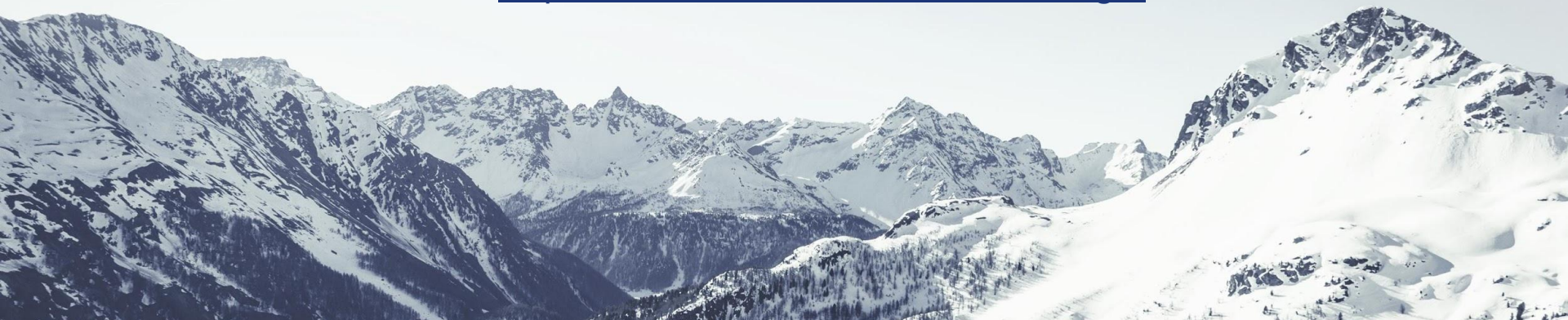
Tokenized Zero Knowledge Machine Learning and Its Applications

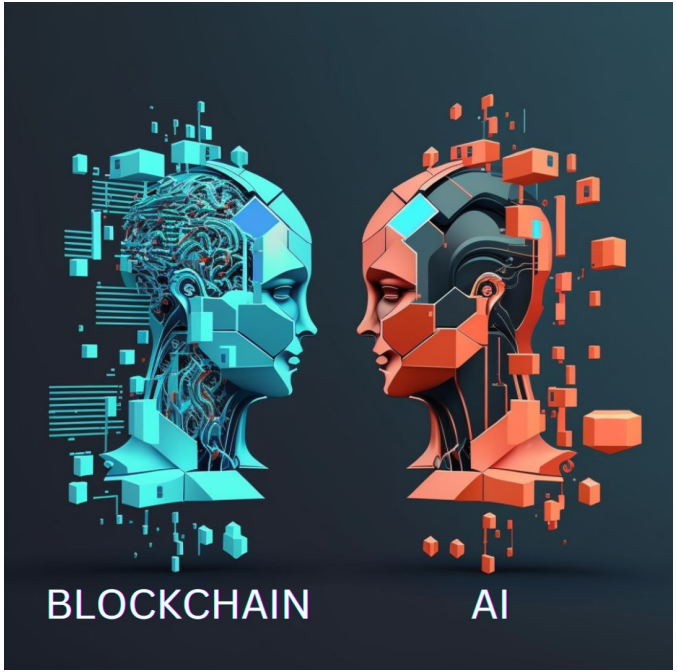
Daniel Szego
In: https://www.linkedin.com/in/daniel-szego/



Blockchain and AI convergence

Interesting possibilities for merging the two field:

- Blockchain data analysis
- Transaction graph analysis
- Crypto asset price prediction
- De-anonymizing user transactions
- Decentralized marketplaces of AI services
- Decentralized marketplaces of machine learning services
- AGI (Artificial General Intelligence) initiatives, e.g. SingularityNET
- Decentralized machine learning coordination (Neureal)
- Tokenized Intellectual Property (IP) of AI generated data



Trustless machine learning on-chain

Executing machine learning on-chain.

Importing machine learning results on-chain in a trustless way.

