CS 594 – Advanced Crypto (Spring 2026)

Alex Block

Lecture 13

March 02, 2026

Zero-knowledge for all of NP

Zero-knowledge for all of NP

- Today, we'll show the following theorem.

Zero-knowledge for all of NP

- Today, we'll show the following theorem.

Theorem 1 (Goldwasser, Micali, Widgerson (86))

If one-way functions exist, then $\mathbf{NP} \subseteq \mathbf{CZK}$. $\simeq \mathcal{IP}$

Zero-knowledge for all of NP

- Today, we'll show the following theorem.

Theorem 1 (Goldwasser, Micali, Widgerson (86))

If one-way functions exist, then **NP** $\subseteq$ **CZK**.

- Our proof will go through Blum's protocol and result.

Zero-knowledge for all of NP

- Today, we'll show the following theorem.

Theorem 1 (Goldwasser, Micali, Widgerson (86))

If one-way functions exist, then $\mathbf{NP} \subseteq \mathbf{CZK}$.

- Our proof will go through Blum's protocol and result.

Theorem 2 (Blum (86))

If statistically-binding, computationally hiding commitments exist, then $\mathbf{HAM} \in \mathbf{CZK}$, *where*

$$\mathbf{HAM} = \{G = (V, E) \colon G \text{ has a Hamiltonian cycle.}\}.$$

One-way Functions

One-way Functions

- *One-way functions* are the minimal assumption in cryptography.

One-way Functions

- *One-way functions* are the minimal assumption in cryptography.

- Intuitive definition: $f\colon \{0,1\}^* \to \{0,1\}^*$ is one-way if $f(x)$ is "easy" to compute for any x, and $y = f(r)$ is hard to invert for random r.

One-way Functions

- *One-way functions* are the minimal assumption in cryptography.
- Intuitive definition: $f\colon \{0,1\}^* \to \{0,1\}^*$ is one-way if $f(x)$ is "easy" to compute for any x, and $y = f(r)$ is hard to invert for random r.
- We first formalize what it means to be "hard to invert" via the following experiment.

One-way Functions

- *One-way functions* are the minimal assumption in cryptography.
- Intuitive definition: $f\colon \{0,1\}^* \to \{0,1\}^*$ is one-way if $f(x)$ is "easy" to compute for any x, and $y = f(r)$ is hard to invert for random r.
- We first formalize what it means to be "hard to invert" via the following experiment.

Inverting Experiment $\mathsf{Inv}(\mathcal{A}, f, \lambda)$

One-way Functions

- *One-way functions* are the minimal assumption in cryptography.
- Intuitive definition: $f\colon \{0,1\}^* \to \{0,1\}^*$ is one-way if $f(x)$ is "easy" to compute for any x, and $y = f(r)$ is hard to invert for random r.
- We first formalize what it means to be "hard to invert" via the following experiment.

Inverting Experiment $\mathsf{Inv}(\mathcal{A}, f, \lambda)$

1. Sample $x \xleftarrow{\$} \{0,1\}^\lambda$ and compute $y = f(x)$.

One-way Functions

- *One-way functions* are the minimal assumption in cryptography.
- Intuitive definition: $f\colon \{0,1\}^* \to \{0,1\}^*$ is one-way if $f(x)$ is "easy" to compute for any x, and $y = f(r)$ is hard to invert for random r.
- We first formalize what it means to be "hard to invert" via the following experiment.

Inverting Experiment $\mathsf{Inv}(\mathcal{A}, f, \lambda)$

1. Sample $x \xleftarrow{\$} \{0,1\}^\lambda$ and compute $y = f(x)$.
2. Obtain $x' \leftarrow \mathcal{A}(1^\lambda, f, y)$.

One-way Functions

- *One-way functions* are the minimal assumption in cryptography.
- Intuitive definition: $f\colon \{0,1\}^* \to \{0,1\}^*$ is one-way if $f(x)$ is "easy" to compute for any x, and $y = f(r)$ is hard to invert for random r.
- We first formalize what it means to be "hard to invert" via the following experiment.

Inverting Experiment $\mathsf{Inv}(\mathcal{A}, f, \lambda)$

1. Sample $x \xleftarrow{\$} \{0,1\}^\lambda$ and compute $y = f(x)$.
2. Obtain $x' \leftarrow \mathcal{A}(1^\lambda, f, y)$.
3. The output of the experiment is 1 if $f(x') = y$ and 0 otherwise.

One-way Functions

Definition 1 (One-way Function)

One-way Functions

Definition 1 (One-way Function)

A function $f : \{0,1\}^* \to \{0,1\}^*$ is *one-way* if the following hold.

One-way Functions

Definition 1 (One-way Function)

A function $f : \{0,1\}^* \to \{0,1\}^*$ is *one-way* if the following hold.

- There is a polynomial-time algorithm M_f for computing f; that is, $M_f(x) = f(x)$ for all x.

One-way Functions

Definition 1 (One-way Function)

A function $f : \{0,1\}^* \to \{0,1\}^*$ is *one-way* if the following hold.

- There is a polynomial-time algorithm M_f for computing f; that is, $M_f(x) = f(x)$ for all x.
- For all (non-uniform) PPT algorithms $\mathcal{A}$, there is a negligible function negl such that

One-way Functions

Definition 1 (One-way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is *one-way* if the following hold.

- There is a polynomial-time algorithm M_f for computing f; that is, $M_f(x) = f(x)$ for all x.
- For all (non-uniform) PPT algorithms $\mathcal{A}$, there is a negligible function negl such that

$$\Pr[\mathsf{Inv}(\mathcal{A}, f, \lambda) = 1] \leqslant \mathsf{negl}(\lambda).$$

One-way Functions

Definition 1 (One-way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is *one-way* if the following hold.

- There is a polynomial-time algorithm M_f for computing f; that is, $M_f(x) = f(x)$ for all x.
- For all (non-uniform) PPT algorithms $\mathcal{A}$, there is a negligible function negl such that

$$\Pr[\mathsf{Inv}(\mathcal{A}, f, \lambda) = 1] \leqslant \mathsf{negl}(\lambda).$$

Theorem 3

If one-way functions exist, then the following also exist.

One-way Functions

Definition 1 (One-way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is *one-way* if the following hold.

- There is a polynomial-time algorithm M_f for computing f; that is, $M_f(x) = f(x)$ for all x.
- For all (non-uniform) PPT algorithms $\mathcal{A}$, there is a negligible function negl such that

$$\Pr[\mathsf{Inv}(\mathcal{A}, f, \lambda) = 1] \leqslant \mathsf{negl}(\lambda).$$

Theorem 3

If one-way functions exist, then the following also exist.

- *PRGs*
- *PRFs*
- *PRPs*

One-way Functions

Definition 1 (One-way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is *one-way* if the following hold.

- There is a polynomial-time algorithm M_f for computing f; that is, $M_f(x) = f(x)$ for all x.
- For all (non-uniform) PPT algorithms $\mathcal{A}$, there is a negligible function negl such that

$$\Pr[\mathsf{Inv}(\mathcal{A}, f, \lambda) = 1] \leqslant \mathsf{negl}(\lambda).$$

Theorem 3

If one-way functions exist, then the following also exist.

- *PRGs*
- *PRFs*
- *PRPs*
- *MACs*
- *CPA-SKE*
- *CCA-SKE*

One-way Functions

Definition 1 (One-way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is *one-way* if the following hold.

- There is a polynomial-time algorithm M_f for computing f; that is, $M_f(x) = f(x)$ for all x.
- For all (non-uniform) PPT algorithms $\mathcal{A}$, there is a negligible function negl such that

$$\Pr[\mathsf{Inv}(\mathcal{A}, f, \lambda) = 1] \leqslant \mathsf{negl}(\lambda).$$

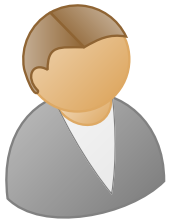
Theorem 3

If one-way functions exist, then the following also exist.

- *PRGs*
- *PRFs*
- *PRPs*
- *MACs*
- *CPA-SKE*
- *CCA-SKE*
- *Stat. Bind., Comp. Hid. Commitments*
- *Comp. Bind., Stat. Hid. Commitments*

