Zero knowledge design patterns for Hyperledger Fabric

Daniel Szego

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

Agenda

- *Zero knowledge proofs introduction*
- *Hyperledger Fabric*
- *Private signer*
- *Confidential transaction*
- *Mixer*
- *Off-chain transaction*
- *zkRollup*
- *Exchange: Off-chain order book matching*
- *Other use-cases*
- *Hunting for the SNARK*
- *Q&A and discussion*

Zero knowledge proof

"Proof" of a statement

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

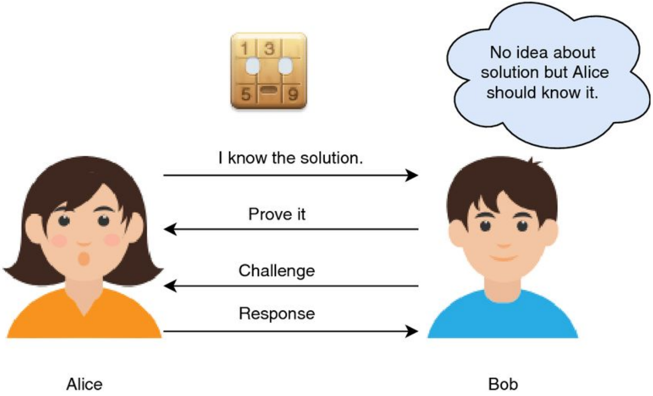
I know some kind of secret information, I "prove" that I know without saying it

Roles:

- Prover: prover

- Verifier: verifier, validator

Interactive / non-interactive



SNARK / zkSNARK

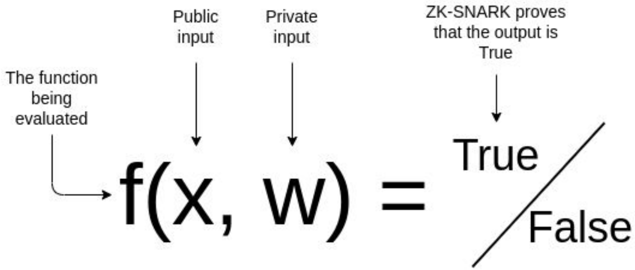
(zk) SNARK

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.


Public input
Private input
ZK-SNARK proves that the output is True
The function being evaluated
f(x, w) =
True
False

ZK programming - engineering flow

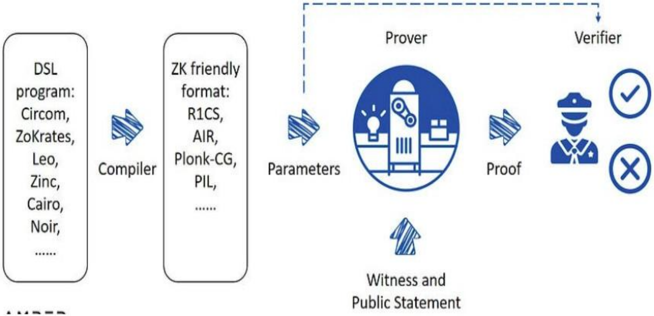
Domain specific languages for SNARK or *zk*SNARK programming

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.


DSL program: Circom, ZoKrates, Leo, Zinc, Cairo, Noir, ...
Compiler
ZK friendly format: R1CS, AIR, Plonk-CG, PIL, ...
Parameters
Prover
Proof
Verifier
Witness and Public Statement

Core algorithm consideration

Under very active development

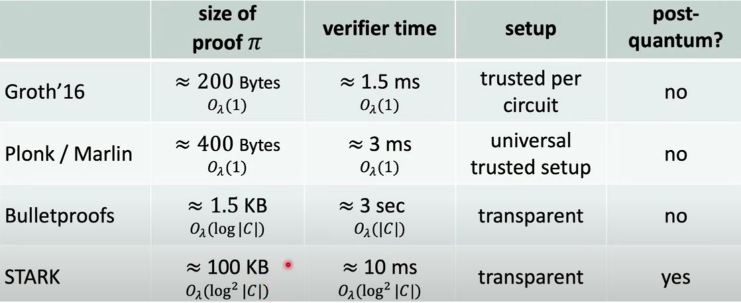
Proof size

Verification time

Setup :