Application areas:

- Privacy preserving credit scores for lending protocols
- Private but trustless KYC processes
- Improved stablecoin protocols, predicting stablecoin rates
- Improved DeFi use-cases, like AMM intelligent pricing
- Private machine learning models with tokenized IP (intellectual property) protection
- On-chain verifiable trading, e.g. trading bot.
-

Zero knowledge proof

"Proof" of a statement

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

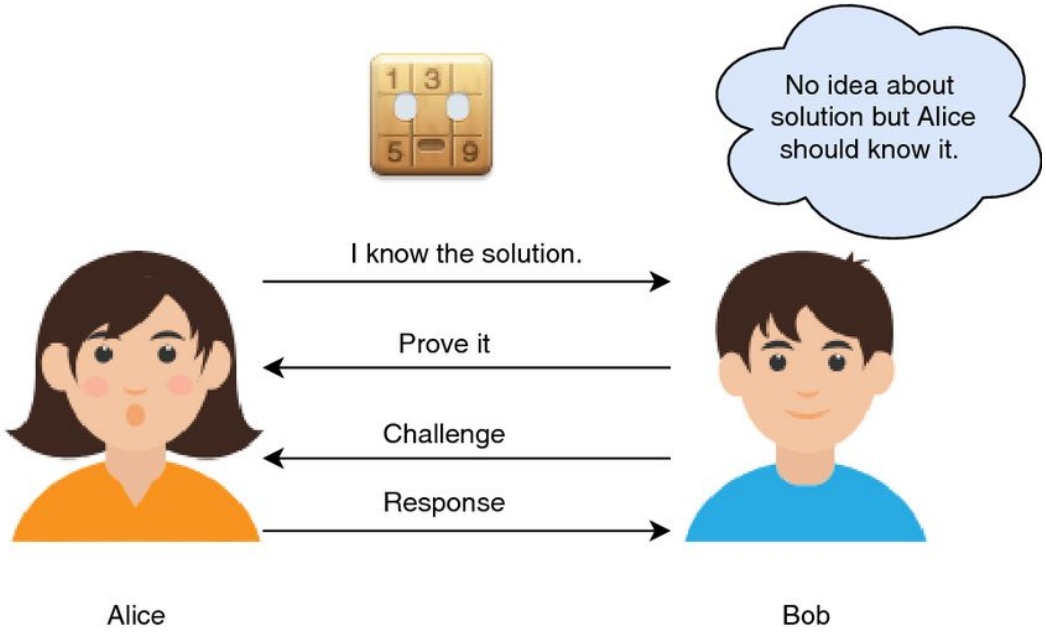
I know some kind of secret information, I "prove" that I know without saying it

Roles:

- Prover: prover

- Verifier: verifier, validator

Interactive / non-interactive



SNARK / zkSNARK

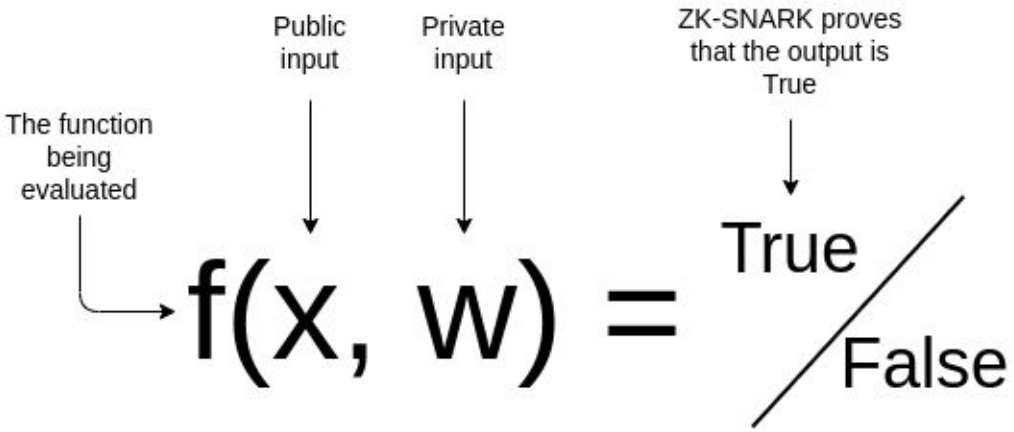
(zk) SNARK

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.



