Hard Languages in $\mathbf{NP} \cap \mathbf{coNP}$ and NIZK Proofs from Unstructured Hardness

Riddhi Ghosal, Yuval Ishai, Alexis Korb, Eyal Kushilevitz, Paul Lou, Amit Sahai

Hardness of $\mathbf{NP} \cap \mathbf{coNP}$

- Hard Language: Fully specified decision problems not in $\mathbf{P}$.

Hardness of $\mathbf{NP} \cap \mathbf{coNP}$

- Hard Language: Fully specified decision problems not in $\mathbf{P}$.
- $\mathbf{NP} \cap \mathbf{coNP}$: Languages for which there exists an efficient $\mathbf{NP}$ verifier for both membership and non-membership.

Hardness of $\mathbf{NP} \cap \mathbf{coNP}$

- Hard Language: Fully specified decision problems not in $\mathbf{P}$.
- $\mathbf{NP} \cap \mathbf{coNP}$: Languages for which there exists an efficient $\mathbf{NP}$ verifier for both membership and non-membership.
- Candidate hard languages in $\mathbf{NP} \cap \mathbf{coNP}$ are highly structured and few.
  - Languages related to factoring and discrete log.
  - Stochastic Games [Condon92]
  - Construction from OWPs [Brassard79, BennettGill81]
    - Only known constructions of OWPs rely on factoring or discrete log

Hardness of $\mathbf{NP} \cap \mathbf{coNP}$

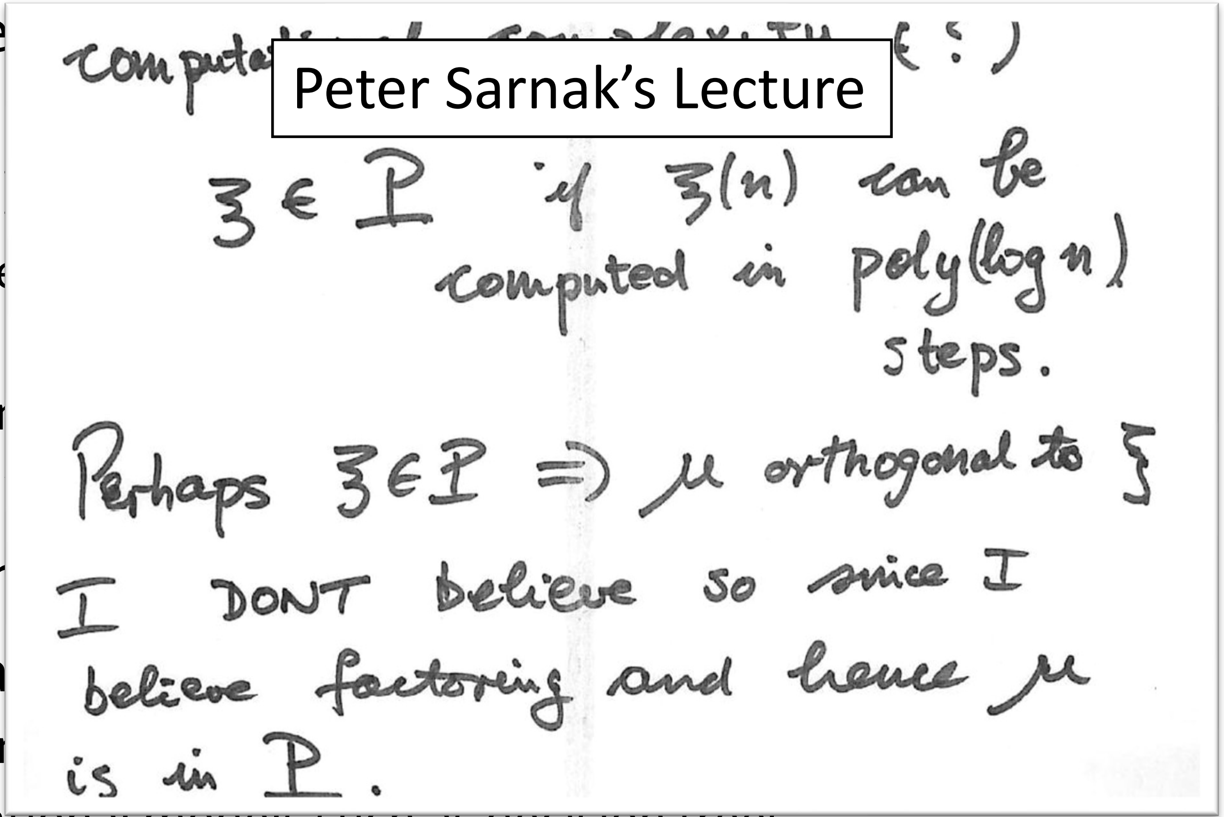
- Hard Language: Fully specified decision problems not in $\mathbf{P}$.
- $\mathbf{NP} \cap \mathbf{coNP}$: Languages for which there exists an efficient $\mathbf{NP}$ verifier for both membership and non-membership.
- Candidate hard languages in $\mathbf{NP} \cap \mathbf{coNP}$ are highly structured and few.
    - Languages related to factoring and discrete log.
    - Stochastic Games [Condon92]
    - Construction from OWPs [Brassard79, BennettGill81]
        - Only known constructions of OWPs rely on factoring or discrete log
- Note: This is not the case for promise problems.