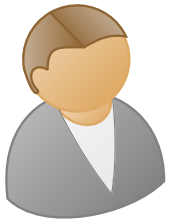
Commitment Schemes, Intuitively

Commitment Schemes, Intuitively



Committer C

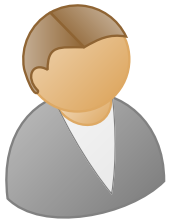
Commitment Schemes, Intuitively



Committer C

Reciever R

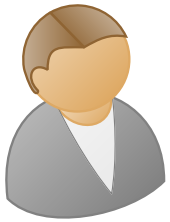
Commitment Schemes, Intuitively



Committer C

Reciever R

Commitment Schemes, Intuitively



Committer C

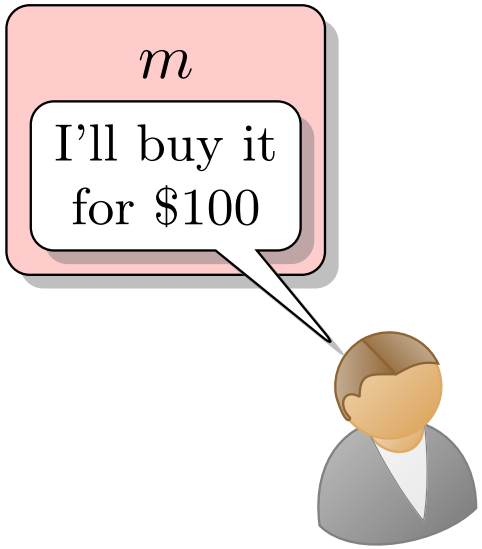
Reciever R

Commitment Schemes, Intuitively

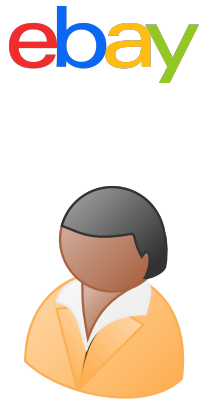
Committer C

Reciever R

Commitment Schemes, Intuitively

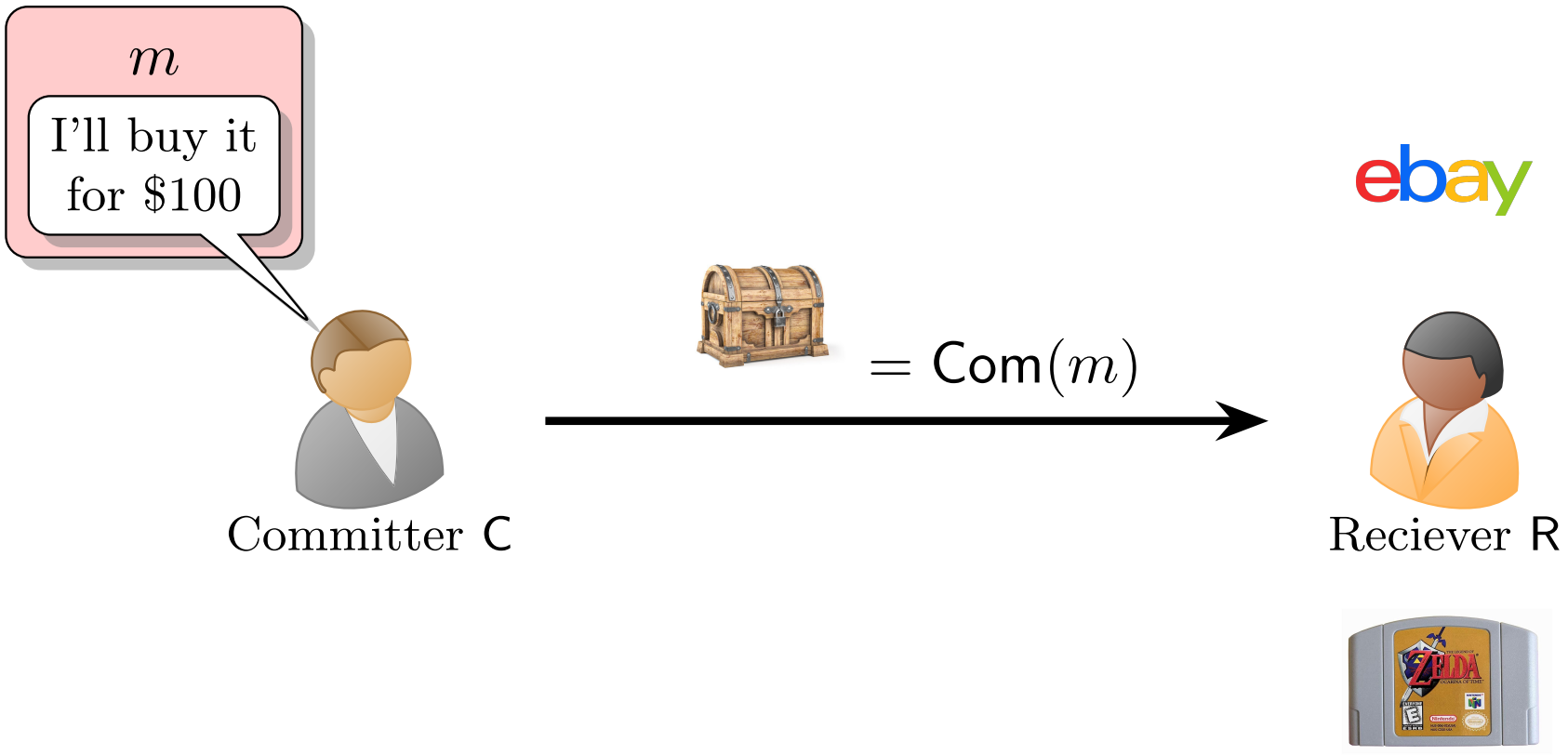


Committer C

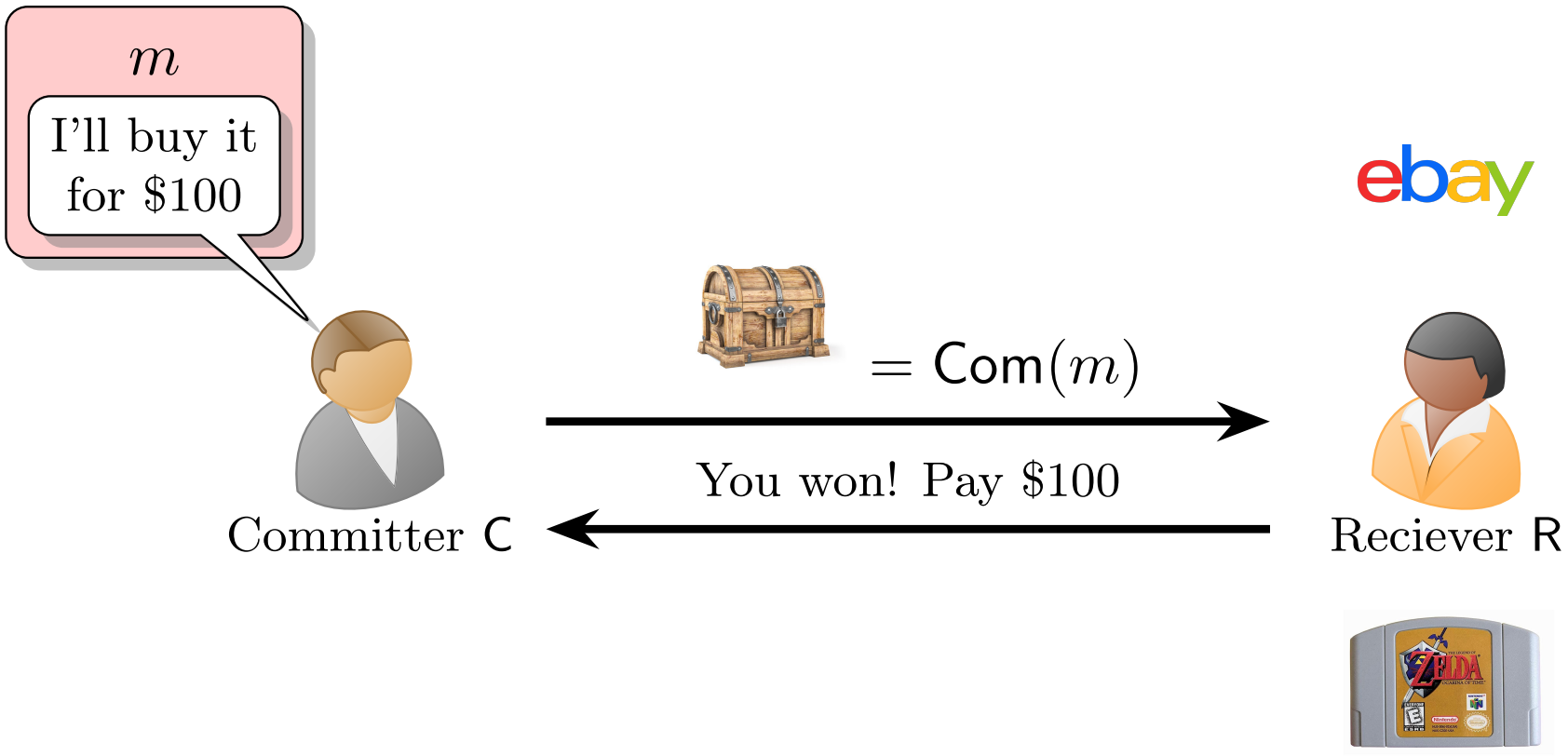


Reciever R

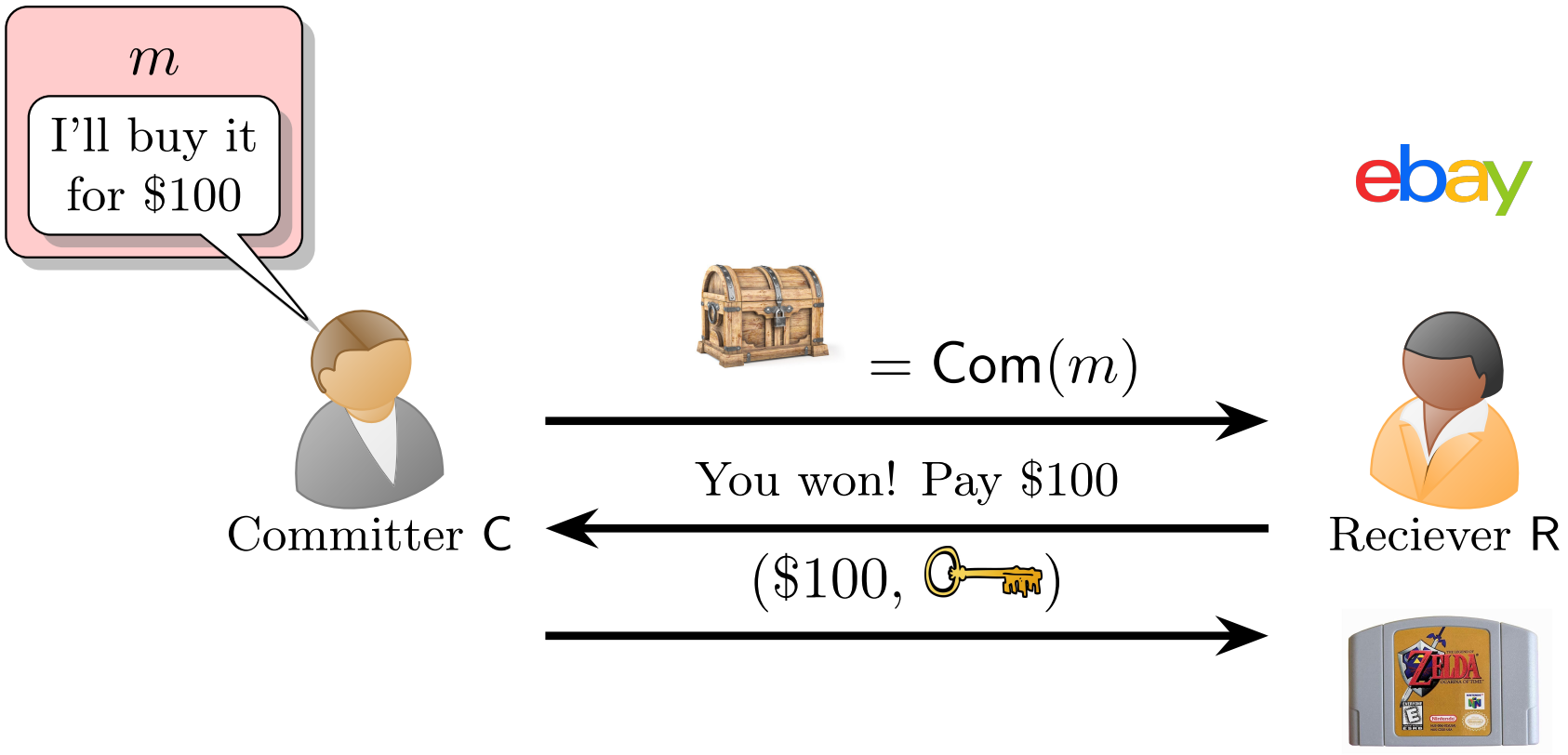
Commitment Schemes, Intuitively



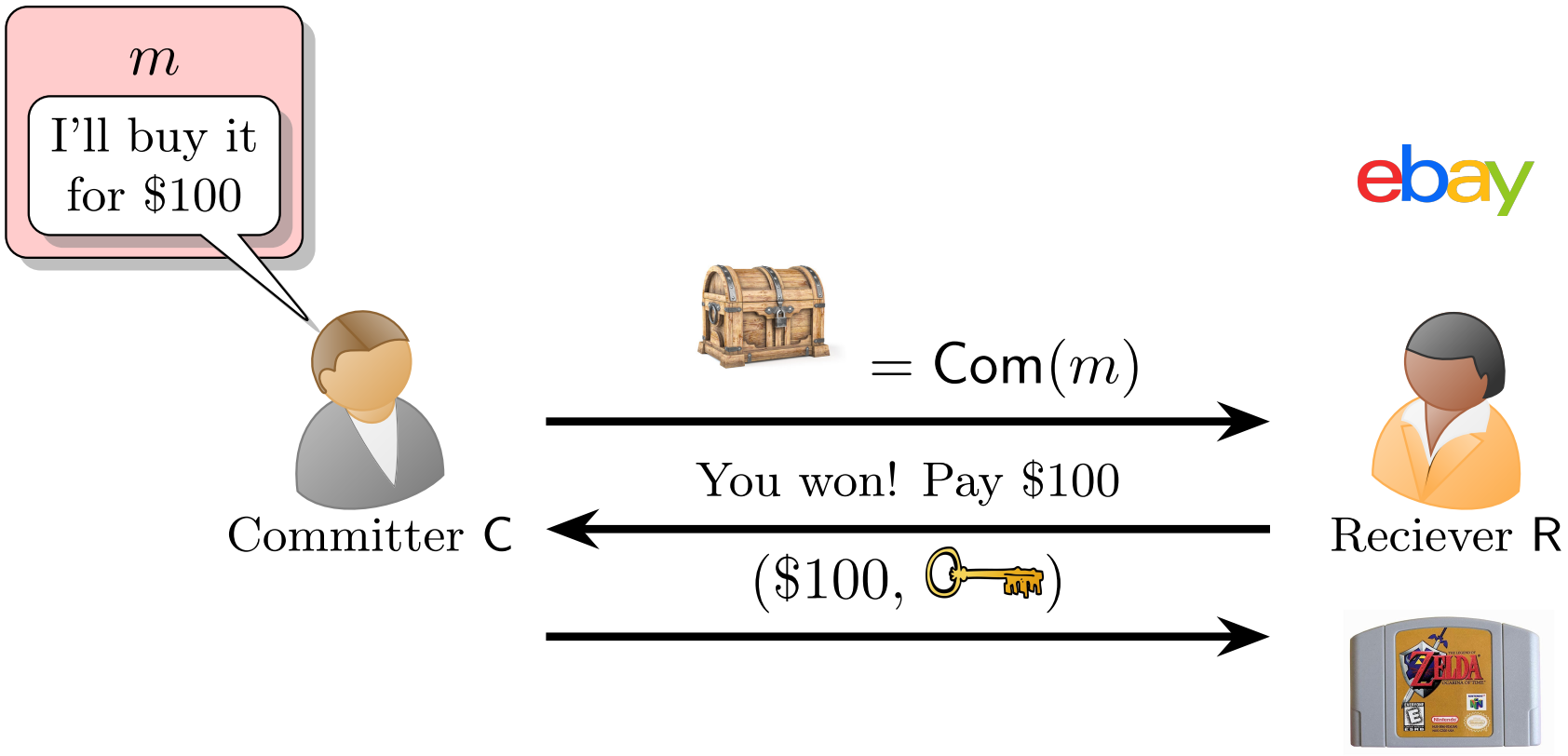
Commitment Schemes, Intuitively



Commitment Schemes, Intuitively

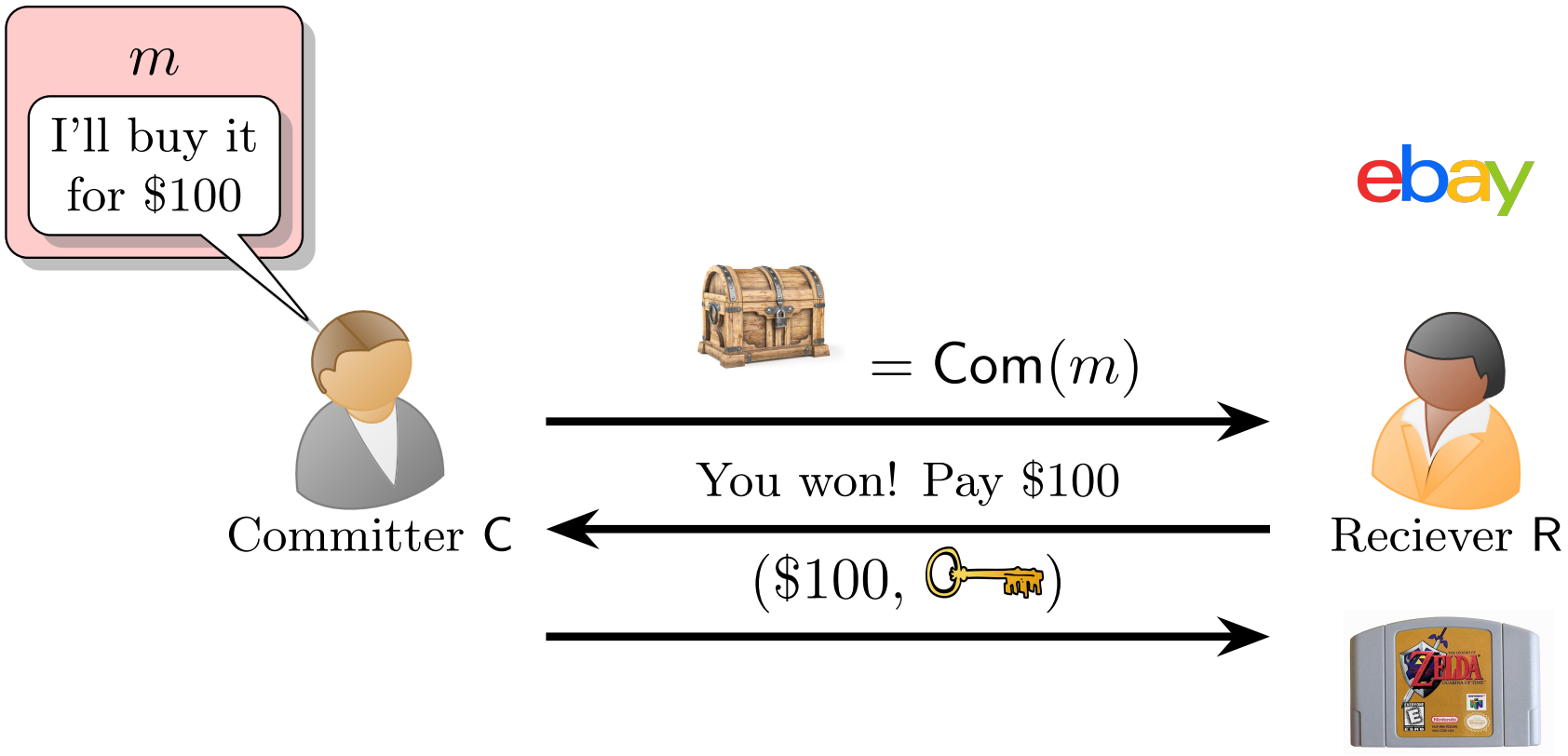


Commitment Schemes, Intuitively



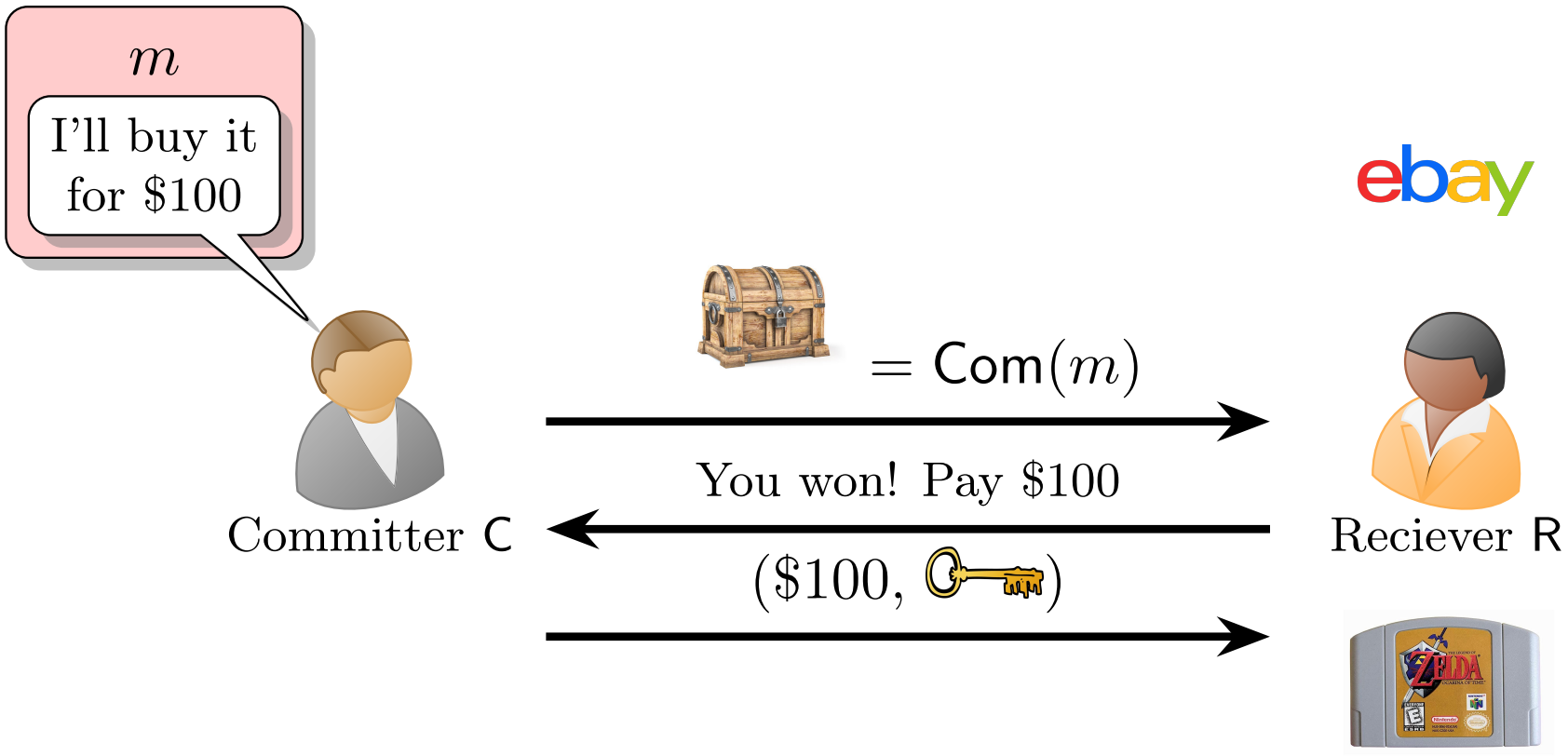
- C can't send a different key which opens the box to \$50.

Commitment Schemes, Intuitively



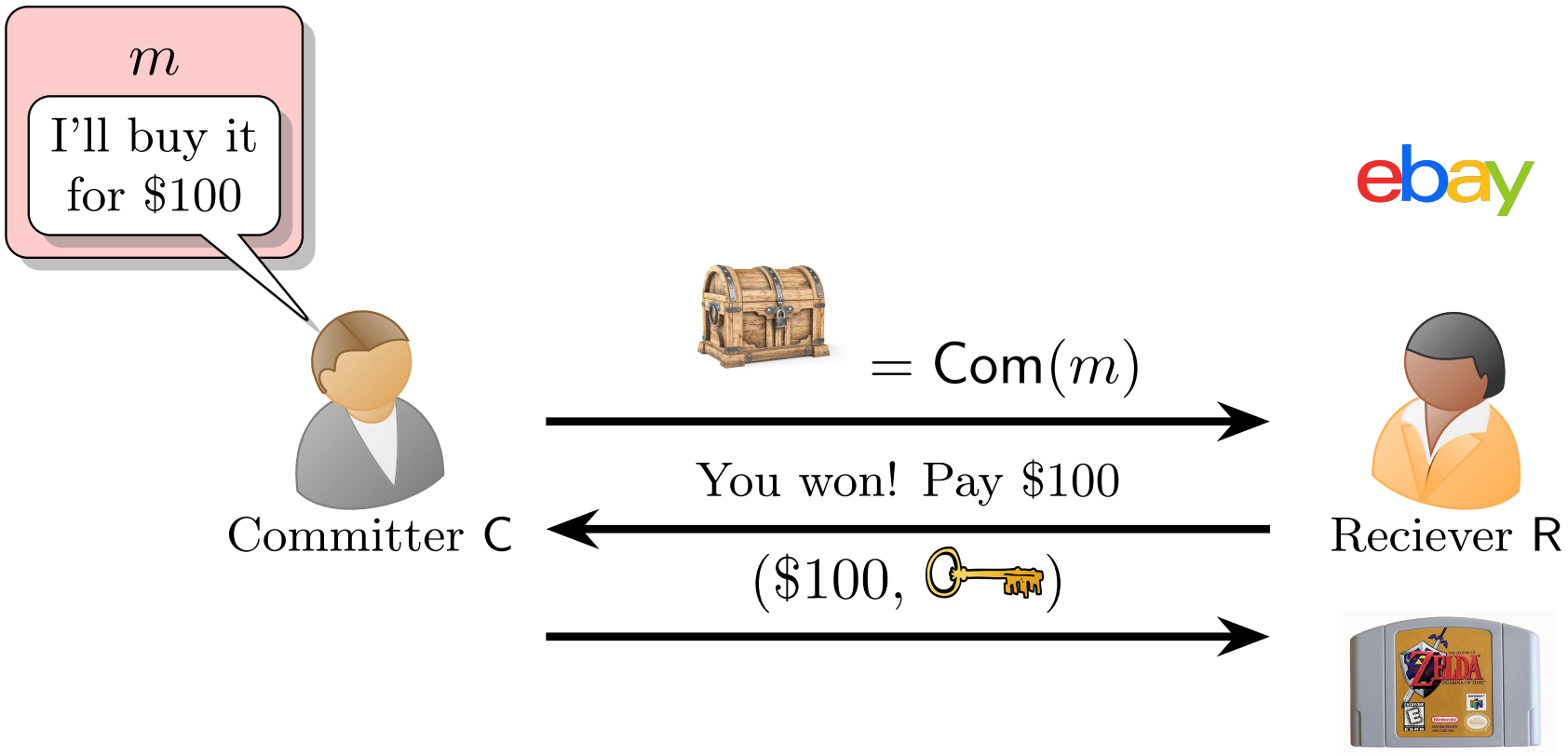
- C can't send a different key which opens the box to \$50.
  - Commitment is *binding*.

Commitment Schemes, Intuitively



- C can't send a different key which opens the box to \$50.
  - Commitment is *binding*.
- In some cases, don't want R to see what is in the box!

Commitment Schemes, Intuitively



- C can't send a different key which opens the box to \$50.
  - Commitment is *binding*.
- In some cases, don't want R to see what is in the box!
  - Commitment is *hiding*.

Commitment Schemes: Definition

Definition 2 (Commitment Scheme)

Commitment Schemes: Definition

Definition 2 (Commitment Scheme)

A *commitment scheme* consists of two parties, a committer C and receiver R, and a tuple of (possibly interactive) PPT algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ with the following syntax.

Commitment Schemes: Definition

Definition 2 (Commitment Scheme)

A *commitment scheme* consists of two parties, a committer C and receiver R, and a tuple of (possibly interactive) PPT algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ with the following syntax.

- $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$: a PPT algorithm for generating public parameters pp given security parameter λ as input.

Commitment Schemes: Definition

Definition 2 (Commitment Scheme)

A *commitment scheme* consists of two parties, a committer C and receiver R, and a tuple of (possibly interactive) PPT algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ with the following syntax.

- $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$: a PPT algorithm for generating public parameters pp given security parameter λ as input.
- $(C, d) \leftarrow \mathsf{Com}(\mathsf{pp}, m)$: a PPT (possibly interactive) commitment algorithm, which takes as input public parameters pp and a message m, and outputs a commitment C and decommitment information d.

Commitment Schemes: Definition

Definition 2 (Commitment Scheme)

A *commitment scheme* consists of two parties, a committer C and receiver R, and a tuple of (possibly interactive) PPT algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ with the following syntax.

- $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$: a PPT algorithm for generating public parameters pp given security parameter λ as input.
- $(C, d) \leftarrow \mathsf{Com}(\mathsf{pp}, m)$: a PPT (possibly interactive) commitment algorithm, which takes as input public parameters pp and a message m, and outputs a commitment C and decommitment information d.
- $0/1 = \mathsf{Open}(\mathsf{pp}, m, C, d)$: a deterministic (possibly interactive) opening algorithm, which takes as input public parameters pp, a message m, a commitment C, and decommitment information d, and outputs 0 or 1.

Commitment Schemes: Definition

Definition 2 (Commitment Scheme)

A *commitment scheme* consists of two parties, a committer C and receiver R, and a tuple of (possibly interactive) PPT algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ with the following syntax.

- $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$: a PPT algorithm for generating public parameters pp given security parameter λ as input.
- $(C, d) \leftarrow \mathsf{Com}(\mathsf{pp}, m)$: a PPT (possibly interactive) commitment algorithm, which takes as input public parameters pp and a message m, and outputs a commitment C and decommitment information d.
- $0/1 = \mathsf{Open}(\mathsf{pp}, m, C, d)$: a deterministic (possibly interactive) opening algorithm, which takes as input public parameters pp, a message m, a commitment C, and decommitment information d, and outputs 0 or 1.

- Usually, d is just the randomness used by Com.

Commitment Schemes: Correctness

Definition 3 (Correctness of Commitment Scheme)

Commitment Schemes: Correctness

Definition 3 (Correctness of Commitment Scheme)

For any commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$, we require for all λ and messages m:

Commitment Schemes: Correctness

Definition 3 (Correctness of Commitment Scheme)

For any commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$, we require for all λ and messages m:

$$\Pr_{\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)} [1 = \mathsf{Open}(\mathsf{pp}, m, \mathsf{Com}(\mathsf{pp}, m))] = 1.$$

Commitment Schemes: Correctness

Definition 3 (Correctness of Commitment Scheme)

For any commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$, we require for all λ and messages m:

$$\Pr_{\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)} [1 = \mathsf{Open}(\mathsf{pp}, m, \mathsf{Com}(\mathsf{pp}, m))] = 1.$$

- Probability above is also taken over the randomness of Com.

Commitment Schemes: Hiding

Definition 4 (Computationally Hiding)

Commitment Schemes: Hiding

Definition 4 (Computationally Hiding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *computationally hiding* if for all (possibly malicious) PPT receivers R^* and all pairs of messages m_0, m_1, the commitments C_0 and C_1 are computationally indistinguishable, where $(C_0, d_0) \leftarrow \mathsf{Com}(\mathsf{pp}, m_0)$ and $(C_1, d_1) \leftarrow \mathsf{Com}(\mathsf{pp}, m_1)$.

Commitment Schemes: Hiding

Definition 4 (Computationally Hiding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *computationally hiding* if for all (possibly malicious) PPT receivers R^* and all pairs of messages m_0, m_1, the commitments C_0 and C_1 are computationally indistinguishable, where $(C_0, d_0) \leftarrow \mathsf{Com}(\mathsf{pp}, m_0)$ and $(C_1, d_1) \leftarrow \mathsf{Com}(\mathsf{pp}, m_1)$. Equivalently, R^* wins the below game with at most $1/2 + \mathsf{negl}(\lambda)$ probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

Commitment Schemes: Hiding

Definition 4 (Computationally Hiding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *computationally hiding* if for all (possibly malicious) PPT receivers R^* and all pairs of messages m_0, m_1, the commitments C_0 and C_1 are computationally indistinguishable, where $(C_0, d_0) \leftarrow \mathsf{Com}(\mathsf{pp}, m_0)$ and $(C_1, d_1) \leftarrow \mathsf{Com}(\mathsf{pp}, m_1)$. Equivalently, R^* wins the below game with at most $1/2 + \mathsf{negl}(\lambda)$ probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

HidGame(pp):

Commitment Schemes: Hiding

Definition 4 (Computationally Hiding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *computationally hiding* if for all (possibly malicious) PPT receivers R^* and all pairs of messages m_0, m_1, the commitments C_0 and C_1 are computationally indistinguishable, where $(C_0, d_0) \leftarrow \mathsf{Com}(\mathsf{pp}, m_0)$ and $(C_1, d_1) \leftarrow \mathsf{Com}(\mathsf{pp}, m_1)$. Equivalently, R^* wins the below game with at most $1/2 + \mathsf{negl}(\lambda)$ probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

HidGame(pp):

- R^* sends m_0, m_1 to C.

Commitment Schemes: Hiding

Definition 4 (Computationally Hiding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *computationally hiding* if for all (possibly malicious) PPT receivers R^* and all pairs of messages m_0, m_1, the commitments C_0 and C_1 are computationally indistinguishable, where $(C_0, d_0) \leftarrow \mathsf{Com}(\mathsf{pp}, m_0)$ and $(C_1, d_1) \leftarrow \mathsf{Com}(\mathsf{pp}, m_1)$. Equivalently, R^* wins the below game with at most $1/2 + \mathsf{negl}(\lambda)$ probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

HidGame(pp):

- R^* sends m_0, m_1 to C.
- C samples $b \xleftarrow{\$} \{0,1\}$, computes $(C_b, d_b) \leftarrow \mathsf{Com}(\mathsf{pp}, m_b)$, and sends C_b to R^*.

