

---

# Comparing Various Zero-Knowledge-Proofs

Himanshu Singhal  
Abhilash Chauhan  
Soumith Reddy Edla  
Bhagya Rishiroop Boda  
Brandon Pearl

---

# What is Blockchain?

Decentralized, Distributed ledger to record transactions and store data.

## The Properties of Distributed Ledger Technology (DLT)

### Programmable

A blockchain is programmable (i.e. Smart Contracts)

### Secure

All records are individually encrypted

### Anonymous

The identity of participants is either anonymous or pseudonymous

### Distributed

All network participants have a copy of the ledger for complete transparency

### Immutable

Any validated records are irreversible and cannot be changed

### Time-stamped

A transaction timestamp is recorded on a block

### Unanimous

All network participants agree to the validity of each of the records

# Privacy Concerns in Blockchains

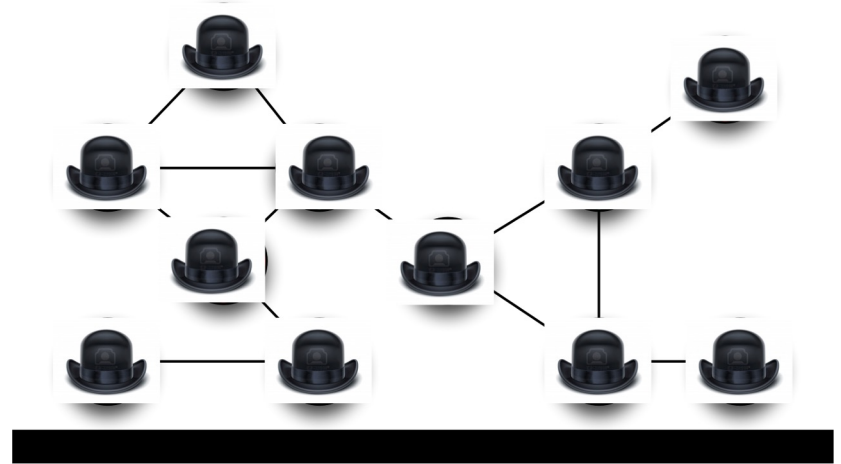
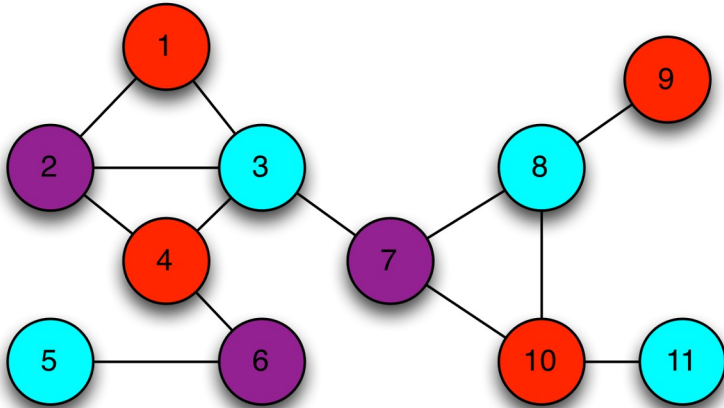


- Usage of pseudonymous identities
- Linkability of transactions
- Lack of confidentiality

