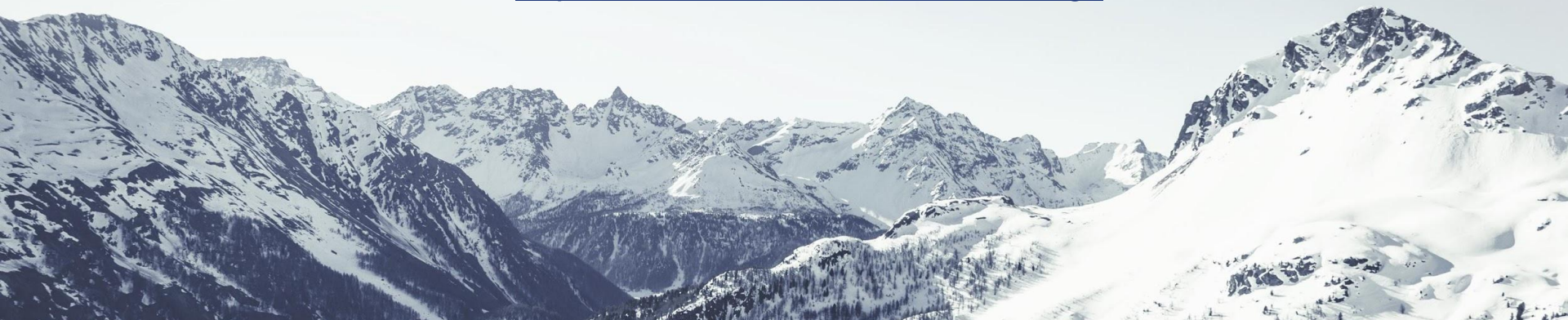


Zero knowledge proof and SNARK programming workshop

Daniel Szego

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



Tartalom



- Zero knowledge proofs
- Example - Waldo, cave of Ali Baba
- SNARK / zkSNARK
- Applications
- Conclusions
- ZK programming
-

Zero knowledge proof

"Proof" of a statement

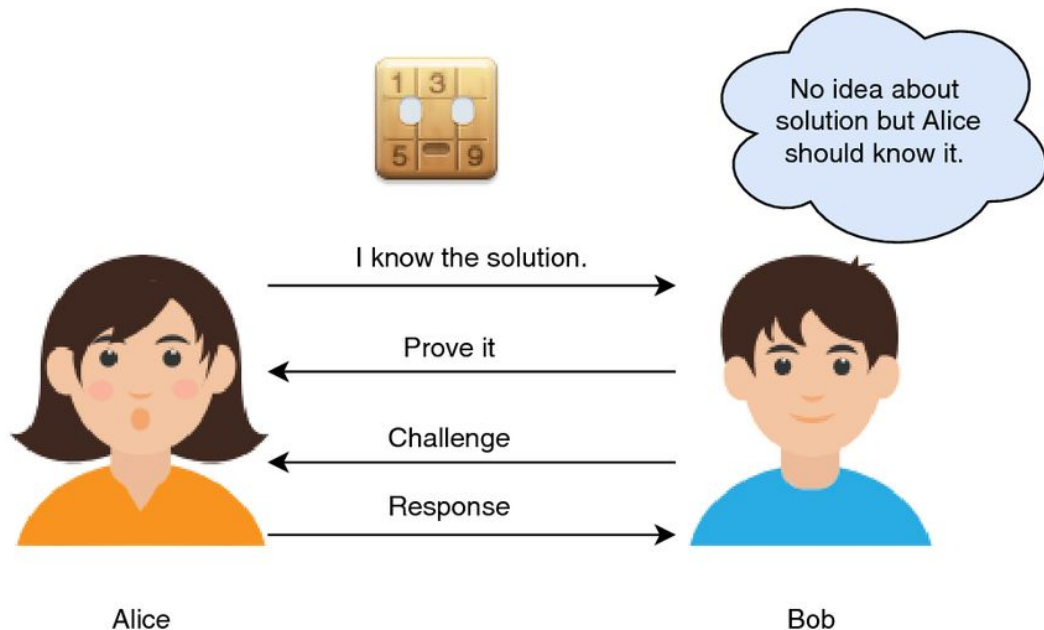
It's not a "classic" mathematical proof, it's stochastic, I know with high probability

I know some kind of secret information, I "prove" that I know without saying it

Roles:

- Prover: prover
- Verifier: verifier, validator

Interactive / non-interactive



Example - Waldo



Example - Cave of Ali Baba

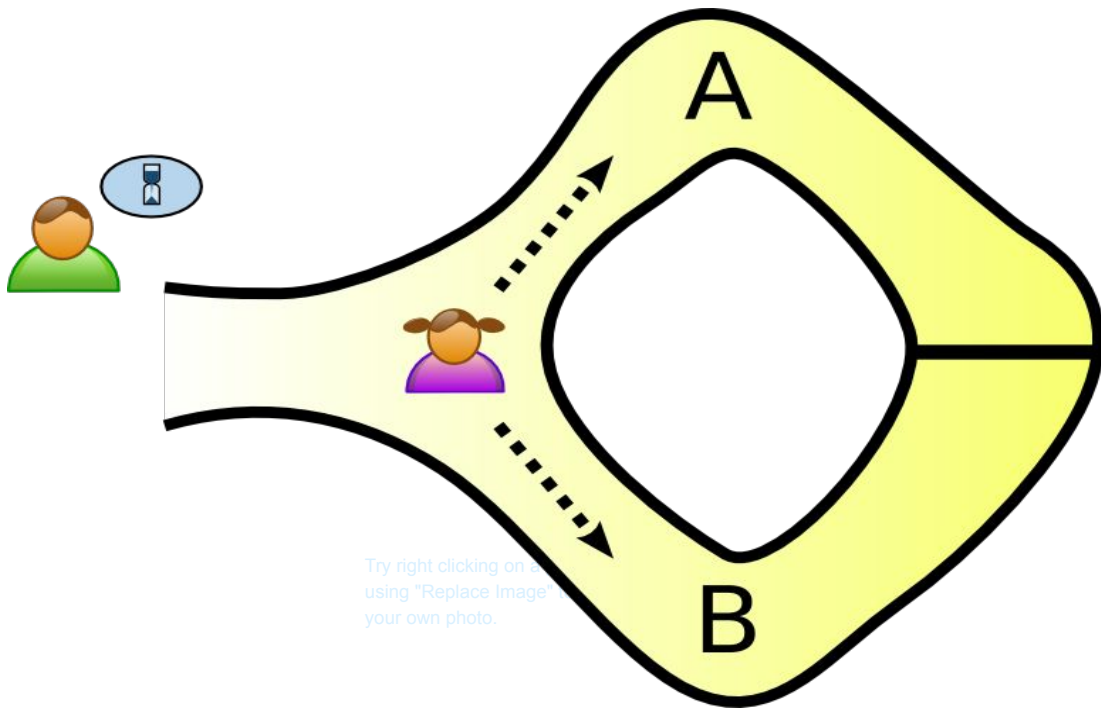
Alice and Bob

Alibaba's cave, which is closed in the middle with a door that opens with a "magic word".

Alice enters the cave and chooses a side to enter.

Bob goes into the cave and decides which side Alice should come back from.

He can cheat 50% of the time -> when repeated many times, he approaches 0%



SNARK / zkSNARK

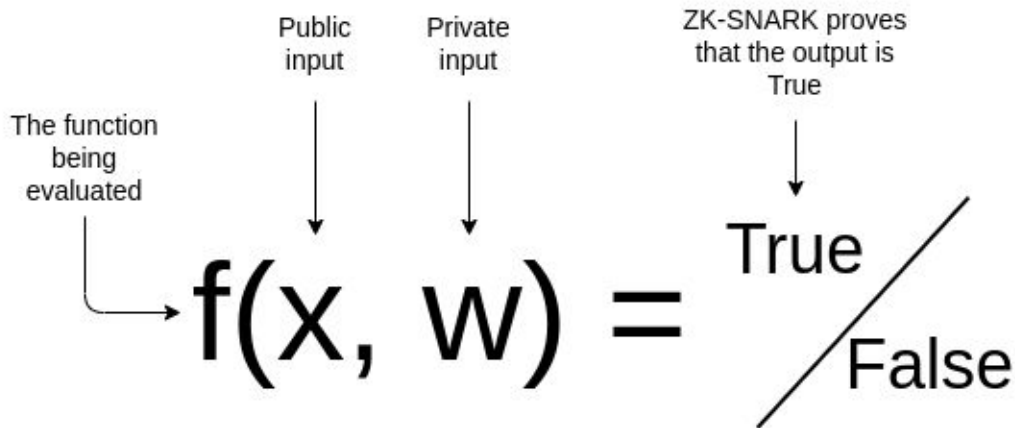
(zk) SNARK

Succinct: short, concise proof

Non-Interactive: there is no interaction, the prover produces it and sends it to the verifier.