Commitment Schemes: Hiding

Definition 4 (Computationally Hiding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *computationally hiding* if for all (possibly malicious) PPT receivers R^* and all pairs of messages m_0, m_1, the commitments C_0 and C_1 are computationally indistinguishable, where $(C_0, d_0) \leftarrow \mathsf{Com}(\mathsf{pp}, m_0)$ and $(C_1, d_1) \leftarrow \mathsf{Com}(\mathsf{pp}, m_1)$. Equivalently, R^* wins the below game with at most $1/2 + \mathsf{negl}(\lambda)$ probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

HidGame(pp):

- R^* sends m_0, m_1 to C.
- C samples $b \xleftarrow{\$} \{0,1\}$, computes $(C_b, d_b) \leftarrow \mathsf{Com}(\mathsf{pp}, m_b)$, and sends C_b to R^*.
- R^* outputs $b' \in \{0,1\}$ (R^* *wins if* $b = b'$).

Commitment Schemes: Binding

Definition 5 (Statistically Binding)

Commitment Schemes: Binding

Definition 5 (Statistically Binding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *statistically binding* if for all (possibly malicious) committers C^* and all pairs of messages $m_0 \neq m_1$, C^* wins the following game with at most negligible probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

Commitment Schemes: Binding

Definition 5 (Statistically Binding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *statistically binding* if for all (possibly malicious) committers C^* and all pairs of messages $m_0 \neq m_1$, C^* wins the following game with at most negligible probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

$\mathsf{BinGame}(\mathsf{pp}, m_0, m_1)$:

Commitment Schemes: Binding

Definition 5 (Statistically Binding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *statistically binding* if for all (possibly malicious) committers C^* and all pairs of messages $m_0 \neq m_1$, C^* wins the following game with at most negligible probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

$\mathsf{BinGame}(\mathsf{pp}, m_0, m_1)$:

- $(C, d_0, d_1) \leftarrow \mathsf{C}^*(\mathsf{pp}, m_0, m_1)$.

Commitment Schemes: Binding

Definition 5 (Statistically Binding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *statistically binding* if for all (possibly malicious) committers C^* and all pairs of messages $m_0 \neq m_1$, C^* wins the following game with at most negligible probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

$\mathsf{BinGame}(\mathsf{pp}, m_0, m_1)$:

- $(C, d_0, d_1) \leftarrow \mathsf{C}^*(\mathsf{pp}, m_0, m_1)$.
- Output (C, d_0, d_1).

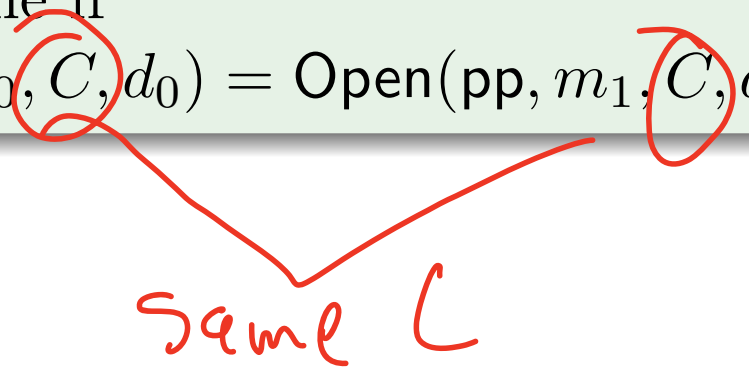
Commitment Schemes: Binding

Definition 5 (Statistically Binding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *statistically binding* if for all (possibly malicious) committers C^* and all pairs of messages $m_0 \neq m_1$, C^* wins the following game with at most negligible probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

BinGame(pp, m_0, m_1):

- $(C, d_0, d_1) \leftarrow \mathsf{C}^*(\mathsf{pp}, m_0, m_1)$.
- Output (C, d_0, d_1).
- C^* wins the game if $1 = \mathsf{Open}(\mathsf{pp}, m_0, C, d_0) = \mathsf{Open}(\mathsf{pp}, m_1, C, d_1)$.



Commitment Schemes: Binding

Definition 5 (Statistically Binding)

We say that commitment scheme $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ is *statistically binding* if for all (possibly malicious) committers C^* and all pairs of messages $m_0 \neq m_1$, C^* wins the following game with at most negligible probability for all λ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

BinGame(pp, m_0, m_1):

- $(C, d_0, d_1) \leftarrow \mathsf{C}^*(\mathsf{pp}, m_0, m_1)$.
- Output (C, d_0, d_1).
- C^* wins the game if $1 = \mathsf{Open}(\mathsf{pp}, m_0, C, d_0) = \mathsf{Open}(\mathsf{pp}, m_1, C, d_1)$.

- **Note**: We say "statistically-binding" commitments to mean they are both statistically binding and computationally hiding.

Examples of Statistically-binding Commitments

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).
  - $\mathsf{Open}(\mathsf{pp}, m, C, d)$:

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).
  - $\mathsf{Open}(\mathsf{pp}, m, C, d)$:
    - Parse C as (G, H).

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).
  - $\mathsf{Open}(\mathsf{pp}, m, C, r)$:
    - Parse C as (G, H).
    - Check if $g^R = G$ and $h^R \cdot m = H$.

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).
  - $\mathsf{Open}(\mathsf{pp}, m, C, d)$:
    - Parse C as (G, H).
    - Check if $g^R = G$ and $h^R \cdot m = H$.
    - Output 1 if the checks pass; otherwise output 0.

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).
  - $\mathsf{Open}(\mathsf{pp}, m, C, d)$:
    - Parse C as (G, H).
    - Check if $g^R = G$ and $h^R \cdot m = H$.
    - Output 1 if the checks pass; otherwise output 0.

- *Bit-commitments* from any PRG (and hence OWF):

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).
  - $\mathsf{Open}(\mathsf{pp}, m, C, d)$:
    - Parse C as (G, H).
    - Check if $g^R = G$ and $h^R \cdot m = H$.
    - Output 1 if the checks pass; otherwise output 0.

- *Bit-commitments* from any PRG (and hence OWF):
  - $(G) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$: $G\colon \{0,1\}^\lambda \to \{0,1\}^{\lambda'}$ is a PRG. $\lambda' > \lambda$

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).
  - $\mathsf{Open}(\mathsf{pp}, m, C, d)$:
    - Parse C as (G, H).
    - Check if $g^R = G$ and $h^R \cdot m = H$.
    - Output 1 if the checks pass; otherwise output 0.

- *Bit-commitments* from any PRG (and hence OWF):
  - $(G) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$: $G\colon \{0,1\}^\lambda \to \{0,1\}^{\lambda'}$ is a PRG.
  - For any bit $b \in \{0,1\}$, $\mathsf{Com}(G, b)$ does the following.

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).
  - $\mathsf{Open}(\mathsf{pp}, m, C, d)$:
    - Parse C as (G, H).
    - Check if $g^R = G$ and $h^R \cdot m = H$.
    - Output 1 if the checks pass; otherwise output 0.

- *Bit-commitments* from any PRG (and hence OWF):
  - $(G) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$: $G\colon \{0,1\}^\lambda \to \{0,1\}^{\lambda'}$ is a PRG.
  - For any bit $b \in \{0,1\}$, $\mathsf{Com}(G, b)$ does the following.
    - R samples and sends $\mathsf{r} \xleftarrow{\$} \{0,1\}^{\lambda'}$.

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).
  - $\mathsf{Open}(\mathsf{pp}, m, C, d)$:
    - Parse C as (G, H).
    - Check if $g^R = G$ and $h^R \cdot m = H$.
    - Output 1 if the checks pass; otherwise output 0.

- *Bit-commitments* from any PRG (and hence OWF):
  - $(G) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$: G: $\{0,1\}^\lambda \to \{0,1\}^{\lambda'}$ is a PRG.
  - For any bit $b \in \{0,1\}$, $\mathsf{Com}(G, b)$ does the following.
    - R samples and sends $\mathsf{r} \xleftarrow{\$} \{0,1\}^{\lambda'}$.
    - C samples $k \xleftarrow{\$} \{0,1\}^\lambda$, computes $C = G(k) \oplus (b \cdot \mathsf{r})$, and outputs (C, k).

Examples of Statistically-binding Commitments

- El-Gamal style (assuming DDH in a group $\mathbb{G}$).
  - $\mathsf{pp} = (\mathbb{G}, p, g, h) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, where g, h are generators of $\mathbb{G}$ (of prime order p).
  - For any $m \in \mathbb{G}$, $\mathsf{Com}(\mathsf{pp}, m)$ samples $r \xleftarrow{\$} \mathbb{Z}_p$, sets $C = (g^r, h^r \cdot m)$ and outputs (C, r).
  - $\mathsf{Open}(\mathsf{pp}, m, C, d)$:
    - Parse C as (G, H).
    - Check if $g^R = G$ and $h^R \cdot m = H$.
    - Output 1 if the checks pass; otherwise output 0.

- *Bit-commitments* from any PRG (and hence OWF):
  - $(G) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$: G: $\{0,1\}^\lambda \to \{0,1\}^{\lambda'}$ is a PRG.
  - For any bit $b \in \{0,1\}$, $\mathsf{Com}(G, b)$ does the following.
    - R samples and sends $\mathsf{r} \xleftarrow{\$} \{0,1\}^{\lambda'}$.
    - C samples $k \xleftarrow{\$} \{0,1\}^\lambda$, computes $C = G(k) \oplus (b \cdot \mathsf{r})$, and outputs (C, k).
  - $\mathsf{Open}(G, \mathsf{r}, b, C, d)$: if $C = G(d) \oplus (b \cdot \mathsf{r})$ output 1 else output 0.

ZK for all of NP

Here,

$v_i \in [n]$ for every i.

ZK for all of NP

- We prove that **HAM** has a computational zero-knowledge proof.

Here,

$v_i \in [n]$ for every i.

ZK for all of NP

- We prove that **HAM** has a computational zero-knowledge proof.
  - **HAM** is **NP**-complete, so it suffices to show **HAM** as a ZK proof to show all of **NP** does.

Here,

$v_i \in [n]$ for every i.

ZK for all of NP

- We prove that **HAM** has a computational zero-knowledge proof.
  - **HAM** is **NP**-complete, so it suffices to show **HAM** as a ZK proof to show all of **NP** does.

- Let $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ be a statistically-binding bit commitment.

Here,

$v_i \in [n]$ for every i.

ZK for all of NP

- We prove that **HAM** has a computational zero-knowledge proof.
  - **HAM** is **NP**-complete, so it suffices to show **HAM** as a ZK proof to show all of **NP** does.

- Let $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ be a statistically-binding bit commitment.

$\swarrow |V| = n,\ V \subseteq [n]$

- Recall that $G \in \mathbf{HAM}$ if there exists a cycle $w = (v_1, \ldots, v_n)$ of length n that goes through every vertex of G exactly once. Here, $v_i \in [n]$ for every i.

ZK for all of NP

- We prove that **HAM** has a computational zero-knowledge proof.
  - **HAM** is **NP**-complete, so it suffices to show **HAM** as a ZK proof to show all of **NP** does.

- Let $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ be a statistically-binding bit commitment.

✓ undirected graphs

- Recall that $G \in \mathbf{HAM}$ if there exists a cycle $w = (v_1, \ldots, v_n)$ of length n that goes through every vertex of G exactly once. Here, $v_i \in [n]$ for every i.

- For any G on n vertices, we represent G as a $n \times n$ adjacency matrix $G \in \{0,1\}^{n \times n}$.

$G_{i,j} = 1 \iff (i,j) \in E$

ZK for all of NP

- We prove that **HAM** has a computational zero-knowledge proof.
  - **HAM** is **NP**-complete, so it suffices to show **HAM** as a ZK proof to show all of **NP** does.
- Let $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ be a statistically-binding bit commitment.
- Recall that $G \in \mathbf{HAM}$ if there exists a cycle $w = (v_1, \ldots, v_n)$ of length n that goes through every vertex of G exactly once. Here, $v_i \in [n]$ for every i.
- For any G on n vertices, we represent G as a $n \times n$ adjacency matrix $G \in \{0,1\}^{n \times n}$.
- $\mathsf{Com}(G)$ denotes the $n \times n$ matrix of commitments to each bit of G.

ZK for all of NP

- We prove that **HAM** has a computational zero-knowledge proof.
  - **HAM** is **NP**-complete, so it suffices to show **HAM** as a ZK proof to show all of **NP** does.

- Let $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ be a statistically-binding bit commitment.

- Recall that $G \in \mathbf{HAM}$ if there exists a cycle $w = (v_1, \ldots, v_n)$ of length n that goes through every vertex of G exactly once. Here, $v_i \in [n]$ for every i.

- For any G on n vertices, we represent G as a $n \times n$ adjacency matrix $G \in \{0,1\}^{n \times n}$.

- $\mathsf{Com}(G)$ denotes the $n \times n$ matrix of commitments to each bit of G.
  - $\mathsf{Com}(G)_{i,j} = \mathsf{Com}(\mathsf{pp}, G_{i,j})$ for all $i, j \in [n]$.

$(c_{i,j}, d_{i,j})$

ZK for all of NP

(Gen, Com, Open)

ZK for all of NP

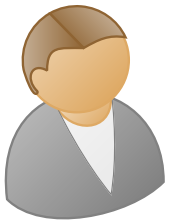
$(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$

$((G, w) \in R_{\mathbf{HAM}}, \mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda))$

ZK for all of NP

$(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$

$((G, w) \in R_{\mathbf{HAM}}, \mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda))$

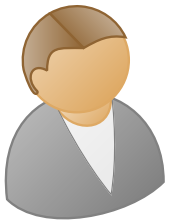


P

ZK for all of NP

$(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$

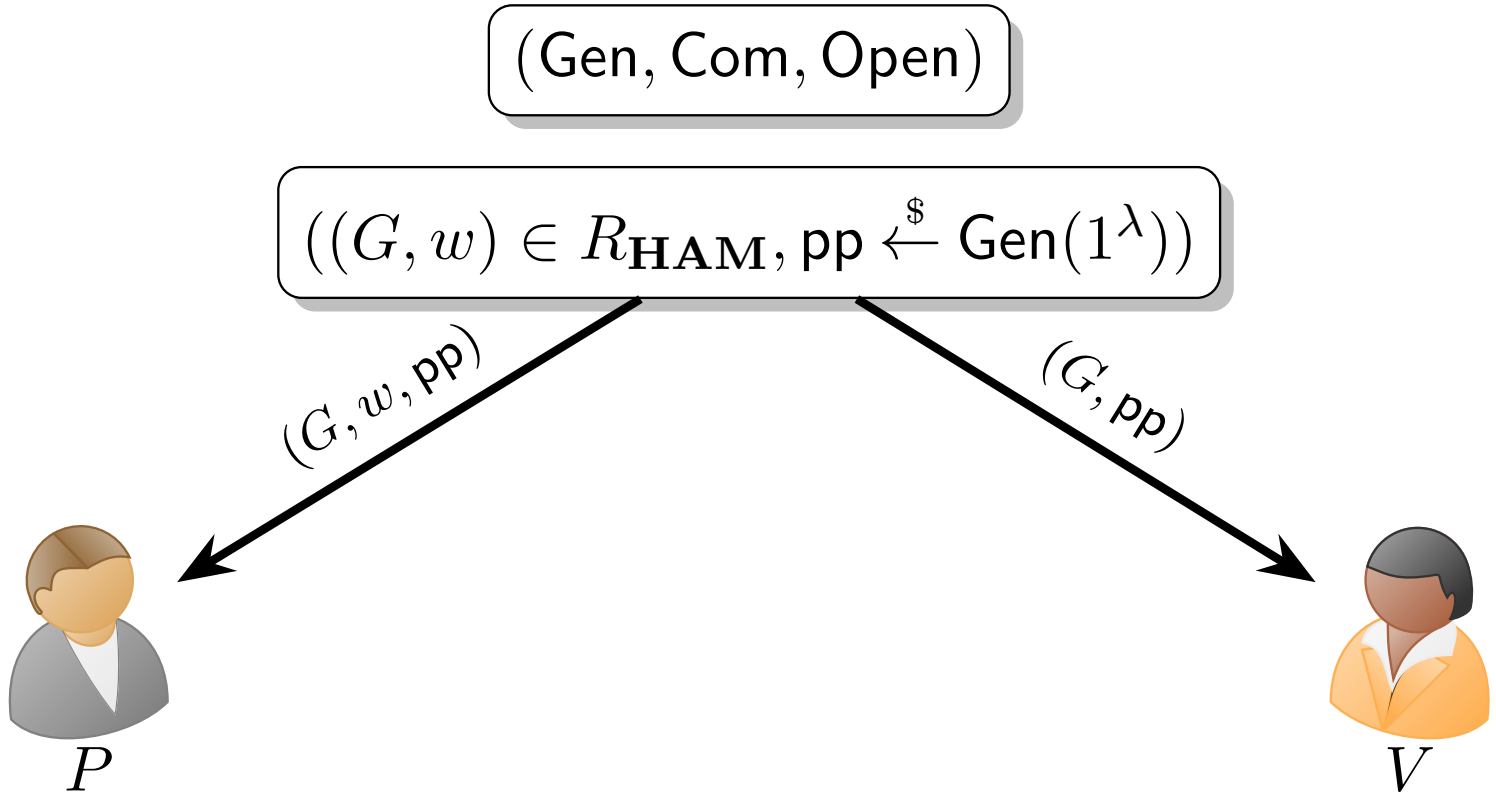
$((G, w) \in R_{\mathbf{HAM}}, \mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda))$



