

# Introduction to Zero-Knowledge Proofs

Distributed Lab

August 22, 2024



# Plan

## 1 Introduction

- Classical Proofs
- Goal of the course

## 2 Relations. Languages. NP Statements.

- Language of true statements. Examples.
- P and NP Statements

## 3 Interactive Proofs

- Quadratic Residue Interactive Proof
- Completeness and Soundness
- Zero-Knowledge and Honest-Verifier Zero-Knowledge
- Proof of Knowledge

## 4 Fiat-Shamir Heuristic

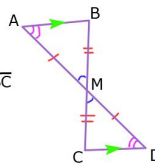
- Cryptographic Oracles
- Fiat-Shamir Transformation

# Introduction

# Classical Proofs

- First proofs you have probably encountered were **geometry proofs**.
- You were given **axioms** and you can prove certain **statements**  $\times$  using them.
- The proof  $\pi$  is a sequence of logical steps that lead from axioms to the statement. Essentially, you have a witness  $w$  that proves the statement.
- Your teacher is the **verifier**  $\mathcal{V}$  who checks your proof, while you are the **prover**  $\mathcal{P}$ .
- This is a **classical proof** and in a sense, it is a **non-interactive proof**.

Given: M is the midpoint of  $\overline{AD}$  and  $\overline{BC}$   
 Prove:  $\overline{AB} \parallel \overline{CD}$



Statements	Reasons
1. Given: M is the midpoint of $\overline{AD}$ and $\overline{BC}$	1. Given
2. $\overline{AM} \cong \overline{MD}$ $\overline{BM} \cong \overline{MC}$	2. Definition of Midpoint
3. $\angle AMB \cong \angle DMC$	3. Vertical Angles Theorem
4. $\triangle ABM \cong \triangle DMC$	4. SAS Thm
5. $\angle A \cong \angle D$	5. CPCTC
6. $\overline{AB} \parallel \overline{CD}$	6. Converse of Alt. Interior Angles Thm

Figure: Geometry proof.

# Motivation

## Note

However, we cannot use such proofs in the digital world.

- Proofs must be verified by computers. Therefore, we need to develop **mathematic framework** to be able to program them.
- This leads to the question: what is **statement**? What is **proof**? What is **witness**? How to formally define them?
- We need to formalize these concepts.

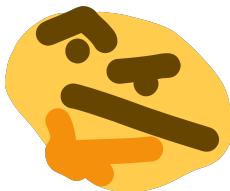


Figure: Hmm...

# The most basic setting

- We have a **prover**  $\mathcal{P}$  and a **verifier**  $\mathcal{V}$ .
- Prover  $\mathcal{P}$  wants to prove some statement  $x$  to the verifier.
- Prover  $\mathcal{P}$  has a **witness**  $w$  that contains all necessary information to prove the statement  $x$ . He sends  $\pi$  as a proof.
- Verifier  $\mathcal{V}$  wants to be convinced that the statement  $x$  is true.

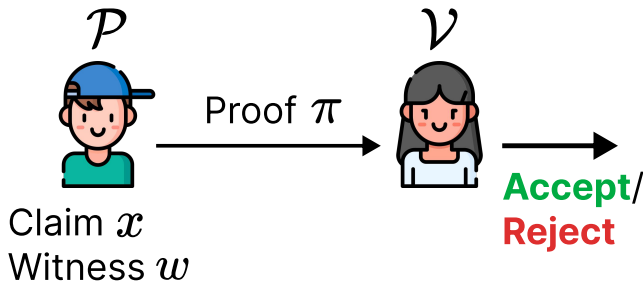


Figure: Typical setup for cryptographic proofs.