Argument of Knowledge: Some information that the prover knows.

Zero-Knowledge: None of the private information reaches the validator.



Application - Authenticated but edited photos

Fake pictures, articles,
information online.

Camera with digital signature,
location and time information.

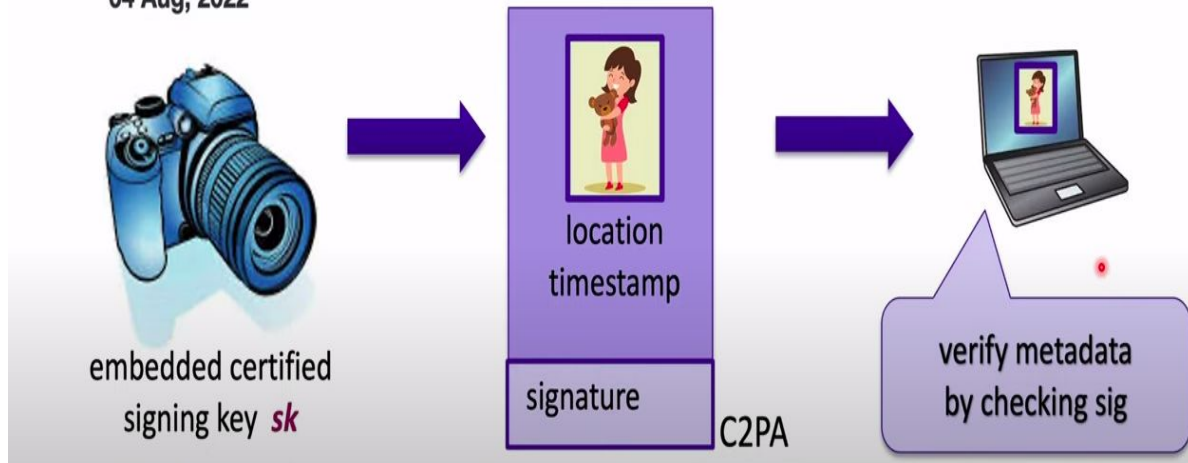
Image transformation, resizing,
cropping, sharpening, changing
the color palette, etc.

The digital signature is not valid
after transformation

Digital signature + transformation
+ SNARK

Sony Unlocks In-Camera Forgery-Proof Technology

04 Aug, 2022



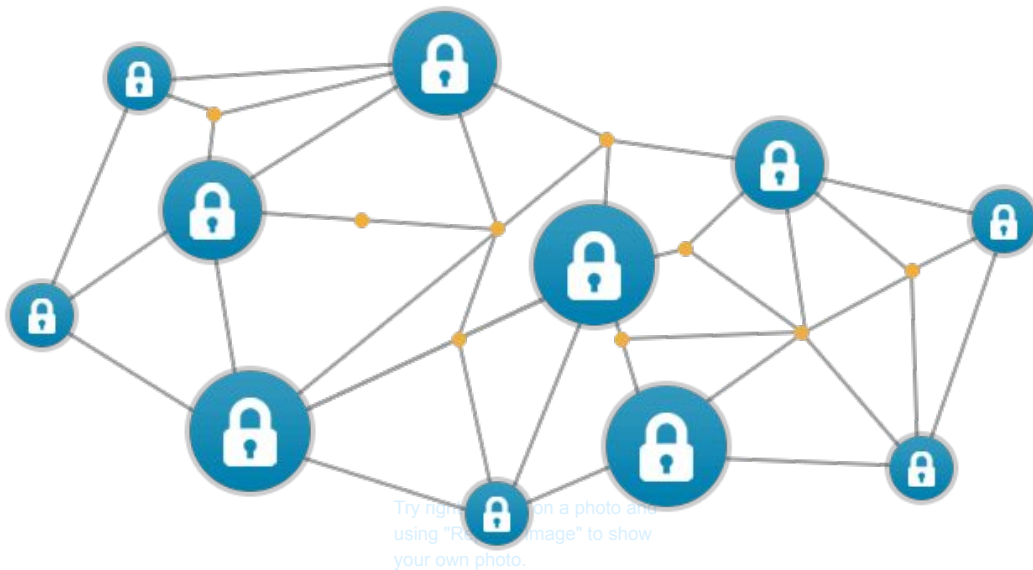
Application - Private blockchain transaction

Blockchain: independent nodes that authenticate transactions.