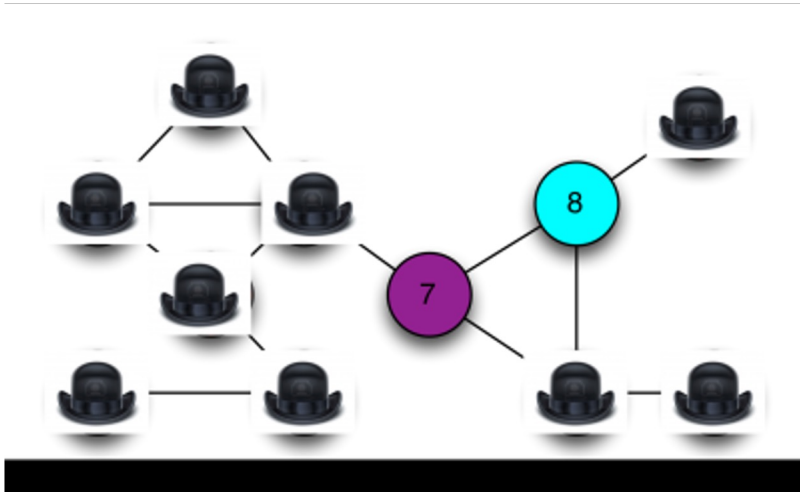
# Zero knowledge proofs

The ability to prove honest computation  
without revealing inputs

# Zero knowledge proofs Example



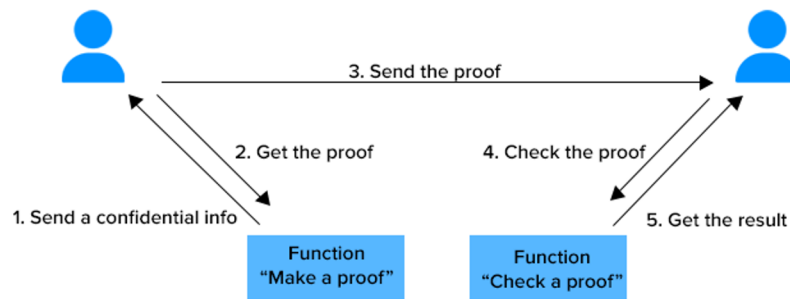
# Power of Zero knowledge proofs

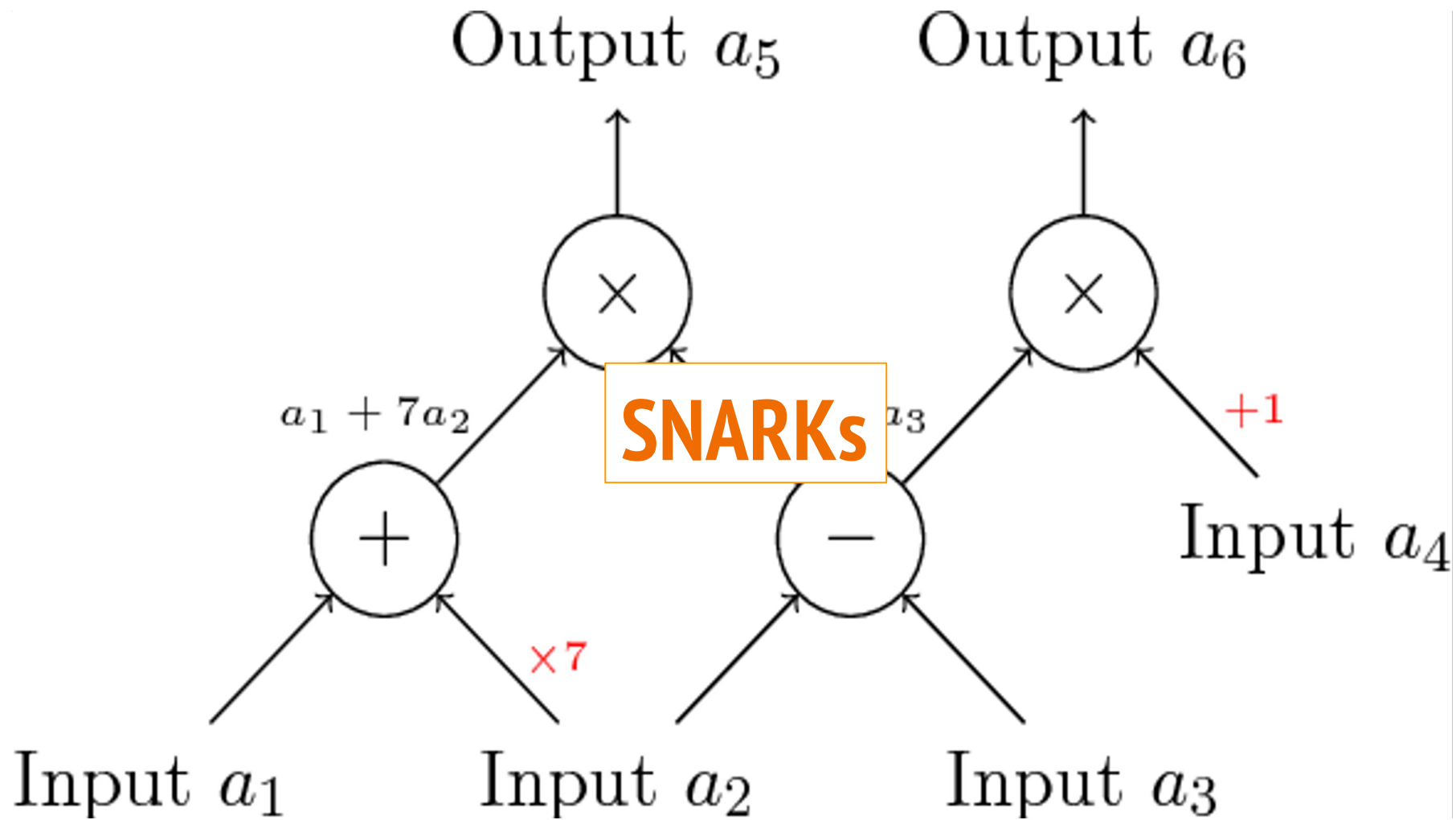


- Probability of incorrect algorithm when chosen once =  $(E-1)/E$
- When chosen again,  $(E-1)/E * (E-1)/E$

# Types of ZKPs

- ❑ Interactive
- ❑ Non-Interactive
- ❑ Succinct





# The Basics

**SNARK** = Succinct **N**on-interactive **AR**guments of **K**nowledge



**BOB**

I have the hash!  
Prove that you have  
the OG value



**ALICE**

# The Basics



**ALICE**



```
function C(x, w) {  
  return ( sha256(w) == x );  
}
```



**TRUE / FALSE**

# The Basics



**ALICE**



```
function C(x, w) {  
  return ( sha256(w) == x );  
}
```

**TRUE**




**BOB**

# The 3 Algorithms

**G** takes a  $\lambda$  and **C** (a program) and generates **pk** & **vk**

**P** takes a **pk** and **x** (public input) and generates a proof  $\pi$

**V** takes a  $\pi$ , **x**, and **pk** and returns **true** or **false**

Generator (C circuit,  $\lambda$  is 

$(pk, vk) = G(\lambda, C)$

Prover (x pub inp, w sec inp):

$\pi = P(pk, x, w)$

Verifier:

$V(vk, x, \pi) == (\exists w \text{ s.t. } C(x, w))$

# Alice vs Bob using SNARKs



**BOB**

**G**  
→



**ALICE**

**P**  
→



**BOB**

**v**  
→  
**TRUE**  
or  
**FALSE**

# Evaluating SNARKs

1. Complexity
2. Trust assumptions
3. Setup required
4. Limited flexibility
5. Performance

1. Strong security guarantee
  2. Privacy
  3. Scalability
  4. Non-interactive design
  5. Cross-chain compatibility
-

# SNARKs in Action



# Bulletproofs - Short like Bullets



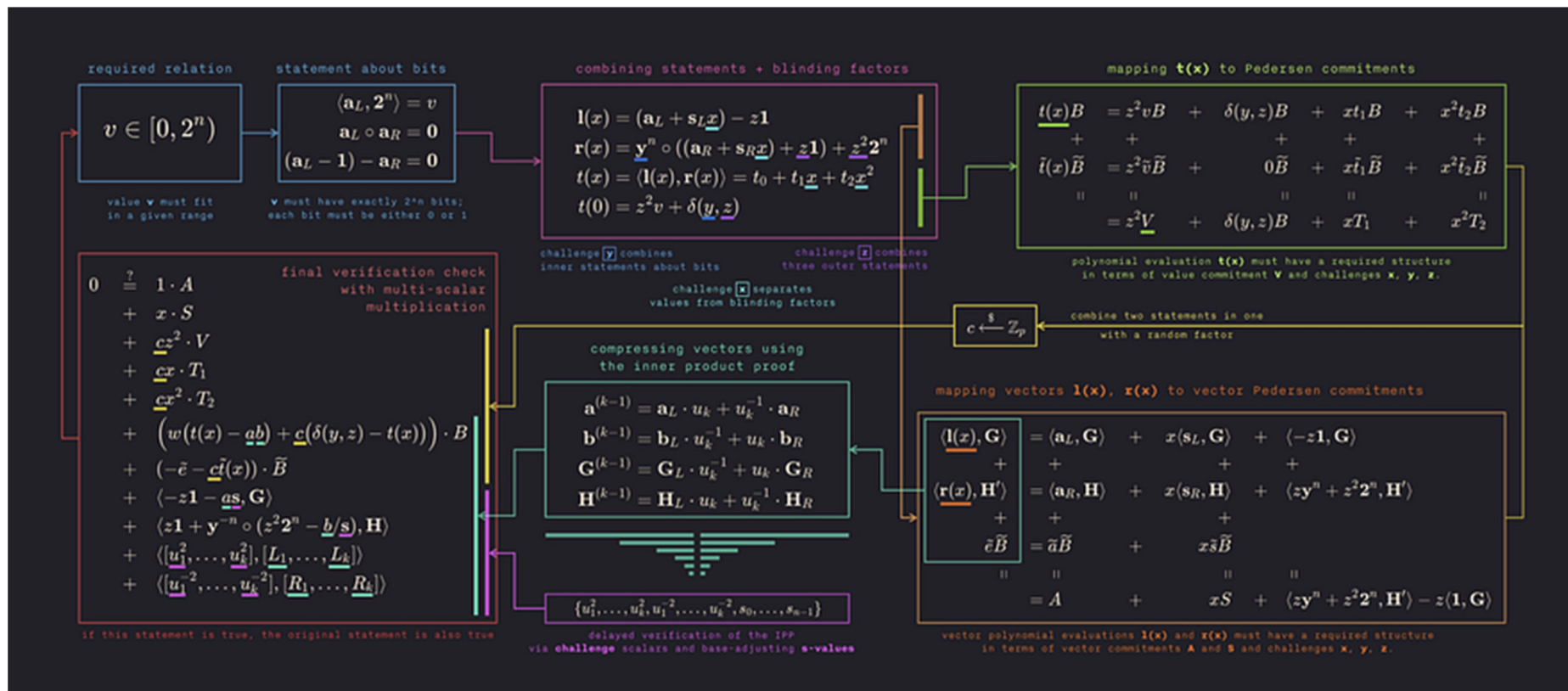
# Bulletproofs - Introduction

- Bulletproofs were introduced in 2017 by Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell.
- Non-interactive zkp protocol with very short proofs and no trusted setup
- Enhance the privacy and security of various applications, such as cryptocurrencies

## Bulletproofs - Problem they solve ?

- Does not require a trusted setup like zk-Snarks.
- Reduces Size & Overall Verification time.

# Bulletproofs - Working



# Bulletproofs - Working - Pedersen commitment

- Suppose a user wants to commit to a secret value 'v'. The user will generate a random blinding factor 'r' and compute the commitment 'C' using the following formula:

$$\mathbf{C} = \mathbf{v} * \mathbf{G} + \mathbf{r} * \mathbf{H}$$

- Here, 'v' is the secret value, and 'r' is the random blinding factor. The commitment 'C' is a point on the elliptic curve.