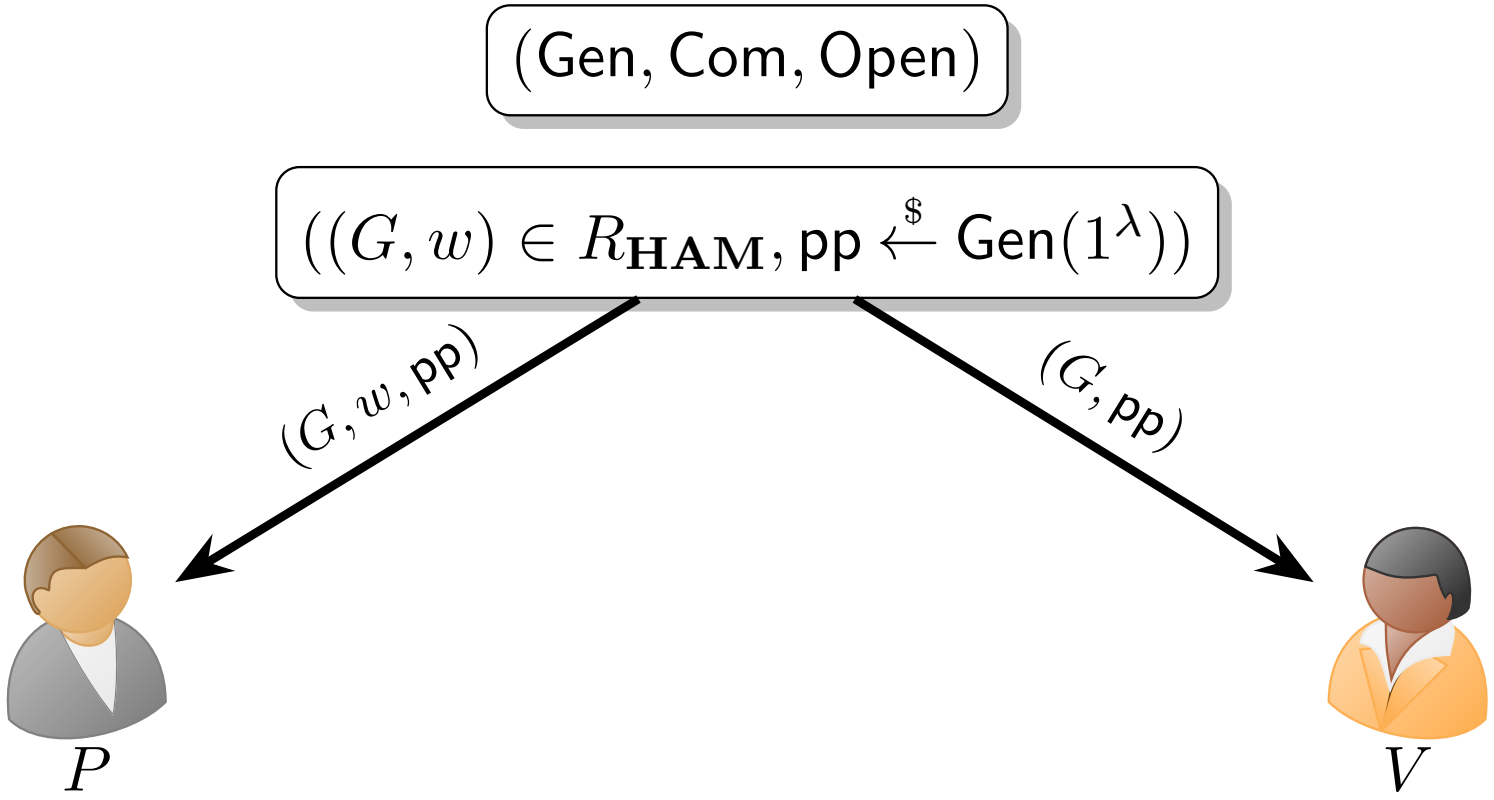
P

V

ZK for all of NP

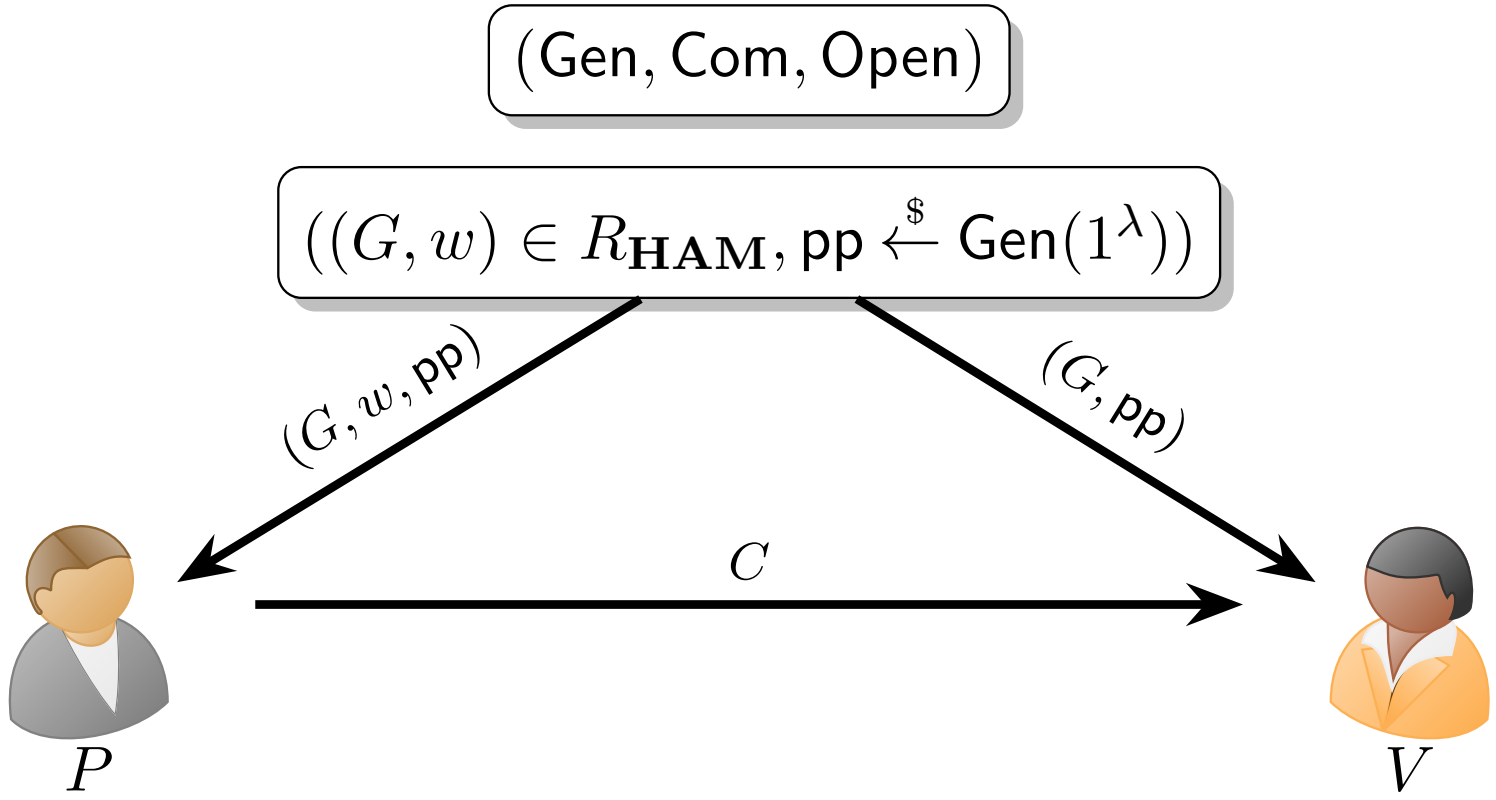


ZK for all of NP



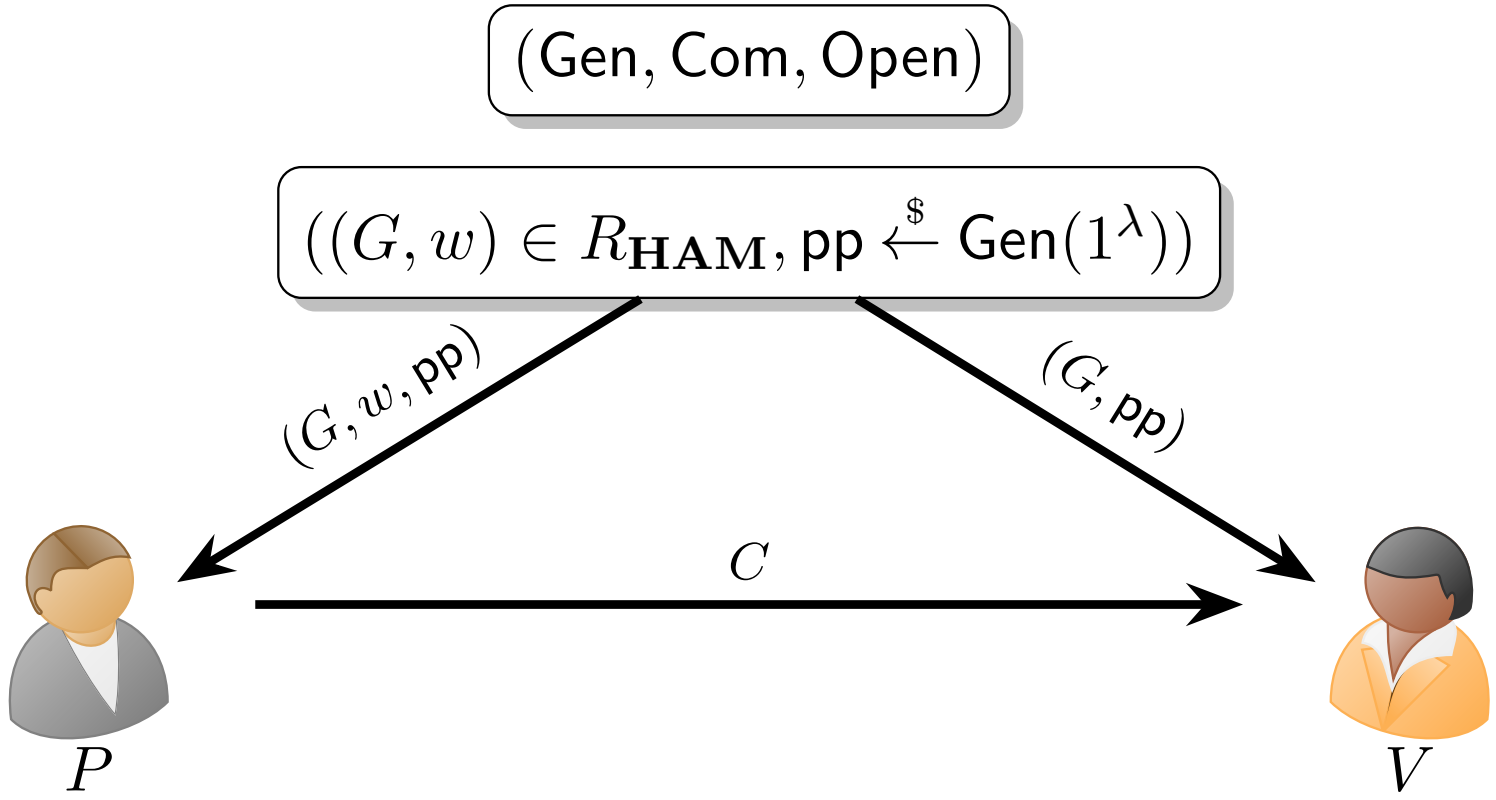
(P1) Sample random permutation ϕ and compute $(C, D) \leftarrow \mathsf{Com}(\phi(G))$

ZK for all of NP



(P1) Sample random permutation ϕ and compute $(C, D) \leftarrow \mathsf{Com}(\phi(G))$

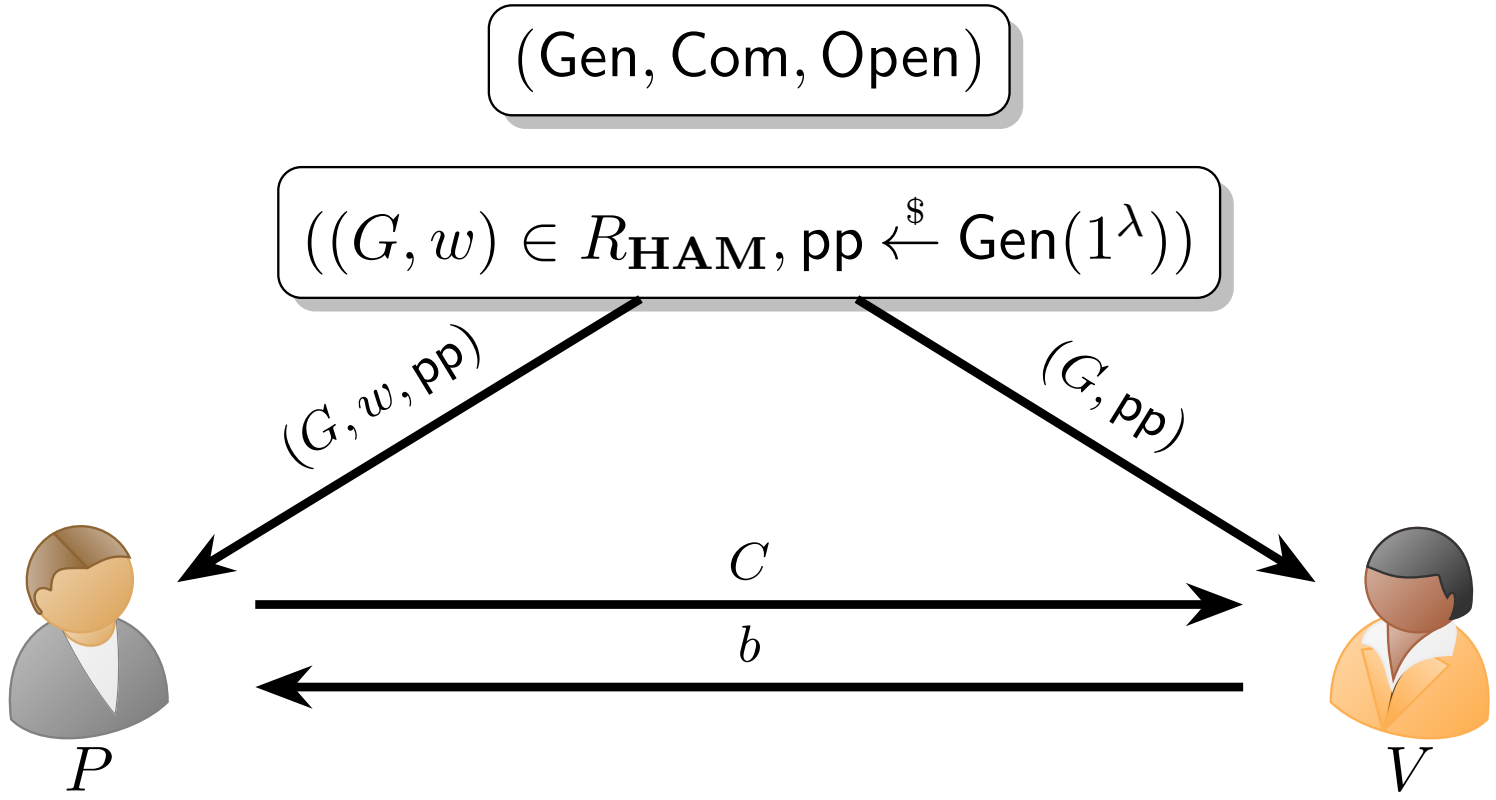
ZK for all of NP



(P1) Sample random permutation ϕ and compute $(C, D) \leftarrow \mathsf{Com}(\phi(G))$

(V1) Sample $b \xleftarrow{\$} \{0, 1\}$.

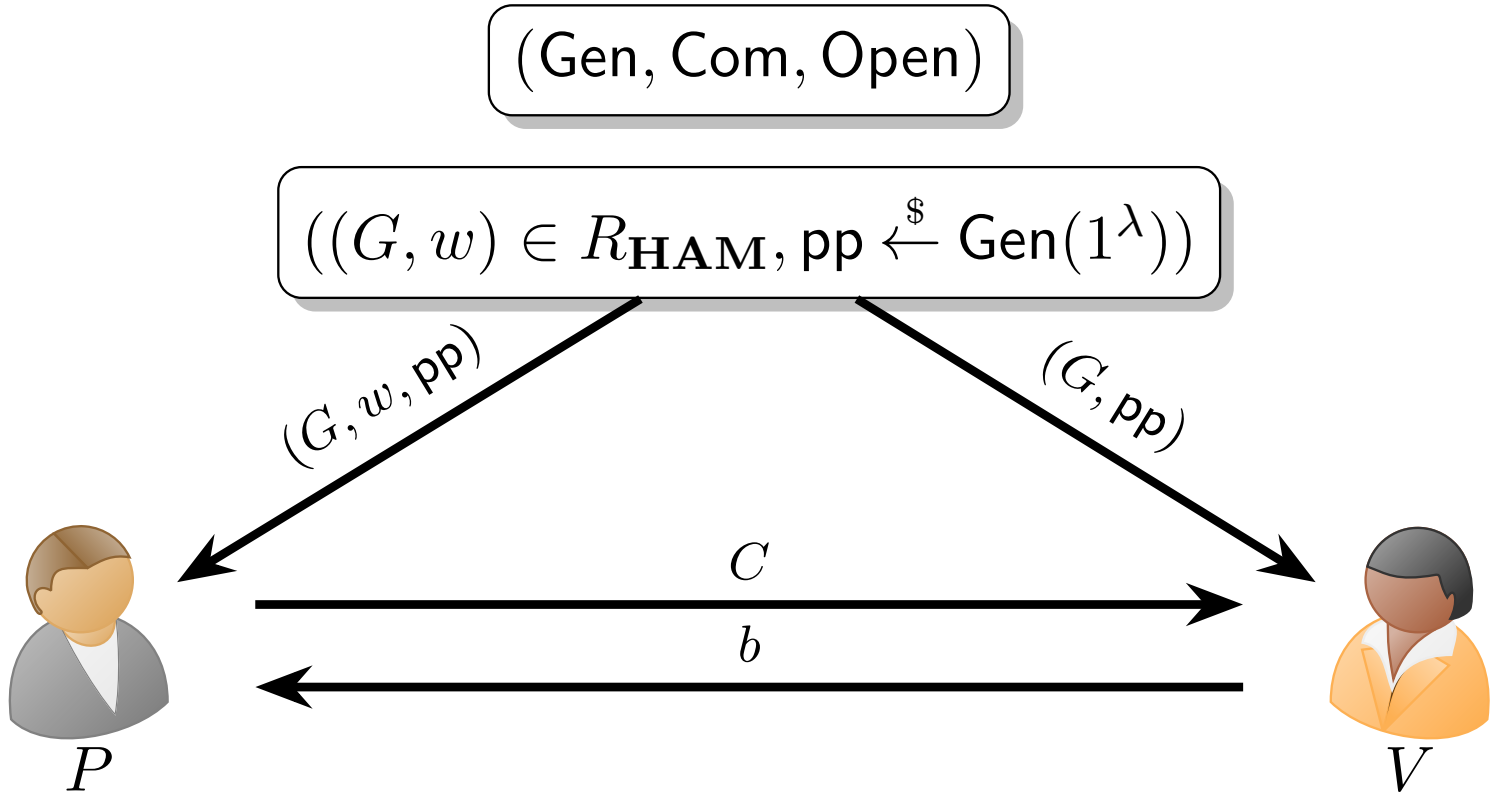
ZK for all of NP



(P1) Sample random permutation ϕ and compute $(C, D) \leftarrow \mathsf{Com}(\phi(G))$

(V1) Sample $b \xleftarrow{\$} \{0, 1\}$.

ZK for all of NP

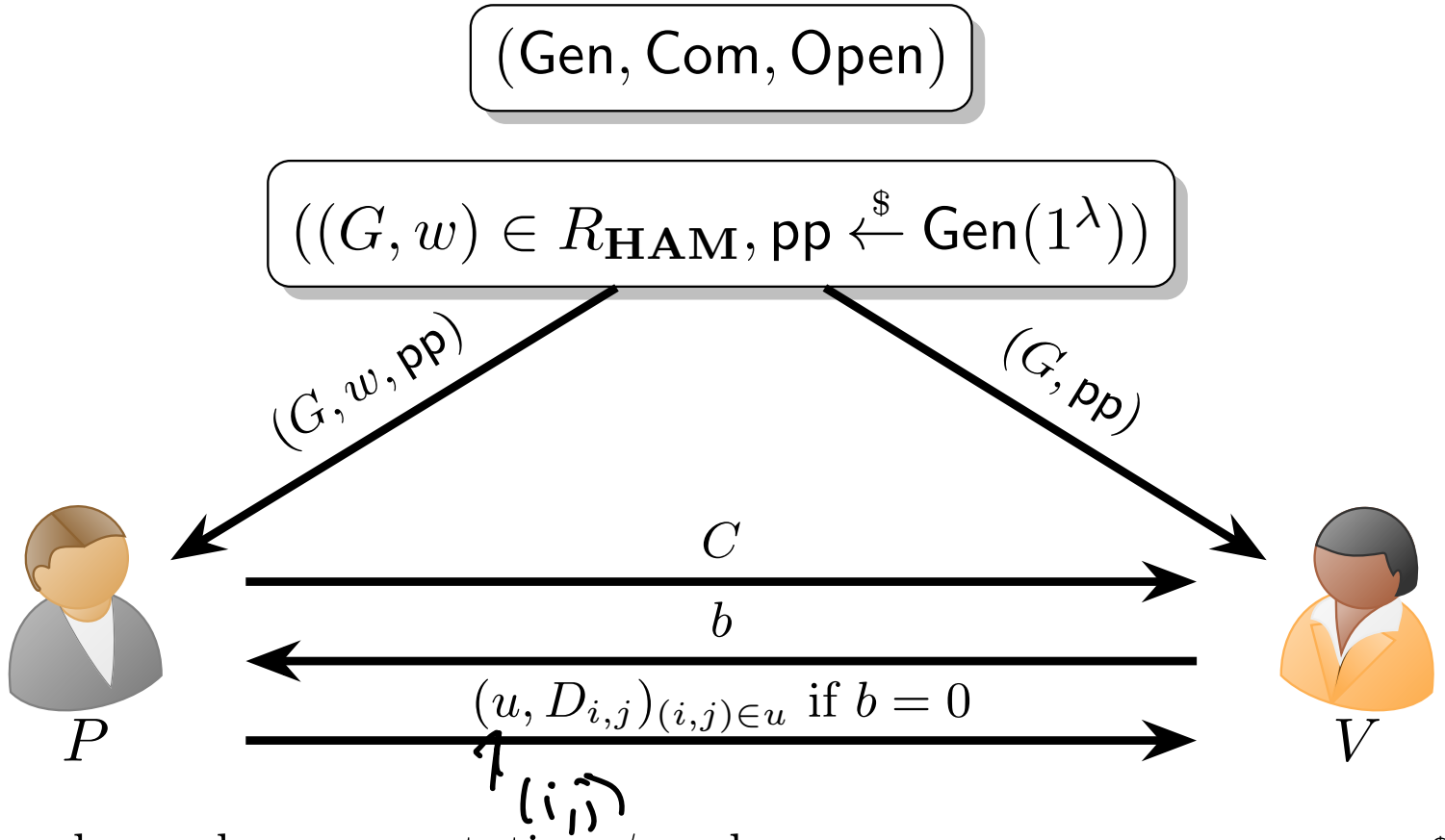


(P1) Sample random permutation ϕ and compute $(C, D) \leftarrow \mathsf{Com}(\phi(G))$

(P2) Set $u = \phi(w)$.

(V1) Sample $b \xleftarrow{\$} \{0, 1\}$.

