ProgCryptoSG

Intro to ZK

Flying Nobita

2024-10-16

Content

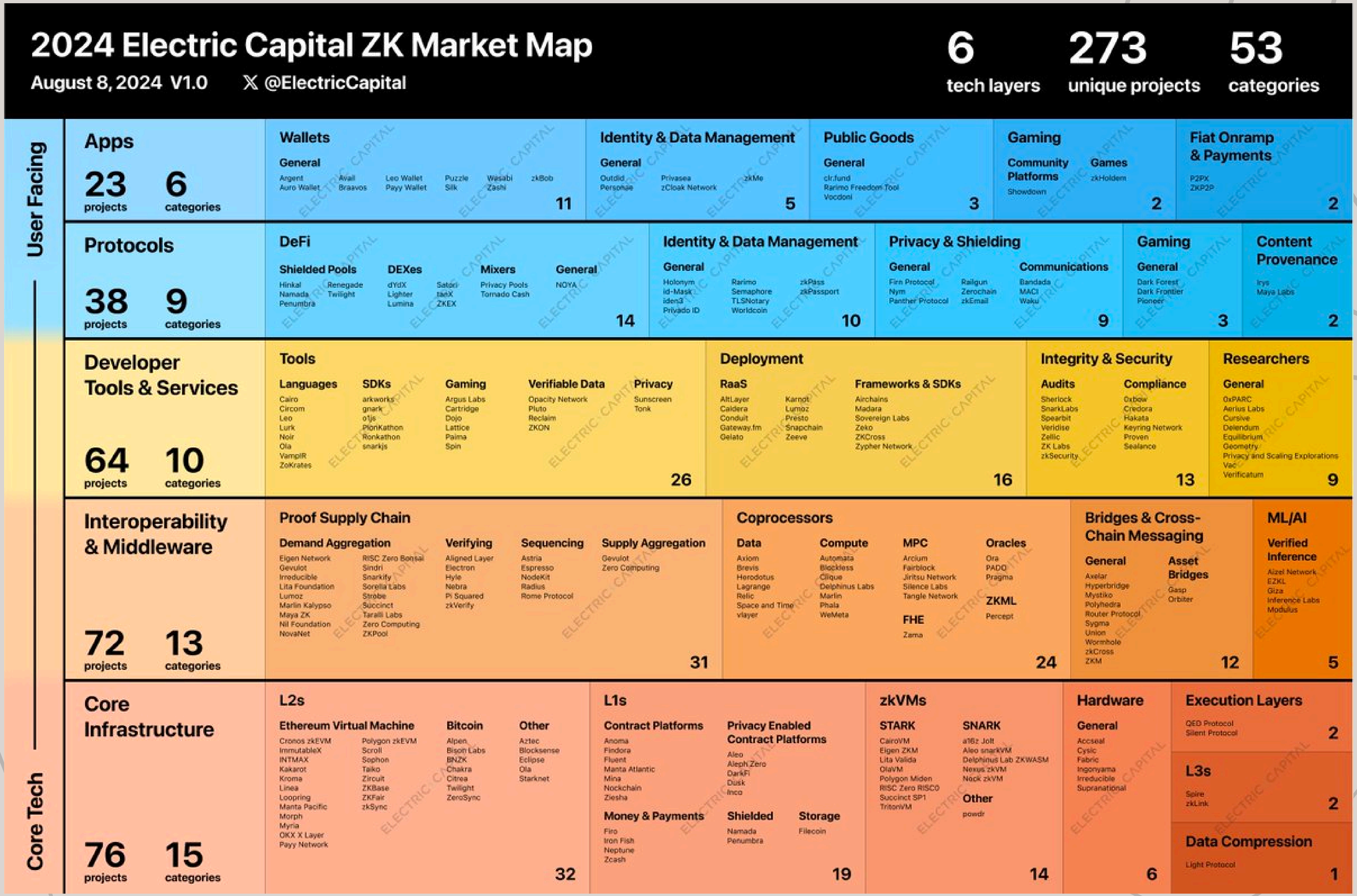
01

What is ZK Use for?

02

What is ZK?

