Zero Knowledge Proofs

and their applications to Blockchain

image credits

Tanmayi Jandhyala
MASc ECE, University of Waterloo

Waterloo Blockchain

About Me

Master’s candidate in Applied Science, Computer Engineering

Working with implementing zk-Digital Signature Schemes that are post-quantum secure at the Communication Security Lab.

Used to be a Software Engineer working with telecommunication protocols.

also love indulging in poetry, dance and Indian literature.

I’d love to connect with you!

cough
i’m sick :(

(hashed) patient name
date of admission
insurance #
medical history

cough
i'm sick :(

date posted

(hashed) patient name
date of admission
insurance #
medical history

link

Some more real problems with that

Insufficient Consent

- **Patients might consent to data sharing without realizing that this information is permanently visible to anyone who can access the blockchain.**

Linkage Attacks

- **An adversary could link seemingly unrelated data points across different transactions and external databases to de-anonymize patient information.**

Some more real problems with that

Insufficient Consent

- **Patients might consent to data sharing without realizing that this information is permanently visible to anyone who can access the blockchain.**

Linkage Attacks

- **An adversary could link seemingly unrelated data points across different transactions and external databases to de-anonymize patient information.**

the result: stigma, discrimination

A great motivation to apply zero-knowledge proofs!

Agenda

- A brief introduction to ZKP
- ZK Proof of Knowledge
- ZK Arguments of Knowledge (zkSNARKS)
- Zcash
- Current Research in the area

Agenda

- A brief introduction to ZKP
- ZK Proof of Knowledge
- ZK Arguments of Knowledge (zkSNARKS)
- Zcash
- Current Research in the area

A brief intro to ZKPs

- What is a proof?
- What does it mean for a proof to be zero-knowledge?

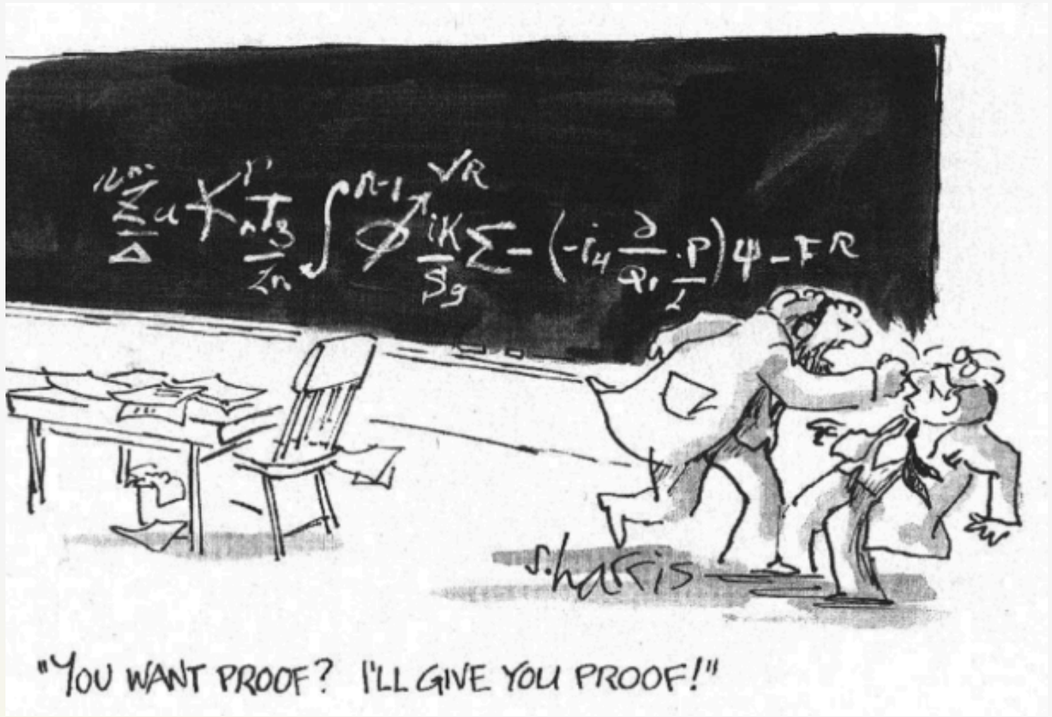


image credit: Princeton University Press

What entails a (classical) proof?