ZK for all of NP

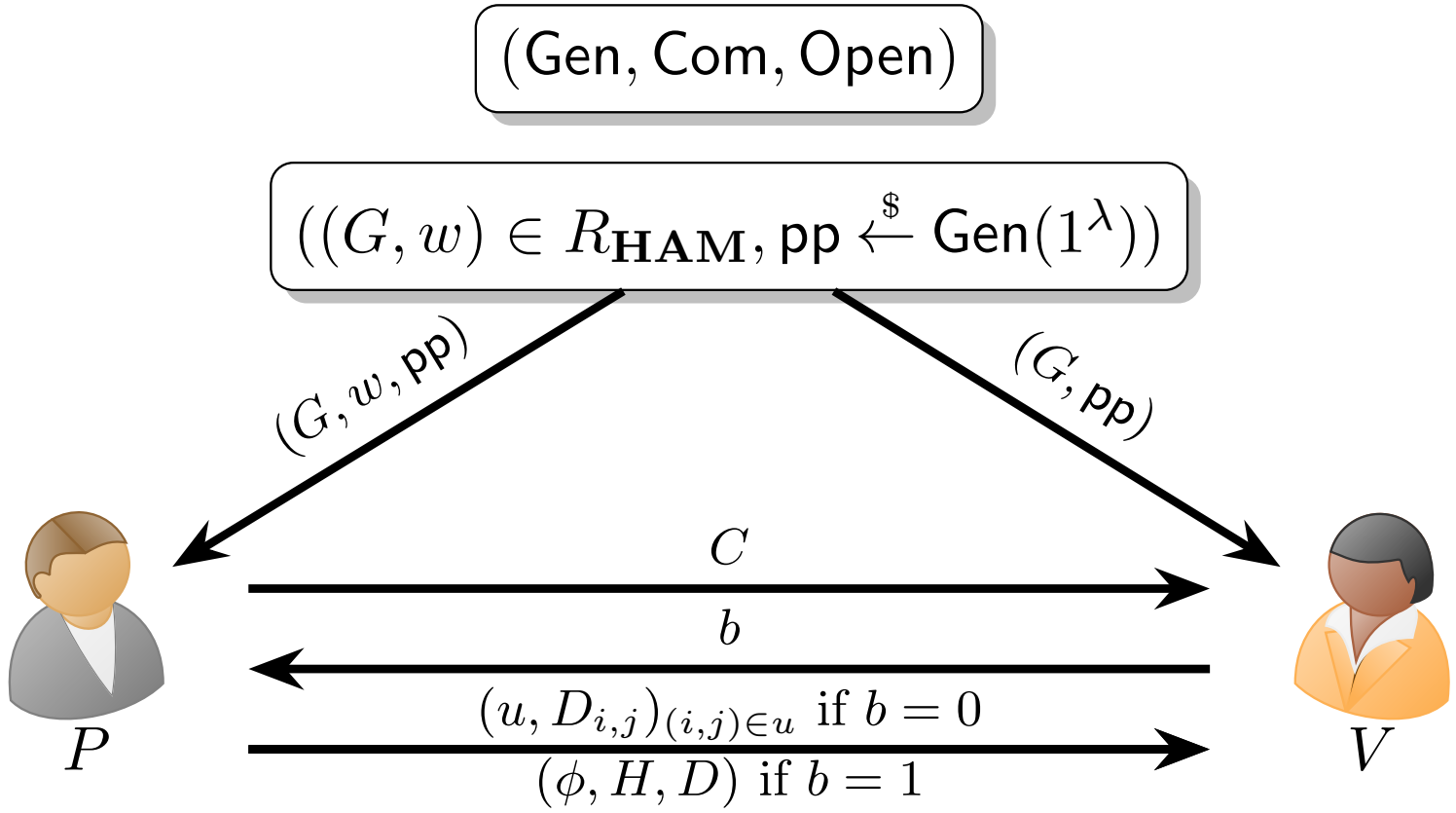


(P1) Sample random permutation ϕ and compute $(C, D) \leftarrow \mathsf{Com}(\phi(G))$

(P2) Set $u = \phi(w)$.

(V1) Sample $b \xleftarrow{\$} \{0, 1\}$.

ZK for all of NP

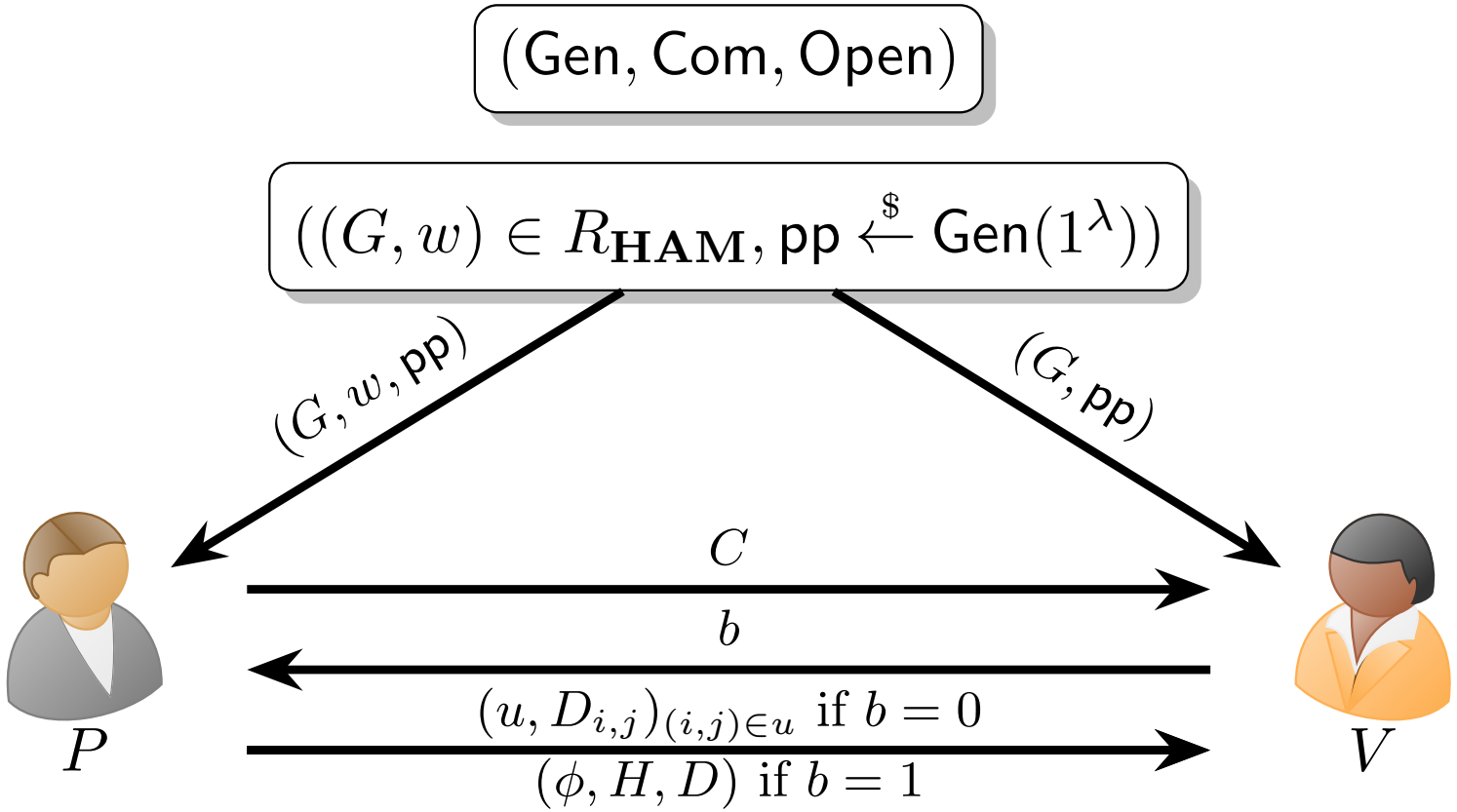


(P1) Sample random permutation ϕ and compute $(C, D) \leftarrow \mathsf{Com}(\phi(G))$

(P2) Set $u = \phi(w)$.

(V1) Sample $b \xleftarrow{\$} \{0, 1\}$.

ZK for all of NP



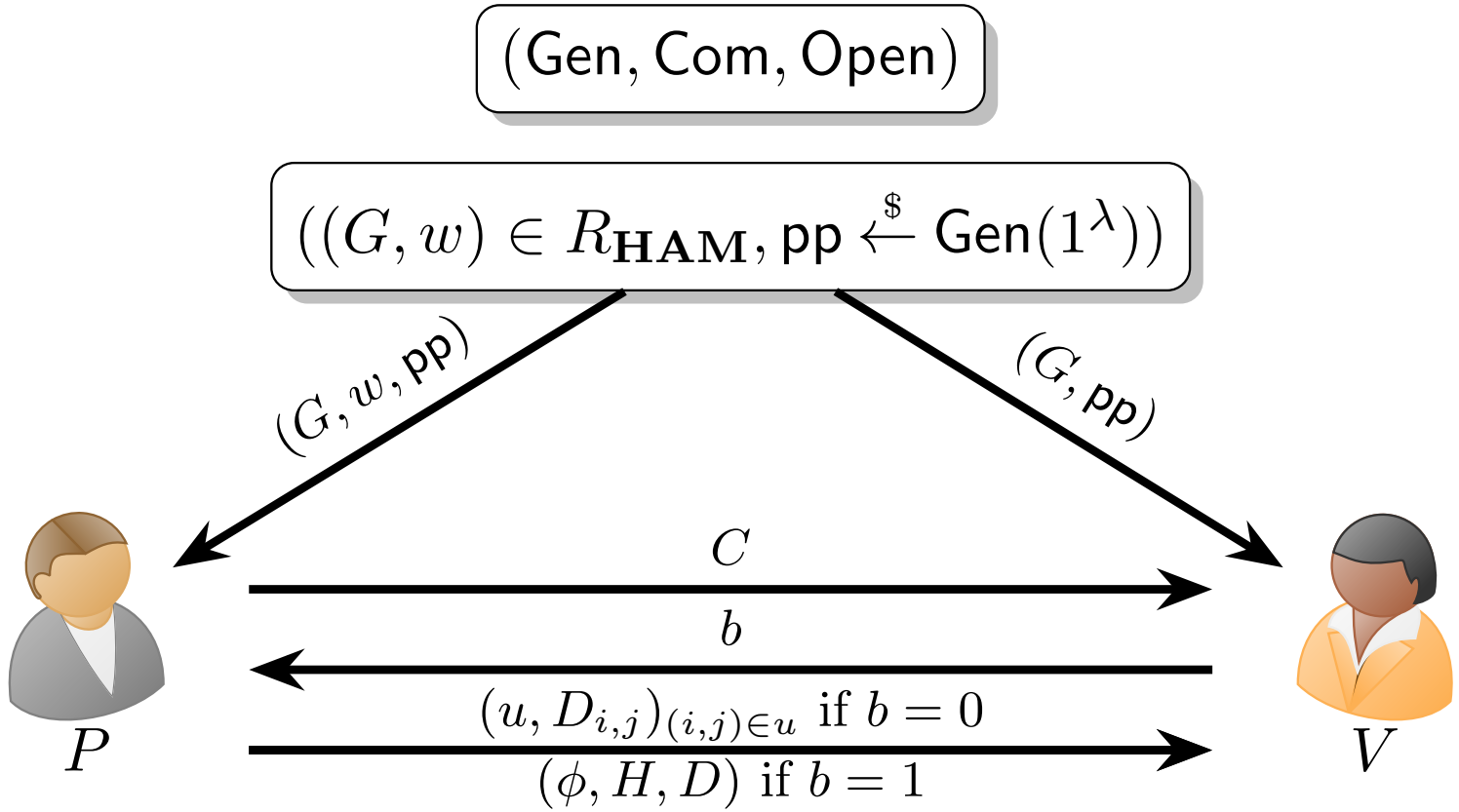
(P1) Sample random permutation ϕ and compute $(C, D) \leftarrow \mathsf{Com}(\phi(G))$

(P2) Set $u = \phi(w)$.

(V1) Sample $b \xleftarrow{\$} \{0, 1\}$.

(V2-0) $b = 0$: check that u is a cycle and $\forall (i, j) \in u$, check $1 = \mathsf{Open}(1, C_{i,j}, D_{i,j})$

ZK for all of NP



(P1) Sample random permutation ϕ and compute $(C, D) \leftarrow \mathsf{Com}(\phi(G))$

(P2) Set $u = \phi(w)$.

(V1) Sample $b \xleftarrow{\$} \{0, 1\}$.

(V2-0) $b = 0$: check that u is a cycle and $\forall (i, j) \in u$, check $1 = \mathsf{Open}(1, C_{i,j}, D_{i,j})$

(V2-1) $b = 1$: check that $H = \phi(G)$ and $1 = \mathsf{Open}(H, C, D)$

ZK for all of NP: Correctness

- Let $\Pi = (P, V)$ denote the protocol on the previous page.

ZK for all of NP: Correctness

- Let $\Pi = (P, V)$ denote the protocol on the previous page.

Lemma 1

Π *has perfect correctness.*

ZK for all of NP: Correctness

- Let $\Pi = (P, V)$ denote the protocol on the previous page.

Lemma 1

Π *has perfect correctness.*

Proof.

Let $(G, w) \in R_{\mathbf{HAM}}$.

ZK for all of NP: Correctness

- Let $\Pi = (P, V)$ denote the protocol on the previous page.

Lemma 1

Π *has perfect correctness.*

Proof.

Let $(G, w) \in R_{\mathbf{HAM}}$. By definition, w is a Hamiltonian cycle in G.

□

ZK for all of NP: Correctness

- Let $\Pi = (P, V)$ denote the protocol on the previous page.

Lemma 1

Π *has perfect correctness.*

Proof.

Let $(G, w) \in R_{\mathbf{HAM}}$. By definition, w is a Hamiltonian cycle in G. Suppose that $b = 0$ in Π.

ZK for all of NP: Correctness

- Let $\Pi = (P, V)$ denote the protocol on the previous page.

Lemma 1

Π *has perfect correctness.*

Proof.

Let $(G, w) \in R_{\mathbf{HAM}}$. By definition, w is a Hamiltonian cycle in G. Suppose that $b = 0$ in Π. Then, $u = \phi(w)$ is a permutation of a cycle in G, and the prover correctly provides decommitments $D_{i,j}$ for every edge of u.

□

ZK for all of NP: Correctness

- Let $\Pi = (P, V)$ denote the protocol on the previous page.

Lemma 1

Π *has perfect correctness.*

Proof.

Let $(G, w) \in R_{\mathbf{HAM}}$. By definition, w is a Hamiltonian cycle in G. Suppose that $b = 0$ in Π. Then, $u = \phi(w)$ is a permutation of a cycle in G, and the prover correctly provides decommitments $D_{i,j}$ for every edge of u. Similarly, for $b = 1$, $H = \phi(G)$ and D is the correct decommitment matrix.

□

ZK for all of NP: Correctness

- Let $\Pi = (P, V)$ denote the protocol on the previous page.

Lemma 1

Π *has perfect correctness.*

Proof.

Let $(G, w) \in R_{\mathbf{HAM}}$. By definition, w is a Hamiltonian cycle in G. Suppose that $b = 0$ in Π. Then, $u = \phi(w)$ is a permutation of a cycle in G, and the prover correctly provides decommitments $D_{i,j}$ for every edge of u. Similarly, for $b = 1$, $H = \phi(G)$ and D is the correct decommitment matrix. In both cases, perfect correctness follows by the correctness of $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$. □

ZK for all of NP: Soundness

Lemma 2

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is statistically binding, then* Π *has soundness error* $\leqslant 1/2$.

ZK for all of NP: Soundness

Lemma 2

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is statistically binding, then* Π *has soundness error* $\leqslant 1/2$.

Proof by Contradiction.

□

ZK for all of NP: Soundness

Lemma 2

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is statistically binding, then* Π *has soundness error* $\leqslant 1/2$.

Proof by Contradiction.

Let $G \notin \mathbf{HAM}$.

□

ZK for all of NP: Soundness

Lemma 2

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is statistically binding, then* Π *has soundness error* $\leqslant 1/2$.

Proof by Contradiction.

Let $G \notin \mathbf{HAM}$. Suppose that $\Pr_b[\langle P^*, V\rangle(G, \mathbf{pp}) = 1] > 1/2$ for any prover strategy P^*.

□

ZK for all of NP: Soundness

Lemma 2

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is statistically binding, then* Π *has soundness error* $\leqslant 1/2$.

Proof by Contradiction.

Let $G \notin \mathbf{HAM}$. Suppose that $\Pr_b[\langle P^*, V\rangle(G, \mathbf{pp}) = 1] > 1/2$ for any prover strategy P^*. This implies that (1) u is a cycle, and (2) $H = \pi(G)$.

□

ZK for all of NP: Soundness

Lemma 2

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is statistically binding, then* Π *has soundness error* $\leqslant 1/2$.

Proof by Contradiction.

Let $G \notin \mathbf{HAM}$. Suppose that $\Pr_b[\langle P^*, V\rangle(G, \mathbf{pp}) = 1] > 1/2$ for any prover strategy P^*. This implies that (1) u is a cycle, and (2) $H = \pi(G)$. Note that if this were *not true*, then the prover only convinces the verifier with probability $\leqslant 1/2$.

□

ZK for all of NP: Soundness

Lemma 2

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is statistically binding, then* Π *has soundness error* $\leqslant 1/2$.

Proof by Contradiction.

Let $G \notin \mathbf{HAM}$. Suppose that $\Pr_b[\langle P^*, V\rangle(G, \mathbf{pp}) = 1] > 1/2$ for any prover strategy P^*. This implies that (1) u is a cycle, and (2) $H = \pi(G)$. Note that if this were *not true*, then the prover only convinces the verifier with probability $\leqslant 1/2$.

However, if u is a cycle and $H = \pi(G)$, this implies that $\pi^{-1}(u)$ is a cycle in G.

□

ZK for all of NP: Soundness

Lemma 2

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is statistically binding, then* Π *has soundness error* $\leqslant 1/2$.

Proof by Contradiction.

Let $G \notin \mathbf{HAM}$. Suppose that $\Pr_b[\langle P^*, V\rangle(G, \mathbf{pp}) = 1] > 1/2$ for any prover strategy P^*. This implies that (1) u is a cycle, and (2) $H = \pi(G)$. Note that if this were *not true*, then the prover only convinces the verifier with probability $\leqslant 1/2$.

However, if u is a cycle and $H = \pi(G)$, this implies that $\pi^{-1}(u)$ is a cycle in G. This violates the assumption that $G \notin \mathbf{HAM}$.

□

ZK for all of NP: Soundness

Lemma 2

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is statistically binding, then* Π *has soundness error* $\leqslant 1/2$.

Proof by Contradiction.

Let $G \notin \mathbf{HAM}$. Suppose that $\Pr_b[\langle P^*, V\rangle(G, \mathbf{pp}) = 1] > 1/2$ for any prover strategy P^*. This implies that (1) u is a cycle, and (2) $H = \pi(G)$. Note that if this were *not true*, then the prover only convinces the verifier with probability $\leqslant 1/2$.

However, if u is a cycle and $H = \pi(G)$, this implies that $\pi^{-1}(u)$ is a cycle in G. This violates the assumption that $G \notin \mathbf{HAM}$. This implies that the soundness error is at most $1/2$. □

$\pi^{-1}(u)$ cannot be a cycle, u cannot be a cycle

ZK for all of NP: CZK

Lemma 3

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* Π *has computational zero-knowledge.*

ZK for all of NP: CZK

Lemma 3

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* Π *has computational zero-knowledge.*

- First, we define the following simulator $S^{V^*}(G, \mathsf{pp})$ for any PPT verifier strategy V^* and $G \in \mathbf{HAM}$.

ZK for all of NP: CZK

Lemma 3

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* Π *has computational zero-knowledge.*

- First, we define the following simulator $S^{V^*}(G, \mathsf{pp})$ for any PPT verifier strategy V^* and $G \in \mathbf{HAM}$.

$S^{V^*}(G, \mathsf{pp})$:

ZK for all of NP: CZK

Lemma 3

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* Π *has computational zero-knowledge.*

- First, we define the following simulator $S^{V^*}(G, \mathsf{pp})$ for any PPT verifier strategy V^* and $G \in \mathbf{HAM}$.

$S^{V^*}(G, \mathsf{pp})$:

1. Set $G_0 = u$, where u is a random cycle of size n.

ZK for all of NP: CZK

Lemma 3

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* Π *has computational zero-knowledge.*

- First, we define the following simulator $S^{V^*}(G, \mathsf{pp})$ for any PPT verifier strategy V^* and $G \in \mathbf{HAM}$.

$S^{V^*}(G, \mathsf{pp})$:

1. Set $G_0 = u$, where u is a random cycle of size n.
2. Set $G_1 = \phi(G)$ for random permutation ϕ.

ZK for all of NP: CZK

Lemma 3

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* Π *has computational zero-knowledge.*

- First, we define the following simulator $S^{V^*}(G, \mathsf{pp})$ for any PPT verifier strategy V^* and $G \in \mathbf{HAM}$.

$S^{V^*}(G, \mathsf{pp})$:

1. Set $G_0 = u$, where u is a random cycle of size n.
2. Set $G_1 = \phi(G)$ for random permutation ϕ.
3. Sample $b \xleftarrow{\$} \{0, 1\}$ and set $(C, D) = \mathsf{Com}(G_b)$.

ZK for all of NP: CZK

Lemma 3

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* Π *has computational zero-knowledge.*

- First, we define the following simulator $S^{V^*}(G, \mathsf{pp})$ for any PPT verifier strategy V^* and $G \in \mathbf{HAM}$.

$S^{V^*}(G, \mathsf{pp})$:

1. Set $G_0 = u$, where u is a random cycle of size n.
2. Set $G_1 = \phi(G)$ for random permutation ϕ.
3. Sample $b \xleftarrow{\$} \{0, 1\}$ and set $(C, D) = \mathsf{Com}(G_b)$.
4. If $V^*(G, \mathsf{pp}, C) = b$, then

ZK for all of NP: CZK

Lemma 3

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* Π *has computational zero-knowledge.*

- First, we define the following simulator $S^{V^*}(G, \mathsf{pp})$ for any PPT verifier strategy V^* and $G \in \mathbf{HAM}$.

$S^{V^*}(G, \mathsf{pp})$:

1. Set $G_0 = u$, where u is a random cycle of size n.
2. Set $G_1 = \phi(G)$ for random permutation ϕ.
3. Sample $b \xleftarrow{\$} \{0, 1\}$ and set $(C, D) = \mathsf{Com}(G_b)$.
4. If $V^*(G, \mathsf{pp}, C) = b$, then
   - Output (C, b, u, D_u) if $b = 0$;

ZK for all of NP: CZK

Lemma 3

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* Π *has computational zero-knowledge.*

- First, we define the following simulator $S^{V^*}(G, \mathsf{pp})$ for any PPT verifier strategy V^* and $G \in \mathbf{HAM}$.