Hardness of $\mathbf{NP} \cap \mathbf{coNP}$

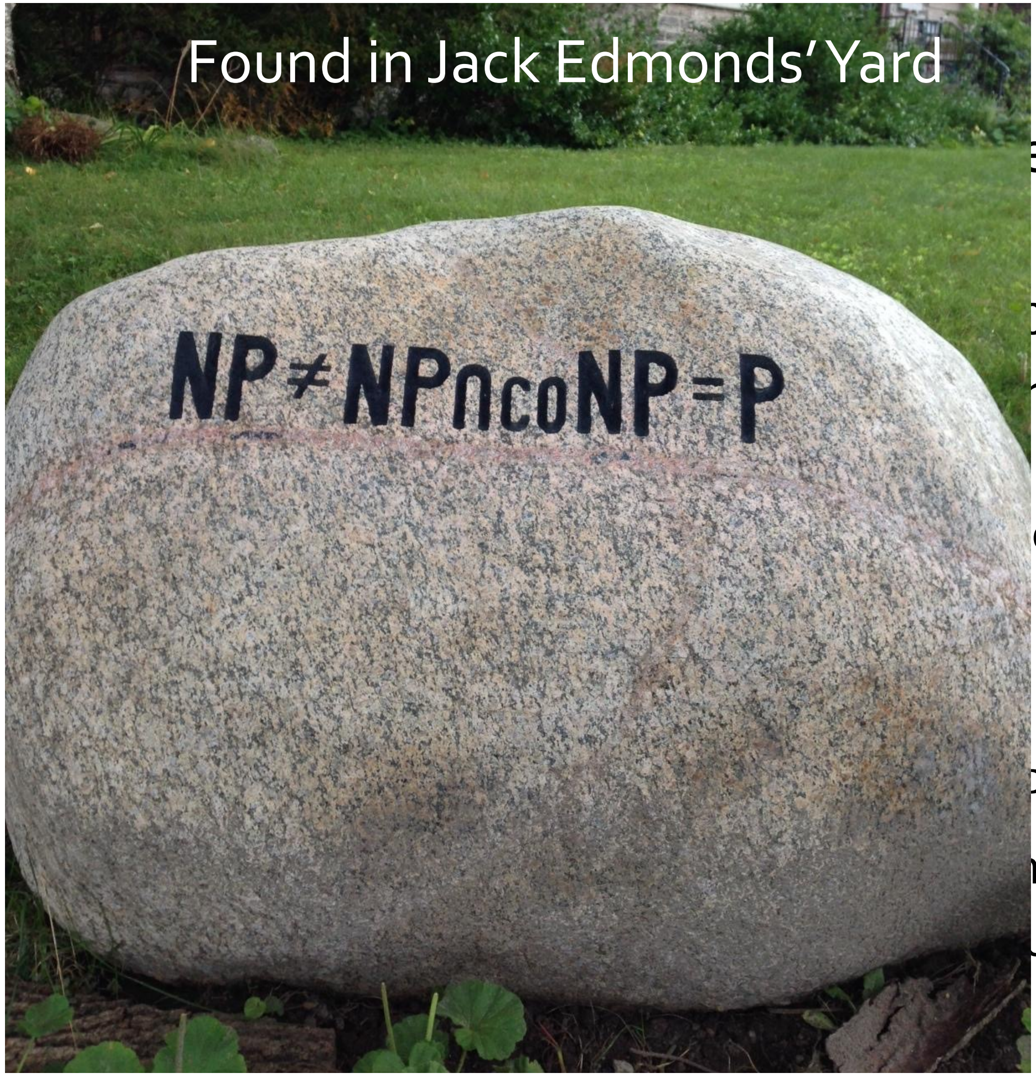
- Hard Language: Fully specified decision problems not in $\mathbf{P}$.
- $\mathbf{NP} \cap \mathbf{coNP}$: Languages for which there exists an efficient $\mathbf{NP}$ verifier for both membership and non-membership.
- Candidate hard languages in $\mathbf{NP} \cap \mathbf{coNP}$ are highly structured and few.
  - Languages related to factoring and discrete log.
  - Stochastic Games [Condon92]
  - Construction from OWPs [Brassard79, BennettGill81]
    - Only known constructions of OWPs rely on factoring or discrete log.
- Note: This is not the case for promise problems.
- Maybe: Unclear how hard $\mathbf{NP} \cap \mathbf{coNP}$ actually is?
  - Most current candidates broken by quantum algorithms.
  - $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$ holds for simple computational models such as decision trees.
  - No complete languages known.

Hardness of $\mathbf{NP} \cap \mathbf{coNP}$

- Hard Language: Fully specified decision problems not in $\mathbf{P}$.
- $\mathbf{NP} \cap \mathbf{coNP}$: Languages for which there
  membership and non-membership.
- Candidate hard languages in $\mathbf{NP} \cap \mathbf{coN}$
  - Languages related to factoring and discrete
  - Stochastic Games [Condon92]
  - Construction from OWPs [Brassard79, Benn
    - Only known constructions of OWPs rel
- Note: This is not the case for promise pr
- Maybe: Unclear how hard $\mathbf{NP} \cap \mathbf{coNP}$ a
  - Most current candidates broken by quantu
  - $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$ holds for simple computational models such as decision trees.
  - No complete languages known.



Hardness of **NP ∩ coNP**



on problems not in **P**.

ere

oN

rete

enn

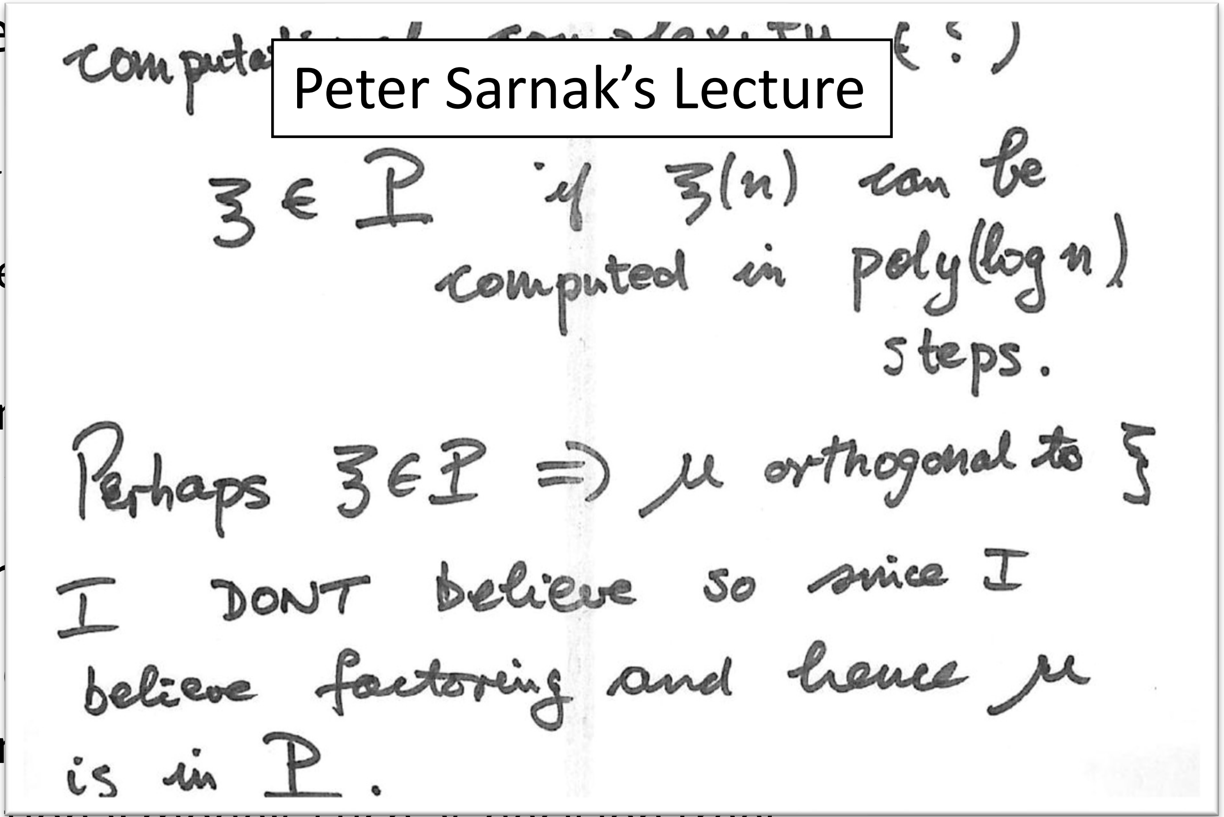
rel

pr

P a

ntu

utational models such as decision trees.



