ChatGPT

Projets open source combinant DID et UCAN pour l'authentification

Contexte et objectif de la recherche

Les *Decentralized Identifiers* (DID) et les *User Controlled Authorization Networks* (UCAN) sont des technologies émergentes qui permettent aux utilisateurs de posséder et contrôler leurs identités numériques via des paires de clés (souvent Ed25519) sans dépendre d'une autorité centrale. L'idée est de fournir une authentification décentralisée (sans mot de passe, basée sur des signatures) pouvant néanmoins s'intégrer avec les systèmes web existants – typiquement via des passerelles ou adaptateurs compatibles OAuth2/OpenID Connect.

L'objectif ici est d'identifier des projets open source (même expérimentaux ou en phase alpha) qui :

- Utilisent **les DID et UCAN** pour l'authentification (identification par clés publiques Ed25519 et délégation d'autorisations via UCAN).
- Proposent soit *directement*, soit via des adaptateurs, une **compatibilité OAuth2/OIDC**, afin de permettre par exemple à des applications web classiques (Discourse, etc.) d'accepter une identité décentralisée comme fournisseur d'identités.
- Disposent d'un dépôt public (GitHub), d'une licence open source et idéalement de documentation technique.

Ci-dessous, nous listons plusieurs projets répondant à ces critères, en décrivant leurs fonctionnalités, leur état d'avancement et les technologies utilisées. Nous mettons en évidence ceux qui pourraient être utilisés ou adaptés pour brancher une connexion DID/UCAN sur un site web prenant en charge OAuth2 (par exemple via un fournisseur OIDC custom).

Projets open source identifiés

- **Fission ODD SDK (ex-Webnative)** – *Framework* JavaScript/TypeScript pour applications web local-first. Fission a conçu un modèle d'identité distribué où chaque utilisateur génère localement une paire de clés Ed25519 formant un DID (généralement de type `did:key`). L'authentification se fait en signant des *tokens* UCAN (extension de JWT) contenant les permissions de l'utilisateur. Aucune infrastructure centralisée n'est requise : le *token* signé par la clé privée de l'utilisateur fait foi et peut être vérifié par les services via la clé publique (DID). Fission décrit cela comme un modèle "identité distribuée + UCAN" (identité par DID + capacités décentralisées) [1]. Ce SDK permet donc de gérer des comptes sans mot de passe, l'utilisateur créant son propre ID et prouvant son contrôle cryptographiquement [2]. En l'état, Fission ODD n'interface pas nativement avec OAuth2, mais il fournit les briques (DID utilisateur, UCAN porteur de permissions) pour construire une passerelle OAuth2 si nécessaire. Le projet est **actif** (développé par Fission depuis ~2019, renommé ODD en 2023) et majoritairement écrit en **TypeScript** (avec des composants Rust/Wasmtime côté file system). *Usage potentiel*: Nécessite de développer un adaptateur OIDC custom si on souhaite l'utiliser comme fournisseur d'identité sur, par exemple, Discourse (ce n'est pas fourni prêt-à-l'emploi).

