KU LEUVEN

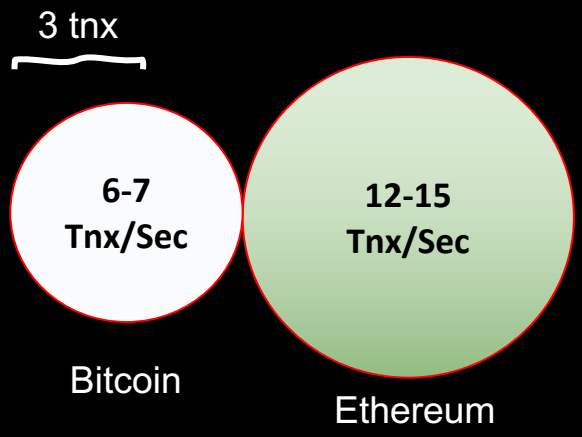
COSIC

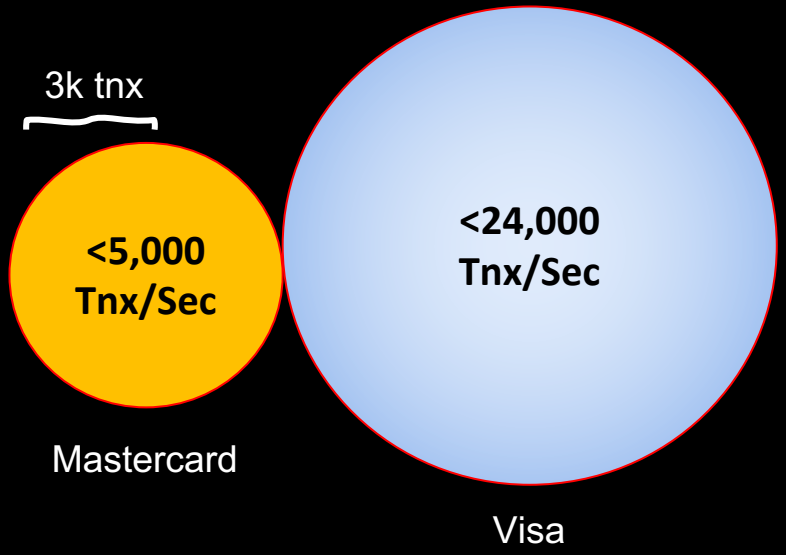
Zero-Knowledge Proofs: Applications to Blockchain

Mahdi Sedaghat
COSIC, KU Leuven

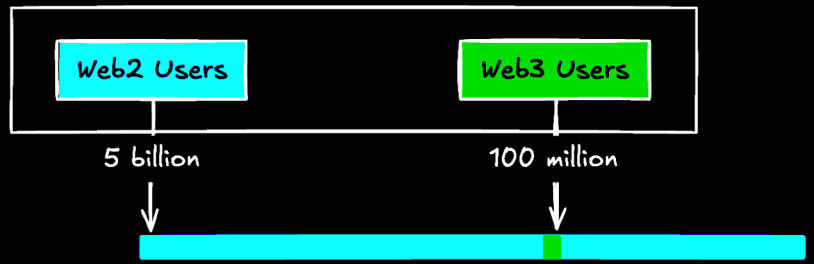
Cybersecurity Flanders Industry event - Mechelen - November 14

Web 3.0 vs. Web 2.0: Limitations

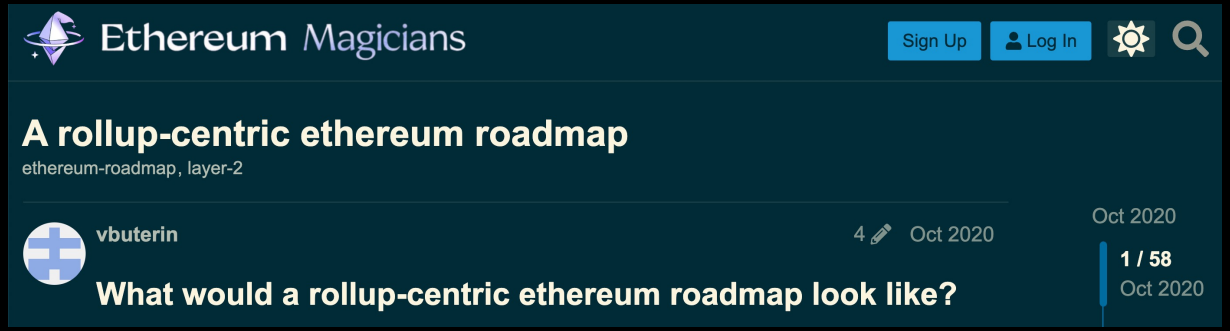




100 million active crypto wallets and there are several **BILLIONS** of Web2.0 accounts.

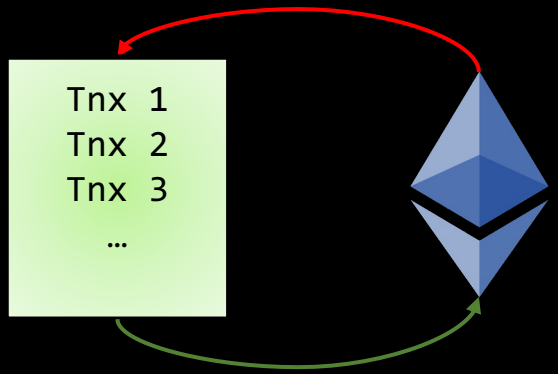


Rollup-centric Roadmap: To improve Ethereum's scalability



Rollups: As a scaling solution by executing the tnxs outside L1 while posting them on-chain.

How does Ethereum ensure that executed transactions are processed correctly and weren't submitted by a bad actor for their own benefit?



Rollups: A simple analogy

To make sure everyone has a valid ticket.

To bring passengers to their destination as fast as possible, with minimal stops or delays.

Ethereum main chain

To make sure all transactions are valid (Based on EVM).

To settle transactions as many and quickly as possible.