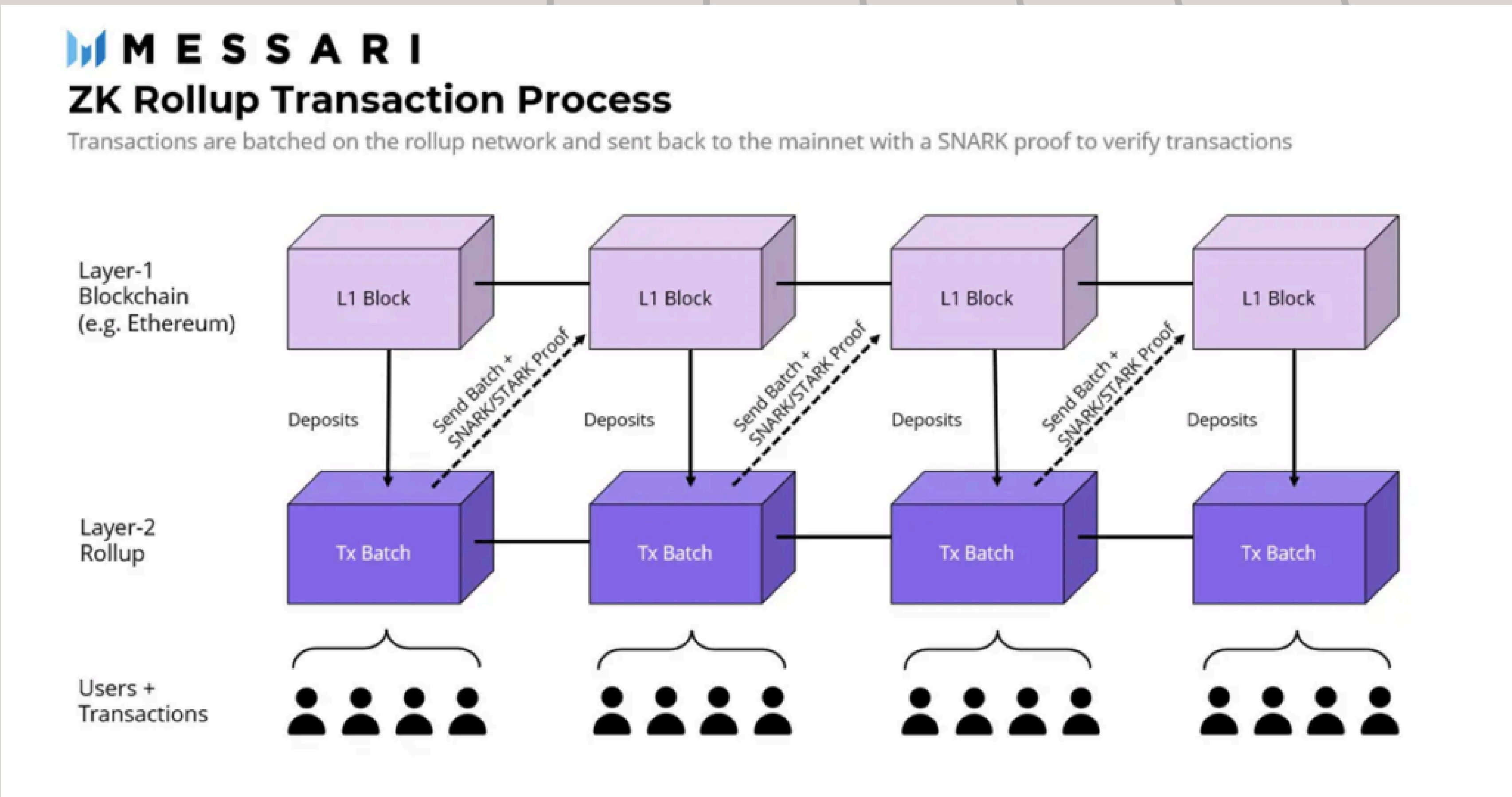
How ZK is used in blockchain?



Outsourcing Computation

Weaker machines (blockchain):

- **outsources** hard computations to powerful machines
- **verifies** the **succinct** proofs that ensure the transactions are valid



ZK Rollups scale blockchain by reducing **costs**:

1. transactions executed and batched on L2 by rollup nodes
2. proof is generated that prove the transactions are valid
3. batched transaction and proof are submitted to L1