- **SpruceID SIWE OIDC Provider** – *Serveur OpenID Connect pour « Sign-In with Ethereum »* développé par SpruceID (🏷 **spruceid/siwe-oidc** [3]). Ce projet permet à un utilisateur de s'authentifier avec son portefeuille Ethereum (signature EIP-4361) et expose cette authentication via **OpenID Connect** (basé sur OAuth2) aux applications web. Techniquement, l'utilisateur signe un challenge avec sa clé Ethereum (ECDSA secp256k1), ce qui est traité comme une identité décentralisée (DID de type `did:pkh:eth`). Le serveur SIWE-OIDC vérifie la signature et émet un jeton **OIDC** standard (JWT contenant l'adresse Ethereum en subject, etc.) que des sites comme Discourse peuvent consommer. Spruce a notamment publié un plugin Discourse permettant d'activer ce login Ethereum [3]. Le serveur OIDC SIWE est open source (double licence Apache/MIT) en **Rust** (framework Axum) avec une interface web (WalletConnect en front). Il est **actif** (plusieurs contributeurs, ~2022-2023) et a été utilisé pour une instance OIDC communautaire gérée par ENS DAO [4]. *Usage potentiel*: **immédiat** – ce projet est spécifiquement conçu pour fournir un IdP OIDC décentralisé. En configurant Discourse (ou tout client OIDC) avec ce fournisseur, on peut autoriser la connexion par identité Ethereum (ou potentiellement autre blockchain prise en charge). Notons que ce projet n'emploie pas UCAN en interne, mais utilise des DIDs (did:pkh) et la notion de capacités via le standard *CACAO* (extension d'autorisation similaire aux UCAN).

- **SpruceID Kepler** – *Stockage décentralisé auto-hébergé par l'utilisateur avec DIDs et capacités*. Kepler (🏷 **spruceid/kepler**) est un projet open source en **Rust** fournissant un réseau de stockage où les données sont organisées en *Orbits*. Chaque Orbit est contrôlé par un DID, et l'accès se gère via des **Authorization Capabilities** (c'est-à-dire des jetons de capacité) délégués aux autres DIDs [5]. En pratique, un utilisateur (personne, application ou DAO) possède un DID (clés Ed25519 ou autre méthode) qui sert d'identité pour administrer son espace de stockage. Kepler utilise les UCAN (ou standards proches) pour signer et vérifier les permissions d'accès aux données. Le projet est **actif** (développement soutenu par Spruce, ~200 commits, audit de sécurité effectué) et open source (Apache 2.0). C'est plutôt une brique de stockage et de partage de fichiers/credentials **(tech : Rust côté serveur, SDK TypeScript côté client)**. *Usage potentiel*: Kepler n'est pas une solution d'authentification web généraliste, mais montre une implémentation concrète de DID+UCAN pour contrôler l'accès aux ressources. On pourrait imaginer *détourner* son mécanisme d'émission de capacités pour construire un jeton d'identification, mais à date Kepler vise surtout l'accès API/données.

- **Web3.Storage / w3up (Protocol Labs)** – *API d'upload IPFS avec authentification décentralisée*. Web3.Storage (et son successeur w3up/Stocha**r**a) est un service d'hébergement de données IPFS géré par Protocol Labs. Ils ont adopté les **UCAN** pour leur nouveau système d'auth (annonce Bêta fin 2022). Le fonctionnement : chaque utilisateur obtient un DID (généré via une clé Ed25519) et crée un *UCAN token* pour toute requête API. Le serveur n'a plus besoin de stocker d'utilisateurs : il vérifie simplement que le UCAN présenté est signé par la clé privée correspondant au DID et inclut la capacité requise (upload, etc.) [6] [2]. Cela supprime la nécessité d'un login/password classique – « l'utilisateur crée son propre ID et le partage au service » [2]. Ce projet inclut des **bibliothèques JS/TS** (ex: `ucan-storage` côté client) et un service cloud (dont certaines parties sont fermées). L'utilisation d'UCAN comme *bearer token* le rapproche d'OAuth2 dans l'esprit (le UCAN est passé en en-tête HTTP `Authorization: Bearer`), mais ce n'est pas directement un serveur OAuth2 standard. **Actif**, il sert de base aux services NFT.Storage et Web3.Storage de production. *Usage potentiel*: En l'état, Web3.Storage est une solution verticale (auth + stockage), pas un fournisseur d'identité générique. Toutefois, il démontre qu'un **pont UCAN ↔ OAuth2** est concevable : par exemple, un site web pourrait accepter un UCAN valide en lieu et place d'un JWT OAuth2 en vérifiant sa signature et ses claims de DID, ce qui est proche de ce qu'ils font en interne [6]. On pourrait imaginer fournir un point de terminaison d'échange où un UCAN valide émis par l'utilisateur

donne lieu à un code ou jeton OAuth2 pour le site – à notre connaissance, ceci n'est pas encore proposé clef en main par PL, il faudrait le développer.

