Zero-Knowledge Proofs (ZKPs)

By Niyati Bishop
Lewis and Clark College

Simple ZKP Definition

- Proves possession of some piece of information without revealing what that information is, or how to find it.

Basic Example (PCP*)

- Alice (prover) has a red ball and a green ball, and she wants to prove to her colorblind friend, Bob (verifier), that the two are distinct in appearance without revealing they are different colors.
- So, she hands the red and green balls to Bob, one in his right hand, one in his left, and closes her eyes, asking him to (secretly) either switch or not switch the two between his hands.
- Alice then opens her eyes, and guesses whether or not Bob has switched the two balls (Alice would immediately be able to tell, because can see the two balls are different colors)
- This process is continued multiple times until Bob is convinced that the two balls are actually different, and that Alice cannot simply be taking wild correct guesses.

*Probabilistically Checkable Proofs

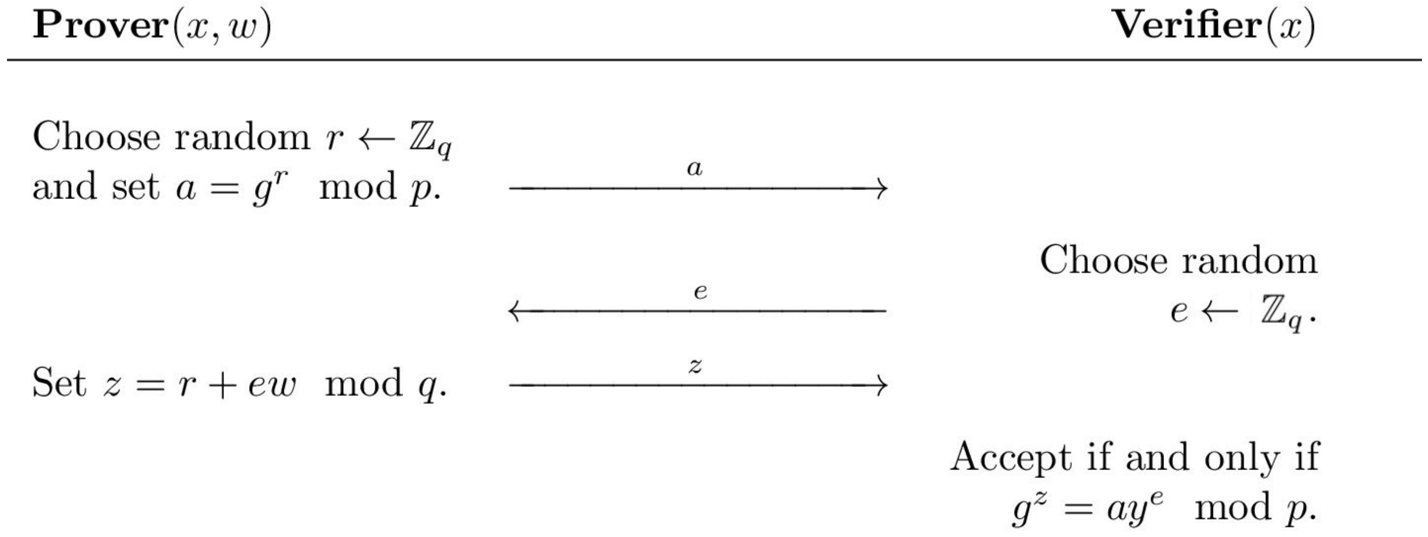
Formal Definition

In order for a protocol to formally be considered a ZKP, it must display

- **Zero-knowledge** - The verifier does not learn any information about the witness (secret information) from the proof.
- **Completeness** - If the proof is valid, the verifier will accept it.
- **Soundness** - If the prover can convince the verifier they know the witness, then they actually do know the witness, they are not lying.

Ex. Schnorr's (Sigma) Protocol

- Let p, q be prime, where $<g>=\mathbb{Z}_q \leq \mathbb{F}_p^*$ and the witness (secret value) be w.
- The prover (P) wants to prove to the verifier (V) that they know some w such that $g^w = y \mod p$. Let $x=(p,q,g,y)$.



Prover(x, w) | **Verifier**(x)

Choose random $r \leftarrow \mathbb{Z}_q$ and set $a = g^r \mod p$. $\xrightarrow{a}$

Choose random $e \leftarrow \mathbb{Z}_q$. $\xleftarrow{e}$

Set $z = r + ew \mod q$. $\xrightarrow{z}$

Accept if and only if $g^z = ay^e \mod p$.

