Wrap-Up, ZK-Logins

A learning group for ZK and SNARK application development

Daniel Szego

In: https://www.linkedin.com/in/daniel-szego/

Agenda

- *Wrap up - season 2025*
- *ZK proof - summary*
- *Web3 UX problem*
- *OAuth and federated identity*
- *OAuth and blockchain access*
- *ZK login flow*
- *Generate ZK proof*
- *Demo*
- *Links and resources*
- *ZK Learning group - Season 2026*
- *Q&A*

Wrap up - season 2025

February - Introduction, Theory : Definitions and building blocks

March - Theory : Polynomial commitments

April - Theory : Interactive oracle proofs

May - Programming : Circom

June - Programming : Circom

July - Programming : Noir - basics

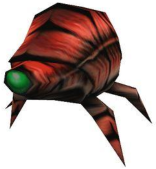
August - Programming : Noir - advanced

September : Applications : Proof of reserve, proof of solvency

October : Applications : ZK machine learning

November : Applications : Rollup

December : Wrap up, Applications, ZK-login



Discord channel:

https://discord.com/channels/905194001349627914/1329201532628898036

Meetup.com:

https://www.meetup.com/lfdt-hungary/events/305634614/

Repo with all the contents:

https://github.com/LF-Decentralized-Trust-labs/zk-learning-group

Zero knowledge proofs - summary

Computation: arithmetic circuit : *C(x, w) -> F*

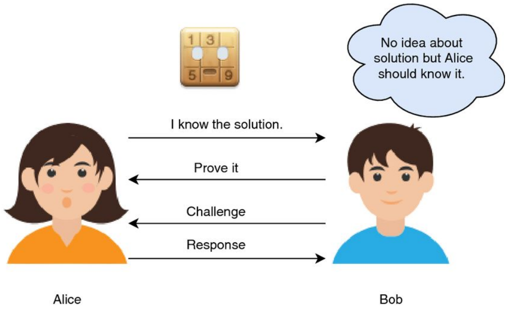
- *x* public input
- *w* private input, witness
- high level computation
- arithmetic circuit
- polynomials

Prover algorithm: *P (pp, x, w) -> proof*

Verifier algorithm: *V (vp, x, proof) -> accept / reject*

Properties:

- *Succinct:*
- *Complete:*
- *Knowledge sound:*
- *Zero knowledge*



Web3 UX problem

Problems with onboarding new users

Handling private-public keys, addresses, mnemonics, etc …

Non-custodial wallets

- Risks related to keys
- Secure storage
- Safe backup
- Protocol risks

Custodial wallets:

- Counterparty risks
- Centralization risks
- Legal and compliance risks

OAuth (OpenID) and federated identity

Delegated authorization / access

Authentication server: token for authentication and authorization:

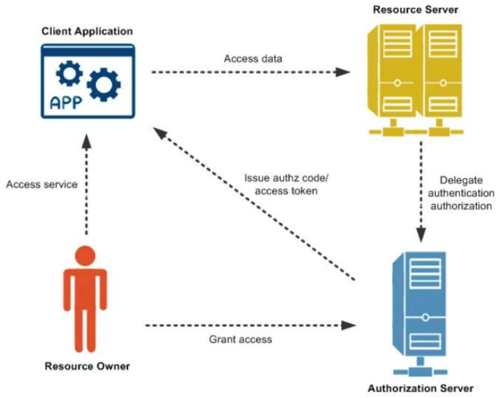
- Owner
- Client
- Resource server
- Authentication server
- Access / refresh token

Third party login

Authorization

Microservice communication

Mobile / desktop apps



Source: https://en.wikipedia.org/wiki/OAuth

OAuth and blockchain access

Token (e.g. JWT):

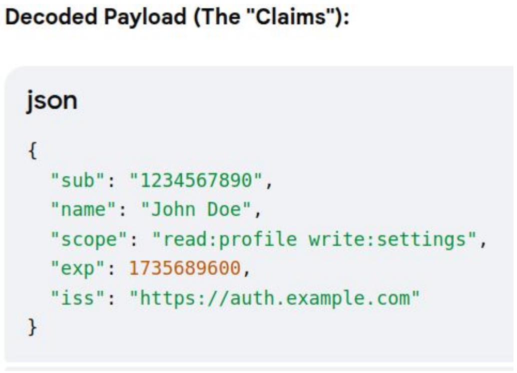
- Header: metainfo
- Payload: claims
- Signature: signed by issuer

ZK-login : a novel idea to use OAuth and JWT tokens for authenticating with DApps and initiating blockchain transactions.

