Zero-Knowledge Proof (ZKP) Authentication API Documentation

Version: 1.0.0
Last Updated: May 14, 2025
Author: Maria Sultana — Technical & SEO Content Writer

Overview

The Zero-Knowledge Proof (ZKP) Authentication API allows decentralized applications (dApps), zero-trust systems, and privacy-first web platforms to authenticate users without revealing sensitive data, using zk-SNARKs or zk-STARKs.

Unlike traditional token or password-based authentication, this API uses advanced cryptographic proofs to verify identity without exposing underlying secrets—ideal for GDPR-compliant apps, crypto wallets, and identity-less authentication models.

SEO Keywords

- Zero-Knowledge Proof Authentication API
- zk-SNARKs authentication example
- Privacy-first authentication system
- Web3 identity without KYC
- Zero-trust authentication with zk-STARK
- Decentralized identity verification
- zkLogin API for Ethereum and Solana

Key Use Cases

Use Case	Description
Decentralized Login	Authenticate users via ZK identity proofs without email/password.
KYC-Free Access Control	Verify proof of age/citizenship without collecting personal data.
Zero-Trust Systems	Authenticate employees based on verifiable claims without storing credentials.

| zk-Social Login | Log in with GitHub/Twitter using signed ZK proofs instead of OAuth tokens. |

Core Features

- ZKP-based authentication using Groth16 or PLONK prover systems
- Stateless verification: no data stored on server
- zkLogin via Ethereum/Solana wallet signatures
- Anonymous session token generation
- Supports Circom, Halo2, and zkSync circuit formats

Architecture Overview

[User Device] --> [ZK Circuit Compiler] --> [Proof Generator] --> [ZKP Auth API] --> [Session Token]

Proof Format Requirements

Supported ZK Circuits: circom 2.0, snarkjs, halo2, zkSync custom circuits

POST /api/v1/auth/verify

Verifies a zk-SNARK proof and returns a session token if valid.

POST /api/v1/auth/zkLogin

Authenticates users with a signed message from a supported wallet and validates the zero-knowledge identity claim.

Security Model

- No user PII stored
- Proofs are stateless and ephemeral
- JWT tokens are signed with ECDSA/P-256
- Supports end-to-end zk rollups for scalable multi-user auth

Implementation Tips

- Use snarkjs in browsers for lightweight proof generation
- Avoid hardcoding public signals

- Use a CDN for large proving keys
- Pre-compile circuits for production

SDKs and Tools

Language	Package
JavaScript	zk-auth-js
Rust	zk_auth_rs
Python	zk_auth
Golang	github.com/yourorg/zk-auth

Example: Age Verification Circuit (Circom)

```
template IsOver18() {
  signal input birthYear;
  signal output isOver18;

  component comp = LessThan(1);
  comp.in[0] <== 2006; // Current year - 18
  comp.in[1] <== birthYear;

  isOver18 <== comp.out;
}
```

Resources & Whitepapers

- zk-SNARKs Explained (Vitalik Buterin)
- Circom Language Docs
- Zero-Knowledge Proofs for Identity
- zkLogin: Privacy-Preserving Login Protocol
- Decentralized ID with ZKPs (W3C Draft)

License