Schnorr's Protocol: Security

- There is the (negligible) possibility that the prover has correctly *guessed* w such that the proof holds, rather than actually knowing it specifically.
- However, this would entail correctly guessing the (0/1) value of each bit in w (so a ½ chance of guessing each bit correctly), meaning if w is *t* bits long, the prover has a $(\frac{1}{2})^t$ chance of guessing the entire witness w correctly (they would have to guess each bit correctly t times).
- Thus, the security of the proof is dependent on how many bits long w is, the longer it is, the more secure the proof is.

Interactivity

- Sigma protocols (ex. Schnorr’s Protocol) require 3 information exchanges total between the prover and verifier, so two after the initial interaction. They are **interactive ZKPs**.
- **Non-interactive ZKPs (NIZKs)** would require no additional interactions after the initial exchange.

Zk-SNARKs

- Stands for **Zero-knowledge Succinct Non-interactive ARgument of Knowledge**
- **Succinct* here is subjective, the specific definition depends on the specific system, may mean polylogarithmic runtime, constant runtime, etc.

J. Thaler, 2023 [6]

Zk-SNARKs

- Consists of 3 algorithms – a key generator (`G`), prover (`P`), verifier (`V`).
- `G` takes a secret parameter *`lambda`* and the specific program we are generating a proof for (`C`) as inputs, and outputs a proving key (`pk`) and a verification key (`vk`)

  → `G(`*`lambda`*`, C) = (`*`pk`*`,`*`vk`*`)`

- `P` then takes *`pk`*, public input *`x`*, and its private witness *`w`* as its input, and outputs a proof.

  → `P(x,w,pk) = prf`

- `V` then takes *`prf`*, *`vk`*, and *`x`* as input and outputs true if the proof is valid, false if not.

  → `V(prf,vk,x) = [True or False]`

*Note: *`lambda`* must not be known to the prover or the verifier, otherwise the security of the proof can be compromised.

Trusted Setups*

- It is imperative that the *lambda parameters* described previously remain unknown to both the prover and the verifier.
- Thus, protocols called *trusted setups* are used to generate the values that are a part of *lambda*.
- A commonly-used trusted setup technique is the *Powers of Tau* ceremony.

*A side topic, but an integral process for Groth16 implementations

Zk-SNARKs Setup - Arithmetic Circuits → R1CS → QAP

Arithmetic Circuits

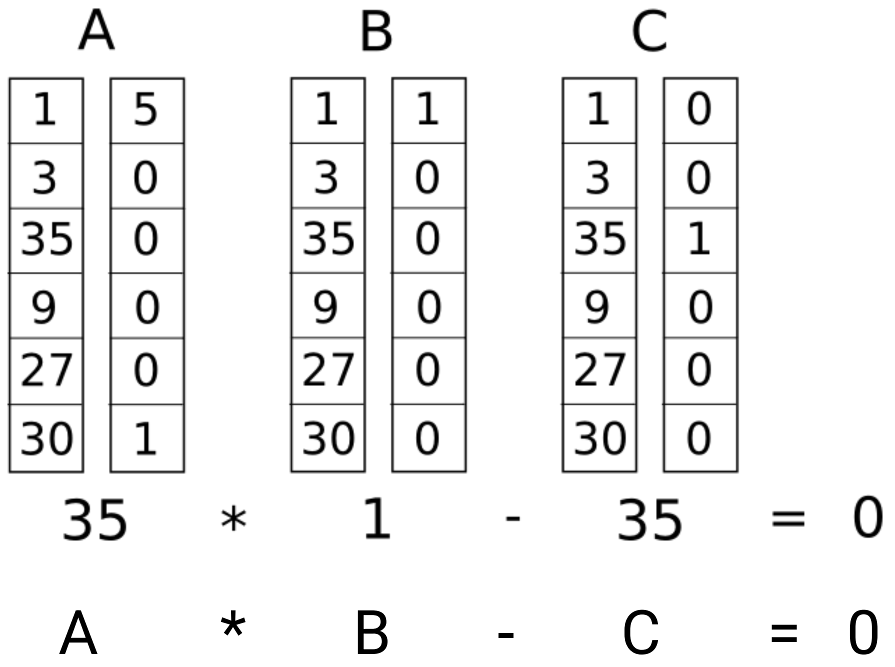
- A circuit diagram with arithmetic operations in place of logic gates, and variables as inputs.
- For our purposes, we will view it as a simple list of constraints on these variables.

