CS 594 – Advanced Crypto (Spring 2026)

Alex Block

Lecture 11

February 23, 2026

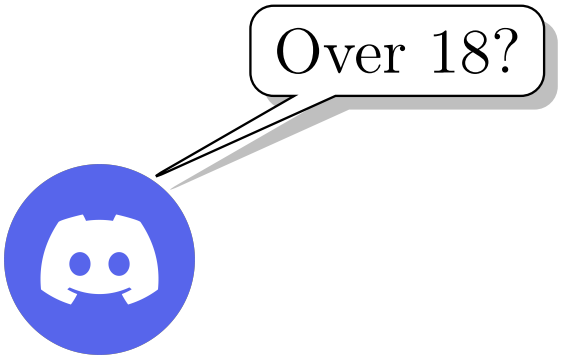
Zero-knowledge, Intuitively

What is "Zero-knowledge"?

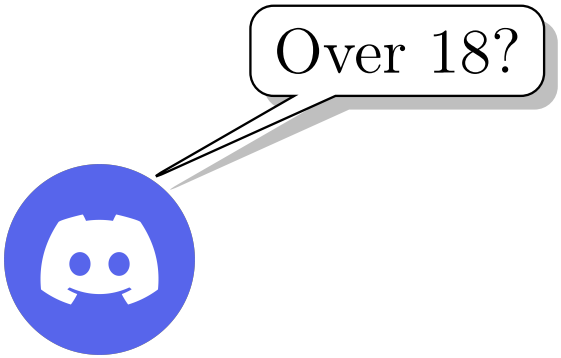
What is "Zero-knowledge"?

What is "Zero-knowledge"?

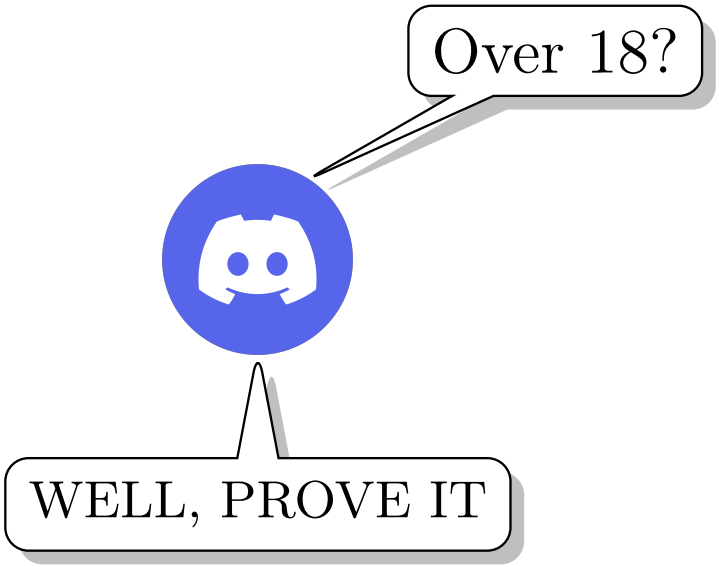
What is "Zero-knowledge"?



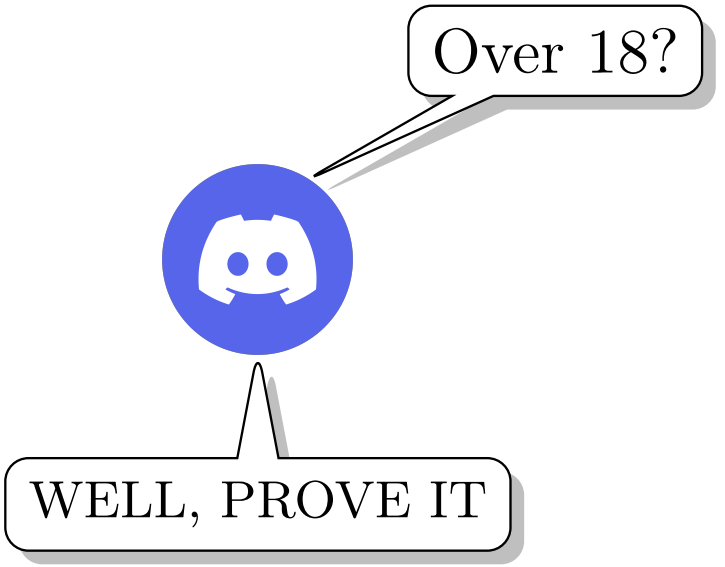
What is "Zero-knowledge"?

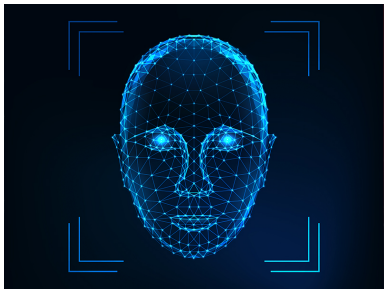


What is "Zero-knowledge"?

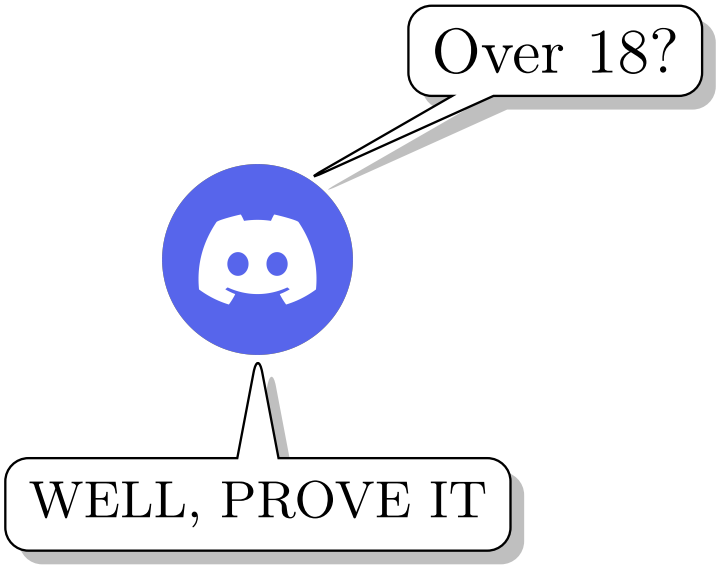


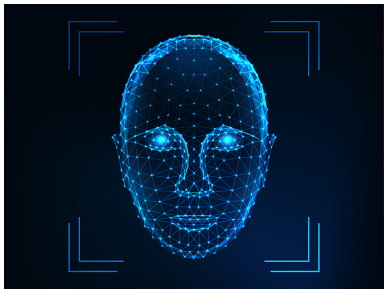
What is "Zero-knowledge"?

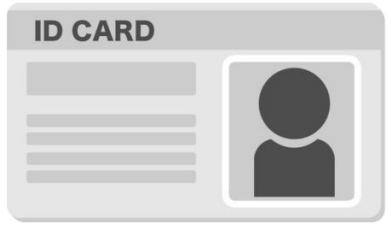




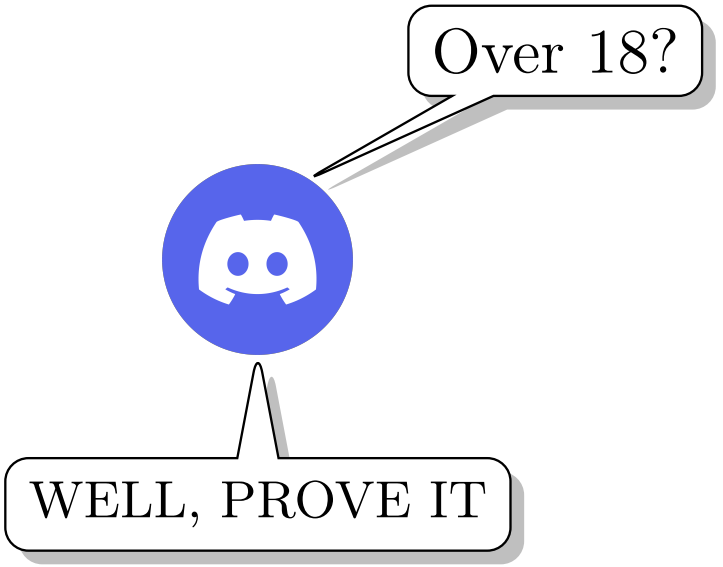
What is "Zero-knowledge"?

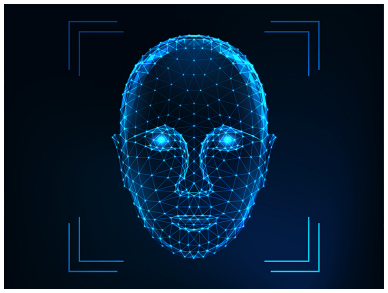


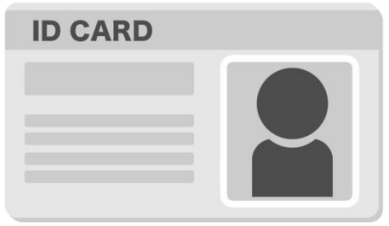




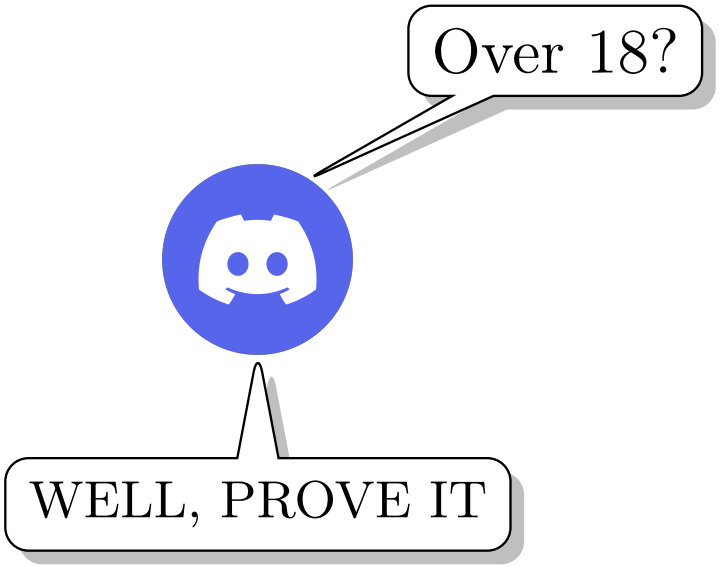
What is "Zero-knowledge"?

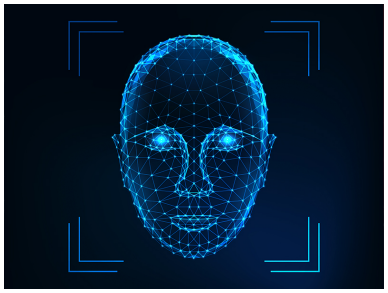


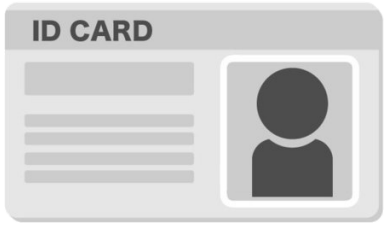




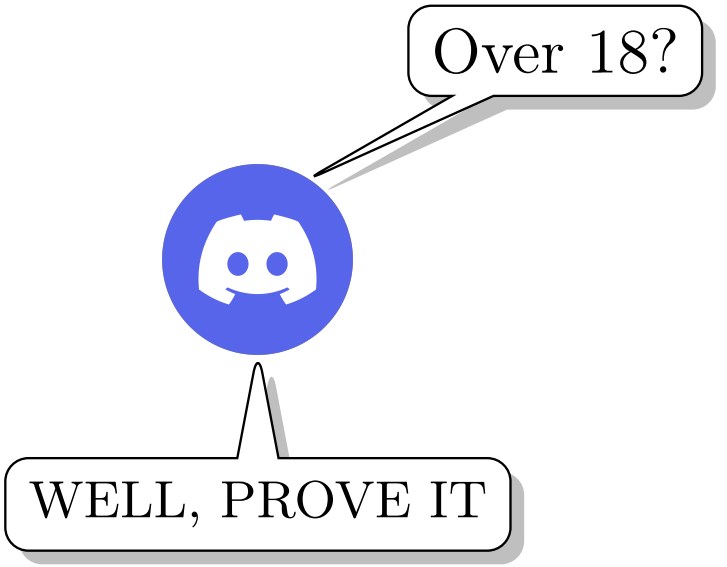
What is "Zero-knowledge"?

