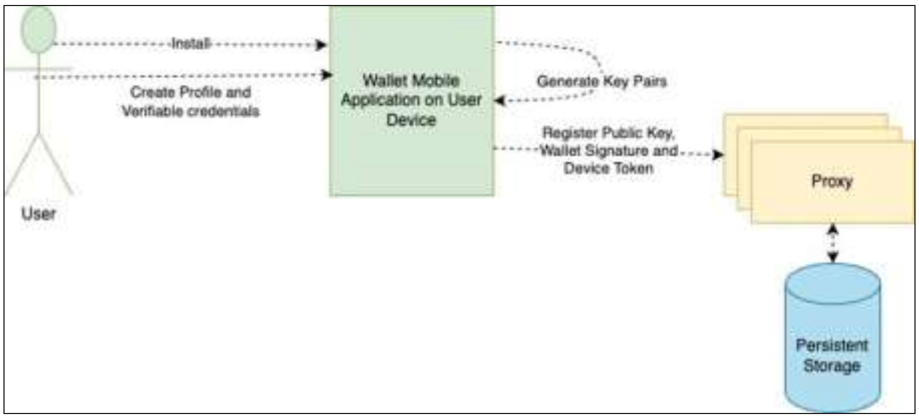
Install
Create Profile and Verifiable credentials
User
Wallet Mobile Application on User Device
Generate Key Pairs
Register Public Key, Wallet Signature and Device Token
Proxy
Persistent Storage

Fig 5: Wallet registration

1. User would need to download and install the Digital Identity Secure mobile app from its application store. For android it would be ”Play Store” and for iOS, it would be ”App Store”
2. User will have to access register option in Digital Identity Secure mobile app
3. User needs to select enable notification to receive request for verifiable credentials that are stored in Digital Identity Secure of mobile app. This generates device token. Device token and mobile app gets registered with push notification server of respective mobile operating system. For iOS, its Apple Push Notification service (APNs); while for Android, its Firebase Cloud Messaging (FCM). This helps to establish a secure channel between proxy and user device via push notification server.
4. Digital Identity Secure generates two key pairs - signing public key, signing private key and encryption public, encryption private key.
5. Digital Identity Secure sends details to proxy with Decentralized Identifier (DID), device token, encryption public key, signing public key for registering user device for further communication.
6. This completes user subscription.

4.1.3 User Verifiable Credentials

1. User opens up Digital Identity Secure mobile app enter values for fields like - name, address, contact number, contact email, date of birth, etc.
2. User can choose to create single or multiple verifiable credentials that have one or more of the fields mentioned above.
3. These credentials and fields are encrypted using encryption private key of the user and are decrypted only when user wants to view them

4.1.4 Client Application User Registration/Signup

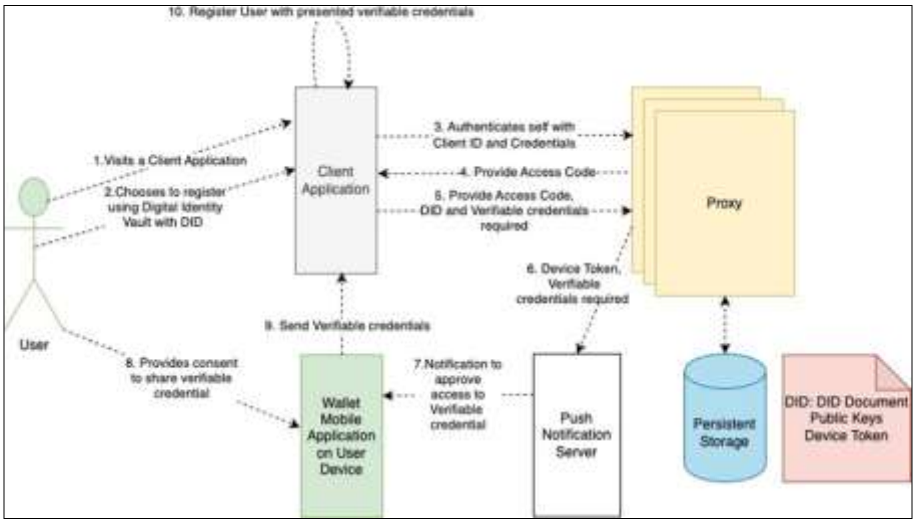
10. Register User with presented verifiable credentials
1.Visits a Client Application
2.Chooses to register using Digital Identity Vault with DID
Client Application
3. Authenticates self with Client ID and Credentials
4. Provide Access Code
5. Provide Access Code, DID and Verifiable credentials required
Proxy
6. Device Token, Verifiable credentials required
User
9. Send Verifiable credentials
8. Provides consent to share verifiable credential
Wallet Mobile Application on User Device
7.Notification to approve access to Verifiable credential
Push Notification Server
Persistent Storage
DID: DID Document Public Keys Device Token

