The Hunting of the zk-SNARK

CMI Student Seminars

Naman Kumar

August 15, 2022

Outline

Introduction

What is this talk about?

- A primitive in cryptography is a set of algorithms used to build **protocols**

What is this talk about?

- A primitive in cryptography is a set of algorithms used to build **protocols**
- We will be talking about one such primitive that has been of great interest in the past decade

What is this talk about?

- A primitive in cryptography is a set of algorithms used to build **protocols**
- We will be talking about one such primitive that has been of great interest in the past decade

zk-SNARK

A **Z**ero-**K**nowledge, **S**uccinct **N**on-interactive **AR**gument of **K**nowledge

What is this talk about?

- A primitive in cryptography is a set of algorithms used to build **protocols**
- We will be talking about one such primitive that has been of great interest in the past decade

zk-SNARK

A **Z**ero-**K**nowledge, **S**uccinct **N**on-interactive **AR**gument of **K**nowledge

- This talk is about what these words mean

What does the name of this talk mean?

- Cryptographers love to use puns and pop-culture references to name their work

What does the name of this talk mean?

- Cryptographers love to use puns and pop-culture references to name their work
- Poem 'The Hunting of the SNARK' published by Lewis Carroll in 1876

What does the name of this talk mean?

- Cryptographers love to use puns and pop-culture references to name their work
- Poem 'The Hunting of the SNARK' published by Lewis Carroll in 1876
- Also the title of a paper by Goldwasser et. al. in 2014

Zero-Knowledge

What is a Zero-Knowledge Proof?

- Consider a setup with a prover **Alice** and a verifier **Bob**

What is a Zero-Knowledge Proof?

- Consider a setup with a prover **Alice** and a verifier **Bob**
- **Alice** possesses some secret information, usually some kind of statement

What is a Zero-Knowledge Proof?

- Consider a setup with a prover **Alice** and a verifier **Bob**
- **Alice** possesses some secret information, usually some kind of statement

- **Bob** wants to verify that this secret information is indeed true

What is a Zero-Knowledge Proof?

- Consider a setup with a prover **Alice** and a verifier **Bob**
- **Alice** possesses some secret information, usually some kind of statement

- **Bob** wants to verify that this secret information is indeed true
- However, Alice wants to make sure that Bob does not gain any other knowledge apart from the truth of the statement!

What is a Zero-Knowledge Proof?

- Consider a setup with a prover **Alice** and a verifier **Bob**
- **Alice** possesses some secret information, usually some kind of statement

- **Bob** wants to verify that this secret information is indeed true
- However, Alice wants to make sure that Bob does not gain any other knowledge apart from the truth of the statement!
- Can do this using a zero-knowledge proof (ZKP)

What is a Zero-Knowledge Proof?

Formally,

- Alice (usually) has unlimited computational power

What is a Zero-Knowledge Proof?

Formally,

- Alice (usually) has unlimited computational power
- Bob is a **PPT** (probabilistic polynomial time) machine

What is a Zero-Knowledge Proof?

Formally,

- Alice (usually) has unlimited computational power
- Bob is a **PPT** (probabilistic polynomial time) machine
- Her **statement** is usually whether some NP problem is a yes- or a no- instance

What is a Zero-Knowledge Proof?

Formally,

- Alice (usually) has unlimited computational power
- Bob is a **PPT** (probabilistic polynomial time) machine
- Her **statement** is usually whether some NP problem is a yes- or a no- instance
- We also want **protection against a dishonest Alice**

What is a Zero-Knowledge Proof?

In total, we want essentially three properties:

What is a Zero-Knowledge Proof?

In total, we want essentially three properties:

Completeness

If Alice's information is indeed true, we want Bob to always be convinced by it.

What is a Zero-Knowledge Proof?

In total, we want essentially three properties:

Completeness

If Alice's information is indeed true, we want Bob to always be convinced by it.

Soundness

Alice can only convince Bob of the truth of the statement if it is in fact true.

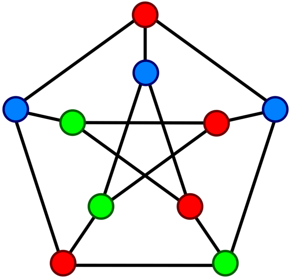
What is a Zero-Knowledge Proof?

