Keystone

What is it?

TLDR;

Keystone is a Cosmos-native system for:

1. Using and managing a key*chain* associated with a human user
2. Integrating Web 2.x (social login, passwordless login) notions of user identity with Web 3.x notions of identity.

Keystone uses Cosmos groups to achieve a similar multi-key signing as that provided by tor.us using Shamir.

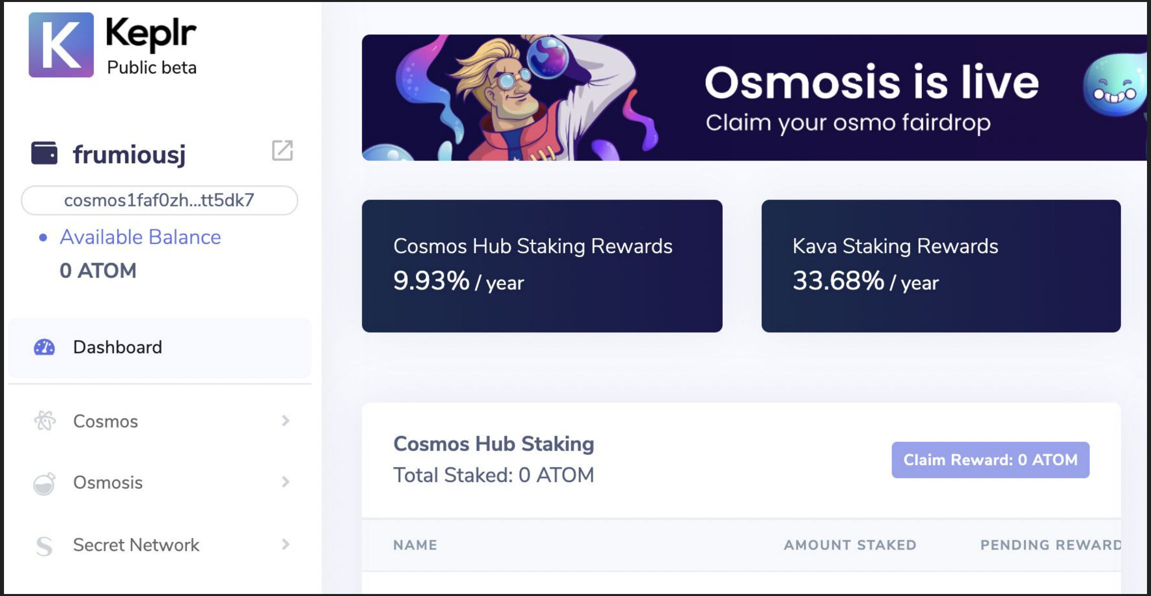
Keystone provides no API for “returning a key” (key bytes) but instead offers a sign/verify/encrypt/decrypt API where the key bytes themselves are not provided, allowing higher security (keys cannot be “stolen”) and environments where keys are stored separately from the services that use them.

Keystone offers a Javascript SDK for client integration, an IdP, and a HTTP/gRPC service API for key management.

Keystone allows not only custodially-held keys, but also user-controlled keys to be used to sign transactions

Existing marketspace

- Kepler wallet: a Cosmos-native wallet for multiple chains
- Tor.us: an elegant blockchain-friendly distributed key management system
- FIDO/W3C WebAuthn: passwordless login
- Auth0 + other “proxy verifiers”: social login identity provider (proxy, integrator)