1. $sym_1 = x * x$
2. $y = sym_1 * x$
3. $sym_2 = y + x$
4. $OUT = sym_2 + 5$

V. Buterin, 2016. [2]

Arithmetic Circuits → R1CS (Rank 1 Constraint System)

- Given a vector of values for each variable (possible witness), the vector can be checked against each of these constraints using the following formula, using a second “check vector” in addition.



A		B		C	
1	5	1	1	1	0
3	0	3	0	3	0
35	0	35	0	35	1
9	0	9	0	9	0
27	0	27	0	27	0
30	1	30	0	30	0

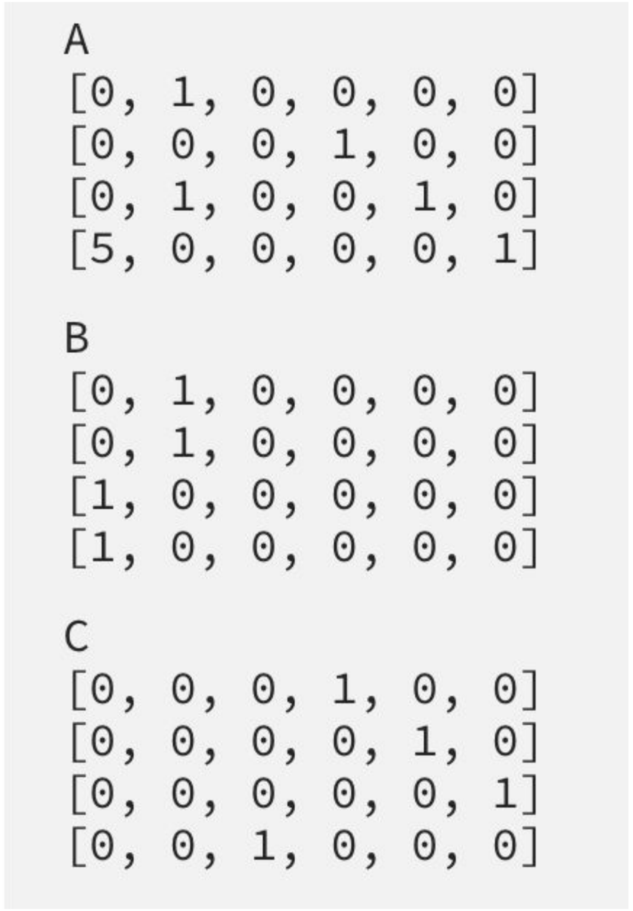
$$35 * 1 - 35 = 0$$

$$A * B - C = 0$$

V. Buterin, 2016. [2]

Arithmetic Circuits → R1CS

- We can then compile all of these "check vectors" into $m+1 \times n+1$ matrices, $n+1$ is the length of the vectors, $m+1$ the number of constraints.



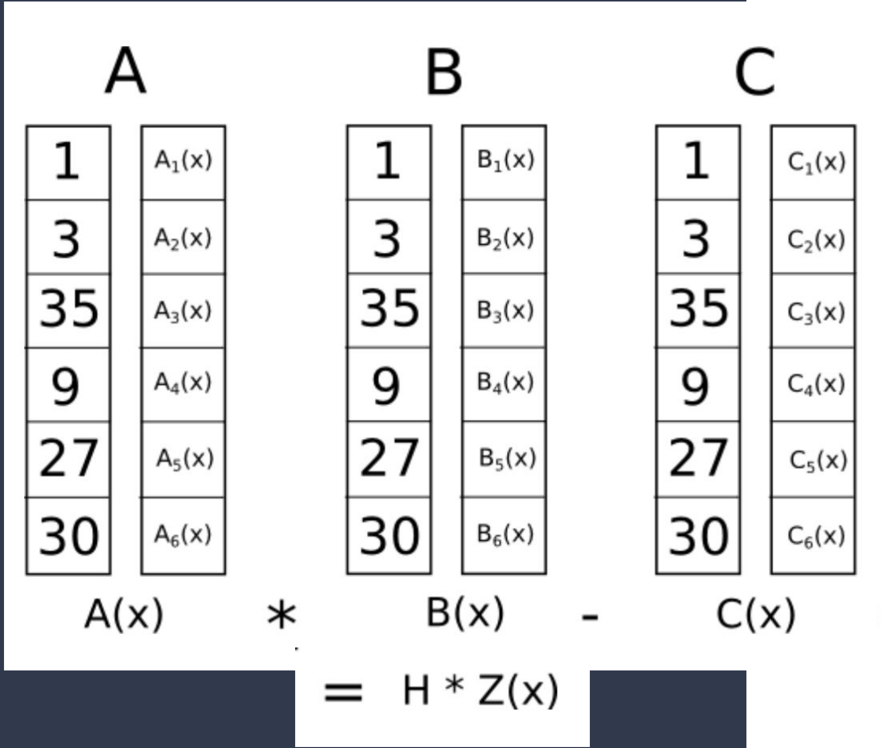
```
A
[0, 1, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0]
[0, 1, 0, 0, 1, 0]
[5, 0, 0, 0, 0, 1]

B
[0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0]

C
[0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 1]
[0, 0, 1, 0, 0, 0]
```

