Zero Knowledge Proof through Graph 3-Coloring

DIVYANSH AGARWAL

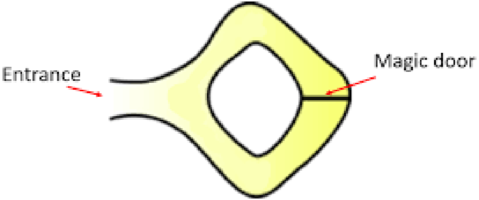
CS747: Randomized Methods in Computational Complexity

Indian Institute of Technology, Kanpur

Introduction to Zero Knowledge

Introduction to Zero Knowledge

Introduction to Zero Knowledge



Interactive Proof System

A pair of interactive machines (P, V) is called an **interactive proof system** for a language L if V is PPT TM and the following two conditions hold:

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

$$\Pr[\langle P, V\rangle(x) = 1] \geq \frac{2}{3}$$

- **Soundness:** For every $x \notin L$ and every interactive machine B,

$$\Pr[\langle B, V\rangle(x) = 1] \leq \frac{1}{3}$$

Perfect Zero-Knowledge

Let (P, V) be an interactive proof system for some language L. We say that (P, V) is **perfect zero-knowledge** if

Perfect Zero-Knowledge

Let (P, V) be an interactive proof system for some language L. We say that (P, V) is **perfect zero-knowledge** if **for every PPT TM**

Perfect Zero-Knowledge

Let (P, V) be an interactive proof system for some language L. We say that (P, V) is **perfect zero-knowledge** if **for every PPT TM** V^*, there exists a PPT algorithm M^* such that for every $x \in L$, the following hold:

1. With probability at most $\frac{1}{2}$, on input x, machine M^* outputs a special symbol $\perp$, i.e.,

$$\Pr[M^*(x) = \perp] \leq \frac{1}{2}.$$

Perfect Zero-Knowledge

Let (P, V) be an interactive proof system for some language L. We say that (P, V) is **perfect zero-knowledge** if **for every PPT TM** V^*, there exists a PPT algorithm M^* such that for every $x \in L$, the following hold:

1. With probability at most $\frac{1}{2}$, on input x, machine M^* outputs a special symbol $\perp$, i.e.,
$$\Pr[M^*(x) = \perp] \leq \frac{1}{2}.$$
2. $m^*(x) \stackrel{d}{=} \langle P, V^* \rangle(x)$ (output of V^* after interacting with P on x)

Perfect Zero-Knowledge

Let (P, V) be an interactive proof system for some language L. We say that (P, V) is **perfect zero-knowledge** if **for every PPT TM** V^*, there exists a PPT algorithm M^* such that for every $x \in L$, the following hold:

1. With probability at most $\frac{1}{2}$, on input x, machine M^* outputs a special symbol $\perp$, i.e.,

$$\Pr[M^*(x) = \perp] \leq \frac{1}{2}.$$

2. $m^*(x) \stackrel{d}{=} \langle P, V^* \rangle(x)$ (output of V^* after interacting with P on x)

where, $m^*(x)$ denote the random variable describing the distribution of $M^*(x)$ conditioned on $M^*(x) \neq \perp$, i.e.,

$$\Pr[m^*(x) = \alpha] = \Pr[M^*(x) = \alpha \mid M^*(x) \neq \perp], \quad \forall \alpha \in \{0, 1\}^*.$$

Computational Zero-Knowledge

Let (P, V), L, and V^* be as above.

Computational Zero-Knowledge

Let (P, V), L, and V^* be as above.
$\text{view}_{V^*}^{P}(x) :=$ the random variable describing the content of the random tape of V^*

Computational Zero-Knowledge

Let (P, V), L, and V^* be as above.
$\text{view}_{V^*}^{P}(x) :=$ the random variable describing the content of the random tape of V^* and the messages V^* receives from P during their interaction on common input x.