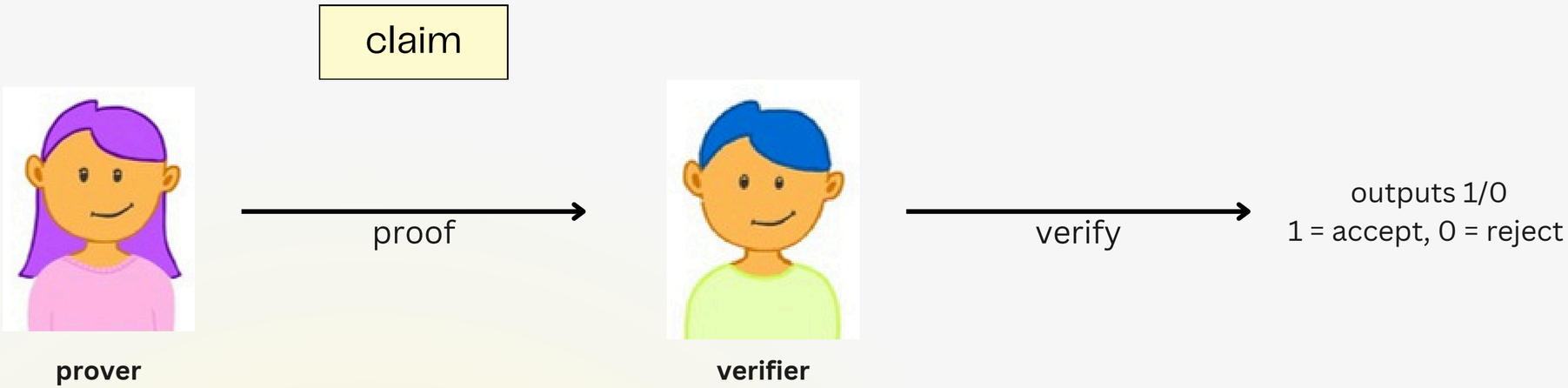
Suppose, N is a product of two large primes, p and q.

prover

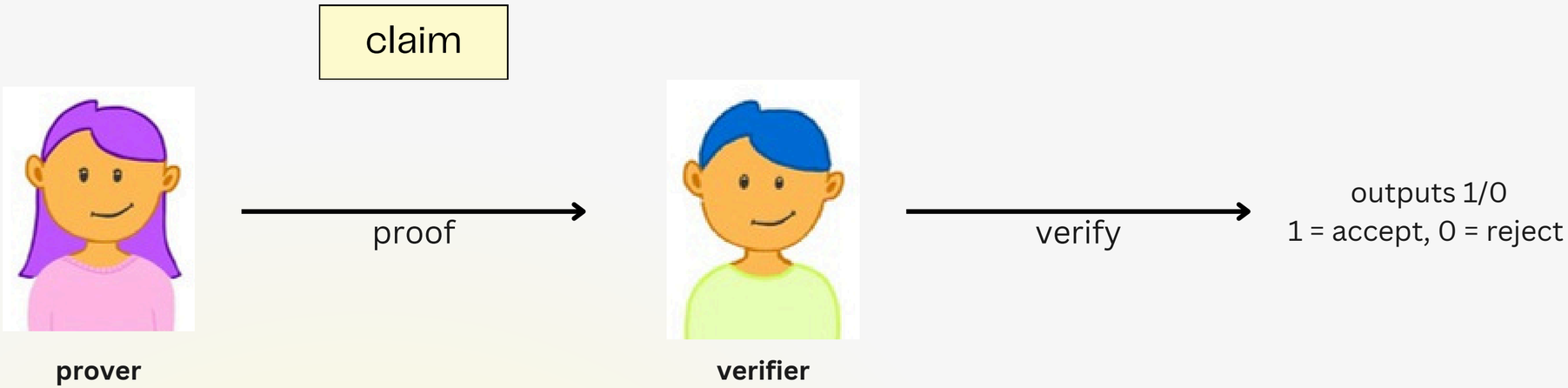
verifier

there is a claim which is an input to both prover and verifier.
For example, the claim is that the prover knows the prime factors of N.

What entails a (classical) proof?

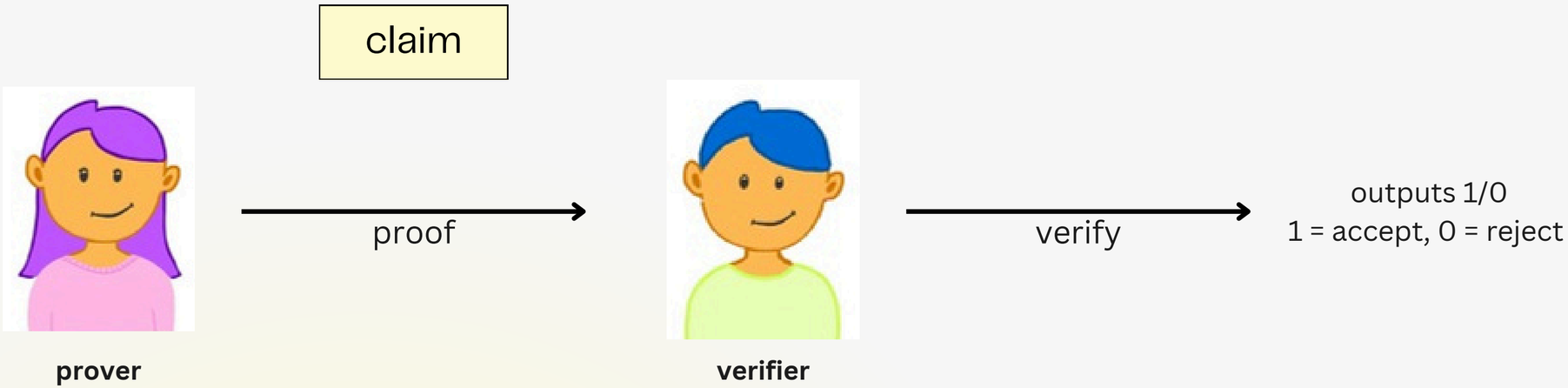


What entails a (classical) proof?



how can the prover convince the verifier of the claim?