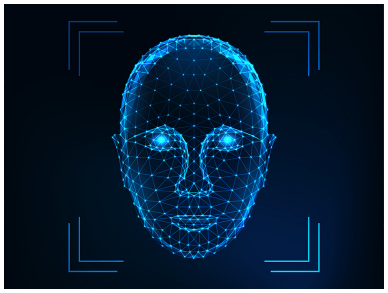


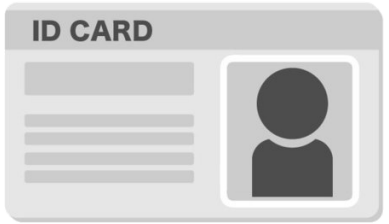




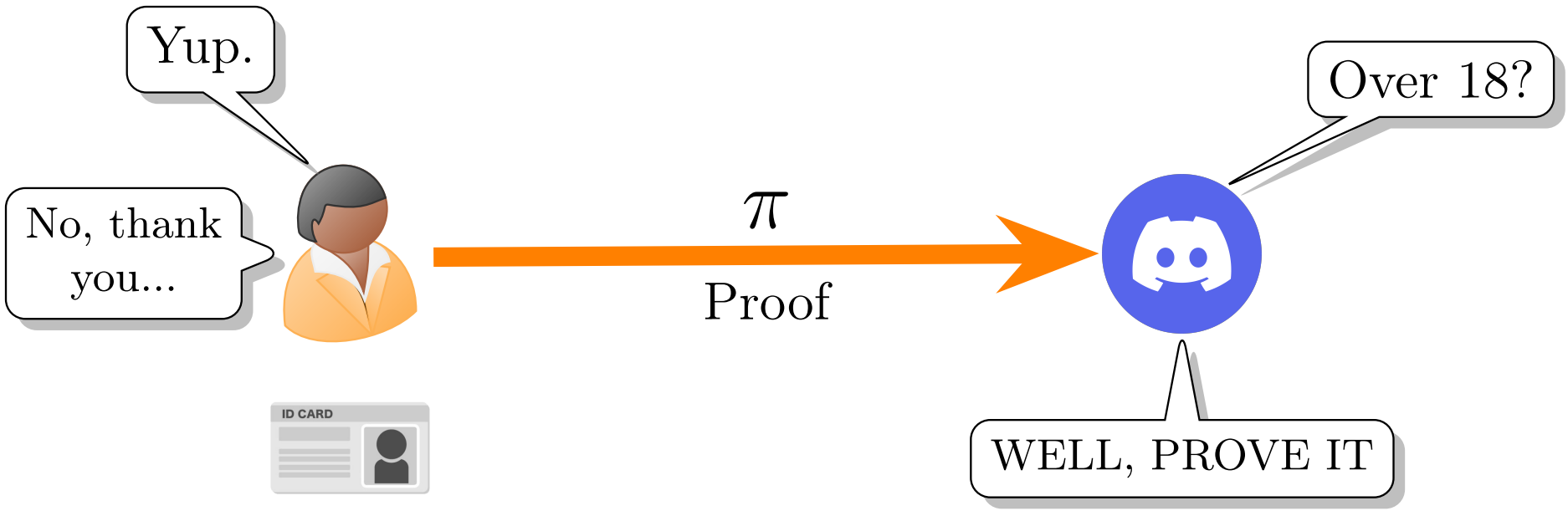
What is "Zero-knowledge"?

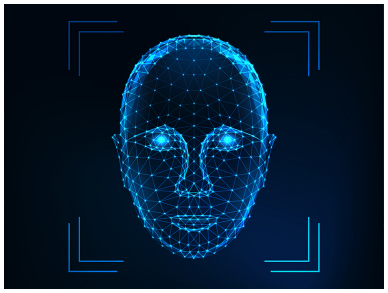


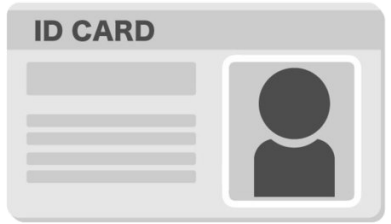




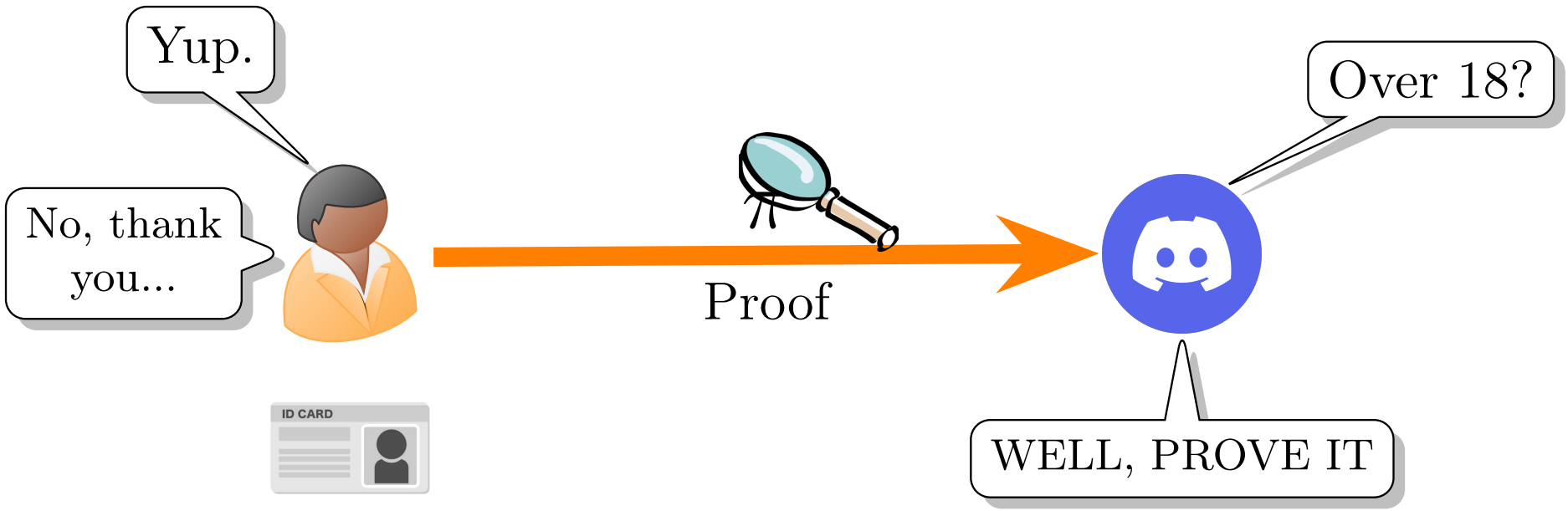
What is "Zero-knowledge"?

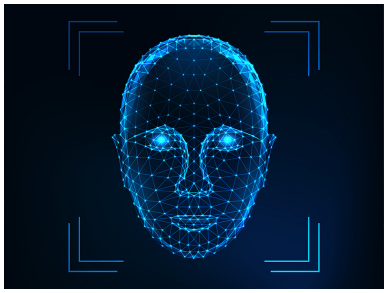


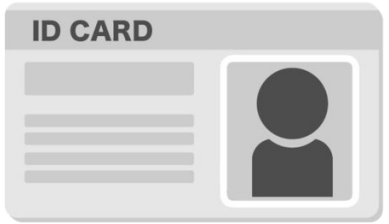




What is "Zero-knowledge"?

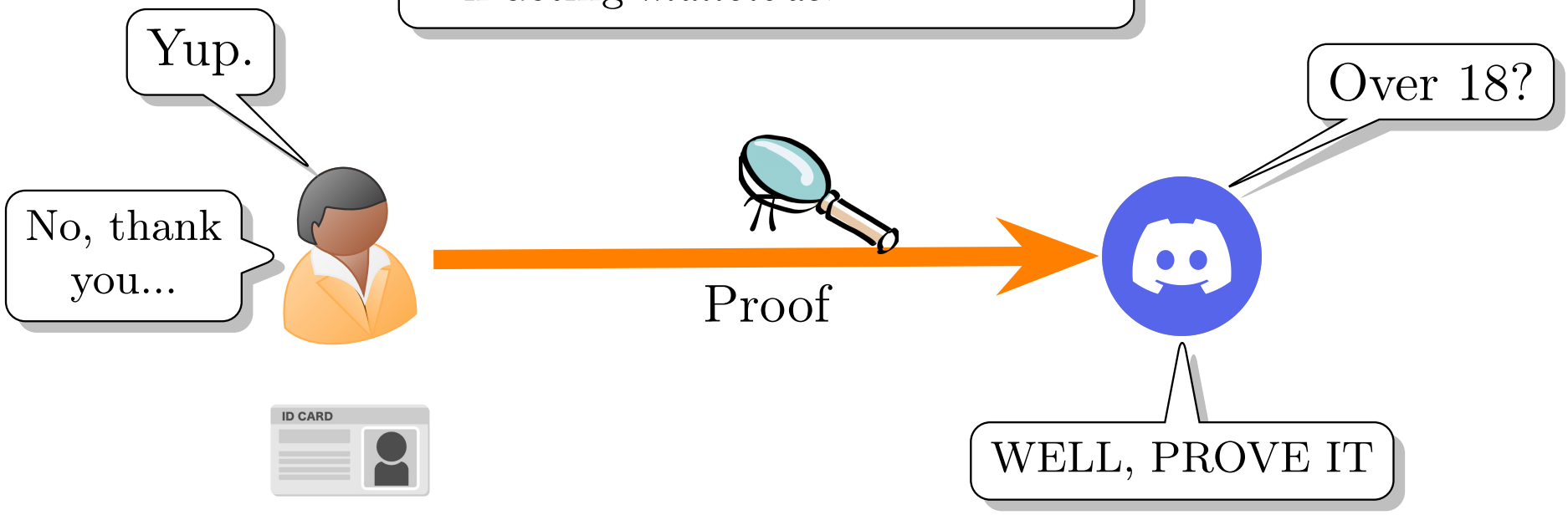


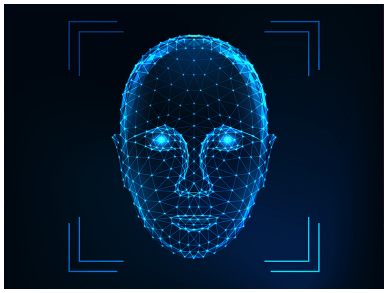


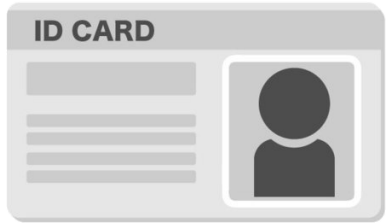


What is "Zero-knowledge"?

