Zero-Knowledge Proofs II

601.442/642 Modern Cryptography

26th March 2026

Recap: Proof Systems

Proof System

A proof system for a language $L \subset \{0,1\}^*$ is a pair of interactive algorithms (P, V), where V is a PPT algorithm, if it satisfies the following properties.

Recap: Proof Systems

Proof System

A proof system for a language $L \subset \{0,1\}^*$ is a pair of interactive algorithms (P, V), where V is a PPT algorithm, if it satisfies the following properties.

- **Completeness:** For every $x \in L$,

$$\Pr\left[\mathsf{Out}_V\left[P(x) \leftrightarrow V(x)\right] = 1\right] = 1.$$

Recap: Proof Systems

Proof System

A proof system for a language $L \subset \{0,1\}^*$ is a pair of interactive algorithms (P, V), where V is a PPT algorithm, if it satisfies the following properties.

- **Completeness:** For every $x \in L$,

$$\Pr\left[\mathsf{Out}_V\left[P(x) \leftrightarrow V(x)\right] = 1\right] = 1.$$

- **Soundness:** There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $x \notin L$ and all $\widehat{P}$

$$\Pr\left[\mathsf{Out}_V\left[\widehat{P}(x) \leftrightarrow V(x)\right] = 1\right] \leq \mathsf{negl}(|x|).$$

Recap: Proof Systems

Proof System

A proof system for a language $L \subset \{0,1\}^*$ is a pair of interactive algorithms (P, V), where V is a PPT algorithm, if it satisfies the following properties.

- **Completeness:** For every $x \in L$,

$$\Pr\left[\mathrm{Out}_V\left[P(x) \leftrightarrow V(x)\right] = 1\right] = 1.$$

- **Soundness:** There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $x \notin L$ and all $\widehat{P}$

$$\Pr\left[\mathrm{Out}_V\left[\widehat{P}(x) \leftrightarrow V(x)\right] = 1\right] \leq \mathsf{negl}(|x|).$$

The focus of a proof system is the verification procedure. We want verification to be efficient.

Recap: Proof Systems

Proof System

A proof system for a language $L \subset \{0,1\}^*$ is a pair of interactive algorithms (P, V), where V is a PPT algorithm, if it satisfies the following properties.

- **Completeness:** For every $x \in L$,

$$\Pr\left[\mathrm{Out}_V\left[P(x) \leftrightarrow V(x)\right] = 1\right] = 1.$$

- **Soundness:** There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $x \notin L$ and all $\widehat{P}$

$$\Pr\left[\mathrm{Out}_V\left[\widehat{P}(x) \leftrightarrow V(x)\right] = 1\right] \leq \mathsf{negl}(|x|).$$

The focus of a proof system is the verification procedure. We want verification to be efficient.

Prover is not required to be efficient (similar to definition of NP).

Recap: Proof Systems

Proof System

A proof system for a language $L \subset \{0,1\}^*$ is a pair of interactive algorithms (P, V), where V is a PPT algorithm, if it satisfies the following properties.

- **Completeness:** For every $x \in L$,

$$\Pr\left[\mathrm{Out}_V\left[P(x) \leftrightarrow V(x)\right] = 1\right] = 1.$$

- **Soundness:** There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $x \notin L$ and all $\widehat{P}$

$$\Pr\left[\mathrm{Out}_V\left[\widehat{P}(x) \leftrightarrow V(x)\right] = 1\right] \leq \mathsf{negl}(|x|).$$

The focus of a proof system is the verification procedure. We want verification to be efficient.

Prover is not required to be efficient (similar to definition of NP).

Are both properties required?

Recap: The Power of Interaction and Randomness

- Proof system for languages beyond what is captured by non-interactive proofs.
  - Languages in NP have a non-interactive proof.
  - Interactive proofs can prove statements in languages not known to be in NP.
    - Single prover [Shamir]: IP = PSPACE.
    - Multiple provers [Babai-Fortnow-Lund]: MIP = NEXP.

Recap: The Power of Interaction and Randomness

- Proof system for languages beyond what is captured by non-interactive proofs.
  - Languages in NP have a non-interactive proof.
  - Interactive proofs can prove statements in languages not known to be in NP.
    - Single prover [Shamir]: $\mathsf{IP} = \mathsf{PSPACE}$.
    - Multiple provers [Babai-Fortnow-Lund]: $\mathsf{MIP} = \mathsf{NEXP}$.
- Achieving privacy guarantee for the prover.
  - **Zero knowledge:** Verifier learns nothing from the proof beyond the validity of the statement.

Recap: The Power of Interaction and Randomness

- Proof system for languages beyond what is captured by non-interactive proofs.
  - Languages in NP have a non-interactive proof.
  - Interactive proofs can prove statements in languages not known to be in NP.
    - Single prover [Shamir]: IP = PSPACE.
    - Multiple provers [Babai-Fortnow-Lund]: MIP = NEXP.
- Achieving privacy guarantee for the prover.
  - **Zero knowledge:** Verifier learns nothing from the proof beyond the validity of the statement.

Our goal is to construct zero-knowledge proofs.

Towards Zero-Knowledge

The verifier must not gain any knowledge beyond the validity of the statement.

Towards Zero-Knowledge

The verifier must not gain any knowledge beyond the validity of the statement.

- How do we capture “not gaining knowledge”?

Towards Zero-Knowledge

The verifier must not gain any knowledge beyond the validity of the statement.

- How do we capture “not gaining knowledge”?
- What is knowledge?

Towards Zero-Knowledge

The verifier must not gain any knowledge beyond the validity of the statement.

- How do we capture “not gaining knowledge”?
- What is knowledge? This seems hard to define formally.

Towards Zero-Knowledge

The verifier must not gain any knowledge beyond the validity of the statement.

- How do we capture “not gaining knowledge”?
- What is knowledge? This seems hard to define formally.
  - Our goal is not to define knowledge, nor to characterize every scenario in which knowledge is not gained.

Towards Zero-Knowledge

The verifier must not gain any knowledge beyond the validity of the statement.

- How do we capture “not gaining knowledge”?
- What is knowledge? This seems hard to define formally.
  - Our goal is not to define knowledge, nor to characterize every scenario in which knowledge is not gained.
  - Instead, we want to identify a sufficient condition under which we can definitively say no knowledge is gained.

Towards Zero-Knowledge

The verifier must not gain any knowledge beyond the validity of the statement.

- How do we capture “not gaining knowledge”?
- What is knowledge? This seems hard to define formally.
  - Our goal is not to define knowledge, nor to characterize every scenario in which knowledge is not gained.
  - Instead, we want to identify a sufficient condition under which we can definitively say no knowledge is gained.
- When can we definitely say the verifier gains no knowledge?

Towards Zero-Knowledge

The verifier must not gain any knowledge beyond the validity of the statement.

- How do we capture “not gaining knowledge”?
- What is knowledge? This seems hard to define formally.
  - Our goal is not to define knowledge, nor to characterize every scenario in which knowledge is not gained.
  - Instead, we want to identify a sufficient condition under which we can definitively say no knowledge is gained.
- When can we definitely say the verifier gains no knowledge?
  - **Key Idea:** If the verifier could have computed it on its own, it didn't learn it from the prover.

Towards Zero-Knowledge

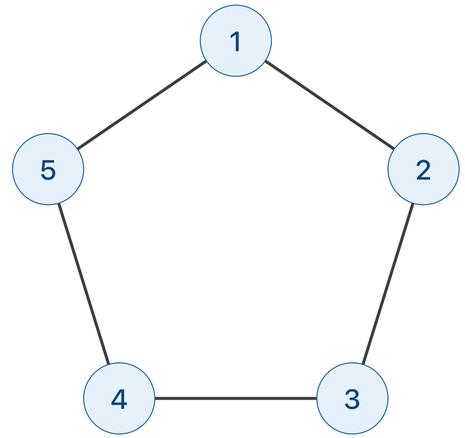
> The verifier must not gain any knowledge beyond the validity of the statement.

- How do we capture “not gaining knowledge”?
- What is knowledge? This seems hard to define formally.
  - Our goal is not to define knowledge, nor to characterize every scenario in which knowledge is not gained.
  - Instead, we want to identify a sufficient condition under which we can definitively say no knowledge is gained.
- When can we definitely say the verifier gains no knowledge?
  - **Key Idea:** If the verifier could have computed it on its own, it didn't learn it from the prover.
  - In other words, the verifier gains knowledge only if it receives the result of a computation that is infeasible for it.

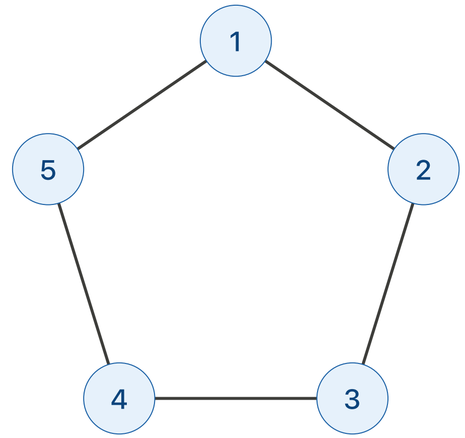
Zero-Knowledge Proof for Graph Isomorphism

Zero-Knowledge Proof for Graph Isomorphism

- Graph $G = (V, E)$ consists of a vertex set V and edge set $E \subseteq V \times V$.

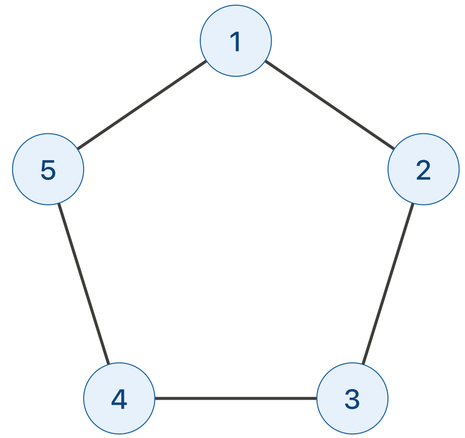


Zero-Knowledge Proof for Graph Isomorphism



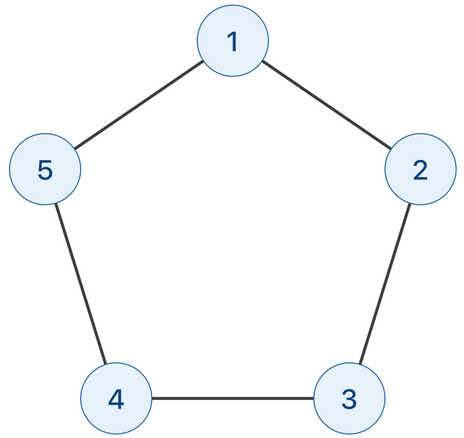
- Graph $G = (V, E)$ consists of a vertex set V and edge set $E \subseteq V \times V$.
  - We will consider simple undirected graphs.