Cost is **amortized** across many transactions and users

Outsourcing Computation

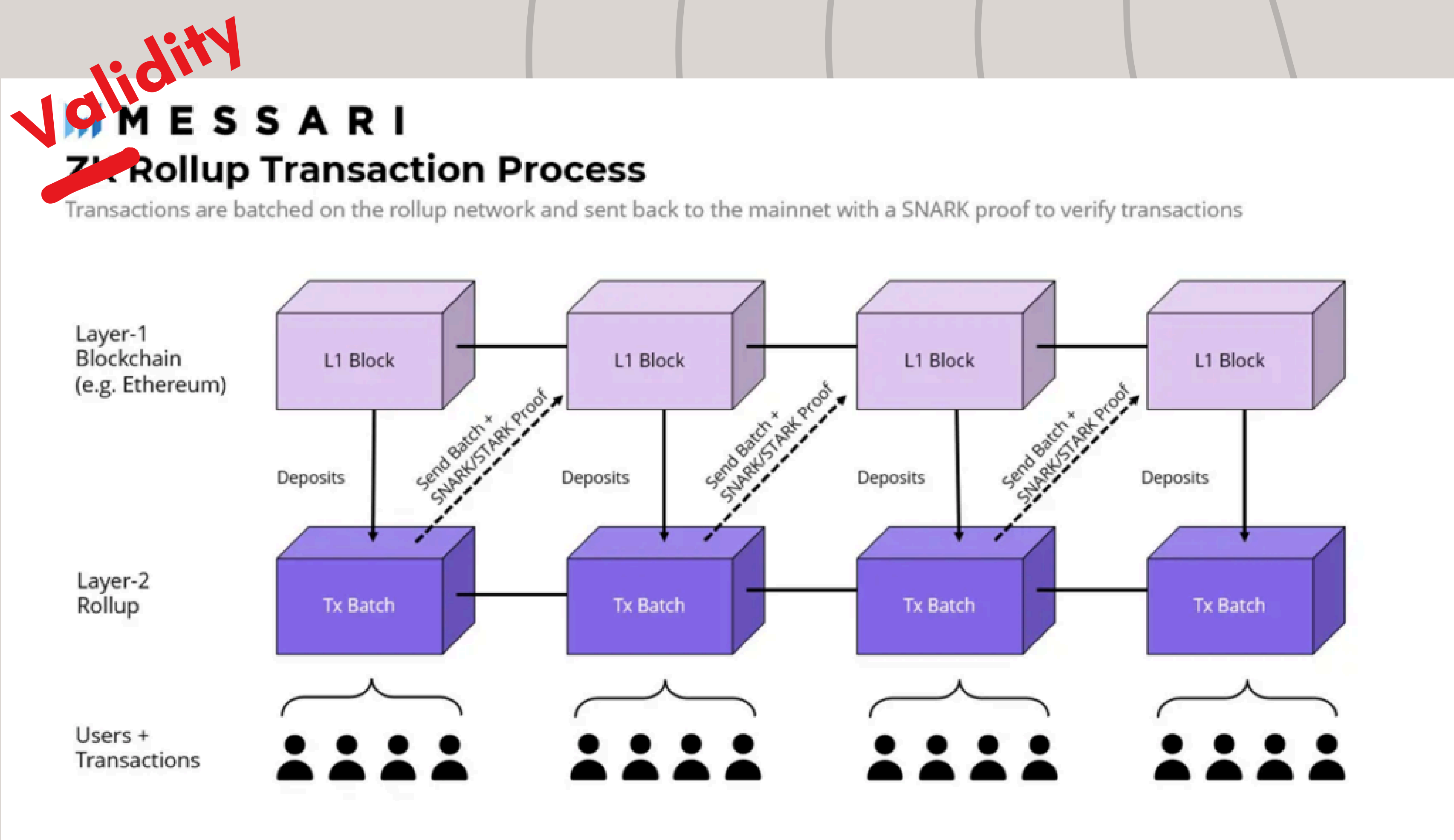
Weaker machines (blockchain):

- **outsources** hard computations to powerful machines
- **verifies** the **succinct** proofs that ensure the transactions are valid

~~ZK~~ Validity **Rollups scale** blockchain by reducing **costs**:

1. transactions executed and batched on L2 by rollup nodes
2. proof is generated that prove the transactions are valid
3. batched transaction and proof are submitted to L1