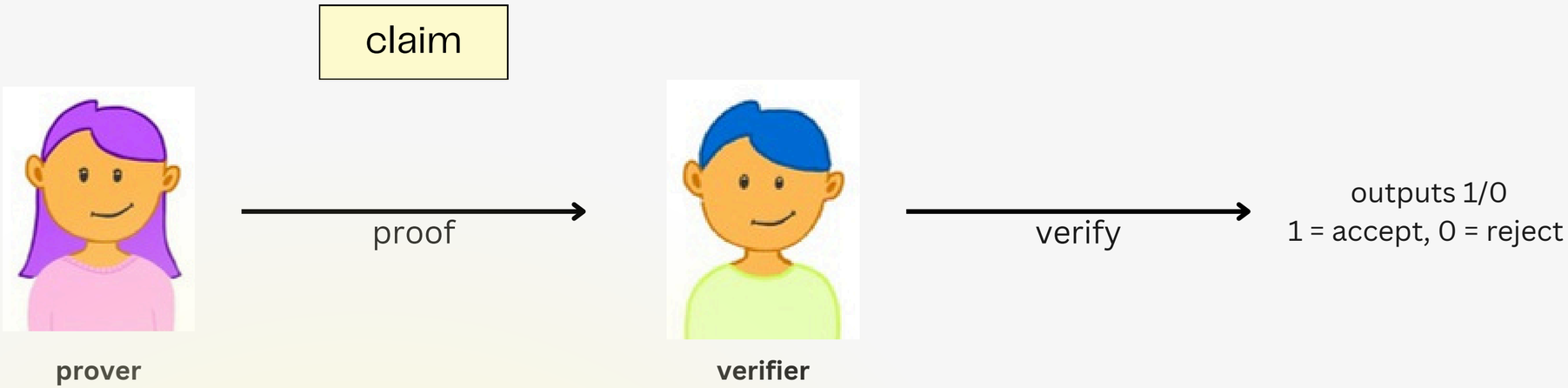
What entails a (classical) proof?



how can the prover convince the verifier of the claim?

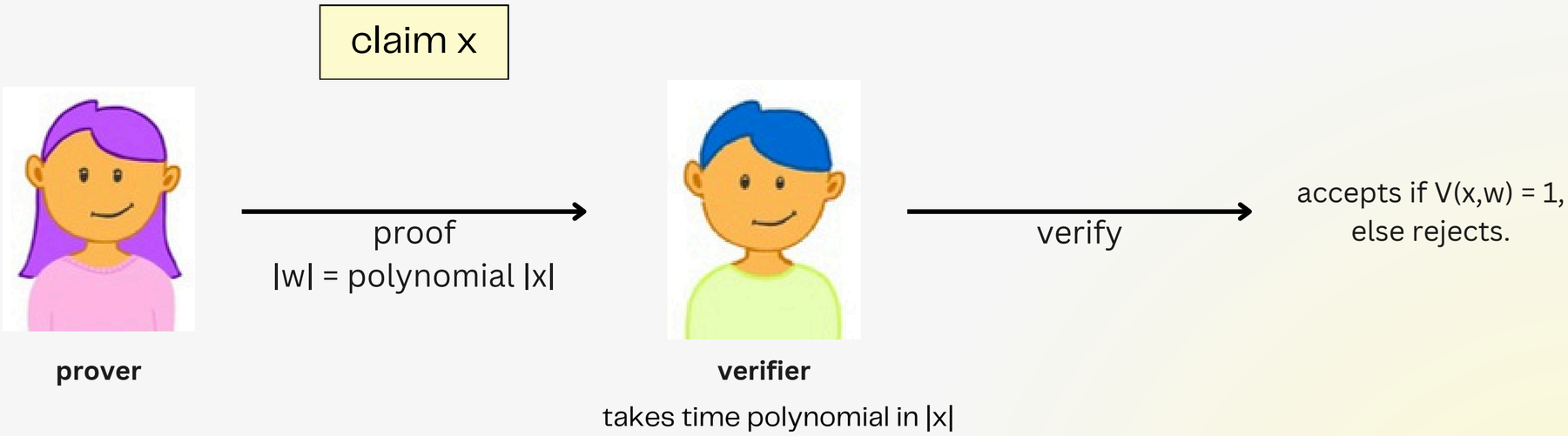
she puts the prime factors {p,q} in the proof.

What entails an efficiently verifiable proof?



a prover works hard to make the proof short
so that the verifier is able to verify it in polynomial time.
This is an NP-proof, which is efficiently verifiable.

What entails an efficiently verifiable proof?