- **Keycloak SSI Provider (Dizme)** – *Extension de Keycloak pour authentification via identités décentralisées*. Ce projet (🏷 **dizme/keycloak-ssi-provider**) intègre la vérification de **credentials auto-souverains** dans Keycloak, un fournisseur IDM open source populaire. Concrètement, il ajoute un *Identity Provider* custom dans Keycloak capable de demander à l'utilisateur de présenter un **W3C Verifiable Credential** (par exemple un JWT signé par un DID issu d'un portefeuille EBSI/EUDI) pour se connecter [7] [8]. Un module *verifier* vérifie le credential (via l'API EUDI par ex.), puis Keycloak crée une session OIDC normale. L'utilisateur final peut donc se connecter sur des apps (comme Discourse) connectées à Keycloak, **en utilisant son DID/credential** plutôt qu'un mot de passe. Le projet est en **Java/TypeScript**, lié à Dizme/InfoCert, et semble **actif (2023)** avec des images Docker fournies. *Usage potentiel*: **élevé** – Cette approche fournit un pont concret entre le monde SSI (DID + attestations) et OAuth2/OIDC standard via Keycloak. Bien qu'il utilise des *Verifiable Credentials* (SD-JWT) plus que des UCAN, le concept est similaire (preuve signée par clé privée utilisateur). En adaptant ce plugin ou en l'étendant pour accepter des UCAN comme preuve d'identité, on pourrait intégrer un login 100% décentralisé sur n'importe quelle appli supportant OIDC (Discourse, Nextcloud, etc.).

- **Ceramic DID Session / CACAO** – *Bibliothèque d'authentification distribuée avec délégation temporelle*. Ceramic, un réseau de gestion de données décentralisées, propose **DID-Sessions** (🏷 *@ceramicnetwork/did-session* en JavaScript) qui permet aux utilisateurs de créer un jeton de session signé par leur DID pour une durée limitée [9]. L'utilisateur signe une *capability* (via le standard CACAO – inspiré des UCAN) indiquant que "le DID X autorise l'application Y à agir en son nom sur la période T". Ce token de session se présente comme un JWT contenant le DID du signataire, les *claims* autorisées et une expiration [9]. Il évite à l'utilisateur de signer chaque requête, tout en restant auto-porté et vérifiable. Le projet est **actif** (3Box Labs), en **TypeScript**. *Usage potentiel*: DID-Session n'est pas un fournisseur OAuth2 en soi, mais pourrait être combiné avec un wrapper OIDC. Par exemple, une application web pourrait accepter un *DID session token* de l'utilisateur (prouvé via son wallet blockchain ou autre) et en échange émettre un cookie de session ou un token OAuth. Ceramic s'oriente plus vers la gestion de données que l'auth web, mais la technologie CACAO/UCAN qu'il utilise peut servir de base à un système d'auth décentralisée interopérable.

Comparatif des projets

Projet	Fonctionnalité principale	Intégration OAuth2/OIDC	Statut (activité)	Technologies clés
Fission ODD SDK (Webnative)	Auth local-first via DID `did:key` + tokens UCAN (capacités) [1]. Pas d'infra nécessaire (auth sans backend).	Pas native – nécessiterait un adaptateur OIDC custom.	Actif (renommé ODD en 2023).	*TypeScript* (SDK web), Rust (FS sécurisé).