Cost is **amortized** across many transactions and users

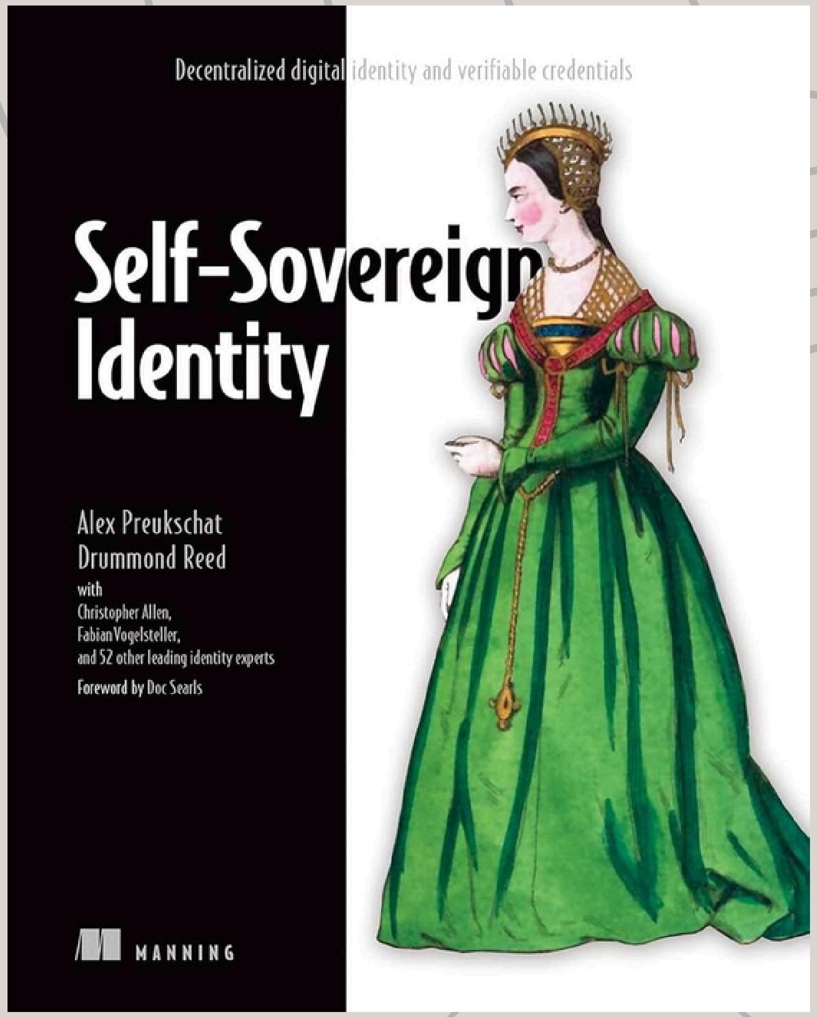


Self-Sovereign Identity

Self-sovereign identity requires:

- **selective disclosure** of attributes to **prove predicates** without disclosing the attributes themselves
- **unlinkability** between proofs
- cheap on-chain verification with **succinct** proofs for on-chain KYC

Require **Zero Knowledge Proofs** + other cryptographic primitives



Some more..

Privacy-oriented blockchain & Dapps

- **private** transactions and states
- e.g. Zcash, Aleo, Aztec, Mina

ZK Coprocessors

- a form of **outsourcing computation**
- used at the smart contracts layer

ZK Bridges

- states on chain 1 can be **succinctly** verified on chain 2
- allow transactions to flow across chains
- replaces multisigs or centralized bridges

Content

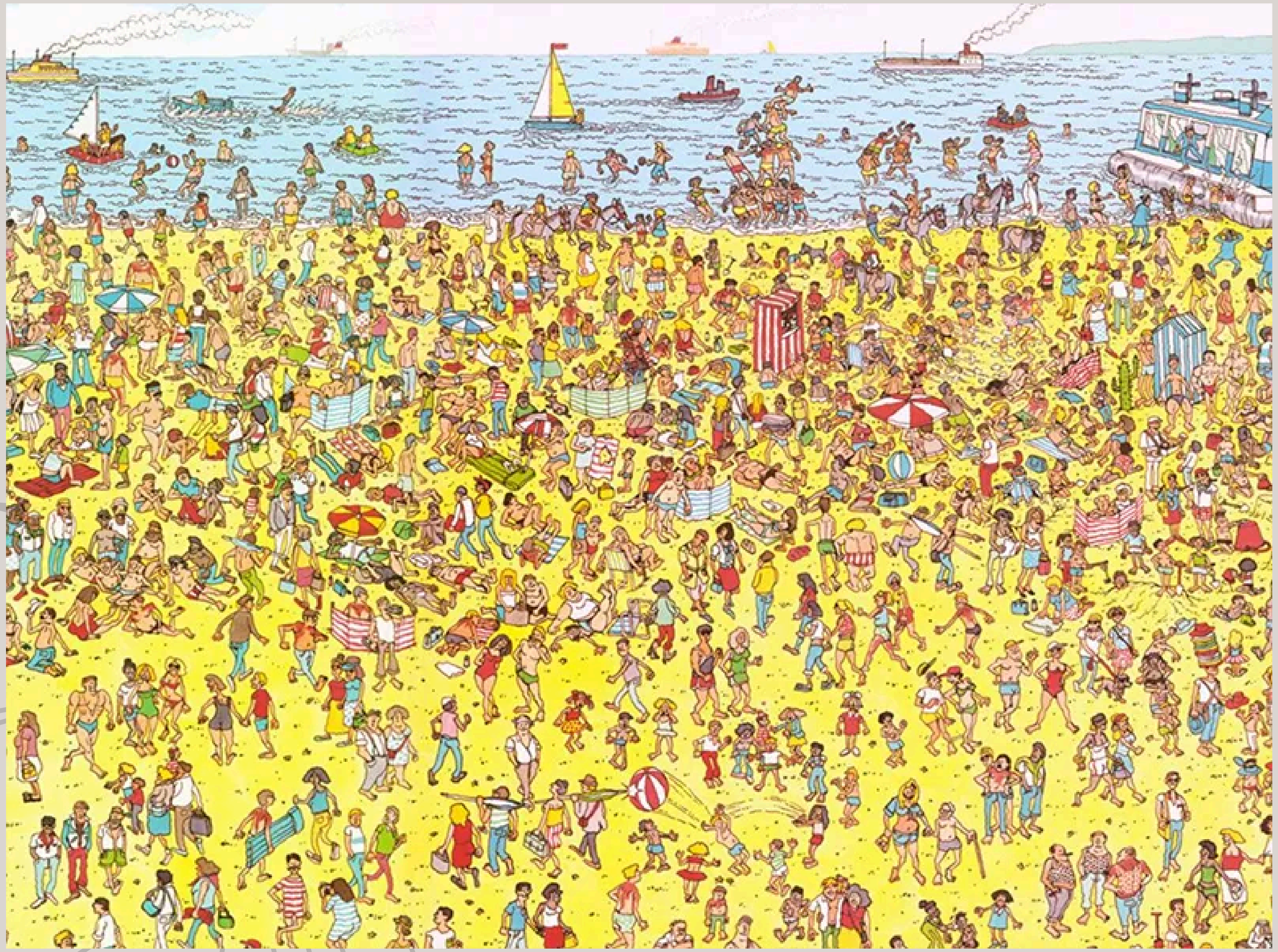
01

What is ZK used for?

