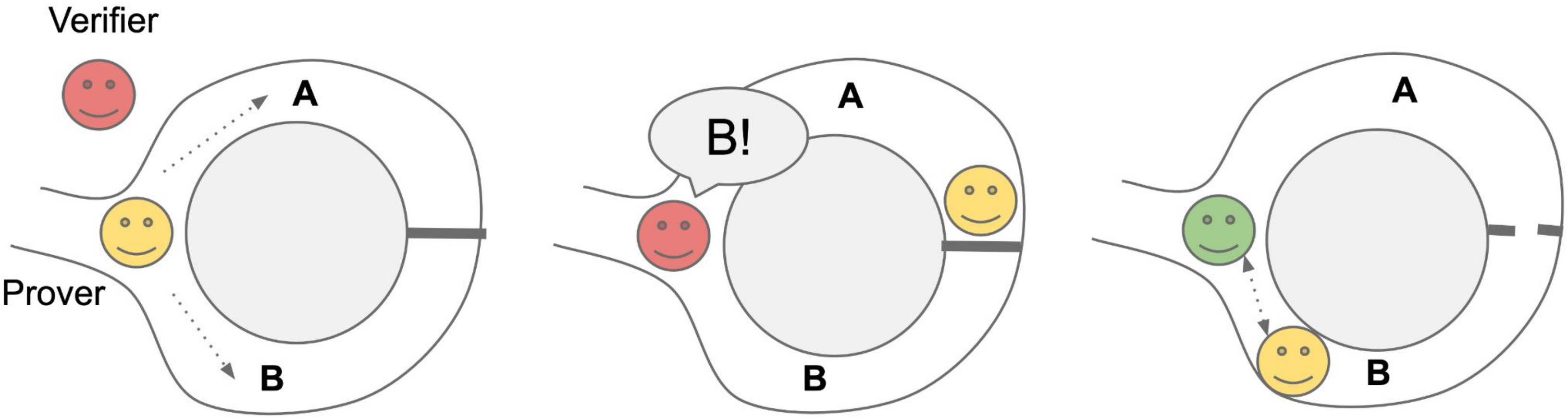
TUM Blockchain Conference

ZK 101: The Magic of Proving Without Revealing

Jan Lauinger

Munich, September 2024

A Principle to Convince Someone



Construction of a Cryptographic Proof

Prover		Verifier
$\alpha \in \mathbb{Z}_q$		$(U, A = U^{\alpha}) \in \mathbb{G}$
$r \leftarrow \mathbb{Z}_q$	$\xrightarrow{R = g^r \in \mathbb{G}}$	
	$\xleftarrow{c \in C}$	$c \leftarrow C = \{1, \ldots, 2^{\lambda}\} \subseteq \mathbb{Z}_q$
$z = c \cdot \alpha + r \in \mathbb{Z}$	$\xrightarrow{z}$	accept iff $U^z = A^c \cdot R$

1. Completeness Commitment scheme (opening, challenge, response)

$$U^z = g^z = g^{c \cdot \alpha + r} = g^r \cdot (g^{\alpha})^c = R \cdot (U^{\alpha})^c = R \cdot A^c$$

2. Soundness *Extractor* concept with the ability to extract secret knowledge from *P* convinces *V* of the existence of a satisfying solution.