- per circuit
- universal
- transparent

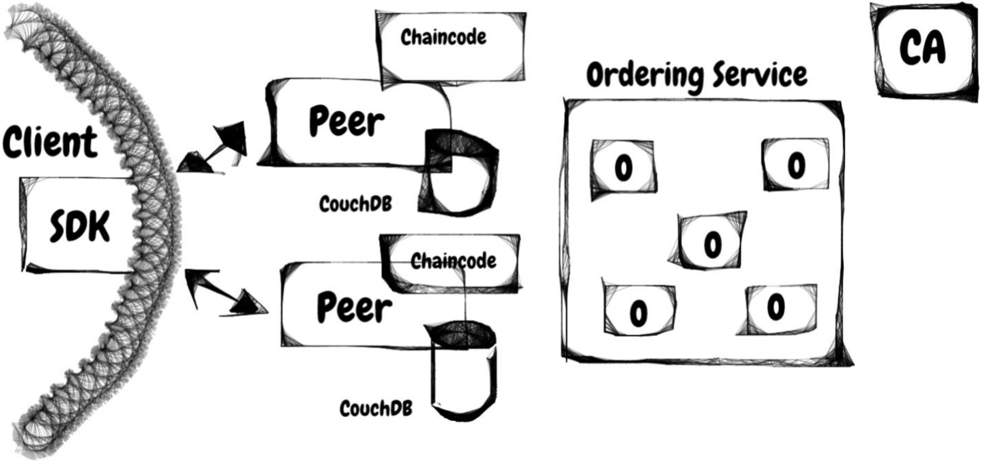
Post quantum readiness



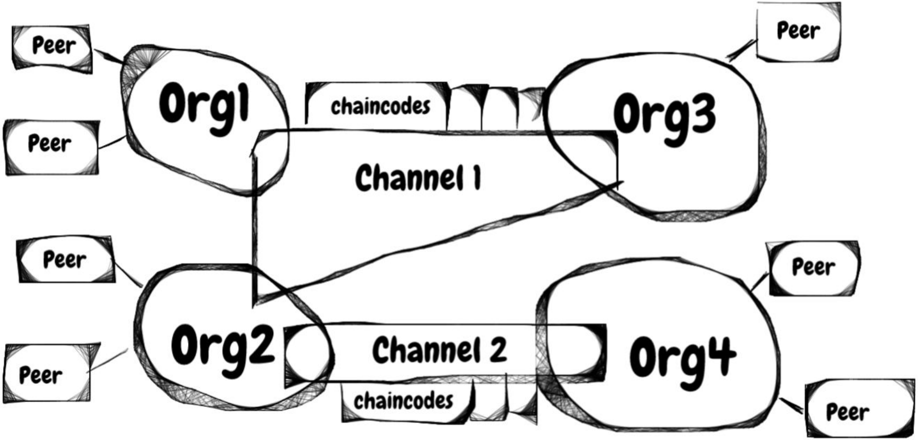
	size of proof π	verifier time	setup	post-quantum?
Groth'16	≈ 200 Bytes $O_\lambda(1)$	≈ 1.5 ms $O_\lambda(1)$	trusted per circuit	no
Plonk / Marlin	≈ 400 Bytes $O_\lambda(1)$	≈ 3 ms $O_\lambda(1)$	universal trusted setup	no
Bulletproofs	≈ 1.5 KB $O_\lambda(\log\|C\|)$	≈ 3 sec $O_\lambda(\|C\|)$	transparent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

Hyperledger Fabric Physical Architecture



Hyperledger Fabric Logical Architecture



Private signer

Instead of X509, verifiable credential / zero knowledge proof based authentication