We say that (P, V) is **zero-knowledge** if for every PPT TM V^*, there exists PPT algorithm M^* such that the ensembles

$$\{\text{view}_{V^*}^{P}(x)\}_{x \in L} \quad \text{and} \quad \{M^*(x)\}_{x \in L}$$

are **computationally indistinguishable.**

Computational Indistinguishable and Complexity Class Relation

We consider ensembles indexed by strings from a language L. We say that the ensembles $\{R_x\}_{x \in L}$ and $\{S_x\}_{x \in L}$ are **computationally indistinguishable** if for every probabilistic polynomial-time algorithm D, for every polynomial $p(\cdot)$, and for all sufficiently long $x \in L$, it holds that

$$|\Pr[D(x, R_x) = 1] - \Pr[D(x, S_x) = 1]| < \frac{1}{p(|x|)}.$$

Computational Indistinguishable and Complexity Class Relation

We consider ensembles indexed by strings from a language L. We say that the ensembles $\{R_x\}_{x \in L}$ and $\{S_x\}_{x \in L}$ are **computationally indistinguishable** if for every probabilistic polynomial-time algorithm D, for every polynomial $p(\cdot)$, and for all sufficiently long $x \in L$, it holds that

$$|\Pr[D(x, R_x) = 1] - \Pr[D(x, S_x) = 1]| < \frac{1}{p(|x|)}.$$

Complexity Class Relations:

$$\text{BPP} \subseteq \text{PZK} \subseteq \text{CZK} \subseteq \text{IP}$$

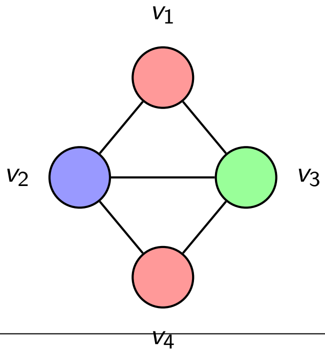
Graph 3-Coloring (G3C)

The language **Graph 3-Coloring**, denoted $G3C$, consists of all simple (finite) undirected graphs that can be vertex-colored using three colors such that no two adjacent vertices are given the same color.

Formally, a graph $G = (V, E)$ is **3-colorable** if there exists a mapping

$$\phi : V \to \{1, 2, 3\}$$

such that $\phi(u) \neq \phi(v)$ for every $(u, v) \in E$.



Zero-Knowledge Proof for Graph 3-Coloring: Intuition

- To prove that a graph $G = (V, E)$ is 3-colorable, the prover generates a random 3-coloring ϕ.

Zero-Knowledge Proof for Graph 3-Coloring: Intuition

- To prove that a graph $G = (V, E)$ is 3-colorable, the prover generates a random 3-coloring ϕ.
- Each vertex color represents a small piece of information about the global coloring.

Zero-Knowledge Proof for Graph 3-Coloring: Intuition

- To prove that a graph $G = (V, E)$ is 3-colorable, the prover generates a random 3-coloring ϕ.
- Each vertex color represents a small piece of information about the global coloring.
- The verifier will only inspect the colors of a few edges to confirm correctness.

Zero-Knowledge Proof for Graph 3-Coloring: Intuition

- To prove that a graph $G = (V, E)$ is 3-colorable, the prover generates a random 3-coloring ϕ.
- Each vertex color represents a small piece of information about the global coloring.
- The verifier will only inspect the colors of a few edges to confirm correctness.
- Each inspected edge $(u, v) \in E$ yields a **template** $(\phi(u), \phi(v))$.
  - Each individual template reveals no information.
  - If all templates are correct (distinct colors), the graph must be 3-colorable.

Zero-Knowledge Proof for Graph 3-Coloring: Intuition

- To prove that a graph $G = (V, E)$ is 3-colorable, the prover generates a random 3-coloring ϕ.
- Each vertex color represents a small piece of information about the global coloring.
- The verifier will only inspect the colors of a few edges to confirm correctness.
- Each inspected edge $(u, v) \in E$ yields a **template** $(\phi(u), \phi(v))$.
  - Each individual template reveals no information.
  - If all templates are correct (distinct colors), the graph must be 3-colorable.
- Graphs that are not 3-colorable must contain at least one "bad" edge (same color), so they are rejected with noticeable probability.