$S^{V^*}(G, \mathsf{pp})$:

1. Set $G_0 = u$, where u is a random cycle of size n.
2. Set $G_1 = \phi(G)$ for random permutation ϕ.
3. Sample $b \xleftarrow{\$} \{0, 1\}$ and set $(C, D) = \mathsf{Com}(G_b)$.
4. If $V^*(G, \mathsf{pp}, C) = b$, then
   - Output (C, b, u, D_u) if $b = 0$;
   - Output $(C, b, (\pi, G_1), D)$ if $b = 1$.

ZK for all of NP: CZK

Lemma 3

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* Π *has computational zero-knowledge.*

- First, we define the following simulator $S^{V^*}(G, \mathsf{pp})$ for any PPT verifier strategy V^* and $G \in \mathbf{HAM}$.

$S^{V^*}(G, \mathsf{pp})$:

1. Set $G_0 = u$, where u is a random cycle of size n.
2. Set $G_1 = \phi(G)$ for random permutation ϕ.
3. Sample $b \xleftarrow{\$} \{0, 1\}$ and set $(C, D) = \mathsf{Com}(G_b)$.
4. If $V^*(G, \mathsf{pp}, C) = b$, then
   - Output (C, b, u, D_u) if $b = 0$;
   - Output $(C, b, (\pi, G_1), D)$ if $b = 1$.
5. Else, goto (1).

ZK for all of NP: CZK

- To establish zero-knowledge, we need two claims.

ZK for all of NP: CZK

- To establish zero-knowledge, we need two claims.

Claim 1

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* S^{V^*} *runs in expected polynomial time.*

ZK for all of NP: CZK

- To establish zero-knowledge, we need two claims.

Claim 1

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* S^{V^*} *runs in expected polynomial time.*

Proof.

□

ZK for all of NP: CZK

- To establish zero-knowledge, we need two claims.

Claim 1

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* S^{V^*} *runs in expected polynomial time.*

Proof.

It suffices to show that $\Pr_{C,b}[V^*(G, \mathsf{pp}, C) = b] \approx 1/2$.

□

ZK for all of NP: CZK

- To establish zero-knowledge, we need two claims.

Claim 1

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* S^{V^*} *runs in expected polynomial time.*

Proof.

It suffices to show that $\Pr_{C,b}[V^*(G, \mathsf{pp}, C) = b] \approx 1/2$. By the computational hiding property of Com, we know that $C_0 = \mathsf{Com}(G_0)$ is computationally indistinguishable from $C_1 = \mathsf{Com}(G_1)$.

□

ZK for all of NP: CZK

- To establish zero-knowledge, we need two claims.

Claim 1

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* S^{V^*} *runs in expected polynomial time.*

Proof.

It suffices to show that $\Pr_{C,b}[V^*(G, \mathsf{pp}, C) = b] \approx 1/2$. By the computational hiding property of Com, we know that $C_0 = \mathsf{Com}(G_0)$ is computationally indistinguishable from $C_1 = \mathsf{Com}(G_1)$. However, if V^* outputs b with probability much smaller than $1/2$, we can use V^* to distinguish between C_0, C_1 in $\mathrm{poly}(\lambda)$ time, which breaks computational hiding since V^* is a PPT algorithm. □

ZK for all of NP: CZK

- To establish zero-knowledge, we need two claims.

Claim 1

If $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is computationally hiding, then* S^{V^*} *runs in expected polynomial time.*

Proof.

It suffices to show that $\Pr_{C,b}[V^*(G, \mathsf{pp}, C) = b] \approx 1/2$. By the computational hiding property of Com, we know that $C_0 = \mathsf{Com}(G_0)$ is computationally indistinguishable from $C_1 = \mathsf{Com}(G_1)$. However, if V^* outputs b with probability much smaller than $1/2$, we can use V^* to distinguish between C_0, C_1 in $\mathrm{poly}(\lambda)$ time, which breaks computational hiding since V^* is a PPT algorithm. □

- This implies that the simulator repeats approximately 2 times in expectation, so it is expected polynomial time.

ZK for all of NP: CZK

Claim 2

If Com *is computationally hiding, then* $S^{V^*}(G, \mathsf{pp}) \approx_c \langle P(w), V^* \rangle (G, \mathsf{pp})$.

ZK for all of NP: CZK

Claim 2

If Com *is computationally hiding, then* $S^{V^*}(G, \mathsf{pp}) \approx_c \langle P(w), V^* \rangle (G, \mathsf{pp})$.

- Proof is more involved and requires a hybrid argument.

ZK for all of NP: CZK

Claim 2

If Com *is computationally hiding, then* $S^{V^*}(G, \mathsf{pp}) \approx_c \langle P(w), V^* \rangle (G, \mathsf{pp})$.

- Proof is more involved and requires a hybrid argument.
- We need the following intermediate simulator H^{V^*}.

ZK for all of NP: CZK

Claim 2

If Com *is computationally hiding, then* $S^{V^*}(G, \mathsf{pp}) \approx_c \langle P(w), V^* \rangle (G, \mathsf{pp})$.

- Proof is more involved and requires a hybrid argument.
- We need the following intermediate simulator H^{V^*}.

$H^{V^*}(G, w, \mathsf{pp})$

HAM cycle that β gets in protocol

ZK for all of NP: CZK

Claim 2

If Com *is computationally hiding, then* $S^{V^*}(G, \mathsf{pp}) \approx_c \langle P(w), V^* \rangle (G, \mathsf{pp})$.

- Proof is more involved and requires a hybrid argument.
- We need the following intermediate simulator H^{V^*}.

$H^{V^*}(G, w, \mathsf{pp})$

1. Set $(C, D) \leftarrow \mathsf{Com}(\phi(G))$ for random permutation ϕ.

ZK for all of NP: CZK

Claim 2

If Com *is computationally hiding, then* $S^{V^*}(G, \mathsf{pp}) \approx_c \langle P(w), V^* \rangle (G, \mathsf{pp})$.

- Proof is more involved and requires a hybrid argument.
- We need the following intermediate simulator H^{V^*}.

$H^{V^*}(G, w, \mathsf{pp})$

1. Set $(C, D) \leftarrow \mathsf{Com}(\phi(G))$ for random permutation ϕ.
2. $b \leftarrow V^*(G, \mathsf{pp}, C)$.

ZK for all of NP: CZK

Claim 2

If Com *is computationally hiding, then* $S^{V^*}(G, \mathsf{pp}) \approx_c \langle P(w), V^* \rangle (G, \mathsf{pp})$.

- Proof is more involved and requires a hybrid argument.
- We need the following intermediate simulator H^{V^*}.

$H^{V^*}(G, w, \mathsf{pp})$

1. Set $(C, D) \leftarrow \mathsf{Com}(\phi(G))$ for random permutation ϕ.
2. $b \leftarrow V^*(G, \mathsf{pp}, C)$.
3. Output $(C, b, \phi(w), D_{\phi(w)})$ if $b = 0$, else output $(C, b, (\phi, \phi(G)), D)$ if $b = 1$.

ZK for all of NP: CZK

Claim 2

If Com *is computationally hiding, then* $S^{V^*}(G, \mathsf{pp}) \approx_c \langle P(w), V^* \rangle (G, \mathsf{pp})$.

- Proof is more involved and requires a hybrid argument.
- We need the following intermediate simulator H^{V^*}.

$H^{V^*}(G, w, \mathsf{pp})$

1. Set $(C, D) \leftarrow \mathsf{Com}(\phi(G))$ for random permutation ϕ.
2. $b \leftarrow V^*(G, \mathsf{pp}, C)$.
3. Output $(C, b, \phi(w), D_{\phi(w)})$ if $b = 0$, else output $(C, b, (\phi, \phi(G)), D)$ if $b = 1$.

- Hybrid argument:

ZK for all of NP: CZK

Claim 2

If Com *is computationally hiding, then* $S^{V^*}(G, \mathsf{pp}) \approx_c \langle P(w), V^* \rangle (G, \mathsf{pp})$.

- Proof is more involved and requires a hybrid argument.
- We need the following intermediate simulator H^{V^*}.

$H^{V^*}(G, w, \mathsf{pp})$

1. Set $(C, D) \leftarrow \mathsf{Com}(\phi(G))$ for random permutation ϕ.
2. $b \leftarrow V^*(G, \mathsf{pp}, C)$.
3. Output $(C, b, \phi(w), D_{\phi(w)})$ if $b = 0$, else output $(C, b, (\phi, \phi(G)), D)$ if $b = 1$.

- Hybrid argument:

$$S^{V^*}(G, \mathsf{pp}) \approx_c H^{V^*}(G, w, \mathsf{pp}) \equiv \langle P(w), V^* \rangle (G, \mathsf{pp}).$$

ZK for all of NP: CZK

- The righthand side of the above hybrid argument follows by definition of H^{V^*} and the protocol Π.

ZK for all of NP: CZK

- The righthand side of the above hybrid argument follows by definition of H^{V^*} and the protocol Π.

- To show the lefthand side, we recall that $\mathsf{Com}(G_0) \approx_c \mathsf{Com}(G_1)$ *even if* G, w, ϕ *are all known.*

ZK for all of NP: CZK

- The righthand side of the above hybrid argument follows by definition of H^{V^*} and the protocol Π.

- To show the lefthand side, we recall that $\mathsf{Com}(G_0) \approx_c \mathsf{Com}(G_1)$ *even if* G, w, ϕ *are all known.*
- If $b = 0$:

ZK for all of NP: CZK

- The righthand side of the above hybrid argument follows by definition of H^{V^*} and the protocol Π.

- To show the lefthand side, we recall that $\mathsf{Com}(G_0) \approx_c \mathsf{Com}(G_1)$ *even if* G, w, ϕ *are all known.*
- If $b = 0$:
  - u is a random cycle on n vertices (from S), but $\phi(w)$ is also a random cycle on n vertices.

ZK for all of NP: CZK

- The righthand side of the above hybrid argument follows by definition of H^{V^*} and the protocol Π.

- To show the lefthand side, we recall that $\mathsf{Com}(G_0) \approx_c \mathsf{Com}(G_1)$ *even if* G, w, ϕ *are all known.*
- If $b = 0$:
  - u is a random cycle on n vertices (from S), but $\phi(w)$ is also a random cycle on n vertices.
  - $\mathsf{Com}(u) \approx_c \mathsf{Com}(\phi(w))$ even if G, w, ϕ are known.

ZK for all of NP: CZK

- The righthand side of the above hybrid argument follows by definition of H^{V^*} and the protocol Π.

- To show the lefthand side, we recall that $\mathsf{Com}(G_0) \approx_c \mathsf{Com}(G_1)$ *even if* G, w, ϕ *are all known.*
- If $b = 0$:
  - u is a random cycle on n vertices (from S), but $\phi(w)$ is also a random cycle on n vertices.
  - $\mathsf{Com}(u) \approx_c \mathsf{Com}(\phi(w))$ even if G, w, ϕ are known.
- If $b = 1$

ZK for all of NP: CZK

- The righthand side of the above hybrid argument follows by definition of H^{V^*} and the protocol Π.

- To show the lefthand side, we recall that $\mathsf{Com}(G_0) \approx_c \mathsf{Com}(G_1)$ *even if* G, w, ϕ *are all known.*
- If $b = 0$:
  - u is a random cycle on n vertices (from S), but $\phi(w)$ is also a random cycle on n vertices.
  - $\mathsf{Com}(u) \approx_c \mathsf{Com}(\phi(w))$ even if G, w, ϕ are known.
- If $b = 1$
  - The output distribution of both simulators is identical in this case.

ZK for all of NP: CZK

- The righthand side of the above hybrid argument follows by definition of H^{V^*} and the protocol Π.

- To show the lefthand side, we recall that $\mathsf{Com}(G_0) \approx_c \mathsf{Com}(G_1)$ *even if* G, w, ϕ *are all known.*
- If $b = 0$:
  - u is a random cycle on n vertices (from S), but $\phi(w)$ is also a random cycle on n vertices.
  - $\mathsf{Com}(u) \approx_c \mathsf{Com}(\phi(w))$ even if G, w, ϕ are known.
- If $b = 1$
  - The output distribution of both simulators is identical in this case.

- Therefore $S^{V^*}(G, \mathsf{pp}) \approx_c H^{V^*}(G, w, \mathsf{pp})$.

ZK for all of NP: CZK

- The righthand side of the above hybrid argument follows by definition of H^{V^*} and the protocol Π.

- To show the lefthand side, we recall that $\mathsf{Com}(G_0) \approx_c \mathsf{Com}(G_1)$ *even if* G, w, ϕ *are all known.*
- If $b = 0$:
  - u is a random cycle on n vertices (from S), but $\phi(w)$ is also a random cycle on n vertices.
  - $\mathsf{Com}(u) \approx_c \mathsf{Com}(\phi(w))$ even if G, w, ϕ are known.
- If $b = 1$
  - The output distribution of both simulators is identical in this case.

- Therefore $S^{V^*}(G, \mathsf{pp}) \approx_c H^{V^*}(G, w, \mathsf{pp})$.

- Thus, Π has computational zero-knowledge! □

Alternate Proof: 3-coloring

- Goldreich, Micali, and Widgerson showed that **NP** $\subseteq$ **CZK** by showing that the problem of 3-coloring on a graph was in **CZK**.

Alternate Proof: 3-coloring

- Goldreich, Micali, and Widgerson showed that **NP** $\subseteq$ **CZK** by showing that the problem of 3-coloring on a graph was in **CZK**.

- $\mathsf{3Col} = \{G = (V, E) \colon G \text{ is 3-colorable}\}$.

Alternate Proof: 3-coloring

- Goldreich, Micali, and Widgerson showed that **NP** $\subseteq$ **CZK** by showing that the problem of 3-coloring on a graph was in **CZK**.

- $\mathsf{3Col} = \{G = (V, E) : G \text{ is 3-colorable}\}$.
  - $\exists \mathsf{col} : V \to \{0, 1, 2\}$ such that $\forall (u, v) \in E$, $\mathsf{col}(u) \neq \mathsf{col}(v)$.

Alternate Proof: 3-coloring

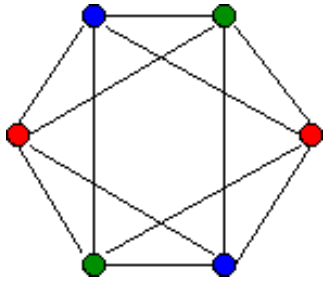
- Goldreich, Micali, and Widgerson showed that **NP** $\subseteq$ **CZK** by showing that the problem of 3-coloring on a graph was in **CZK**.

- $\mathsf{3Col} = \{G = (V, E) : G \text{ is 3-colorable}\}$.
  - $\exists \mathsf{col} : V \to \{0, 1, 2\}$ such that $\forall (u, v) \in E$, $\mathsf{col}(u) \neq \mathsf{col}(v)$.
  - Below is an example from Wikipedia: https://commons.wikimedia.org/w/index.php?curid=614041.

Alternate Proof: 3-coloring

- Goldreich, Micali, and Widgerson showed that **NP** $\subseteq$ **CZK** by showing that the problem of 3-coloring on a graph was in **CZK**.

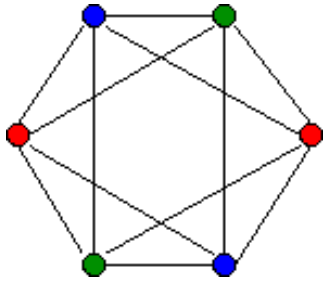
- $\mathsf{3Col} = \{G = (V, E) : G \text{ is 3-colorable}\}$.
  - $\exists \mathsf{col} : V \to \{0, 1, 2\}$ such that $\forall (u, v) \in E$, $\mathsf{col}(u) \neq \mathsf{col}(v)$.
  - Below is an example from Wikipedia: https://commons.wikimedia.org/w/index.php?curid=614041.



Alternate Proof: 3-coloring

- Goldreich, Micali, and Widgerson showed that **NP** $\subseteq$ **CZK** by showing that the problem of 3-coloring on a graph was in **CZK**.

- $\mathsf{3Col} = \{G = (V, E) : G \text{ is 3-colorable}\}$.
  - $\exists \mathsf{col} : V \to \{0, 1, 2\}$ such that $\forall (u, v) \in E$, $\mathsf{col}(u) \neq \mathsf{col}(v)$.
  - Below is an example from Wikipedia: https://commons.wikimedia.org/w/index.php?curid=614041.

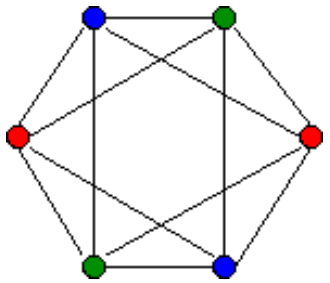


- Notation: permutation of coloring col.

Alternate Proof: 3-coloring

- Goldreich, Micali, and Widgerson showed that **NP** $\subseteq$ **CZK** by showing that the problem of 3-coloring on a graph was in **CZK**.

- $\mathsf{3Col} = \{G = (V, E) : G \text{ is 3-colorable}\}$.
  - $\exists \mathsf{col} : V \to \{0, 1, 2\}$ such that $\forall (u, v) \in E$, $\mathsf{col}(u) \neq \mathsf{col}(v)$.
  - Below is an example from Wikipedia: https://commons.wikimedia.org/w/index.php?curid=614041.

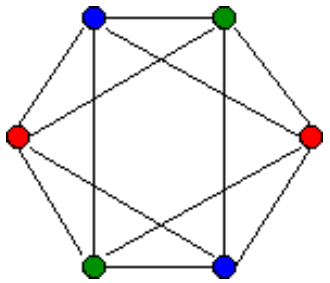


- Notation: permutation of coloring col.
  - Permutation $\pi : \{0, 1, 2\} \to \{0, 1, 2\}$.

Alternate Proof: 3-coloring

- Goldreich, Micali, and Widgerson showed that **NP** $\subseteq$ **CZK** by showing that the problem of 3-coloring on a graph was in **CZK**.

- $\mathsf{3Col} = \{G = (V, E) : G \text{ is 3-colorable}\}$.
  - $\exists \mathsf{col} : V \to \{0, 1, 2\}$ such that $\forall (u, v) \in E$, $\mathsf{col}(u) \neq \mathsf{col}(v)$.
  - Below is an example from Wikipedia: https://commons.wikimedia.org/w/index.php?curid=614041.



- Notation: permutation of coloring col.
  - Permutation $\pi : \{0, 1, 2\} \to \{0, 1, 2\}$.
  - Permuted coloring $\mathsf{col}' = \pi \circ \mathsf{col}$, i.e., $\mathsf{col}'(v) = \pi(\mathsf{col}(v))$.