ZK programming - engineering flow

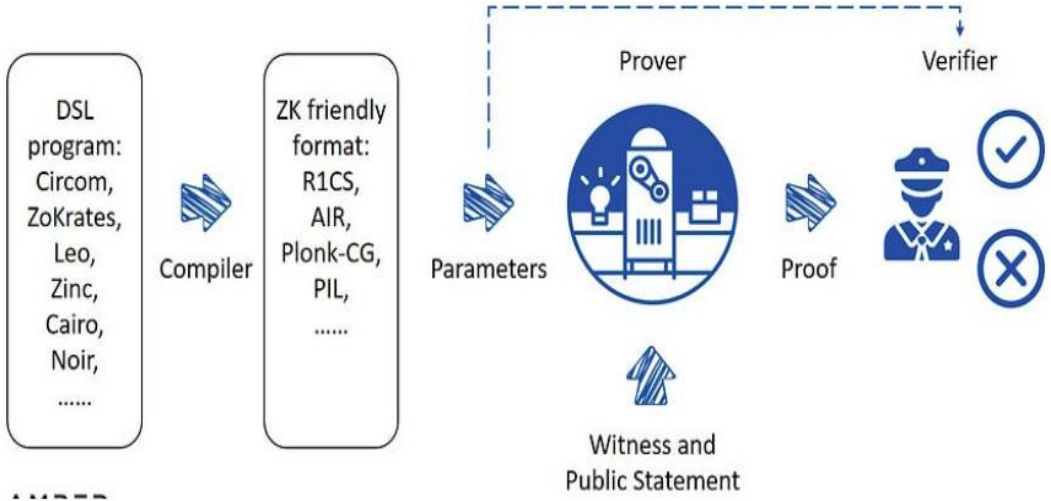
Domain specific languages for SNARK
***zk*SNARK programming**

Abstracting away some of the mathematical and theoretical complexit

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

Compilation to mathematical representation, R1CS

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



zkML - zero knowledge machine learning

Machine Learning:

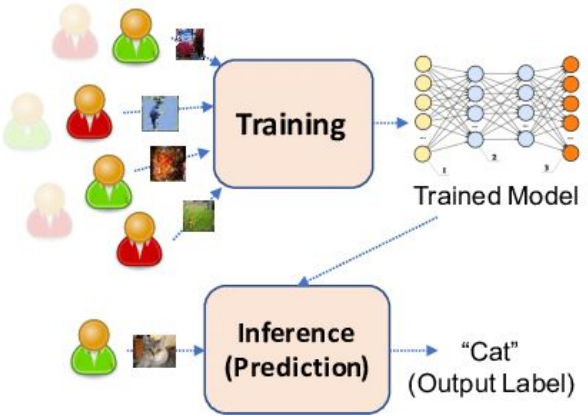
- learning phase:
  - supervised / non-supervised learning
- inference: input out association based on the trained model

ML + zkSNARK:

- Trained machine learning model off-chain
- Inference phase is supported by zero knowledge proofs
- Proofs and the results can be validated on-chain

zkML models (mostly inference):

- public / private : trained model / input / output



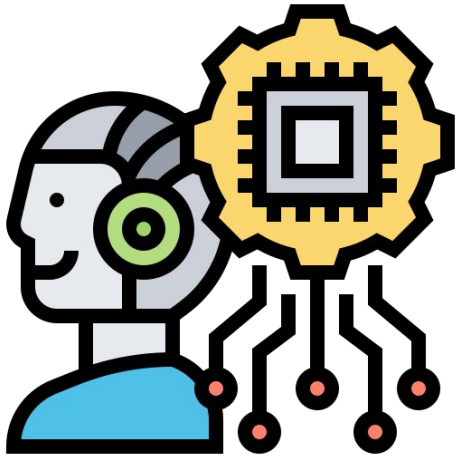
opML - optimistic machine learning

Ensuring the result / correctness of the machine learning with economic incentives.

Training / inference is executed off-chain and the result is imported on-chain.