# The Goal of SNARKs, STARKs etc.

We will try to solve the following problems:

- **Completeness:** If  $x$  is true,  $\pi$  proves the statement.
- **Soundness:** If  $x$  is false, the prover  $\mathcal{P}$  should not be able to convince the verifier  $\mathcal{V}$  via any  $\pi^*$ .
- **Zero-knowledge:**  $\pi$  does not reveal anything about  $w$ .
- **Argument of knowledge:** Sometimes, the prover  $\mathcal{P}$  should convince the verifier  $\mathcal{V}$  that besides  $x$  is true, he **knows** the witness  $w$ .
- **Succinctness:** The proof should be short, ideally polylogarithmic in the size of the statement ( $|\pi| = \text{polylog}(|x|)$ ) + fast verification.
- **Arithmetization:** We need to convert the statement  $x$  into some algebraic form + make it relatively universal.

## Note

SNARK, STARK, etc. will solve these problems!

# Example to demonstrate the goal

## Example

Given a hash function  $H : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ ,  $\mathcal{P}$  wants to convince  $\mathcal{V}$  that he knows the preimage  $x \in \{0, 1\}^*$  such that  $H(x) = y$ .

- **Zero-knowledge:** The prover  $\mathcal{P}$  does not want to reveal *anything* about the pre-image  $x$  to the verifier  $\mathcal{V}$ .
- **Argument of knowledge:** Proving  $y$  has a pre-image is useless.  $\mathcal{P}$  must show he **knows**  $x \in \{0, 1\}^*$  s.t.  $H(x) = y$ .
- **Succinctness:** If the hash function takes  $n$  operations to compute, the proof should be **much** shorter than  $n$  operations. **State-of-art:** size is  $\text{polylog}(n) = O((\log n)^c)$ . Verification time is also typically polylogarithmic (or even  $O(1)$  in some cases).

## Note

But first, let us start with the basics.



# Relations. Languages. NP Statements.

## Definition (Relation)

Given two sets  $\mathcal{X}$  and  $\mathcal{Y}$ , the **relation** is  $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ .

- $\mathcal{X}$  is typically a set of **statements**.
- $\mathcal{Y}$  is a set of **witnesses**.

## Definition (Language of true statements)

Let  $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$  be a relation. We say that a statement  $x \in \mathcal{X}$  is a **true** statement if  $(x, y) \in \mathcal{R}$  for some  $y \in \mathcal{Y}$ , otherwise the statement is called **false**. We define by  $\mathcal{L}_{\mathcal{R}}$  (the language over relation  $\mathcal{R}$ ) the set of all true statements, that is:

$$\mathcal{L}_{\mathcal{R}} = \{x \in \mathcal{X} : \exists y \in \mathcal{Y} \text{ such that } (x, y) \in \mathcal{R}\}.$$

# Language Example #1: Semiprimes

## Example (Product of Two Primes (Semiprimes))

**Claim:** number  $n \in \mathbb{N}$  is the product of two prime numbers  $w = (p, q) \in \mathbb{N} \times \mathbb{N}$ . The **relation** is given by:

$$\mathcal{R} = \{(n, p, q) \in \mathbb{N}^3 : n = p \cdot q \text{ where } p, q \text{ are primes}\}$$

In this particular case, the **language of true statements** is defined as

$$\mathcal{L}_{\mathcal{R}} = \{n \in \mathbb{N} : \exists w = (p, q) \text{ are primes such that } n = p \cdot q\}$$

- **Valid witness #1:**  $n = 15 \in \mathcal{L}_{\mathcal{R}}$ . Witness:  $w = (3, 5)$ .
- **Invalid witness:**  $n = 16 \notin \mathcal{L}_{\mathcal{R}}$ . There is no valid witness.
- **Valid witness #2:**  $n = 50252009 \in \mathcal{L}_{\mathcal{R}}$ . Witness:  $w = (5749, 8741)$ .

**Question:** Is  $n = 27$  a true statement? What about  $n = 26$ ?

# Language Example #2: Square Root

## Reminder

$\mathbb{Z}_N^\times = \{x \in \mathbb{Z}_N : \gcd\{x, N\} = 1\}$ . **Example:**  $\mathbb{Z}_{10}^\times = \{1, 3, 7, 9\}$

## Example

**Claim:** number  $x \in \mathbb{Z}_N^\times$  is a **quadratic residue** modulo  $N$ :

$(\exists w \in \mathbb{Z}_N^\times) : \{x \equiv w^2 \pmod{N}\}$  ( $w$  is **modular square root** of  $x$ ).