Zero-Knowledge Proof for Graph Isomorphism



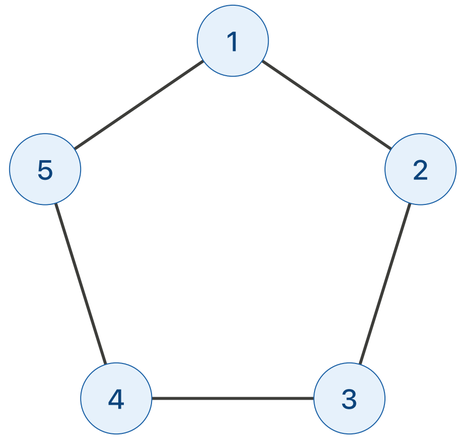
- Graph $G = (V, E)$ consists of a vertex set V and edge set $E \subseteq V \times V$.
  - We will consider simple undirected graphs.
  - $|V| = n$ and $|E| = m$.

Zero-Knowledge Proof for Graph Isomorphism



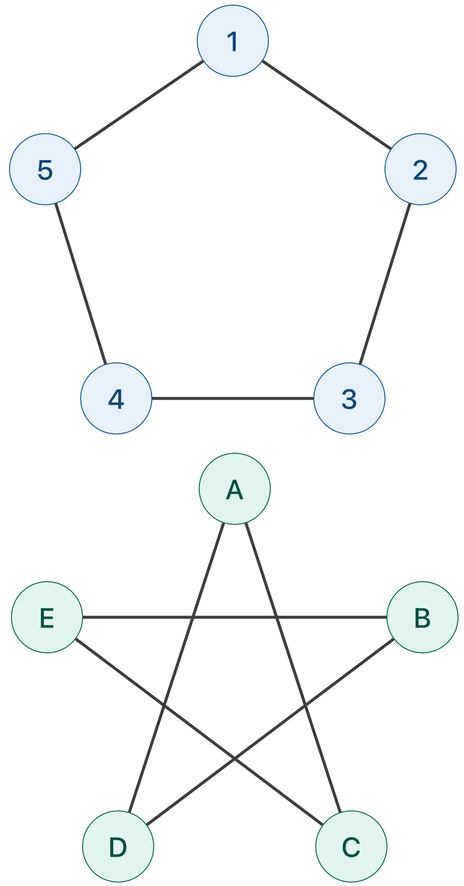
- Graph $G = (V, E)$ consists of a vertex set V and edge set $E \subseteq V \times V$.
  - We will consider simple undirected graphs.
  - $|V| = n$ and $|E| = m$.
- Let Perm_n denote the set of all permutations ϕ on the vertices.

Zero-Knowledge Proof for Graph Isomorphism



- Graph $G = (V, E)$ consists of a vertex set V and edge set $E \subseteq V \times V$.
  - We will consider simple undirected graphs.
  - $|V| = n$ and $|E| = m$.
- Let Perm_n denote the set of all permutations ϕ on the vertices.
- **Graph Isomorphism:** $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$ are isomorphic if there exists a permutation $\phi \in \mathsf{Perm}_n$ such that
  - $V_1 = \{\phi(v) \mid v \in V_0\}$, and
  - $E_1 = \{\left(\phi(v_1), \phi(v_2)\right) \mid (v_1, v_2) \in E_0\}$.
  - Alternatively, $G_1 = \phi(G_0)$.

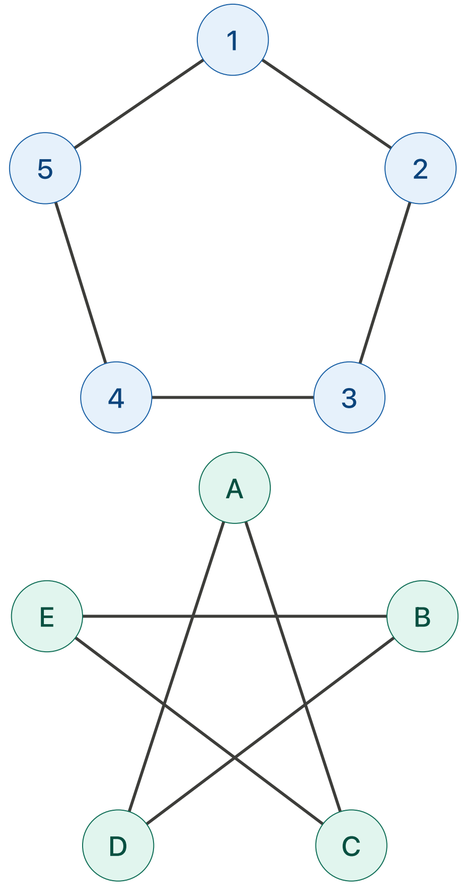
Zero-Knowledge Proof for Graph Isomorphism



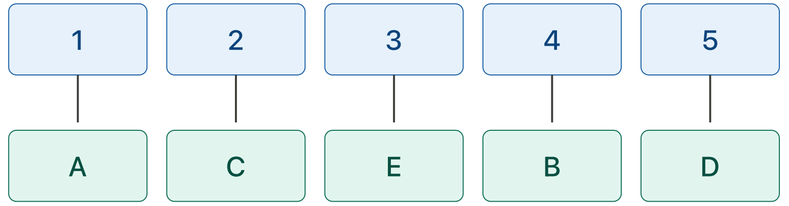
- Graph $G = (V, E)$ consists of a vertex set V and edge set $E \subseteq V \times V$.
  - We will consider simple undirected graphs.
  - $|V| = n$ and $|E| = m$.
- Let Perm_n denote the set of all permutations ϕ on the vertices.
- **Graph Isomorphism:** $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$ are isomorphic if there exists a permutation $\phi \in \mathsf{Perm}_n$ such that
  - $V_1 = \{\phi(v) \mid v \in V_0\}$, and
  - $E_1 = \{\big(\phi(v_1), \phi(v_2)\big) \mid (v_1, v_2) \in E_0\}$.
  - Alternatively, $G_1 = \phi(G_0)$.

Zero-Knowledge Proof for Graph Isomorphism

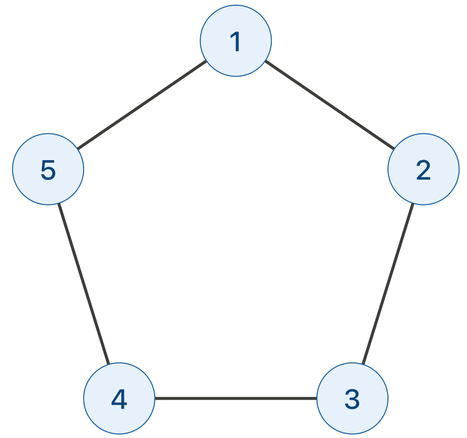
- Graph $G = (V, E)$ consists of a vertex set V and edge set $E \subseteq V \times V$.
  - We will consider simple undirected graphs.
  - $|V| = n$ and $|E| = m$.
- Let Perm_n denote the set of all permutations ϕ on the vertices.
- **Graph Isomorphism:** $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$ are isomorphic if there exists a permutation $\phi \in \mathsf{Perm}_n$ such that
  - $V_1 = \{\phi(v) \mid v \in V_0\}$, and
  - $E_1 = \{\big(\phi(v_1), \phi(v_2)\big) \mid (v_1, v_2) \in E_0\}$.
  - Alternatively, $G_1 = \phi(G_0)$.

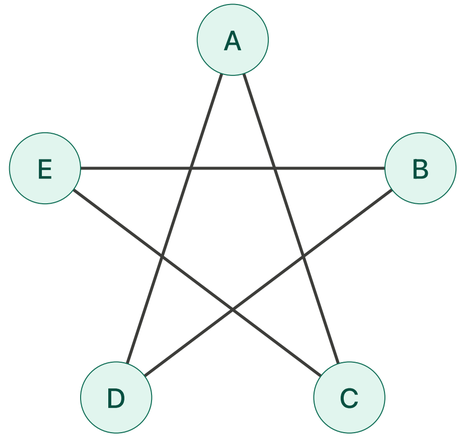


Isomorphism ϕ : G → H

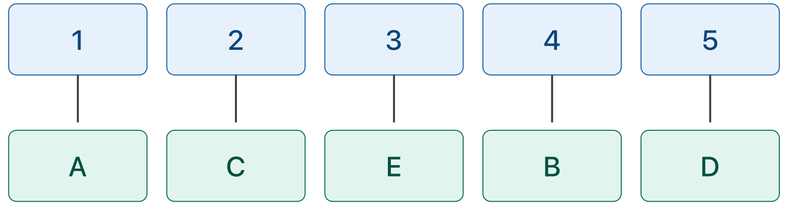


Zero-Knowledge Proof for Graph Isomorphism



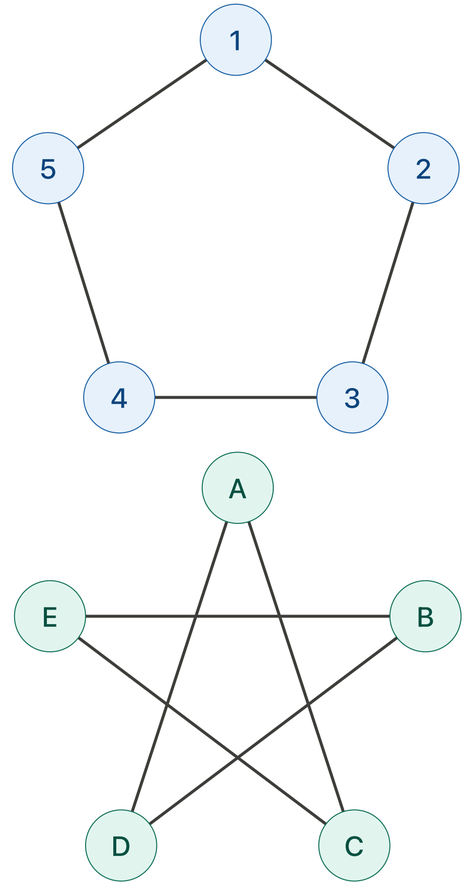


Isomorphism $\phi : G \rightarrow H$

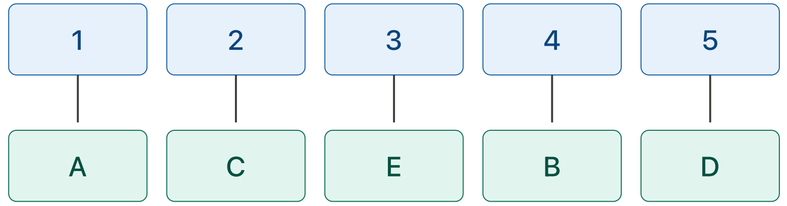


Zero-Knowledge Proof for Graph Isomorphism

- Graph Isomorphism is in NP.