V. Buterin, 2016. [2]

R1CS → QAP (Quadratic Arithmetic Program)



- Using Lagrange interpolation, we can construct an m- degree polynomial for each column in the matrix, which evaluates to each value in its column when imputed using the number of the value's corresponding constraint (t).
- Constructing vectors of these polynomials, we can check all constraints at once using the original check equation, which is satisfied by the witness, will result in a new polynomial with roots/"zeroes" corresponding to the constraint numbers.
- The full polynomial will be some $h(t)z(t)$ where $z(t)$ is the polynomial constructed by the constraint roots, ex.

$$z(t) = (x-1)(x-2)(x-3)(x-4)$$

V. Buterin, 2016. [2]

Groth16 Protocol

Elliptic Curve Pairings

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be elliptic curve groups of prime order p.
Define bilinear map

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

Where $\mathbb{G}_1 =< g_1 >$ and $\mathbb{G}_2 =< g_2 >$, and thus $\mathbb{G}_T =< e(g_1, g_2) >$

Elliptic Curves Pairings

$$e(g_1, g_2) * a = e(ag_1, g_2) = e(g_1, ag_2)$$

$$e(g_1, g_2) * ab = e(ag_1, bg_2) = e(bg_1, ag_2)$$

M. Petkus, 2019 [9]

Important Notation

$$[a]_1 := ag_1$$

$$[b]_2 := bg_2$$

- Witness (column) vector, where w is the secret component and x is the public input:

$$\vec{z} = (1, w, x)$$

(One can choose which components of the witness are secret vs. public)

Lambda Parameters

$$\lambda = (\alpha, \beta, \delta, \gamma, t)$$

- Generated by a trusted setup
- Top secret, neither prover nor verifier can know these values.

J. Groth, 2016 [1]
K. George, 2022 [8]

Key Components

Proving Key		Verification Key
$[\alpha]_1$	$[A_0(t)]_1, ..., [A_n(t)]_1$	$e([\alpha]_1, [\beta]_2)$
$[\beta]_1$	$[B_0(t)]_1, ..., [B_n(t)]_1$	$[\delta]_2$
$[\beta]_2$	$[B_0(t)]_2, ..., [B_n(t)]_2$	$[\gamma]_2$
$[\delta]_1$	$[K^p_{k+1}(t)]_1, ..., [K^p_n(t)]_1$	$[K^v_0(t)]_1, ..., [K^v_k(t)]_1$
$[\delta]_2$	$[\frac{Z(t)}{\delta}]_1$	

J. Groth, 2016 [1]
LambdaClass, 2023 [3]

Key Components

- Where

$$K_i^v(t) = \frac{\beta A_i(t) + \alpha B_i(t) + C_i(t)}{\gamma}$$

$$i = 0, \ldots, k$$

$$K_i^p(t) = \frac{\beta A_i(t) + \alpha B_i(t) + C_i(t)}{\delta}$$

$$i = k+1, \ldots, n$$

Proof → $\pi = ([\pi_1]_1, [\pi_2]_2, [\pi_3]_1)$

- Where

$$[\pi_1]_1 = [\alpha]_1 + \sum z_k [A_k(t)]_1 + r[\delta]_1$$

$$[\pi_2]_2 = [\beta]_2 + \sum z_k [B_k(t)]_2 + s[\delta]_2$$

*random scalars r and s

$$[\pi_2]_1 = [\beta]_1 + \sum z_k [B_k(t)]_1 + s[\delta]_1$$

$$[\pi_3]_1 = \sum w_i [K_i^p(t)]_1 + [h(t)Z(t)]_1 + s[\pi_1]_1 + r[\pi_2]_1 - rs[\delta]_1$$