Zero-Knowledge Proof for Graph 3-Coloring: Intuition

- To prove that a graph $G = (V, E)$ is 3-colorable, the prover generates a random 3-coloring ϕ.
- Each vertex color represents a small piece of information about the global coloring.
- The verifier will only inspect the colors of a few edges to confirm correctness.
- Each inspected edge $(u, v) \in E$ yields a **template** $(\phi(u), \phi(v))$.
  - Each individual template reveals no information.
  - If all templates are correct (distinct colors), the graph must be 3-colorable.
- Graphs that are not 3-colorable must contain at least one "bad" edge (same color), so they are rejected with noticeable probability.
- The zero-knowledge property holds intuitively because a simulator can mimic the interaction by picking random distinct colors for the verifier's chosen edge.

Properties:

- If G is 3-colorable, verifier always accepts.
- If not, verifier rejects with probability at least $1/|E|$.
- Repeating the protocol increases confidence.

However, for implementation, we don't have perfect boxes, hence we use commitment schemes.

Commitment Schemes: Two-Phase Protocol

Commitment Schemes: Two-Phase Protocol

Phase 1 — Commit (Hiding Phase)

- The sender (prover) chooses a secret value v and some random coins r.
- It computes a **commitment** $C = \text{Commit}(v; r)$ and sends C to the receiver (verifier).
- The receiver learns nothing about v (secrecy ensured by randomness).

Commitment Schemes: Two-Phase Protocol

Phase 1 — Commit (Hiding Phase)

- The sender (prover) chooses a secret value v and some random coins r.
- It computes a **commitment** $C = \text{Commit}(v; r)$ and sends C to the receiver (verifier).
- The receiver learns nothing about v (secrecy ensured by randomness).

Phase 2 — Reveal (Binding Check)

- The sender reveals v and r.
- The receiver recomputes $\text{Commit}(v; r)$ and checks it matches C.
- If it matches $\rightarrow$ accept. Otherwise $\rightarrow$ reject.

Commitment Schemes: Two-Phase Protocol

Phase 1 — Commit (Hiding Phase)

- The sender (prover) chooses a secret value v and some random coins r.
- It computes a **commitment** $C = \text{Commit}(v; r)$ and sends C to the receiver (verifier).
- The receiver learns nothing about v (secrecy ensured by randomness).

Phase 2 — Reveal (Binding Check)

- The sender reveals v and r.
- The receiver recomputes $\text{Commit}(v; r)$ and checks it matches C.
- If it matches $\rightarrow$ accept. Otherwise $\rightarrow$ reject.

$$\textbf{Prover} \quad \begin{array}{c} \xrightarrow{\quad\text{Commit } C\quad} \\ \xrightarrow{\quad\text{Reveal } (v, r)\quad} \end{array} \quad \textbf{Verifier}$$

Commitment Scheme

A **commitment scheme** is a two-phase protocol between:

- **Sender (S):** commits to a value $v \in \{0, 1\}$ using randomness r.
- **Receiver (R):** obtains commitment C and later verifies v.

Requirements:

- **Hiding:** For any receiver R^*, the distributions $\{\langle S(0), R^* \rangle(1^n)\}$ and $\{\langle S(1), R^* \rangle(1^n)\}$ are computationally indistinguishable.

Commitment Scheme

A **commitment scheme** is a two-phase protocol between:

- **Sender (S):** commits to a value $v \in \{0, 1\}$ using randomness r.
- **Receiver (R):** obtains commitment C and later verifies v.