1. Proves you are 18+
2. Discord learns *nothing else*, even if acting *malicious*.

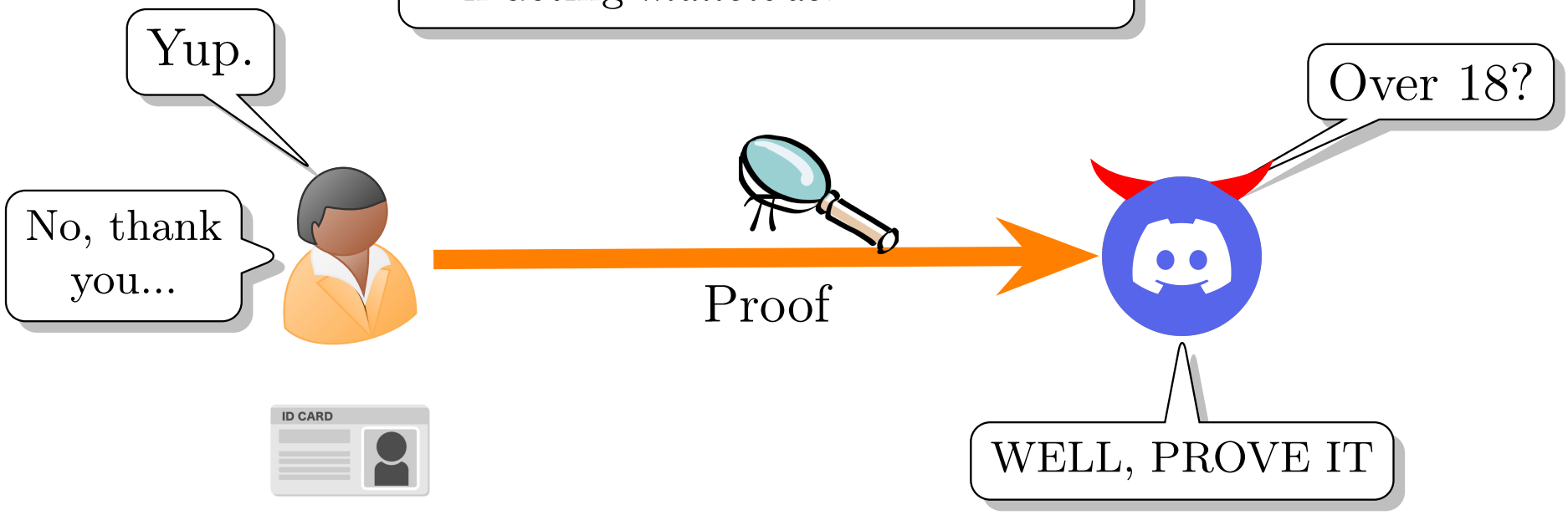


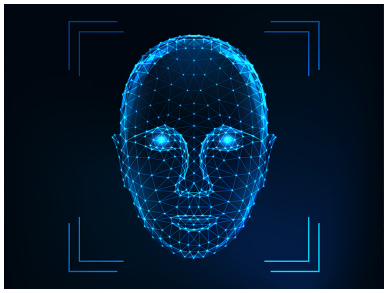


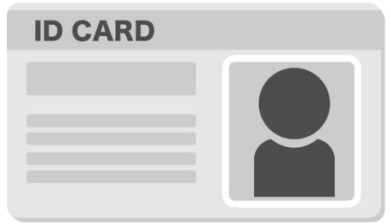


What is "Zero-knowledge"?

1. Proves you are 18+
2. Discord learns *nothing else*, even if acting *malicious*.







What is "Zero-knowledge"?

What is "Zero-knowledge"?

- Intuition: prove a statement is *true*, while revealing *nothing* else.

What is "Zero-knowledge"?

- Intuition: prove a statement is *true*, while revealing *nothing* else.
- Examples:

What is "Zero-knowledge"?

- Intuition: prove a statement is *true*, while revealing *nothing* else.
- Examples:
  - Prove you have the correct security clearance without revealing identity/clearance.

What is "Zero-knowledge"?

- Intuition: prove a statement is *true*, while revealing *nothing* else.
- Examples:
  - Prove you have the correct security clearance without revealing identity/clearance.
  - Prove your bank account has the most money among a group of friends without revealing the amount.

What is "Zero-knowledge"?

- Intuition: prove a statement is *true*, while revealing *nothing* else.
- Examples:
  - Prove you have the correct security clearance without revealing identity/clearance.
  - Prove your bank account has the most money among a group of friends without revealing the amount.
  - Prove your system is in compliance with some laws without revealing user data.

What is "Zero-knowledge"?

- Intuition: prove a statement is *true*, while revealing *nothing* else.
- Examples:
  - Prove you have the correct security clearance without revealing identity/clearance.
  - Prove your bank account has the most money among a group of friends without revealing the amount.
  - Prove your system is in compliance with some laws without revealing user data.

- Natural Questions:

What is "Zero-knowledge"?

- Intuition: prove a statement is *true*, while revealing *nothing* else.
- Examples:
  - Prove you have the correct security clearance without revealing identity/clearance.
  - Prove your bank account has the most money among a group of friends without revealing the amount.
  - Prove your system is in compliance with some laws without revealing user data.

- Natural Questions:
  - Prove?

What is "Zero-knowledge"?

- Intuition: prove a statement is *true*, while revealing *nothing* else.
- Examples:
  - Prove you have the correct security clearance without revealing identity/clearance.
  - Prove your bank account has the most money among a group of friends without revealing the amount.
  - Prove your system is in compliance with some laws without revealing user data.

- Natural Questions:
  - Prove?
  - Statement?

What is "Zero-knowledge"?

- Intuition: prove a statement is *true*, while revealing *nothing* else.
- Examples:
  - Prove you have the correct security clearance without revealing identity/clearance.
  - Prove your bank account has the most money among a group of friends without revealing the amount.
  - Prove your system is in compliance with some laws without revealing user data.

- Natural Questions:
  - Prove?
  - Statement?
  - Reveal nothing else?

Review: Languages and Complexity Classes

Review: Languages and Complexity Classes

Review: Languages and Complexity Classes

- A *language* is a set of strings $L \subseteq \{0,1\}^*$.

Review: Languages and Complexity Classes

- A *language* is a set of strings $L \subseteq \{0,1\}^*$. A *relation* is a tuple of strings $R \subseteq \{0,1\}^* \times \{0,1\}^*$.

Review: Languages and Complexity Classes

- A *language* is a set of strings $L \subseteq \{0,1\}^*$. A *relation* is a tuple of strings $R \subseteq \{0,1\}^* \times \{0,1\}^*$.

- A language L is *Turing decidable* if there exists a Turing machine M_L such that for all $x \in \{0,1\}^*$, $M_L(x) = 1$ if and only if $x \in L$, and $M_L(x) = 0$ otherwise.

Review: Languages and Complexity Classes

- A *language* is a set of strings $L \subseteq \{0,1\}^*$. A *relation* is a tuple of strings $R \subseteq \{0,1\}^* \times \{0,1\}^*$.

- A language L is *Turing decidable* if there exists a Turing machine M_L such that for all $x \in \{0,1\}^*$, $M_L(x) = 1$ if and only if $x \in L$, and $M_L(x) = 0$ otherwise.
  - L is *efficiently decidable* if M_L runs in polynomial time.

Review: Languages and Complexity Classes

- A *language* is a set of strings $L \subseteq \{0,1\}^*$. A *relation* is a tuple of strings $R \subseteq \{0,1\}^* \times \{0,1\}^*$.

- A language L is *Turing decidable* if there exists a Turing machine M_L such that for all $x \in \{0,1\}^*$, $M_L(x) = 1$ if and only if $x \in L$, and $M_L(x) = 0$ otherwise.
  - L is *efficiently decidable* if M_L runs in polynomial time.

- $\mathbf{P} = \{L \colon L \text{ is efficiently decidable}\}$.

Review: Languages and Complexity Classes

- A *language* is a set of strings $L \subseteq \{0,1\}^*$. A *relation* is a tuple of strings $R \subseteq \{0,1\}^* \times \{0,1\}^*$.

- A language L is *Turing decidable* if there exists a Turing machine M_L such that for all $x \in \{0,1\}^*$, $M_L(x) = 1$ if and only if $x \in L$, and $M_L(x) = 0$ otherwise.
  - L is *efficiently decidable* if M_L runs in polynomial time.

- $\mathbf{P} = \{L : L \text{ is efficiently decidable}\}$.

- A language L is *verifiable* if there exists a Turing machine M_L such that for all $x \in \{0,1\}^*$:

Review: Languages and Complexity Classes

- A *language* is a set of strings $L \subseteq \{0,1\}^*$. A *relation* is a tuple of strings $R \subseteq \{0,1\}^* \times \{0,1\}^*$.
- A language L is *Turing decidable* if there exists a Turing machine M_L such that for all $x \in \{0,1\}^*$, $M_L(x) = 1$ if and only if $x \in L$, and $M_L(x) = 0$ otherwise.
  - L is *efficiently decidable* if M_L runs in polynomial time.
- $\mathbf{P} = \{L : L \text{ is efficiently decidable}\}$.
- A language L is *verifiable* if there exists a Turing machine M_L such that for all $x \in \{0,1\}^*$:
  - $x \in L \iff \exists w \in \{0,1\}^* \text{ s.t. } M_L(x,w) = 1$.

Review: Languages and Complexity Classes

- A *language* is a set of strings $L \subseteq \{0,1\}^*$. A *relation* is a tuple of strings $R \subseteq \{0,1\}^* \times \{0,1\}^*$.
- A language L is *Turing decidable* if there exists a Turing machine M_L such that for all $x \in \{0,1\}^*$, $M_L(x) = 1$ if and only if $x \in L$, and $M_L(x) = 0$ otherwise.
  - L is *efficiently decidable* if M_L runs in polynomial time.
- $\mathbf{P} = \{L\colon L \text{ is efficiently decidable}\}$.
- A language L is *verifiable* if there exists a Turing machine M_L such that for all $x \in \{0,1\}^*$:
  - $x \in L \iff \exists w \in \{0,1\}^* \text{ s.t. } M_L(x,w) = 1$.
- A *relation for language* L, denoted as R_L, is the set of all tuples (x,w) such that $M_L(x,w) = 1$.

Review: Languages and Complexity Classes

Review: Languages and Complexity Classes

- A language/relation R_L is *efficiently verifiable* if M_L runs in polynomial time and for all $(x, w) \in R_L$, $|w| = \text{poly}(|x|)$.

Review: Languages and Complexity Classes

- A language/relation R_L is *efficiently verifiable* if M_L runs in polynomial time and for all $(x, w) \in R_L$, $|w| = \text{poly}(|x|)$.

- **NP** $= \{L : L \text{ is efficiently verifiable}\}$. $\mathsf{P} \subseteq \mathsf{NP}$

Review: Languages and Complexity Classes

- A language/relation R_L is *efficiently verifiable* if M_L runs in polynomial time and for all $(x, w) \in R_L$, $|w| = \text{poly}(|x|)$.

- $\mathbf{NP} = \{L : L \text{ is efficiently verifiable}\}$.

- $\mathbf{coNP} = \{L : \overline{L} \in \mathbf{NP}\}$. $\quad coNP \neq \overline{NP}$

Review: Languages and Complexity Classes

- A language/relation R_L is *efficiently verifiable* if M_L runs in polynomial time and for all $(x, w) \in R_L$, $|w| = \text{poly}(|x|)$.
- $\mathbf{NP} = \{L : L \text{ is efficiently verifiable}\}$.
- $\mathbf{coNP} = \{L : \overline{L} \in \mathbf{NP}\}$.

Theorem 1

$$\mathbf{P} \subset \mathbf{NP} \cap \mathbf{coNP}.$$

Review: Languages and Complexity Classes

- A language/relation R_L is *efficiently verifiable* if M_L runs in polynomial time and for all $(x, w) \in R_L$, $|w| = \text{poly}(|x|)$.
- $\mathbf{NP} = \{L : L \text{ is efficiently verifiable}\}$.
- $\mathbf{coNP} = \{L : \overline{L} \in \mathbf{NP}\}$.

Theorem 1

$$\mathbf{P} \subset \mathbf{NP} \cap \mathbf{coNP}.$$

- For complexity class $\mathbf{C}$, a language $L \in \mathbf{C}$ is $\mathbf{C}$-*complete* if for all $L' \in \mathbf{C}$, $L' \leqslant_p L$ (L' is polynomial-time reducible to L).

Review: Languages and Complexity Classes

- A language/relation R_L is *efficiently verifiable* if M_L runs in polynomial time and for all $(x, w) \in R_L$, $|w| = \text{poly}(|x|)$.
- $\mathbf{NP} = \{L : L \text{ is efficiently verifiable}\}$.
- $\mathbf{coNP} = \{L : \overline{L} \in \mathbf{NP}\}$.

Theorem 1

$$\mathbf{P} \subset \mathbf{NP} \cap \mathbf{coNP}.$$

- For complexity class $\mathbf{C}$, a language $L \in \mathbf{C}$ is $\mathbf{C}$-*complete* if for all $L' \in \mathbf{C}$, $L' \leqslant_p L$ (L' is polynomial-time reducible to L).
  - $\exists$ poly-time computable function f such that $x \in L' \iff f(x) \in L$.

Review: Languages and Complexity Classes

Review: Languages and Complexity Classes

Theorem 2 (Cook-Levin)

$\mathsf{3SAT} = \{\phi : \phi \textit{ is a satisfiable 3CNF formula}\}$ *is* **NP***-complete.*

Review: Languages and Complexity Classes

Theorem 2 (Cook-Levin)

$\mathsf{3SAT} = \{\phi : \phi \textit{ is a satisfiable 3CNF formula}\}$ *is* **NP***-complete.*

- $\mathbf{PSPACE} = \{L : L \text{ is Turing decidable in polynomial } \textit{space}\}$.

