Zero-Knowledge Proofs III

601.442/642 Modern Cryptography

31st March 2026

Logistics

- Homework 7 is due **this Thursday** (2nd April).
- **Midterm next Tuesday** (7th April).
    - No homework this week!

Aside: ZKP for Disclosing Vulnerabilities

Aside: ZKP for Disclosing Vulnerabilities

Alice finds a catastrophic vulnerability e.g., she can break RSA.

Aside: ZKP for Disclosing Vulnerabilities

Alice finds a catastrophic vulnerability e.g., she can break RSA.

She cannot publicly disclose the attack – it can be used to attack existing systems.

Aside: ZKP for Disclosing Vulnerabilities

Alice finds a catastrophic vulnerability e.g., she can break RSA.

She cannot publicly disclose the attack – it can be used to attack existing systems.

But she cannot make a convincing argument without disclosing the attack.

Aside: ZKP for Disclosing Vulnerabilities

Alice finds a catastrophic vulnerability e.g., she can break RSA.

She cannot publicly disclose the attack – it can be used to attack existing systems.

But she cannot make a convincing argument without disclosing the attack.

Solution: She can use a ZKP!

Aside: ZKP for Disclosing Vulnerabilities

Alice finds a catastrophic vulnerability e.g., she can break RSA.

She cannot publicly disclose the attack – it can be used to attack existing systems.

But she cannot make a convincing argument without disclosing the attack.

Solution: She can use a ZKP!

Zero Knowledge Proof Statement 1 (Low-Qubit Variant) *We possess a quantum kickmix circuit* $C_{\textit{low-qubit}}$ *(uniquely committed to via its cryptographic hash) with resource counts of at most:*

- $2,700,000$ *non-Clifford gates (CCX + CCZ)*
- $1,175$ *logical qubits*
- $17,000,000$ *total operations*

that correctly computes point addition on the elliptic curve `secp256k1` *across all 9,024 pseudo-random inputs deterministically derived from the circuit's own hash.*

Circuit SHA-256 Hash:
`0xcc8f532ffea1583ceed3c9af75de3263ebaddd5fdf3cddfb3dea848b94d0396a`

Groth16 Proof Bytes:

`0x0e78f4db0000000000000000000000000000000000000000000000000000000000000008cd56e10c2fe24795cff1e1d1f40d3a`
`324528d315674da45d26afb376e867000000000000000000000000000000000000000000000000000000000215c7fe4fc59`
`7b861d82370ab556684ae36e98cf073e7f754f2788ad58721dbd012927516f316e7b4f3effb1dbd567732611cb0334f2d75e529c5e3`
`becd0629c17605c7ff87c6f23324328744454bdec0df425a4a63e3358c10079c85ef757412ae86ae1f85bf47ef6980852d6f65423be`
`2d90adb5b29896493324128b1cda0a0042f7138c850a1ca441210ba770a2eee39d56f6f90bf68b7a346e1658c6529715334621b6e1a`
`63b85875b8c8a610e0d885662879755803027dad57d97140afb2498bbb63215b236575f95b0019f2b9713bc810e1e044d47ab360e92`
`b899c46512fc97460609186bf1fe01c892a8015fb00e7fdea11b08f88c6adb79b1243518`

Verification Key:

`0x00ca4af6cb15dbd83ec3eaab3a0664023828d90a98e650d2d340712f5f3eb0d4`

Aside: ZKP for Disclosing Vulnerabilities

Alice finds a catastrophic vulnerability e.g., she can break RSA.

She cannot publicly disclose the attack – it can be used to attack existing systems.

But she cannot make a convincing argument without disclosing the attack.

Solution: She can use a ZKP!

Zero Knowledge Proof Statement 1 (Low-Qubit Variant) *We possess a quantum kickmix circuit* $C_{\textit{low-qubit}}$ *(uniquely committed to via its cryptographic hash) with resource counts of at most:*

- $2,700,000$ *non-Clifford gates (CCX + CCZ)*
- $1,175$ *logical qubits*
- $17,000,000$ *total operations*

that correctly computes point addition on the elliptic curve `secp256k1` *across all 9,024 pseudo-random inputs deterministically derived from the circuit's own hash.*

Circuit SHA-256 Hash:
`0xcc8f532ffea1583ceed3c9af75de3263ebaddd5fdf3cddfb3dea848b94d0396a`

Groth16 Proof Bytes:

`0x0e78f4db0000000000000000000000000000000000000000000000000000000000000008cd56e10c2fe24795cff1e1d1f40d3a`
`324528d315674da45d26afb376e867000000000000000000000000000000000000000000000000000000000000215c7fe4fc59`
`7b861d82370ab556684ae36e98cf073e7f754f2788ad58721dbd012927516f316e7b4f3effb1dbd567732611cb0334f2d75e529c5e3`
`becd0629c17605c7ff87c6f23324328744454bdec0df425a4a63e3358c10079c85ef757412ae86ae1f85bf47ef6980852d6f65423be`
`2d90adb5b29896493324128b1cda0a0042f7138c850a1ca441210ba770a2eee39d56f6f90bf68b7a346e1658c6529715334621b6e1a`
`63b85875b8c8a610e0d885662879755803027dad57d97140afb2498bbb63215b236575f95b0019f2b9713bc810e1e044d47ab360e92`
`b899c46512fc97460609186bf1fe01c892a8015fb00e7fdea11b08f88c6adb79b1243518`

Verification Key:

`0x00ca4af6cb15dbd83ec3eaab3a0664023828d90a98e650d2d340712f5f3eb0d4`

Groth16 is a Non-interactive ZKP (NIZK).

NIZK: Given a public proving key, the prover can compute a single message π that proves validity of the statement. π can be verified by anyone and ensures zero-knowledge.