Hardness of $\mathbf{NP} \cap \mathbf{coNP}$

- Hard Language: Fully specified decision problems not in $\mathbf{P}$.
- $\mathbf{NP} \cap \mathbf{coNP}$: Languages for which there exists an efficient $\mathbf{NP}$ verifier for both membership and non-membership.
- Candidate hard languages in $\mathbf{NP} \cap \mathbf{coNP}$ are highly structured and few.
  - Languages related to factoring and discrete log.
  - Stochastic Games [Condon92]
  - Construction from OWPs [Brassard79, BennettGill81]
    - Only known constructions of OWPs rely on factoring or discrete log.
- Note: This is not the case for promise problems.
- Maybe: Unclear how hard $\mathbf{NP} \cap \mathbf{coNP}$ actually is?
  - Most current candidates broken by quantum algorithms.
  - $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$ holds for simple computational models such as decision trees.
  - No complete languages known.

Hardness of $\mathbf{NP} \cap \mathbf{coNP}$

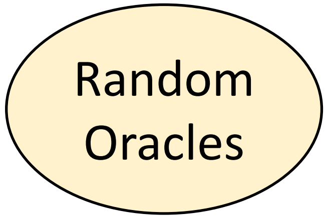
- Hard Language: Fully specified decision problems not in $\mathbf{P}$.
- $\mathbf{NP} \cap \mathbf{coNP}$: Languages for which there exists an efficient $\mathbf{NP}$ verifier for both membership and non-membership.
- Candidate hard languages in $\mathbf{NP} \cap \mathbf{coNP}$ are highly structured and few.
  - Languages related to factoring and discrete log

Can we build a hard language in

$\mathrm{NP} \cap \mathrm{coNP}$ from unstructured assumptions?

- Note: This is not the case for promise problems.
- Maybe: Unclear how hard $\mathbf{NP} \cap \mathbf{coNP}$ actually is?
  - Most current candidates broken by quantum algorithms.
  - $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$ holds for simple computational models such as decision trees.
  - No complete languages known.

Which assumptions represent unstructured hardness?



Unstructured Assumptions: assumptions that follow from random oracles.

Which assumptions represent unstructured hardness?

Random Oracles

Private Key Encryption (Unstructured) vs Public Key Encryption (Structure)
[Formalized by Impagliazzo and Rudich]

Unstructured Assumptions: assumptions that follow from random oracles.

Which assumptions represent unstructured hardness?

OWPs

Injective OWFs (Slightly Expanding)

Injective OWFs (Length Tripling)

Unstructured Assumptions: assumptions that follow from random oracles.

Which assumptions represent unstructured hardness?

$O: \{0,1\}^n \rightarrow \{0,1\}^n$

$O: \{0,1\}^n \rightarrow \{0,1\}^{n+O(1)}$

$O: \{0,1\}^n \rightarrow \{0,1\}^{3n}$

OWPs

Injective OWFs (Slightly Expanding)

Injective OWFs (Length Tripling)

Unstructured Assumptions: assumptions that follow from random oracles.

Which assumptions represent unstructured hardness?

Structured

Unstructured

$O: \{0,1\}^n \to \{0,1\}^n$

$O: \{0,1\}^n \to \{0,1\}^{n+O(1)}$

$O: \{0,1\}^n \to \{0,1\}^{3n}$

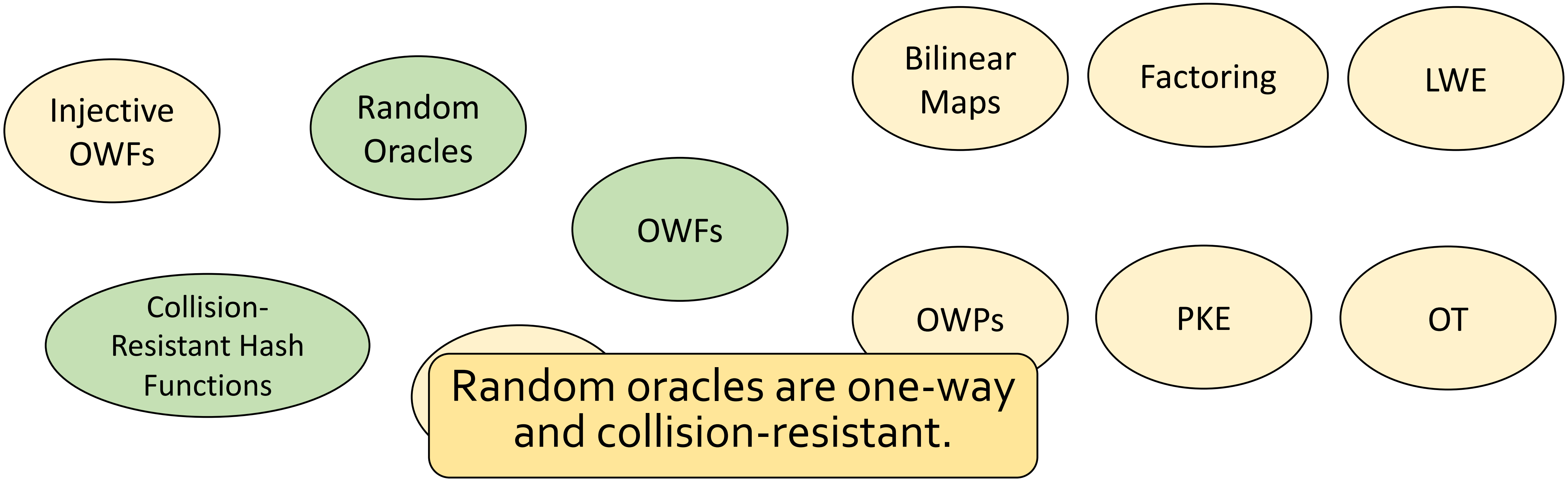
OWPs

Injective OWFs (Slightly Expanding)

Injective OWFs (Length Tripling)

Unstructured Assumptions: assumptions that follow from random oracles.

Which assumptions represent unstructured hardness?



Unstructured Assumptions: assumptions that follow from random oracles.

Which assumptions represent unstructured hardness?

Length-tripling random oracle is injective w.h.p.