02

What is ZK?

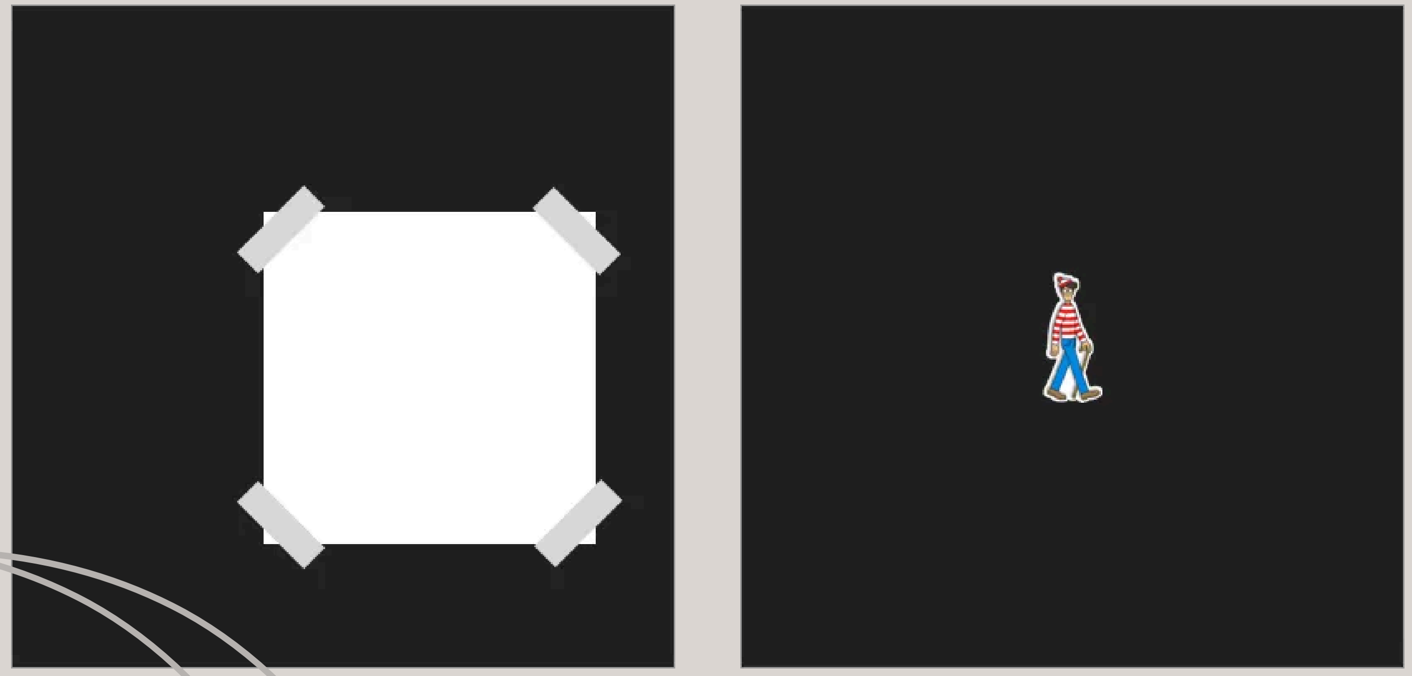
Where's Wally?

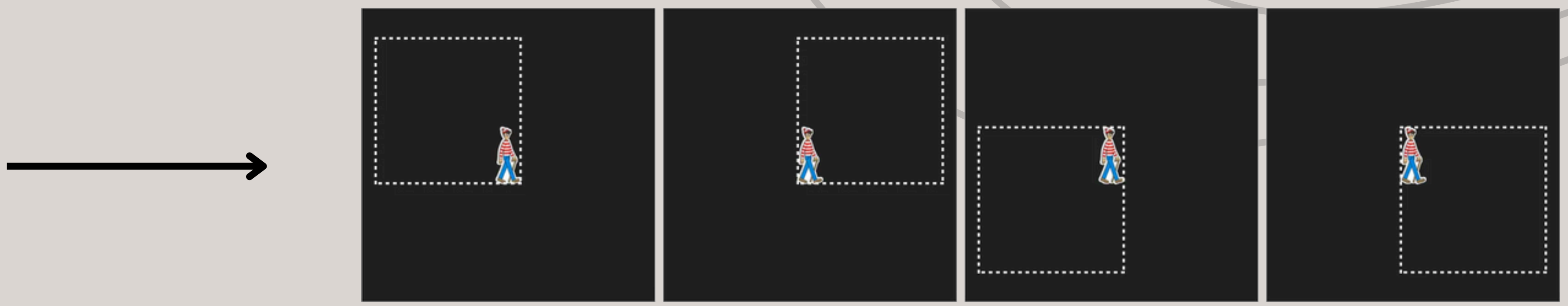


How can I prove to you I know where Wally is without giving the answer away?