2. Soundness Assumption: *V* is able to receive two accepting conversations (R, c, z) and (R, c', z').

With $U^z = A^c \cdot R$ and $U^{z'} = A^{c'} \cdot R$

$$\implies \quad U^{z-z'} = A^{c-c'}$$

$$\implies \quad \alpha = \frac{z-z'}{c-c'} = \log_g(U)$$

3. HVZK *Simulator* concept with simulated transcript T_{sim} and real transcript T_{real} of interactive protocol [4].

3. Honest Verifier Zero Knowledge

Select $z, c \leftarrow \mathbb{Z}_q$

Calculate $\alpha = \frac{g^z}{U^c}$

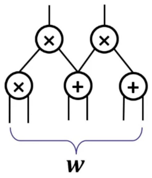
Output $T_{sim} = (\alpha, c, z)$

4. Zero Knowledge (Fiat-Shamir Heuristic)

$$c = H(R) \implies c = H(T)$$

From Dedicated Proofs to General-purpose Proof Systems

P claims to know a w such that $C(x, w) = y$



P claims to know a vector c such that $p(x) = V(x)q(x)$

Building blocks

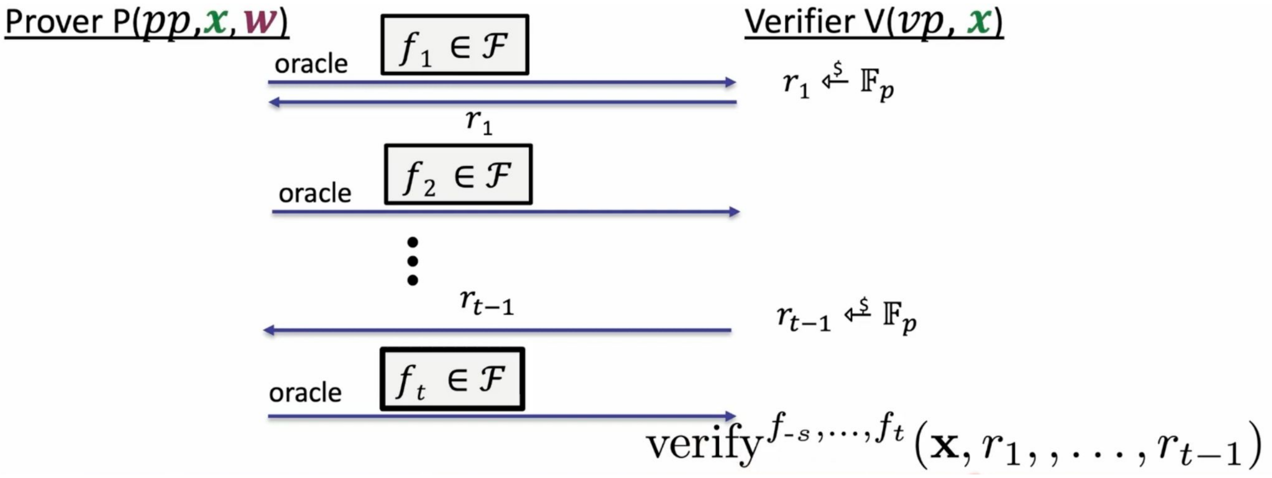
- Arithmetic representations & circuit satisfiability
  - R1CS, Plonkish, AIR, Custom CSS
- Quadratic arithmetic program - QAP (System of equations over polynomials)
- Functional commitment scheme (cryptographic object)
  - Cryptographic setup procedure?
- Compatible interactive oracle proof - IOP (information theoretic object)
  - Polynomial, Multilinear, Vector, etc.. IOP

- Computation
- Algebraic Circuit
- R1CS
- QAP
- Linear PCP
- Linear Interactive Proof
- zkSNARK

Zero knowledge proofs MOOC https://www.youtube.com/playlist?list=PLS01nW3Rtgor_yJmQsGBZAg5XM4TSGpPs

Functional IOP - generalized Linear Probabilistic Checkable Proofs - PCPs

A Linear PCP is **a PCP in which the proof is a vector of elements of a finite field , and such that the PCP oracle is only allowed to do linear operations on the proof.** Namely, the response from the oracle to a verifier query is a linear function.



Modern SNARK Constructions

https://www.youtube.com/watch?v=bGEXYpt3sj0&list=PLS01nW3Rtgor_yJmQsGBZAg5XM4TSGpPs&index=2

Functional Commitment and IOPs

Popular commitment schemes

- Polynomial, Multilinear, Vector (Merkle tree) commitment
- Inner product commitments (inner product arguments - IPAs)

Verifier access (powers of the verifer) in IOPs

- Oracle access: verifier may query provers messages
- Randomness: verifier is probabilistic
- Interaction: prover and verifier exchange messages
- Multi-prover: multiple provers that are isolated

How to Make SNARKs https://www.youtube.com/watch?v=KjkIQLJk4eQ

Constructing Functional Commitments

Popular constructions

- Bilinear Groups
  - KZG commitment
- Hash functions
  - e.g. for Fast Reed-Solomon IOP of Proximity - FRI
- Elliptic curves
  - for Bulletproofs
- Groups of unknown order