Protocol for 3Col

(Gen, Com, Open)

Protocol for 3Col

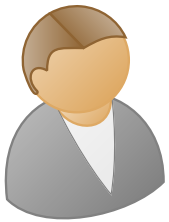
$(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$

$((G, \mathsf{col}) \in R_{\mathsf{3Col}}, \mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda))$

Protocol for 3Col

$(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$

$((G, \mathsf{col}) \in R_{\mathsf{3Col}}, \mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda))$

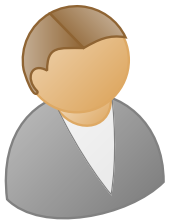


P

Protocol for 3Col

$(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$

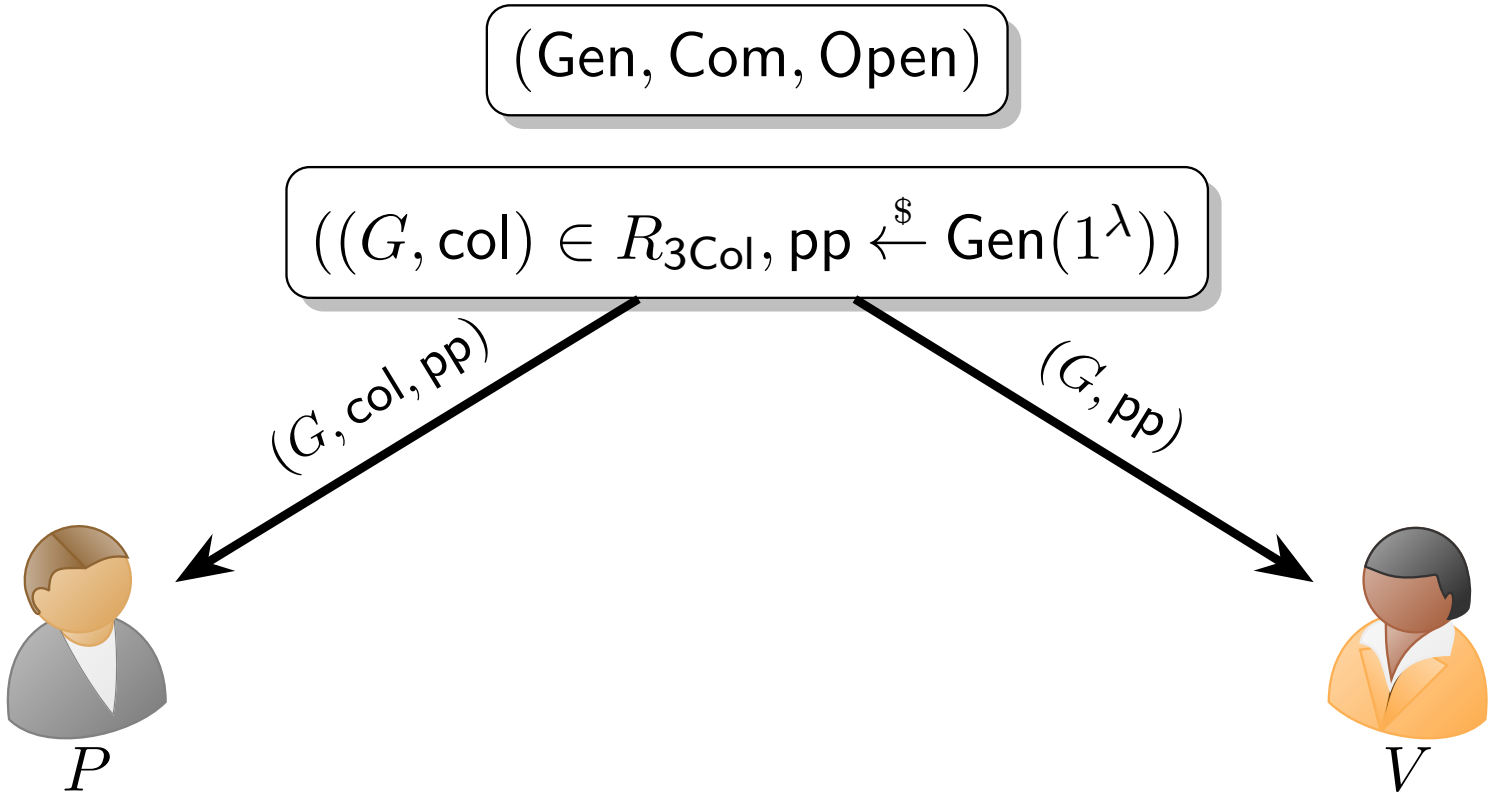
$((G, \mathsf{col}) \in R_{\mathsf{3Col}}, \mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda))$



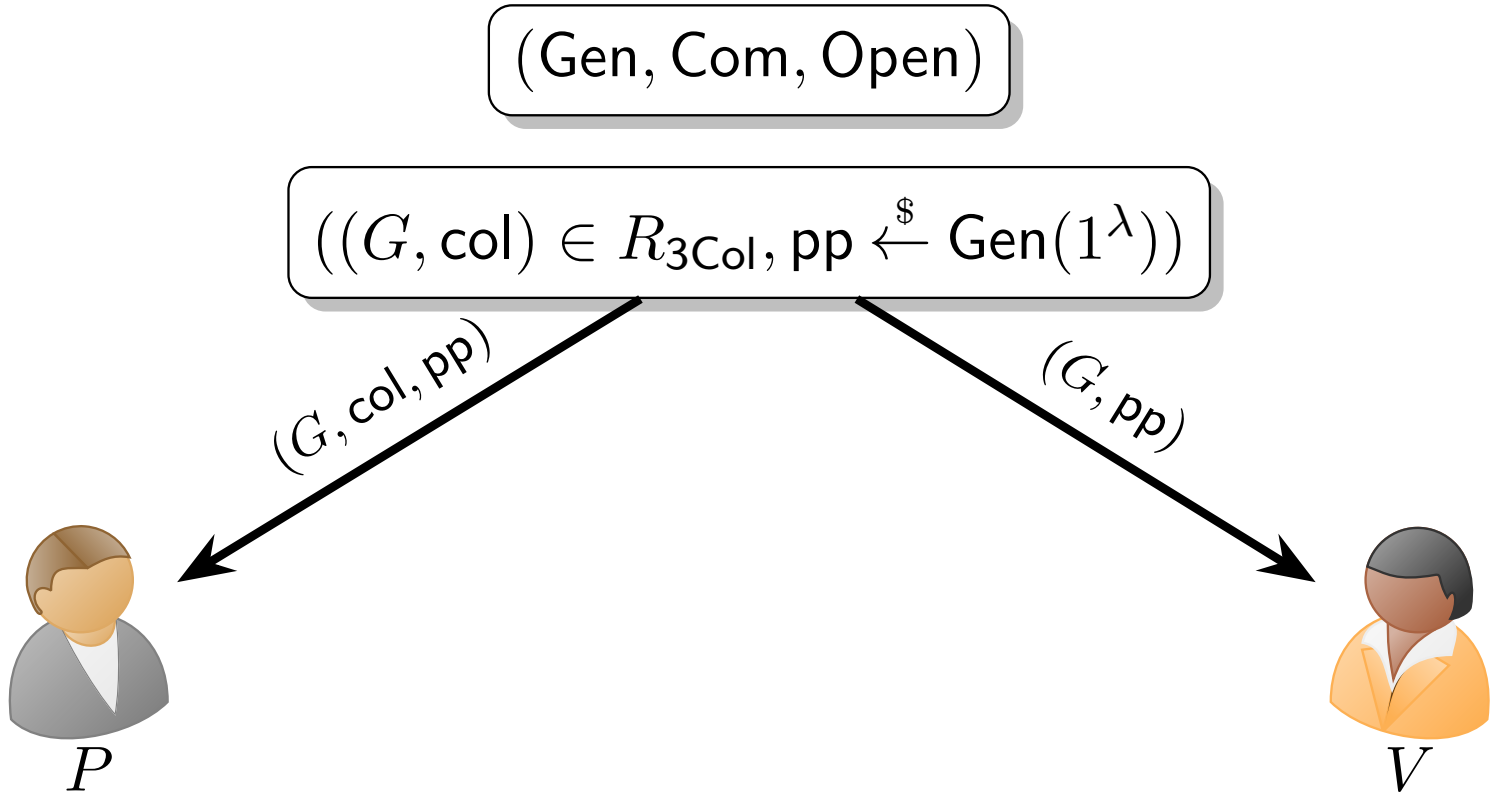
P

V

Protocol for 3Col

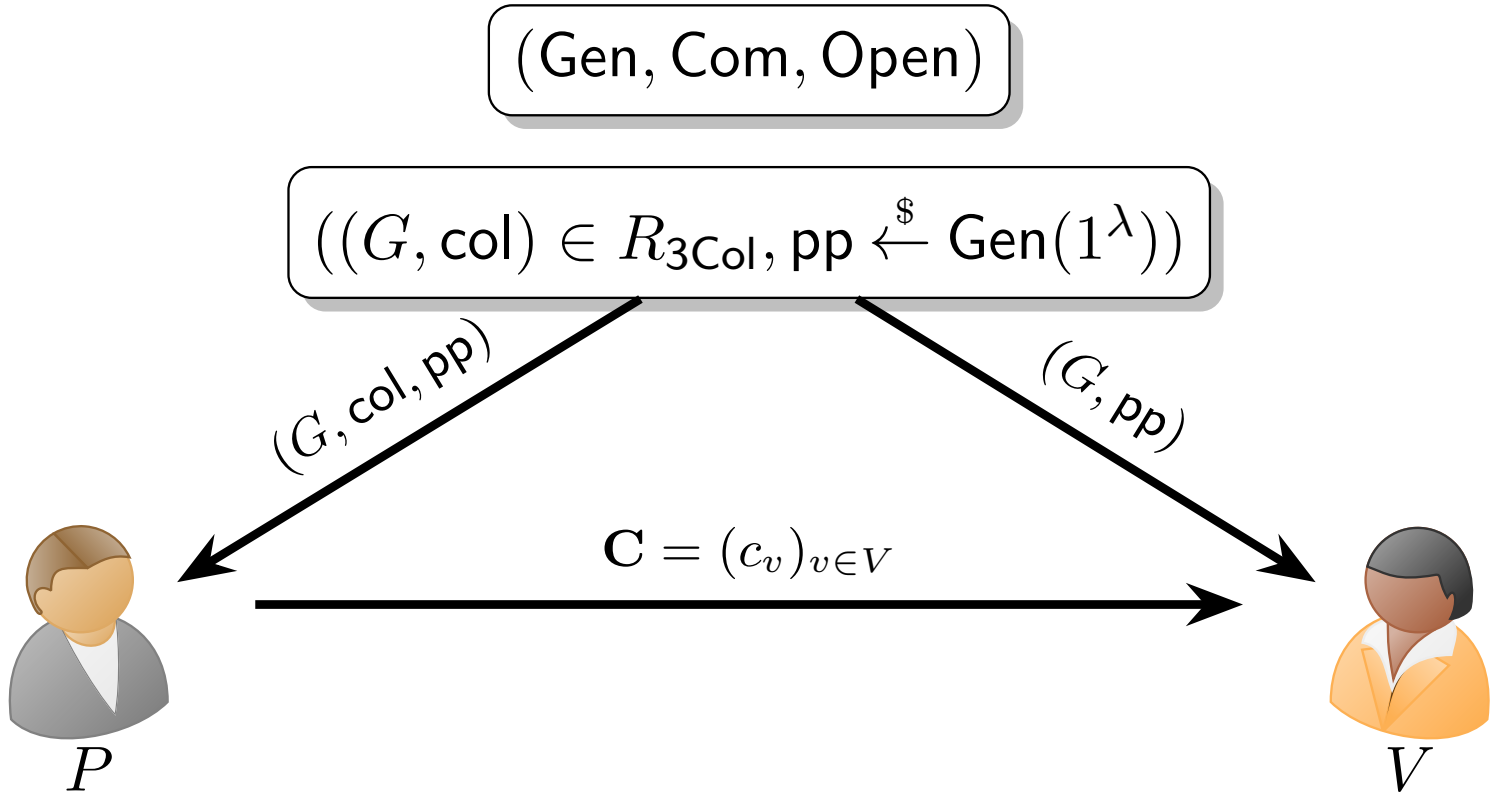


Protocol for 3Col



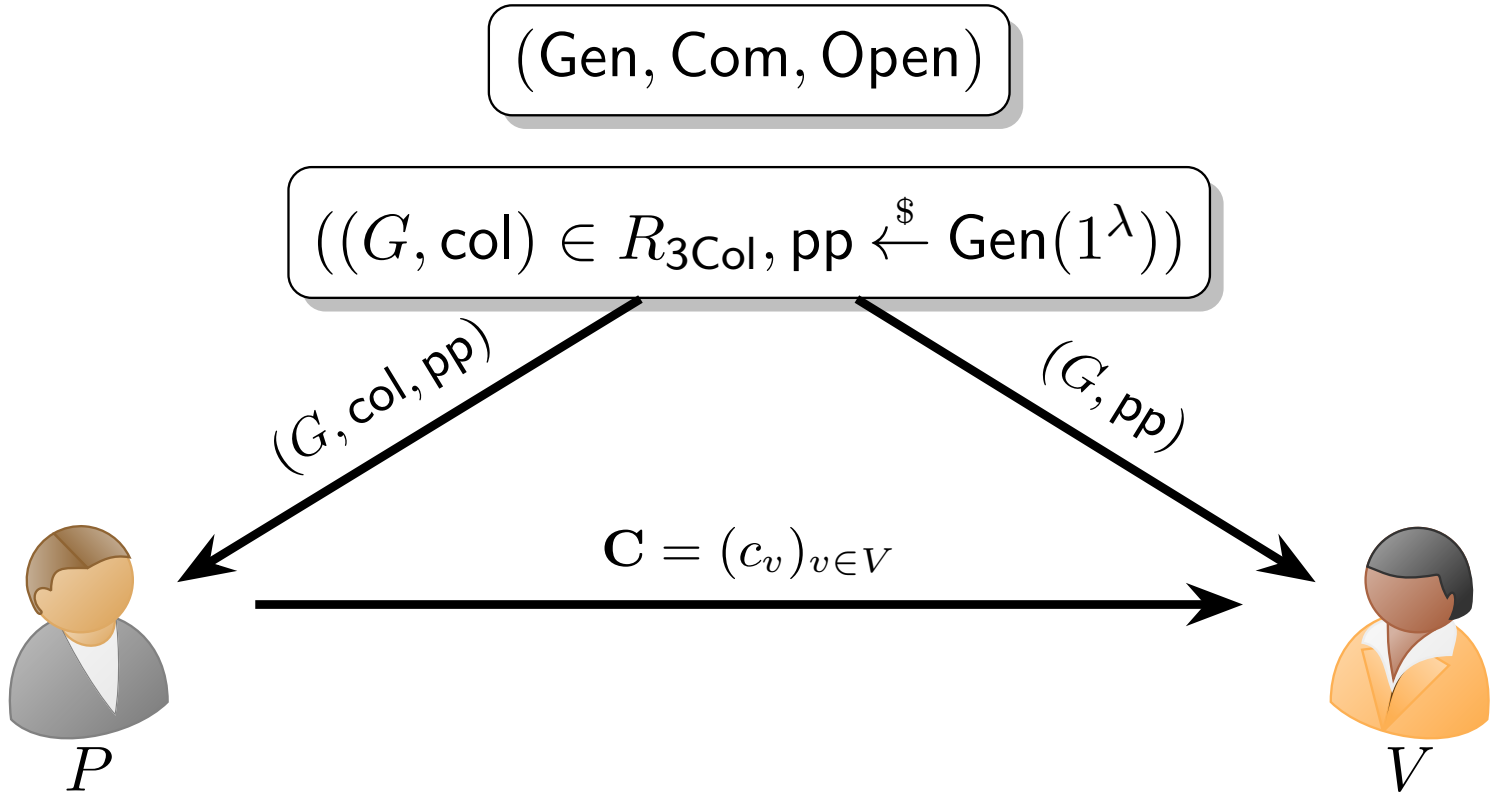
(P1) Sample random permutation π and compute $(c_v, d_v) \leftarrow \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(v))$, $\forall v \in V$

Protocol for 3Col



(P1) Sample random permutation π and compute $(c_v, d_v) \leftarrow \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(v))$, $\forall v \in V$

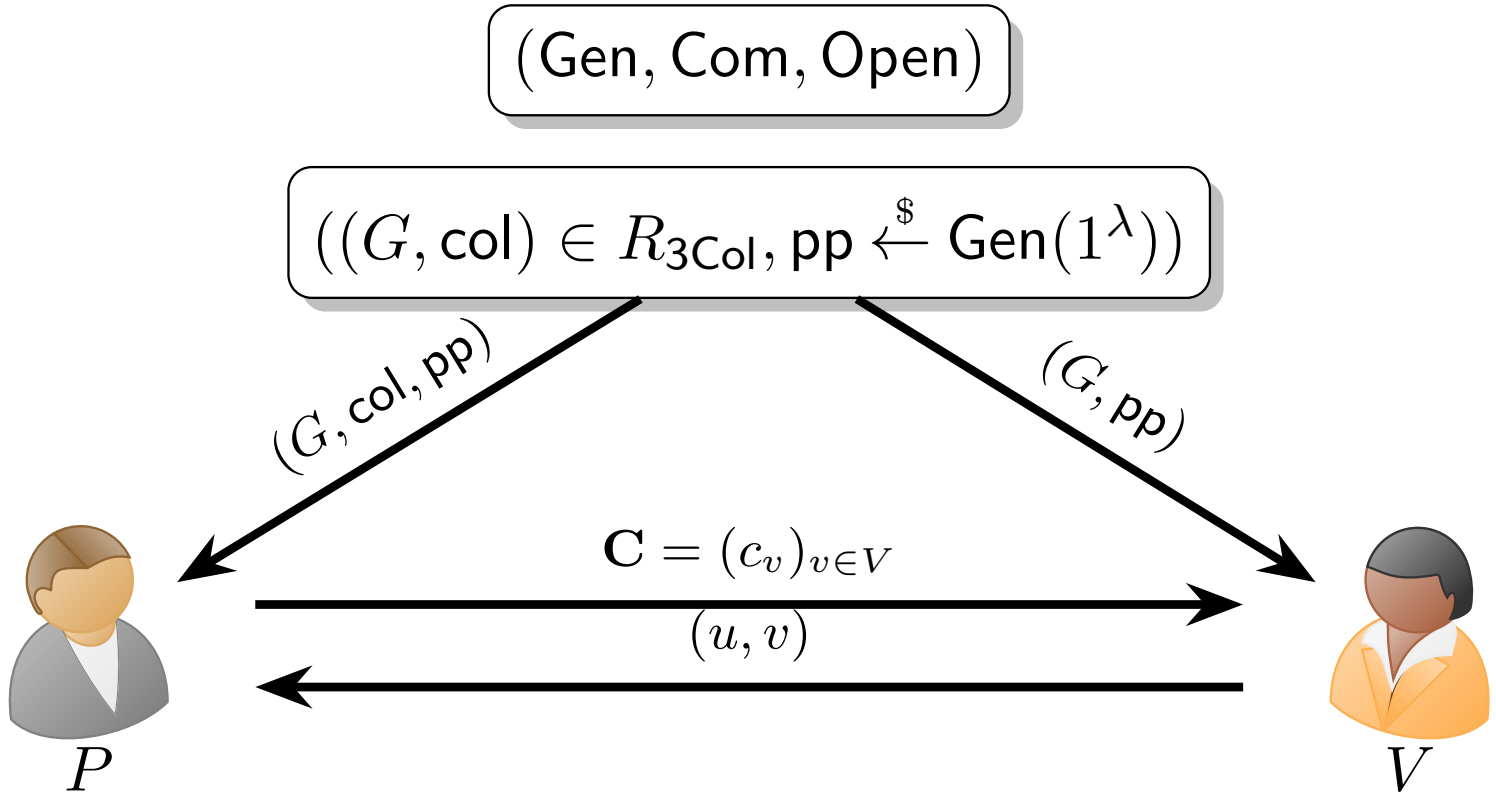
Protocol for 3Col



(P1) Sample random permutation π and compute $(c_v, d_v) \leftarrow \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(v))$, $\forall v \in V$

(V1) Sample $(u, v) \xleftarrow{\$} E$.

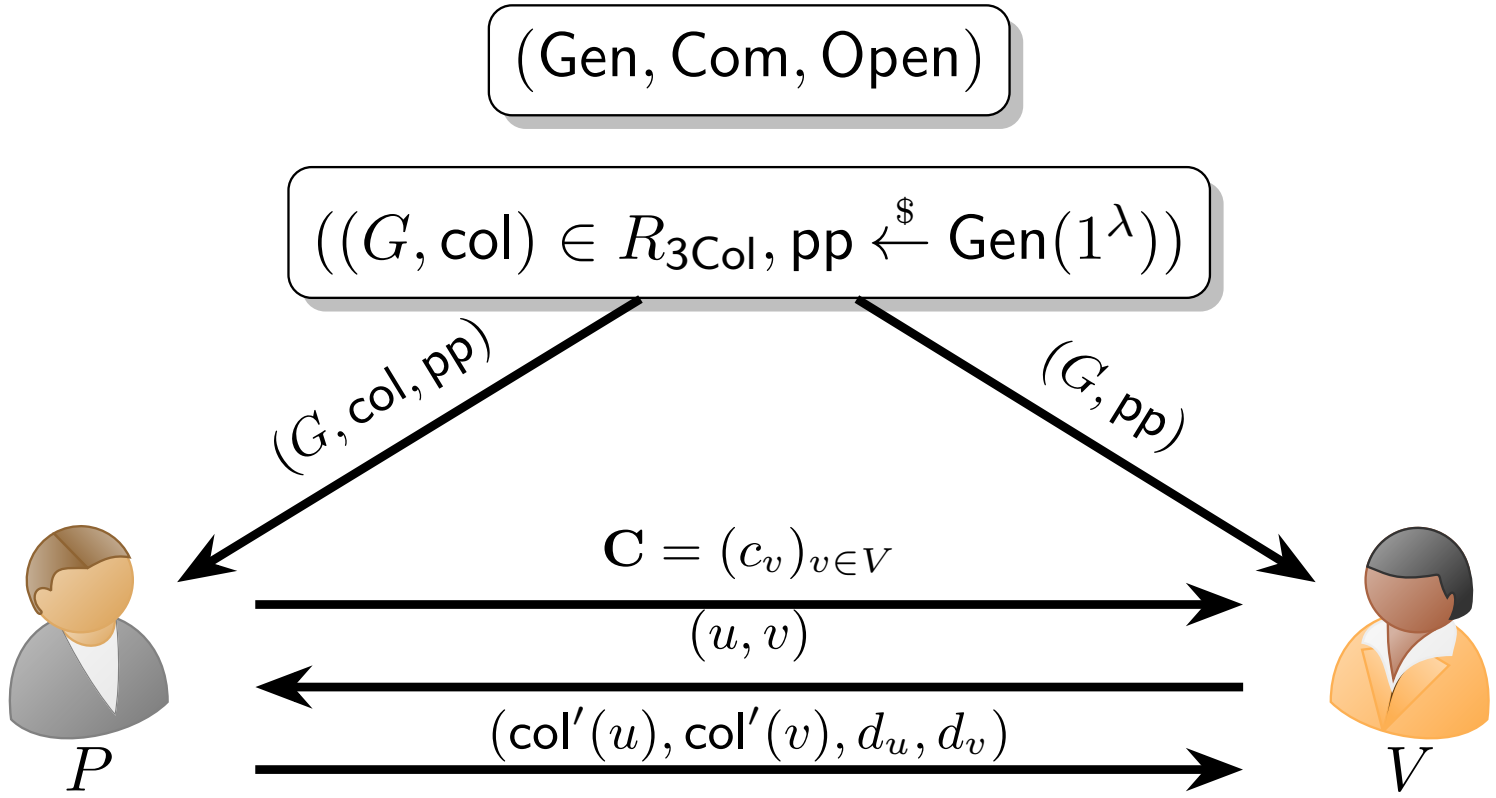
Protocol for 3Col



(P1) Sample random permutation π and compute $(c_v, d_v) \leftarrow \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(v))$, $\forall v \in V$

(V1) Sample $(u, v) \xleftarrow{\$} E$.