Fig 6: Client Application User Registration/Signup

1. User visits a Client Application.
2. User chooses to register using Digital Identity Secure with DID. User provides it’s DID that he intends to use for registration.
3. Server side Client Application shall encrypt client ID and credentials issued by proxy with public key of proxy and sent it to proxy over TLS HTTP for ”Client Credentials Grant” mode of oAuth 2.0 protocol as per RFC 6749. The client application authenticates with the proxy server acting as Authorization server and requests an access token from the token endpoint. With grant type=client credentials and scope=as required by client application.
4. Using decryption key the proxy server (Authorization Server), it decrypts the request body and if the request for an access token is deemed valid and has received authorization, the proxy server (authorization server) proceeds to generate and issue an access token.
5. Client application provide the access token, DID of user, Scope/Verifiable credentials required, callback URL for getting verifiable credentials/scope from user device.
6. Proxy checks for entry of DID by lookup and gets the DID Document that contains details of DID, Device Token and verifiable credentials/scope required, callback URL for getting verifiable credentials/scope from user device are taken from the request of client application. Public key and OS of user device from DID document is used to encrypt the request body details other than device token and sent it to push notification server. For iOS, its Apple Push Notification service (APNs); while for Android, its Firebase Cloud Messaging (FCM).
7. Push notification server sending the notification to user device and mobile application.
8. Mobile application decrypts the message received using decryption private key. User gets the prompt to approve sharing of scope/verifiable credentials. User provides approval/consent to share the same.
9. Mobile application sends the verifiable credentials/scope to the URL received by encrypting using public key of the client application.
10. Client application server decrypts the message using decryption private key and checks the response and completes signup request of the user using verifiable credentials/scope received from user device.

4.1.5 Client Application User Login

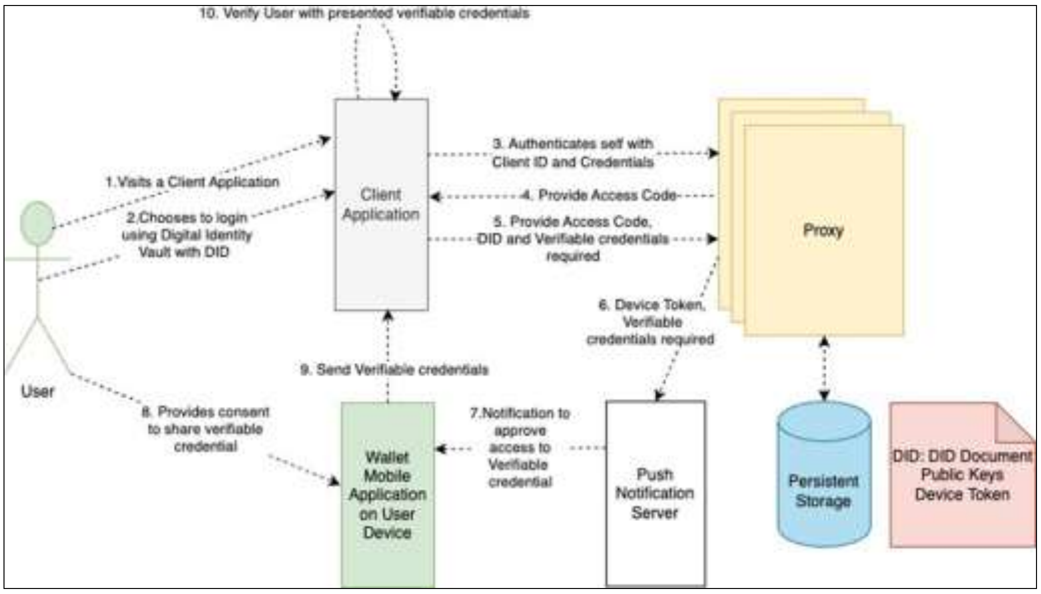
10. Verify User with presented verifiable credentials
1.Visits a Client Application
2.Chooses to login using Digital Identity Vault with DID
Client Application
3. Authenticates self with Client ID and Credentials
4. Provide Access Code
5. Provide Access Code, DID and Verifiable credentials required
Proxy
6. Device Token, Verifiable credentials required
User
9. Send Verifiable credentials
8. Provides consent to share verifiable credential
7.Notification to approve access to Verifiable credential
Wallet Mobile Application on User Device
Push Notification Server
Persistent Storage
DID: DID Document Public Keys Device Token