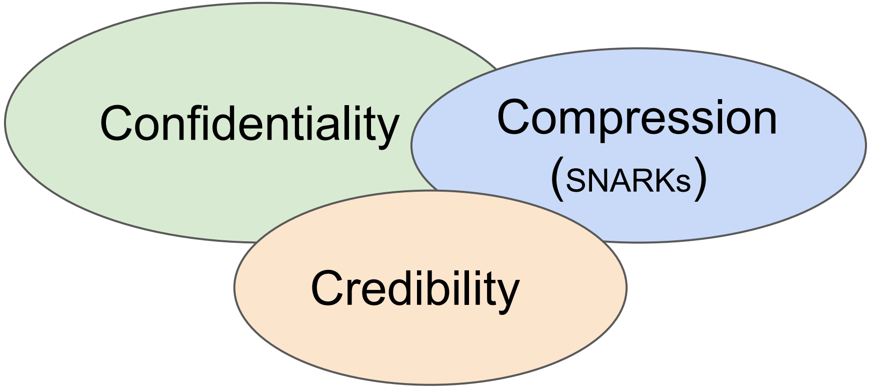
Types of Cryptographic Proof Systems

Construction and key properties

- zkSNARK
  - Succinctness: short proofs and fast to verify
- zkSTARK
  - Scalable, transparent
- MPC-based ZKP
  - Efficient evaluation of non-algebraic statements
- Recursive ZKPs
  - Proof of proof
- zkEVMs
  - Efficient ZK computation for any type of computation

ZKP Application Domains

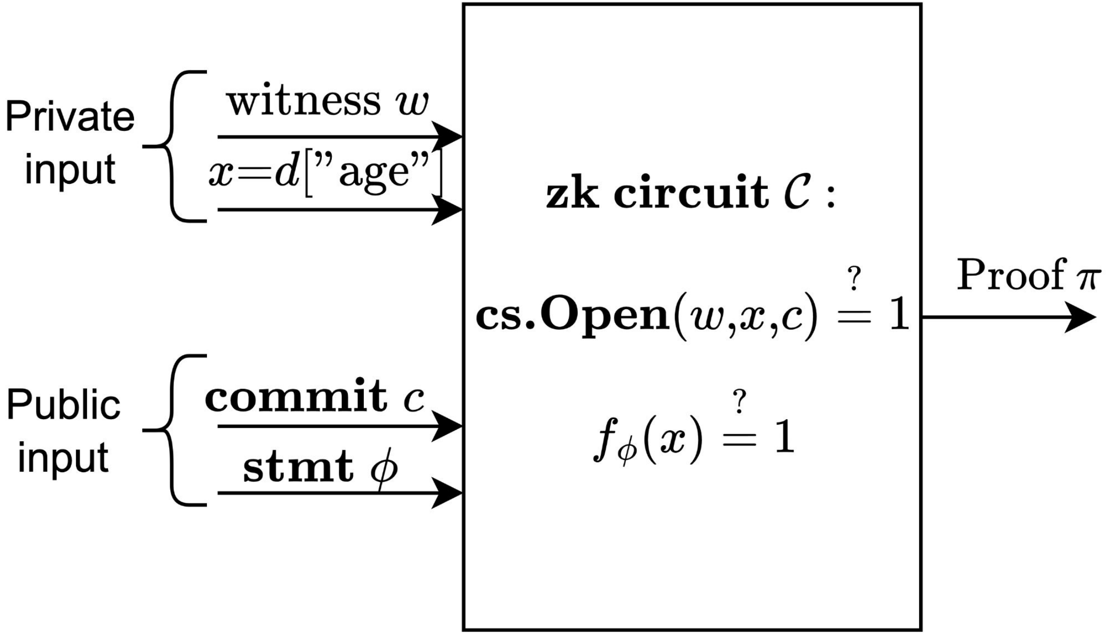
- **Privacy**
  - Credentials, group membership
- **Compression**
  - Rollups, compact credentials
- **Content Provenance**
  - zkTLS, image compression, data feeds, zkBridges
- **Blockchain Applications**
  - Tornado cash, Zcash