Requirements:

- **Hiding:** For any receiver R^*, the distributions $\{\langle S(0), R^* \rangle(1^n)\}$ and $\{\langle S(1), R^* \rangle(1^n)\}$ are computationally indistinguishable.
- **Binding:** For almost all random coins r of R, there exists at most one message m forming a valid opening.

Zero-Knowledge Proof for Graph 3-Coloring

Common input: A simple 3-colorable graph $G = (V, E)$ with $n = |V|$.
Auxiliary input to the prover: A valid 3-coloring ψ of G.

Zero-Knowledge Proof for Graph 3-Coloring

Common input: A simple 3-colorable graph $G = (V, E)$ with $n = |V|$.
Auxiliary input to the prover: A valid 3-coloring ψ of G.
Protocol Steps:
P1: The prover picks a random permutation π over $\{1, 2, 3\}$ and defines $\phi(v) = \pi(\psi(v))$ for all $v \in V$.

Zero-Knowledge Proof for Graph 3-Coloring

Common input: A simple 3-colorable graph $G = (V, E)$ with $n = |V|$.
Auxiliary input to the prover: A valid 3-coloring ψ of G.
Protocol Steps:

P1: The prover picks a random permutation π over $\{1, 2, 3\}$ and defines $\phi(v) = \pi(\psi(v))$ for all $v \in V$.

- Uses the **commitment scheme** to commit to each color:

$$s_1, \dots, s_n \leftarrow \{0,1\}^n, \quad c_i = C_{s_i}(\phi(i))$$

- Sends $c_1, \dots, c_n$ to the verifier.

Zero-Knowledge Proof for Graph 3-Coloring

Common input: A simple 3-colorable graph $G = (V, E)$ with $n = |V|$.

Auxiliary input to the prover: A valid 3-coloring ψ of G.

Protocol Steps:

P1: The prover picks a random permutation π over $\{1, 2, 3\}$ and defines $\phi(v) = \pi(\psi(v))$ for all $v \in V$.

- Uses the **commitment scheme** to commit to each color:

$$s_1, \ldots, s_n \leftarrow \{0,1\}^n, \quad c_i = C_{s_i}(\phi(i))$$

- Sends $c_1, \ldots, c_n$ to the verifier.

V1: The verifier chooses a random edge $(u, v) \in E$ and sends it to the prover.

Zero-Knowledge Proof for Graph 3-Coloring

Common input: A simple 3-colorable graph $G = (V, E)$ with $n = |V|$.

Auxiliary input to the prover: A valid 3-coloring ψ of G.

Protocol Steps:

P1: The prover picks a random permutation π over $\{1, 2, 3\}$ and defines $\phi(v) = \pi(\psi(v))$ for all $v \in V$.

- Uses the **commitment scheme** to commit to each color:

$$s_1, \ldots, s_n \leftarrow \{0, 1\}^n, \quad c_i = C_{s_i}(\phi(i))$$

- Sends $c_1, \ldots, c_n$ to the verifier.

V1: The verifier chooses a random edge $(u, v) \in E$ and sends it to the prover.

P2: The prover reveals colors of u and v by sending $(s_u, \phi(u))$, $(s_v, \phi(v))$.

Zero-Knowledge Proof for Graph 3-Coloring

Common input: A simple 3-colorable graph $G = (V, E)$ with $n = |V|$.
Auxiliary input to the prover: A valid 3-coloring ψ of G.
Protocol Steps:

P1: The prover picks a random permutation π over $\{1, 2, 3\}$ and defines $\phi(v) = \pi(\psi(v))$ for all $v \in V$.

- Uses the **commitment scheme** to commit to each color:

$$s_1, \ldots, s_n \leftarrow \{0, 1\}^n, \quad c_i = C_{s_i}(\phi(i))$$

- Sends $c_1, \ldots, c_n$ to the verifier.

V1: The verifier chooses a random edge $(u, v) \in E$ and sends it to the prover.

P2: The prover reveals colors of u and v by sending $(s_u, \phi(u))$, $(s_v, \phi(v))$.