Fig 7: Client Application User Login

1. User visits a Client Application.
2. User chooses to login using Digital Identity Secure with DID. User provides it's DID that he intends to use for login.
3. Server side Client Application shall encrypt client ID and credentials issued by proxy with public key of proxy and sent it to proxy over TLS HTTP for "Client Credentials Grant" mode of oAuth 2.0 protocol as per RFC 6749. The client application authenticates with the proxy server acting as Authorization server and requests an access token from the token endpoint. With grant type=client credentials and scope=as required by client application.
4. Using decryption key the proxy server (Authorization Server), it decrypts the request body and if the request for an access token is deemed valid and has received authorization, the proxy server (authorization server) proceeds to generate and issue an access token.
5. Client application provide the access token, DID of user, Scope/Verifiable credentials required, callback URL for getting verifiable credentials/scope from user device.
6. Proxy checks for entry of DID by lookup and gets the DID Document that contains details of DID, Device Token and verifiable credentials/scope required, callback URL for getting verifiable credentials/scope from user device are taken from the request
7. Of client application. Public key and OS of user device from DID document is used to encrypt the request body details other than device token and sent it to push notification server. For iOS, its Apple Push Notification service (APNs); while for Android, its Firebase Cloud Messaging (FCM).
8. Push notification server sending the notification to user device and mobile application.
9. Mobile application decrypts the message received using decryption private key. User gets the prompt to approve sharing of scope/verifiable credentials. User provides approval/consent to share the same.
10. Mobile application sends the verifiable credentials/scope to the URL received by encrypting using public key of the client application.
11. Client application server decrypts the message using decryption private key and checks the response and completes login request of the user using verifiable credentials/scope received from user device.

5. Security Analysis
This section presents the security analysis of the proposed Digital Identity Secure system. The proposed system adheres to the widely adopted OAuth 2.0 framework, which has undergone extensive security scrutiny, leading to confirmed security robustness. Consequently, the security assessment of Digital Identity Secure system was carried out with the assumption that OAuth 2.0 is a secure foundation.
The security analysis of Digital Identity Secure system primarily focuses on its resilience against two major threats: identity theft and the unauthorized disclosure of personal information. These aspects will be elaborated on in the following sections to provide an understanding of Digital Identity Secure system's security measures and capabilities in safeguarding against these critical security concerns.