Zero-Knowledge

Bob does not know anything other than the truth of the statement.

Formal example: He does not learn the witness or any information about it whatsoever.

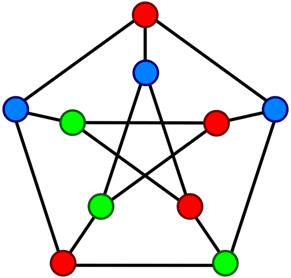
A first Zero-Knowledge Proof



A 3-colorable graph.

- Consider the problem of **graph 3-coloring**

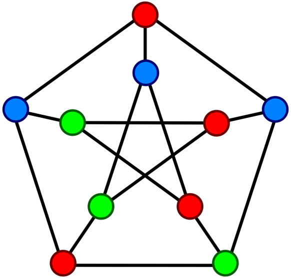
A first Zero-Knowledge Proof



A 3-colorable graph.

- Consider the problem of **graph 3-coloring**
- This problem is NP-hard

A first Zero-Knowledge Proof



A 3-colorable graph.

- Consider the problem of **graph 3-coloring**
- This problem is NP-hard
- We will show a zero-knowledge proof for it

The Setup

- Alice and Bob agree on a graph $G = (V, E)$ beforehand

The Setup

- Alice and Bob agree on a graph $G = (V, E)$ beforehand
- Bob does not know whether G is 3-colorable

The Setup

- Alice and Bob agree on a graph $G = (V, E)$ beforehand
- Bob does not know whether G is 3-colorable
- Alice can figure it out with her unlimited computational power

The Setup

- Alice and Bob agree on a graph $G = (V, E)$ beforehand
- Bob does not know whether G is 3-colorable
- Alice can figure it out with her unlimited computational power
- If $G \in 3Color$, Alice can cook up witnesses

The Algorithm

- Alice chooses a witness for the graph (this witness may or may not be correct) and colors it.
- Bob asks Alice for some edge e.
- Alice sends e along with the node colorings to Bob.
- If the colors are different, Bob accepts, otherwise he rejects.
- Alice permutes the colors of the graph, and they repeat the above until Bob either rejects or is ‘satisfied’.

Analysis - Completeness

- If Alice does indeed have a witness, Bob will **never reject**

Analysis - Completeness

- If Alice does indeed have a witness, Bob will **never reject**
- He will **always** find that the two edges have different colors

Analysis - Completeness

- If Alice does indeed have a witness, Bob will **never reject**
- He will **always** find that the two edges have different colors
- Alice can **always** convince him regardless of how many iterations he requires before he is satisfied

Analysis - Completeness

- If Alice does indeed have a witness, Bob will **never reject**
- He will **always** find that the two edges have different colors
- Alice can **always** convince him regardless of how many iterations he requires before he is satisfied

The proof is complete.

Analysis - Soundness

- Suppose there is no witness. Then for all purposes Alice has just randomly colored the graph.

Analysis - Soundness

- Suppose there is no witness. Then for all purposes Alice has just randomly colored the graph.
- The probability that Alice cheated and was not caught is at most $\frac{|E|-1}{|E|}$

Analysis - Soundness

- Suppose there is no witness. Then for all purposes Alice has just randomly colored the graph.
- The probability that Alice cheated and was not caught is at most $\frac{|E|-1}{|E|}$
- After n iterations, the probability she was not caught is

$$\left(\frac{|E|-1}{|E|}\right)^n$$

Analysis - Soundness

- Suppose there is no witness. Then for all purposes Alice has just randomly colored the graph.
- The probability that Alice cheated and was not caught is at most $\frac{|E|-1}{|E|}$
- After n iterations, the probability she was not caught is

$$\left(\frac{|E|-1}{|E|}\right)^n$$

- We can make this probability **as small as we want**

Analysis - Soundness

- Suppose there is no witness. Then for all purposes Alice has just randomly colored the graph.
- The probability that Alice cheated and was not caught is at most $\frac{|E|-1}{|E|}$
- After n iterations, the probability she was not caught is

$$\left(\frac{|E|-1}{|E|}\right)^n$$

- We can make this probability **as small as we want**
- Bob can be sure to any reasonable degree just by increasing the number of iterations

Analysis - Soundness

- Suppose there is no witness. Then for all purposes Alice has just randomly colored the graph.
- The probability that Alice cheated and was not caught is at most $\frac{|E|-1}{|E|}$
- After n iterations, the probability she was not caught is

$$\left(\frac{|E|-1}{|E|}\right)^n$$

- We can make this probability **as small as we want**
- Bob can be sure to any reasonable degree just by increasing the number of iterations

The proof is sound.