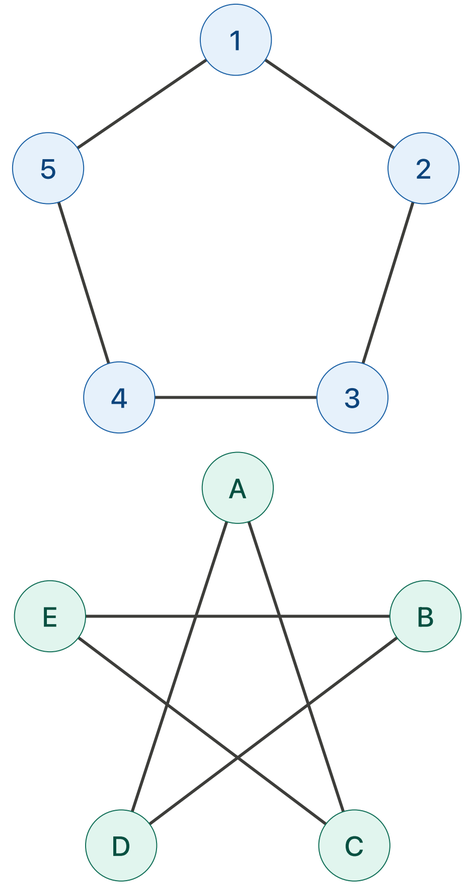


Isomorphism $\phi : G \to H$

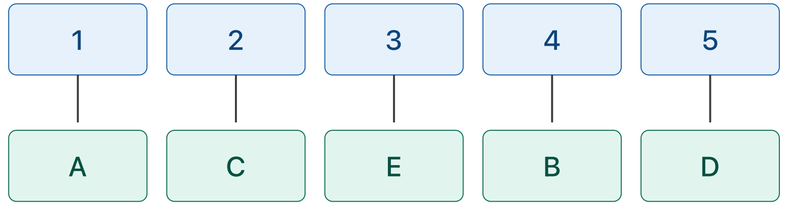


Zero-Knowledge Proof for Graph Isomorphism

- Graph Isomorphism is in NP.
  - The NP proof or witness is the permutation ϕ.

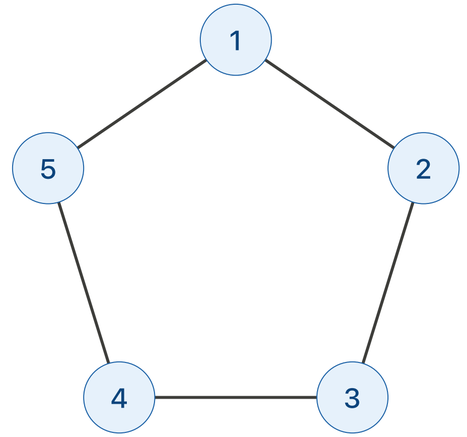


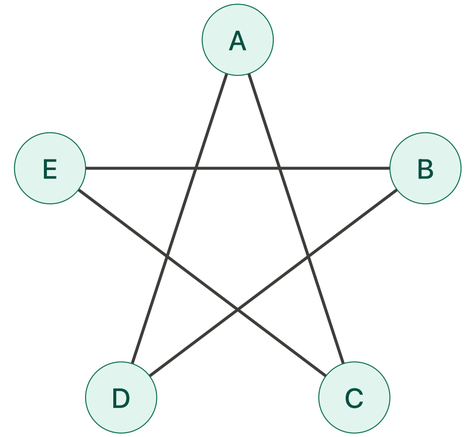
Isomorphism ϕ : G → H



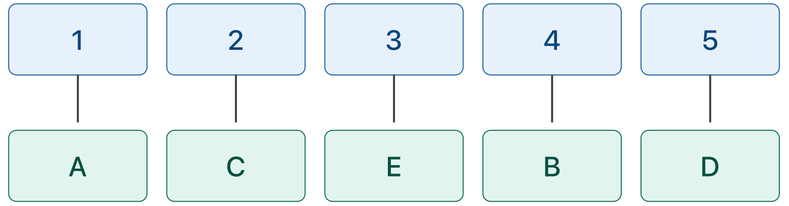
Zero-Knowledge Proof for Graph Isomorphism

- Graph Isomorphism is in NP.
  - The NP proof or witness is the permutation ϕ.
- Graph isomorphism is not known to have a PPT algorithm.



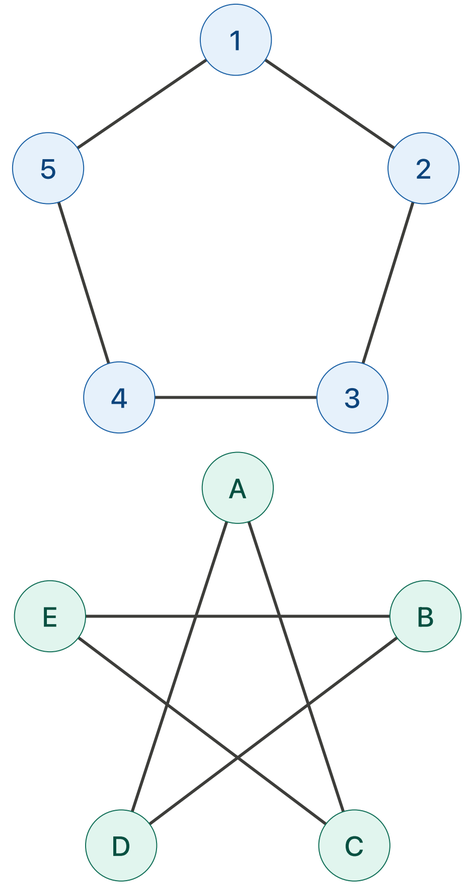


Isomorphism ϕ : G → H

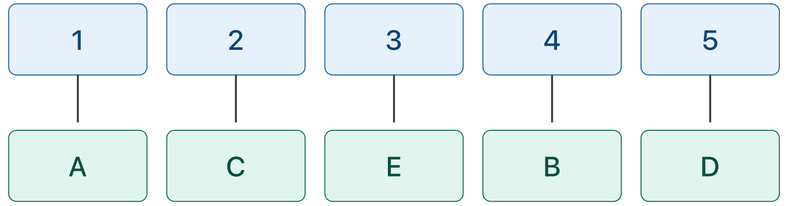


Zero-Knowledge Proof for Graph Isomorphism

- Graph Isomorphism is in NP.
  - The NP proof or witness is the permutation ϕ.
- Graph isomorphism is not known to have a PPT algorithm.
- **Goal:** Zero-knowledge proof for graph isomorphism.

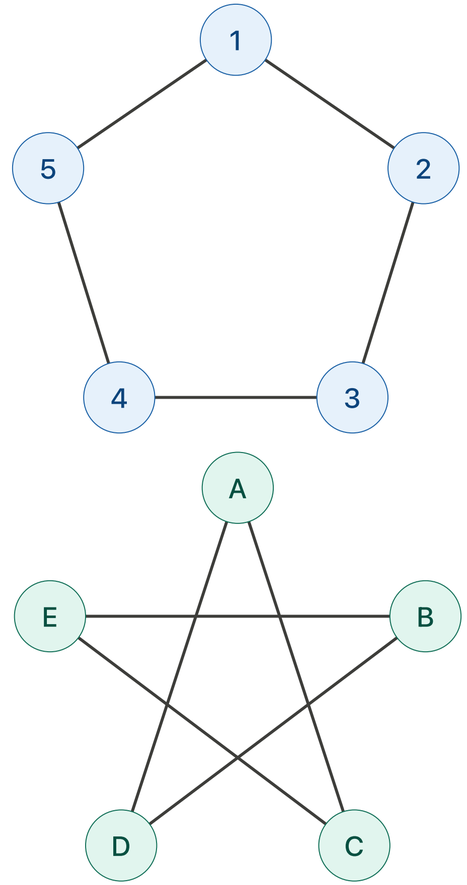


Isomorphism $\phi : G \rightarrow H$

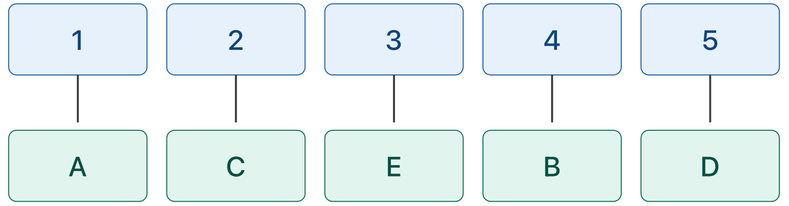


Zero-Knowledge Proof for Graph Isomorphism

- Graph Isomorphism is in NP.
  - The NP proof or witness is the permutation ϕ.
- Graph isomorphism is not known to have a PPT algorithm.
- **Goal:** Zero-knowledge proof for graph isomorphism.
  - **Completeness:** Prover will use ϕ to convince the verifier.

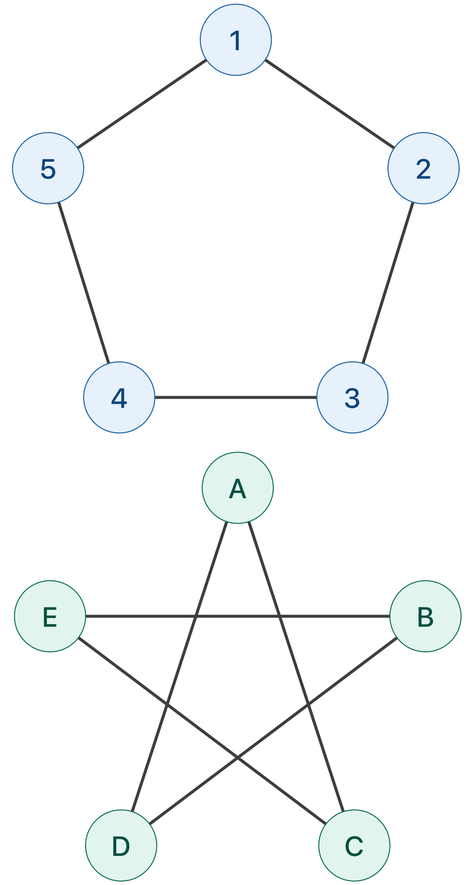


Isomorphism ϕ : G → H

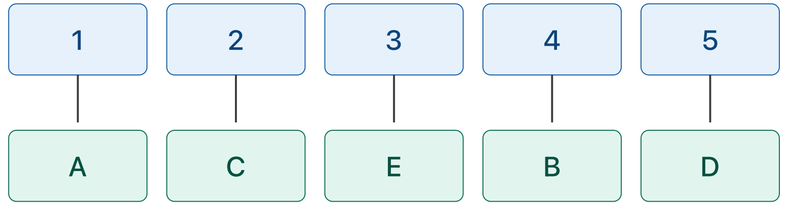


Zero-Knowledge Proof for Graph Isomorphism

- Graph Isomorphism is in NP.
  - The NP proof or witness is the permutation ϕ.
- Graph isomorphism is not known to have a PPT algorithm.
- **Goal:** Zero-knowledge proof for graph isomorphism.
  - **Completeness:** Prover will use ϕ to convince the verifier.
  - **Soundness:** If the graphs are not isomorphic, the verifier should reject with high probability.