L1 vs. L2: Bus onboarding analogy

Layer 1

The bus driver is responsible to check all the passengers' tickets

L1 is responsible to check the validity of all transactions

Optimistic Rollups

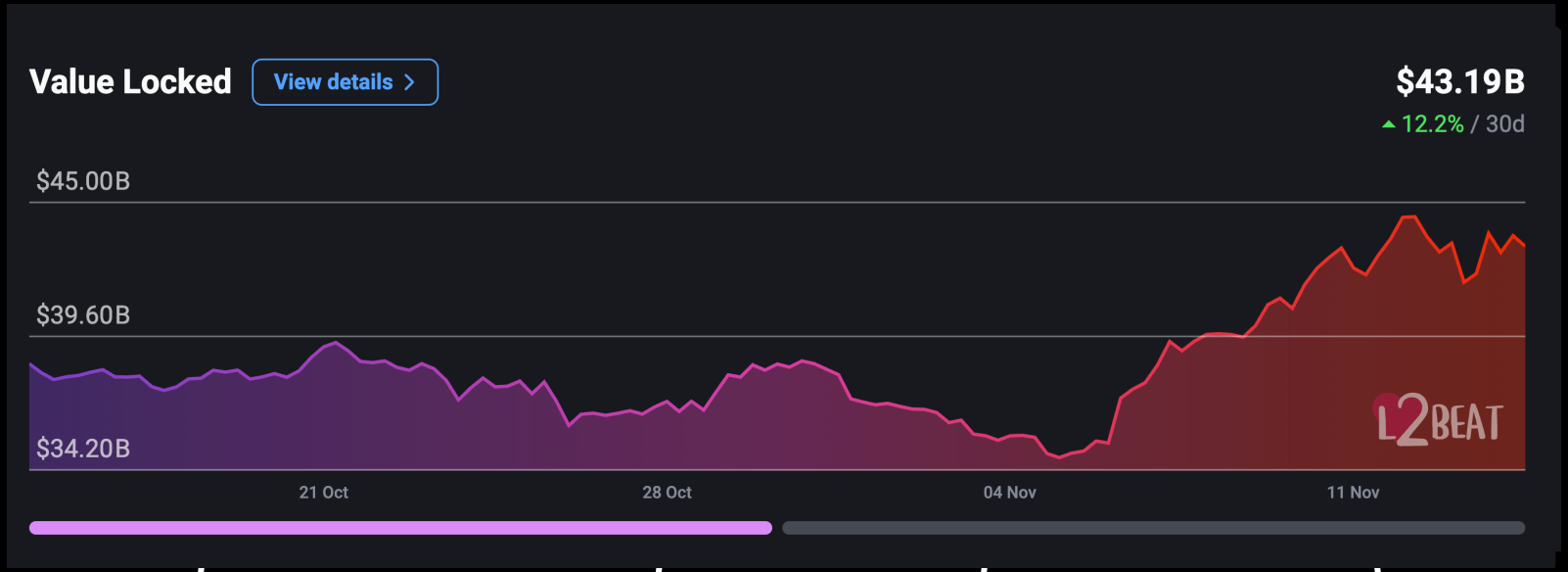
Let the passenger enter without immediate ticket inspection.

All tnx are valid, unless proven otherwise.

zk-Rollups

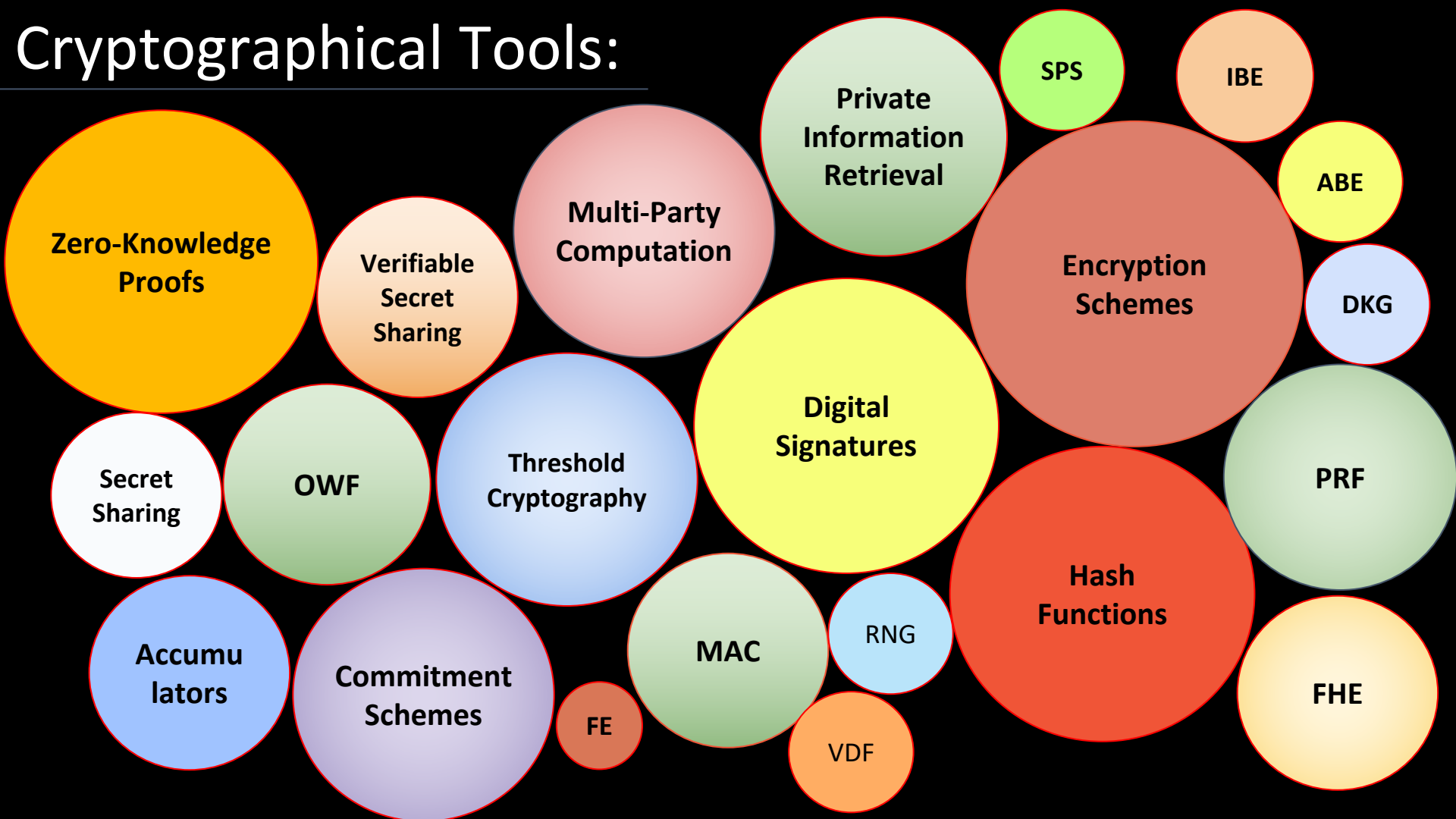
The ticket booth issues tickets along with a validity proof.

The validity of the transactions being proved.