**Relation:**  $\mathcal{R} = \{(x, w) \in (\mathbb{Z}_N^\times)^2 : x \equiv w^2 \pmod{N}\}$

**Language:**  $\mathcal{L}_{\mathcal{R}} = \{x \in \mathbb{Z}_N^\times : \exists w \in \mathbb{Z}_N^\times \text{ such that } x \equiv w^2 \pmod{N}\}$ .

**Examples** for  $N = 7$ :

- $4 \in \mathcal{L}_{\mathcal{R}}$  since  $5^2 \equiv 4 \pmod{7}$ .
- $3 \notin \mathcal{L}_{\mathcal{R}}$  since there is no valid witness for 3.

**Question:** Is  $x = 1$  a true statement for  $N = 5$ ? What about  $x = 4$ ?

# NP Statements: Demonstration

Well. . . We are simply going to send witness  $w$  to the verifier  $\mathcal{V}$  and he will check if the statement is true (meaning, whether  $x \in \mathcal{L}_{\mathcal{R}}$ ).

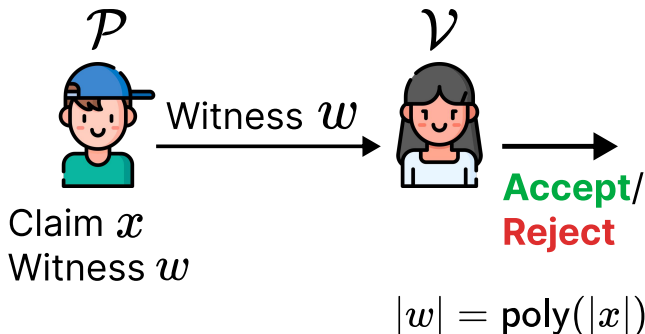


Figure: Typical setup for cryptographic proofs.

# NP Statements

## Definition (P Language)

Problem is in the **P** class if exists a polytime algorithm checking  $x \in \mathcal{L}$ .

## Definition (NP Language)

A language  $\mathcal{L}_{\mathcal{R}}$  belongs to the **NP** class if there exists a polynomial-time verifier  $\mathcal{V}$  such that the following two properties hold:

- **Completeness:** If  $x \in \mathcal{L}_{\mathcal{R}}$ , then there is a witness  $w$  such that  $\mathcal{V}(x, w) = 1$  with  $|w| = \text{poly}(|x|)$ . Essentially, it states that true claims have *short* proofs.
- **Soundness:** If  $x \notin \mathcal{L}_{\mathcal{R}}$ , then for any  $w$  it holds that  $\mathcal{V}(x, w) = 0$ . Essentially, it states that false claims have no proofs.

## Theorem

Any **NP** problem has a zero-knowledge proof.

# Question (aka Motivation)

But can we do better?

Sending witness is... Weird...