5.1 Mitigating Identity Theft
The theft of a user's identity by a malicious attacker poses a significant and severe threat. In this context, we will examine three scenarios in which a hacker endeavors to steal an individual's identity, along with the corresponding protective measures implemented by Digital Identity Secure system.
In the first attack scenario, a malicious attacker adopts the guise of the target user when approaching the client. If the attacker manages to substitute the Firebase Cloud Messaging (FCM) token of the target user within the notification contract with their own token, it would redirect the authentication request outlined in above section to the attacker's device instead of the intended user's device, thereby achieving a successful attack.
However, it's crucial to note that even after forging the FCM/APN/Device token, since other request body parameters would be encrypted using public key of user and shall be only be decrypted using private key of user, and since the private key never ever leaves wallet mobile app of the user device, the malicious attacker will never be able to make use of the message and nor it wold know the callback URL to be called for completing the registration or login

request on client application. Therefore, it's
Of no use to attacker to replace the target user's token without gaining access to the user's corresponding private key.
The second attack scenario unfolds when the target user initiates an authentication request. In this scenario, the attacker impersonates the target user and simultaneously initiates their own authentication request. Consequently, both the legitimate user and the attacker receive two authentication request messages on the target user's device. If the timing is in favor of the attacker, their request may arrive first, and the user might mistakenly approve it, assuming it to be the genuine request they themselves initiated.
To thwart this type of attack, Digital Identity Secure system implements a protective measure by generating and utilizing a unique random secret code for each authentication request. Regular users can validate their own authentication request by verifying whether the secret code presented in their browser matches the secret code displayed on their device. This countermeasure ensures that users can confidently discern and authenticate their legitimate requests, enhancing security against this form of attack.
In the third attack scenario, the attacker authenticates themselves as their own identity but subsequently supplies the client with the phone number of the target user along with fabricated information. This deceptive maneuver allows the attacker to gain access to the client's services, effectively impersonating the target user and potentially engaging in malicious activities that could harm the genuine user.
To thwart this potential threat, Digital Identity Secure System introduces a safe- guard by enabling the client's verification process. This involves the proxy including the phone number of the user from whom the client requests authorization when delivering user information to the client. This verification step ensures that the client can cross-verify the phone number and user details, making it more challenging for an attacker to impersonate the target user and carry out harmful actions undetected.

5.2 Mitigating Personal Identifiable Information leakage

Digital Identity Secure System employs a decentralized architecture in which each user device assumes the role of an authorization server, eliminating the reliance on a centralized authorization server. While the proxy serves as the intermediary connecting clients and user devices, it's important to acknowledge that the proxy represents a single point of failure and, as such, may attract the interest of potential attackers.
However, it's worth noting that the threat associated with this scenario mirrors the threats that existing authorization servers supporting OAuth also face. To mitigate this vulnerability, the proxy's implementation is rooted in cloud computing, ensuring fault tolerance and high availability. This approach helps safeguard against service interruptions or attacks targeting the proxy.
Furthermore, the potential damage resulting from attacks on the proxy is considerably lower than the impact of attacks on conventional authorization servers. This is primarily because Digital Identity Secure System does not store users' personal or critical information within the proxy, minimizing the potential exposure of sensitive data in the event of an attack.
In the context of Digital Identity Secure System, the proxy's role doesn't involves receiving user information from the user device and passing it on to the client. Hence a potential risk where a malicious proxy could collect and misuse data from multiple users doesn't exists.
All the information being exchanged between client, proxy and user device is encrypted to mitigate the risk and enhance security.
Before transferring user information, Digital Identity Secure System encrypts the data within the user device using the client's public key. This encryption ensures that even if the malicious attacker were to intercept the data during transmission, it would be unable to access or decrypt the user's information without the client's private key.
Digital Identity Secure System employs a robust strategy to counter the threat of attackers attempting to acquire user information by eavesdropping on messages exchanged between Digital Identity Secure System's components through the network. The security measures include the following:

1. **HTTPS Protocol**: Digital Identity Secure System employs the HTTPS protocol for communication among the user, client, and proxy components. This encryption protocol safeguards the confidentiality and integrity of data during transmission. It ensures that eavesdroppers cannot intercept or tamper with the messages exchanged among these components.
2. **PNS Server**: When the proxy requests a push message from the Firebase Cloud Messaging (FCM) or Apple Notification Server, it also utilizes the HTTPS protocol for secure communication. Additionally, the PNS server delivers messages to the user device using the XMPP over TLS protocol, enhancing security and thwarting eavesdropping attempts.
3. **Data Encryption**: As previously mentioned, user information is transmitted in an encrypted format. This encryption ensures that even if an attacker were to intercept the data, it would be indecipherable without the appropriate decryption keys.