Recap: 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 $z \in \{0,1\}^*$

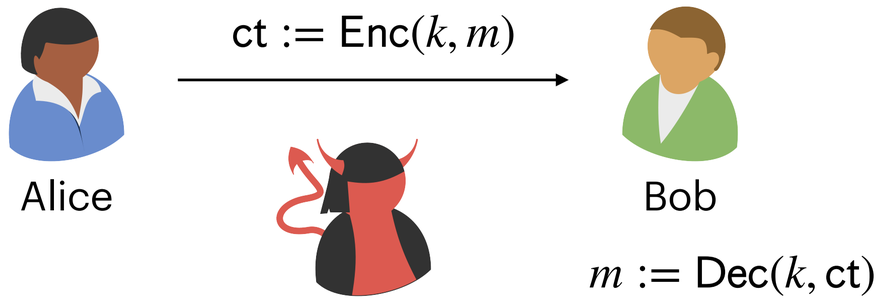
$$\{\text{View}_{\widehat{V}}[P(x) \leftrightarrow \widehat{V}(x,z)]\} \overset{c}{\approx} \{S(x,z)\}.$$

Applying the Simulation Paradigm for Encryption

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

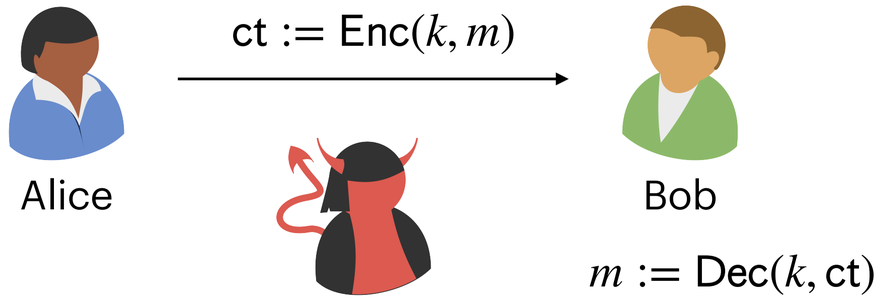
Applying the Simulation Paradigm for Encryption

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



Applying the Simulation Paradigm for Encryption

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



Semantic Security*

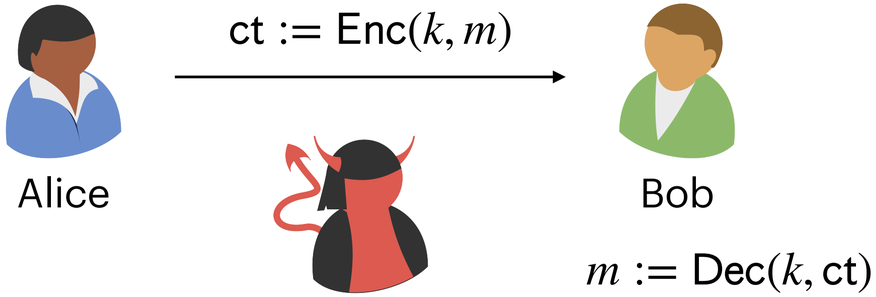
An encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with message length $\ell := \ell(\lambda)$ is semantically secure if there exists a PPT simulator S such that $\forall m \in \{0,1\}^{\ell}$ and all $z \in \{0,1\}^*$

$$\{\mathsf{Enc}(\mathsf{KeyGen}(1^\lambda), m), z\} \quad \overset{c}{\approx} \quad \{\mathsf{S}(1^\lambda, z)\}\,.$$

*This is an overly simplified version.
The full definition is more complex.

Applying the Simulation Paradigm for Encryption

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



Semantic Security*

An encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with message length $\ell := \ell(\lambda)$ is semantically secure if there exists a PPT simulator S such that $\forall m \in \{0,1\}^{\leq \ell}$ and all $z \in \{0,1\}^*$

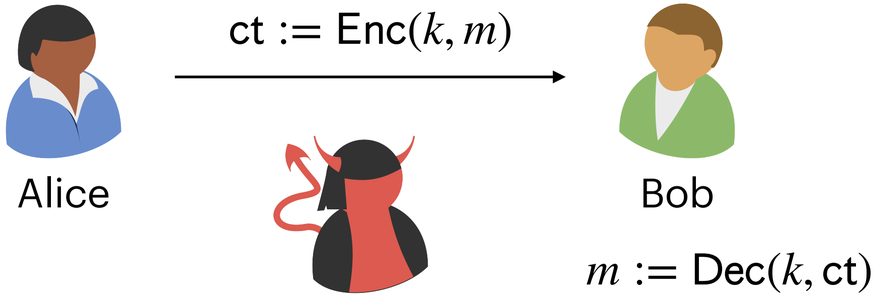
$$\{\mathsf{Enc}(\mathsf{KeyGen}(1^\lambda), m), z\} \quad \overset{c}{\approx} \quad \{\mathsf{S}(1^\lambda, 1^{|m|}, z)\} .$$

*This is an overly simplified version.
The full definition is more complex.

The simulator captures whatever is leaked to the adversary.

Applying the Simulation Paradigm for Encryption

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



Semantic Security*

An encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with message length $\ell := \ell(\lambda)$ is semantically secure if there exists a PPT simulator S such that $\forall m \in \{0,1\}^{\leq \ell}$ and all $z \in \{0,1\}^*$

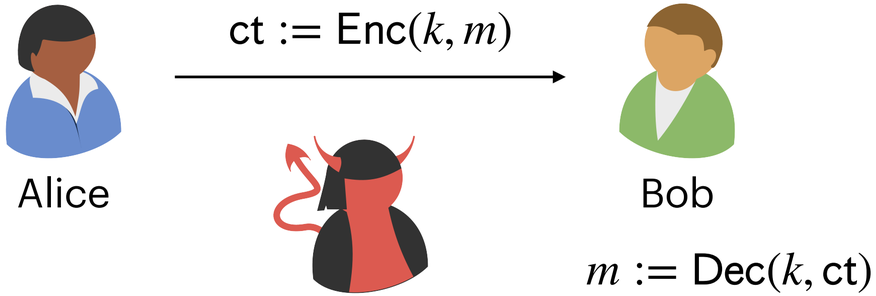
$$\{\mathsf{Enc}(\mathsf{KeyGen}(1^\lambda), m), z\} \quad \overset{c}{\approx} \quad \{\mathsf{S}(1^\lambda, 1^{|m|}, z)\} .$$

*This is an overly simplified version.
The full definition is more complex.

Semantic security is equivalent to one-time computational security.