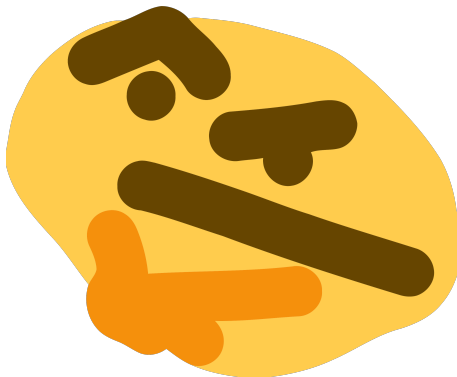


Figure: Hmm... #2

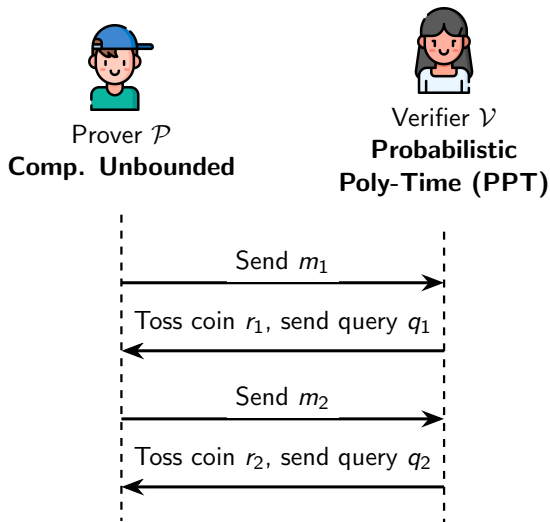
# Interactive Proofs



# Solution!

We add two more ingredients:

- **Interaction:** instead of **passively** receiving the proof, the verifier  $\mathcal{V}$  can **interact** with the prover  $\mathcal{P}$  by sending **challenges** and receiving **responses**.
- **Randomness:**  $\mathcal{V}$  can send random coins (challenges) to the prover, which  $\mathcal{P}$  can use to generate responses.



# Quadratic Residue Interactive Proof

## Problem Statement

- **Statement:**  $x \in \mathcal{L}_{\mathcal{R}}$  where our **language** is defined as:

$$\mathcal{L}_{\mathcal{R}} = \{x \in \mathbb{Z}_N^{\times} : \exists w \in \mathbb{Z}_N^{\times} \text{ such that } x \equiv w^2 \pmod{N}\}$$

- **Witness:**  $w =$  modular square root of  $x$ .

How does  $\mathcal{P}$  and  $\mathcal{V}$  interact? Consider the figure below.

$\mathcal{P}$



1. Sample  $r$  from  $\mathbf{Z}_N$  uniformly
2. Send  $a = r^2 \pmod{N}$

I know  $w$  s.t.  
 $w^2 = x \pmod{N}$

$\mathcal{V}$



Is  $x$  indeed a  
quadr. residue?

# Quadratic Residue Interactive Proof



I know  $w$  s.t.  
 $w^2 = x \pmod{N}$

1. Sample  $r$  from  $\mathbf{Z}_N$  uniformly
  2. Send  $a = r^2 \pmod{N}$
- If I gave you the square root of  $a$  and  $ax$ , you would be convinced that the claim is true, but you learn the witness  $w$ .
  - Instead, I will send you either  $r$  or  $rw$ , but you are to choose!



Is  $x$  indeed a  
quadr. residue?

# Quadratic Residue Interactive Proof

$\mathcal{P}$



I know  $w$  s.t.  
 $w^2 = x \pmod{N}$

1. Sample  $r$  from  $\mathbf{Z}_N$  uniformly
  2. Send  $a = r^2 \pmod{N}$
- If I gave you the square root of  $a$  and  $ax$ , you would be convinced that the claim is true, but you learn the witness  $w$ .
  - Instead, I will send you either  $r$  or  $rw$ , but you are to choose!

$\mathcal{V}$



Is  $x$  indeed a  
quadr. residue?



Ok, I choose random bit  $b$

# Quadratic Residue Interactive Proof

$\mathcal{P}$



I know  $w$  s.t.  
 $w^2 = x \pmod{N}$

1. Sample  $r$  from  $\mathbf{Z}_N$  uniformly
  2. Send  $a = r^2 \pmod{N}$
- If I gave you the square root of  $a$  and  $ax$ , you would be convinced that the claim is true, but you learn the witness  $w$ .
  - Instead, I will send you either  $r$  or  $rw$ , but you are to choose!

$\mathcal{V}$



Is  $x$  indeed a  
quadr. residue?