Where's Wally?

Back Front





front black card > 2X size of puzzle

you only know that I know the answer,
but you still need to find Wally yourself

Photo source: https://medium.com/swlh/a-zero-knowledge-proof-for-wheres-wally-930c21e55399

So What Does ZK Really Mean?

By **ZK**, we mean

Zero Knowledge Proof System

Classical Proofs

Traditionally, proofs are:

- static statements that follow axioms and logical deductions
- checked step-by-step for correctness

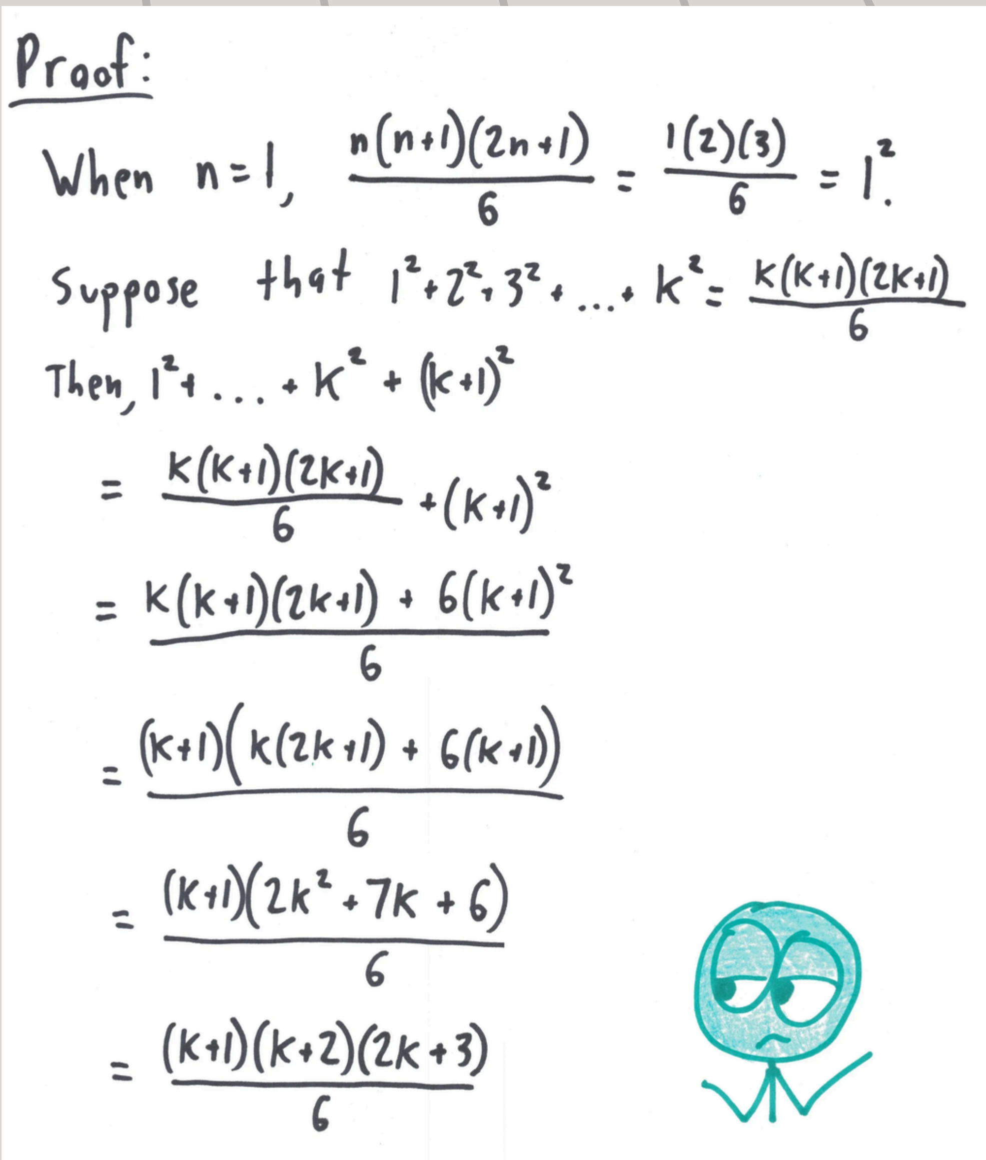
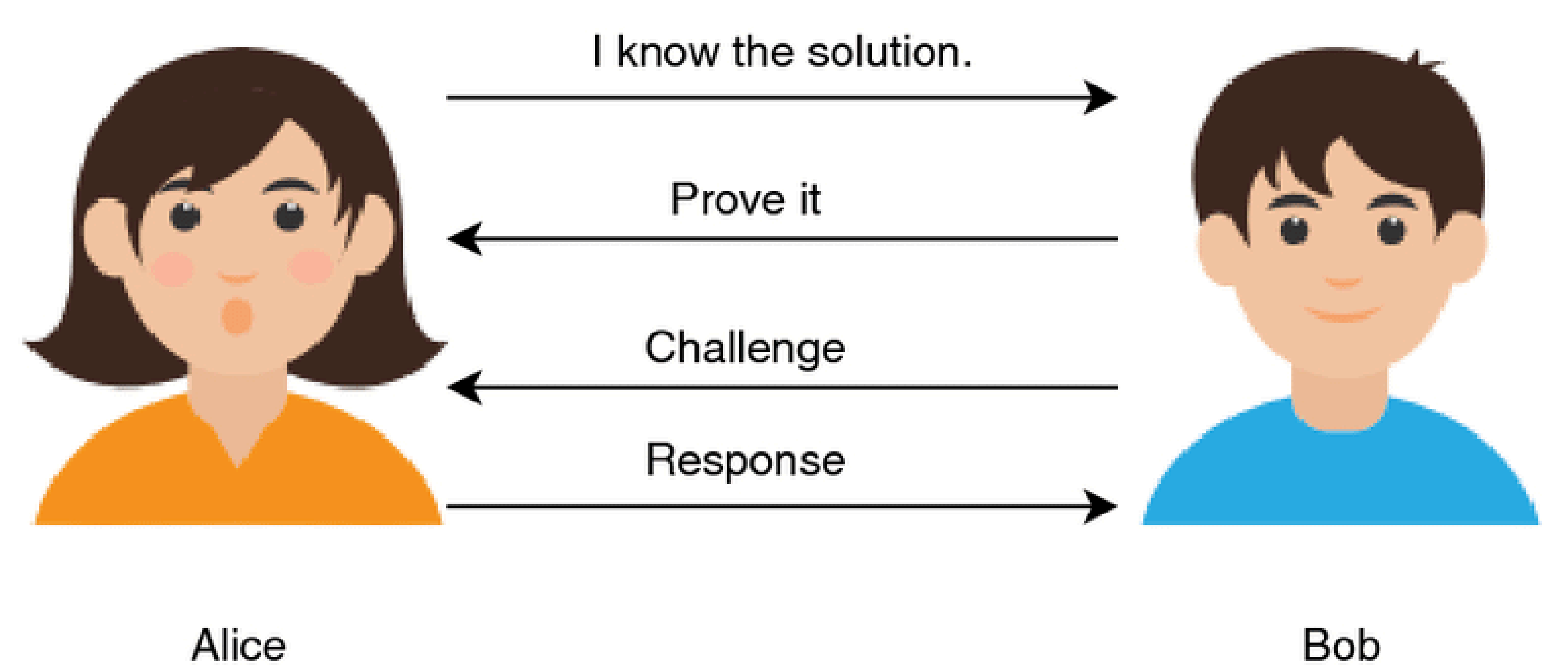