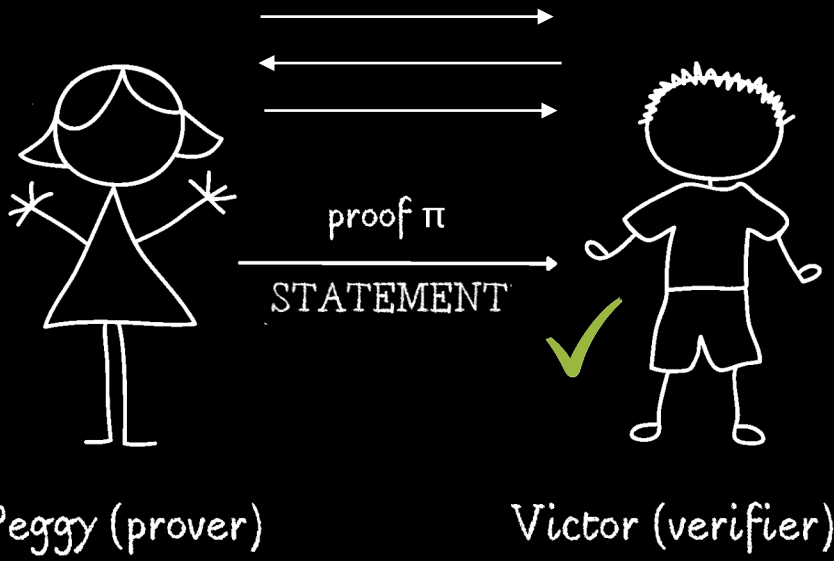
	Security	Withdrawal period	Off-chain Computation cost	Complexity of technology
Optimistic Rollups	Honest actions & Economic incentizes	1-2 weeks	Low	Simple
Zk-Rollups	Cryptographic certainty	Nearly instant	High	Hard

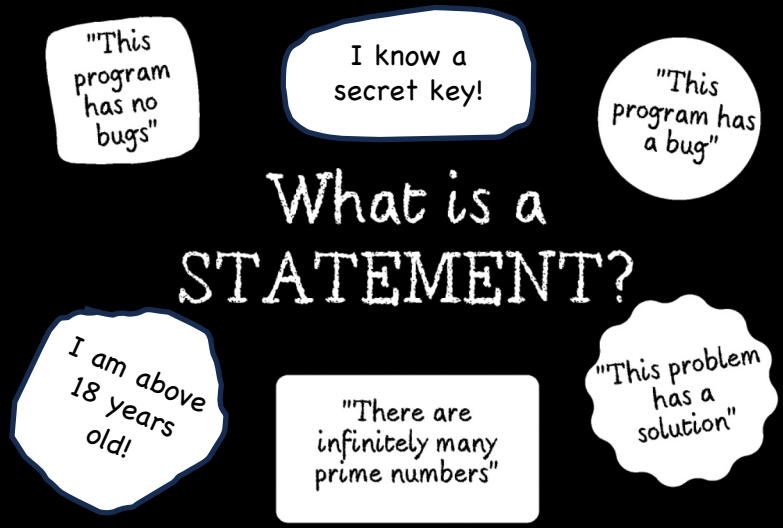
Cryptographical Tools:



Zero-Knowledge Proofs:

A strong cryptographic pritimve



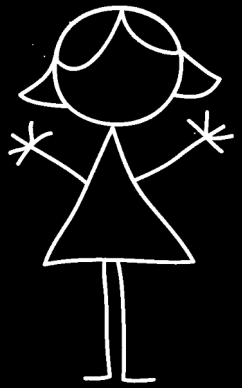


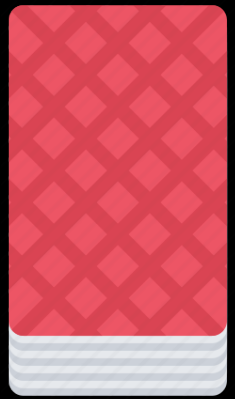
1. Completeness:
An honest prover can always convince a verifier.

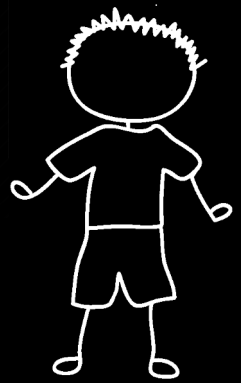
2. Soundness:
A dishonest prover cannot convince a verifier.

3. Zero-Knowledge:
The proof does not leak any information beyond the validity of the statement.

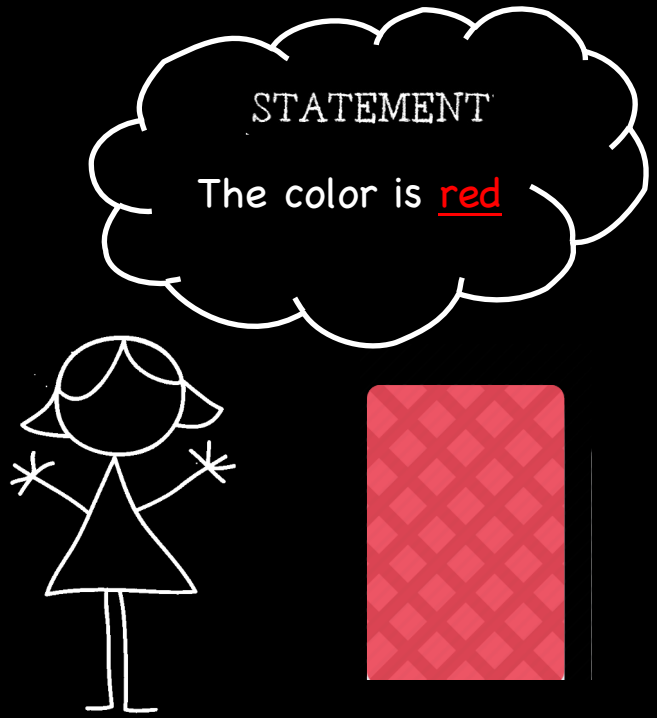
Let’s play a game:

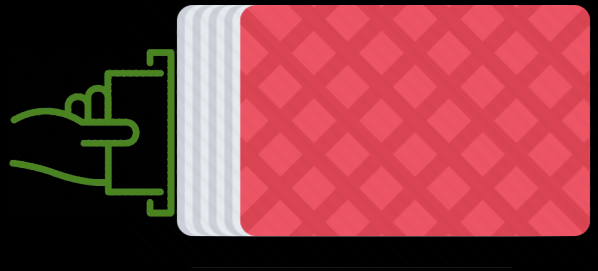


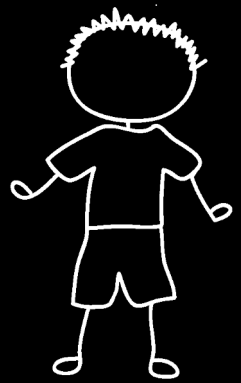




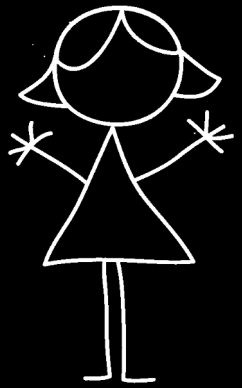
Let’s play a game:

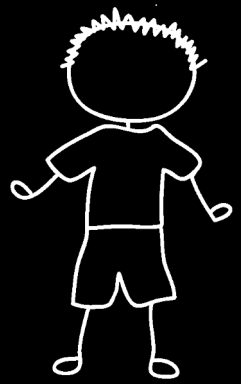




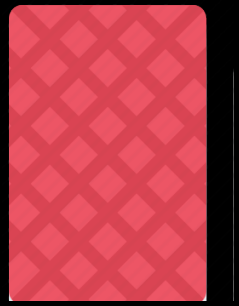


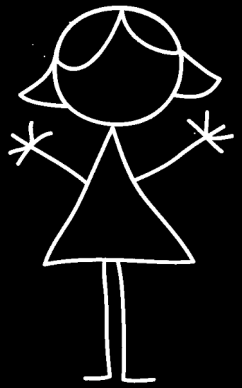
Let’s play a game:

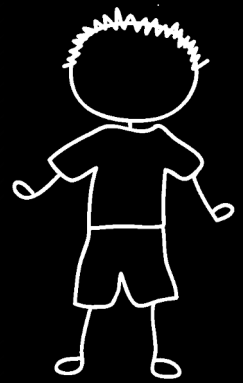




Let’s play a game:

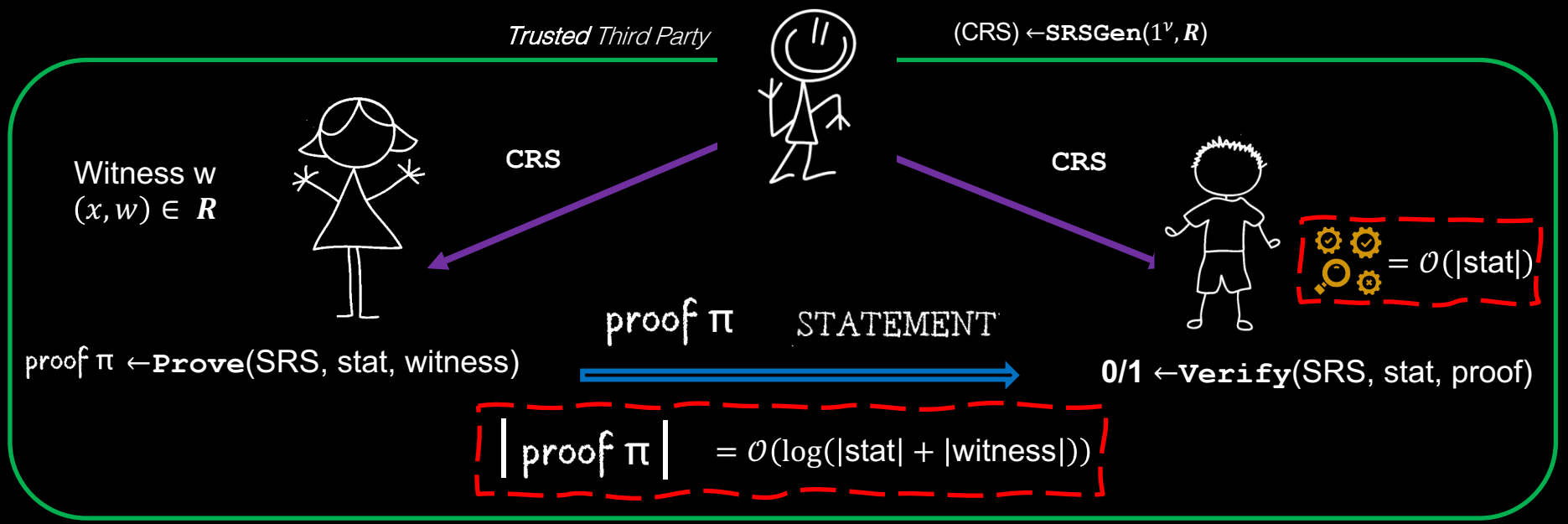






Victor Accpets if #black_cards = 26

zk-SNARKs: A practical version



3 **Zero-Knowledge**

The verifier doesn't learn anything beyond the validity of the statement.

4 **Knowledge Soundness**

A malicious prover cannot convince a verifier, unless it knows some secret "witness".

* CRS:Common Reference String

MAP OF ZK in 2024

Source: ZK Podcast

MAP OF ZK in 2024

Payments
ZCASH
firo
Daimo
payy
IRON FISH

L1s
MINA
PENUMBRA
Aleo
HORIZEN
NAMADA
ALEPH ZERO
anoma

Hardware
Ilrreducible
INGONYAMA
SUPRA NATIONAL
CYSIC
FABRIC

L2s
Aztec
polygon
ZKsync
Linea
taiko
STARKNET
Scroll
MANTA

DeFi
RENEGADE
PANTHER
Dark-fi
PENUMBRA
offshift

ZKML/AI
EZKL
NOYA
Modulus Labs
ZkAGI
ZKML

Coprocessor
AXIOM
BREVIS
HERODOTUS
marlin
lagrange
RITUAL
vlayer

Gaming
dark forest
Immutable
BLADE
ZYPHER AI NETWORK
ZORDLE
zkHoldem
TileVille
cartridge

Verifiable
Compute
Jolt
RISC ZERO
NEXUS
Succinct
NovaNet

Cross-chain
Hyperlane
=nil;
zeko
polygon
Succinct
Union

ID
zkLogin
Sui

ZKTLS
PADO
PLUTO
TLSN
ZK EMAIL

Prover Network
GEVULOT
Succinct
FERMAH
lagrange
ZKPOOL
RISC ZERO

Proof Verif.
NEBRA
ALIGNED
HYLÉ
zkVerify
Soundness Lab

R&D/Audit
geometry RESEARCH
0xPARC
reilabs
Zellic
ZKS
NETHERMIND
LAMBDA CLASS
ZK HACK
Veridise

ZK in BTC
Alpen
BVM
Citrea

Misc/Tools
Nouns DAO
TACEO
Anonymous DAO
Delphinus Lab
Snarkify
CLIQUE
Cursive
powdr
Argument Computer Corporation
RAILGUN_

Source: ZK Podcast

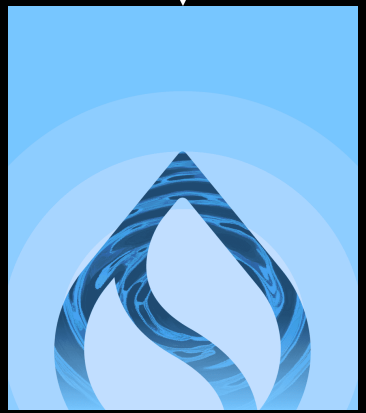
zkLogin: Simplifying Onboarding to Web3 Using SNARKs

Foteini Baldimitsi | Kostas Chalkias | Yan Ji | Jonas Lindstrøm | Deepak Maram |
Ben Riva | **Mahdi Sedaghat** | Arnab Roy | Joy Wang

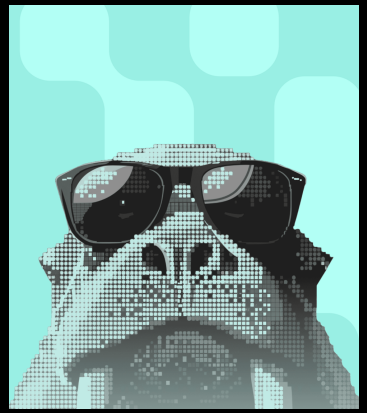
Mysten Labs and Sui:

Deployed on

Over 500k transactions made using this mechanism.