Applying the Simulation Paradigm for Encryption

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



Semantic Security*

An encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with message length $\ell := \ell(\lambda)$ is semantically secure if there exists a PPT simulator S such that $\forall m \in \{0,1\}^{\leq \ell}$ and all $z \in \{0,1\}^*$

$$\{\mathsf{Enc}(\mathsf{KeyGen}(1^\lambda), m), z\} \quad \overset{c}{\approx} \quad \{\mathsf{S}(1^\lambda, 1^{|m|}, z)\} .$$

*This is an overly simplified version.
The full definition is more complex.

Semantic security is equivalent to one-time computational security.

Indistinguishability-based definitions are easier to work with for encryption schemes.

Zero-Knowledge Proofs for NP

Theorem: If one-way functions exist, then every language $L \in \mathsf{NP}$ has a zero-knowledge proof.

Zero-Knowledge Proofs for NP

Theorem: If one-way permutations exist, then every language $L \in \mathsf{NP}$ has a zero-knowledge proof.

Zero-Knowledge Proofs for NP

Theorem: If one-way permutations exist, then every language $L \in \mathsf{NP}$ has a zero-knowledge proof.

- Should we construct a ZKP for every NP language?

Zero-Knowledge Proofs for NP

Theorem: If one-way permutations exist, then every language $L \in \mathsf{NP}$ has a zero-knowledge proof.

- Should we construct a ZKP for every NP language?
  - Impractical!

Zero-Knowledge Proofs for NP

Theorem: If one-way permutations exist, then every language $L \in \mathsf{NP}$ has a zero-knowledge proof.

- Should we construct a ZKP for every NP language?
  - Impractical!
- Instead, we will construct a ZKP for an NP-complete language.

Zero-Knowledge Proofs for NP

Theorem: If one-way permutations exist, then every language $L \in \mathsf{NP}$ has a zero-knowledge proof.

- Should we construct a ZKP for every NP language?
  - Impractical!
- Instead, we will construct a ZKP for an NP-complete language.
- To construct a ZKP for any NP language, we do the following.

Zero-Knowledge Proofs for NP

Theorem: If one-way permutations exist, then every language $L \in \mathsf{NP}$ has a zero-knowledge proof.

- Should we construct a ZKP for every NP language?
  - Impractical!
- Instead, we will construct a ZKP for an NP-complete language.
- To construct a ZKP for any NP language, we do the following.
  - Given an instance x, the prover and verifier reduce x into an instance x' of the NP-complete language.

Zero-Knowledge Proofs for NP

Theorem: If one-way permutations exist, then every language $L \in \mathsf{NP}$ has a zero-knowledge proof.

- Should we construct a ZKP for every NP language?
  - Impractical!
- Instead, we will construct a ZKP for an NP-complete language.
- To construct a ZKP for any NP language, we do the following.
  - Given an instance x, the prover and verifier reduce x into an instance x' of the NP-complete language.
  - The prover transforms its witness w for x into a witness w' for x'.

Zero-Knowledge Proofs for NP

Theorem: If one-way permutations exist, then every language $L \in \mathsf{NP}$ has a zero-knowledge proof.

- Should we construct a ZKP for every NP language?
  - Impractical!
- Instead, we will construct a ZKP for an NP-complete language.
- To construct a ZKP for any NP language, we do the following.
  - Given an instance x, the prover and verifier reduce x into an instance x' of the NP-complete language.
  - The prover transforms its witness w for x into a witness w' for x'.
  - Use the ZKP for the NP-complete language to prove validity of x'.

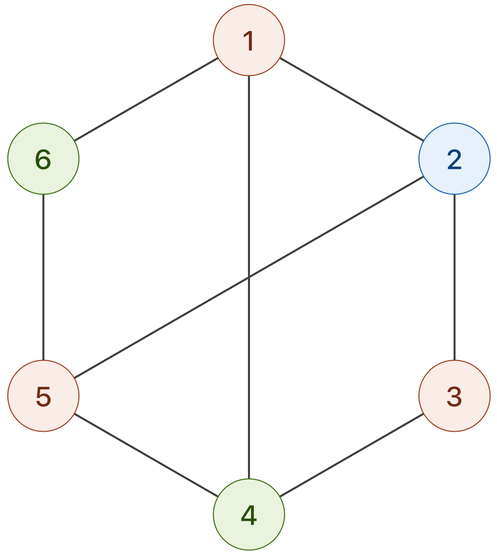
The Graph 3-Coloring Problem

- A graph G is 3-colorable if each vertex can be assigned one among three colors such that no two connected vertices have the same color.

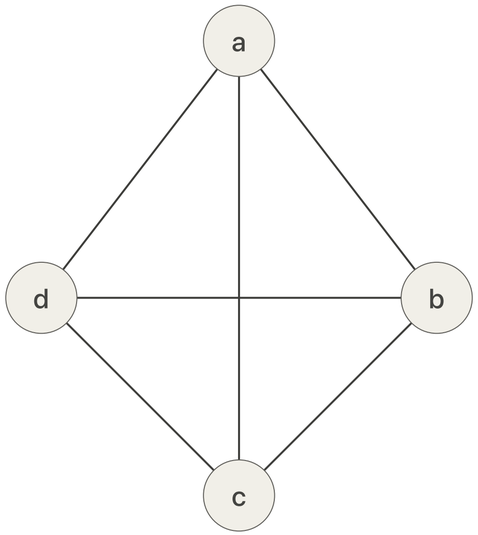
The Graph 3-Coloring Problem

- A graph G is 3-colorable if each vertex can be assigned one among three colors such that no two connected vertices have the same color.

3-colorable



Not 3-colorable



The Graph 3-Coloring Problem

- A graph G is 3-colorable if each vertex can be assigned one among three colors such that no two connected vertices have the same color.

- The language L consists of all graphs that are 3-colorable.

The Graph 3-Coloring Problem

- A graph G is 3-colorable if each vertex can be assigned one among three colors such that no two connected vertices have the same color.
- The language L consists of all graphs that are 3-colorable.
- Graph 3-coloring is in NP.

The Graph 3-Coloring Problem

- A graph G is 3-colorable if each vertex can be assigned one among three colors such that no two connected vertices have the same color.
- The language L consists of all graphs that are 3-colorable.
- Graph 3-coloring is in NP.
  - Given a statement $G = (V, E)$ in the language L, the witness is a valid assignment of colors denoted by the function $\phi : V \to \{1, 2, 3\}$.