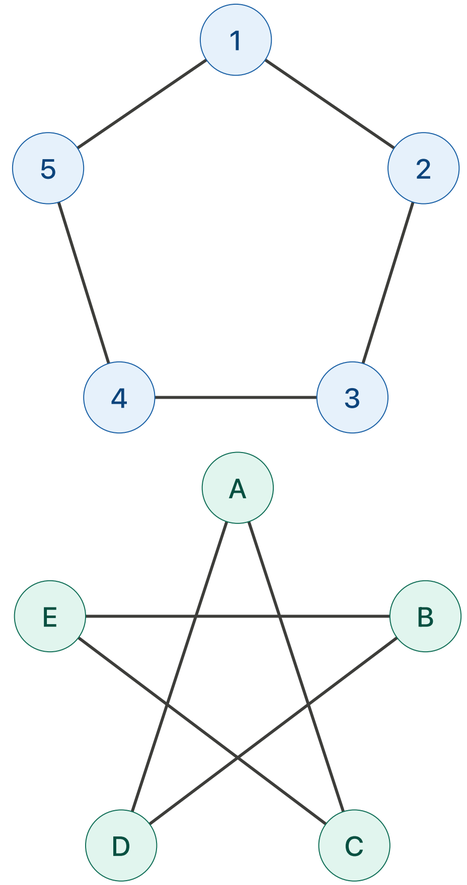


Isomorphism ϕ : G → H

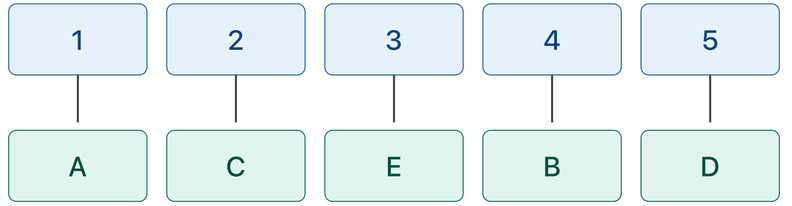


Zero-Knowledge Proof for Graph Isomorphism

- Graph Isomorphism is in NP.
  - The NP proof or witness is the permutation ϕ.
- Graph isomorphism is not known to have a PPT algorithm.
- **Goal:** Zero-knowledge proof for graph isomorphism.
  - **Completeness:** Prover will use ϕ to convince the verifier.
  - **Soundness:** If the graphs are not isomorphic, the verifier should reject with high probability.
  - **Zero-knowledge:** Verifier must not learn anything beyond the fact that the graphs are isomorphic.
    - Importantly, it shouldn't learn ϕ since it cannot compute it from G_0 and G_1.



Isomorphism ϕ : G → H



Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Sample $b \xleftarrow{\$} \{0,1\}$

b

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Prove G' is isomorphic to G_b.

b

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Sample $b \xleftarrow{\$} \{0,1\}$

Compute ϕ s.t.
$G_1 = \phi(G_0)$

b

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Sample $b \xleftarrow{\$} \{0,1\}$

Compute ϕ s.t. $G_1 = \phi(G_0)$

b

If $b = 0$, $\phi' :=$

If $b = 1$, $\phi' :=$

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

b

If $b = 0$, $\phi' :=$

If $b = 1$, $\phi' :=$

Accept if $G' = \phi'(G_b)$

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' :=$

Accept if $G' = \phi'(G_b)$

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

Zero-Knowledge Proof for Graph Isomorphism

- **Completeness**

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$ →

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$ →

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

- **Completeness**
  - If $b = 0$

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

→ Accept if $G' = \phi'(G_b)$

- **Completeness**
  - If $b = 0$

$$\phi'(G_0) = \psi(\phi(G_0)) = \psi(G_1) = G'.$$

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

→ Accept if $G' = \phi'(G_b)$

- **Completeness**
  - If $b = 0$

$$\phi'(G_0) = \psi(\phi(G_0)) = \psi(G_1) = G'.$$

  - If $b = 1$

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Sample $b \xleftarrow{\$} \{0,1\}$

Compute ϕ s.t. $G_1 = \phi(G_0)$

← b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

→ Accept if $G' = \phi'(G_b)$

- **Completeness**
  - If $b = 0$

$$\phi'(G_0) = \psi(\phi(G_0)) = \psi(G_1) = G'.$$

  - If $b = 1$

$$\phi'(G_1) = \psi(G_1) = G'.$$

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \text{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

- **Completeness**
  - If $b = 0$

$$\phi'(G_0) = \psi(\phi(G_0)) = \psi(G_1) = G'.$$

  - If $b = 1$

$$\phi'(G_1) = \psi(G_1) = G'.$$

- **Soundness**

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

- **Completeness**
  - If $b = 0$

$$\phi'(G_0) = \psi(\phi(G_0)) = \psi(G_1) = G'.$$

  - If $b = 1$

$$\phi'(G_1) = \psi(G_1) = G'.$$

- **Soundness**
  - G' can be isomorphic to at most one among G_0 and G_1.
  - Say G' is isomorphic to $G_{b'}$.

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \text{Perm}_n$

$G' := \psi(G_1)$

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

- **Completeness**
  - If $b = 0$

$$\phi'(G_0) = \psi(\phi(G_0)) = \psi(G_1) = G'.$$

  - If $b = 1$

$$\phi'(G_1) = \psi(G_1) = G'.$$

- **Soundness**
  - G' can be isomorphic to at most one among G_0 and G_1.
  - Say G' is isomorphic to $G_{b'}$.
  - Verifier rejects when $b' \neq b \Longrightarrow$ with probability 1/2.

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \text{Perm}_n$

$G' := \psi(G_1)$

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

- **Completeness**
  - If $b = 0$

$$\phi'(G_0) = \psi(\phi(G_0)) = \psi(G_1) = G'.$$

  - If $b = 1$

$$\phi'(G_1) = \psi(G_1) = G'.$$

- **Soundness**
  - G' can be isomorphic to at most one among G_0 and G_1.
  - Say G' is isomorphic to $G_{b'}$.
  - Verifier rejects when $b' \neq b \Longrightarrow$ with probability 1/2.
  - Repeat to amplify probability.

Zero-Knowledge Proof for Graph Isomorphism

- **Zero-Knowledge**

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$ →

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

Zero-Knowledge Proof for Graph Isomorphism

- **Zero-Knowledge**
  - Why does the verifier not gain any knowledge?

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Sample $b \xleftarrow{\$} \{0,1\}$

Compute ϕ s.t. $G_1 = \phi(G_0)$

← b

If $b = 0$, $\phi' := \psi \circ \phi$ →

Accept if $G' = \phi'(G_b)$

If $b = 1$, $\phi' := \psi$

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

- **Zero-Knowledge**
  - Why does the verifier not gain any knowledge?
  - Does it learn ϕ?

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

→ Accept if $G' = \phi'(G_b)$

- **Zero-Knowledge**
  - Why does the verifier not gain any knowledge?
  - Does it learn ϕ?
  - When $b = 1$: Verifier receives $\psi(G_1)$ and ψ.

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Sample $b \xleftarrow{\$} \{0,1\}$

Compute ϕ s.t. $G_1 = \phi(G_0)$

← b

If $b = 0$, $\phi' := \psi \circ \phi$ →

Accept if $G' = \phi'(G_b)$

If $b = 1$, $\phi' := \psi$

- **Zero-Knowledge**
  - Why does the verifier not gain any knowledge?
  - Does it learn ϕ?
  - When $b = 1$: Verifier receives $\psi(G_1)$ and ψ.
    - Messages are independent of ϕ.

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$ →

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

- **Zero-Knowledge**
  - Why does the verifier not gain any knowledge?
  - Does it learn ϕ?
  - When $b = 1$: Verifier receives $\psi(G_1)$ and ψ.
    - Messages are independent of ϕ.
  - When $b = 0$: Verifier receives $\psi(G_1)$ and $\psi \circ \phi$.

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

- **Zero-Knowledge**
  - Why does the verifier not gain any knowledge?
  - Does it learn ϕ?
  - When $b = 1$: Verifier receives $\psi(G_1)$ and ψ.
    - Messages are independent of ϕ.
  - When $b = 0$: Verifier receives $\psi(G_1)$ and $\psi \circ \phi$.
    - ϕ is masked by a random permutation ψ.

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$ →

If $b = 1$, $\phi' := \psi$ →

Accept if $G' = \phi'(G_b)$

- **Zero-Knowledge**
  - Why does the verifier not gain any knowledge?
  - Does it learn ϕ?
  - When $b = 1$: Verifier receives $\psi(G_1)$ and ψ.
    - Messages are independent of ϕ.
  - When $b = 0$: Verifier receives $\psi(G_1)$ and $\psi \circ \phi$.
    - ϕ is masked by a random permutation ψ.
    - The verifier could compute ψ from G'.

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \text{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$ →

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

- **Zero-Knowledge**
  - Why does the verifier not gain any knowledge?
  - Does it learn ϕ?
  - When $b = 1$: Verifier receives $\psi(G_1)$ and ψ.
    - Messages are independent of ϕ.
  - When $b = 0$: Verifier receives $\psi(G_1)$ and $\psi \circ \phi$.
    - ϕ is masked by a random permutation ψ.
    - The verifier could compute ψ from G'.
      - This is equivalent to solving graph isomorphism for (G', G_1).

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \text{Perm}_n$

$G' := \psi(G_1)$ →

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

← b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$ →

Accept if $G' = \phi'(G_b)$

- **Zero-Knowledge**
  - Why does the verifier not gain any knowledge?
  - Does it learn ϕ?
  - When $b = 1$: Verifier receives $\psi(G_1)$ and ψ.
    - Messages are independent of ϕ.
  - When $b = 0$: Verifier receives $\psi(G_1)$ and $\psi \circ \phi$.
    - ϕ is masked by a random permutation ψ.
    - The verifier could compute ψ from G'.
      - This is equivalent to solving graph isomorphism for (G', G_1).
      - But then it could solve graph isomorphism for (G_0, G_1) and compute ϕ by itself too.

Zero-Knowledge Proof for Graph Isomorphism

Statement $x = (G_0, G_1)$

Prover

Verifier

Sample $\psi \xleftarrow{\$} \text{Perm}_n$

$G' := \psi(G_1)$

Compute ϕ s.t. $G_1 = \phi(G_0)$

Sample $b \xleftarrow{\$} \{0,1\}$

b

If $b = 0$, $\phi' := \psi \circ \phi$

If $b = 1$, $\phi' := \psi$

Accept if $G' = \phi'(G_b)$

- **Zero-Knowledge**
  - Why does the verifier not gain any knowledge?
  - Does it learn ϕ?
  - When $b = 1$: Verifier receives $\psi(G_1)$ and ψ.
    - Messages are independent of ϕ.
  - When $b = 0$: Verifier receives $\psi(G_1)$ and $\psi \circ \phi$.
    - ϕ is masked by a random permutation ψ.
    - The verifier could compute ψ from G'.
      - This is equivalent to solving graph isomorphism for (G', G_1).
      - But then it could solve graph isomorphism for (G_0, G_1) and compute ϕ by itself too.
      - Learning ϕ is not gaining knowledge in this case.

The Simulation Paradigm

Anything a party could compute from its own inputs is not knowledge it gains by interacting with an external entity.