To authenticate, the node must see the transaction.

Privacy issues.

Instead of full transactions, transactions with secret input and zkSNARK proof.



Application - Rollup

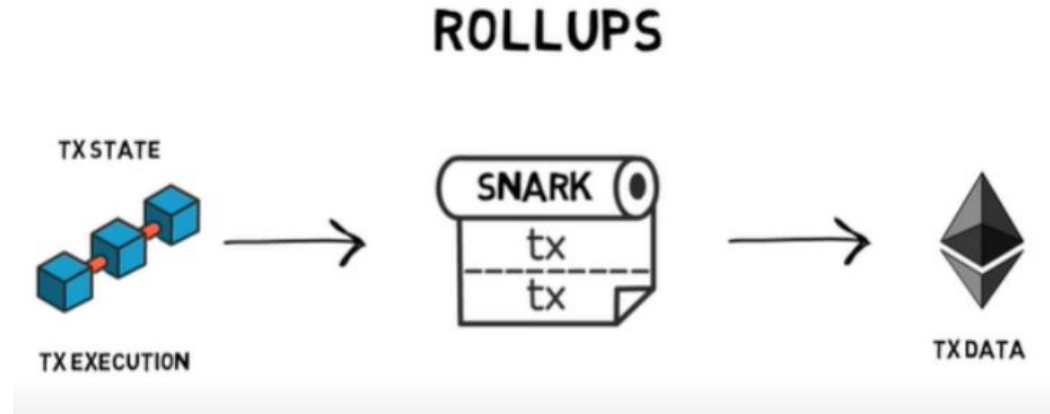
Scalability problem: Most blockchains can perform few transactions and that too slowly.

Reason: independent validators.

Batch execution of transactions outside the blockchain..

The problem is reliability, correct execution, exclusion of fraud.

SNARK / zkSNARK proof of correct execution



your own photo.

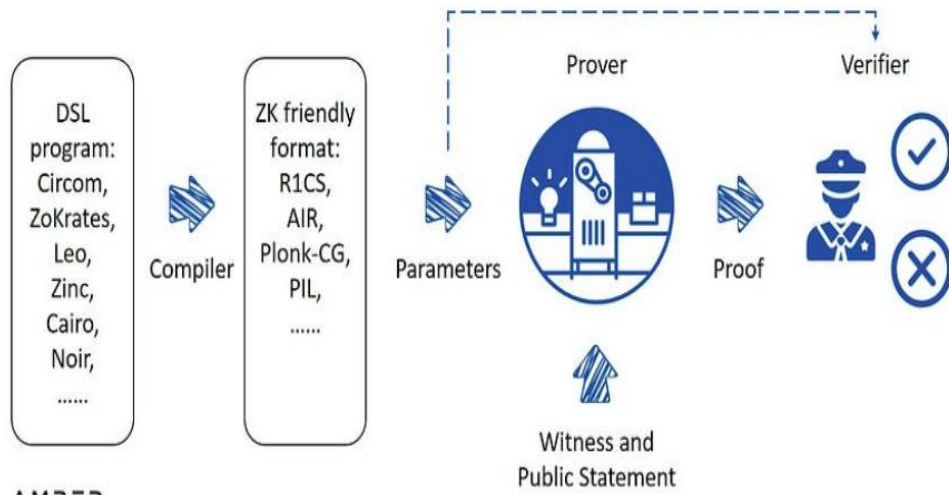
ZKP programming considerations

It is an area that requires serious theoretical preparation.

Especially mathematicians.

DSL (domain specific language) simplified, domain specific languages, simpler (!) for zero knowledge and SNARK programming.

However, simplified programming and cryptographic basics are still necessary.



Conclusions

An exciting emerging field,

Dynamic development both on the research - theoretical side and on the application side.

Min. 15 years of active development.

Tools and frameworks are also available on the engineering / programming side.

The area can be at least as important as the digital signature.