The Graph 3-Coloring Problem

- A graph G is 3-colorable if each vertex can be assigned one among three colors such that no two connected vertices have the same color.
- The language L consists of all graphs that are 3-colorable.
- Graph 3-coloring is in NP.
  - Given a statement $G = (V, E)$ in the language L, the witness is a valid assignment of colors denoted by the function $\phi : V \rightarrow \{1, 2, 3\}$.
  - Statement can be verified in $O(|E|)$ time.

The Graph 3-Coloring Problem

- A graph G is 3-colorable if each vertex can be assigned one among three colors such that no two connected vertices have the same color.
- The language L consists of all graphs that are 3-colorable.
- Graph 3-coloring is in NP.
  - Given a statement $G = (V, E)$ in the language L, the witness is a valid assignment of colors denoted by the function $\phi : V \rightarrow \{1, 2, 3\}$.
  - Statement can be verified in $O(|E|)$ time.
- Graph 3-coloring is in fact NP-complete!

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Graph 3-coloring being NP-complete is not known to be in BPP.

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Graph 3-coloring being NP-complete is not known to be in BPP.

Learning ϕ would constitute knowledge for the verifier.

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Intuition: As in ZKP for graph-isomorphism, randomize the witness to hide it from the verifier.

Graph 3-coloring being NP-complete is not known to be in BPP.

Learning ϕ would constitute knowledge for the verifier.

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Intuition: As in ZKP for graph-isomorphism, randomize the witness to hide it from the verifier.

Can the prover send $\psi \circ \phi$ to the verifier?

Graph 3-coloring being NP-complete is not known to be in BPP.

Learning ϕ would constitute knowledge for the verifier.

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Intuition: As in ZKP for graph-isomorphism, randomize the witness to hide it from the verifier.

Graph 3-coloring being NP-complete is not known to be in BPP.

Learning ϕ would constitute knowledge for the verifier.

Can the prover send $\psi \circ \phi$ to the verifier?

No!

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Intuition: As in ZKP for graph-isomorphism, randomize the witness to hide it from the verifier.

Can the prover send $\psi \circ \phi$ to the verifier? No!

Graph 3-coloring being NP-complete is not known to be in BPP.

Learning ϕ would constitute knowledge for the verifier.

Any valid color assignment is new knowledge for the verifier.

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

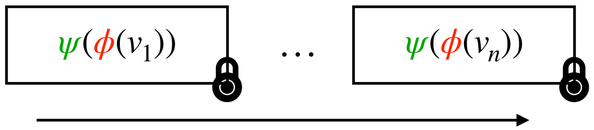
Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Store the color of the i-th vertex as per $\psi \circ \phi$
in the i-th lockbox.



Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

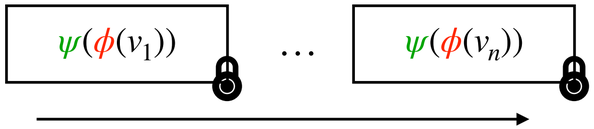
Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Store the color of the i-th vertex as per $\psi \circ \phi$ in the i-th lockbox.



Sample a random edge $(u, v) \xleftarrow{\$} E$.

(u, v)

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

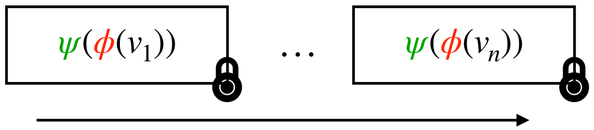
Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Store the color of the i-th vertex as per $\psi \circ \phi$ in the i-th lockbox.



Sample a random edge $(u, v) \xleftarrow{\$} E$.

(u, v)

Prove that the endpoints of this edge have different colors.

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

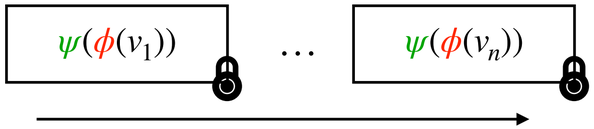
Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Store the color of the i-th vertex as per $\psi \circ \phi$ in the i-th lockbox.



Sample a random edge $(u, v) \xleftarrow{\$} E$.

(u, v)

Send the keys for box u and v.

Physical ZKP for Graph 3-Coloring

Statement $x : G = (V, E)$

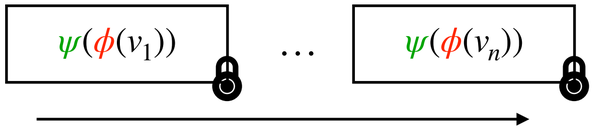
Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Store the color of the i-th vertex as per $\psi \circ \phi$ in the i-th lockbox.



Sample a random edge $(u, v) \xleftarrow{\$} E$.

(u, v)

Send the keys for box u and v.

Check if vertices u and v have different colors.

Physical ZKP for Graph 3-Coloring

Completeness?

Statement $x : G = (V, E)$

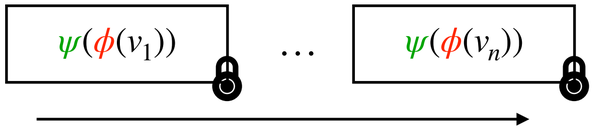
Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Store the color of the i-th vertex as per $\psi \circ \phi$ in the i-th lockbox.



Sample a random edge $(u, v) \xleftarrow{\$} E$.

(u, v)

Send the keys for box u and v.

Check if vertices u and v have different colors.

Physical ZKP for Graph 3-Coloring

Completeness?

Soundness?

Statement $x : G = (V, E)$

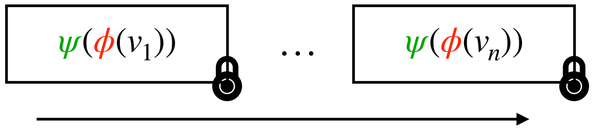
Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Store the color of the i-th vertex as per $\psi \circ \phi$ in the i-th lockbox.



Sample a random edge $(u, v) \xleftarrow{\$} E$.

(u, v)

Send the keys for box u and v.

Check if vertices u and v have different colors.

Physical ZKP for Graph 3-Coloring

Completeness?

Soundness?

Zero-Knowledge?

Statement $x : G = (V, E)$

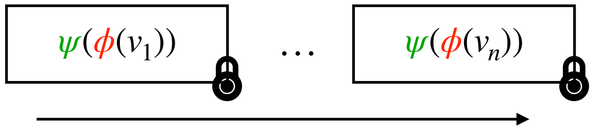
Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Store the color of the i-th vertex as per $\psi \circ \phi$ in the i-th lockbox.