The Simulation Paradigm

Anything a party could compute from its own inputs is not knowledge it gains by interacting with an external entity.

x

Prover

x

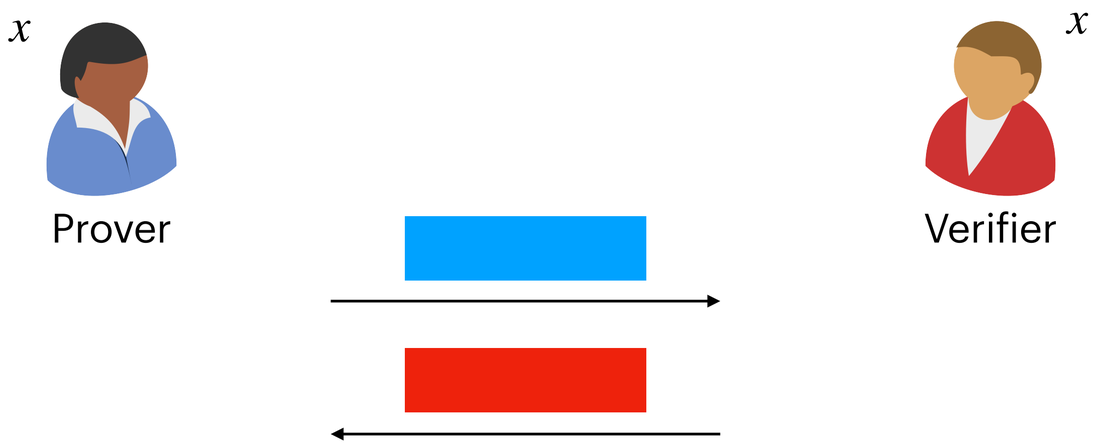
Verifier

The Simulation Paradigm

Anything a party could compute from its own inputs is not knowledge it gains by interacting with an external entity.

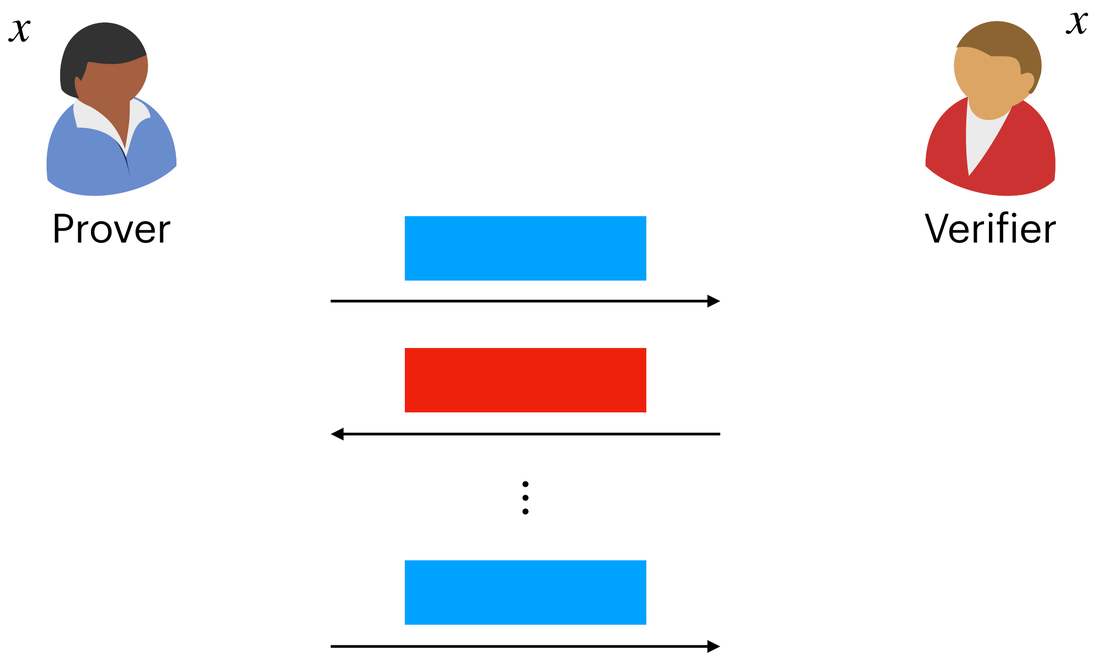
The Simulation Paradigm

Anything a party could compute from its own inputs is not knowledge it gains by interacting with an external entity.



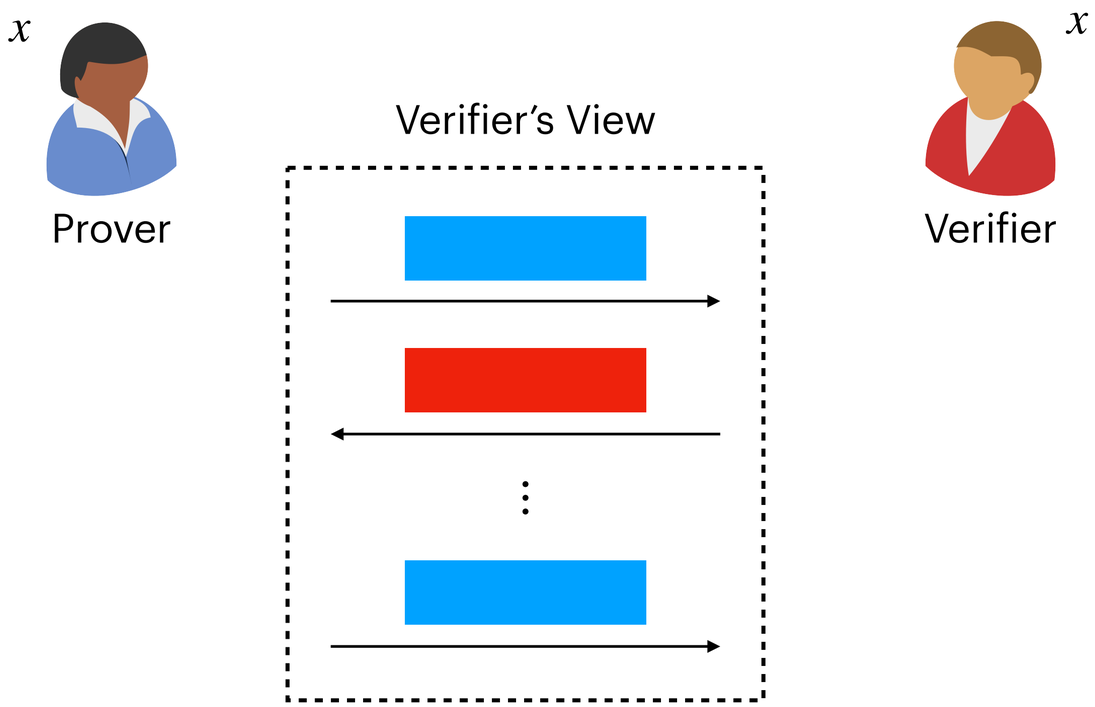
The Simulation Paradigm

Anything a party could compute from its own inputs is not knowledge it gains by interacting with an external entity.



The Simulation Paradigm

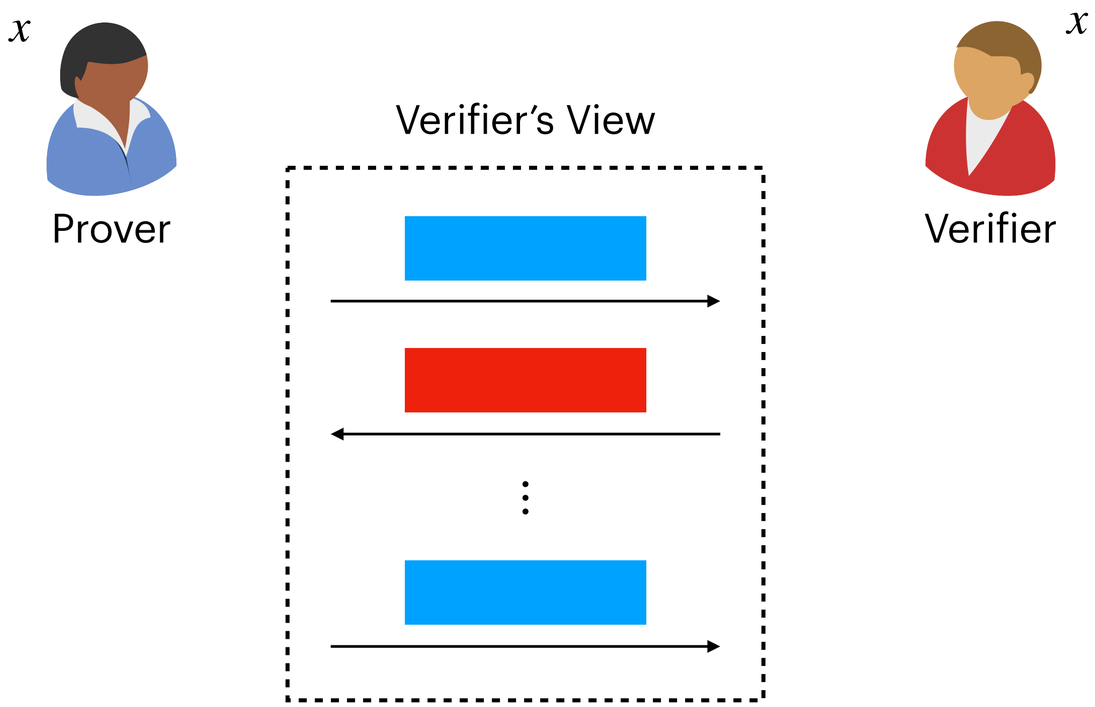
Anything a party could compute from its own inputs is not knowledge it gains by interacting with an external entity.



- If the verifier learns anything from interacting with the prover, then this knowledge must be reflected in the protocol transcript (the verifier's view).