# Bulletproofs - Working - Range Proof

•The prover wants to prove that the secret value 'v' is within a specific range **[0,  $2^n - 1$ ]**, without revealing the actual value. The prover creates a vector 'aL' representing the binary decomposition of 'v' and computes

**'aR' as  $aR_i = aL_i - 1$ .**

**$aL = (aL_1, aL_2, \dots, aL_n)$   $aR = (aR_1, aR_2, \dots, aR_n)$**

# Bulletproofs - Working - Vector Pedersen commitments

- The prover generates Vector Pedersen commitments for the vectors 'aL' and 'aR' using distinct generator points ( $G_i$ ,  $H_i$ ) and random blinding factors ( $rL_i$ ,  $rR_i$ ):

$$A_i = aL_i * G_i + aR_i * H_i + rL_i * P$$

$$S_i = aR_i * G_i + (aL_i - aR_i) * H_i + rR_i * P$$

- Where  $P$  is an additional public generator point on the elliptic curve.
- The prover then computes the Vector Pedersen commitments 'A' and 'S' as the sum of the individual commitments:  $A = \sum A_i$ ,  $S = \sum S_i$

# Bulletproofs - Working - Inner Product Proof

- The prover computes the inner product 't' of the vectors 'aL' and 'aR':

$$\mathbf{t} = \langle \mathbf{aL}, \mathbf{aR} \rangle$$

- The prover then creates commitments 'T1' and 'T2' for the coefficients of 't' using random blinding factors 'tau1' and 'tau2':

$$\mathbf{T1} = \mathbf{tau1} * \mathbf{G} + \mathbf{t1} * \mathbf{H}$$

$$\mathbf{T2} = \mathbf{tau2} * \mathbf{G} + \mathbf{t2} * \mathbf{H}$$

- Where  $\mathbf{t} = \mathbf{t0} + \mathbf{t1} * \mathbf{x} + \mathbf{t2} * \mathbf{x}^2$ , and x is a random challenge generated using the Fiat-Shamir heuristic.

# Bulletproofs - Working - Inner Product Proof

- During each step, the prover reduces the size of the vectors  $a_L$  and  $a_R$  by half. The prover sends commitments for the newly computed vectors, and the verifier responds with challenges. This process is repeated until a final proof is generated.
- The verifier checks the validity of the inner product proof using the commitments  $A$  and  $S$ , the provided proof, the Pedersen commitment  $V$ , and the known value  $c$  (in our case, the secret value  $v$ ).
- The verifier computes the expected inner product of the vectors  $a_L$  and  $a_R$  using the provided proof and the challenges derived during the protocol. If the computed inner product matches the known value  $c$ , the verifier accepts the proof as valid, ensuring that the prover knows the correct vectors  $a_L$  and  $a_R$  without revealing the actual vectors.

# Bulletproofs - Applications & Future Use

- Cryptocurrencies
- Online voting
- Decentralized finance
- Healthcare
- Identity management
- Internet of Things

# Bulletproofs - Benefits & Limitations

- Improved privacy
- Decentralization
- Efficiency
- Flexibility
- Short proofs
- Scalability
- No trusted setup
- Complexity
- Verification time
- Trust assumptions
- Limited adoption

# STARKs

STARK = **S**uccinct **T**ransparent **AR**guments of **K**nowledge

- Introduced in 2018 by Eli Ben-Sasson and his fellow team of academics
- **GOAL:** Generate cryptographic proofs that are both secure and transparent

# STARKs - What makes them different?

1. **Transparency:** Do not rely on any trusted setup (generation of a set of public parameters that must be trusted by all parties)
2. **Scalable:** Generates a proof in less time than other methods
3. **Post-quantum security:** More resistant to attacks utilizing the power of quantum computation



# STARKs - How do they work?

1. Generate a proof
  - a. Prover constructs a polynomial representation of the computation and evaluates at random points
  - b. Prover constructs a low-degree polynomial from the original, whose evaluation is easier to compute
2. Verify the proof
  - a. Verifier evaluates low-degree polynomial at random points, and checks that output matches using Fast Fourier Transform