Zero Knowledge Proofs





Zero knowledge programming



ZK programming - engineering flow

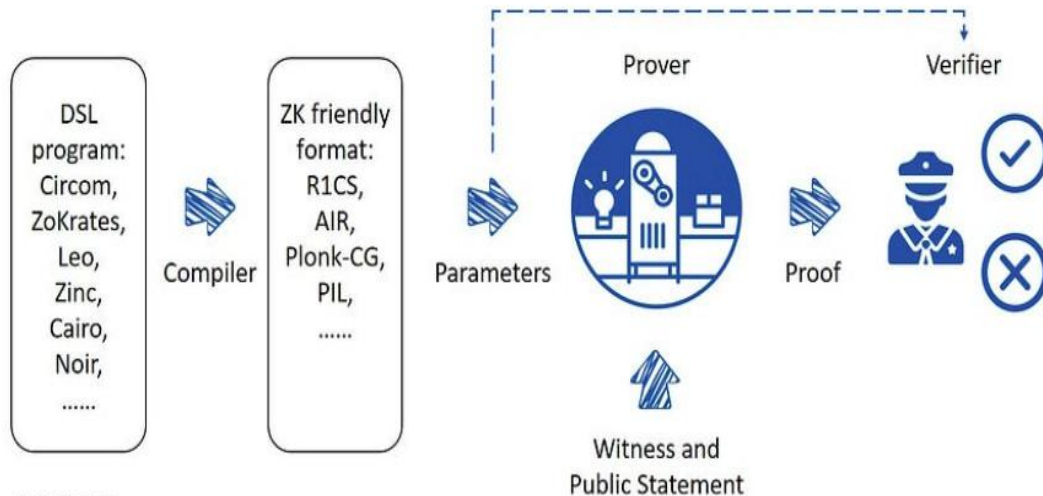
Domain specific languages for SNARK
or zkSNARK programming

Abstracting away some of the
mathematical and theoretical
complexity

ZK and SNARK programming without
cryptographic knowledge ? Not yet :)

Compilation to mathematical
representation, R1CS

Complex development frameworks,
compile, test, prover, verifier module
integrations.



Core algorithm consideration



Under very active development

Proof size

Verification time

Setup :

- per circuit
- universal
- transparent

Post quantum readiness

	size of proof π	verifier time	setup	post-quantum?
Groth'16	≈ 200 Bytes $O_\lambda(1)$	≈ 1.5 ms $O_\lambda(1)$	trusted per circuit	no
Plonk / Marlin	≈ 400 Bytes $O_\lambda(1)$	≈ 3 ms $O_\lambda(1)$	universal trusted setup	no
Bulletproofs	≈ 1.5 KB $O_\lambda(\log C)$	≈ 3 sec $O_\lambda(C)$	transparent	no
STARK	≈ 100 KB $O_\lambda(\log^2 C)$	≈ 10 ms $O_\lambda(\log^2 C)$	transparent	yes

using "Replace Image" to show your own photo.

DSL language and tool selection



Core algorithm consideration






Imperative / description / circuit languages

Different base programming language

Different programming language and framework integration modules

Technological life cycle: all are early stage, but:

- Successful productive usage
- Stable releases

Language	Team
Noir	 Aztec
SnarkyJS	O(1) Labs
Leo	 Aleo
Circom	 iden3
Cairo	 STARKWARE
Lurk	

Circom

DLS / circuit programming language
and development environment for
arithmetic circuits and constraints