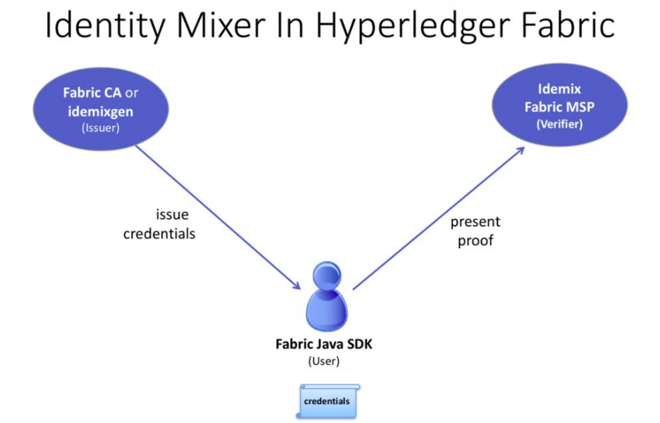
Limitations:

No endorsement

Fix set of attributes:

- ou : revealed
- role: revealed
- enrollment id: hidden
- revocation handle: hidden

No revocation


Identity Mixer In Hyperledger Fabric
Fabric CA or idemixgen (Issuer)
Idemix Fabric MSP (Verifier)
issue credentials
present proof
Fabric Java SDK (User)
credentials

Idemix: [*https://hyperledger-fabric.readthedocs.io/en/release-2.5/idemix.html*](https://hyperledger-fabric.readthedocs.io/en/release-2.5/idemix.html)

Confidential transaction (Privacy)

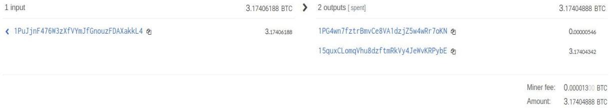
UTXO based ledger

Hyperledger UTXO example Fabric samples: https://github.com/hyperledger/fabric-samples/tree/main/token-utxo

Hiding amounts:

- Instead of amounts, cryptographic commitments for the amounts (hiding amounts)
- Verification: zK proof for the correctness of the transaction: *Correctness of commitments* *Sum of amounts*
- Transaction + zK Proof

Efficiency : general SNARK vs Pedersen Commitment



1 input 3.17406188 BTC › 2 outputs [spent] 3.17404888 BTC

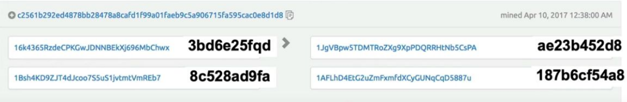
‹ 1PuJjnF476W3zXfVYmJfGnouzFDAXakkL4 3.17406188

1PG4wn7fztrBmvCe8VA1dzjZ5w4wRr7oKN 0.00000546

15quxCLomqVhu8dzftmRkVy4JeWvKRPybE 3.17404342

Miner fee: 0.00001300 BTC

Amount: 3.17404888 BTC



c2561b292ed4878bb28478a8cafd1f99a01faeb9c5a906715fa595cac0e8d1d8 mined Apr 10, 2017 12:38:00 AM

16k4365RzdeCPKGwJDNNBEkXj696MbChwx 3bd6e25fqd › 1JgVBpw5TDMTRoZXg9XpPDQRRHtNb5CsPA ae23b452d8

1Bsh4KD9ZJT4dJcoo7S5uS1jvtmtVmREb7 8c528ad9fa 1AFLhD4EtG2uZmFxmfdXCyGUNqCqD5887u 187b6cf54a8

FabZK: https://ieeexplore.ieee.org/document/8809514

Mixer (Privacy)

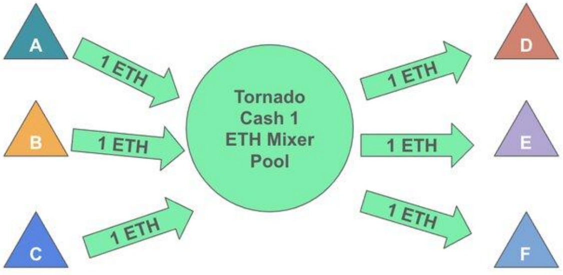
Hiding source and destination of funds.