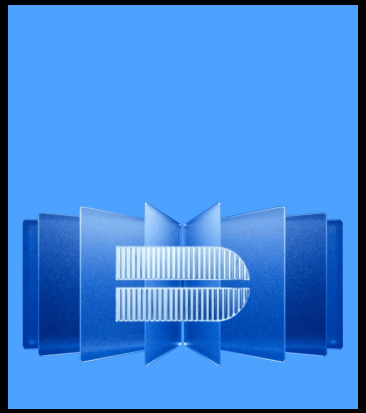


Sui (L1)

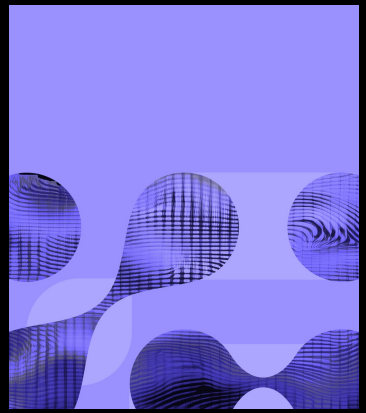


Walrus (DDA)

Move
(Rust Smart Contract)

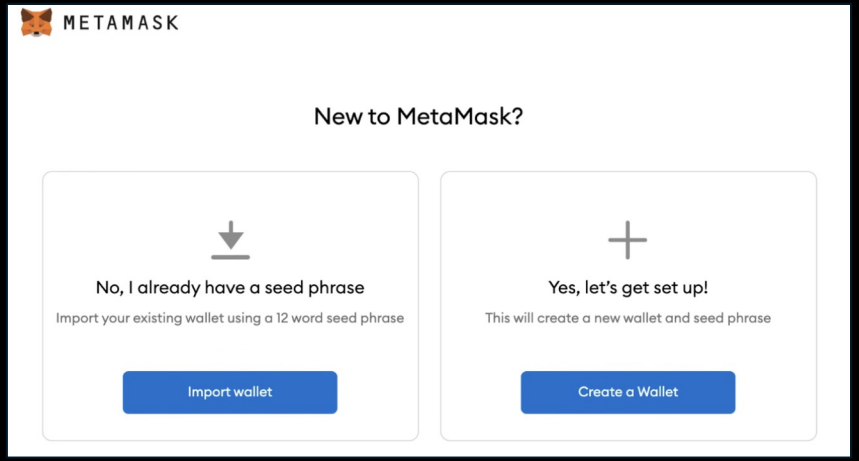


Deepbook
(DeFi)