On-chain result is verified by a verification game:

- On-chain result is “temporel” and can be “verified” by different actors.
- Verification and trustless dispute mechanism.
- Different off-chain verifiers
- Economic incentive to disclose false inference, e.g. token
- Similar to optimistic rollup
- Performance / privacy



Link: https://arxiv.org/abs/2401.17555

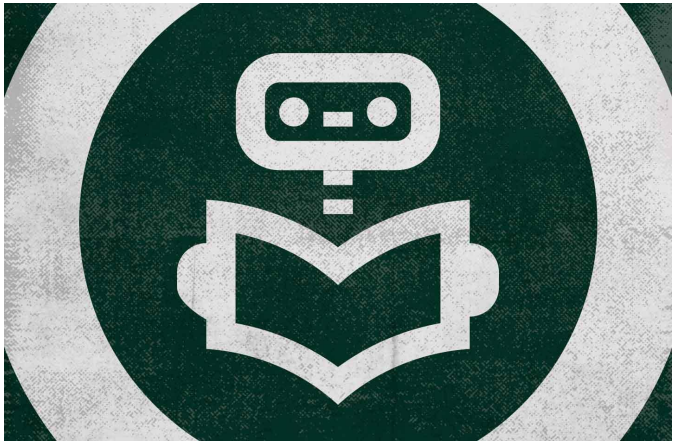
zkML - Private Input, Public Model

Inference phase is supported by zero knowledge proofs

Public parameters of the trained model

Private input for the on-chain output, but there must be a proof that the output is based on the private input and trained model:

0. commit parameters of the machine learning model publicly.
1. commit private input hash, *hash(x)*
2. give a snark proof that *hash(x)* was committed into the ledger and the output is if we apply the machine learning to x.



E.g. credit scoring

Link: https://0xparc.org/blog/zk-mnist

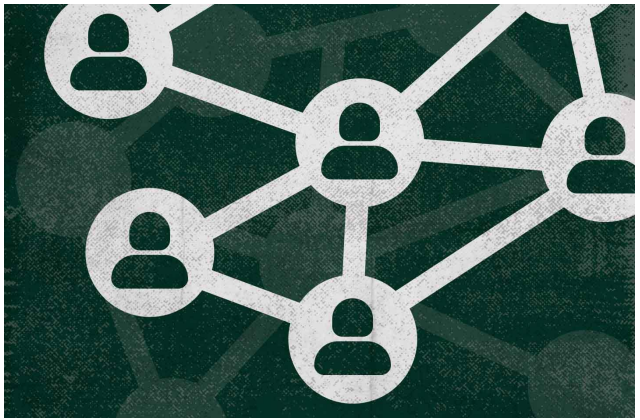
zkML - Private Model, Public Input

The trained machine learning must be kept secret.

It is industry standard or under Intellectual Property (IP) rights

It must be proven that the same model used for different computations:

0. Commit hash of the parameters of the private machine learning model, like *hash(params)*
1. Give a *zk*SNARK proof that the *x* input produces a certain o output by applying the machine learning model which parameters were committed into as *hash(params)*



Link: https://0xparc.org/blog/zk-mnist

zkML - Other scenarios

Private input, private model:

- The input is confidential and model is private as well (like protected by IP).
- E.g. healthcare applications
- Complex: Compositional ZK or multiparty computation

Public input, public model:

- Computational intensive models
- Succinct proof on the computation

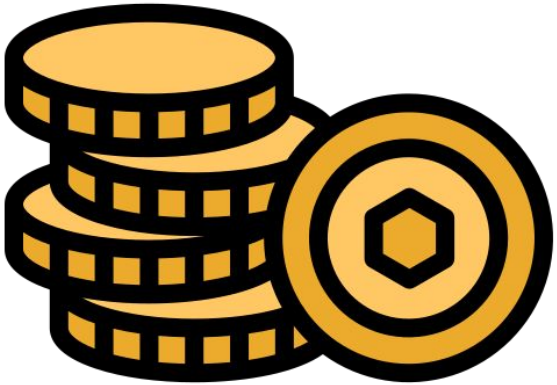
Proof of training

Link:
https://medium.com/1kxnetwork/zkml-evolving-the-intelligence-of-smart-contracts-through-zero-knowledge-cryptography-e6725412bbd1