Analysis - Zero-Knowledge

- Formal proof is complex

Analysis - Zero-Knowledge

- Formal proof is complex
- Basic idea comes from **randomness**

Analysis - Zero-Knowledge

- Formal proof is complex
- Basic idea comes from **randomness**
- The permutation of colors after every step is **random**

Analysis - Zero-Knowledge

- Formal proof is complex
- Basic idea comes from **randomness**
- The permutation of colors after every step is **random**
- Bob can gain *no information about the witness* since every time the edges colors are randomly permuted

Analysis - Zero-Knowledge

- Formal proof is complex
- Basic idea comes from **randomness**
- The permutation of colors after every step is **random**
- Bob can gain *no information about the witness* since every time the edges colors are randomly permuted

The proof is zero-knowledge.

Natural Question

Most people after hearing this protocol ask the question:

What if Alice simply gives Bob the wrong edge? Or worse, what if Alice just lies?

Natural Question

Most people after hearing this protocol ask the question:

What if Alice simply gives Bob the wrong edge? Or worse, what if Alice just lies?

- This question is **not** about the security of the proof, it is about the security of the *setup*

Natural Question

Most people after hearing this protocol ask the question:

What if Alice simply gives Bob the wrong edge? Or worse, what if Alice just lies?

- This question is **not** about the security of the proof, it is about the security of the *setup*
- There are **secure computation** protocols to make sure that this is possible

Natural Question

Most people after hearing this protocol ask the question:

What if Alice simply gives Bob the wrong edge? Or worse, what if Alice just lies?

- This question is **not** about the security of the proof, it is about the security of the *setup*
- There are **secure computation** protocols to make sure that this is possible
- For example, if the graph is held by an honest third party, there is no problem

Natural Question

Most people after hearing this protocol ask the question:

What if Alice simply gives Bob the wrong edge? Or worse, what if Alice just lies?

- This question is **not** about the security of the proof, it is about the security of the *setup*
- There are **secure computation** protocols to make sure that this is possible
- For example, if the graph is held by an honest third party, there is no problem
- A third party is not required, however. It can be done without it.

Non-Interactivity

Non-Interactivity

- The previous proof was *interactive*

Non-Interactivity

- The previous proof was *interactive*
- Can we make the proof **single-message**?

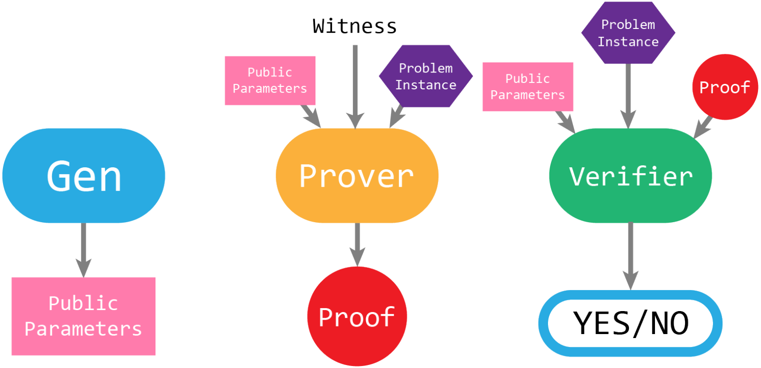
Non-Interactivity

- The previous proof was *interactive*
- Can we make the proof **single-message**?
- Alice sends Bob only **one** message

Non-Interactivity

- The previous proof was *interactive*
- Can we make the proof **single-message**?
- Alice sends Bob only **one** message
- Can he be ‘satisfied’?