Sample a random edge $(u, v) \xleftarrow{\$} E$.

(u, v)

Send the keys for box u and v.

Check if vertices u and v have different colors.

Physical ZKP for Graph 3-Coloring

Completeness?

Soundness?

Zero-Knowledge?

How do we realize lockboxes?

Statement $x : G = (V, E)$

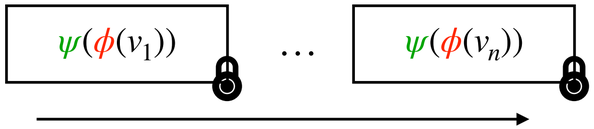
Prover

Verifier

Let $\phi : V \to \{1, 2, 3\}$ be a 3-coloring of G.

Sample $\psi \xleftarrow{\$} \mathsf{Perm}_3$.

Store the color of the i-th vertex as per $\psi \circ \phi$ in the i-th lockbox.



Sample a random edge $(u, v) \xleftarrow{\$} E$.

(u, v)

Send the keys for box u and v.

Check if vertices u and v have different colors.

Commitment Schemes

- Commitment Schemes are digital lockboxes.
  - **Commit phase:** Sender locks a value v inside a box.
  - **Open phase:** Sender unlocks the box and reveals v.

Commitment Schemes

- Commitment Schemes are digital lockboxes.
  - **Commit phase:** Sender locks a value v inside a box.
  - **Open phase:** Sender unlocks the box and reveals v.
- Can be implemented using interactive protocols, but we will focus on non-interactive commitments.
  - Each phase consists of a single message.

Commitment Schemes

- Commitment Schemes are digital lockboxes.
  - **Commit phase:** Sender locks a value v inside a box.
  - **Open phase:** Sender unlocks the box and reveals v.
- Can be implemented using interactive protocols, but we will focus on non-interactive commitments.
  - Each phase consists of a single message.
- We need two properties.

Commitment Schemes

- Commitment Schemes are digital lockboxes.
  - **Commit phase:** Sender locks a value v inside a box.
  - **Open phase:** Sender unlocks the box and reveals v.
- Can be implemented using interactive protocols, but we will focus on non-interactive commitments.
  - Each phase consists of a single message.
- We need two properties.
  - **Hiding:** Receiver cannot see the content inside a locked box.

Commitment Schemes

- Commitment Schemes are digital lockboxes.
  - **Commit phase:** Sender locks a value v inside a box.
  - **Open phase:** Sender unlocks the box and reveals v.
- Can be implemented using interactive protocols, but we will focus on non-interactive commitments.
  - Each phase consists of a single message.
- We need two properties.
  - **Hiding:** Receiver cannot see the content inside a locked box.
  - **Binding:** Sender cannot modify the content inside the box once it is locked.

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^{\ell}$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}.$$

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^\ell$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}.$$

r is randomness used for Com.

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^{\ell}$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}.$$

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^{\ell}$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}.$$

Hiding

Commitment Schemes

Commitment Scheme

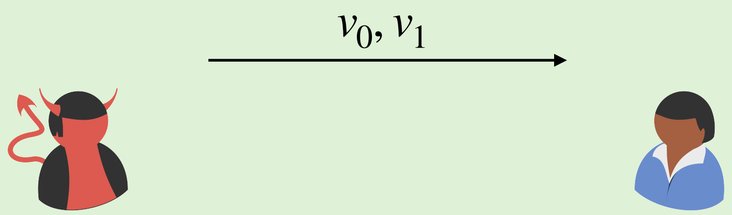
Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^{\ell}$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}.$$

Hiding



Commitment Schemes

Commitment Scheme

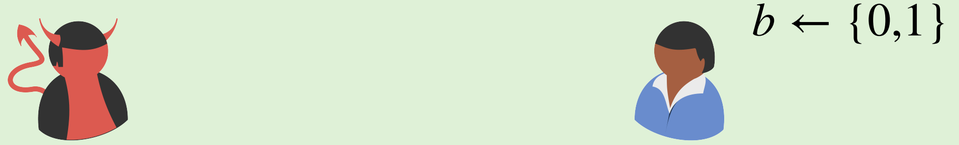
Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^{\ell}$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}.$$

Hiding



Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^{\ell}$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}.$$

Hiding

v_0, v_1

$b \leftarrow \{0,1\}$

$r \xleftarrow{\$} \{0,1\}^{\lambda}$

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^{\ell}$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}.$$

Hiding

v_0, v_1

$c = \mathsf{Com}(v_b\ ;\ r)$

$b \leftarrow \{0,1\}$

$r \xleftarrow{\$} \{0,1\}^{\lambda}$

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^{\ell}$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}.$$

Hiding

v_0, v_1

$c = \mathsf{Com}(v_b\ ;\ r)$

b'

$b \leftarrow \{0,1\}$

$r \xleftarrow{\$} \{0,1\}^{\lambda}$

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^{\ell}$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^{\lambda}\}.$$

Hiding

v_0, v_1

$c = \mathsf{Com}(v_b\ ;\ r)$

b'

$b \leftarrow \{0,1\}$

$r \xleftarrow{\$} \{0,1\}^{\lambda}$

Computationally bounded adversary: ZKP requires hiding against the verifier.

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^\ell$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}.$$

- **Binding:** For all $v_0, v_1 \in \{0,1\}^\ell$ and $r_0, r_1 \in \{0,1\}^\lambda$, such that $v_0 \neq v_1$, we have

$$\mathsf{Com}(v_0\ ;\ r_0) \neq \mathsf{Com}(v_1\ ;\ r_1).$$

Hiding

v_0, v_1

$c = \mathsf{Com}(v_b\ ;\ r)$

b'

$b \leftarrow \{0,1\}$

$r \xleftarrow{\$} \{0,1\}^\lambda$

Computationally bounded adversary: ZKP requires hiding against the verifier.