frumiousj

cosmos1faf0zh...tt5dk7

Available Balance
0 ATOM

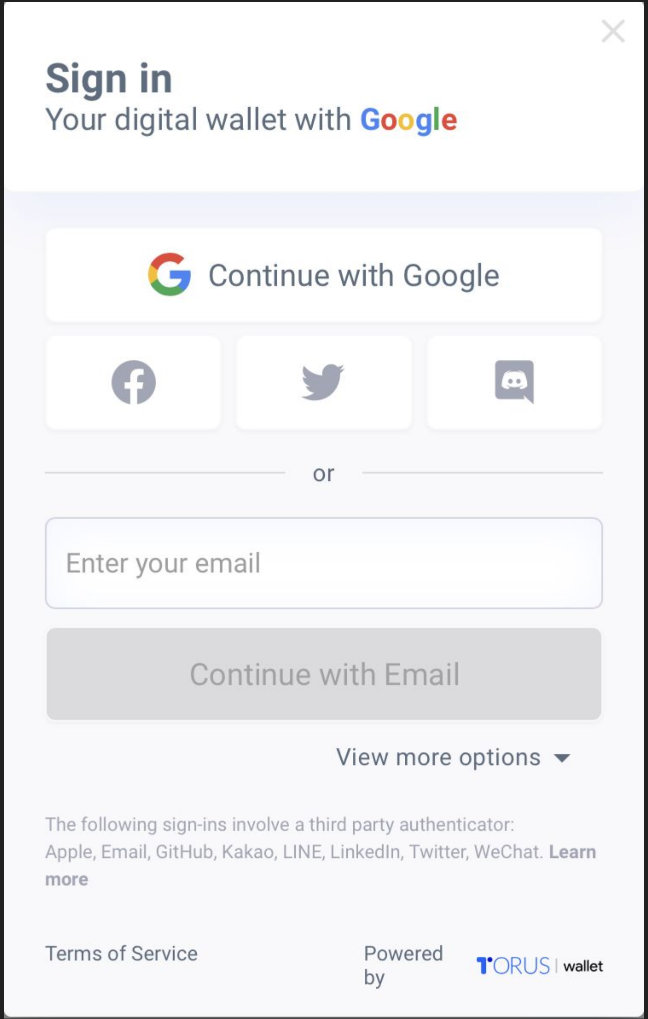
Cosmos Hub Staking Rewards
9.93% / year

Kava Staking Rewards
33.68% / year

Cosmos Hub Staking
Total Staked: 0 ATOM

NAME	AMOUNT STAKED	PENDING REWARD

- Blockchain-native Chrome extension wallet
- Cosmos-native, but not (yet!) using groups-based key management (keplr keystone login feature?)
- Uses Chainapsis “tor.us signin” private repo, suggesting tor.us is used for key distribution/management


Sign in

Your digital wallet with Google

Continue with Google

or

Enter your email

Continue with Email

View more options

The following sign-ins involve a third party authenticator: Apple, Email, GitHub, Kakao, LINE, LinkedIn, Twitter, WeChat. **Learn more**

Terms of Service

Powered by TORUS | wallet

TORUS

- Distributed key management via “key shares” stored on-chain
- Key sharing using Shamir Secret Sharing algorithm to assemble a single key from shards
- Social login integrated with blockchain-based key management
- Provides *CustomAuth* integration that could integrate with Cosmos/Regen
- Crucially, presents a getTorusKey() API which results in full key assembly for users

Keystone Components

Keystone Identity Provider, UI & HTTP/gRPC service

- OAuth 2-based identity provider that can act as a proxy verifier, using auth0
- Offers direct passwordless registration & login (via WebAuthn JS API) to blockchains
- Maintains a *user profile* where keys may be added, removed and rotated manually, and signature thresholds (m of n) may be created and maintained.
- Mints OAuth tokens for other user-based Regen systems (e.g. Registry) that act as OAuth 2 relying parties (same as today, but with Keystone IdP instead of auth0)
- A human user is represented as a Cosmos group of addresses/keys

Keystone KeyKey

- Key Management Service based on Cosmos Groups
- Sign = propose transaction within the user group, vote within the user group, and execute the transaction.
- Verification - post-consensus of block inclusions, light client check for fork detection (if no fork detection, signed transactions are verified)
- Add address (key) to group
- Remove address (key) from group

Keystone JS API

- Provides operations from both Keystone IdP and Keystone KeyKey, as a client-side API
- Allows other UI integrations (Keystone “widget” in someone else’s UI)
- Offers the equivalent of the tor.us CustomAuth API, built on Cosmos Groups instead of SSS
- No getKey() API (but sign/verify instead…)