Creating a pool of funds for several people (accounts) -

Centralized / decentralized (on-chain) mixers

Real privacy preserving on-chain mixer is challenging.

Tornado Cash:
- Pseudonymised funds in a merkle tree
- Secrets for each fund: “ownership”
- Withdraw by giving a ZK proof on the secret of “ownership” of a fund



Tornado Cash Privacy Solution: https://berkeley-defi.github.io/assets/material/Tornado%20Cash%20Whitepaper.pdf

zkML - zero knowledge machine learning

Off-chain machine learning or model evaluation and importing the result on-chain. E.g. credit rating of a customer.

Machine Learning:

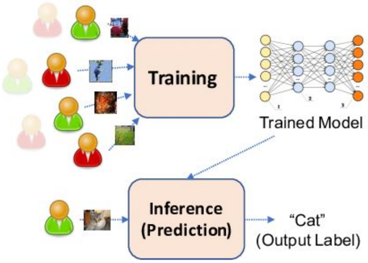
- learning phase:
- supervised / non-supervised learning
- inference: input out association based on the trained model

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



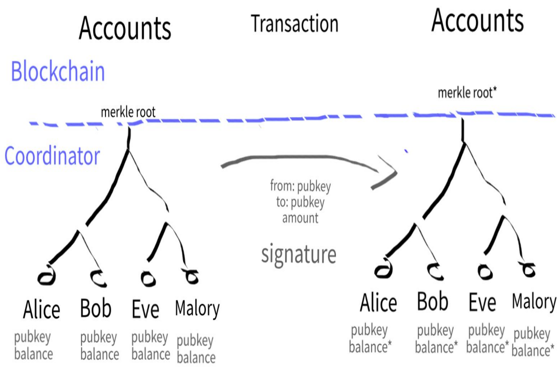
Link: https://0xparc.org/blog/zk-mnist

Off-chain transaction (Scaling)

Payment or token transfer transactions

Accounts are represented as leafs in the merkle tree.

Transaction validation steps (simplified):
- Transaction : send <from> , <to>, <amount>
- check if <from> and <to> are on the account tree, which merkle root is already in the blockchain
- check if <from> has enough balance
- debit <from> with <amount>
- credit <to> with the <amount>
- calculate new account tree and merkle root for it
- create zk proof about the calculation
- publish new root + zkProof to the ledger
- if the proof is valid, record new root to in the ledger



ZK rollup blog: https://github.com/tanpx12/zk-rollup-tutorial

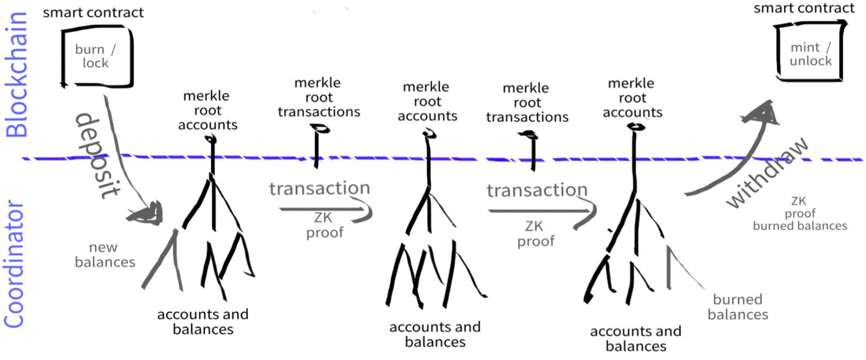
Rollup: off-chain transactions (scaling)