# STARKs - Transparency and Scalability

- **Transparency** is achieved by making the choice of random points used in the proof generation and verification publicly verifiable
- **Scalability** is achieved by recursively breaking down the original proof into sub-proofs, and eventually combined back into a single proof using interpolation
  - Scales well with large computations
  - Utilizes parallel processing

# STARKs - Starks In Action

- Eli Ben-Sasson cofound **StarkWare**, a company focused on developing and commercializing STARK technology
  - Crypto gaming company, Immutable, use StarkWare for mass-minting of NFTs
  - DeFi platform DeversiFi uses StarkEx to settle trades at better prices, decrease minimum order-sizes, and keep trade history private and secure



## Any Size

Whatever the project size, STARK technology can power it. Today's STARK solutions give you even greater scaling ability than before; The first generation of STARK solutions radically scaled blockchain and reduced gas costs by batching thousands of transactions from a single application, and processing them via just one proof. Now, STARK solutions also pack together batches from **several disparate** applications into a single proof. This is the blockchain equivalent of sharing a cab with others, for efficiency.

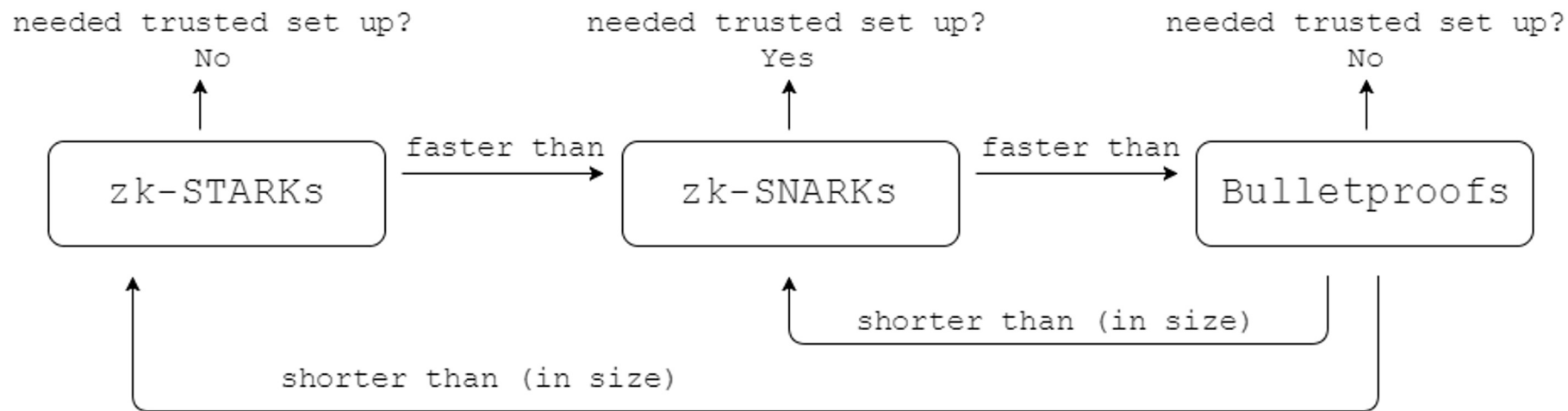
# STARKs - Limitations

- **Complexity:** STARKs use complex math and the implementation of them can be difficult and may require a high amount of resources
- **Proof size:** Proofs can potentially be large, due to them being designed for high levels of security and designed to handle highly complex computations

# How do they stack up against each other?

|                               | Proof Size | Prover Time | Verification Time |
|-------------------------------|------------|-------------|-------------------|
| SNARKs<br>(has trusted setup) | 288 bytes  | 2.3s        | 10ms              |
| STARKs                        | 45KB-200KB | 1.6s        | 16ms              |
| Bulletproofs                  | ~1.3KB     | 30s         | 1100ms            |