Review: Languages and Complexity Classes

Theorem 2 (Cook-Levin)

$\mathsf{3SAT} = \{\phi\colon \phi \textit{ is a satisfiable 3CNF formula}\}$ *is* **NP***-complete.*

- $\mathbf{PSPACE} = \{L\colon L \text{ is Turing decidable in polynomial } \textit{space}\}$.
- Note: $\mathbf{NP} \subseteq \mathbf{PSPACE}$.

Review: Languages and Complexity Classes

Theorem 2 (Cook-Levin)

$\mathsf{3SAT} = \{\phi : \phi \textit{ is a satisfiable 3CNF formula}\}$ *is* **NP***-complete.*

- $\mathbf{PSPACE} = \{L : L \text{ is Turing decidable in polynomial } \textit{space}\}$.
- Note: $\mathbf{NP} \subseteq \mathbf{PSPACE}$.

What's the point here?

Review: Languages and Complexity Classes

Theorem 2 (Cook-Levin)

$\mathsf{3SAT} = \{\phi : \phi \textit{ is a satisfiable 3CNF formula}\}$ *is* **NP***-complete.*

- $\mathbf{PSPACE} = \{L : L \text{ is Turing decidable in polynomial } \textit{space}\}$.
- Note: $\mathbf{NP} \subseteq \mathbf{PSPACE}$.

What's the point here?

The "statements" we prove will be with respect to some fixed language.

Interactive Proofs and Arguments

Interactive Proofs

Interactive Proofs

- Let $L \in \mathbf{NP}$ with relation R_L.

Interactive Proofs

- Let $L \in \mathbf{NP}$ with relation R_L.
- For $(x, w) \in R_L$, in some sense, w is a "proof" that $x \in L$.

$(x, w) \Rightarrow M_L(x, w)$

Interactive Proofs

- Let $L \in \mathbf{NP}$ with relation R_L.

- For $(x, w) \in R_L$, in some sense, w is a "proof" that $x \in L$.
  - Given (x, w) and some description of L, computing $M_L(x, w)$ convinces you if $x \in L$ or $x \notin L$.

Interactive Proofs

- Let $L \in \mathbf{NP}$ with relation R_L.
- For $(x, w) \in R_L$, in some sense, w is a "proof" that $x \in L$.
  - Given (x, w) and some description of L, computing $M_L(x, w)$ convinces you if $x \in L$ or $x \notin L$.
- *Interactive Proofs*: attempt to re-define **NP** using *interaction*.

Interactive Proofs

- Let $L \in \mathbf{NP}$ with relation R_L.
- For $(x, w) \in R_L$, in some sense, w is a "proof" that $x \in L$.
  - Given (x, w) and some description of L, computing $M_L(x, w)$ convinces you if $x \in L$ or $x \notin L$.
- *Interactive Proofs*: attempt to re-define **NP** using *interaction*.
- Goal: using interaction, for some fixed $x \in \{0,1\}^*$, can a (all-powerful) prover P with $w \in \{0,1\}^*$ such that $(x, w) \in R_L$ convince a (computationally-weak) verifier V that $x \in L$ while only exchanging $o(|w|)$ bits?

Interactive Proofs

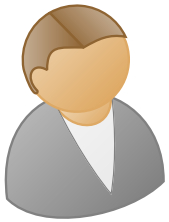
Interactive Proofs

$\chi = \phi(x_1, \ldots, x_n)$ $w = (a_1, \ldots, a_n)$ s.t. $\phi(a_1, \ldots, a_n) = 1$