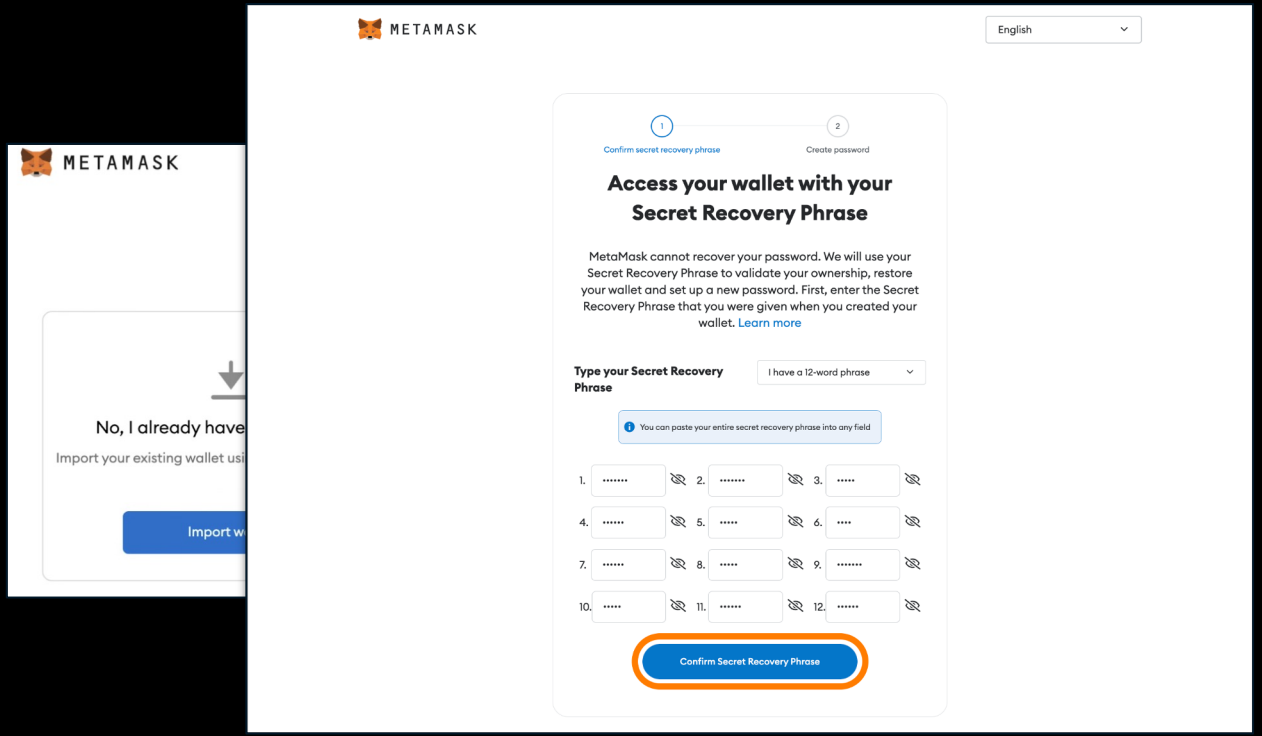


Enoki

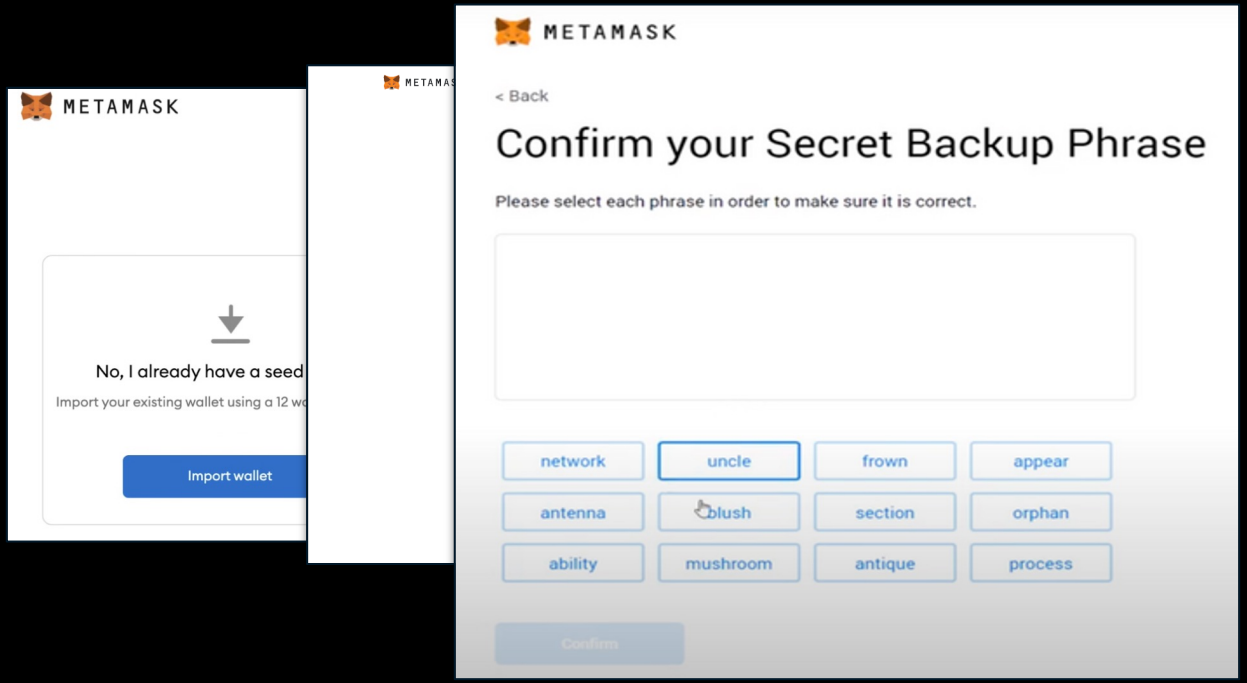
Web3 has an onboarding problem



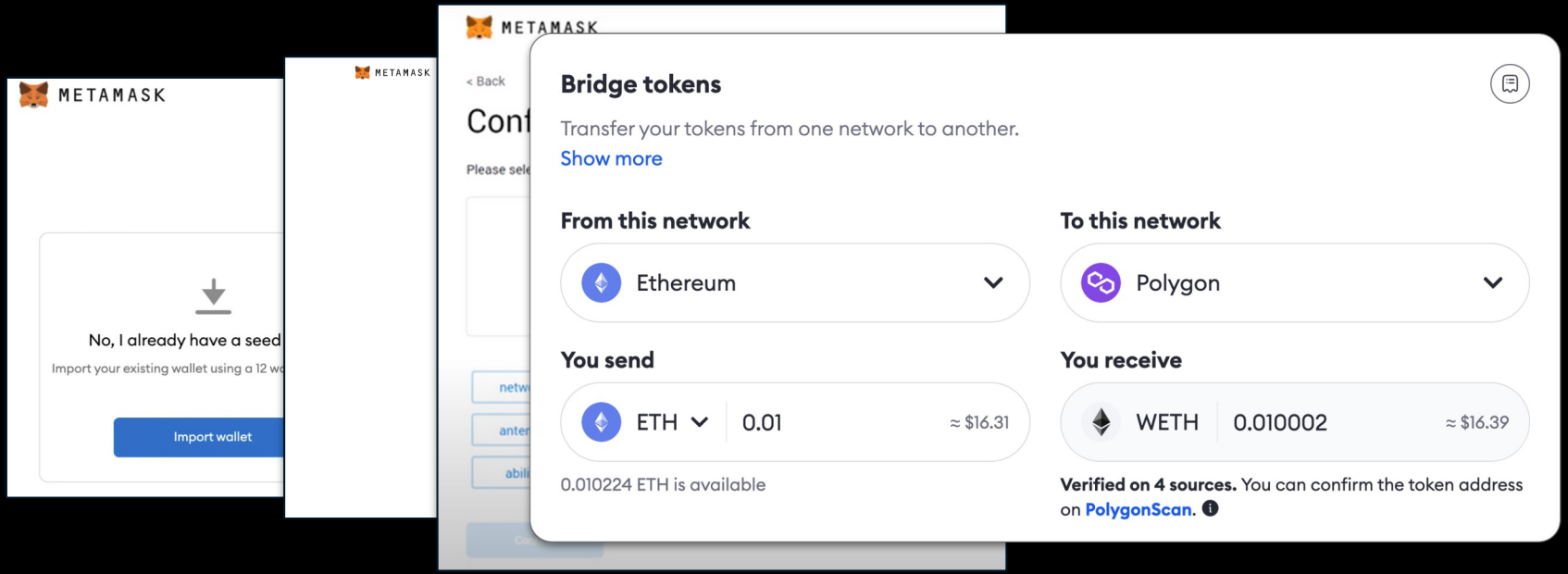
Web3 has an onboarding problem



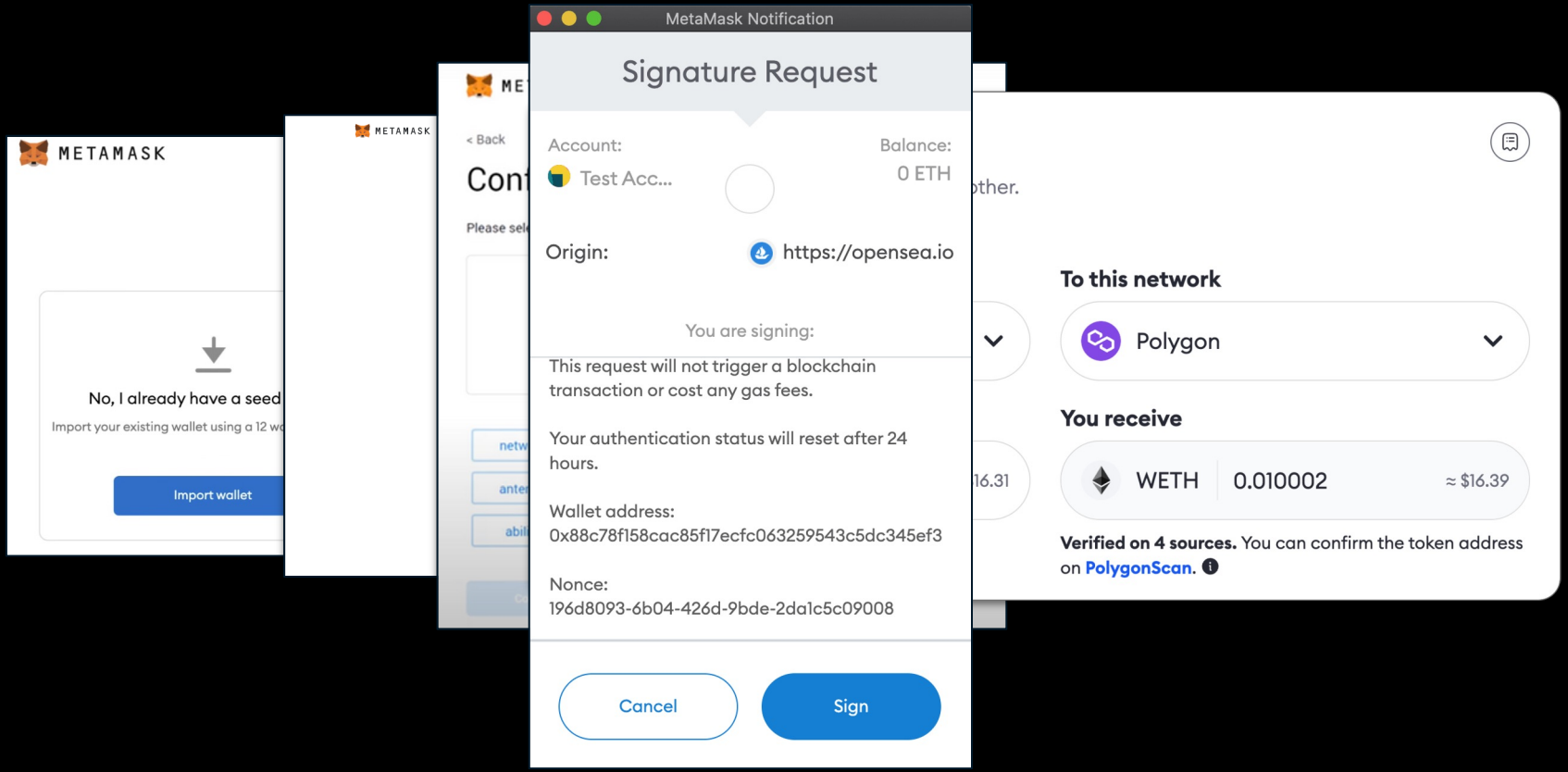
Web3 has an onboarding problem



Web3 has an onboarding problem

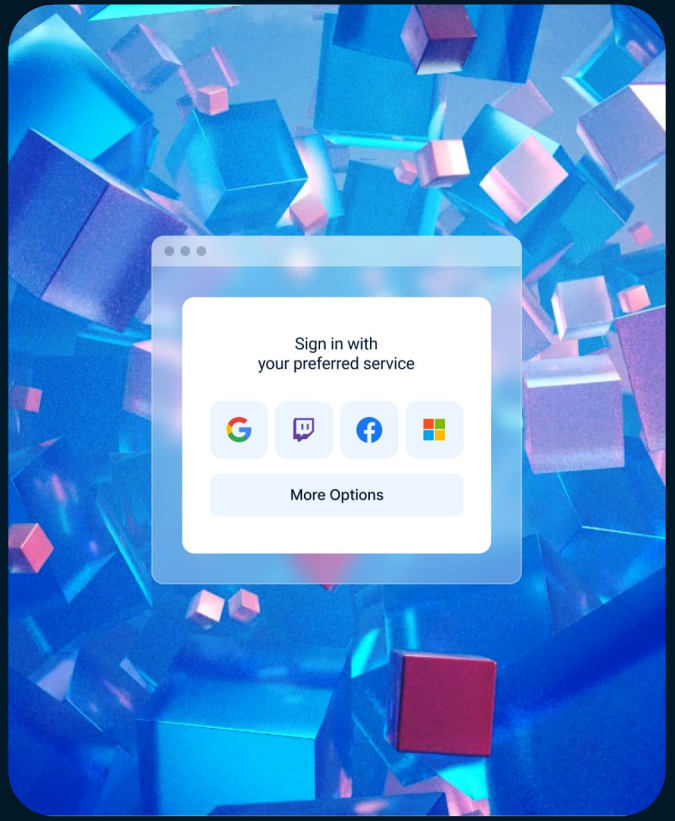


Web3 has an onboarding problem



Can we make it as easy as signing in with Google, Facebook and co?