Introducing the NARG: Non-interactive Argument



A non-interactive argument.

Non-Interactive Arguments

A Non-Interactive argument is a tuple of 3 algorithms: (Gen, Prove, Verify).

- *Gen*$(1^\kappa) \to (\sigma, \tau)$: Takes input a *security parameter* κ which contains information about how 'satisfied' Bob needs to be, and generates two reference strings, σ and τ which are given to Alice and Bob respectively.

Non-Interactive Arguments

A Non-Interactive argument is a tuple of 3 algorithms: (Gen, Prove, Verify).

- $\mathit{Gen}(1^\kappa) \to (\sigma, \tau)$: Takes input a *security parameter* κ which contains information about how 'satisfied' Bob needs to be, and generates two reference strings, σ and τ which are given to Alice and Bob respectively.
- $\mathit{Prove}(\sigma, x, w) \to \pi$: Takes the reference string, a problem instance x, a witness w, and produces a proof π.

Non-Interactive Arguments

A Non-Interactive argument is a tuple of 3 algorithms: (Gen, Prove, Verify).

- $\mathit{Gen}(1^\kappa) \to (\sigma, \tau)$: Takes input a *security parameter* κ which contains information about how 'satisfied' Bob needs to be, and generates two reference strings, σ and τ which are given to Alice and Bob respectively.
- $\mathit{Prove}(\sigma, x, w) \to \pi$: Takes the reference string, a problem instance x, a witness w, and produces a proof π.
- $\mathit{Verify}(\tau, \pi, x) \to b$: Takes the reference string, the proof and the problem instance, and outputs a bit b corresponding to a success or a failure respectively.

What conditions do NARGs have?

Like ZKPs, NARGs also have some properties attached with them.

Completeness

If w is indeed a witness for x, then Bob will always be convinced.

This is standard.

What conditions do NARGs have?

Here's where things get fun - we now start thinking like cryptographers, and consider an *adversarial* model of soundness.

Soundness (Non-Adaptive)

Let our language be L and consider an adversary $\mathcal{A}$ to which we give $x \notin L$ and σ. Then $\mathcal{A}$ cannot produce a proof π which will be verified by Bob.

What conditions do NARGs have?

Soundness (Adaptive)

Let our language be L and consider an adversary $\mathcal{A}$ to which we give and σ. Then $\mathcal{A}$ cannot produce a proof π for some $x \notin L$ of his choice which will be verified by Bob.

These notions are important: they turn up again later.

Knowledge Soundness

Finally, we can add a final condition which turns our NARG into a NARK - A non-interactive argument **of knowledge**.

Knowledge Soundness

Suppose Alice produces a proof *pi* for some statement x. Then there exists an *extractor* $\mathcal{E}$ with equal power as Alice which can produce a witness w for x.
In other words, Alice can only produce a proof if she actually possesses a witness - regardless of whether $x \in L$.

The zk-NARK

We have now defined our primitive, the **zk-NARK**. To reiterate:

- A zk-NARK is an argument system through which Alice can convince Bob of the truth of some statement.

The zk-NARK

We have now defined our primitive, the **zk-NARK**. To reiterate:

- A zk-NARK is an argument system through which Alice can convince Bob of the truth of some statement.
- If the statement is true, she can **always** convince him.

The zk-NARK

We have now defined our primitive, the **zk-NARK**. To reiterate:

- A zk-NARK is an argument system through which Alice can convince Bob of the truth of some statement.
- If the statement is true, she can **always** convince him.
- She only needs to send him a **single message**.

The zk-NARK

We have now defined our primitive, the **zk-NARK**. To reiterate:

- A zk-NARK is an argument system through which Alice can convince Bob of the truth of some statement.
- If the statement is true, she can **always** convince him.
- She only needs to send him a **single message**.
- No adversary can forge a fake proof for a false statement.

The zk-NARK

We have now defined our primitive, the **zk-NARK**. To reiterate:

- A zk-NARK is an argument system through which Alice can convince Bob of the truth of some statement.
- If the statement is true, she can **always** convince him.
- She only needs to send him a **single message**.
- No adversary can forge a fake proof for a false statement.
- Alice actually needs to have some information about the validity of the statement to make a proof.