The Simulation Paradigm

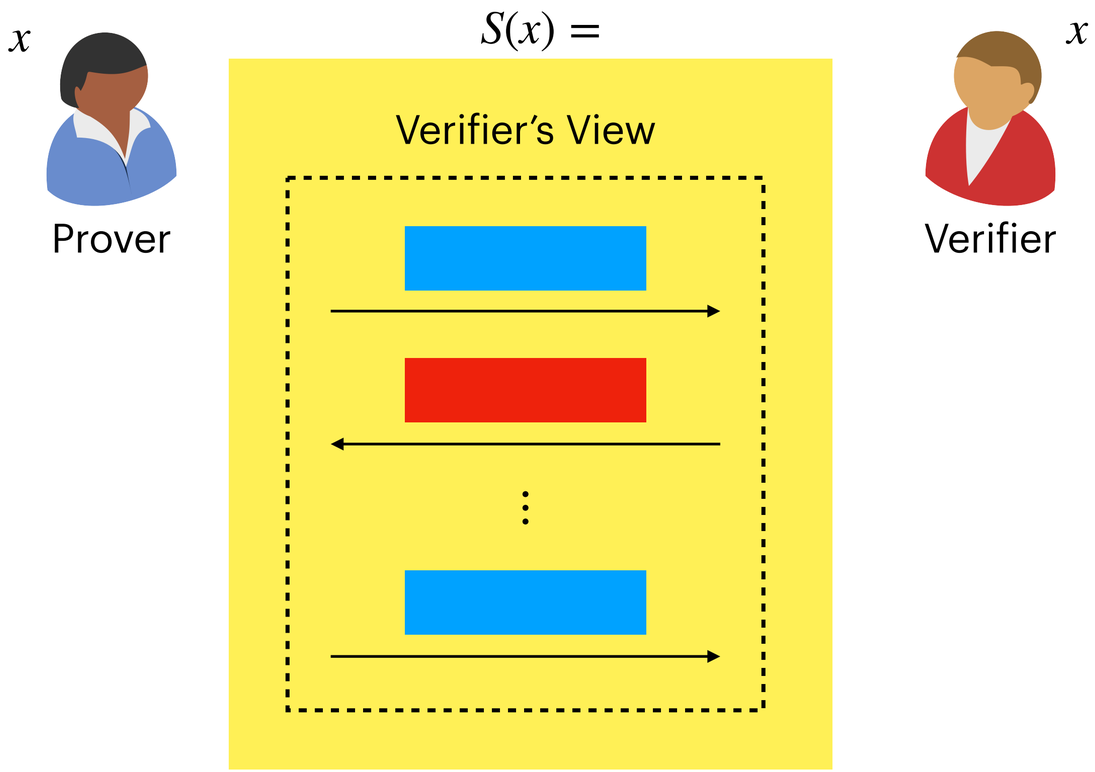
Anything a party could compute from its own inputs is not knowledge it gains by interacting with an external entity.



- If the verifier learns anything from interacting with the prover, then this knowledge must be reflected in the protocol transcript (the verifier's view).
- If the verifier gains no knowledge, then it could have computed the view from its own inputs.

The Simulation Paradigm

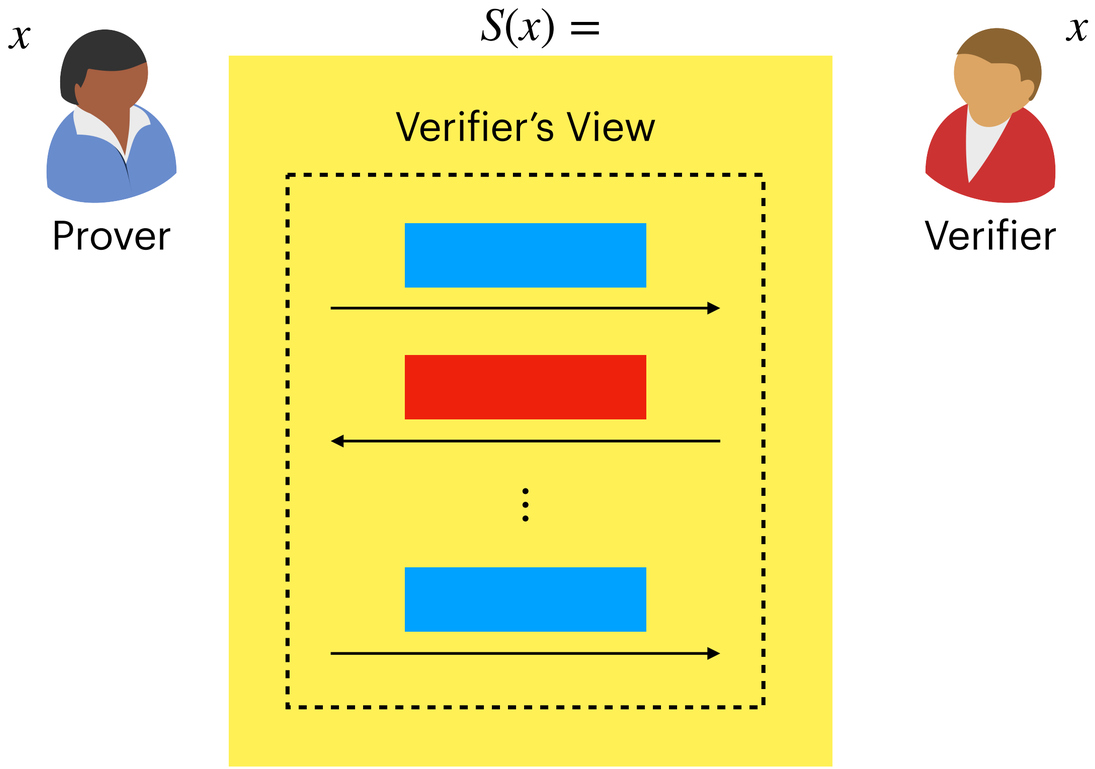
Anything a party could compute from its own inputs is not knowledge it gains by interacting with an external entity.



- If the verifier learns anything from interacting with the prover, then this knowledge must be reflected in the protocol transcript (the verifier's view).
- If the verifier gains no knowledge, then it could have computed the view from its own inputs.
- **Simulation paradigm:** There exists a PPT algorithm S, called the **simulator**, that generates the verifier's view using only the verifier's inputs.

The Simulation Paradigm

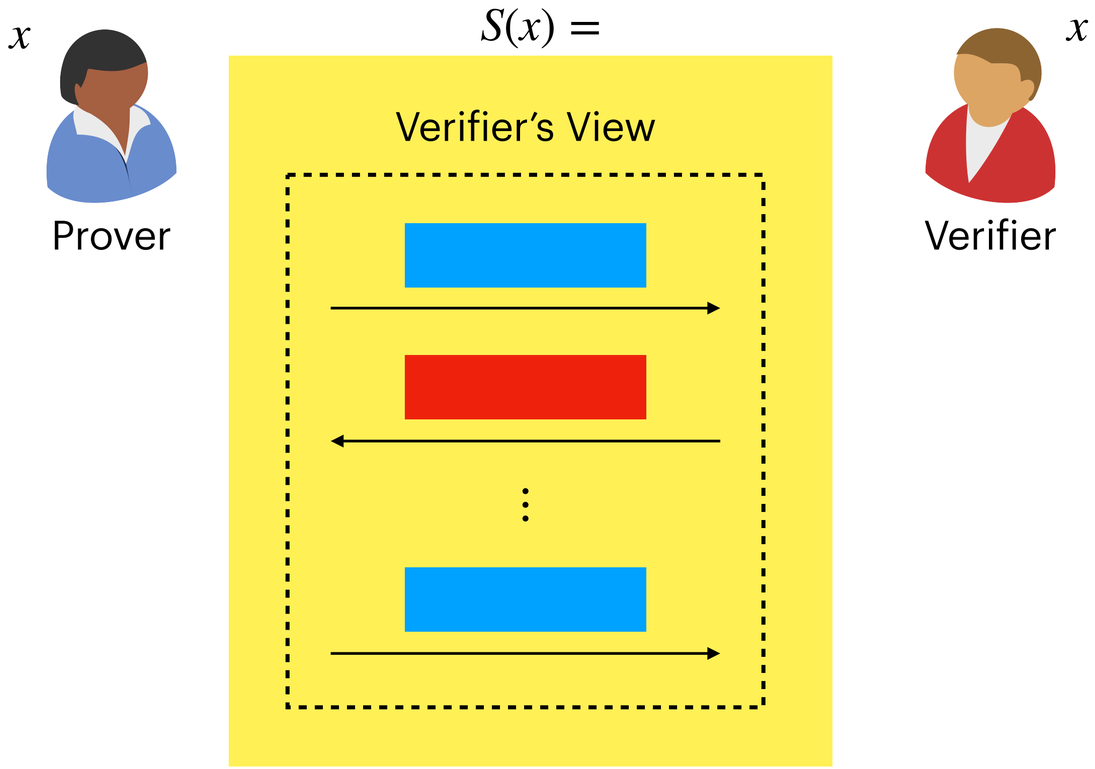
Anything a party could compute from its own inputs is not knowledge it gains by interacting with an external entity.



- If the verifier learns anything from interacting with the prover, then this knowledge must be reflected in the protocol transcript (the verifier's view).
- If the verifier gains no knowledge, then it could have computed the view from its own inputs.
- **Simulation paradigm:** There exists a PPT algorithm S, called the **simulator**, that generates the verifier's view using only the verifier's inputs.
- The verifier is a PPT machine, the existence of such a simulator means it could have efficiently computed the view by itself.

The Simulation Paradigm

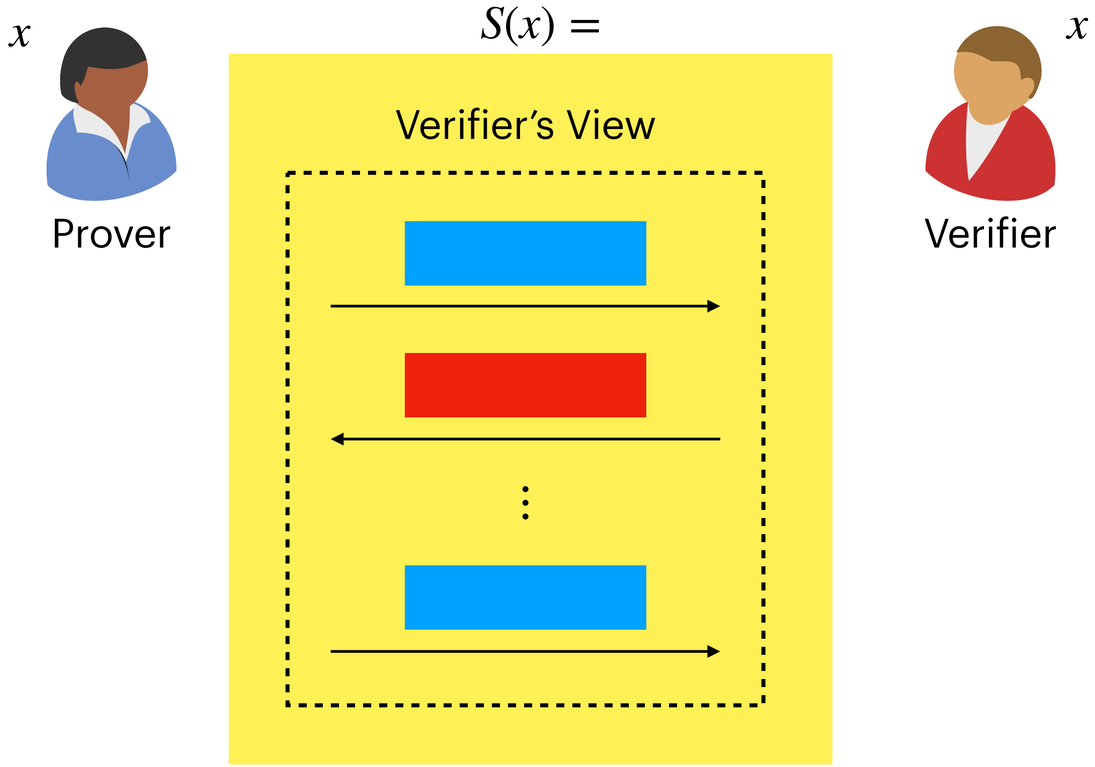
Anything a party could compute from its own inputs is not knowledge it gains by interacting with an external entity.