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^\ell$, we have

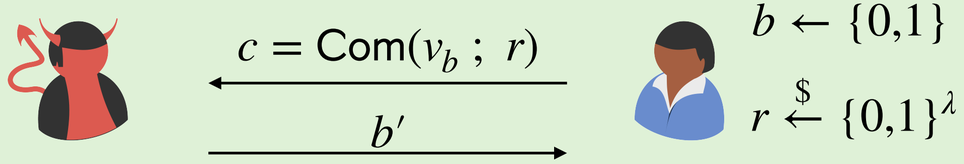
$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}.$$

- **Binding:** For all $v_0, v_1 \in \{0,1\}^\ell$ and $r_0, r_1 \in \{0,1\}^\lambda$, such that $v_0 \neq v_1$, we have

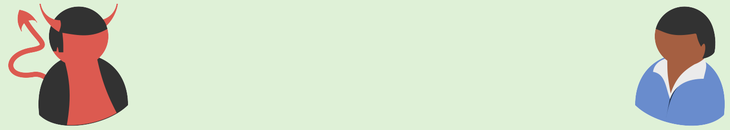
$$\mathsf{Com}(v_0\ ;\ r_0) \neq \mathsf{Com}(v_1\ ;\ r_1).$$

Hiding



Computationally bounded adversary: ZKP requires hiding against the verifier.

Binding



Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^\ell$, we have

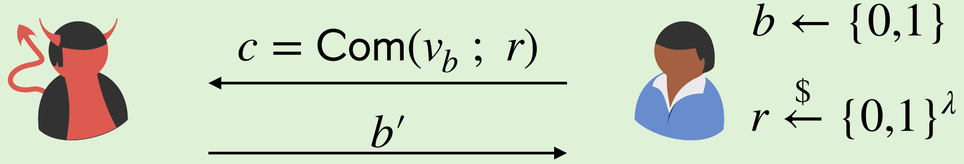
$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}.$$

- **Binding:** For all $v_0, v_1 \in \{0,1\}^\ell$ and $r_0, r_1 \in \{0,1\}^\lambda$, such that $v_0 \neq v_1$, we have

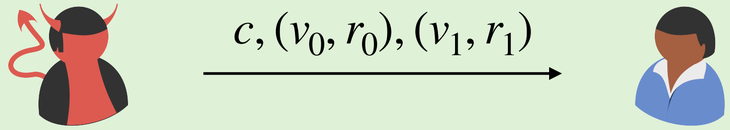
$$\mathsf{Com}(v_0\ ;\ r_0) \neq \mathsf{Com}(v_1\ ;\ r_1).$$

Hiding



Computationally bounded adversary: ZKP requires hiding against the verifier.

Binding



Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^\ell$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}.$$

- **Binding:** For all $v_0, v_1 \in \{0,1\}^\ell$ and $r_0, r_1 \in \{0,1\}^\lambda$, such that $v_0 \neq v_1$, we have

$$\mathsf{Com}(v_0\ ;\ r_0) \neq \mathsf{Com}(v_1\ ;\ r_1).$$

Hiding

v_0, v_1

$c = \mathsf{Com}(v_b\ ;\ r)$

b'

$b \leftarrow \{0,1\}$

$r \xleftarrow{\$} \{0,1\}^\lambda$

Computationally bounded adversary: ZKP requires hiding against the verifier.

Binding

$c, (v_0, r_0), (v_1, r_1)$

$c = \mathsf{Com}(v_0\ ;\ r_0)$

$c = \mathsf{Com}(v_1\ ;\ r_1)$

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^\ell$, we have

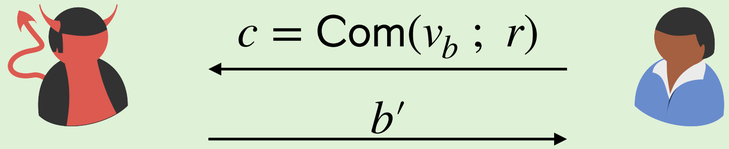
$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}.$$

- **Binding:** For all $v_0, v_1 \in \{0,1\}^\ell$ and $r_0, r_1 \in \{0,1\}^\lambda$, such that $v_0 \neq v_1$, we have

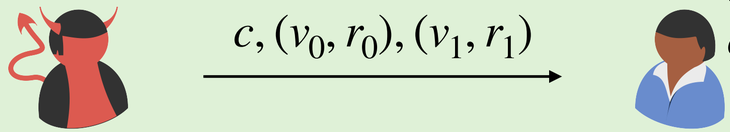
$$\mathsf{Com}(v_0\ ;\ r_0) \neq \mathsf{Com}(v_1\ ;\ r_1).$$

Hiding



Computationally bounded adversary: ZKP requires hiding against the verifier.

Binding



Computationally unbounded adversary: ZKP requires binding against the prover.

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^\ell$, we have

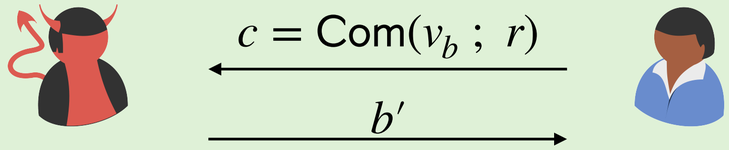
$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}.$$

- **Binding:** For all $v_0, v_1 \in \{0,1\}^\ell$ and $r_0, r_1 \in \{0,1\}^\lambda$, such that $v_0 \neq v_1$, we have

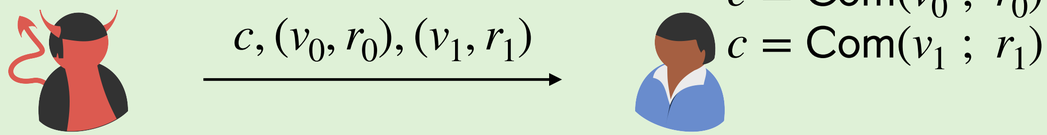
$$\mathsf{Com}(v_0\ ;\ r_0) \neq \mathsf{Com}(v_1\ ;\ r_1).$$

Hiding



Computationally bounded adversary: ZKP requires hiding against the verifier.

Binding



Computationally unbounded adversary: ZKP requires binding against the prover.

Intuitively, we require that set of commitments for distinct values is disjoint.

Commitment Schemes

Commitment Scheme

Let $\ell := \ell(\lambda)$ be a polynomial. A PPT algorithm Com is a commitment scheme for ℓ-bit strings if it satisfies the following properties.