The zk-NARK

We have now defined our primitive, the **zk-NARK**. To reiterate:

- A zk-NARK is an argument system through which Alice can convince Bob of the truth of some statement.
- If the statement is true, she can **always** convince him.
- She only needs to send him a **single message**.
- No adversary can forge a fake proof for a false statement.
- Alice actually needs to have some information about the validity of the statement to make a proof.
- Bob does not learn anything other than the obvious.

Succinctness

What about the S?

- ZKPs are some of the most exciting developments in cryptography

What about the S?

- ZKPs are some of the most exciting developments in cryptography
- These proofs were first developed in the 1980s

What about the S?

- ZKPs are some of the most exciting developments in cryptography
- These proofs were first developed in the 1980s
- However, they have skyrocketed in popularity in the last decade

What about the S?

- ZKPs are some of the most exciting developments in cryptography
- These proofs were first developed in the 1980s
- However, they have skyrocketed in popularity in the last decade
- The modern development of ZKPs and NARKs began after work by Alessandro Chiesa, professor at UC Berkeley

What about the S?

- ZKPs are some of the most exciting developments in cryptography
- These proofs were first developed in the 1980s
- However, they have skyrocketed in popularity in the last decade
- The modern development of ZKPs and NARKs began after work by Alessandro Chiesa, professor at UC Berkeley
- These proofs carry an additional, ground-breaking property

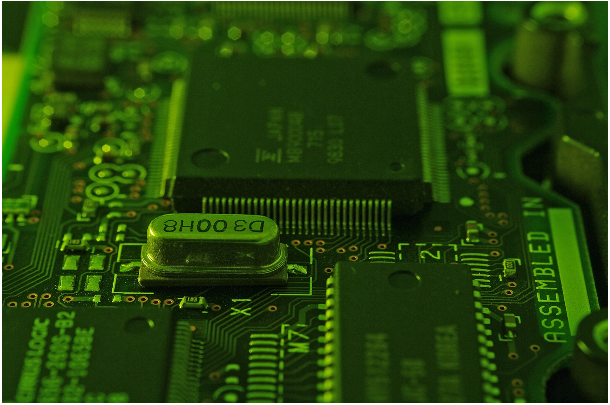
Succinctness

The proof is *short* and finding it is *efficient*. In particular, the size of the proof is usually a parameter like

$$\mathrm{poly}(|x| + |\kappa|)$$

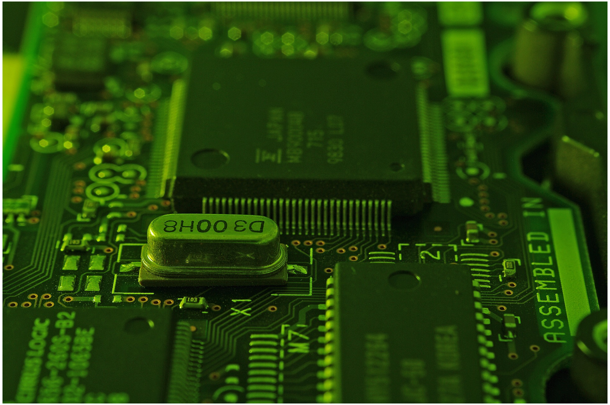
where x is the size of the problem instance and κ is the security parameter. Furthermore, the algorithms are efficient, and work in time polynomial in κ.

What does this mean in practice?



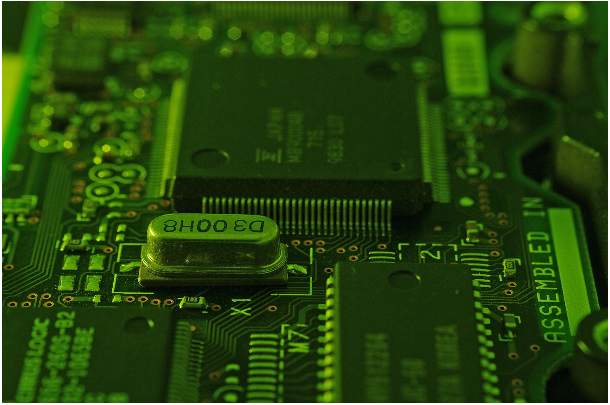
- In practice, this means that ZKPs and NARKs can be verified in **microseconds**

What does this mean in practice?