The combination of these security measures collectively creates a robust defense against data leakage and eavesdropping attacks, enhancing the overall security of Digital Identity's communications and data transfers.

6. Conclusions

This research introduces a novel open api network based Self-Sovereign Identity (SSI) approach that aligns seamlessly with OAuth 2.0. Users gain enhanced access to their information, free from the constraints of specific service providers. Users can actively engage in the authentication and authorization processes using their own devices. Importantly, the proposed model offers a user experience reminiscent of the existing OAuth procedure, facilitating straightforward adoption by clients already familiar with OAuth. Consequently, the model exhibits exceptional scalability.
The study demonstrates the security analysis underscores its resilience against identity theft and data leakage, bolstering its trustworthiness.
Moreover, the proposed approach provides a remedy to the issue of centralized control over user information by major IT companies. It empowers users by securing
Their data sovereignty, ensuring they have the right to

utilize and manage their own information. Future research endeavors will explore the potential for a novel user- centric web, built upon this proposed approach, where users can recover from lost devices, history of approved consents for sharing verifiable credentials. This represents a promising avenue for the evolution of online user interactions and data management. In [7], it has been indicated that oAuth token do not have any well-defined privacy properties as user signature is not used in oAuth, whereas in our proposed approach since the scope/verifiable credential is shared by user device using its private key,
Hence privacy is maintained.
[8] Proposes to make use of verifiable credentials for identity management, instead of token. We have explored similar concept of sharing the verifiable credential instead of token to client application with higher degree of user control on attributes that can be part of verifiable credentials.
In our next work, we shall explore other perspective of key management - key rotations by client application and user device and verifiable presentations with zero knowledge proofs.

References

1. Kuperberg M, Klemens R. Integration of Self-Sovereign Identity into Conventional Software Using Established IAM Protocols: A Survey; c2020. https://dl.gi.de/server/api/core/bitstreams/aab190f5-45f9-4ec1-88dd-7c9727b169e8/content
2. Hong S, Kim H. Vaultpoint: A blockchain-based SSI model that complies with oauth 2.0. Electronics, 2020, 9(8). https://doi.org/10.3390/electronics9081231
3. Hardt D. RFC 6749: The OAuth 2.0 Authorization Framework; c2012. https: //datatracker.ietf.org/doc/html/rfc6749
4. Windley PJ. Sovrin. An identity metasystem for self-sovereign identity. Frontiers in blockchain, 2021, 4. https://doi.org/10.3389/fbloc.2021.626726
5. Naik N, Jenkins P. Sovrin network for decentralized digital identity: Analyzing a self-sovereign identity system based on distributed ledger technology. In: 2021 IEEE International Symposium on Systems Engineering (ISSE); c2021. p. 1-7. https://doi.org/10.1109/ISSE51541.2021.9582551
6. Lemmon A. Re-shaping the future of identity through user-owned verifiable credentials. In: 2021 IEEE International Conference on Blockchain and Cryp-tocurrency (ICBC); c2021. p. 1-2. https://doi.org/10.1109/ICBC51069.2021. 9461078
7. Halpin H. Nym credentials: Privacy-preserving decentralized identity with blockchains. In: 2020 Crypto Valley Conference on Blockchain Technology (CVCBT); c2020. p. 56-67. https://doi.org/10.1109/CVCBT50464.2020.00010
8. Chadwick DW, Laborde R, Oglaza A, Venant R, Wazan S, Nijjar M, *et al*. 17 Improved identity management with verifiable credentials and fido. IEEE Communications Standards Magazine. 2019;3(4):14-20. https://doi.org/10.1109/ MCOMSTD.001.1900020