Off-chain transactions

Payment / token transfer

Transactions are executed in batch / block by a coordinator

Deposit and withdraw transactions



ZK rollup blog: https://github.com/tanpx12/zk-rollup-tutorial

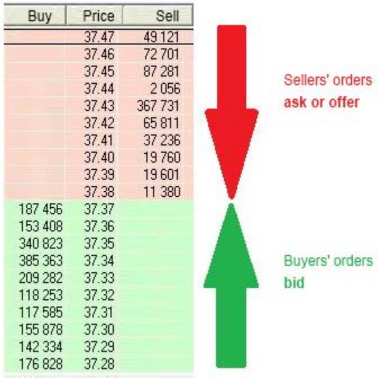
Exchange: Off-chain order book matching

Order book matching on-chain: performance problems, due to scalability (even in Fabric)

Alternative on-chain exchanges, e.g. Liquidity pool based: efficiency problems like, capital inefficiency, slippage, impermanent loss …

Executing order book matching off-chain, but ZK / SNARK proof on the correctness

- orders and balances represented as merkle trees
- matching is a new state of merkle trees
- correctness of order matching
- correctness of new balances and order book
- correctness of merkle trees



Buy	Price	Sell
	37.47	49 121
	37.46	72 701
	37.45	87 281
	37.44	2 056
	37.43	367 731
	37.42	65 811
	37.41	37 236
	37.40	19 760
	37.39	19 601
	37.38	11 380
187 456	37.37	
153 408	37.36	
340 823	37.35	
385 363	37.34	
209 282	37.33	
118 253	37.32	
117 585	37.31	
155 878	37.30	
142 334	37.29	
176 828	37.28	

Sellers' orders
ask or offer
Buyers' orders
bid

ZK matching: https://www.thestreet.com/crypto/innovation/trustless-order-matching-a-zk-matching-engine-poc-

Off-chain computation

Smart contract logic - business logic is realized by a SNARK computation model.

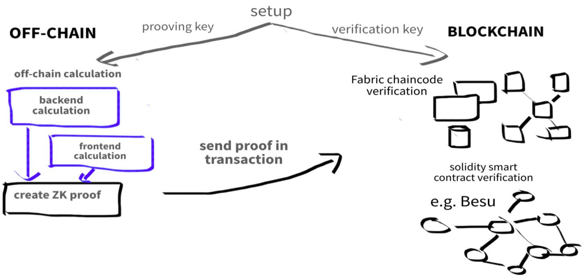
Setup: preprocessing and parameter registering in chaincode.

Business is executed off-chain (server - client, organisational aspects).

Result of the computation and zk (SNARK) proof is validated by the chaincode, chain is modified by the public data.

Privacy vs scaling

Multi organisational aspects.



And many more ….

Anonym endorsement:

- Anonym organisational endorsement with predefined organisational policy.

Use-cases with verifiable credentials:

- Verifiable credential : off-chain document signed by different parties and validated on-chain or off-chain

- Verification with partial data sharing (presentation) with validity proof : zk(SNARK)

- Authentication and authorization use-cases

Zero knowledge compliance:

- Proof of reserve
- Proof of solvency
- Zero knowledge taxation

Application specific ZK use-cases

Improved cross-channel communication and use-cases

Improved use-cases with private data collections

using "Replace image" to show your own photo.

Hunting for the SNARK

A learning group for ZK and SNARK application development

Every month, thirds thursday in 2025, from 18 (CET), online

Different topics on zero knowledge proof:

- mostly from application developers perspective
- basic theoretical introduction
- programming with different zk(SNARK) programming frameworks
- programming cool applications
- using Hyperledger Fabric as a major blockchain

Coordinated in LF Decentralized Trust discord channel

Quizzes and small programming challenges, LFDT merchs at the end

Hunting for the SNARK

A learning group for ZK and SNARK application development

First session is in February:

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

Coordination in LF Decentralized trust discord

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

Repo with all the contents:https://github.com/LF-Decentralized-Trust-labs/

Happy Hunting for the SNARK :)

Happy Hunting for the SNARK :)

Q & A

Daniel Szego

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