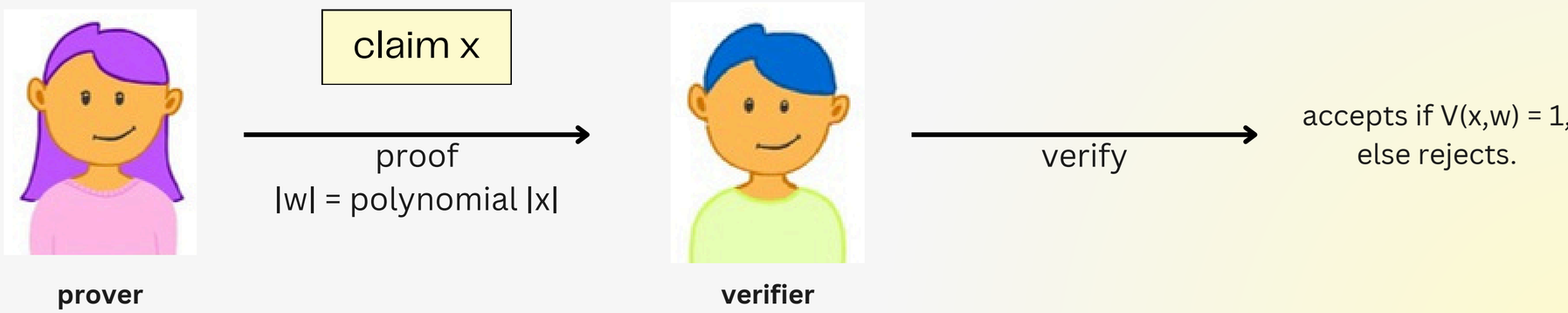


a prover works hard to make the proof short

so that the verifier is able to verify it in polynomial time.

This is an NP-proof, which is efficiently verifiable.

What entails an efficiently verifiable proof?



Def: $\mathcal{L}$ is an **NP**-language (or NP-decision problem), if there is a **poly (|x|) time** verifier V where

- **Completeness [True claims have (short) proofs].**
  if $x \in \mathcal{L}$, there is a **poly(|x|)-long** witness $w \in \{0,1\}^*$ s.t. $V(x,w) = 1$.
- **Soundness [False theorems have no proofs}.**
  if $x \notin \mathcal{L}$, there is no witness. That is, for all $w \in \{0,1\}^*$, $V(x,w) = 0$.

this defenition is presented in the free ZKP MOOC lecture linked here.

But is there another way of making this proof work

claim “I know the factors of N”

proof = {p,q}

verify

accepts if V(x,w) = 1,
else rejects.

such that the prover does not have to give out {p,q}?

in 1989



information theorists and computer scientists

SIAM J. COMPUT.
Vol. 18, No. 1, pp. 186–208, February 1989

012

THE KNOWLEDGE COMPLEXITY OF INTERACTIVE PROOF SYSTEMS*

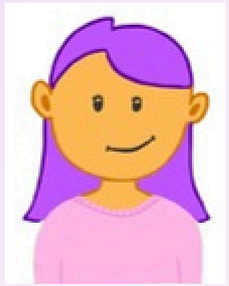
SHAFI GOLDWASSER†, SILVIO MICALI†, AND CHARLES RACKOFF‡

Abstract. Usually, a proof of a theorem contains more knowledge than the mere fact that the theorem is true. For instance, to prove that a graph is Hamiltonian it suffices to exhibit a Hamiltonian tour in it; however, this seems to contain more knowledge than the single bit Hamiltonian/non-Hamiltonian.

In this paper a computational complexity theory of the "knowledge" contained in a proof is developed. Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question. Examples of zero-knowledge proof systems are given for the languages of quadratic residuosity and quadratic nonresiduosity. These are the first examples of zero-knowledge proofs for languages not known to be efficiently recognizable.

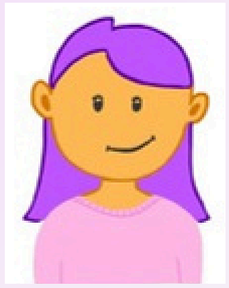
Key words. cryptography, zero knowledge, interactive proofs, quadratic residues

AMS(MOS) subject classifications. 68Q15, 94A60



“I could prove it if I felt like it”

“you would need to convince me of that”



“I could prove it if I felt like it”

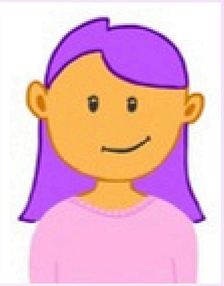
“you would need to convince me of that”

at the end, the verifier does not learn the NP proof

ZK Proof of knowledge

Interaction: prover and verifier engage in a non-trivial interaction.

Randomness: verifier runs a randomness parameter (usually a coin toss) which means there can be an erroneous accept/reject with a small probablity.



“please, please, please,
i’ll prove I’m right”

A simple real-world example

“I know how many coins are in the piggy bank”

prover

A simple real-world example

prover

“I know how many coins are in the piggy bank”

flips a coin to decide whether to add a coin or do nothing. Heads indicates adding a coin, tales indicates doing nothing.

The coin flip results in heads, which prompts the verifier to add a coin to the bank.

“I just added a coin to the lot.”

verifier

A simple real-world example

prover

“I know how many coins are in the piggy bank”

flips a coin to decide whether to add a coin or do nothing. Heads indicates adding a coin, tales indicates doing nothing.

The coin flip results in heads, which prompts the verifier to add a coin to the bank.

“I just added a coin to the lot.”

verifier

prover

Counts the number of coins in the piggy bank.