Ok, I choose random bit  $b$

- If  $b=0$ , send  $z = r$
- If  $b=1$ , send  $z = rw \pmod{N}$

→ Check if  $z^2 = ax^b$

# Quadratic Residue Interactive Proof: Analysis

## Interactive Protocol

- 1  $\mathcal{P}$  samples  $r \xleftarrow{R} \mathbb{Z}_N^\times$  and sends  $a = r^2$  to  $\mathcal{V}$ .
- 2  $\mathcal{V}$  sends a random bit  $b \in \{0, 1\}$  to  $\mathcal{P}$ .
- 3  $\mathcal{P}$  sends  $z = r \cdot w^b$  to  $\mathcal{V}$ .
- 4  $\mathcal{V}$  accepts if  $z^2 = a \cdot x^b$ , otherwise it rejects.
- 5 Repeat  $\lambda \in \mathbb{N}$  times.

## Lemma

*The aforementioned protocol is **complete** and **sound**.*

**Completeness.** If  $b = 0$ , then  $z = r$  and thus  $z^2 = r^2 = a$ , check passes. If  $b = 1$ , then  $z = rw$  and thus  $z^2 = r^2 w^2 = ax$ , check passes.

# Quadratic Residue Interactive Proof: Analysis

**Soundness.** The main reason why the protocol is sound is inscribed in the theorem below.

## Theorem

*For any prover  $\mathcal{P}^*$  with  $x \notin \mathcal{L}_{\mathcal{R}}$ , the probability of  $\mathcal{V}$  accepting the proof is at most  $1/2$ .*

**Corollary.** After repeating the protocol  $\lambda$  times, we have

$$\Pr[\mathcal{V} \text{ accepts after } \lambda \text{ rounds}] \leq \frac{1}{2^\lambda} = \text{negl}(\lambda).$$

Thus, we showed both **completeness** and **soundness** of the protocol.

# Interactive Protocol Definition

$\langle \mathcal{P}, \mathcal{V} \rangle(x)$  reads as “interaction between  $\mathcal{P}$  and  $\mathcal{V}$  on the statement  $x$ ”.

## Definition

A pair of algorithms  $(\mathcal{P}, \mathcal{V})$  is called an **interactive proof** for a language  $\mathcal{L}_{\mathcal{R}}$  if  $\mathcal{V}$  is a polynomial-time verifier and the following two properties hold:

- **Completeness:** For any  $x \in \mathcal{L}_{\mathcal{R}}$ ,  $\Pr[\langle \mathcal{P}, \mathcal{V} \rangle(x) = \text{accept}] = 1$ .
- **Soundness:** For any  $x \notin \mathcal{L}_{\mathcal{R}}$  and for any prover  $\mathcal{P}^*$ , we have

$$\Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle(x) = \text{accept}] \leq \text{negl}(\lambda)$$

## Definition

The class of **interactive proofs (IP)** is defined as:

$$\text{IP} = \{\mathcal{L} : \text{there is an interactive proof } (\mathcal{P}, \mathcal{V}) \text{ for } \mathcal{L}\}.$$



# Zero-Knowledge Informal Definition

## Definition

An interactive proof system  $(\mathcal{P}, \mathcal{V})$  is called **zero-knowledge** if for any polynomial-time verifier  $\mathcal{V}^*$  and any  $x \in \mathcal{L}_{\mathcal{R}}$ , the interaction  $\langle \mathcal{P}, \mathcal{V}^* \rangle(x)$  gives nothing new about the witness  $w$ .

## Definition

The pair of algorithms  $(\mathcal{P}, \mathcal{V})$  is called a **zero-knowledge interactive protocol** if it is *complete*, *sound*, and *zero-knowledge*.

I know witness,  
but I will not show  
you it!



Well, the claim is true,  
but what was the witness  
anyway?!

# Verifier's View

## Question #1

What has the verifier learned during the interaction?

- First things first, he learned that the statement  $x$  is true.
- He also knows queries  $(q_1, \dots, q_\ell)$  and random coins  $(r_1, \dots, r_\ell)$  he tossed (since he is the one who has sent them).
- Moreover, he knows the prover's messages  $(m_1, m_2, \dots, m_\ell)$ .