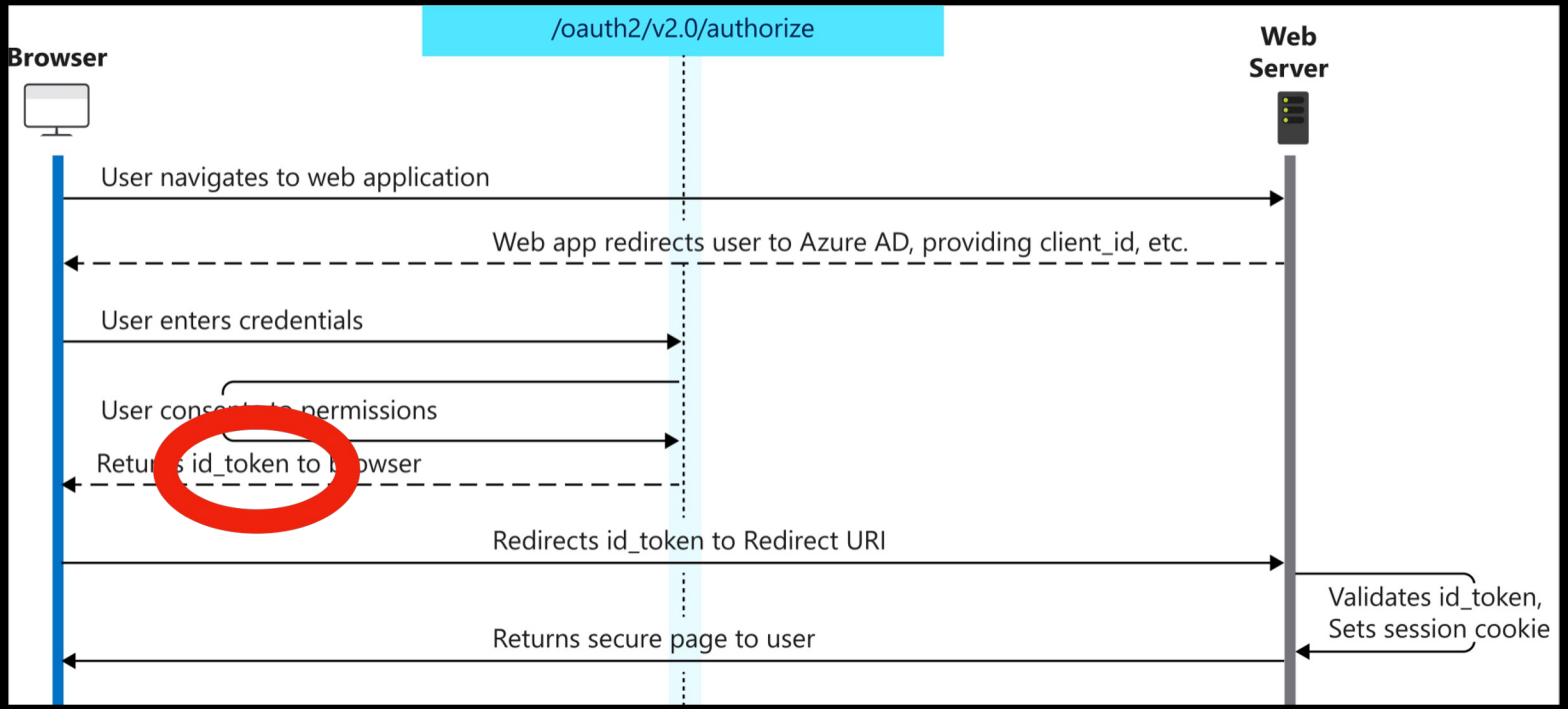
- People don’t want to use separate passwords for each and every app, each and every web2 service
- Extremely likely they already have a Google, Facebook, Amazon account
- Solution: use OAuth to leverage these already existing accounts



zkLogin:
OAuth + zk-SNARKs

Non-custodial
User-friendly
Privacy-preserving

OpenID Connect (an extension of OAuth 2.0)



JWT: JSON Web Token

Base64-encoded, RSA-signed

JWT as an alternative to a private key?