Injective OWFs

Random Oracles

OWFs

Collision-Resistant Hash Functions

$UP \not\subseteq RP$

Bilinear Maps

Factoring

LWE

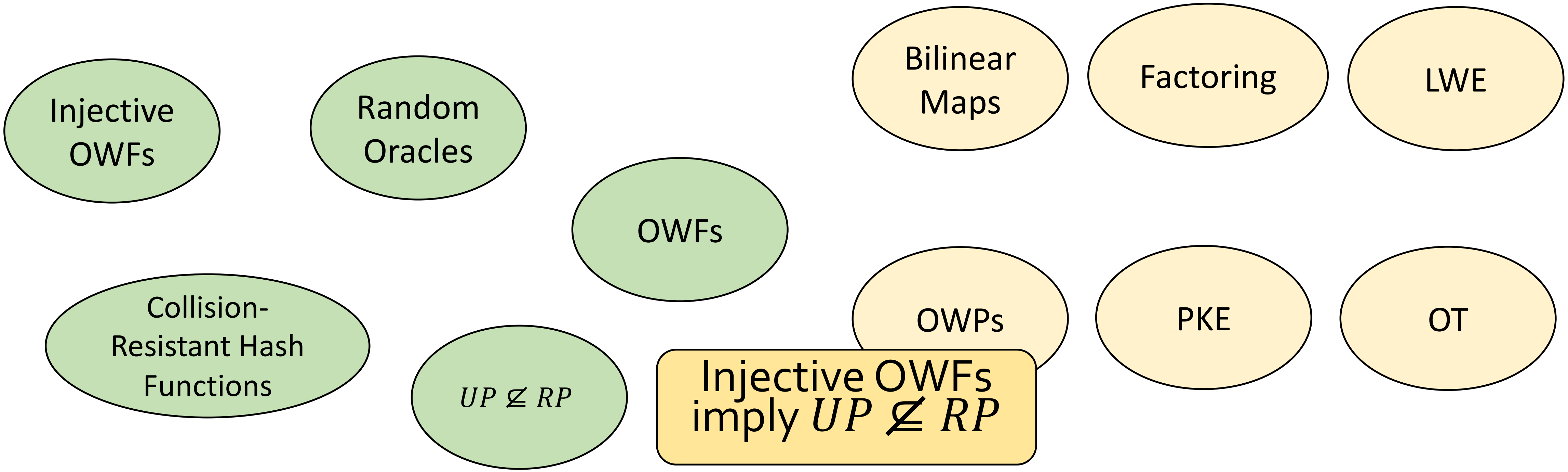
OWPs

PKE

OT

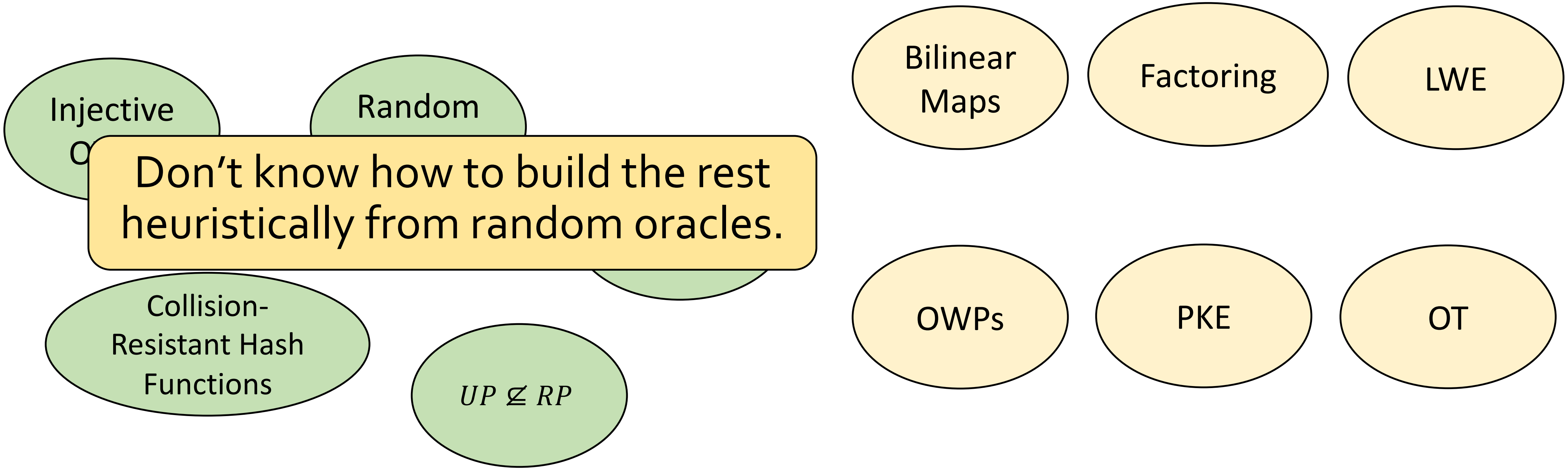
Unstructured Assumptions: assumptions that follow from random oracles.

Which assumptions represent unstructured hardness?



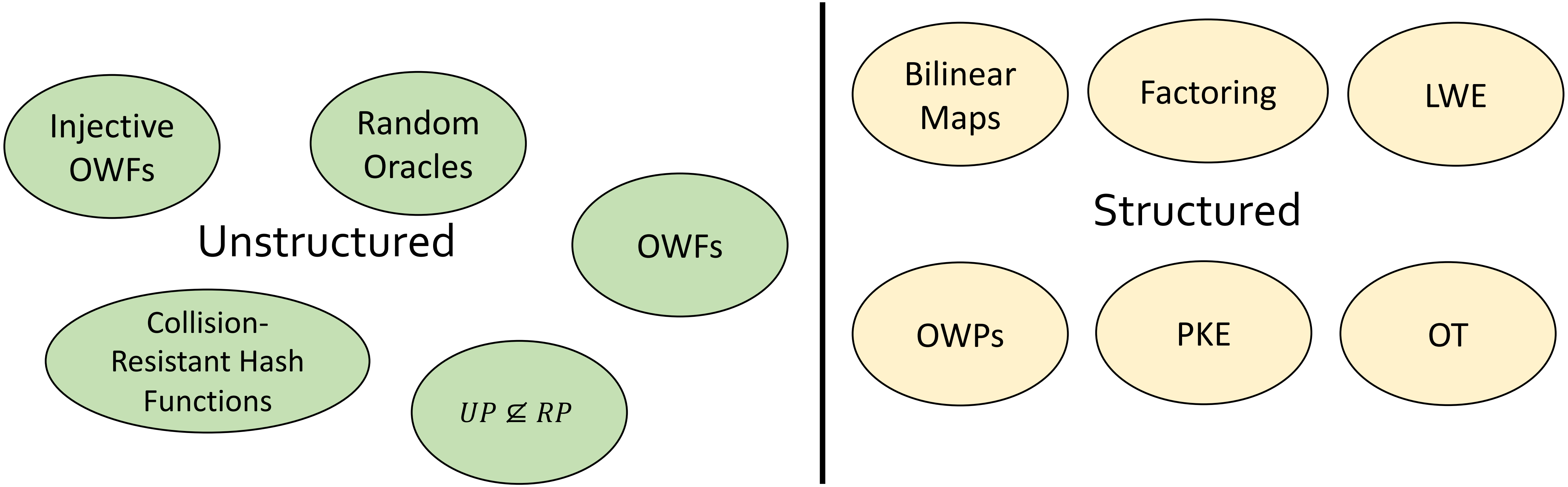
Unstructured Assumptions: assumptions that follow from random oracles.

Which assumptions represent unstructured hardness?



Unstructured Assumptions: assumptions that follow from random oracles.

Which assumptions represent unstructured hardness?



Unstructured Assumptions: assumptions that follow from random oracles.

Hardness of $\mathbf{NP} \cap \mathbf{coNP}$ from Unstructured Assumptions

- No known random oracle separation of $\mathbf{P}$ and $\mathbf{NP} \cap \mathbf{coNP}$
  - [BennettGill81] Open problem since 1981.
  - [Tardos89] details some difficulties with this approach.