## Definition

All the conversation that verifier has witnessed is called **verifier's view** and is denoted as

$$\text{view}_V(\mathcal{P}, \mathcal{V}) = (m_1, r_1, q_1, m_2, r_2, q_2, \dots, m_\ell, r_\ell, q_\ell).$$

**Fact:**  $\text{view}_V(\mathcal{P}, \mathcal{V})$  is a **random variable**.

# Verifier's View: Example

## Example

For QN test, set  $N := 3 \times 2^{30} + 1$  (prime number), and  $\mathcal{P}$  wants to convince that  $1286091780 \in \mathcal{L}_R$ . Conversation is the following:

- 1 During the first round,  $\mathcal{P}$  sends 672192003 to  $\mathcal{V}$ .
- 2  $\mathcal{V}$  sends  $b = 0$  to  $\mathcal{P}$ .
- 3  $\mathcal{P}$  sends 2606437826 to  $\mathcal{V}$ .
- 4  $\mathcal{V}$  verifies that indeed  $2606437826^2 \equiv 672192003 \pmod{N}$ .
- 5 During the second round,  $\mathcal{P}$  sends 2619047580 to  $\mathcal{V}$ .
- 6  $\mathcal{V}$  chooses  $b = 1$  and sends to  $\mathcal{P}$ .
- 7  $\mathcal{P}$  sends 1768388249 to  $\mathcal{V}$ .
- 8  $\mathcal{V}$  verifies that  $1768388249^2 \equiv 2619047580 \times 1286091780 \pmod{N}$ .
- 9 Conversation ends.

# Verifier's View: Example

## Example

The **view of the verifier**  $\mathcal{V}$  is the following:

$$\begin{aligned} & \text{view}_{\mathcal{V}}(\mathcal{V}, \mathcal{P})[1286091780] \\ &= (672192003, 0, 2606437826, 2619047580, 1, 1768388249) \end{aligned}$$

- Essentially, this view is the same as you have witnessed.
- You have not learned anything about  $w$  that prover  $\mathcal{P}$  knows.
- The witness was  $w = 3042517305$  and two randomnesses were  $r_1 = 2606437826$  and  $r_2 = 3023142760$ .
- This is a random variable: conversation could be different.

# Zero-Knowledge Formally: Simulation Paradigm

## Question #2

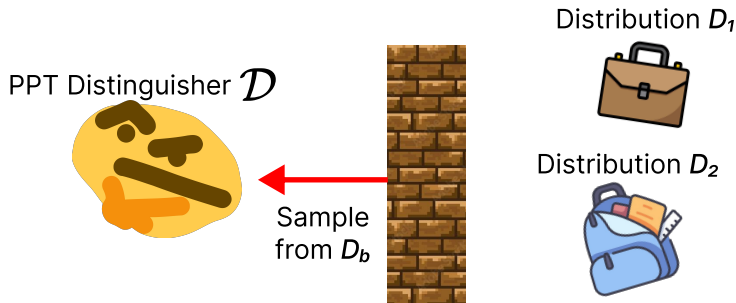
What does it mean that the protocol is zero-knowledge?

- Protocol is zero-knowledge if, given the verifier's view  $\mathcal{V}(\mathcal{P}, \mathcal{V})$ , verifier cannot infer any information about the witness  $w$ .
- What does it mean that verifier  $\mathcal{V}$  learns nothing new? It means that this view could have been simulated by  $\mathcal{V}$  *without even running an interaction*.
- Call the view after the real interaction as **real view**, while the view after the simulation as **simulated view**.

## Note

Such idea of defining the zero-knowledge is called **simulation paradigm** and currently the most widely used way to prove zero-knowledge.

# Computational Indistinguishability



## Definition (Informal Computational Indistinguishability)

$D_1$  and  $D_2$  are **computationally indistinguishable** (denoted by  $D_1 \approx D_2$ ) if for any PPT distinguisher  $\mathcal{D}$ , even after polynomial number  $k$  of samples from  $D_b$  (where  $b \xleftarrow{R} \{0, 1\}$ ), for prediction  $\hat{b}$ :  $\Pr[\hat{b} = b] < \frac{1}{2} + \text{negl}(k)$ .