“You’re telling the truth.”

A simple real-world example

prover

“I know how many coins are in the piggy bank”

flips a coin to decide whether to add a coin or do nothing. Heads indicates adding a coin, tales indicates doing nothing.

The coin flip results in heads, which prompts the verifier to add a coin to the bank.

“I just added a coin to the lot.”

verifier

prover

Counts the number of coins in the piggy bank.

“You’re telling the truth.”

you’re correct, but what if you just guessed that? Answer me again…

verifier

A simple real-world example

prover

“I know how many coins are in the piggy bank”

flips a coin to decide whether to add a coin or do nothing. Heads indicates adding a coin, tales indicates doing nothing.
The coin flip results in heads, which prompts the verifier to add a coin to the bank.
“I just added a coin to the lot.”

verifier

prover

Counts the number of coins in the piggy bank.
“You’re telling the truth.”

you’re correct, but what if you just guessed that? Answer me again…

verifier

for the proof to be accepted, the verifier should be convinced in polynomial time that the prover indeed knows how many coins are in the piggy bank. But also notice that the verifier never learns how many coins there are, at any given interaction.

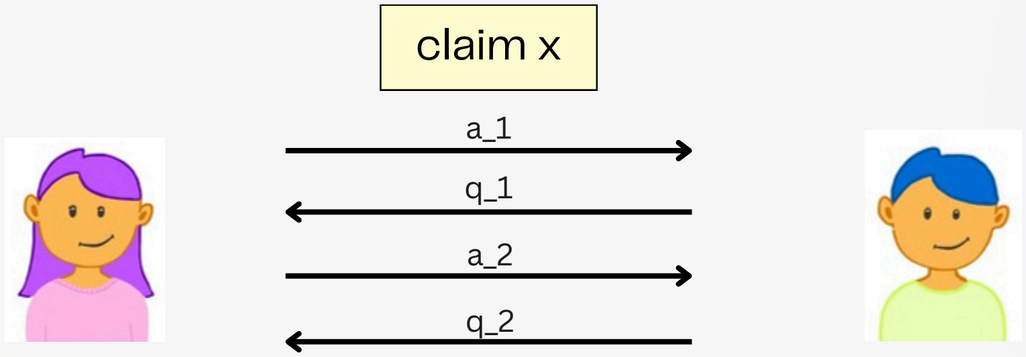
What did we learn from this?

For a true statement, for any verfier,

What a verifer computes after an interaction **=** **What a verifer should be able to compute before an interaction with the prover.**

Which is to say: it learns nothing new from the interactions.

A definition for interactive Zero Knowledge Proofs



Def: (P, V) is an interactive proof for L, if V is probabilistic poly ($|x|$) time &

- **Completeness**: If $x \in \mathcal{L}$, V always accepts.
- **Soundness:** If $x \notin \mathcal{L}$, for all cheating prover strategy, V will not accept except with negligible probability.

Also important to note:

For a dishonest prover, the probability of the verifier accepting their proof is negligible in the length of the claim.

Recap

ZKPs
• Proof needs to be short. • Proof consists of a secret witness. • Verification must be done in polynomial time. • prover should not learn anything new after an interaction.

These ZK proofs of knowledge can be translated into zkSNARKS!

Vitalik thinks zkSNARKs are cool!

SNARK: a **succinct** proof that a statement is true.

Example claim: I know an *m*, where SHA256(*m*) = 0.

a SNARK would make the proof for this very short and fast.
very importantly, it is also non-interactive.

a zkSNARK: preserves zero knowledge.

the proof is short and fast to verify, and also reveals nothing about *m*.

the construction of zkSNARKs are beyond the scope of this presentation, but resources for self-study are pointed to.

Applications of SNARKs in Blockchain

zkRollups (scalability)

- Batch multiple transactions off-chain and then roll them up into a single transaction.
- For each batch of transactions, a zkSNARK proof is generated.
- This single transaction is then published onto the blockchain, significantly improves the number of transactions processed by second while maintaining privacy.

zkBridge (proof of consensus)

- Facilitates the transfer of assets and data between different blockchain networks by providing a secure mechanism to validate and execute transactions across chains.
- Generates a zkp that validates the asset lockup on the source blockchain.
- The zkBridge contract on the target blockchain verifies the proof to ensure the assets' validity.

These applications are not necessarily zero-knowledge!

(but they NEED to be non-interactive)