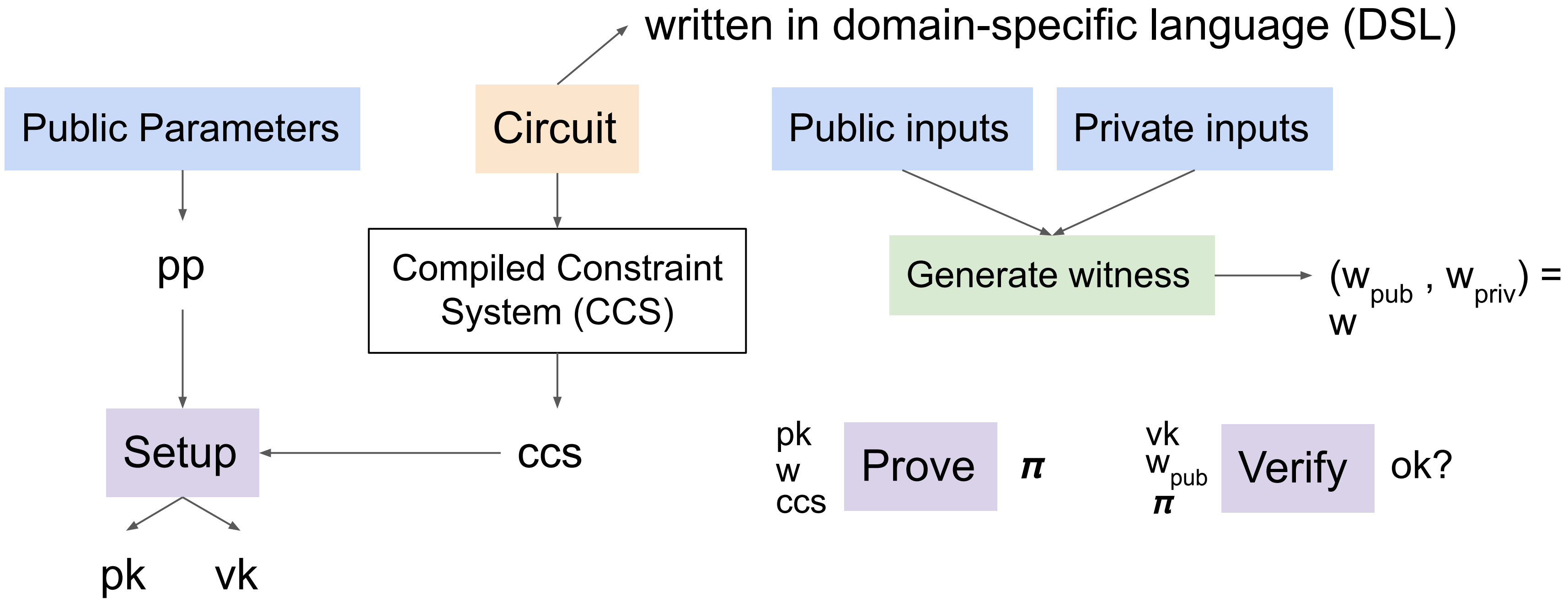
Group membership using ZKPs

- Membership without nullifiers
  - zkMessage
  - heyAnon
- Membership with nullifiers
  - Tornado cash
  - Semaphore
- Other membership proofs
  - zkEmail
  - zkJWT

Proving the Preimage of a Hash Function



Frameworks to Construct zkSNARKs



Running a ZKP Protocol with a Smart Contract Verifier

- **ganache -m "much repair shock … bullet interest solution"**
  - Starting local blockchain network (single instance)
- **go run main.go -init true**
  - Compiles a gnark circuit (cubic function)
  - Runs a *ZKP* setup procedure to generate prover and verifier keys
  - Exports a solidity proof verification contract
- **go run main.go -bindings true**
  - Compile contract into ABI and BIN files with *solc*
  - Use *abigen* to generate Go code bindings in a Go *verifier* package
- **go run main.go -deploy true**
  - Use Go *verifier* package to deploy the contract and interact with it
- **go run main.go -address 0x2e144…c888f1fF1a -verify true**
  - Call the contract to verify the computed proof

Link to Repo https://github.com/jplaui/gnark_lib/tree/main/deploy_solidity

Useful Resources

Proof of knowledge: https://asecuritysite.com/golang/go_proof

Schnorr Proof: https://asecuritysite.com/zero/schnorr

General-purpose Proof from Scratch: https://github.com/arnaucube/go-snark-study

ZK Circuit Library (gnark): https://github.com/jplaui/gnark_lib

ZK MOOC: https://www.youtube.com/playlist?list=PLS01nW3Rtgor_yJmQsGBZAg5XM4TSGpPs

Questions?

About myself

Name: Jan Lauinger
Company: **unaffiliated**, please hire me
Website: https://ianzn.com
Github: https://github.com/jplaui
Uni/TUM Profile: https://www.ce.cit.tum.de/esi/mitarbeiter/lauinger/
Resume: https://ianzn.com/assets/files/resume_latest.pdf