- If the verifier learns anything from interacting with the prover, then this knowledge must be reflected in the protocol transcript (the verifier's view).
- If the verifier gains no knowledge, then it could have computed the view from its own inputs.
- **Simulation paradigm:** There exists a PPT algorithm S, called the **simulator**, that generates the verifier's view using only the verifier's inputs.
- The verifier is a PPT machine, the existence of such a simulator means it could have efficiently computed the view by itself.
  - Interacting with the prover provides no knowledge.

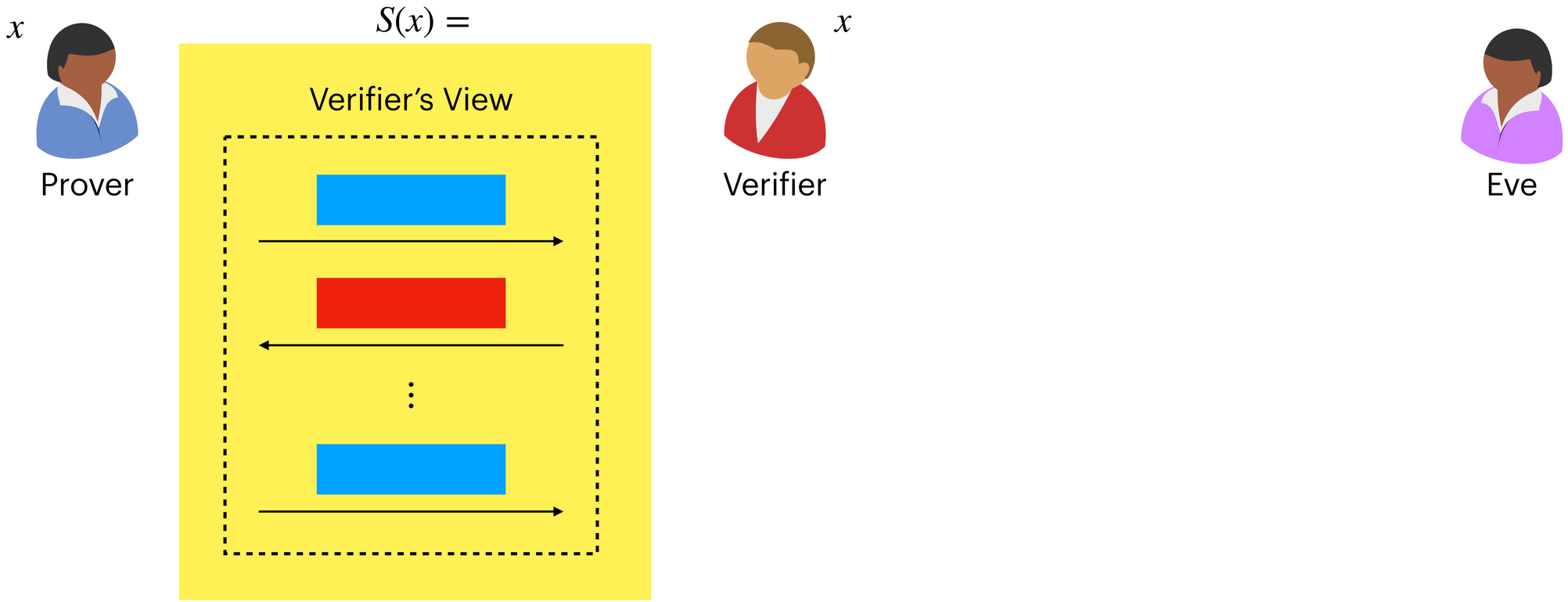
The Simulation Paradigm

- Paradox?
  - Protocol execution convinces the verifier of the validity of x.
  - The verifier could have generated the transcript by itself.



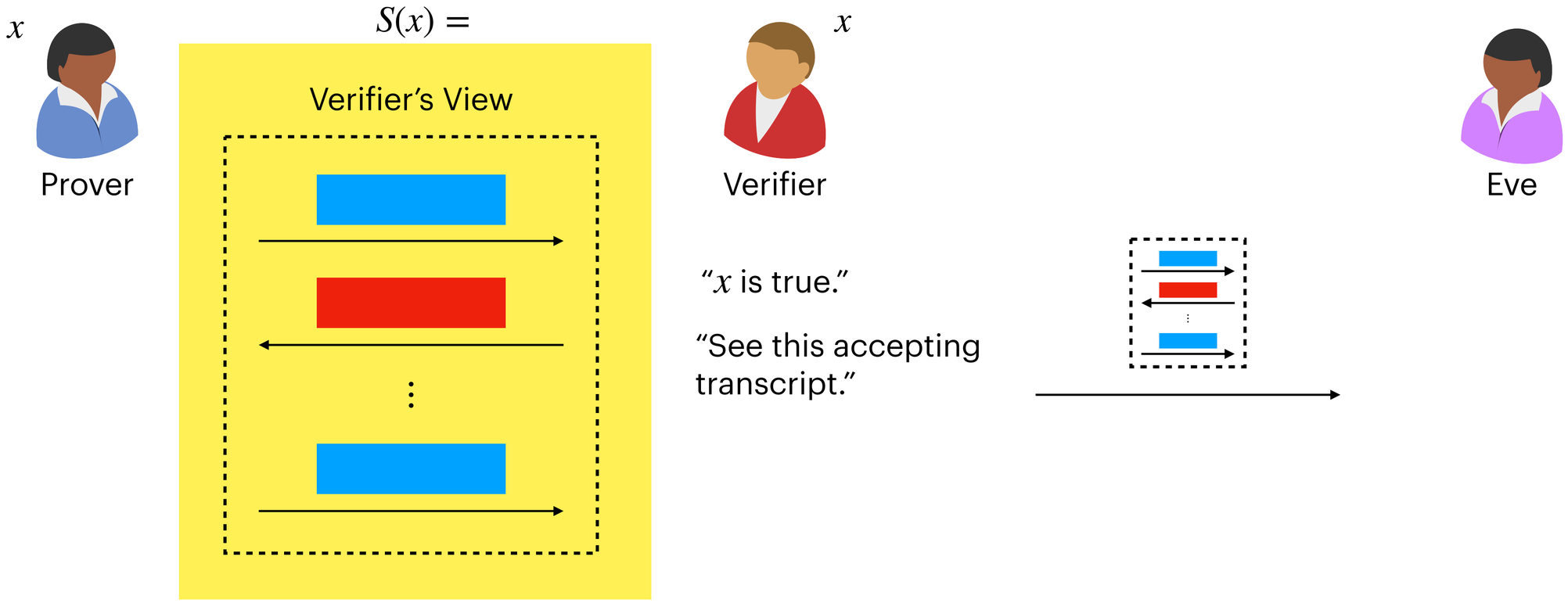
The Simulation Paradigm

- Paradox?
  - Protocol execution convinces the verifier of the validity of x.
  - The verifier could have generated the transcript by itself.



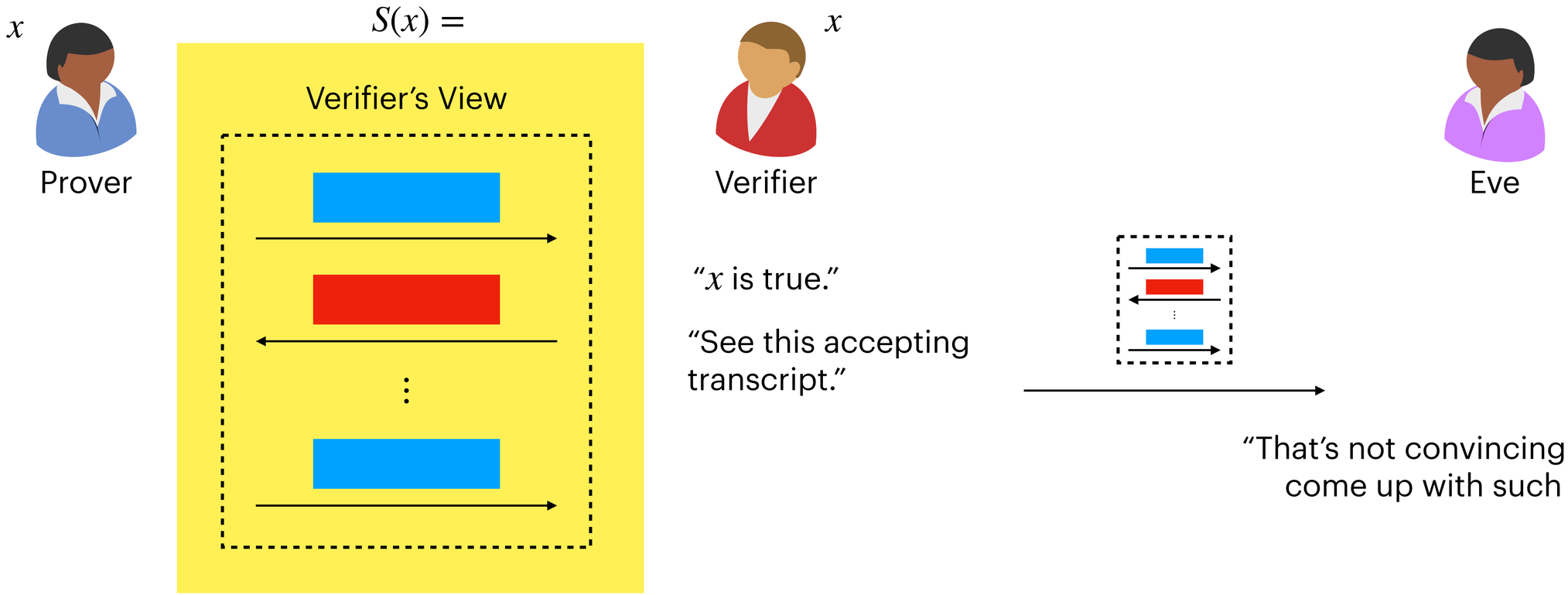
The Simulation Paradigm

- Paradox?
  - Protocol execution convinces the verifier of the validity of x.
  - The verifier could have generated the transcript by itself.