Used e.g. in tornado cash

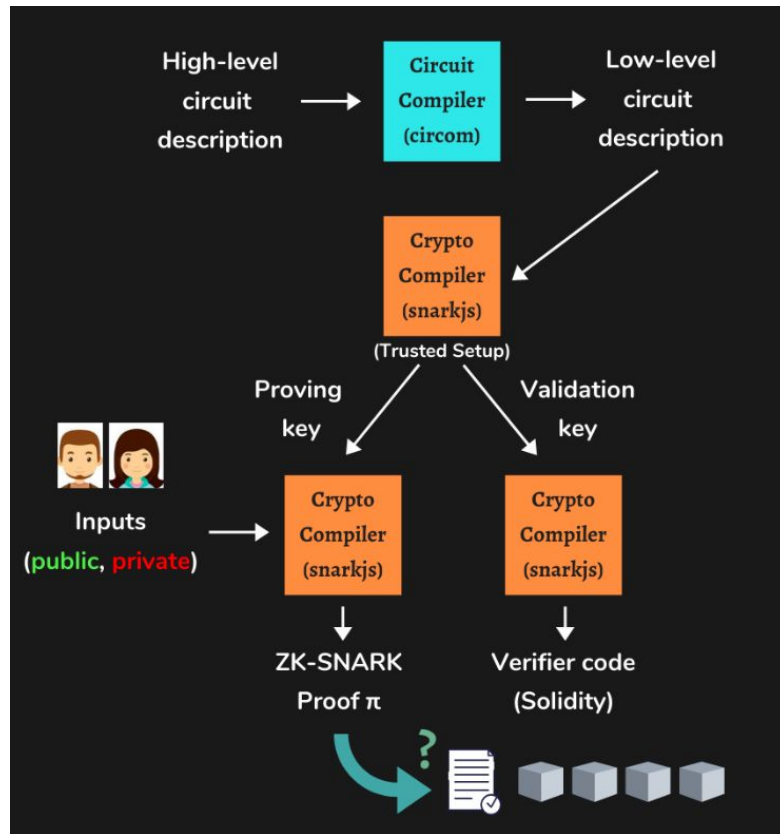
Supports:

- Groth16
- Plonk

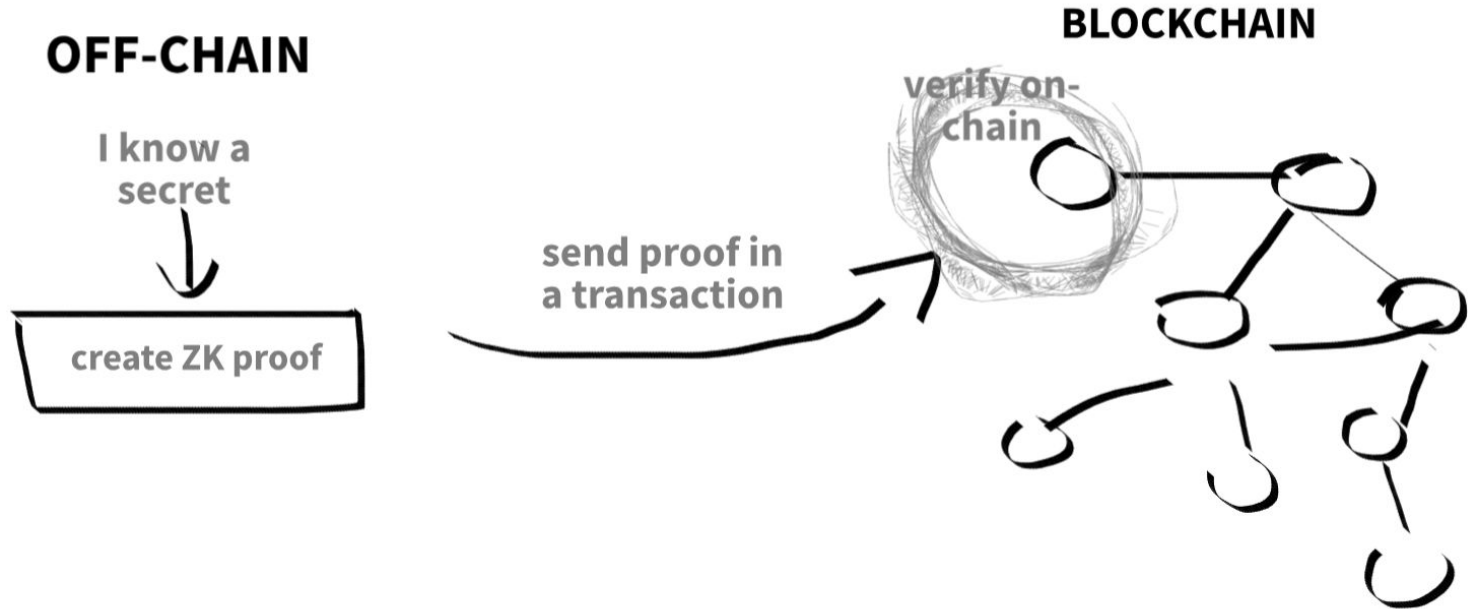
Well established (exist 3 years :)

Supported integration:

- javascript (snarkjs)
- cpp
- solidity verifier



Demo





Q & A

Daniel Szego

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