Encoded PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ
zdWIiOiJwaGlsaXBwZUBwcmFnbWF0aWN3ZWJzZWN
1cml0eS5jb20iLCJyb2xlIjoiYWRtaW4iLCJpc3M
iOiJwcmFnbWF0aWN3ZWJzZWN1cml0eS5jb20ifQ.
jW4cq__pkcq-r6H1Ebiq8toW-
4Igstk1ibRgxECUhdExvZTzhvXqfrPewgtRHEApB
WXpUqGqRY6LSj2Gklxt3O6kxUaky-
VTl8jbL0OV5HEQVOnL3VVgPv65ddGRYaCOuyzYcf
6M1fA4PeFme9lL2ZTNtjiE0OjUR3LH1Dptm_u9_
aQRtJ_IU8xiywctV1JLeQcMJFDXCS2N5oU0Gkatu
oJNbjMdSTg3BsU5yUsGLyuPnJTeUWJajin5e0NuB
A1Bc6oLee6KtPAM8-
1ufhHr1fpT78iGyrSQLpiVd2naPA0CvUyZ6W_4ar
nmZDKRF9N9zOR_Jxyfv5xFMi4G67EhA
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "RS256"
}
```

PAYLOAD: DATA

```
  "sub": "philippe@pragmaticwebsecurity.com",
  "role": "admin",
  "iss": "pragmaticwebsecurity.com"
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Lg8ulqDgdXLFwS/lHXV/OKcBOXBrIp
  B4DWQ0c16zLZU7NTe657rWqKlwIDAQ
  AB
  -----END RSA PUBLIC KEY-----
```

zkLogin tricks

sample openID JWT token signed by Google / FB

```
{
  "iss": "https://accounts.google.com",
  "azp": "575519204237-msop9ep45u2uo98hapqmngv8d84qdc8k.apps.googleusercontent.com",
  "aud": "575519204237-msop9ep45u2uo98hapqmngv8d84qdc8k.apps.googleusercontent.com",
  "sub": "1104634521            ",
  "nonce": "166379188139080602618705289039940387216697996138036016166781555121812732894 77",
  "iat": 1682002642,
  "exp": 1682006242,
  "jti": "a8a0728a3ffd5dc81ecfd0ea81d0d33d803eb830"
}
```

aud = walletID
sub = userID

nonce =
eph. pubKey
+ expiration

+

+ **ZK proof** =

ADDRESS
~hash(providerID + zkhash(walletID + userID + zkhash(salt)))

&

verify ZKproof + verify eph key sig

zkLogin latency

Implemented in Circom: ~1M R1CS constraints

Groth16

These numbers correspond only to the **first transaction of a session**

Operation	zkLogin	Ed25519
Fetch salt from salt service	0.2 s	NA
Fetch ZKP from ZK service	2.78 s	NA
Signature verification	2.04 ms	56.3 μs
E2E transaction confirmation	3.52 s	120.74 ms

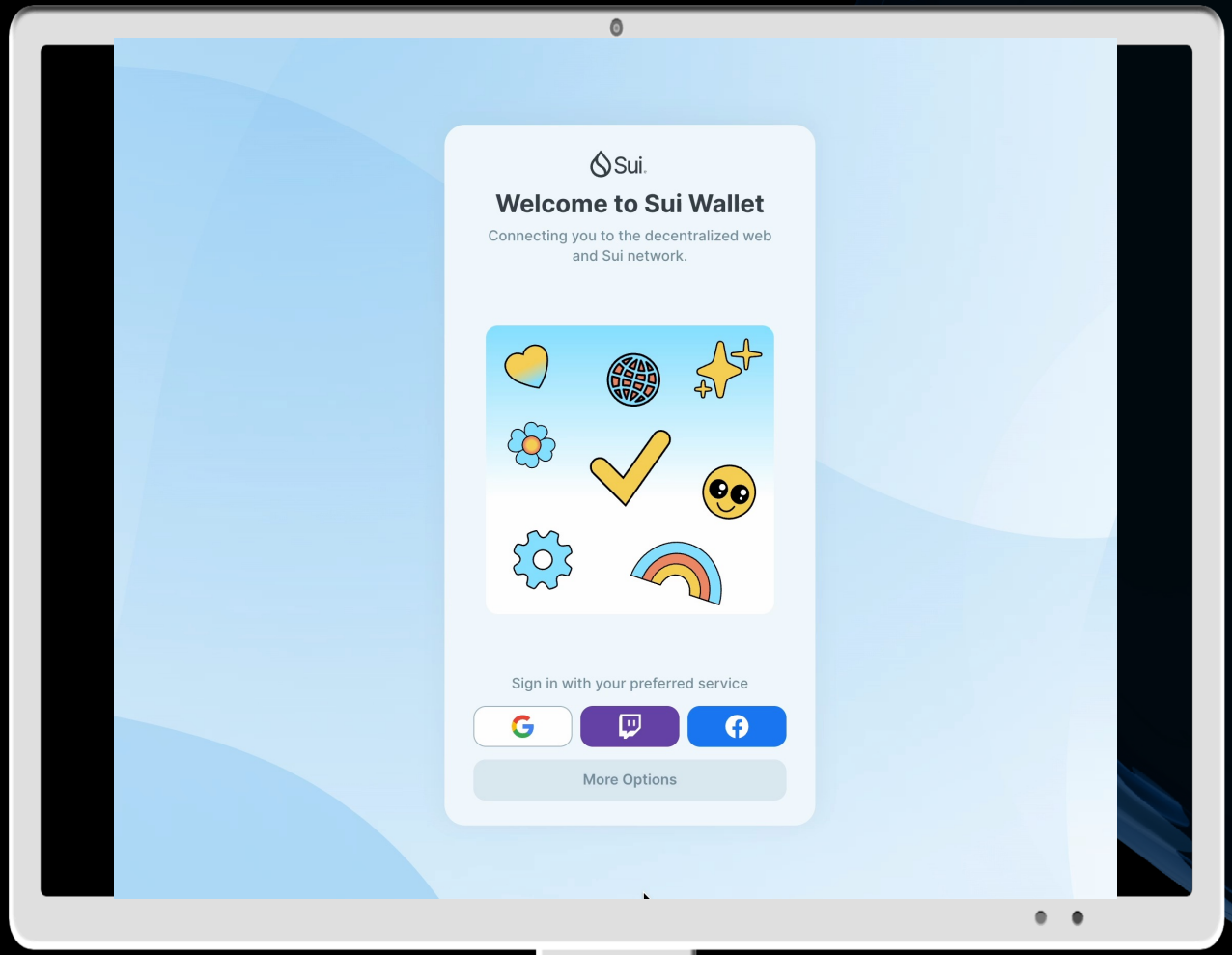
Latency for most zkLogin transactions is **very similar** to traditional ones!

zkLogin

single-click accounts w/

- Google
- Facebook
- Twitch
- Apple
- Slack
- Microsoft

native authenticator
non-custodial
*discoverable, claimable
invisible wallets
semi-portable, 2FA



KU LEUVEN

COSIC

Thank You!

- ssedagha@esat.kuleuven.be

Homepage: https://mahdi171.github.io/

Some Slides credited to Mysten Labs crypto team.