

THE DEFENSE CHRONICLES



Giorgos Lagos
Researcher at ICCS

Proof-First Access Control: Bringing Zero Trust to Decentralised Identity

Most access control systems make a practical assumption: once a user has signed in and received a token, the identity behind that token can be trusted until it expires. This is convenient, but it is not always a good fit for distributed infrastructure. A credential may be revoked after logging in. A software component may change operator. A request may also target a different resource or action from the one that was originally approved. In CIPHER, we address this gap by checking verifiable credential evidence before authorisation, close to the point where access is requested.

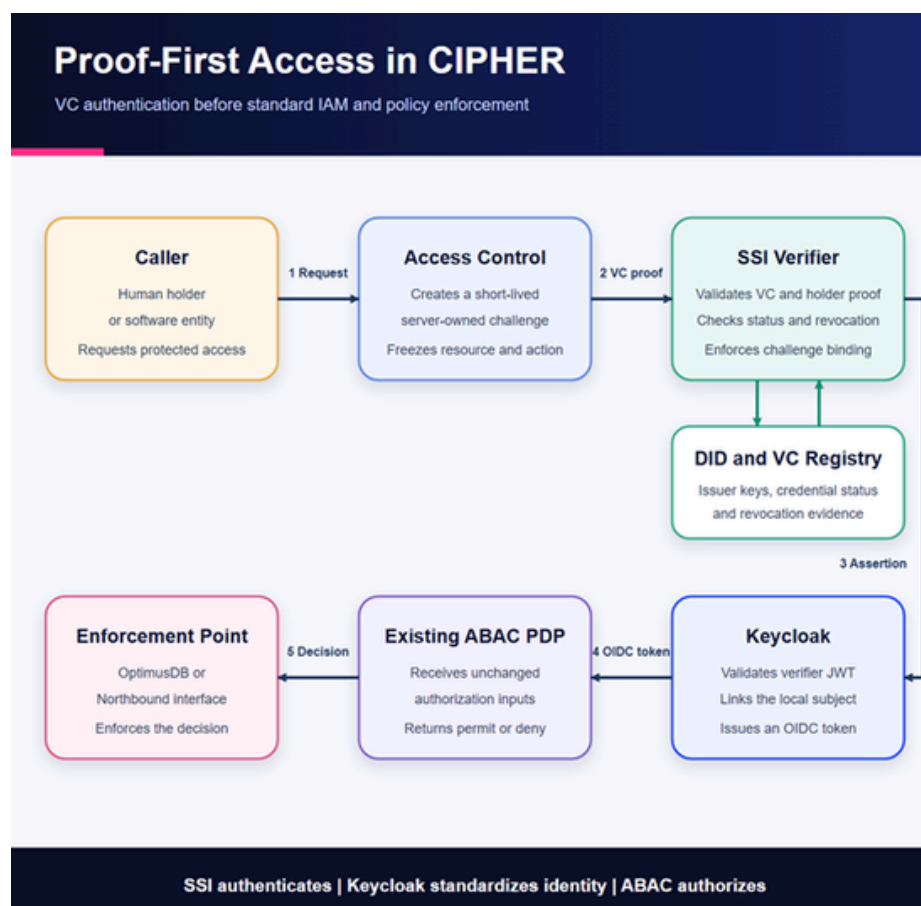
Separating authentication from authorisation

One design choice was important from the start. Self-sovereign identity (SSI) should not replace Keycloak or the existing attribute-based access control engine. These systems solve different problems and are already familiar to infrastructure operators. The Self-sovereign identity verifier establishes that a holder controls a valid credential. Keycloak maps that verified holder to an operational identity and issues a standard OpenID Connect access token. The existing Attributed-based Access Control (ABAC) policy decision point still makes the final permit or deny decisions.

Keeping these responsibilities separate allows decentralised identity to be introduced without rewriting the Identity and Access Management (IAM) and policy infrastructure already in place.

A proof-first request

An access attempt starts with a challenge from the Access Control API. The challenge is short-lived and controlled by the server. It fixes the resource, action, audience, verifier, and purpose of the request. A human holder can respond through the Web User Interface, while a software entity can use the Software Development Kit or REST API. In both cases, the holder creates a verifiable presentation and signs the challenge with the private key tied to the credential. The verifier resolves the issuer Decentralised identifier (DID) and public key, then checks the credential hash, expiry, registry status, revocation state, holder DID and wallet binding. It also validates the EIP-712 signatures protecting both the credential and its presentation. Any change to the challenge, audience, purpose or expected holder category causes the request to fail. If verification succeeds, the verifier creates a short-lived signed JSON Web Token (JWT) assertion. Keycloak validates that assertion through its JSON Web Token Authorisation Grant capability and returns an OpenID Connect access token for the matching local subject. We do not copy the full verifiable credential into Keycloak. Keycloak only needs the operational identity and the minimum set of attributes required by the existing authorisation process.



Binding proof to the exact decision

For each authorisation request, CIPHER binds a hash of the exact Keycloak token to the one-time challenge, the authenticated subject, the verified wallet, the resource and the action. When access is evaluated, the token must still be active and must contain the expected Access Control information. The wallet associated with the Keycloak subject must also match the wallet proven by the credential. If any part of this chain is missing or inconsistent, processing stops before the Policy Decision Point (PDP) is called. A Self-sovereign identity claim cannot grant access by itself, and an SSI-derived role, group, organisation's value or scope cannot override an Attributed-based Access Control denial. The decision is returned to the policy enforcement point, such as OptimusDB (CIPHER's decentralised data management system) or a Northbound interface, which must enforce it before allowing the protected operation.

Value

The aim is to shorten the distance between the credential evidence and the access decision. Proof is checked when protected access is requested, Keycloak remains the bridge to standard OpenID Connect and Attributed-based Access Control remains the policy authority. This gives CIPHER a practical route toward continuous assurance without asking operators to discard the systems they already use.

References

- [W3C Verifiable Credentials Data Model 2.0](#)
- [W3C Decentralised Identifiers Core 1.0](#)
- [Keycloak JWT Authorisation Grant](#)
- [EIP-712 Typed Structured Data Hashing and Signing](#)
- [CIPHER Project](#)

CIPHER WP3, Task 3.3. The article describes the integrated development prototype for human holders and software entities.



FOLLOW US!