Photo source: https://medium.com/swlh/a-zero-knowledge-proof-for-wheres-wally-930c21e55399

Interactive Proofs

Proofs involve **interaction** between **Prover** and **Verifier**



How to get started?

Let's translate Where's Wally into a ZK Proof problem!

Let’s Rephrase It *(informal)*

Objective:
Alice wants to prove to Bob that she knows where Wally is, without revealing the answer

Statement s – can only be True or False:
Alice knows where Wally is in Where’s Wally Beach puzzle

Witness w – Alice’s solution:
Wally’s location (only one in this case)

Instance x – a specific problem:
Where’s Wally Beach puzzle (Wally can be hiding in other places in another puzzle)

Relation R for NP – set of ordered pairs (x, w):
R_Wally. We say “The pair (Beach, Wally’s Location) is in R_Wally”

Language L – the set of all satisfiable instances for R i.e. L(R)
Set of all Where’s Wally puzzles and their locations. i.e. S is in L(R_Wally)

Now we're ready to define our ZKP Statement

To give proof that an **instance** is in the **language** defined by some **relation**, without revealing the **witness**.

I can prove that I know the solution to this puzzle, without giving away the actual solution. You can take the proof and verify it easily.

Privacy (Zero Knowledge)

Allow the Prover to prove to the Verifier that:

1. Prover knows the solution
2. The proof doesn't reveal anything else except 1

Compression (Succinctness)

How to get **efficient verification**?

We need **short proofs**:

- short proof lengths (O(log))
- quick verification time (O(log) or O(1))

Probabilistically Checkable Proof (PCP) Theorem:
Any problem that can be verified by Classical Proof can also be verified by a special IP proofs where Verifier reads **a few random bits** from the proof

<u>Proofs</u> -> **succinct** by using *Polynomial Commitment Schemes*

But..

What's the catch?