- with w_i as the secret witness components of z_k, which are the components of the full input vector

J. Groth, 2016 [1]
LambdaClass, 2023 [3]

Proof Validity Checking

- The proof is sent to the verifier, which, using the verification key, checks that:

$$P_1 = P_2$$

Where

$$P_1 = e([\pi_1]_1, [\pi_2]_2)$$
$$P_2 = e([\pi_3]_1, [\delta]_2) + e([\alpha]_1, [\beta]_2) + e((\sum x_i [K_i^v(t)]_1), [\gamma]_2)$$

- with x_i as the public components of the input vector z_k

J. Groth, 2016 [1]
LambdaClass, 2023 [3]

Malleability*

- Refers to the ability to alter a proof after its generation
- Has useful applications, but can also be a security risk depending on the specific application of the ZKP.
- Groth16 is in original form is malleable, but it can be easily altered to be non-malleable.

*A side topic, but a very important consideration for Groth16 applications

Sui Foundation, 2023 [10]

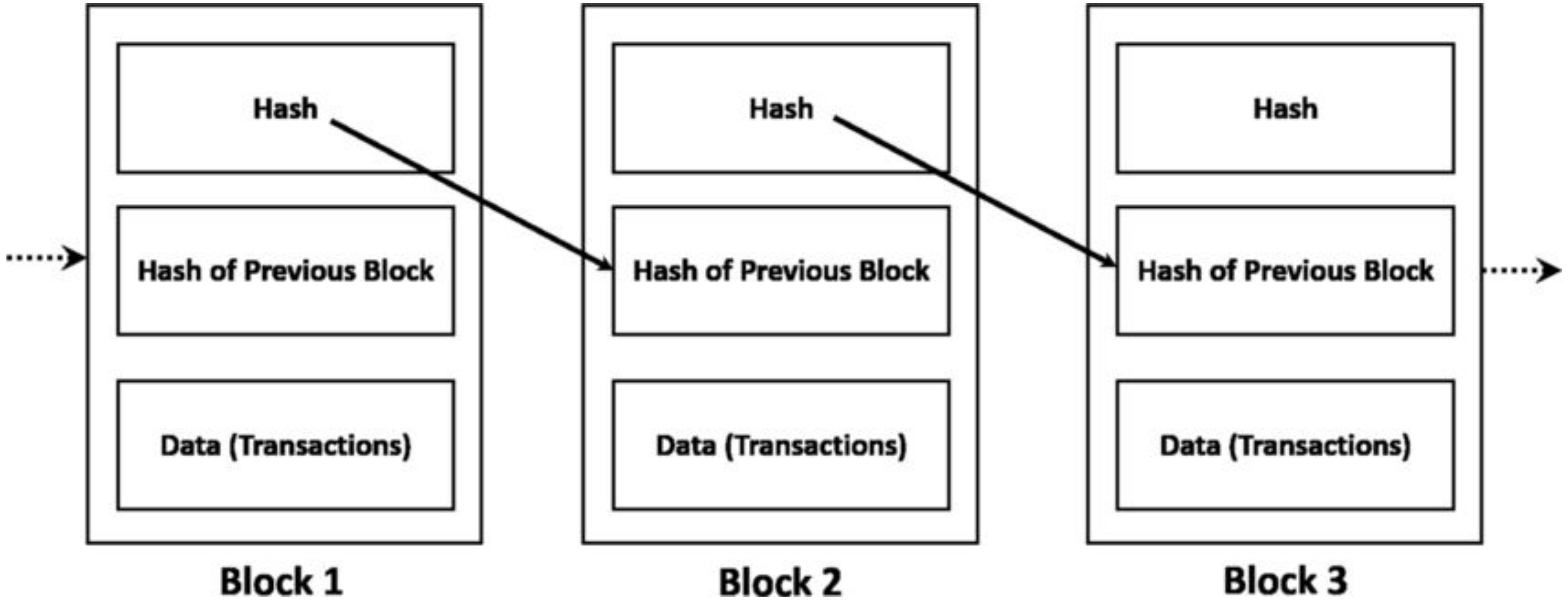
Zk-SNARK Applications: Blockchains

What is a Blockchain?

- A ledger/database where data is stored in blocks linked together by cryptographic hashes.
- Managed by a peer-to-peer (P2P) network of computers (nodes) in various locations running the blockchain’s software.
- Known for being “distributed” or “decentralized” (no one computer or location has full control over the database).

Wikipedia, 2025 [13]

What is a Blockchain?