# Zero-Knowledge Formally (Kind of)

Finally, we are ready to define the **zero-knowledge**.

## Definition (Honest-Verifier Zero-Knowledge (HVZK))

An interactive protocol  $(\mathcal{P}, \mathcal{V})$  is **honest-verifier zero-knowledge (HVZK)** for a language  $\mathcal{L}_{\mathcal{R}}$  there exists a poly-time simulator  $\text{Sim}$  such that for any valid statement  $x \in \mathcal{L}_{\mathcal{R}}$ :

$$\text{view}_{\mathcal{V}}(\mathcal{P}, \mathcal{V})[x] \approx \text{Sim}(x, 1^\lambda)$$

## Definition (Zero-Knowledge (ZK))

An interactive protocol  $(\mathcal{P}, \mathcal{V})$  is **zero-knowledge (ZK)** for a language  $\mathcal{L}_{\mathcal{R}}$  if *for every poly-time verifier  $\mathcal{V}^*$  there exists a poly-time simulator  $\text{Sim}$  such that for any valid statement  $x \in \mathcal{L}_{\mathcal{R}}$ :*

$$\text{view}_{\mathcal{V}^*}(\mathcal{P}, \mathcal{V}^*)[x] \approx \text{Sim}(x, 1^\lambda)$$

# Proof of Knowledge: Why?

Now, the main issue with the above definition is that *we have proven the statement correctness, but we have not proven that the prover **knows** the witness*. These are completely two distinct things!

## Example

Consider the **discrete logarithm relation and language** for a cyclic group  $E(\mathbb{F}_p)$  of order  $r$ :

$$\mathcal{R} = \{(P, \alpha) \in E(\mathbb{F}_p) \times \mathbb{Z}_r : P = [\alpha]G\},$$
$$\mathcal{L}_{\mathcal{R}} = \{P \in E(\mathbb{F}_p) : \exists \alpha \in \mathbb{Z}_r \text{ such that } P = [\alpha]G\}$$

## Question

What does it mean that  $X \in \mathcal{L}_{\mathcal{R}}$ ?

Turns out  $\mathcal{L}_{\mathcal{R}} = E(\mathbb{F}_p)$ , so the proof  $X \in \mathcal{L}_{\mathcal{R}}$  itself is useless.



# Proof of Knowledge: Definition

- 1 The knowledge of witness means that we can **extract** the witness while interacting with the prover.
- 2 Thus, there should be an algorithm called **extractor**  $\mathcal{E}$  which can extract the witness  $w$ .
- 3  $\mathcal{E}$  is given more power than  $\mathcal{V}$  (otherwise, if the protocol is zero-knowledge, we cannot extract  $w$ ).  $\mathcal{E}$  can **rewind** and **call** prover  $\mathcal{P}$  multiple times.
- 4 Sometimes, this is referred to as “extractor  $\mathcal{E}$  uses  $\mathcal{P}$  as an oracle”.

## Definition (Proof of Knowledge)

The interactive protocol  $(\mathcal{P}, \mathcal{V})$  is a **proof of knowledge** for  $\mathcal{L}_{\mathcal{R}}$  if exists a poly-time extractor algorithm  $\mathcal{E}$  such that for any valid statement  $x \in \mathcal{L}_{\mathcal{R}}$ , in expected poly-time  $\mathcal{E}^{\mathcal{P}}(x)$  outputs  $w$  such that  $(x, w) \in \mathcal{R}$ .

# Proof of Knowledge: Example

## Lemma

*The quadratic residue interactive protocol is a proof of knowledge.*

**Proof.** Let us define the extractor  $\mathcal{E}$  for the statement  $x$  as follows:

- ① Run the prover to receive  $a \equiv r^2 \pmod{N}$  ( $r$  is chosen randomly from  $\mathbb{Z}_N^*$ ).
- ② Set verifier's message to  $b = 0$  to get  $z_1 \leftarrow r$ .
- ③ **Rewind** and set verifier's message to  $b = 1$  to get  $z_2 \leftarrow rw \pmod{N}$ .
- ④ Output  $z_2/z_1 \pmod{N}$

The extractor  $\mathcal{E}$  will always output  $w$  if  $x \in \mathcal{L}_{\mathcal{R}}$ . □

# Fiat-Shamir Heuristic

# Cryptographic Oracle

## Definition (Cryptographic Oracle)

Informally, *cryptographic oracle* is simply a function  $\mathcal{O}$  that gives in  $O(1)$  an answer to some typically very hard problem.