Applications of zkSNARKs in Blockchain

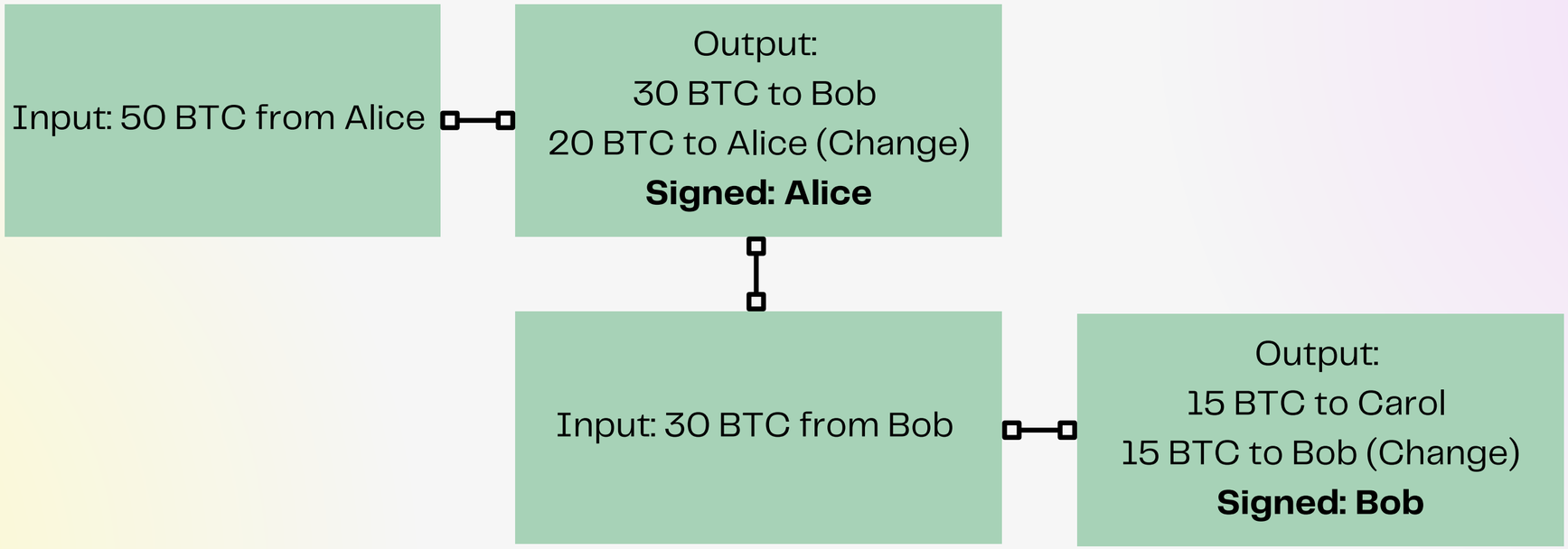
Private transactions

- Everything that a regular blockchain does, but in a privacy-preserving way!
- Zcash, Aleo

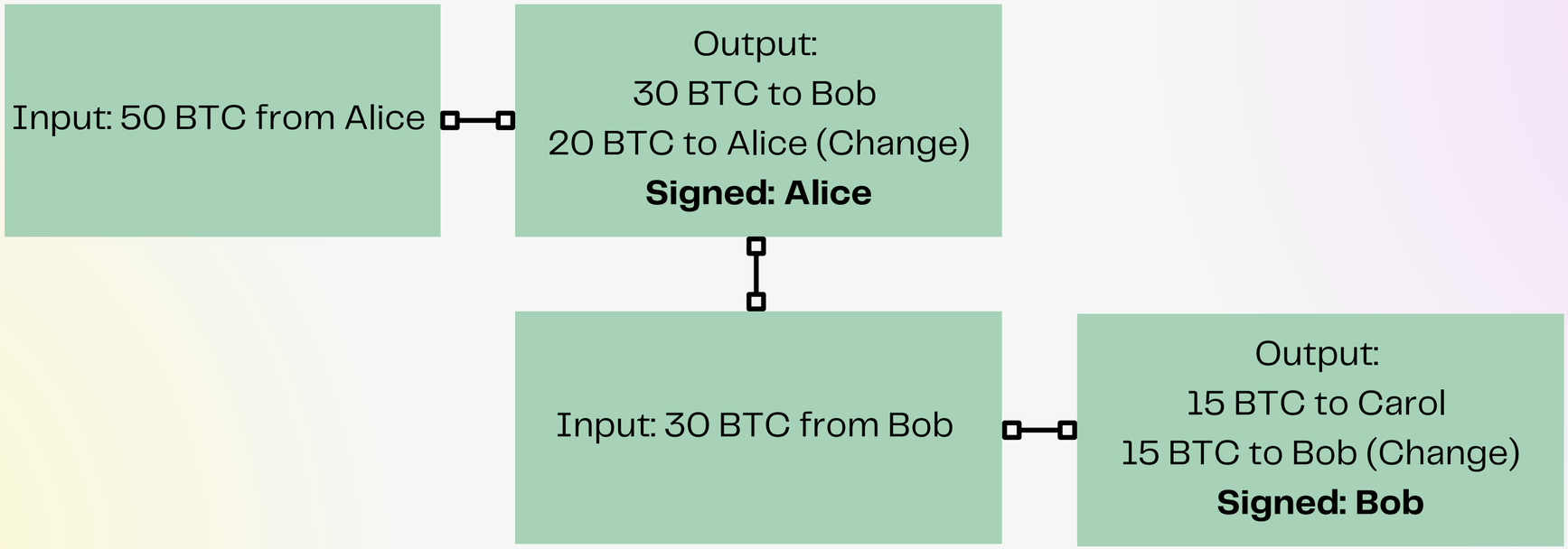
Compliance

- To make technology comply with policies, it is possible to use zkSNARKs to verify if at a given time, a certain protocol complies with the neccessary policies.
- Espresso

A quick recap of what a bitcoin transaction chain looks like



A quick recap of what a bitcoin transaction chain looks like



it is easy to backtrace to the owner (Alice)

Zcash

- a fork of bitcoin with optional anonymity features.
- it is aimed at de-linking senders and recievers in a transaction.
- As with Bitcoin, Zcash coins are sent to addresses, which can be thought of as public keys. Only the owner of the corresponding private key can generate a new transaction to transfer the coins.