V2: The verifier checks:

$$c_u = C_{s_u}(\phi(u)), \quad c_v = C_{s_v}(\phi(v)), \quad \phi(u) \neq \phi(v)$$

Accept if all hold; otherwise reject.

Completeness: If G is 3-colorable, verifier always accepts.
Soundness: If not, reject with probability $\geq 1/|E|$.

Completeness: If G is 3-colorable, verifier always accepts.
Soundness: If not, reject with probability $\geq 1/|E|$.

The next slides will prove computational zero knowledge for above protocol.

Simulator M^* for the Graph 3-Coloring ZKP

M^* incorporates the code of interactive program V^*.

Simulator M^* for the Graph 3-Coloring ZKP

M^* incorporates the code of interactive program V^*. **Simulator M^* (on input graph $G = (V, E)$, $n = |V|, G \in G3C$):**

1. **Set verifier randomness:** Pick random tape $r \in \{0,1\}^{q(n)}$, where $q(\cdot)$ bounds the runtime of V^*.

Simulator M^* for the Graph 3-Coloring ZKP

M^* incorporates the code of interactive program V^*. **Simulator M^* (on input graph $G = (V, E)$, $n = |V|, G \in G3C$):**

1. **Set verifier randomness:** Pick random tape $r \in \{0,1\}^{q(n)}$, where $q(\cdot)$ bounds the runtime of V^*.
2. **Simulate commitments:** Choose random colors $e_1, \ldots, e_n \in \{1,2,3\}$ and random strings $s_1, \ldots, s_n \in \{0,1\}^n$. Compute $c_i' = C_{s_i}(e_i)$ for each $i \in V$.

Simulator M^* for the Graph 3-Coloring ZKP

M^* incorporates the code of interactive program V^*. **Simulator M^* (on input graph $G = (V, E)$, $n = |V|, G \in G3C$):**

1. **Set verifier randomness:** Pick random tape $r \in \{0,1\}^{q(n)}$, where $q(\cdot)$ bounds the runtime of V^*.
2. **Simulate commitments:** Choose random colors $e_1, \dots, e_n \in \{1,2,3\}$ and random strings $s_1, \dots, s_n \in \{0,1\}^n$. Compute $c'_i = C_{s_i}(e_i)$ for each $i \in V$.
3. **Run the verifier V^*:** Give G as input, r as random tape, and $(c'_1, \dots, c'_n)$ as incoming messages. Let its first outgoing message be $m = (u, v) \in E$.

Simulator M^* for the Graph 3-Coloring ZKP

M^* incorporates the code of interactive program V^*. **Simulator M^* (on input graph $G = (V, E)$, $n = |V|, G \in G3C$):**

1. **Set verifier randomness:** Pick random tape $r \in \{0,1\}^{q(n)}$, where $q(\cdot)$ bounds the runtime of V^*.
2. **Simulate commitments:** Choose random colors $e_1, \ldots, e_n \in \{1,2,3\}$ and random strings $s_1, \ldots, s_n \in \{0,1\}^n$. Compute $c'_i = C_{s_i}(e_i)$ for each $i \in V$.
3. **Run the verifier V^*:** Give G as input, r as random tape, and $(c'_1, \ldots, c'_n)$ as incoming messages. Let its first outgoing message be $m = (u, v) \in E$.
4. **Reveal simulated colors:**
   - If $e_u \neq e_v$, output simulated transcript $(G, r, (c'_1, \ldots, c'_n), (s_u, e_u, s_v, e_v))$.
   - Else, output $\perp$ (simulation failure).