Hardness of $\mathbf{NP} \cap \mathbf{coNP}$ from Unstructured Assumptions

- No known random oracle separation of $\mathbf{P}$ and $\mathbf{NP} \cap \mathbf{coNP}$
  - [BennettGill81] Open problem since 1981.
  - [Tardos89] details some difficulties with this approach.
- No black-box constructions of hard languages in $\mathbf{NP} \cap \mathbf{coNP}$ from
  - OWFs [BlumImpagliazzo87, Rudich88, KahnSaksSmythoo]
  - Injective OWFs and Indistinguishability Obfuscation (iO) [BitanskyDegwekarVaikuntanathan21]
    - Implies no black-box constructions from many cryptographic primitives since iO + OWFs can be used to build a lot of crypto.

Hardness of $\mathbf{NP} \cap \mathbf{coNP}$ from Unstructured Assumptions

- No known random oracle separation of $\mathbf{P}$ and $\mathbf{NP} \cap \mathbf{coNP}$
  - [BennettGill81] Open problem since 1981.
  - [Tardos89] details some difficulties with this approach.
- No black-box constructions of hard languages in $\mathbf{NP} \cap \mathbf{coNP}$ from
  - OWFs [BlumImpagliazzo87, Rudich88, KahnSaksSmythoo]
  - Injective OWFs and Indistinguishability Obfuscation (iO) [BitanskyDegwekarVaikuntanathan21]
    - Implies no black-box constructions from many cryptographic primitives since iO + OWFs can be used to build a lot of crypto.

Can we build a hard language in $\mathbf{NP} \cap \mathbf{coNP}$ from random oracles?

Random Oracle Separations of Complexity Classes

Random Oracle Separations of Complexity Classes

- A lot of exciting work in complexity theory
  - [BennettGill81] P, NP, and coNP separated by random oracles.
  - [RossmanServedioTan15] Polynomial hierarchy is infinite relative to a random oracle.
  - [YamakawaZhandry22] Separation of search-BQP and search-BPP relative to a random oracle.

Random Oracle Separations of Complexity Classes

- A lot of exciting work in complexity theory
  - [BennettGill81] P, NP, and coNP separated by random oracles.
  - [RossmanServedioTan15] Polynomial hierarchy is infinite relative to a random oracle.
  - [YamakawaZhandry22] Separation of search-BQP and search-BPP relative to a random oracle.

- Random Oracle Hypothesis [BG81]: random oracle separations of complexity classes imply a non-random-oracle separation of the same classes
  - [CCGHHRR92] False for IP and PSPACE
  - Plausibly true for feasible complexity classes.
- Similar hypothesis in cryptography:
  - Can heuristically construct a *concrete* language by instantiating the random oracle with a cryptographic hash function.

Our Results

Main Theorem

If there exists injective OWFs, then with probability 1 over the choice of a random oracle O, $P^O \neq NP^O \cap coNP^O$

Our Results

Main Theorem

If there exists injective OWFs, then with probability 1 over the choice of a random oracle O, $P^O \neq NP^O \cap coNP^O$

Our proof is constructive!

Cryptographic Hash Function H

Injective OWF

Polytime Algorithm

Candidate language in $(NP \cap coNP)\backslash \mathrm{P}$

Our Results

Main Theorem

If there exists $\boxed{\text{injective OWFs}}$, then with probability 1 over the choice of a random oracle O, $P^O \neq NP^O \cap coNP^O$

Suffices to assume $UP \nsubseteq RP$ which is implied by injective OWFs.

Our Results

Main Theorem

If $UP \not\subseteq RP$, then with probability 1 over the choice of a random oracle O,
$P^O \neq NP^O \cap coNP^O$

Our Results

Main Theorem

If $UP \not\subseteq RP$, then with probability 1 over the choice of a random oracle O,
$$P^O \neq NP^O \cap coNP^O$$

Main New Ingredient:

A Non-Interactive Zero Knowledge (NIZK) **proof** system in the random oracle model!

(Note: Fiat-Shamir only gives NIZK arguments.)

Our Results

Main Theorem

If $UP \not\subseteq RP$, then with probability 1 over the choice of a random oracle O, $P^O \neq NP^O \cap coNP^O$

NIZK Proofs in Random Oracle Model

There exists an (unbounded-prover) NIZK proof system for NP in the random oracle model.

Our Results

Main Theorem

If $UP \not\subseteq RP$, then with probability 1 over the choice of a random oracle O,
$P^O \neq NP^O \cap coNP^O$

NIZK Proofs in Random Oracle Model

There exists an (unbounded-prover) NIZK proof system for NP in the random oracle model.

Can also build NIZK Proofs in URS model from a concrete cryptographic object we call
δ-Dense-PRHFs.

δ-Dense-Pseudorandom-Hash-Functions

- Functions $H: \{0,1\}^n \rightarrow \{0,1\}^m$ satisfying three properties:
  1. Pseudorandom Output:
     - Let X be uniform over $\{0,1\}^n$ and U_m be uniform over $\{0,1\}^m$.
     - Then $H(X) \approx_c U_m$

δ-Dense-Pseudorandom-Hash-Functions

- Functions $H: \{0,1\}^n \rightarrow \{0,1\}^m$ satisfying three properties:
  1. Pseudorandom Output:
     - Let X be uniform over $\{0,1\}^n$ and U_m be uniform over $\{0,1\}^m$.
     - Then $H(X) \approx_c U_m$
  2. δ-Dense: The image is δ-Dense in the codomain.
     - Constant $\delta \in (0,1)$ which is "efficiently approximable".
     - $Pr[U_m \in Image(H)] = \delta \pm negl(n)$

δ-Dense-Pseudorandom-Hash-Functions

- Functions $H: \{0,1\}^n \rightarrow \{0,1\}^m$ satisfying three properties:
  1. Pseudorandom Output:
     - Let X be uniform over $\{0,1\}^n$ and U_m be uniform over $\{0,1\}^m$.
     - Then $H(X) \approx_c U_m$
  2. δ-Dense: The image is δ-Dense in the codomain.
     - Constant $\delta \in (0,1)$ which is "efficiently approximable".
     - $Pr[U_m \in Image(H)] = \delta \pm negl(n)$
  3. Preimage Pseudorandomness:
     - Let Y be uniform over $Image(H)$ and let $H^{-1}(y)$ output a random preimage of y.
     - Then $(X, H(X)) \approx_c (H^{-1}(Y), Y)$

Our Results

Main Theorem

If $UP \nsubseteq RP$, then with probability 1 over the choice of a random oracle O, $P^O \neq NP^O \cap coNP^O$

NIZK Proofs in Random Oracle Model

There exists an (unbounded-prover) NIZK proof system for NP in the random oracle model.

NIZK Proofs in URS model from δ-Dense-PRHFs

Assuming there exists a δ-Dense-PRHF,

there exists an (unbounded-prover) NIZK proof system for NP in the URS model.

Our Results

Main Theorem

If $UP \not\subseteq RP$, then with probability 1 over the choice of a random oracle O,
$P^O \neq NP^O \cap coNP^O$

- Implied by Random Oracle
- No known instantiation from other unstructured assumptions

The e
random orac model.

NIZK Proofs in URS model from δ-Dense-PRHFs

Assuming there exists a δ-Dense-PRHF,

there exists an (unbounded-prover) NIZK proof system for NP in the URS model.