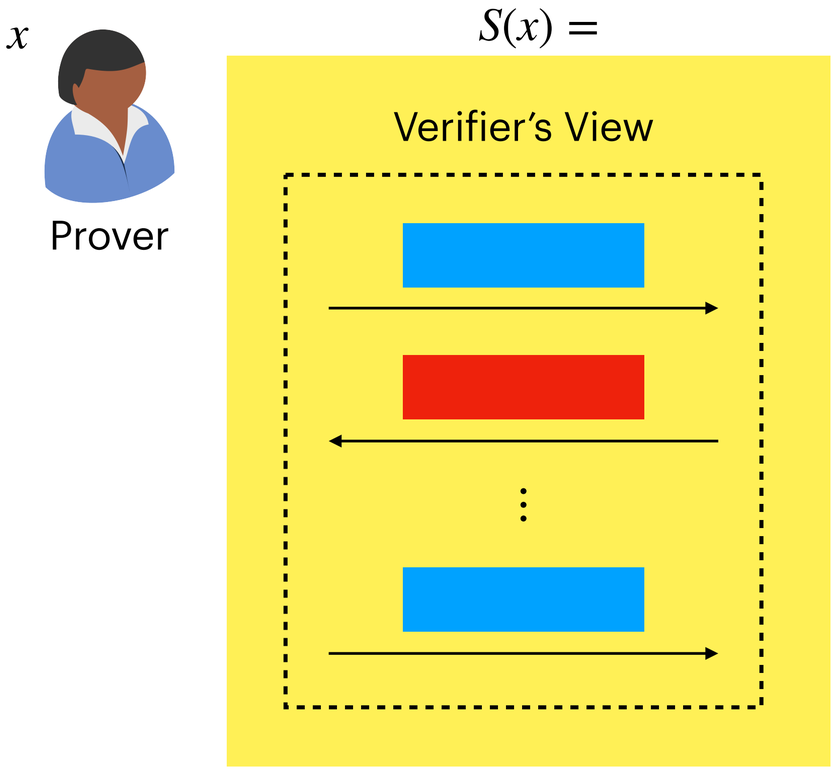
The Simulation Paradigm

- Paradox?
  - Protocol execution convinces the verifier of the validity of x.
  - The verifier could have generated the transcript by itself.



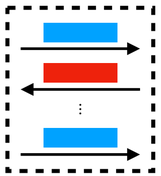
The Simulation Paradigm

- Paradox?
  - Protocol execution convinces the verifier of the validity of x.
  - The verifier could have generated the transcript by itself.



"x is true."

"See this accepting transcript."

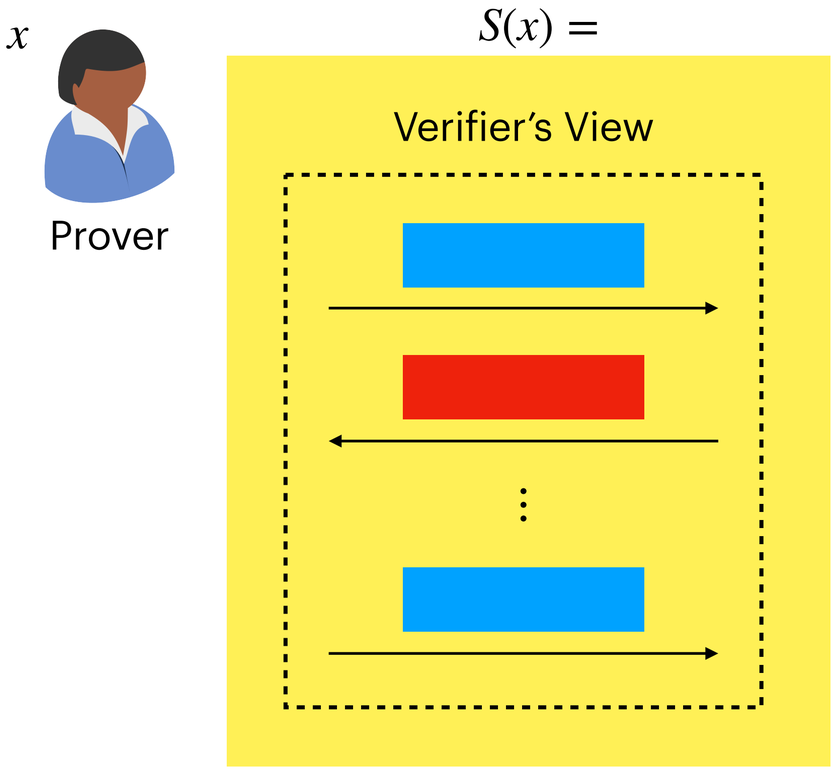


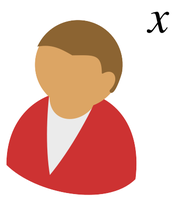
"That's not convincing. Anyone can come up with such a transcript."

Verifier participates in a live conversation.

The Simulation Paradigm

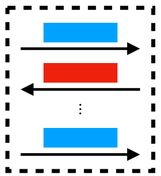
- Paradox?
  - Protocol execution convinces the verifier of the validity of x.
  - The verifier could have generated the transcript by itself.





"x is true."

"See this accepting transcript."



"That's not convincing. Anyone can come up with such a transcript."

Verifier participates in a live conversation.

Eve only sees the final transcript.

The Simulation Paradigm

- Paradox?
  - Protocol execution convinces the verifier of the validity of x.
  - The verifier could have generated the transcript by itself.

The Simulation Paradigm

- Paradox?
  - Protocol execution convinces the verifier of the validity of x.
  - The verifier could have generated the transcript by itself.

- **Takeaway**
  - The prover must respond to the verifier's challenges on the fly. This is what convinces the verifier of the statement's validity.
  - The simulator outputs the verifier's view “all at once” i.e., it can be fabricated. This ensures the view carries no additional knowledge, but it is not a proof of the statement's validity.

The Simulation Paradigm

- Paradox?
  - Protocol execution convinces the verifier of the validity of x.
  - The verifier could have generated the transcript by itself.

- **Takeaway**
  - The prover must respond to the verifier's challenges on the fly. This is what convinces the verifier of the statement's validity.
  - The simulator outputs the verifier's view “all at once” i.e., it can be fabricated. This ensures the view carries no additional knowledge, but it is not a proof of the statement's validity.
- **Analogy:** Think about a viral video.
  - If you witness the incident live, you are convinced it really happened.
  - If you only see the video, you remain skeptical; it could have been fabricated.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

- $\text{View}_{\widehat{V}}$ consists of the joint distribution of the random tape of $\widehat{V}$ and the protocol transcript.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

- $\text{View}_{\widehat{V}}$ consists of the joint distribution of the random tape of $\widehat{V}$ and the protocol transcript.
  - Behavior of $\widehat{V}$ is deterministic given $\text{View}_{\widehat{V}}$.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

- Zero-knowledge is a property of the prescribed prover P.
  - Required to hold only for $x \in L$.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

- Zero-knowledge is a property of the prescribed prover P.
  - Required to hold only for $x \in L$.
- Note the order of quantifiers
  - A possibly different S exists for each verifier $\widehat{V}$.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

- Zero-knowledge is a property of the prescribed prover P.
  - Required to hold only for $x \in L$.
- Note the order of quantifiers
  - A possibly different S exists for each verifier $\widehat{V}$.
- **Trivial Case:** Every language in BPP has a zero-knowledge proof.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

- Zero-knowledge is a property of the prescribed prover P.
  - Required to hold only for $x \in L$.
- Note the order of quantifiers
  - A possibly different S exists for each verifier $\widehat{V}$.
- **Trivial Case:** Every language in BPP has a zero-knowledge proof.
  - An efficient verifier can always check if $x \in L$ itself.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

- Zero-knowledge is a property of the prescribed prover P.
  - Required to hold only for $x \in L$.
- Note the order of quantifiers
  - A possibly different S exists for each verifier $\widehat{V}$.
- **Trivial Case:** Every language in BPP has a zero-knowledge proof.
  - An efficient verifier can always check if $x \in L$ itself.

What if the verifier already possesses additional information?

E.g., when the zero-knowledge proof is a part of a larger protocol.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

- Zero-knowledge is a property of the prescribed prover P.
  - Required to hold only for $x \in L$.
- Note the order of quantifiers
  - A possibly different S exists for each verifier $\widehat{V}$.
- **Trivial Case:** Every language in BPP has a zero-knowledge proof.
  - An efficient verifier can always check if $x \in L$ itself.

What if the verifier already possesses additional information?

E.g., when the zero-knowledge proof is a part of a larger protocol.

The simulation must be consistent with this additional information.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x)]\} \overset{c}{\approx} \{S(x)\}.$$

- Zero-knowledge is a property of the prescribed prover P.
  - Required to hold only for $x \in L$.
- Note the order of quantifiers
  - A possibly different S exists for each verifier $\widehat{V}$.
- **Trivial Case:** Every language in BPP has a zero-knowledge proof.
  - An efficient verifier can always check if $x \in L$ itself.

What if the verifier already possesses additional information?

E.g., when the zero-knowledge proof is a part of a larger protocol.

The simulation must be consistent with this additional information.

This was captured in our earlier definitions by considering a non-uniform adversary.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$ and $z \in \{0,1\}^*$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x, z)]\} \overset{c}{\approx} \{S(x, z)\}.$$

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$ and $z \in \{0,1\}^*$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x, z)]\} \overset{c}{\approx} \{S(x, z)\}.$$

- The auxiliary input z captures any prior information $\widehat{V}$ may have about x.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$ and $z \in \{0,1\}^*$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x, z)]\} \overset{c}{\approx} \{S(x, z)\}.$$

- The auxiliary input z captures any prior information $\widehat{V}$ may have about x.
  - The distinguisher trying to distinguish between $\text{View}_{\widehat{V}}$ and $S(x, z)$ also gets z.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$ and $z \in \{0,1\}^*$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x, z)]\} \overset{c}{\approx} \{S(x, z)\}.$$

- The auxiliary input z captures any prior information $\widehat{V}$ may have about x.
  - The distinguisher trying to distinguish between $\text{View}_{\widehat{V}}$ and $S(x, z)$ also gets z.
  - The definition promises that $\widehat{V}$ does not learn anything new.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$ and $z \in \{0,1\}^*$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x, z)]\} \overset{c}{\approx} \{S(x, z)\}.$$

- The auxiliary input z captures any prior information $\widehat{V}$ may have about x.
  - The distinguisher trying to distinguish between $\text{View}_{\widehat{V}}$ and $S(x, z)$ also gets z.
  - The definition promises that $\widehat{V}$ does not learn anything new.
- Why can't we simply define $\widehat{V}$ to be *non-uniform*?

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$ and $z \in \{0,1\}^*$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x, z)]\} \overset{c}{\approx} \{S(x, z)\}.$$

- The auxiliary input z captures any prior information $\widehat{V}$ may have about x.
  - The distinguisher trying to distinguish between $\text{View}_{\widehat{V}}$ and $S(x, z)$ also gets z.
  - The definition promises that $\widehat{V}$ does not learn anything new.
- Why can't we simply define $\widehat{V}$ to be *non-uniform*?
  - S depends on $\widehat{V}$ → this leads to some issues when using the definition. We will not discuss them in this course.

Zero-Knowledge Proof

Zero-Knowledge Proof System

A proof system (P, V) for a language $L \subset \{0,1\}^*$ is zero-knowledge if for every PPT algorithm $\widehat{V}$, there exists a PPT simulator S such that for every $x \in L$ and $z \in \{0,1\}^*$

$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x, z)]\} \overset{c}{\approx} \{S(x, z)\}.$$

- The auxiliary input z captures any prior information $\widehat{V}$ may have about x.
  - The distinguisher trying to distinguish between $\text{View}_{\widehat{V}}$ and $S(x, z)$ also gets z.
  - The definition promises that $\widehat{V}$ does not learn anything new.
- Why can't we simply define $\widehat{V}$ to be *non-uniform*?
  - S depends on $\widehat{V}$ → this leads to some issues when using the definition. We will not discuss them in this course.
- If the distributions are statistically close we call it statistical zero-knowledge.