$(x, w) \in R_L$

Interactive Proofs

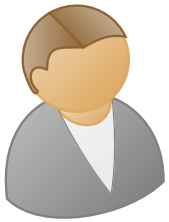
$$(x, w) \in R_L$$



Prover P

Interactive Proofs

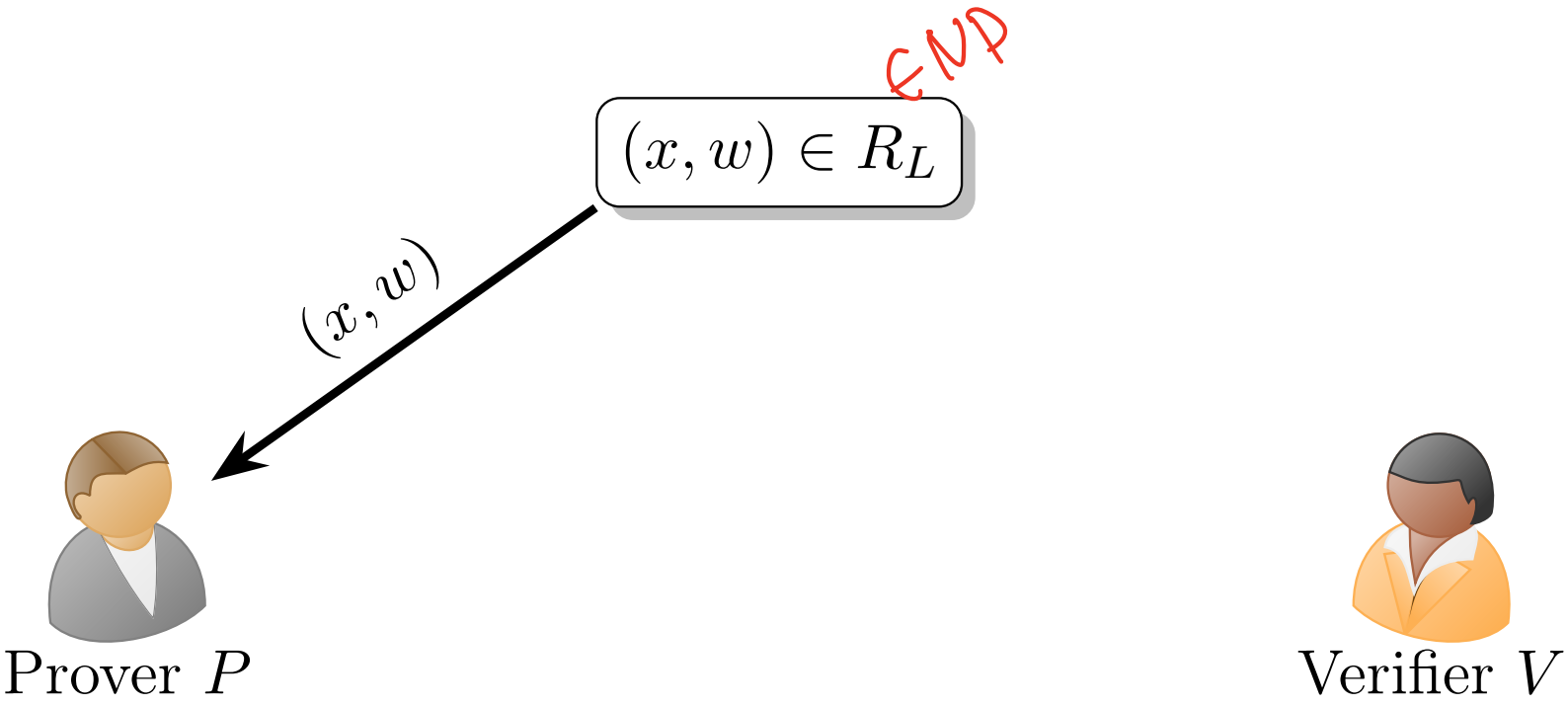
$(x, w) \in R_L$



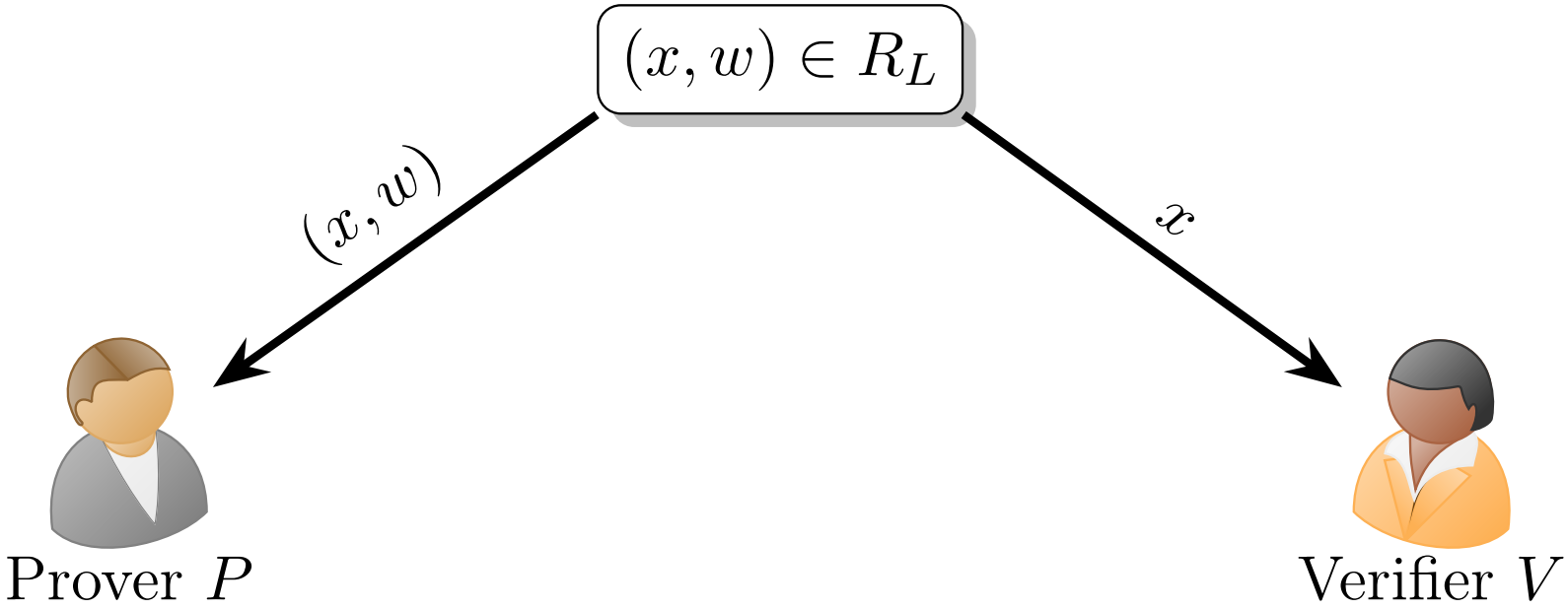
Prover P

Verifier V

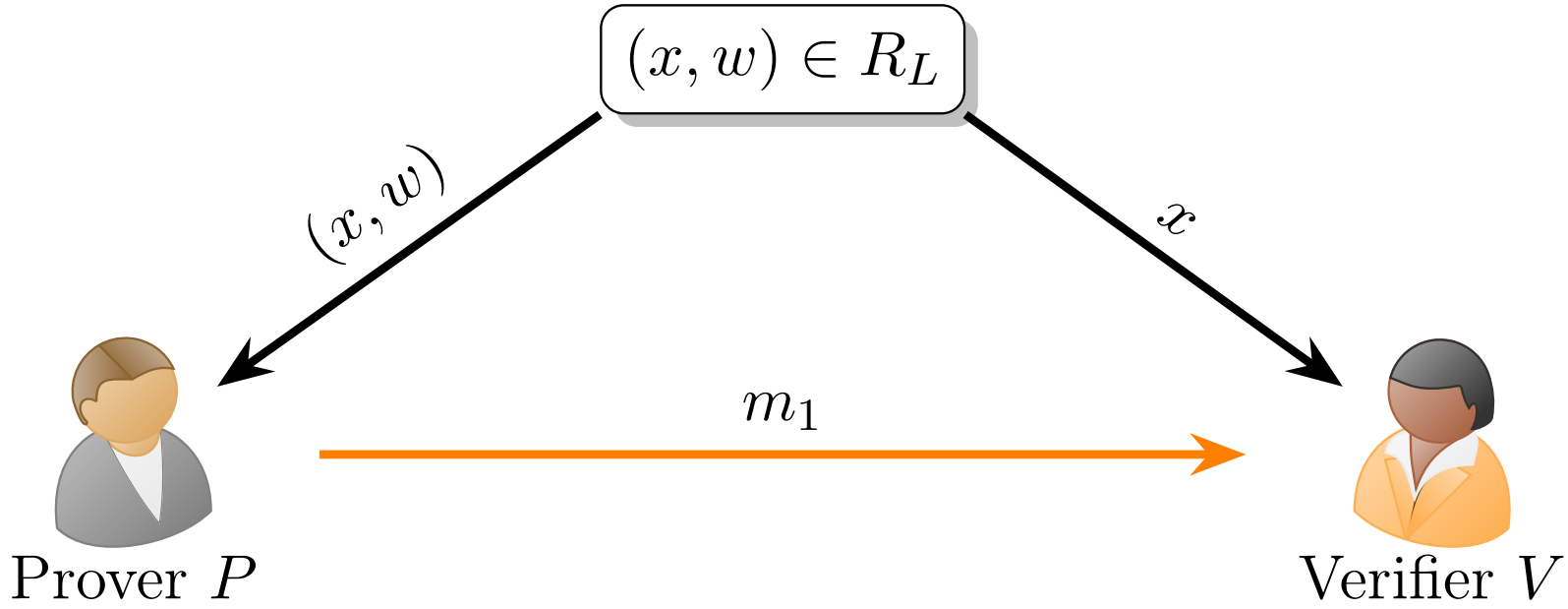
Interactive Proofs



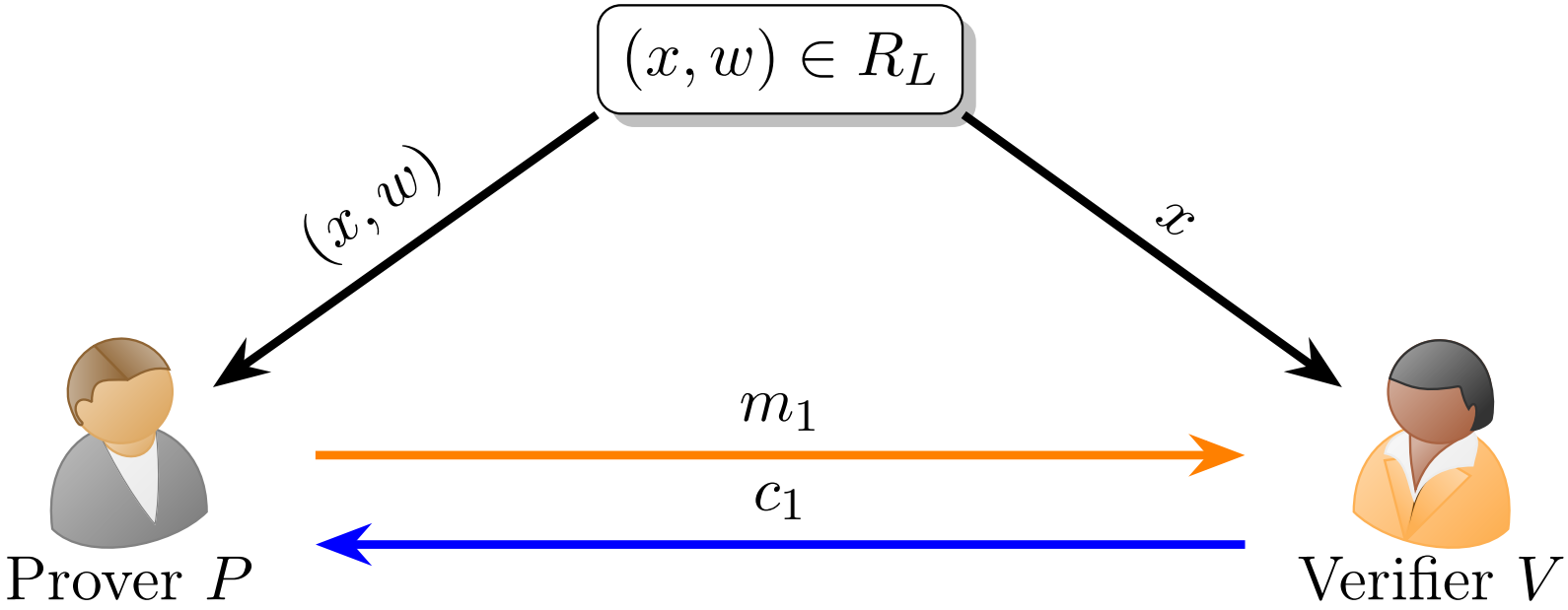
Interactive Proofs



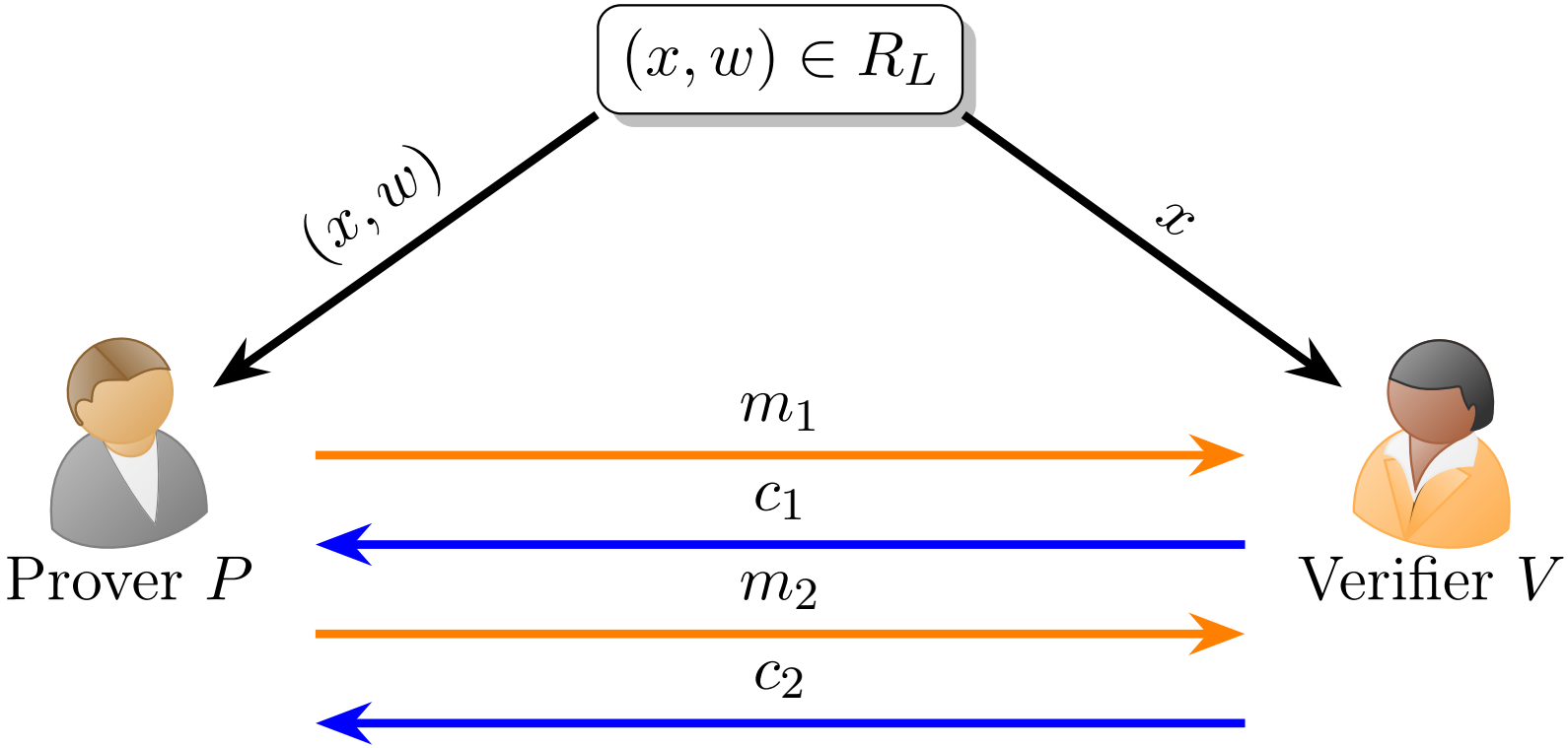
Interactive Proofs



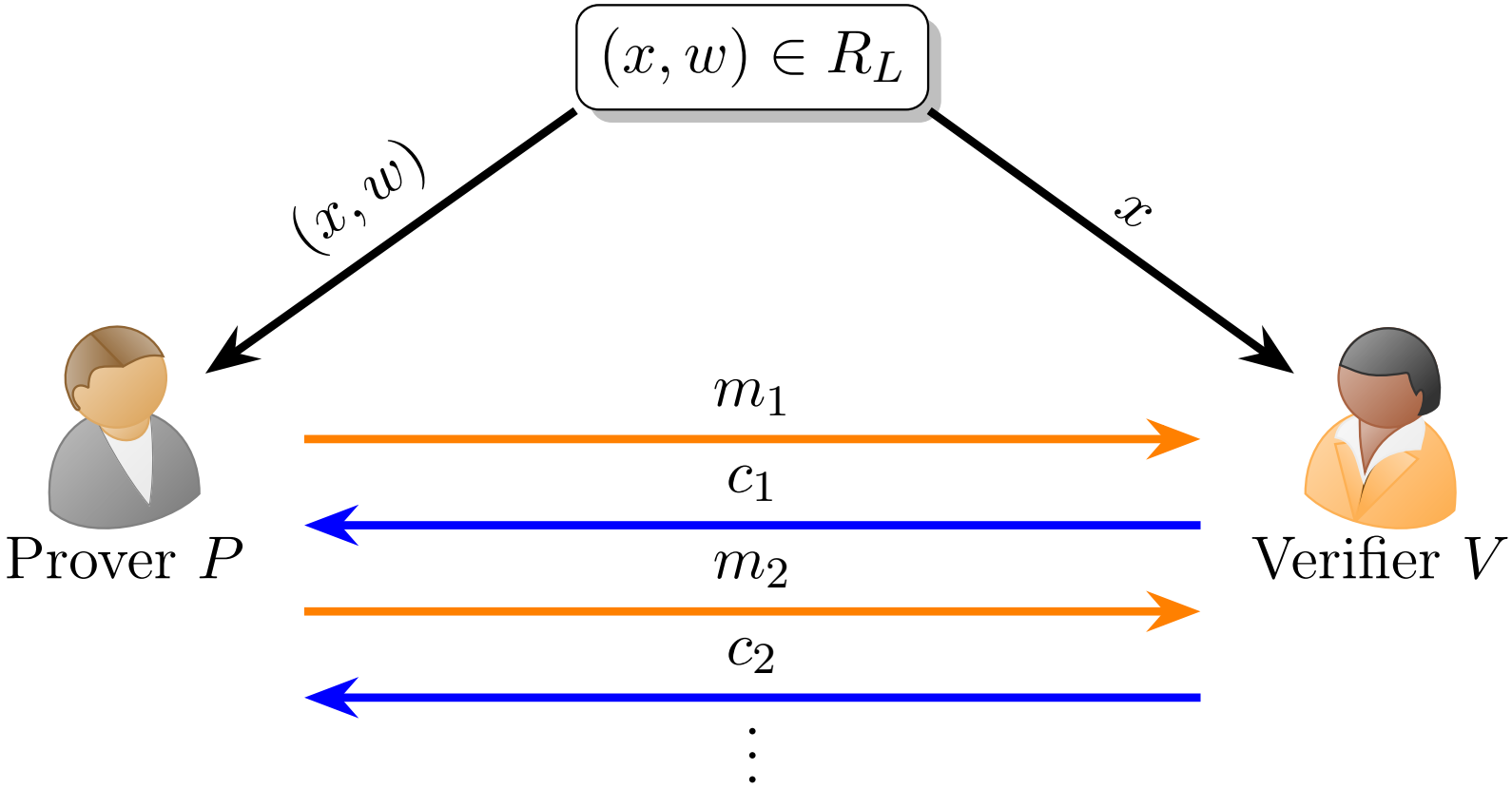
Interactive Proofs



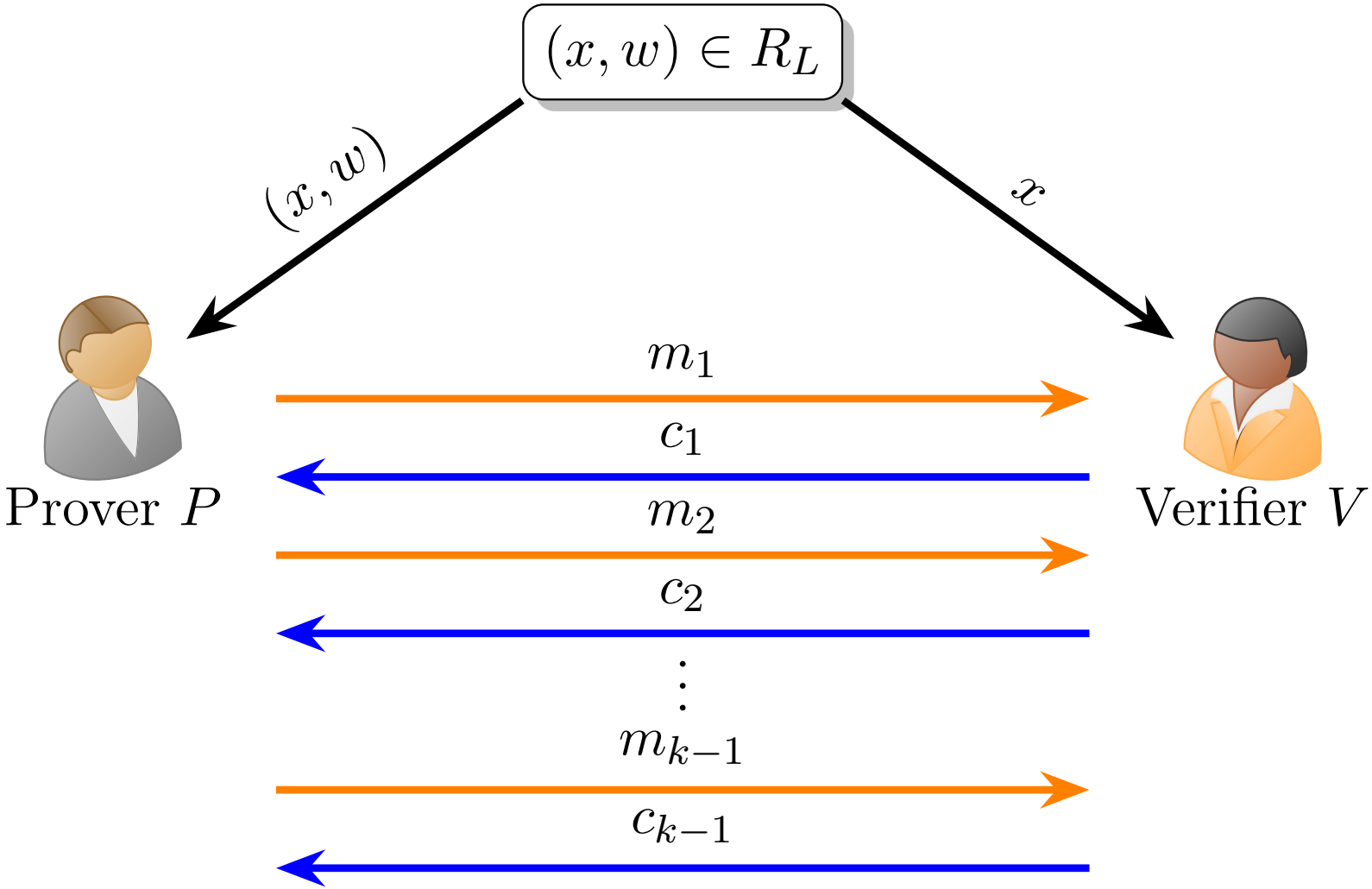
Interactive Proofs



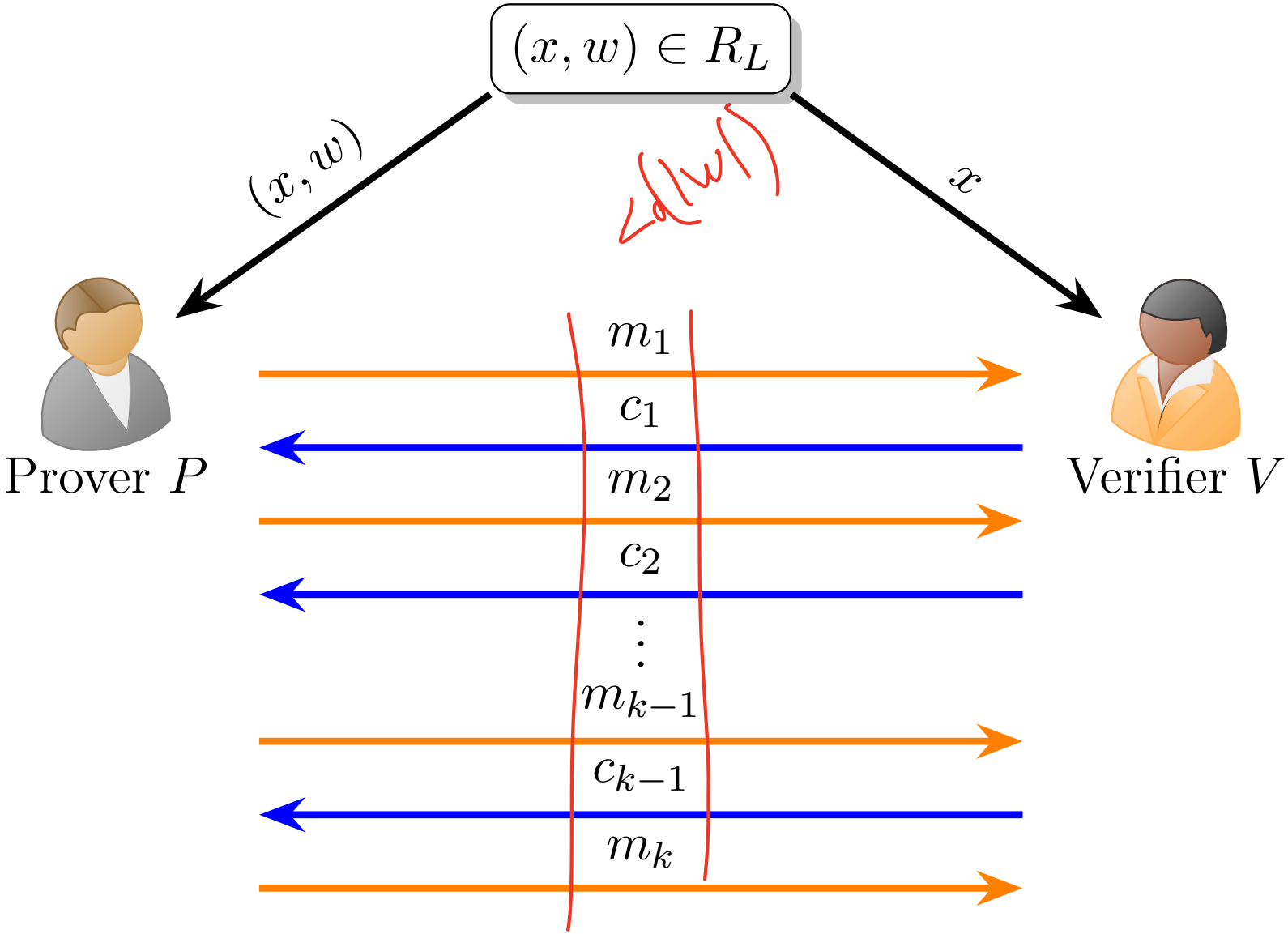
Interactive Proofs



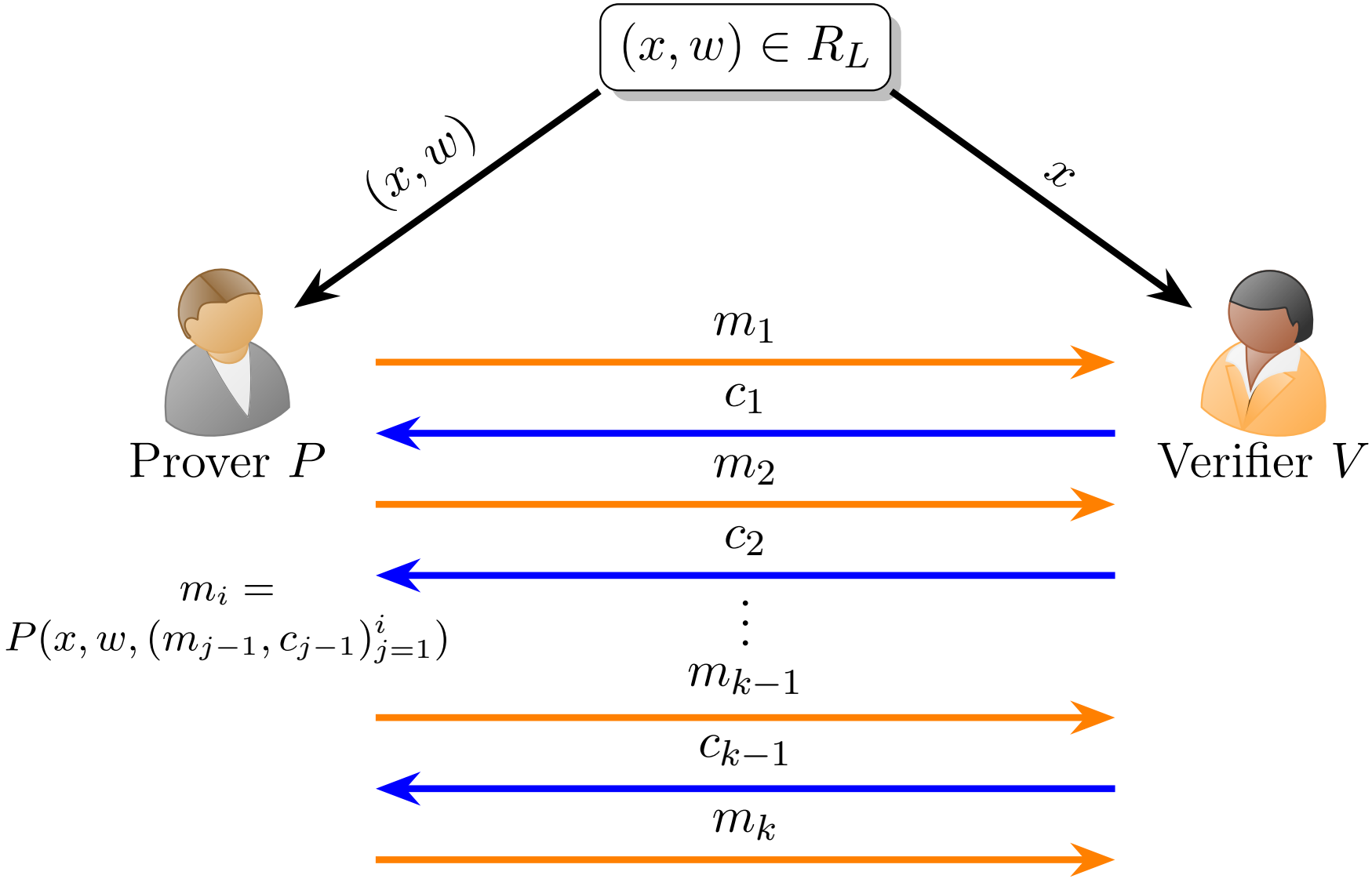
Interactive Proofs



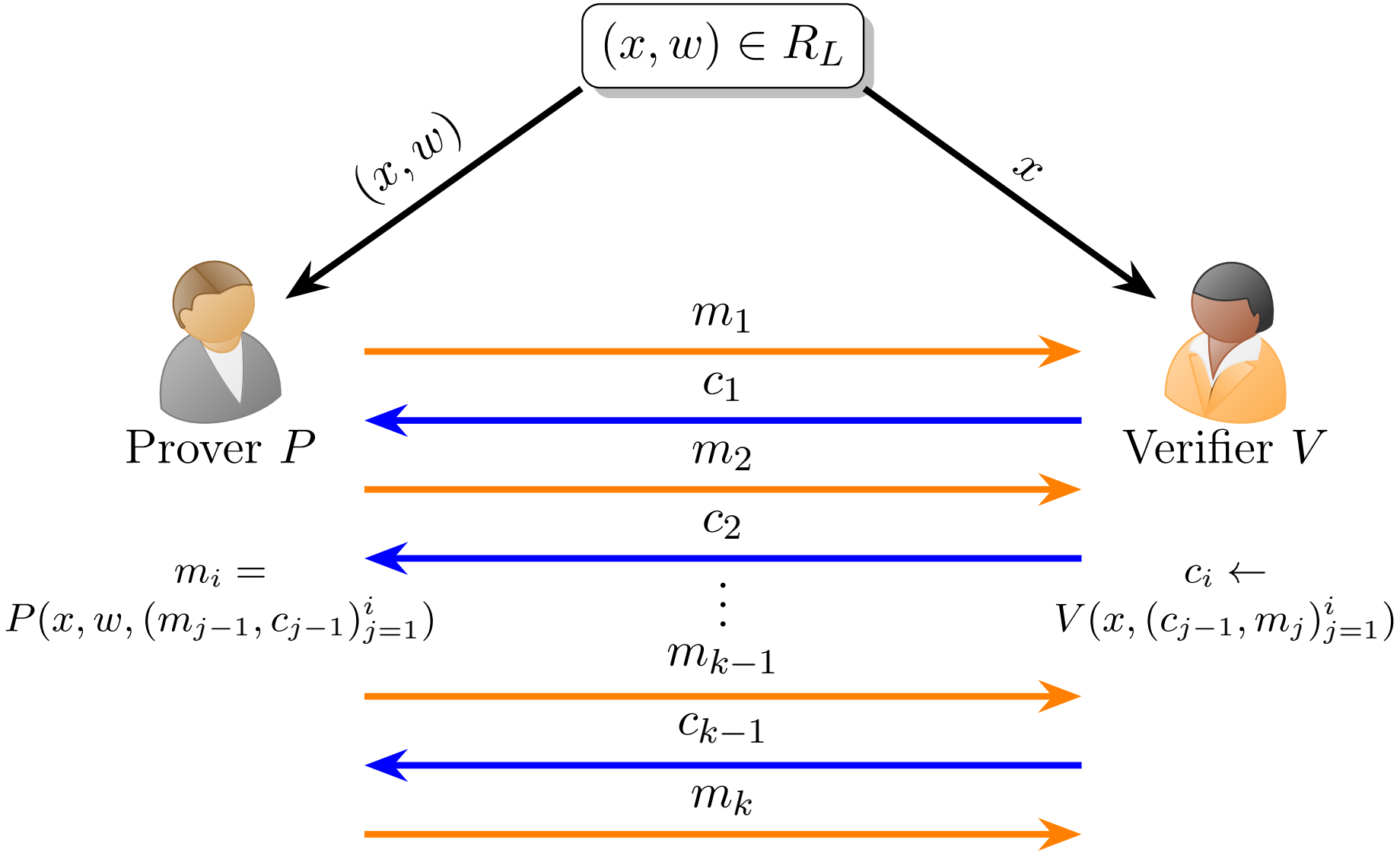
Interactive Proofs



Interactive Proofs



Interactive Proofs



Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$;

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$;
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$;
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- Notation $\langle P(w), V(z)\rangle(x) = b$

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$;
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- Notation $\langle P(w), V(z)\rangle(x) = b$
  - b is the result of the interaction; $b = 1$ means the *verifier accepts*; otherwise it *rejects*.

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$;
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- Notation $\langle P(w), V(z)\rangle(x) = b$ (protocol; common input)
  - b is the result of the interaction; $b = 1$ means the *verifier accepts*; otherwise it *rejects*.
  - x is the common input to both parties; w is private to P and z is private to V.

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$;
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- We usually require *perfect completeness*: the probability $= 1$ above.

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant$ ~~$1/3$~~; $1/(\mathrm{poly}(|x|)$ or $\mathrm{negl}(|x|)$
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- We usually require *perfect completeness*: the probability $= 1$ above.
- Ideally, we want soundness error ε, where $\varepsilon = 1/\mathrm{poly}$ or negl.

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$;
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- We usually require *perfect completeness*: the probability $= 1$ above.
- Ideally, we want soundness error ε, where $\varepsilon = 1/\mathrm{poly}$ or negl.
- Since V is poly-time (in $|x|$), the number of bits exchanged must be $\mathrm{poly}(|x|)$.

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$;
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- $\pi = (m_1, c_1, \ldots, c_{k-1}, m_k)$ is the *transcript* of the interaction, and is also called the *proof*; m_i is P's message, and c_i is V's message.

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$;
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- $\pi = (m_1, c_1, \ldots, c_{k-1}, m_k)$ is the *transcript* of the interaction, and is also called the *proof*; m_i is P's message, and c_i is V's message.
- V may toss *private/hidden randomness*, but this is not necessary.