NIZK Proofs for NP in URS Model [BFM88]

- Goal: Prover P is trying to prove to a verifier V that $x \in L$.
- Setting:
  - Unbounded prover P
  - Computationally bounded (poly-sized) verifier V
  - URS model : P and V share uniformly random string
- Properties
  - **Completeness:** If all players are honest and $x \in L$, the verifier accepts.
  - **Soundness:** If $x \notin L$, no unbounded cheating prover should be able to convince an honest verifier to accept.
  - **Zero Knowledge:** Security against dishonest poly-sized verifiers.
    - There exists a PPT Sim such that $\forall x \in L$, Sim(x) ≈ (urs, P(urs, x))

NIZK Proofs for NP in Random Oracle Model

- Goal: Prover P is trying to prove to a verifier V that $x \in L$.
- Setting:
  - Unbounded prover P
  - Computationally bounded (poly-sized) verifier V
  - **Random Oracle model**: P and V have query access to a random oracle.
- Properties
  - **Completeness:** If all players are honest and $x \in L$, the verifier accepts.
  - **Soundness:** If $x \notin L$, no unbounded cheating prover should be able to convince an honest verifier to accept.
  - **Zero Knowledge:** Security against dishonest verifiers that can make polynomially many queries to the random oracle.
    - There exists a PPT Sim = (SimO, SimP) such that $\forall x \in L$, "(SimO, SimP(x)) ≈ (O, P^O(x))"

Previous Work on NIZKs

	Proofs (secure against unbounded prover)	**Arguments (secure against PPT prover)**
URS (uniform random string)	• OWPs [FLS90, BY96, CL18] • DLIN on bilinear groups [GOS06] • iO and OWFs [BP15]	• Random oracle [FS86] • Many assumptions
SRS (structured random string)	• OWFs [Ps05] (unbounded prover) • Lattices [CCH+19, PS19] • Many assumptions	• Many assumptions

Previous Work on NIZKs

	Proofs (secure against unbounded prover)	**Arguments (secure against PPT prover)**
URS (uniform random string)	• OWPs [FLS90, BY96, CL18] • DLIN on bilinear groups [GOS06] • iO and OWFs [BP15]	• Random oracle [FS86] • Many assumptions
SRS (structured random string)	• OWFs [Ps05] (unbounded prover) • Lattices [CCH+19, PS19] • Many assumptions	• Many assumptions

Structured Hardness
(and not post-quantum,
except maybe iO)

Previous Work on NIZKs

	Proofs (secure against unbounded prover)	**Arguments (secure against PPT prover)**
URS (uniform random string)	• OWPs [FLS90, BY96, CL18] • DLIN on bilinear groups [GOS06] • iO and OWFs [BP15] • **Random oracle or δ-Dense-PRHF [Our Work]**	• Random oracle [FS86] • Many assumptions
SRS (structured random string)	• OWFs [Ps05] (unbounded prover) • Lattices [CCH+19, PS19] • Many assumptions	• Many assumptions

Unstructured Hardness

Our Results

Main Theorem

If $UP \nsubseteq RP$, then with probability 1 over the choice of a random oracle O, $P^O \neq NP^O \cap coNP^O$

NIZK Proofs in Random Oracle Model

There exists an (unbounded-prover) NIZK proof system for NP in the random oracle model.

NIZK Proofs in URS model from δ-Dense-PRHFs

Assuming there exists a δ-Dense-PRHF,

there exists an (unbounded-prover) NIZK proof system for NP in the URS model.

Separating P^O and $NP^O \cap coNP^O$

Separating P^O and $NP^O \cap coNP^O$

- Ingredients
  - Injective OWF: f
  - NIZK Proof $(P^{(\cdot)}, V^{(\cdot)}, Sim)$ in Random Oracle model for the language
    - $L' = \{y\colon \exists x, f(x) = y\}$: "y has a preimage"

Separating P^O and $NP^O \cap coNP^O$

- Ingredients
  - Injective OWF: f
  - NIZK Proof $(P^{(.)}, V^{(.)}, Sim)$ in Random Oracle model for the language
    - $L' = \{y : \exists x, f(x) = y\}$: "y has a preimage"

- $L = \{(y, i) : (\exists x, f(x) = y \wedge x_i = 1)\}$ Promise : y always has a preimage

Separating P^O and $NP^O \cap coNP^O$

- Ingredients
  - Injective OWF: f
  - NIZK Proof $(P^{(.)}, V^{(.)}, Sim)$ in Random Oracle model for the language
    - $L' = \{y : \exists x, f(x) = y\}$: "y has a preimage"

- $L = \{(y, i) : (\exists x, f(x) = y \wedge x_i = 1)\}$ Promise : y always has a preimage

- Our Language (with random oracle O)
  - $L^O = \{(y, i, \pi) : (\exists x, f(x) = y \wedge x_i = 1) \wedge V^O(y, \pi) = 1\}$
    "y has a preimage x where $x_i = 1$" and
    "π is a valid proof that y has a preimage"

Separating P^O and $NP^O \cap coNP^O$

- Ingredients
  - Injective OWF: f
  - NIZK Proof $(P^{(.)}, V^{(.)}, Sim)$ in Random Oracle model for the language
    - $L' = \{y: \exists x, f(x) = y\}$: "y has a preimage"

- $L = \{(y,i): (\exists x, f(x) = y \wedge x_i = 1)\}$ Promise : y always has a preimage

- Our Language (with random oracle O)
  -

Similar proof also works assuming a language

$L'' \in UP \backslash \mathrm{RP}$

in which case

$L' = \{y: \exists w, (y,w) \in R_{L''}\}$

$L^O = \{(y,i,\pi): (\exists w, (y,w) \in R_{L''} \wedge w_i = 1) \wedge V^O(y,\pi) = 1\}$

$L^O \in NP^O$

$$L^O = \{(y, i, \pi) : (\exists x, f(x) = y \wedge x_i = 1) \wedge V^O(y, \pi) = 1\}$$

$L^O \in NP^O$

$$L^O = \{(y, i, \pi) : (\exists x, f(x) = y \wedge x_i = 1) \wedge V^O(y, \pi) = 1\}$$

$D_{NP}^O((y, i, \pi), w)$

1. Check if $V^O(y, \pi)$ verifies. If not, then $(y, i, \pi) \notin L^O$. Reject.
2. Check that for witness w, $f(w) = y$. If not, reject.
3. Accept if $w_i = 1$.

The correctness of $D_{NP}^O((y, i, \pi), w)$ follows from definition of L^O.

If NIZK perfectly sound*, $Pr_O[L^O \in coNP^O]$=1

$$\bar{L}^O = \{(y, i, \pi) : (\nexists\, x, f(x) = y \wedge x_i = 1) \vee (V^O(y, \pi) = 0)\}$$

If NIZK perfectly sound*, $Pr_O[L^O \in coNP^O]$=1

$$\bar{L}^O = \{(y, i, \pi) : (\nexists\, x, f(x) = y \wedge x_i = 1) \vee (V^O(y, \pi) = 0)\}$$

$D^O_{coNP}((y, i, \pi), w)$

1. Check if $V^O(y, \pi)$ verifies. If not, then $(y, i, \pi) \in \bar{L}^O$. Accept.
   - Otherwise, soundness of NIZK proof ensures $\exists x, f(x) = y$.
   - This x is *unique* since f is injective!
   - Expect witness w to be this unique x.
2. Check that for witness w, $f(w) = y$. If not, reject.
3. Accept if $w_i = 0$.