Tokenized zkML

Tokenization:

- cryptographic ownership
- trade of ownership
- ownership of model, like IP or service level agreement
- ownership of generated data,
  - like AI generated content
  - AI generated digital art
- improves market efficiency
- making markets more transparent
- accelerates market dynamics



ERC7007 token standard

ERC7007: Token standard for AI generated NFT,

AI-generated content (AIGC) + ERC721 NFT standard

Proof (+ ownership) if the certain NFT was generated by a certain ML model and input (prompt).

0. User claims prompt, published hash of input and publishes the output.
1. ZK proof is generated that the output is the based on the prompt (input) and the trained machine learning model (inference)
2. Verifier can verify the output and the hash of the input and the ZK proof
3. At successful verification, the user will own the input

Link: https://eips.ethereum.org/EIPS/eip-7007

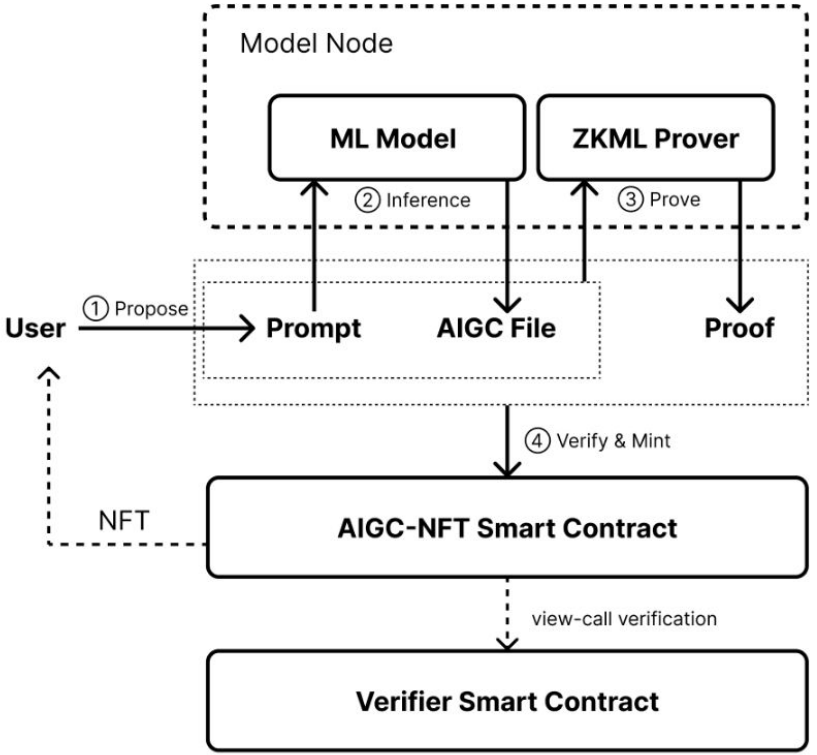
ERC7007 Example architecture

ML model - contains weights of a pre-trained model; given an inference input, generates the output

zkML prover - given an inference task with input and output, generates a ZK proof

AIGC-NFT smart contract - contract compliant with this proposal, with full ERC-721 functionalities

Verifier smart contract - implements a verify function, given an inference task and its ZK proof, returns the verification result as a boolean



Link: https://github.com/ethereum/ERCs/tree/master/assets/erc-7007/contracts

Demo

zkML Demo:

https://zkmnist.netlify.app/

ERC7007 specification:

https://eips.ethereum.org/EIPS/eip-7007

ERC7007 test implementations:

https://github.com/ethereum/ERCs/tree/master/assets/erc-7007/contracts

Conclusions

AI and blockchain mixed applications

Bringing machine learning on-chain

Interesting application areas

Tokenization has market potential (cryptographic ownership)

Tokenization provides the possibility for better dAPP integration

DEFI + machine learning can have especially interesting areas and the possibility for new decentralized applications

Q & A

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