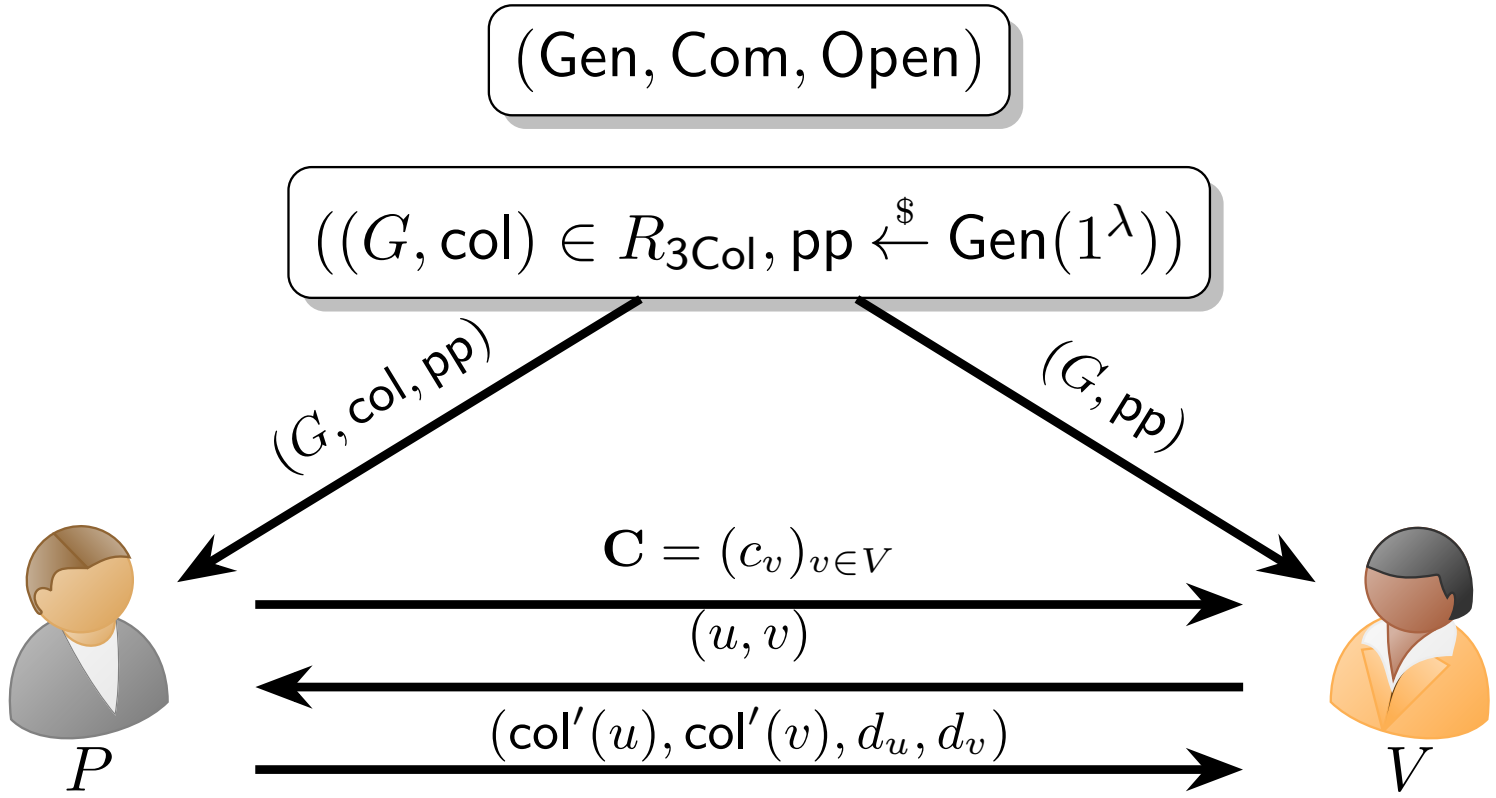
Protocol for 3Col



(P1) Sample random permutation π and compute $(c_v, d_v) \leftarrow \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(v))$, $\forall v \in V$

(V1) Sample $(u, v) \xleftarrow{\$} E$.

Protocol for 3Col



(P1) Sample random permutation π and compute $(c_v, d_v) \leftarrow \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(v))$, $\forall v \in V$

(V1) Sample $(u, v) \xleftarrow{\$} E$.

(V2) Check the following.

(1) $\mathsf{col}'(u), \mathsf{col}'(v) \in \{0, 1, 2\}$

(2) $\mathsf{col}'(u) \neq \mathsf{col}'(v)$

(3) $\mathsf{Open}(\mathsf{col}'(u), c_u, d_u) = 1$

(4) $\mathsf{Open}(\mathsf{col}'(v), c_v, d_v) = 1$

$\mathsf{3Col} \in \mathrm{CZK}$: Completeness

- Let Π_{3Col} denote the protocol on the previous page.

$\mathsf{3Col} \in \mathrm{CZK}$: Completeness

- Let Π_{3Col} denote the protocol on the previous page.

Lemma 4

Π_{3Col} *has perfect completeness.*

$\mathsf{3Col} \in$ CZK: Completeness

- Let Π_{3Col} denote the protocol on the previous page.

Lemma 4

Π_{3Col} *has perfect completeness.*

Proof.

Follows immediately by completeness of Com and that col is a valid 3-coloring of G. Formal proof left as an exercise. □

$\mathsf{3Col} \in \mathrm{CZK}$: Soundness

Lemma 5

If Com *is statistically binding, then* Π_{3Col} *has soundness error at most* $1 - 1/|E|$.

$\mathsf{3Col} \in \mathrm{CZK}$: Soundness

Lemma 5

If Com *is statistically binding, then* Π_{3Col} *has soundness error at most* $1 - 1/|E|$.

Proof.

□

$\mathsf{3Col} \in \mathrm{CZK}$: Soundness

Lemma 5

If Com *is statistically binding, then* Π_{3Col} *has soundness error at most* $1 - 1/|E|$.

Proof.

If G does not have a 3-coloring, then for all functions $\mathsf{col} \colon V \to \{0, 1, 2\}$, there exists an edge $(u, v) \in E$ such that $\mathsf{col}(u) = \mathsf{col}(v)$.

□

$\mathsf{3Col} \in \mathrm{CZK}$: Soundness

Lemma 5

If Com *is statistically binding, then* Π_{3Col} *has soundness error at most* $1 - 1/|E|$.

Proof.

If G does not have a 3-coloring, then for all functions $\mathsf{col} \colon V \to \{0, 1, 2\}$, there exists an edge $(u, v) \in E$ such that $\mathsf{col}(u) = \mathsf{col}(v)$. The probability the verifier samples this edge is at least $1/|E|$.

□

$\mathsf{3Col} \in \mathrm{CZK}$: Soundness

Lemma 5

If Com *is statistically binding, then* Π_{3Col} *has soundness error at most* $1 - 1/|E|$.

Proof.

If G does not have a 3-coloring, then for all functions $\mathsf{col} \colon V \to \{0, 1, 2\}$, there exists an edge $(u, v) \in E$ such that $\mathsf{col}(u) = \mathsf{col}(v)$. The probability the verifier samples this edge is at least $1/|E|$. In this case, the prover P^* could try to convince the verifier that $\mathsf{col}'(u) \neq \mathsf{col}'(v)$ by sending a different value v^* for $\mathsf{col}'(v)$.

□

$\mathsf{3Col} \in \mathrm{CZK}$: Soundness

Lemma 5

If Com *is statistically binding, then* Π_{3Col} *has soundness error at most* $1 - 1/|E|$.

Proof.

If G does not have a 3-coloring, then for all functions $\mathsf{col} \colon V \to \{0, 1, 2\}$, there exists an edge $(u, v) \in E$ such that $\mathsf{col}(u) = \mathsf{col}(v)$. The probability the verifier samples this edge is at least $1/|E|$. In this case, the prover P^* could try to convince the verifier that $\mathsf{col}'(u) \neq \mathsf{col}'(v)$ by sending a different value v^* for $\mathsf{col}'(v)$. However, to fool the verifier, the prover then must provide d_{v^*} such that $\mathsf{Open}(v^*, \mathsf{Com}(\mathsf{col}'(v)), d_{v^*}) = 1$.

□

$\mathsf{3Col} \in \mathrm{CZK}$: Soundness

Lemma 5

If Com *is statistically binding, then* Π_{3Col} *has soundness error at most* $1 - 1/|E|$.

Proof.

If G does not have a 3-coloring, then for all functions $\mathsf{col} \colon V \to \{0, 1, 2\}$, there exists an edge $(u, v) \in E$ such that $\mathsf{col}(u) = \mathsf{col}(v)$. The probability the verifier samples this edge is at least $1/|E|$. In this case, the prover P^* could try to convince the verifier that $\mathsf{col}'(u) \neq \mathsf{col}'(v)$ by sending a different value v^* for $\mathsf{col}'(v)$. However, to fool the verifier, the prover then must provide d_{v^*} such that $\mathsf{Open}(v^*, \mathsf{Com}(\mathsf{col}'(v)), d_{v^*}) = 1$. By statistical binding, P^* can only do this with $\mathsf{negl}(\lambda)$ probability. □

$\mathsf{3Col} \in \mathrm{CZK}$: Soundness

Lemma 5

If Com *is statistically binding, then* Π_{3Col} *has soundness error at most* $1 - 1/|E|$.

Proof.

If G does not have a 3-coloring, then for all functions $\mathsf{col} \colon V \to \{0, 1, 2\}$, there exists an edge $(u, v) \in E$ such that $\mathsf{col}(u) = \mathsf{col}(v)$. The probability the verifier samples this edge is at least $1/|E|$. In this case, the prover P^* could try to convince the verifier that $\mathsf{col}'(u) \neq \mathsf{col}'(v)$ by sending a different value v^* for $\mathsf{col}'(v)$. However, to fool the verifier, the prover then must provide d_{v^*} such that $\mathsf{Open}(v^*, \mathsf{Com}(\mathsf{col}'(v)), d_{v^*}) = 1$. By statistical binding, P^* can only do this with $\mathsf{negl}(\lambda)$ probability. □

- Note that soundness error $1 - 1/|E|$ is quite bad for large E, so one should amplify the soundness of the protocol by sequential repetition.

3Col $\in$ CZK: HVZK

3Col $\in$ CZK: HVZK

- We only show HVZK and leave the full malicious verifier ZK proof as an exercise.

3Col $\in$ CZK: HVZK

- We only show HVZK and leave the full malicious verifier ZK proof as an exercise.
  - Hint: it is similar to the proof for **HAM**.

$\mathsf{3Col} \in$ CZK: HVZK

- We only show HVZK and leave the full malicious verifier ZK proof as an exercise.
  - Hint: it is similar to the proof for **HAM**.

Lemma 6

If Com *is computationally hiding, then* Π_{3Col} *is HVCZK.*

$\mathsf{3Col} \in$ CZK: HVZK

- We only show HVZK and leave the full malicious verifier ZK proof as an exercise.
  - Hint: it is similar to the proof for **HAM**.

Lemma 6

If Com *is computationally hiding, then* Π_{3Col} *is HVCZK.*

- The proof formalizes the intuition of the protocol that the verifier only sees a randomly colored edge.

3Col $\in$ CZK: HVZK

- First, define our simulator.

3Col $\in$ CZK: HVZK

- First, define our simulator.

$S(G, \mathsf{pp})$

$\mathsf{3Col} \in \mathrm{CZK}$: HVZK

- First, define our simulator.

$S(G, \mathsf{pp})$

1. Sample $(u, v) \xleftarrow{\$} E$.

$\mathsf{3Col} \in \mathrm{CZK}$: HVZK

- First, define our simulator.

$S(G, \mathsf{pp})$

1. Sample $(u, v) \xleftarrow{\$} E$.
2. Define col' as $\mathsf{col}'(x) = 0$ for all $x \neq u, v$, and set $\mathsf{col}'(u) \xleftarrow{\$} \{0, 1, 2\}$ and $\mathsf{col}'(v) \xleftarrow{\$} \{0, 1, 2\}$, resampling if $\mathsf{col}'(u) = \mathsf{col}'(v)$.

$\mathsf{3Col} \in \mathrm{CZK}$: HVZK

- First, define our simulator.

$S(G, \mathsf{pp})$

1. Sample $(u, v) \xleftarrow{\$} E$.
2. Define col' as $\mathsf{col}'(x) = 0$ for all $x \neq u, v$, and set $\mathsf{col}'(u) \xleftarrow{\$} \{0, 1, 2\}$ and $\mathsf{col}'(v) \xleftarrow{\$} \{0, 1, 2\}$, resampling if $\mathsf{col}'(u) = \mathsf{col}'(v)$.
3. Compute $(c_w, d_w) = \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(w))$ for all $w \in V$, and set $\mathbf{C} = (c_w)_{w \in V}$.

$\mathsf{3Col} \in \mathrm{CZK}$: HVZK

- First, define our simulator.

$S(G, \mathsf{pp})$

1. Sample $(u, v) \xleftarrow{\$} E$.
2. Define col' as $\mathsf{col}'(x) = 0$ for all $x \neq u, v$, and set $\mathsf{col}'(u) \xleftarrow{\$} \{0, 1, 2\}$ and $\mathsf{col}'(v) \xleftarrow{\$} \{0, 1, 2\}$, resampling if $\mathsf{col}'(u) = \mathsf{col}'(v)$.
3. Compute $(c_w, d_w) = \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(w))$ for all $w \in V$, and set $\mathbf{C} = (c_w)_{w \in V}$.
4. Output $(\mathbf{C}, (u, v), \mathsf{col}'(u), \mathsf{col}'(v), d_u, d_v)$.

$\mathsf{3Col} \in \mathrm{CZK}$: HVZK

- First, define our simulator.

$S(G, \mathsf{pp})$

1. Sample $(u, v) \xleftarrow{\$} E$.
2. Define col' as $\mathsf{col}'(x) = 0$ for all $x \neq u, v$, and set $\mathsf{col}'(u) \xleftarrow{\$} \{0, 1, 2\}$ and $\mathsf{col}'(v) \xleftarrow{\$} \{0, 1, 2\}$, resampling if $\mathsf{col}'(u) = \mathsf{col}'(v)$.
3. Compute $(c_w, d_w) = \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(w))$ for all $w \in V$, and set $\mathbf{C} = (c_w)_{w \in V}$.
4. Output $(\mathbf{C}, (u, v), \mathsf{col}'(u), \mathsf{col}'(v), d_u, d_v)$.

- Remainder of the proof left as an exercise; it is similar to **HAM**.

$\mathsf{3Col} \in \mathrm{CZK}$: HVZK

- First, define our simulator.

$S(G, \mathsf{pp})$

1. Sample $(u, v) \xleftarrow{\$} E$.
2. Define col' as $\mathsf{col}'(x) = 0$ for all $x \neq u, v$, and set $\mathsf{col}'(u) \xleftarrow{\$} \{0, 1, 2\}$ and $\mathsf{col}'(v) \xleftarrow{\$} \{0, 1, 2\}$, resampling if $\mathsf{col}'(u) = \mathsf{col}'(v)$.
3. Compute $(c_w, d_w) = \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(w))$ for all $w \in V$, and set $\mathbf{C} = (c_w)_{w \in V}$.
4. Output $(\mathbf{C}, (u, v), \mathsf{col}'(u), \mathsf{col}'(v), d_u, d_v)$.

- Remainder of the proof left as an exercise; it is similar to **HAM**.
  - Define a hybrid simulator $\widetilde{S}$ which takes $G, \mathsf{col}, \mathsf{pp}$ as input.

$\mathsf{3Col} \in \mathrm{CZK}$: HVZK

- First, define our simulator.

$S(G, \mathsf{pp})$

1. Sample $(u, v) \xleftarrow{\$} E$.
2. Define col' as $\mathsf{col}'(x) = 0$ for all $x \neq u, v$, and set $\mathsf{col}'(u) \xleftarrow{\$} \{0, 1, 2\}$ and $\mathsf{col}'(v) \xleftarrow{\$} \{0, 1, 2\}$, resampling if $\mathsf{col}'(u) = \mathsf{col}'(v)$.
3. Compute $(c_w, d_w) = \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(w))$ for all $w \in V$, and set $\mathbf{C} = (c_w)_{w \in V}$.
4. Output $(\mathbf{C}, (u, v), \mathsf{col}'(u), \mathsf{col}'(v), d_u, d_v)$.

- Remainder of the proof left as an exercise; it is similar to **HAM**.
  - Define a hybrid simulator $\widetilde{S}$ which takes $G, \mathsf{col}, \mathsf{pp}$ as input.
  - $\widetilde{S}$ then randomly samples (u, v) and $\mathsf{col}'(u) \neq \mathsf{col}'(v)$.

$\mathsf{3Col} \in \mathrm{CZK}$: HVZK

- First, define our simulator.

$S(G, \mathsf{pp})$

1. Sample $(u, v) \xleftarrow{\$} E$.
2. Define col' as $\mathsf{col}'(x) = 0$ for all $x \neq u, v$, and set $\mathsf{col}'(u) \xleftarrow{\$} \{0, 1, 2\}$ and $\mathsf{col}'(v) \xleftarrow{\$} \{0, 1, 2\}$, resampling if $\mathsf{col}'(u) = \mathsf{col}'(v)$.
3. Compute $(c_w, d_w) = \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(w))$ for all $w \in V$, and set $\mathbf{C} = (c_w)_{w \in V}$.
4. Output $(\mathbf{C}, (u, v), \mathsf{col}'(u), \mathsf{col}'(v), d_u, d_v)$.

- Remainder of the proof left as an exercise; it is similar to **HAM**.
  - Define a hybrid simulator $\widetilde{S}$ which takes $G, \mathsf{col}, \mathsf{pp}$ as input.
  - $\widetilde{S}$ then randomly samples (u, v) and $\mathsf{col}'(u) \neq \mathsf{col}'(v)$.
  - $\widetilde{S}$ then finds unique col' that is consistent with this choice and col.

$\mathsf{3Col} \in \mathrm{CZK}$: HVZK

- First, define our simulator.

$S(G, \mathsf{pp})$

1. Sample $(u, v) \xleftarrow{\$} E$.
2. Define col' as $\mathsf{col}'(x) = 0$ for all $x \neq u, v$, and set $\mathsf{col}'(u) \xleftarrow{\$} \{0, 1, 2\}$ and $\mathsf{col}'(v) \xleftarrow{\$} \{0, 1, 2\}$, resampling if $\mathsf{col}'(u) = \mathsf{col}'(v)$.
3. Compute $(c_w, d_w) = \mathsf{Com}(\mathsf{pp}, \mathsf{col}'(w))$ for all $w \in V$, and set $\mathbf{C} = (c_w)_{w \in V}$.
4. Output $(\mathbf{C}, (u, v), \mathsf{col}'(u), \mathsf{col}'(v), d_u, d_v)$.

- Remainder of the proof left as an exercise; it is similar to **HAM**.
  - Define a hybrid simulator $\widetilde{S}$ which takes $G, \mathsf{col}, \mathsf{pp}$ as input.
  - $\widetilde{S}$ then randomly samples (u, v) and $\mathsf{col}'(u) \neq \mathsf{col}'(v)$.
  - $\widetilde{S}$ then finds unique col' that is consistent with this choice and col.
  - Computational hiding will show these two simulators are indistinguishable.

More on CZK

More on CZK

- One-way functions are *necessary* for non-trivial zero-knowledge.

More on CZK

- One-way functions are *necessary* for non-trivial zero-knowledge.

Theorem 4 (Ostrovsky-Wigderson (90))

If there exists $L \in \mathbf{CZK} \setminus \mathbf{BPP}$, *then there exist functions with one-way instances.*

More on CZK

- One-way functions are *necessary* for non-trivial zero-knowledge.

Theorem 4 (Ostrovsky-Wigderson (90))

If there exists $L \in \mathbf{CZK} \setminus \mathbf{BPP}$, *then there exist functions with one-way instances.*

Theorem 5 (Ostrovsky-Wigderson (90))

If there exists $L \in \mathbf{CZK}$ *that is hard-on-average, then one-way functions exist.*

More on CZK

- Salil Vadhan (2006) gave a complete characterization of **CZK**.

More on CZK

- Salil Vadhan (2006) gave a complete characterization of **CZK**.

Theorem 6 (Unconditional Characterization of CZK)

More on CZK

- Salil Vadhan (2006) gave a complete characterization of **CZK**.

Theorem 6 (Unconditional Characterization of CZK)

- **HVCZK** = **CZK**;

More on CZK

- Salil Vadhan (2006) gave a complete characterization of **CZK**.

Theorem 6 (Unconditional Characterization of CZK)

- **HVCZK** = **CZK**;
- **CZK** *is closed under union;*

More on CZK

- Salil Vadhan (2006) gave a complete characterization of **CZK**.

Theorem 6 (Unconditional Characterization of CZK)

- **HVCZK** = **CZK**;
- **CZK** *is closed under union;*
- *Public-coin* **CZK** *equals private-coin* **CZK**;

More on CZK

- Salil Vadhan (2006) gave a complete characterization of **CZK**.

Theorem 6 (Unconditional Characterization of CZK)

- **HVCZK** = **CZK**;
- **CZK** *is closed under union;*
- *Public-coin* **CZK** *equals private-coin* **CZK**;
- **CZK** *with imperfect completeness equals* **CZK** *with perfect completeness.*

Next Time: Zero-knowledge Wrap-up