If NIZK is ZK, $Pr_O[L^O \notin P^O] = 1$

$$L^O = \{(y, i, \pi) : (\exists x, f(x) = y \land x_i = 1) \land V^O(y, \pi) = 1\}$$

If NIZK is ZK, $Pr_O[L^O \notin P^O] = 1$

$$L^O = \{(y, i, \pi) : (\exists x, f(x) = y \wedge x_i = 1) \wedge V^O(y, \pi) = 1\}$$

Assume $Pr_O\,[L^O \in P^O] > 0.$

Theorem from [BG81] implies there exists a polytime Turing Machine $D^{(\cdot)}$ which decides $L^{(\cdot)}$ with probability 1 over the choice of O.

If NIZK is ZK, $Pr_O[L^O \notin P^O] = 1$

$$L^O = \{(y, i, \pi) : (\exists x, f(x) = y \wedge x_i = 1) \wedge V^O(y, \pi) = 1\}$$

Assume $Pr_O\,[L^O \in P^O] > 0$.

Theorem from [BG81] implies there exists a polytime Turing Machine $D^{(\cdot)}$ which decides $L^{(\cdot)}$ with probability 1 over the choice of O.

Then, w.h.p we could invert OWF f!

f-Inverter(y):

1. For each i:
   a. Use NIZK simulator to simulate a proof π that y has a preimage.
   b. Set x_i = $D^{SimO}(y, i, \pi)$ (using NIZK simulator to simulate random oracle queries).
      I. If π was a real proof, then D would output correct x_i.
      II. Zero knowledge ensures that D acts similarly on simulated proof!
2. Output x.

Constructing NIZK Proofs in Random Oracle Model

NIZK Proofs for NP in the Random Oracle Model

- Starting Point: [FLS90] NIZK Proof for NP from OWPs in URS model.
- Goal: Replace OWPs with random oracle.
  - (Trivial to replace URS with random oracle.)

NIZK Proofs for NP in the Random Oracle Model

- Starting Point: [FLS90] NIZK Proof for NP from OWPs in URS model.
- Goal: Replace OWPs with random oracle.
  - (Trivial to replace URS with random oracle.)

[FLS90] Proof Overview

1. Build NIZK Proofs for NP in Hidden Bits Model (HB).
2. Instantiate HB with URS and OWP.

NIZK Proofs for NP in the Random Oracle Model

- Starting Point: [FLS90] NIZK Proof for NP from OWPs in URS model.
- Goal: Replace OWPs with random oracle.
  - (Trivial to replace URS with random oracle.)

[FLS90] Proof Overview

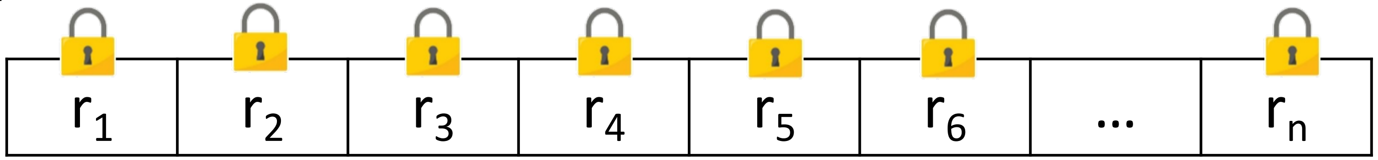
1. Build NIZK Proofs for NP in Hidden Bits Model (HB).
2. Instantiate HB with URS and OWP.

Our Proof Overview

1. Build NIZK Proofs for NP in Z-Tamperable Hidden Bits Model (ZHB).
2. Instantiate ZHB with random oracle.

NIZK Proofs in Hidden Bits Model

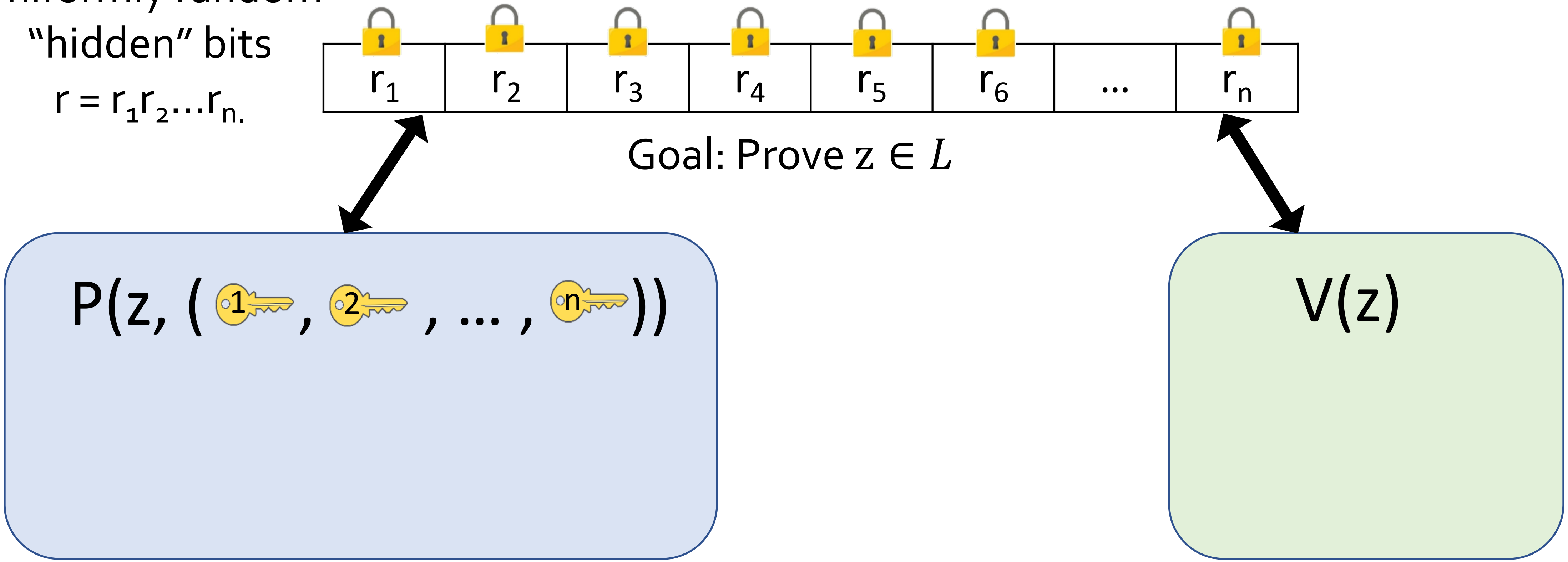
Uniformly random
"hidden" bits
$r = r_1 r_2 \ldots r_n$.