# When to use what



# Related work

Earliest usage of ZKP

- **Graph isomorphism:** Use of ZKP to prove graph isomorphism
  - “The knowledge complexity of Interactive systems”
- **Interactive ZKP:** Electronic voting , Digital signatures ,Multi party Computation
  - “Protocols for secure computations”
- **Zcash:** Privacy focused cryptocurrency to enable anonymous transactions
  - “Zero coin: Anonymous Distributed E-cash from bitcoin ”

# Related work

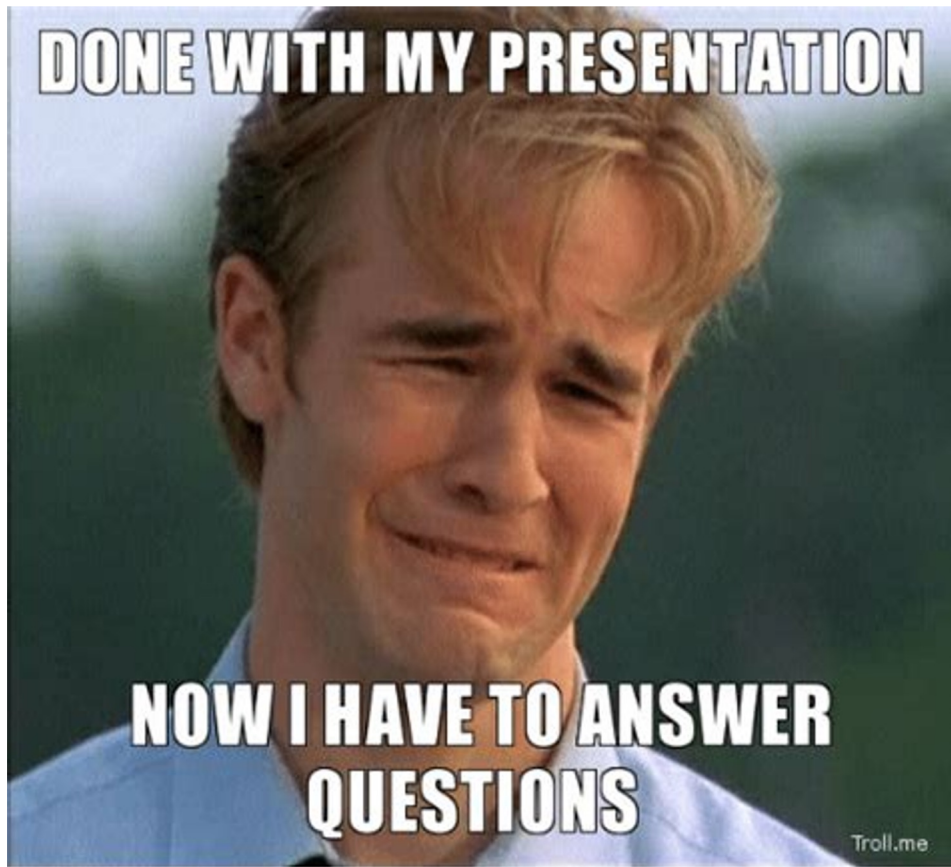
## Advancements in ZKP:

- Succinct ZKP (Bulletproofs , STARKs)
- Interactive ZKP (NIZKP)
- Scalable ZKP (Sonic and Aurora)
- Privacy preserving ZKP (zk-SNARKs, zk-STARKs) etc

# Conclusion

- Optimal option for a ZKP depends on the application's requirements
- Earlier ZKP protocols have limitations
  - Long proof size
  - High computational power
- SNARKs, STARKs and Bulletproofs overcome these challenges
- **Future Scope:**
  - Vulnerable to Quantum attacks (Early Stages)
  - Interoperability between Block Chains

**DONE WITH MY PRESENTATION**



**NOW I HAVE TO ANSWER  
QUESTIONS**

Troll.me