Zcash

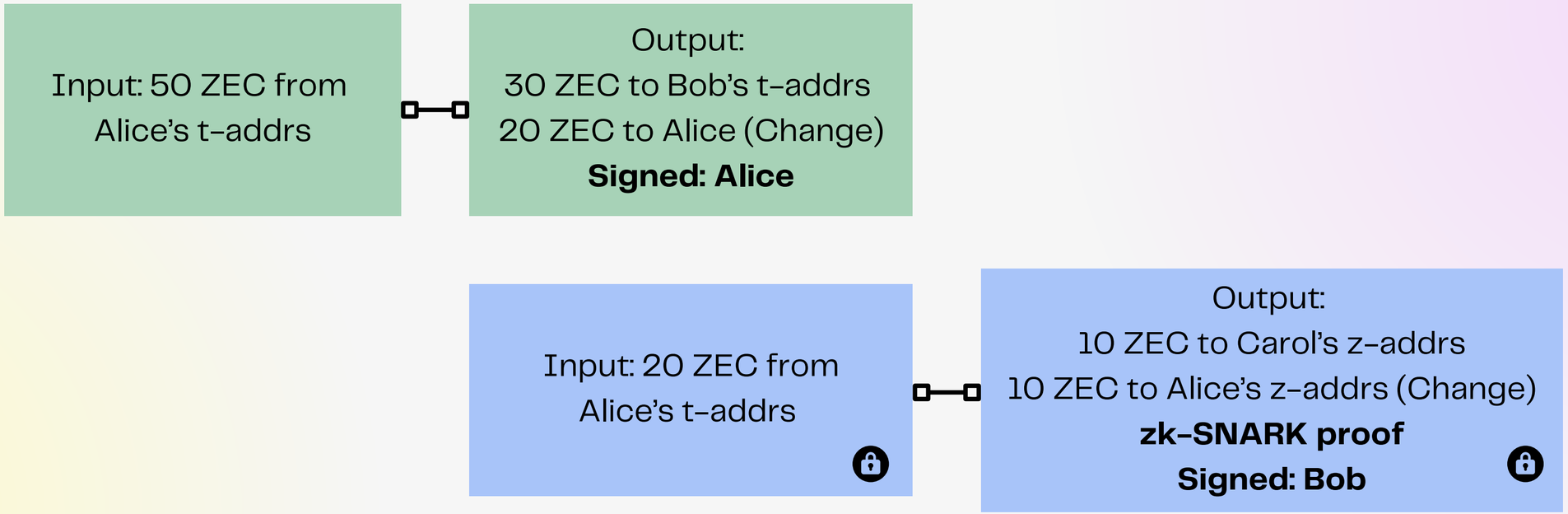
- a fork of bitcoin with optional anonymity features.
- it is aimed at de-linking senders and recievers in a transaction.
- As with Bitcoin, Zcash coins are sent to addresses, which can be thought of as public keys. Only the owner of the corresponding private key can generate a new transaction to transfer the coins.