- In practice, this means that ZKPs and NARKs can be verified in **microseconds**
- The size of proofs is no longer than a few **hundred bits**

What does this mean in practice?



- In practice, this means that ZKPs and NARKs can be verified in **microseconds**
- The size of proofs is no longer than a few **hundred bits**
- These proofs are called SNARKs and (if they possess zero-knowledge) zk-SNARKs

What do we lose?

- Unfortunately, succinct arguments have one big problem

The Statement: **No succinct argument with an adaptive proof of soundness can base its security on a falsifiable assumption (as long as the reduction is black-box).**

What do we lose?

- Unfortunately, succinct arguments have one big problem
- Proved in 2013 by Craig Gentry and Daniel Wichs

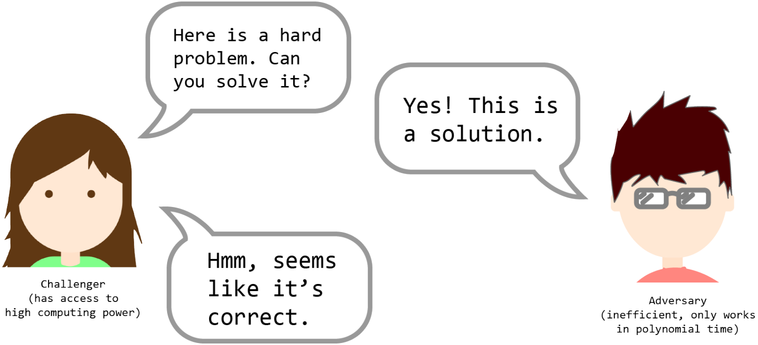
The Statement: **No succinct argument with an adaptive proof of soundness can base its security on a falsifiable assumption (as long as the reduction is black-box).**

What do we lose?

- Unfortunately, succinct arguments have one big problem
- Proved in 2013 by Craig Gentry and Daniel Wichs
- To understand this, let's look at falsifiable assumptions

The Statement: **No succinct argument with an adaptive proof of soundness can base its security on a falsifiable assumption (as long as the reduction is black-box).**

Falsifiable assumptions



A falsifiable assumption can be proved modelled as a game between an adversary and a challenger.

Falsifiable Assumptions

Here's some falsifiable assumptions used in cryptography:

- Prime factorization is hard

Falsifiable Assumptions

Here's some falsifiable assumptions used in cryptography:

- Prime factorization is hard
- Calculating discrete logarithms is hard

Falsifiable Assumptions

Here's some falsifiable assumptions used in cryptography:

- Prime factorization is hard
- Calculating discrete logarithms is hard
- Inverting shifted dot products is hard

Falsifiable Assumptions

Here's some falsifiable assumptions used in cryptography:

- Prime factorization is hard
- Calculating discrete logarithms is hard
- Inverting shifted dot products is hard
- SAT is hard

Falsifiable Assumptions

Here's some falsifiable assumptions used in cryptography:

- Prime factorization is hard
- Calculating discrete logarithms is hard
- Inverting shifted dot products is hard
- SAT is hard
- Separating very similar probability distributions is hard

Unfalsifiable Assumptions

Here's some unfalsifiable assumptions in cryptography:

- Arbitrarily assuming some construction is zero-knowledge

Unfalsifiable Assumptions

Here's some unfalsifiable assumptions in cryptography:

- Arbitrarily assuming some construction is zero-knowledge
- This sucks

Unfalsifiable Assumptions

Here's some unfalsifiable assumptions in cryptography:

- Arbitrarily assuming some construction is zero-knowledge
- This sucks

However, this doesn't mean that our constructions are *not* secure - it just means that we can't prove them assuming (a) a falsifiable assumption, and (b) that we possess no knowledge of the adversary.

Unfalsifiable Assumptions

Here's some unfalsifiable assumptions in cryptography:

- Arbitrarily assuming some construction is zero-knowledge
- This sucks

However, this doesn't mean that our constructions are *not* secure - it just means that we can't prove them assuming (a) a falsifiable assumption, and (b) that we possess no knowledge of the adversary.