Integrate Web2 authentication with Web3 in a non-custodial way.

Individual blockchain addresses for OAuth tokens, without being publicly traceable.

Special key claim in the token for generating addresses



Decoded Payload (The "Claims"):

json

```json
{
  "sub": "1234567890",
  "name": "John Doe",
  "scope": "read:profile write:settings",
  "exp": 1735689600,
  "iss": "https://auth.example.com"
}
```

ZK Login flow

Generate ephemeral keys: generate temporary key pair for the user

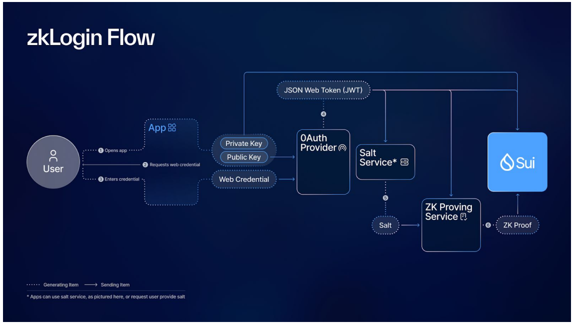
Generate a JSON Web Token (JWT): public key in nonce, key claim

Request the user's unique salt: salt for unique address generation and hiding address

Generate a zk proof: on correctness

Identify the user's address and construct transaction: address is based on key claim and user salt + verify signatures and zk proofs

Validate transaction: send to blockchain and verify signatures and zk proofs. Directly L1 validation, no oracle + smart contract



Source: https://blog.sui.io/zklogin-deep-dive/

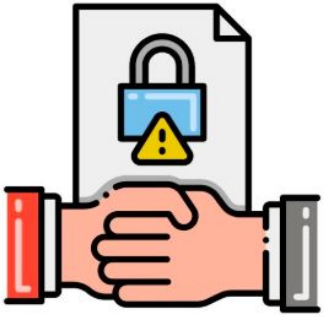
Generate ZK Proof

Address based on the hash value of the persistent values of the token + salt

Salt: secret information for deriving Web3 address

To validate with generating a ZK proof::

- the nonce is defined correctly and includes the public key
- the key claim provided is consistent with the JWT
- the address is consistent with the key claim and user salt
- the OAuth providers signature is correct



Demo

Sui zkLogin Demo

https://sui-zklogin.vercel.app/

Security and privacy

2 Factor authentication model:

- OAuth authentication
- Salt: used for deriving address (secret)

Temporal private key:

- It can be refreshed (temporary)
- Expiry

Privacy:

- Address = H (iss, aud, sub, salt)
- provider, client, user : jws parameters
- salt: private but prover knows (!)

Counterparty risk:

- Oauth provider dependency
- E.g. google inaccessibility ?

Challenge

Test ZK-login with an OAuth authentication provider on a test network

Links, Resources, Literature

OAuth Explained: How It Works & Why It Matters
https://frontegg.com/blog/oauth

zkLogin Demystified: Exploring Sui's Cutting-Edge Authentication
https://blog.sui.io/zklogin-deep-dive/

zkLogin: Privacy-Preserving Blockchain Authentication with Existing Credentials
https://arxiv.org/abs/2401.11735

Sui zkLogin Tutorial
https://hackmd.io/@moritzfelipe/HkEBKKzYa

Sui zkLogin Demo
https://sui-zklogin.vercel.app/

ZK Learning group - season 2026

Every month or two months

Similar structure, similar logistics + ***guest lectures***

Theory:

- Look tables,
- Zk variations: STARKs, SNARGs
- recursive SNARKs

Programming / platforms:

- Zinc / ZKSync
- Leo

Applications:

- Private transactions
- ZK Virtual Machines
- ZK Identity / verifiable credential - presentations
- Voting

Discord channel:

https://discord.com/channels/905194001349627914/1329201532628898036

Meetup.com:

https://www.meetup.com/lfdt-hungary/events/305634614/

Repo with all the contents:

https://github.com/LF-Decentralized-Trust-labs/zk-learning-group

Happy Hunting for the SNARK :)

Q & A

Daniel Szego
In: https://www.linkedin.com/in/daniel-szego/