## Example (CDH Problem)

Consider the **Computational Diffie-Hellman (CDH)** problem on the cyclic elliptic curve  $E(\mathbb{F}_p)$  of prime order  $r$  with a generator  $G$ .

**Hard Problem:**  $[\alpha\beta]G$  given  $[\alpha]G$  and  $[\beta]G$  where  $\alpha, \beta \in \mathbb{Z}_r$ .

**Oracle:** However, we *could* assume that such problem can be solved in  $O(1)$  by a cryptographic oracle  $\mathcal{O}_{\text{CDH}} : ([\alpha]G, [\beta]G) \mapsto [\alpha\beta]G$ .

This way, we can rigorously prove the security of some cryptographic protocols *even* if the Diffie-Hellman problem is suddenly solved.

# Random Oracle (RO)

One of the most popular cryptographic oracles is the **random oracle**  $\mathcal{O}_R$ .

## Definition (Informal definition of RO)

Suppose someone is inputting  $x$  to the random oracle  $\mathcal{O}_R : \mathcal{X} \rightarrow \mathcal{Y}^a$ . The oracle  $\mathcal{O}_R$  does the following:

- 1 If  $x$  has been queried before, the oracle returns the same value as it returned before.
- 2 If  $x$  has not been queried before, the oracle returns a randomly uniformly sampled value from the output space  $\mathcal{Y}$ .

---

<sup>a</sup>Typically, RO works with a family of functions  $f : \mathcal{X} \rightarrow \mathcal{Y}$ , but we are not going too deep into the details.

## Question

Which very well-known cryptographic object can “serve” as a random oracle?

# Fiat-Shamir Transformation

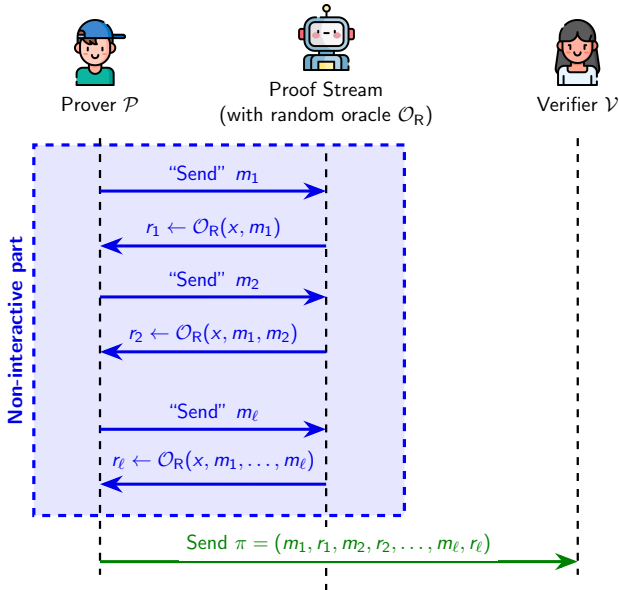
## Statement

**Any** interactive public-coin protocol can be converted into a non-interactive public-coin protocol with preserving completeness, soundness, and zero-knowledge using the random oracle.

One of such transformations is called **Fiat-Shamir heuristic**. Idea:

- 1 If all what  $\mathcal{V}$  does is sending uniformly random values, this is an overkill.
- 2 Instead of  $\mathcal{V}$  sending random values, prover should be able to generate it himself, but he should not know the randomness in advance.
- 3 Thus, we can replace the verifier's messages with the hash (random oracle) of all the previous conversation.

# Fiat-Shamir Heuristic Illustration



**Thank you for your attention!**