New techniques are in development and it's possible we might be able to prove security using them.

Modern Work and Applications

Developments in the last few years

- ZKPs and SNARKs are now a major subfield of cryptography

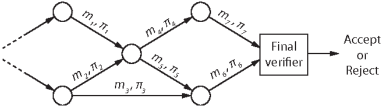
Developments in the last few years

- ZKPs and SNARKs are now a major subfield of cryptography
- We know of several SNARK constructions, such as groth16 or Pinocchio

Developments in the last few years

- ZKPs and SNARKs are now a major subfield of cryptography
- We know of several SNARK constructions, such as groth16 or Pinocchio
- All of these are under active development and new research

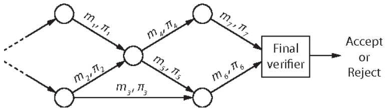
Distributed Computing



Proof-Carrying Data

- SNARKs are used to construct PCD, which is basically a SNARK on drugs

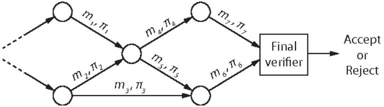
Distributed Computing



Proof-Carrying Data

- SNARKs are used to construct PCD, which is basically a SNARK on drugs
- Consists of a distributed system consisting of data flow and local inputs

Distributed Computing



Proof-Carrying Data

- SNARKs are used to construct PCD, which is basically a SNARK on drugs
- Consists of a distributed system consisting of data flow and local inputs
- Can be used to prevent network fault, securely exchange messages, etc

Blockchain

Bitcoin

ZCash

- Helps in shielding identities of wallet holders and transaction participants

Blockchain

Bitcoin

ZCash

- Helps in shielding identities of wallet holders and transaction participants
- ZCash and Tornado are most common protocols

Blockchain

Bitcoin

ZCash

- Helps in shielding identities of wallet holders and transaction participants
- ZCash and Tornado are most common protocols
- Addresses, tokens, coin history, everything is anonymous yet securely computable

Blockchain

Bitcoin

ZCash

- Helps in shielding identities of wallet holders and transaction participants
- ZCash and Tornado are most common protocols
- Addresses, tokens, coin history, everything is anonymous yet securely computable
- Drawback: doesn't protect against NFT pfp

Gaming

Dark Forest

- Dark Forest is a SNARK-based MMO made by Berkeley undergrads

Gaming

Dark Forest

- Dark Forest is a SNARK-based MMO made by Berkeley undergrads
- The game works on a pattern of planet exploration, where planet coordinates (ID) are private to the player but hashes are public

Gaming

Dark Forest

- Dark Forest is a SNARK-based MMO made by Berkeley undergrads
- The game works on a pattern of planet exploration, where planet coordinates (ID) are private to the player but hashes are public
- Knowing the private ID could be used to attack the player

Gaming

Dark Forest

- Dark Forest is a SNARK-based MMO made by Berkeley undergrads
- The game works on a pattern of planet exploration, where planet coordinates (ID) are private to the player but hashes are public
- Knowing the private ID could be used to attack the player
- Uses SNARK to verify whether hash corresponds to ID

Other Stuff

- Can be used in private auctions to prevent identity leakage

Other Stuff

- Can be used in private auctions to prevent identity leakage
- Can be used in online gaming to prevent the system/company from knowing information about the players

Other Stuff

- Can be used in private auctions to prevent identity leakage
- Can be used in online gaming to prevent the system/company from knowing information about the players
- Used in Privacy-Preserving KYC where no information except existence of a human is known to the company

The Future

- Recent introduction of zk-STARKs

The Future

- Recent introduction of zk-STARKs
- Don't rely on trusted setup and can be instantiated between Alice and Bob

The Future

- Recent introduction of zk-STARKs
- Don't rely on trusted setup and can be instantiated between Alice and Bob
- Also quantum-resilient

The Future

- Recent introduction of zk-STARKs
- Don't rely on trusted setup and can be instantiated between Alice and Bob
- Also quantum-resilient
- Introduced only in 2018, so work is still at the basic level

Thank You

Questions?