- **Hiding:** For every $v_0, v_1 \in \{0,1\}^\ell$, we have

$$\{\mathsf{Com}(v_0\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}$$

$$\overset{c}{\approx}\ \{\mathsf{Com}(v_1\ ;\ r) : r \xleftarrow{\$} \{0,1\}^\lambda\}.$$

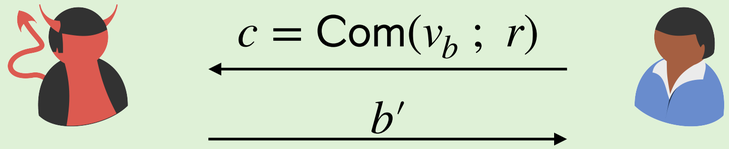
- **Binding:** For all $v_0, v_1 \in \{0,1\}^\ell$ and $r_0, r_1 \in \{0,1\}^\lambda$, such that $v_0 \neq v_1$, we have

$$\mathsf{Com}(v_0\ ;\ r_0) \neq \mathsf{Com}(v_1\ ;\ r_1).$$

This is a statistically-binding computationally-hiding commitment scheme.

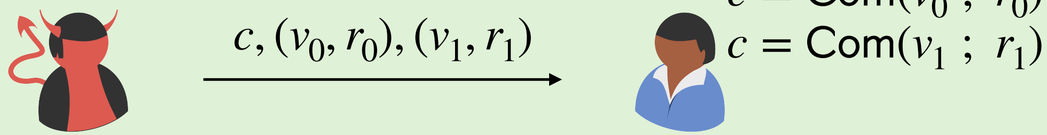
We can similarly define other variants.

Hiding



Computationally bounded adversary: ZKP requires hiding against the verifier.

Binding



Computationally unbounded adversary: ZKP requires binding against the prover.

Intuitively, we require that set of commitments for distinct values is disjoint.

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \Big(f(r),\ \mathsf{hc}(r) \oplus b\Big)$$

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \Big(f(r),\ \mathsf{hc}(r) \oplus b\Big)$$

Hiding:

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \big(f(r),\ \mathsf{hc}(r) \oplus b\big)$$

Hiding: $(f(r), \mathsf{hc}(r))$ is pseudorandom for uniformly random r (recall PRG from OWP).

Therefore, $\mathsf{hc}(r) \oplus b$ hides b.

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \big(f(r),\ \mathsf{hc}(r) \oplus b\big)$$

Hiding: $(f(r), \mathsf{hc}(r))$ is pseudorandom for uniformly random r (recall PRG from OWP).

Therefore, $\mathsf{hc}(r) \oplus b$ hides b.

Binding:

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \big(f(r),\ \mathsf{hc}(r) \oplus b\big)$$

Hiding: $(f(r), \mathsf{hc}(r))$ is pseudorandom for uniformly random r (recall PRG from OWP).

Therefore, $\mathsf{hc}(r) \oplus b$ hides b.

Binding: If $\mathsf{Com}(b_0\ ;\ r_0) = \mathsf{Com}(b_1\ ;\ r_1)$ then $\big(f(r_0),\ \mathsf{hc}(r_0) \oplus b_0\big) = \big(f(r_1),\ \mathsf{hc}(r_1) \oplus b_1\big)$.

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \big(f(r),\ \mathsf{hc}(r) \oplus b\big)$$

Hiding: $(f(r), \mathsf{hc}(r))$ is pseudorandom for uniformly random r (recall PRG from OWP).

Therefore, $\mathsf{hc}(r) \oplus b$ hides b.

Binding: If $\mathsf{Com}(b_0\ ;\ r_0) = \mathsf{Com}(b_1\ ;\ r_1)$ then $\big(f(r_0),\ \mathsf{hc}(r_0) \oplus b_0\big) = \big(f(r_1),\ \mathsf{hc}(r_1) \oplus b_1\big)$.

Since f is a permutation, this implies $r_0 = r_1$

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \big(f(r),\ \mathsf{hc}(r) \oplus b\big)$$

Hiding: $(f(r), \mathsf{hc}(r))$ is pseudorandom for uniformly random r (recall PRG from OWP).

Therefore, $\mathsf{hc}(r) \oplus b$ hides b.

Binding: If $\mathsf{Com}(b_0\ ;\ r_0) = \mathsf{Com}(b_1\ ;\ r_1)$ then $\big(f(r_0),\ \mathsf{hc}(r_0) \oplus b_0\big) = \big(f(r_1),\ \mathsf{hc}(r_1) \oplus b_1\big)$.

Since f is a permutation, this implies $r_0 = r_1$

Therefore, $\mathsf{hc}(r_0) = \mathsf{hc}(r_1)$ which implies that $b_0 = b_1$.

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \Big(f(r),\ \mathsf{hc}(r) \oplus b\Big)$$

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \Big(f(r),\ \mathsf{hc}(r) \oplus b\Big)$$

Can also be constructed from OWFs, but requires a bit more effort.

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \big(f(r),\ \mathsf{hc}(r) \oplus b\big)$$

Can also be constructed from OWFs, but requires a bit more effort.

Lemma: A bit commitment implies a commitment scheme for polynomial length bit strings.

Bit Commitments from OWP

Theorem: If one-way permutations exist, then there exists a commitment scheme for 1-bit messages.

Proof:

Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWP and let $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ be the hardcore predicate for f.

$$\mathsf{Com}(b\ ;\ r) = \Big(f(r),\ \mathsf{hc}(r) \oplus b\Big)$$

Can also be constructed from OWFs, but requires a bit more effort.

Lemma: A bit commitment implies a commitment scheme for polynomial length bit strings.

Simply commit to each bit separately.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

$$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Prover is efficient given the NP witness.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Prover is efficient given the NP witness.

Implies a ZKP for NP assuming commitment schemes.