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$;
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$;
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- $\pi = (m_1, c_1, \ldots, c_{k-1}, m_k)$ is the *transcript* of the interaction, and is also called the *proof*; m_i is P's message, and c_i is V's message.
- V may toss *private/hidden randomness*, but this is not necessary.
- π is *publicly verifiable* if any other verifier V' can examine π and output accept/reject with the same guarantees.

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$; and
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$.
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- We will assume that all our protocols are *public-coin*: V's messages are all uniformly random (with respect to some set).

Interactive Proofs: Definition and Notes

Definition 1 (Interactive Proof)

A pair of interactive algorithms (P, V) is a *k-round interactive proof* for a language L if the following hold.

- (Efficiency) P is a deterministic and computationally unbounded algorithm, while V is a probabilistic polynomial-time algorithm;
- (Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] \geqslant 2/3$; and
- (Soundness) If $x \notin L$, then for all algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant 1/3$.
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

- We will assume that all our protocols are *public-coin*: V's messages are all uniformly random (with respect to some set).
- $|\pi|$ is the *proof size*, and we want $|\pi| = o(|w|)$, where $(x, w) \in R_L$.

Interactive Proofs: Results

Interactive Proofs: Results

- Let $\mathbf{IP}[k]$ be the set of all languages with a k-round interactive proof.

Interactive Proofs: Results

- Let $\mathbf{IP}[k]$ be the set of all languages with a k-round interactive proof.
- $\mathbf{IP} = \cup_{c \in \mathbb{N}} \mathbf{IP}[n^c]$.

poly # rounds in length of $|x|$

Interactive Proofs: Results

- Let $\mathbf{IP}[k]$ be the set of all languages with a k-round interactive proof.

- $\mathbf{IP} = \cup_{c \in \mathbb{N}} \mathbf{IP}[n^c]$.

Theorem 3 (Shamir; Lund, Fortnow, Karloff, and Nisan)

$$\mathbf{IP} = \mathbf{PSPACE}.$$

Interactive Proofs: Results

- Let $\mathbf{IP}[k]$ be the set of all languages with a k-round interactive proof.

- $\mathbf{IP} = \cup_{c \in \mathbb{N}} \mathbf{IP}[n^c]$.

Theorem 3 (Shamir; Lund, Fortnow, Karloff, and Nisan)

$$\mathbf{IP} = \mathbf{PSPACE}.$$

- Let **dIP** be **IP** but with *deterministic verifiers*.

Interactive Proofs: Results

- Let $\mathbf{IP}[k]$ be the set of all languages with a k-round interactive proof.

- $\mathbf{IP} = \cup_{c \in \mathbb{N}} \mathbf{IP}[n^c]$.

Theorem 3 (Shamir; Lund, Fortnow, Karloff, and Nisan)

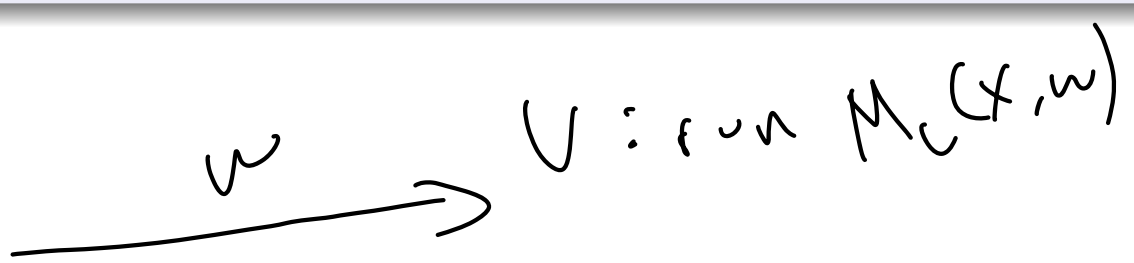
$$\mathbf{IP} = \mathbf{PSPACE}.$$

- Let **dIP** be **IP** but with *deterministic verifiers*.

Theorem 4

$$\mathbf{dIP} = \mathbf{NP}.$$

NP ⊆ dIP



Interactive Proofs: Results

- Let $\mathbf{IP}[k]$ be the set of all languages with a k-round interactive proof.

- $\mathbf{IP} = \cup_{c \in \mathbb{N}} \mathbf{IP}[n^c]$.

Theorem 3 (Shamir; Lund, Fortnow, Karloff, and Nisan)