two classes of addresses

transparent

shielded

a ZCash transaction

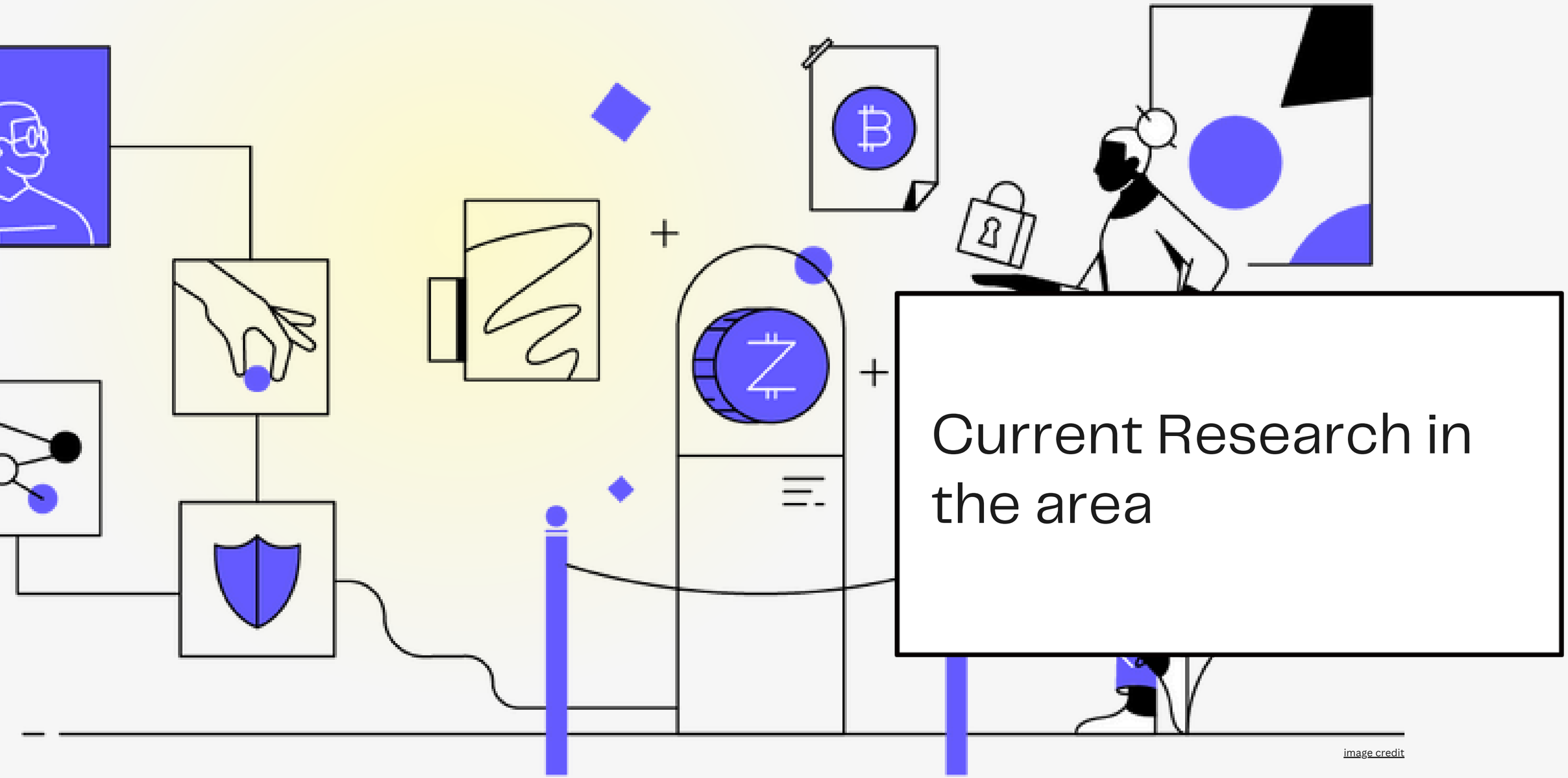


Alice to Carol transaction remains private.

Limitations with Zcash

1. High computational costs of shielded transactions: takes between 30s - 1 min on a modern desktop computer. This discourages users from using shielded transactions, leading to most Zcash transactions not involving shielded addresses.
2. Linkability of Round-Trip Transactions: a pattern of transactions where coins are sent from a transparent address to a shielded address back to a transparent address, often with the same amount. This enables linkability. A paper [1] identified 31% of all coins send to shielded addresses do this.
3. Most cryptocurrency exhanges that trade ZCash only accept transparent addresses.

[1] https://arxiv.org/pdf/1712.01210



Current Research in the area

image credit

Implementation	length of proof	verification time	post-quantum secure?
Groth16	~200 B	1.5 ms	no
PLONK	~400 B	3 ms	no
STARK	> 100KB	smallest circuit: 100ms largest circuit: 10h	Yes
Aurora	130 KB	10s	Yes
Preon(digital signature)	598kB	115s	Yes
Polaris	**204kB**	**13.3ms** (projected)	**Yes**

safe to say that it is vast

1. **implementing post-quantum cryptography algorithms.**
2. **Optimizations in prover and verification time.**
3. **fast implementation of zk-signatures (hash-based ones are concerningly slow).**
4. **using GPUs to accelerate polynomial commitments (an intermediary computation essential for zk-SNARKs) - also what I work with.**

If you want to get started with Zero-knowledge research, here are some readings.

This MOOC is taught by very smart professors that were the foundation of the ZK community as we know it.

zero-knowledge canon parts 1 & 2 by a16zcrypto

Do number theory/algebra when you can

also: zk-learning.org
starkware.co

We talked about

ZKPs

introduced what SNARKs are

ZCash

Current zkSNARK implementations & future scope

what are the main properties that a ZKP must satisfy?

what are the main properties that a ZKP must satisfy?

Completeness: ensures that if the statement is true, the verifier will be convinced by the prover.

Soundness: ensures that if the statement is false, no dishonest prover can convince the verifier otherwise.

Zero-knowledge: ensures that no information other than the validity of the statement is revealed.

a user linkability problem was presented at the beginning, can you think of a way in which zkSNARKs can solve this problem?

a user linkability problem was presented at the beginning, can you think of a way in which zkSNARKs can solve this problem?

Instead of publishing patient data on the blockchain, transactions can be computed off-chain. After a whole bunch of these are computed off-chain, we can summarise them and present a zero knowledge proof of such a summary to the blockchain.

this is kind of what a zkrollup is!

What are some of the challenges associated with implementing zero-knowledge proofs in practical applications?

What are some of the challenges associated with implementing zero-knowledge proofs in practical applications?

computational cost of generating and verifying proofs
ensuring the scalability of the proof system for large-scale applications.

for example, VISA handles upto 24,000 transactions per second under optimal conditions. we have a long way to go.

Thank you. Questions?