Projet	Fonctionnalité principale	Intégration OAuth2/OIDC	Statut (activité)	Technologies clés
Spruce SIWE OIDC (ENS DAO IdP)	Fournisseur **OIDC** pour login Ethereum (DID:pkh). Signature wallet → JWT OIDC [3]. Plugin Discourse et intégration Auth0 dispo.	**Oui** – OIDC complet (Flow Code/Auth).	Actif (2022–2024).	*Rust* (serveur Axum), front TS (WalletConnect).
Spruce Kepler	Stockage décentralisé par **DID + capacités** (*Orbits* privés) [5]. Délégation d'accès via UCAN.	N/A (orienté API stockage, pas login web).	Actif (alpha, dev. continu).	*Rust* (nœud serveur), WASM/TS (SDK client).
Web3.Storage / w3up	API d'upload IPFS avec auth par **UCAN** auto-porté. DID utilisateur = identifiant, plus besoin de mot de passe [2] [6].	Partiel – utilise un Bearer UCAN propriétaire (JWT étendu). Pas d'OIDC public direct.	Actif (prod par Protocol Labs).	*TypeScript* (libs UCAN, CLI), Go/Rust côté service (partiel).
Keycloak SSI Provider	Extension **Keycloak** acceptant une preuve d'identité décentralisée (ex. Credential signé DID) pour ouvrir une session OIDC [7] [8].	**Oui** – via Keycloak (standard OIDC).	Actif (expérimental, 2023).	*Java* (SPI Keycloak), TS/Node (service vérification).
Ceramic DID Session	Jeton de session DID signé (format **CACAO**, proche UCAN) pour déléguer actions à une appli pendant X temps [9].	Pas directe – nécessite un wrapper pour émettre un token OAuth2 depuis le DID session.	Actif (lib v0.5+ en 2023).	*TypeScript* (lib npm did-session, did-jwt).

En résumé, plusieurs projets montrent qu'il est possible de faire le lien entre identités décentralisées et les protocoles d'authentification web existants. Des solutions prêtes à l'emploi commencent à émerger, notamment autour de *Sign-In with Ethereum* (Spruce SIWE) qui a déjà des intégrations OIDC utilisables sur des forums ou sites web. D'autres projets, plus expérimentaux (Fission ODD, Ceramic, etc.), fournissent les briques fondamentales – DIDs, clés Ed25519, tokens UCAN ou assimilés – avec l'idée de redonner le contrôle à l'utilisateur sur son identité. Ces derniers nécessitent encore le développement de passerelles ou d'adaptateurs pour s'interfacer avec OAuth2, mais montrent la voie vers une

authentification sans tiers de confiance, même dans un contexte d'applications web classiques [6]. Les communautés autour de DID/UCAN sont actives et l'on peut s'attendre à ce que l'interopérabilité avec OAuth2 s'améliore rapidement, rendant possible l'utilisation de ces identités décentralisées en **production** sur des plateformes comme Discourse.

Sources : Les détails ci-dessus s'appuient sur la documentation et dépôts officiels des projets (Fission [1], SpruceID [3] [5], Protocol Labs [2] [6], Dizme [7], Ceramic [9], etc.), ainsi que sur les discussions de la communauté spécialisée. Chaque projet référencé est open source et lié via son dépôt GitHub pour plus d'informations techniques.

---

[1] Multi Wiki Server: Authentication and Authorization - fission - Talk TW
https://talk.tiddlywiki.org/t/multi-wiki-server-authentication-and-authorization/9093

[2] [6] Intro to UCAN
https://blog.web3.storage/posts/intro-to-ucan

[3] [4] Spruce 2022 in Review
https://blog.spruceid.com/spruce-2022/

[5] GitHub - spruceid/kepler: Decentralized storage based on permissioned data overlays called orbits.
https://github.com/spruceid/kepler

[7] [8] GitHub - dizme/keycloak-ssi-provider: SSI Provider Implementation for Keycloak to Login with a custom Credential
https://github.com/dizme/keycloak-ssi-provider

[9] Decentralized Identifiers | Ceramic documentation
https://developers.ceramic.network/docs/introduction/did-overview