A. Bosu et al., 2019 [25]

Blockchain Scalability

- Scalability refers to to how many transactions can be completed in a given amount of time.
- Scalability can often be low when all transactions are completed by the main blockchain network.
- Thus, some scalability-improving strategies include sharding, sidechains, and layer 2 networks.

Blockchain Layers

- Layer 0 (L0) - The foundation on which the blockchain is built, so the internet itself, the hardware/computer/nodes that run the software that supports the blockchain, etc.
- Layer 1 (L1) - The blockchain itself
- Layer 2 (L2) - A separate supplementary network of nodes that execute blockchain transitions, taking the burden off of the L1 and enhancing scalability.

T. Revoredo, 2023 [17]

Layer 2

- A separate network of nodes (computers) for executing blockchain transactions.
- Inherits the same security structures implemented by the main blockchain.
- Transactions are moved back and forth between the layer 2 and the main blockchain via **smart contracts**, which bundle multiple transactions into **rollups** and are then sent between the main blockchain (layer 1) and the layer two.

Chainlink, 2023 [16]

Layer 2 Rollups

The two types of rollups are:

- **Optimistic Rollups** - Transactions in the rollups are automatically assumed to be valid, if any blockchain operator dispute the validity of the rollup, a process ensues to isolate the point of dispute and verify the computations.
- **Zk-Rollups** - Transactions are verified via a Zk-SNARK before they are moved to the main blockchain.

Zk-SNARKs in Layer 2

- When a Zk-SNARK is used for Zk-rollup validation, the verifier lies in the smart contract associated with the main blockchain, responsible for moving the transactions back and forth between layer 2 and the blockchain.
- The prover resides on the layer 2, generating a proof that its transaction-executing computations are valid.

ZoKrates – Ethereum's Remix VM

- https://remix.ethereum.org/#lang=en&optimize&runs=200&evmVersion&version=soljson-v0.8.30+commit.73712a01.js

J. Eberhardt, S. Tai, 2018 [24]

Further research

- Formally verifying smart contract implementations of Groth16

Thank you!

References

[1] J. Groth, “On the size of pairing-based non-interactive arguments,” in Annual international conference on the theory and applications of cryptographic techniques, pp. 305–326, Springer, 2016.
[2] V. Buterin, “Quadratic arithmetic programs: from zero to hero,” URL: https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649, 2016.
[3] LambdaClass, “An overview of the groth16 proof system,” 2023.
[4] M. Binello, “Groth16,” 2019.
[5] CryptoHack, “Zkp challenges,” 2025.
[6] J. Thaler, “misconceptions about snarks (and why they hold us back),” 2023.
[7] Consensys, “Introduction to zk-snarks,” 2017.
[8] K. George, “The mathematical mechanics behind the groth16 zero-knowledge proving protocol,” 2022.
[9] M. Petkus, “Why and how zk-snark works,” arXiv preprint arXiv:1906.07221, 2019.
[10] Sui Foundation, “On the malleability of groth16 proofs,” 2023.
[11] C. Nightingale, “A full comparison: What are zk-snarks and zk-starks?,” 2024.
[12] C. Nightingale, “What are blockchain rollups? a full guide to zk and optimistic rollups,” 2024.
13] Wikipedia, “Blockchains,” 2025.
[14] RareSkills, “Zero knowledge programming languages,” 2022.
[15] A. Pinto, “Practical zk-snarks for ethereum,” 2019.
[16] Chainlink, “What is layer 2?,” 2023.
[17] T. Revoredo, “Blockchain layers.,” 2023.
[18] Coinbase, “What are sidechains?.”
[19] Coinbase, “What is crypto sharding and how does it work?.”
[20] Alchemy Team, “Ethereum sidechains vs layer 2s: What’s the difference?,” 2022.
[21] Inquesta, “What exactly is the blockchain [is it private?],” 2024.
[22] A. Hayes, “Blockchain facts: What is it, how it works, and how it can be used,” 2025.
[23] P. Gratton, “What is a block in the crypto blockchain, and how does it work?,” 2025.
[24] J. Eberhardt and S. Tai, “Zokrates-scalable privacy-preserving off-chain computations,” in 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1084–1091, IEEE, 2018.
[25] A. Bosu, A. Iqbal, R. Shahriyar, and P. Chakraborty, “Understanding the motivations, challenges and needs of blockchain software developers: A survey,” Empirical Software Engineering, vol. 24, no. 4, pp. 2636–2673, 2019.