Therefore, we get a ZKP for NP from OWF.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Completeness

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Completeness

If G is 3-colorable, the prover will always be able to answer the verifier's queries.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Soundness

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Soundness

- If G is not 3-colorable, then for any ϕ there is at least one edge with same color on both endpoints.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Soundness

- If G is not 3-colorable, then for any ϕ there is at least one edge with same color on both endpoints.
- From binding property, each c_i opens to unique color_i.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Soundness

- If G is not 3-colorable, then for any ϕ there is at least one edge with same color on both endpoints.
- From binding property, each c_i opens to unique color_i.
- Let $(\hat{i}, \hat{j})$ be the edge such that $\text{color}_{\hat{i}} = \text{color}_{\hat{j}}$.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Soundness

- If G is not 3-colorable, then for any ϕ there is at least one edge with same color on both endpoints.
- From binding property, each c_i opens to unique color_i.
- Let $(\hat{i}, \hat{j})$ be the edge such that $\text{color}_{\hat{i}} = \text{color}_{\hat{j}}$.
- With probability $|E|^{-1}$, V chooses $(\hat{i}, \hat{j})$ and catches P.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Soundness

- If G is not 3-colorable, then for any ϕ there is at least one edge with same color on both endpoints.
- From binding property, each c_i opens to unique color_i.
- Let $(\hat{i}, \hat{j})$ be the edge such that $\text{color}_{\hat{i}} = \text{color}_{\hat{j}}$.
- With probability $|E|^{-1}$, V chooses $(\hat{i}, \hat{j})$ and catches P.
- Probability of P evading in all repetitions is at most

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Soundness

- If G is not 3-colorable, then for any ϕ there is at least one edge with same color on both endpoints.
- From binding property, each c_i opens to unique color_i.
- Let $(\hat{i}, \hat{j})$ be the edge such that $\text{color}_{\hat{i}} = \text{color}_{\hat{j}}$.
- With probability $|E|^{-1}$, V chooses $(\hat{i}, \hat{j})$ and catches P.
- Probability of P evading in all repetitions is at most

$$\left(1 - \frac{1}{|E|}\right)^{n|E|} \approx e^{-n} = \text{negl}(\lambda).$$

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Zero-Knowledge

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Zero-Knowledge

- Intuition
  - V only sees two random colors. Hiding property of Com hides $\psi \circ \phi$.
  - Fresh ψ for each iteration. Verifier cannot correlate responses across iterations.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Zero-Knowledge

- Intuition
  - V only sees two random colors. Hiding property of Com hides $\psi \circ \phi$.
  - Fresh ψ for each iteration. Verifier cannot correlate responses across iterations.

We will prove that a single iteration is zero-knowledge.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

$$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Zero-Knowledge

- Intuition
  - V only sees two random colors. Hiding property of Com hides $\psi \circ \phi$.
  - Fresh ψ for each iteration. Verifier cannot correlate responses across iterations.

We will prove that a single iteration is zero-knowledge.

Theorem: Sequential repetition of any ZKP is also a ZKP.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

Repeat the following procedure $n|E|$ times.

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

  $$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Zero-Knowledge

- Intuition
  - V only sees two random colors. Hiding property of Com hides $\psi \circ \phi$.
  - Fresh ψ for each iteration. Verifier cannot correlate responses across iterations.

We will prove that a single iteration is zero-knowledge.

We need to

1. Construct a simulator S.
2. Prove that S is PPT.
3. Prove that output distribution of S is correct.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

$$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

$$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\text{color}_{i'}, \text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\text{color}_{i'} \neq \text{color}_{j'}$. Set every other $\text{color}_i = 1$.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and
$$c_i := \text{Com}(\text{color}_i)$$
and sends $(c_1, \ldots, c_n)$ to V.
- $V \to P$: V samples a random edge (i, j) and sends it to P.
- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.
- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\text{color}_{i'}, \text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\text{color}_{i'} \neq \text{color}_{j'}$. Set every other $\text{color}_i = 1$.
- For every $i \in \{1, \ldots, n\}$, compute $c_i \leftarrow \text{Com}(\text{color}_i)$.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

$$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\text{color}_{i'}, \text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\text{color}_{i'} \neq \text{color}_{j'}$. Set every other $\text{color}_i = 1$.

- For every $i \in \{1, \ldots, n\}$, compute $c_i \leftarrow \text{Com}(\text{color}_i)$.

- Execute $\widehat{V}$ and send it $(c_1, \ldots, c_n)$. Let its challenge be $(i, j) \in E$.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

$$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\text{color}_{i'}, \text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\text{color}_{i'} \neq \text{color}_{j'}$. Set every other $\text{color}_i = 1$.

- For every $i \in \{1, \ldots, n\}$, compute $c_i \leftarrow \text{Com}(\text{color}_i)$.

- Execute $\widehat{V}$ and send it $(c_1, \ldots, c_n)$. Let its challenge be $(i, j) \in E$.

- If $(i, j) = (i', j')$, then open $c_{i'}$ and $c_{j'}$. Else, restart the above process.

ZKP for Graph 3-Coloring

Statement: $G = (V, E)$ where $|V| = n = \text{poly}(\lambda)$.

Prover's NP witness: Color assignment $\phi : V \to \{1, 2, 3\}$.

ZKP for Graph 3-Coloring

~~Repeat the following procedure $n|E|$ times.~~

- $P \to V$: P samples $\psi \xleftarrow{\$} \text{Perm}_3$ uniformly at random. For each vertex $v_i \in V$ it computes $\text{color}_i := \psi\left(\phi(v_i)\right)$ and

$$c_i := \text{Com}(\text{color}_i)$$

  and sends $(c_1, \ldots, c_n)$ to V.

- $V \to P$: V samples a random edge (i, j) and sends it to P.

- $P \to V$: P opens c_i and c_j to reveal color_i and color_j.

- V: If the openings are valid and $\text{color}_i \neq \text{color}_j$, then V continues to the next iteration. Else, it rejects.

If V does not reject in any iteration, it accepts the proof.

Simulator S

- Sample $(i', j') \xleftarrow{\$} E$ and colors $\text{color}_{i'}, \text{color}_{j'} \xleftarrow{\$} \{1, 2, 3\}$ such that $\text{color}_{i'} \neq \text{color}_{j'}$. Set every other $\text{color}_i = 1$.

- For every $i \in \{1, \ldots, n\}$, compute $c_i \leftarrow \text{Com}(\text{color}_i)$.

- Execute $\widehat{V}$ and send it $(c_1, \ldots, c_n)$. Let its challenge be $(i, j) \in E$.

- If $(i, j) = (i', j')$, then open $c_{i'}$ and $c_{j'}$. Else, restart the above process.

- If simulation has not succeeded after $n|E|$ attempts, then output $\perp$.