$$\mathbf{IP} = \mathbf{PSPACE}.$$

- Let $\mathbf{dIP}$ be $\mathbf{IP}$ but with *deterministic verifiers*.

Theorem 4

$$\mathbf{dIP} = \mathbf{NP}.$$

- Notably: $\mathbf{NP} \subseteq \mathbf{IP}$.

Interactive Arguments

Interactive Arguments

- Natural restriction: *efficient provers*.

Interactive Arguments

- Natural restriction: *efficient provers*.

Definition 2 (Interactive Argument)

Interactive Arguments

- Natural restriction: *efficient provers*.

Definition 2 (Interactive Argument)

A pair of interactive algorithms (P, V) is a *k-round interactive argument* for a language L if the following hold.

Interactive Arguments

- Natural restriction: *efficient provers*.

Definition 2 (Interactive Argument)

A pair of interactive algorithms (P, V) is a *k-round interactive argument* for a language L if the following hold.

- (Efficiency) P and V are probabilistic polynomial-time algorithms;

Interactive Arguments

- Natural restriction: *efficient provers*.

Definition 2 (Interactive Argument)

A pair of interactive algorithms (P, V) is a *k-round interactive argument* for a language L if the following hold.

- (Efficiency) P and V are probabilistic polynomial-time algorithms;
- (Perfect Completeness) If $x \in L$, then $\Pr_V[\langle P, V \rangle(x) = 1] = 1$;

Interactive Arguments

- Natural restriction: *efficient provers*.

Definition 2 (Interactive Argument)

A pair of interactive algorithms (P, V) is a *k-round interactive argument* for a language L if the following hold.

- (Efficiency) P and V are probabilistic polynomial-time algorithms;
- (Perfect Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] = 1$;
- (Soundness) If $x \notin L$, then for all PPT algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant \mathsf{negl}(|x|)$;

Interactive Arguments

- Natural restriction: *efficient provers*.

Definition 2 (Interactive Argument)

A pair of interactive algorithms (P, V) is a *k-round interactive argument* for a language L if the following hold.

- (Efficiency) P and V are probabilistic polynomial-time algorithms;
- (Perfect Completeness) If $x \in L$, then $\Pr_V[\langle P, V\rangle(x) = 1] = 1$;
- (Soundness) If $x \notin L$, then for all PPT algorithms P^*, $\Pr_V[\langle P^*, V\rangle(x) = 1] \leqslant \mathsf{negl}(|x|)$;
- (Rounds) P and V exchange $2k + 1$ messages during the interaction.

Computational!

Zero-knowledge Proofs

IPs So Far

IPs So Far

- In the context of IP's, we have only been concerned with malicious *provers*.

IPs So Far

- In the context of IP's, we have only been concerned with malicious *provers*.

- This makes sense for several reasons.

IPs So Far

- In the context of IP's, we have only been concerned with malicious *provers*.

- This makes sense for several reasons.
  - IPs are used in the context of *verifiable computation/delegation*.

IPs So Far

- In the context of IP's, we have only been concerned with malicious *provers*.

- This makes sense for several reasons.
  - IPs are used in the context of *verifiable computation/delegation*. The verifier is asking the prover to compute some function of her data, so why would she be malicious?