Simulator M^* for the Graph 3-Coloring ZKP

M^* incorporates the code of interactive program V^*. **Simulator M^* (on input graph $G = (V, E)$, $n = |V|, G \in G3C$):**

1. **Set verifier randomness:** Pick random tape $r \in \{0, 1\}^{q(n)}$, where $q(\cdot)$ bounds the runtime of V^*.
2. **Simulate commitments:** Choose random colors $e_1, \ldots, e_n \in \{1, 2, 3\}$ and random strings $s_1, \ldots, s_n \in \{0, 1\}^n$. Compute $c'_i = C_{s_i}(e_i)$ for each $i \in V$.
3. **Run the verifier V^*:** Give G as input, r as random tape, and $(c'_1, \ldots, c'_n)$ as incoming messages. Let its first outgoing message be $m = (u, v) \in E$.
4. **Reveal simulated colors:**
   - If $e_u \neq e_v$, output simulated transcript $(G, r, (c'_1, \ldots, c'_n), (s_u, e_u, s_v, e_v))$.
   - Else, output $\perp$ (simulation failure).

Why the Simulator M^* Works

- If the verifier's edge choice were **oblivious** (independent of commitments), then for random fake colors $e_i \in \{1, 2, 3\}$:

  $$\Pr[e_u \neq e_v] = \frac{2}{3}.$$

  Hence, M^* would succeed (not output $\perp$) with probability $\frac{2}{3}$.

Why the Simulator M^* Works

- If the verifier's edge choice were **oblivious** (independent of commitments), then for random fake colors $e_i \in \{1, 2, 3\}$:

$$\Pr[e_u \neq e_v] = \frac{2}{3}.$$

  Hence, M^* would succeed (not output $\perp$) with probability $\frac{2}{3}$.
- In reality, the verifier's request may depend on the commitments, but due to the **hiding (secrecy)** of the commitment scheme, this dependence is only computationally negligible — i.e., the verifier is **almost oblivious**.
- $\Pr[M^*(G) = \perp] \approx \frac{1}{3} \pm \mathrm{negl}(n)$.

Why the Simulator M^* Works

- If the verifier's edge choice were **oblivious** (independent of commitments), then for random fake colors $e_i \in \{1, 2, 3\}$:

$$\Pr[e_u \neq e_v] = \frac{2}{3}.$$

  Hence, M^* would succeed (not output $\perp$) with probability $\frac{2}{3}$.
- In reality, the verifier's request may depend on the commitments, but due to the **hiding (secrecy)** of the commitment scheme, this dependence is only computationally negligible — i.e., the verifier is **almost oblivious**.
- $\Pr[M^*(G) = \perp] \approx \frac{1}{3} \pm \mathrm{negl}(n)$.
- **Claim 1:** Using request-obliviousness, simulator's failure probability $\approx \frac{1}{3}$.

Why the Simulator M^* Works

- If the verifier's edge choice were **oblivious** (independent of commitments), then for random fake colors $e_i \in \{1, 2, 3\}$:

$$\Pr[e_u \neq e_v] = \frac{2}{3}.$$

Hence, M^* would succeed (not output $\perp$) with probability $\frac{2}{3}$.

- In reality, the verifier's request may depend on the commitments, but due to the **hiding (secrecy)** of the commitment scheme, this dependence is only computationally negligible — i.e., the verifier is **almost oblivious**.
- $\Pr[M^*(G) = \perp] \approx \frac{1}{3} \pm \mathrm{negl}(n)$.
- **Claim 1:** Using request-obliviousness, simulator's failure probability $\approx \frac{1}{3}$.
- **Claim 2:** Conditioned on success, simulator's output is computationally indistinguishable from the real verifier's view (else we break **"nonuniform"** secrecy).

Conclusions

- The simulator M^* succeeds with constant probability ($\approx 2/3$) even without knowing the 3-coloring, by exploiting the hiding property of the commitment scheme.
- Hence, We achieve **computational zero-knowledge** for 3-Colorability: the NP complete problem.

References

- Oded Goldreich, *Foundations of Cryptography, Volume 1: Basic Tools*, Cambridge University Press, 2001.
- Oded Goldreich, Silvio Micali, and Avi Wigderson, *Proofs that Yield Nothing But Their Validity, or All Languages in NP Have Zero-Knowledge Proof Systems*, Journal of the ACM, 1986.