A **tiny** chance that an **incorrect proof** passes verification.

But it's so small that we can basically **ignore** it!

What if I don't want interactivity?

There's a trick we can make IP **non-interactive** again (*Fiat-Shamir transformation*)

Some More Important Properties

Completeness:
If the proposed solution is correct, the Verifier will accept it.

i.e. if it's true, it stays true

Soundness:
If the proposed solution is incorrect, the Verifier will not accept it.

i.e. if it's false, it stays false

We now have ZKP System!

ZKP = Language + IP + Compression +
Completeness + Soundness +
Privacy (optional) +
Convert back to NI (optional but common)

Thank you

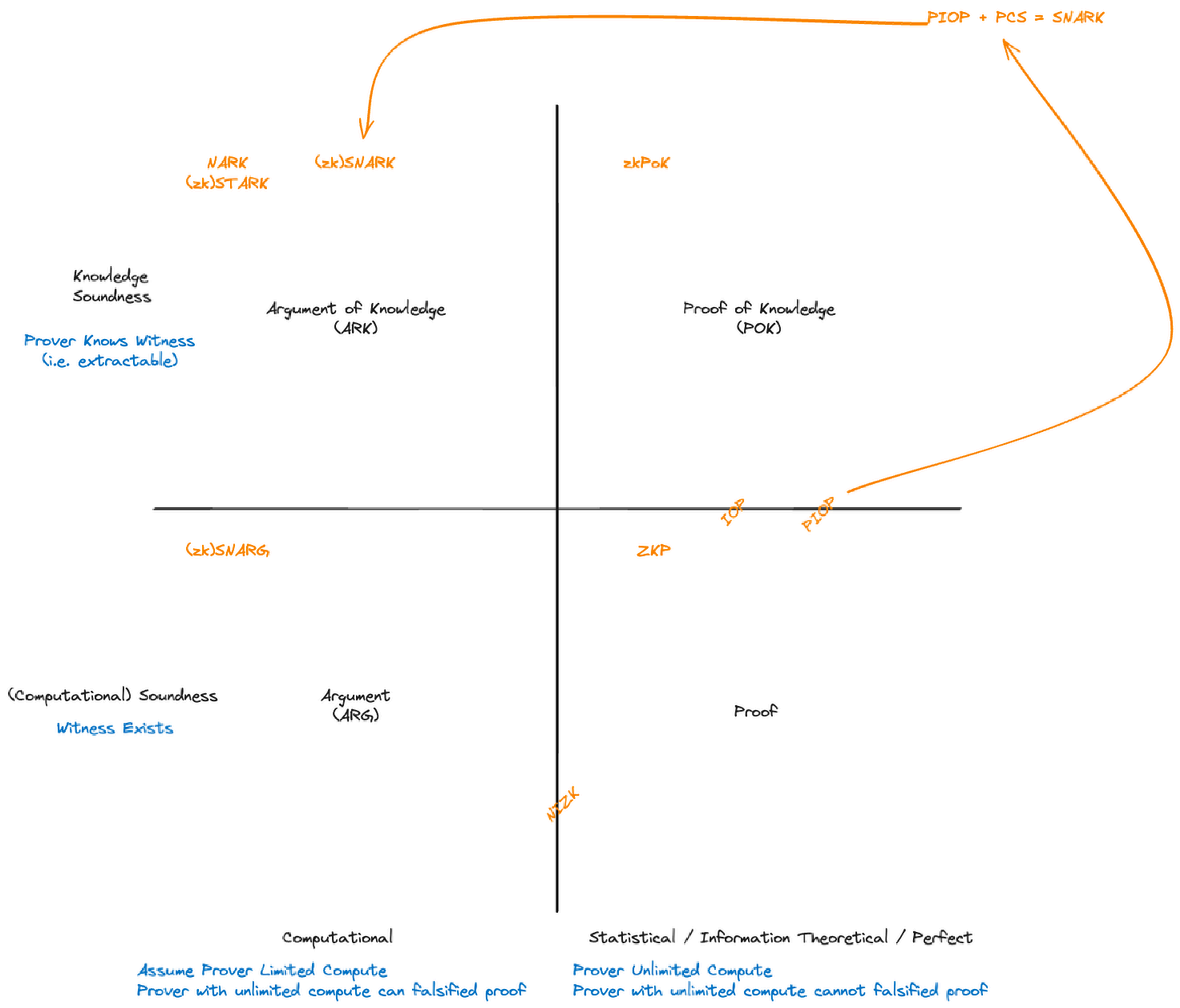
List of ZK Resources

References

1. Proofs, Arguments & Zero-Knowledge, Jul 18, 2023, Justin Thaler
2. Do You Need a Zero Knowledge Proof?, 2024, Ernstberger et al
3. ZK Market Map (Aug 8, 2024), Electric Capital, https://www.cryptomarketmap.org/zk

Appendix

Argument vs Proof vs Knowledge



Source: Nicolas Mohnblatt