IPs So Far

- In the context of IP's, we have only been concerned with malicious *provers*.

- This makes sense for several reasons.
  - IPs are used in the context of *verifiable computation/delegation*. The verifier is asking the prover to compute some function of her data, so why would she be malicious?
  - Prover is assumed to be much more computationally capable, even for interactive arguments.

IPs So Far

- In the context of IP's, we have only been concerned with malicious *provers*.

- This makes sense for several reasons.
  - IPs are used in the context of *verifiable computation/delegation*. The verifier is asking the prover to compute some function of her data, so why would she be malicious?
  - Prover is assumed to be much more computationally capable, even for interactive arguments. Does it make sense for a computationally weak verifier to try and "cheat" this powerful prover?

IPs So Far

- In the context of IP's, we have only been concerned with malicious *provers*.

- This makes sense for several reasons.
  - IPs are used in the context of *verifiable computation/delegation*. The verifier is asking the prover to compute some function of her data, so why would she be malicious?
  - Prover is assumed to be much more computationally capable, even for interactive arguments. Does it make sense for a computationally weak verifier to try and "cheat" this powerful prover?

- *Zero-knowledge* attempts to handle the following threat scenario:

IPs So Far

- In the context of IP's, we have only been concerned with malicious *provers*.

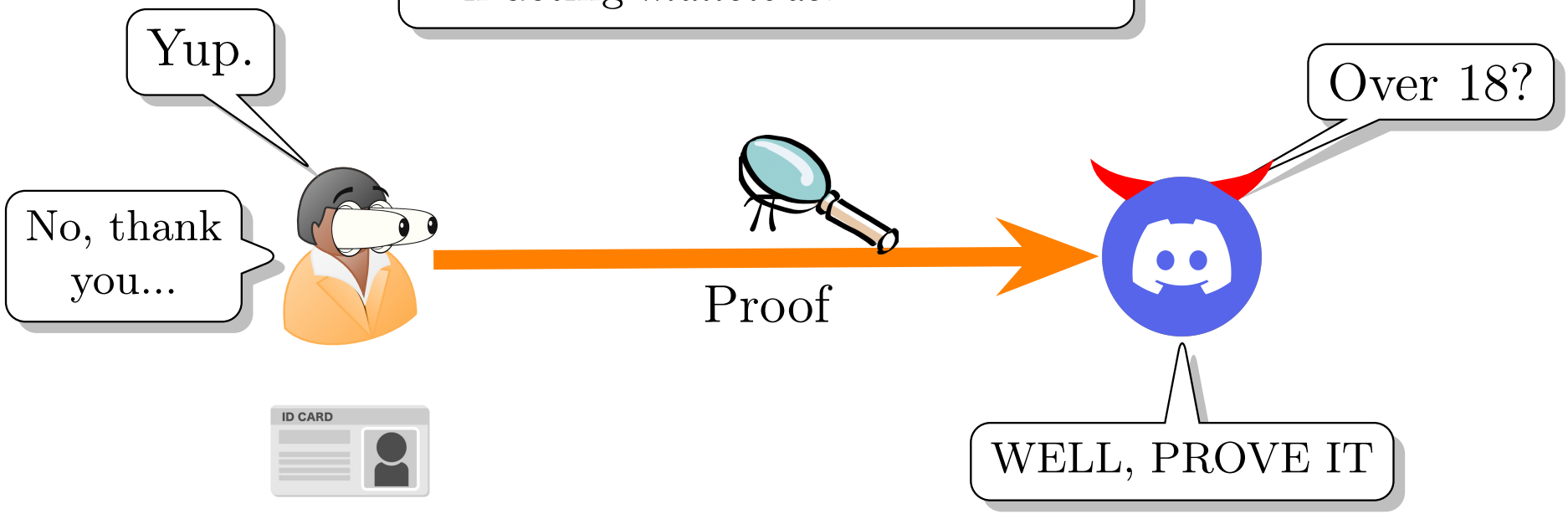
- This makes sense for several reasons.
  - IPs are used in the context of *verifiable computation/delegation*. The verifier is asking the prover to compute some function of her data, so why would she be malicious?
  - Prover is assumed to be much more computationally capable, even for interactive arguments. Does it make sense for a computationally weak verifier to try and "cheat" this powerful prover?

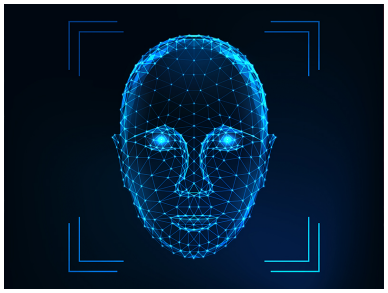
- *Zero-knowledge* attempts to handle the following threat scenario:

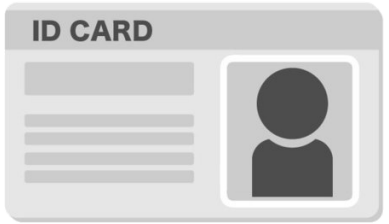
> What if (honest) P's witness contains *sensitive* information?

Example from Before

1. Proves you are 18+
2. Discord learns *nothing else*, even if acting *malicious*.







Zero-knowledge Proofs

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Let (P, V) be a k-round interactive proof for a language L.

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Let (P, V) be a k-round interactive proof for a language L. Then, we say that the proof system has *zero-knowledge* if

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Let (P, V) be a k-round interactive proof for a language L. Then, we say that the proof system has *zero-knowledge* if for every PPT verifier algorithm V^*,

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Let (P, V) be a k-round interactive proof for a language L. Then, we say that the proof system has *zero-knowledge* if for every PPT verifier algorithm V^*, there exists a PPT algorithm S (the *simulator*), which can depend on V^*, such that

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Let (P, V) be a k-round interactive proof for a language L. Then, we say that the proof system has *zero-knowledge* if for every PPT verifier algorithm V^*, there exists a PPT algorithm S (the *simulator*), which can depend on V^*, such that for all $x \in L$, the distribution $S(x)$ is "indistinguishable" from $\text{View}_{V^*}(\langle P, V^* \rangle(x))$.

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Let (P, V) be a k-round interactive proof for a language L. Then, we say that the proof system has *zero-knowledge* if for every PPT verifier algorithm V^*, there exists a PPT algorithm S (the *simulator*), which can depend on V^*, such that for all $x \in L$, the distribution $S(x)$ is "indistinguishable" from $\text{View}_{V^*}(\langle P, V^* \rangle(x))$. Here, $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ denotes the distribution (with respect to V^*) over proofs/transcripts generated by the interaction between P and V^*.

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Let (P, V) be a k-round interactive proof for a language L. Then, we say that the proof system has *zero-knowledge* if for every PPT verifier algorithm V^*, there exists a PPT algorithm S (the *simulator*), which can depend on V^*, such that for all $x \in L$, the distribution $S(x)$ is "indistinguishable" from $\text{View}_{V^*}(\langle P, V^* \rangle(x))$. Here, $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ denotes the distribution (with respect to V^*) over proofs/transcripts generated by the interaction between P and V^*.

- Intuition:

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Let (P, V) be a k-round interactive proof for a language L. Then, we say that the proof system has *zero-knowledge* if for every PPT verifier algorithm V^*, there exists a PPT algorithm S (the *simulator*), which can depend on V^*, such that for all $x \in L$, the distribution $S(x)$ is "indistinguishable" from $\text{View}_{V^*}(\langle P, V^* \rangle(x))$. Here, $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ denotes the distribution (with respect to V^*) over proofs/transcripts generated by the interaction between P and V^*.

- Intuition:
  - If $x \in L$, V *learns nothing from interacting with* P beyond what V could have computed efficiently herself.

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Let (P, V) be a k-round interactive proof for a language L. Then, we say that the proof system has *zero-knowledge* if for every PPT verifier algorithm V^*, there exists a PPT algorithm S (the *simulator*), which can depend on V^*, such that for all $x \in L$, the distribution $S(x)$ is "indistinguishable" from $\text{View}_{V^*}(\langle P, V^* \rangle(x))$. Here, $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ denotes the distribution (with respect to V^*) over proofs/transcripts generated by the interaction between P and V^*.

- Intuition:
  - If $x \in L$, V *learns nothing from interacting with* P beyond what V could have computed efficiently herself.
  - This is because given x, V cannot distinguish between a transcript generated from interacting with P and one generated by $S(x)$.

Zero-knowledge Proofs

Definition 3 (Zero-knowledge Proof, Informal)

Let (P, V) be a k-round interactive proof for a language L. Then, we say that the proof system has *zero-knowledge* if for every PPT verifier algorithm V^*, there exists a PPT algorithm S (the *simulator*), which can depend on V^*, such that for all $x \in L$, the distribution $S(x)$ is "indistinguishable" from $\text{View}_{V^*}(\langle P, V^* \rangle(x))$. Here, $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ denotes the distribution (with respect to V^*) over proofs/transcripts generated by the interaction between P and V^*.

- Intuition:
  - If $x \in L$, V *learns nothing from interacting with* P beyond what V could have computed efficiently herself.
  - This is because given x, V cannot distinguish between a transcript generated from interacting with P and one generated by $S(x)$.
  - Another (bad) view: if $x \in L$, V could just compute $S(x)$ herself and be convinced $x \in L$.

no guarantees if $x \notin L$

Zero-knowledge Proofs

Zero-knowledge Proofs

- Natural Question: what type of "indistinguishability"?

Zero-knowledge Proofs

- Natural Question: what type of "indistinguishability"?
  - *Perfect Zero-knowledge (PZK)*: $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ and $S(x)$ are *the same distribution.*

Zero-knowledge Proofs

- Natural Question: what type of "indistinguishability"?
  - *Perfect Zero-knowledge (PZK)*: $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ and $S(x)$ are *the same distribution.*
  - *Statistical Zero-knowledge (SZK)*: the distributions are *statistically indistinguishable* (negligibly indistinguishable by any unbounded algorithm).

Zero-knowledge Proofs

- Natural Question: what type of "indistinguishability"?
  - *Perfect Zero-knowledge (PZK)*: $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ and $S(x)$ are *the same distribution.*
  - *Statistical Zero-knowledge (SZK)*: the distributions are *statistically indistinguishable* (negligibly indistinguishable by any unbounded algorithm).
  - *Computational Zero-knowledge (CZK)*: the distributions are *computationally indistinguishable* (negligibly indistinguishable by any PPT algorithm).

Zero-knowledge Proofs

- Natural Question: what type of "indistinguishability"?
  - *Perfect Zero-knowledge (PZK)*: $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ and $S(x)$ are *the same distribution.*
  - *Statistical Zero-knowledge (SZK)*: the distributions are *statistically indistinguishable* (negligibly indistinguishable by any unbounded algorithm).
  - *Computational Zero-knowledge (CZK)*: the distributions are *computationally indistinguishable* (negligibly indistinguishable by any PPT algorithm).
- Natural Question: interactive *proof* or *argument*?

Zero-knowledge Proofs

- Natural Question: what type of "indistinguishability"?
  - *Perfect Zero-knowledge (PZK)*: $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ and $S(x)$ are *the same distribution.*
  - *Statistical Zero-knowledge (SZK)*: the distributions are *statistically indistinguishable* (negligibly indistinguishable by any unbounded algorithm).
  - *Computational Zero-knowledge (CZK)*: the distributions are *computationally indistinguishable* (negligibly indistinguishable by any PPT algorithm).
- Natural Question: interactive *proof* or *argument*?
  - Zero-knowledge is defined the same for both.

Zero-knowledge Proofs

- Natural Question: what type of "indistinguishability"?
  - *Perfect Zero-knowledge (PZK)*: $\text{View}_{V^*}(\langle P, V^* \rangle(x))$ and $S(x)$ are *the same distribution.*
  - *Statistical Zero-knowledge (SZK)*: the distributions are *statistically indistinguishable* (negligibly indistinguishable by any unbounded algorithm).
  - *Computational Zero-knowledge (CZK)*: the distributions are *computationally indistinguishable* (negligibly indistinguishable by any PPT algorithm).
- Natural Question: interactive *proof* or *argument*?
  - Zero-knowledge is defined the same for both.
- We now have 6 different meanings of "zero-knowledge proof;" we'll see more.

Next Time: Examples of Zero-knowledge Proofs