Goal: Prove $z \in L$

NIZK Proofs in Hidden Bits Model

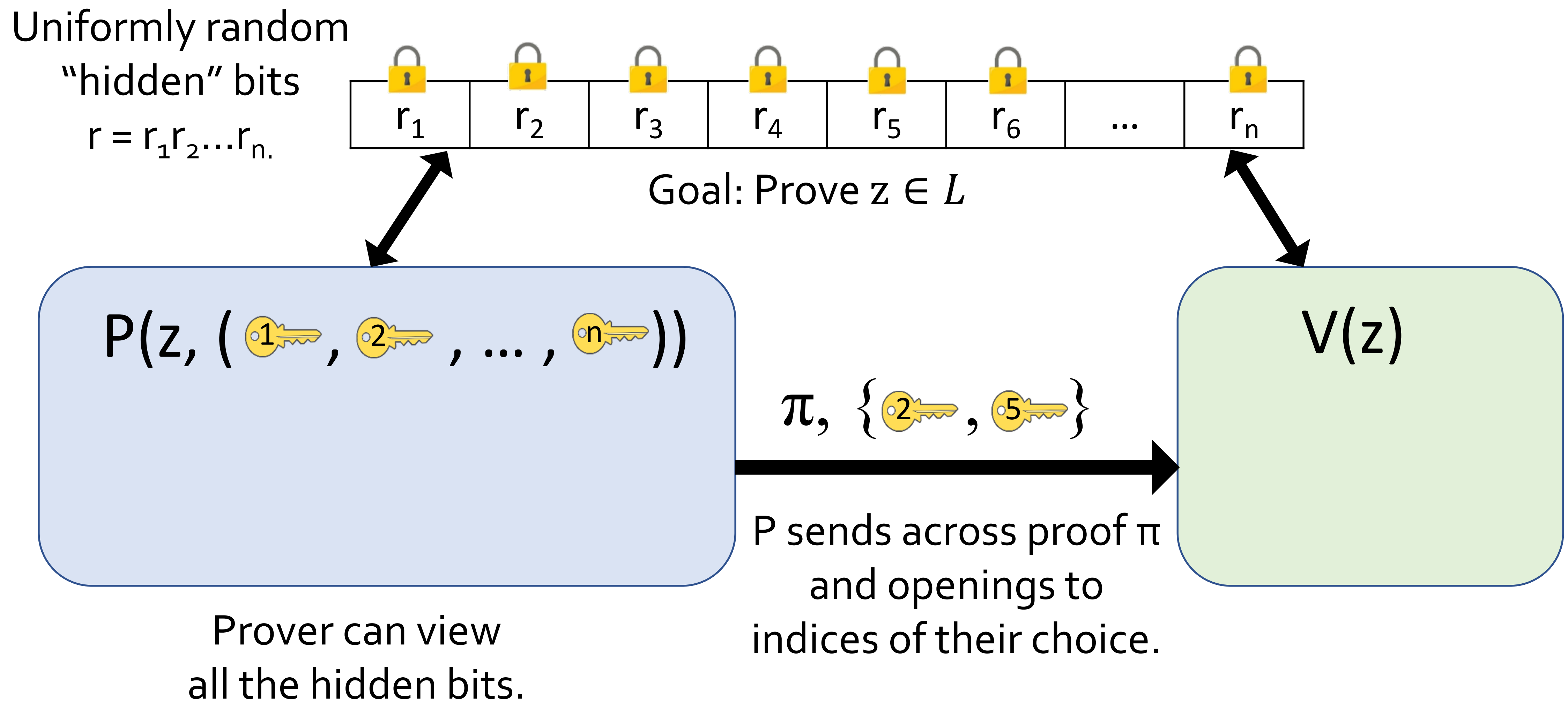
Uniformly random
"hidden" bits
$r = r_1 r_2 \ldots r_n.$



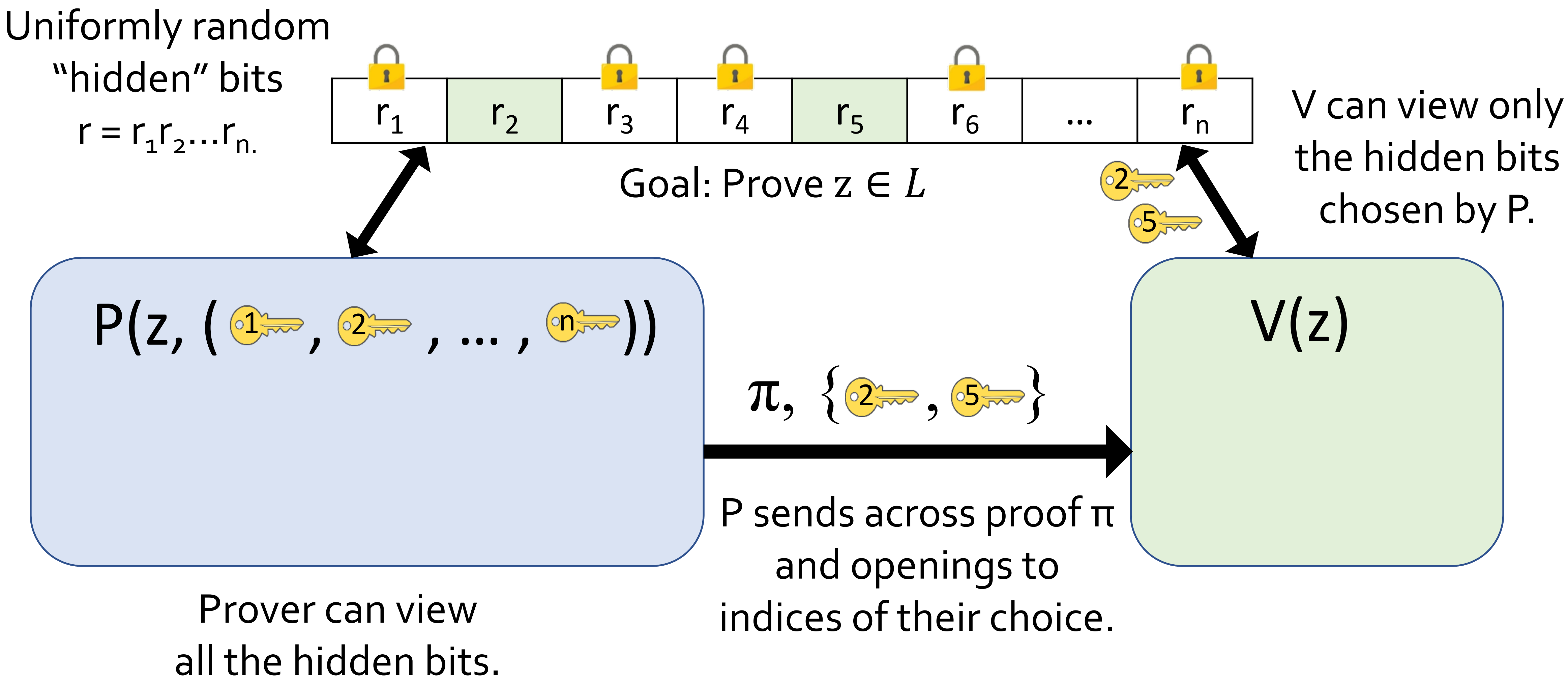
Prover can view
all the hidden bits.

Verifier can't view
the hidden bits.

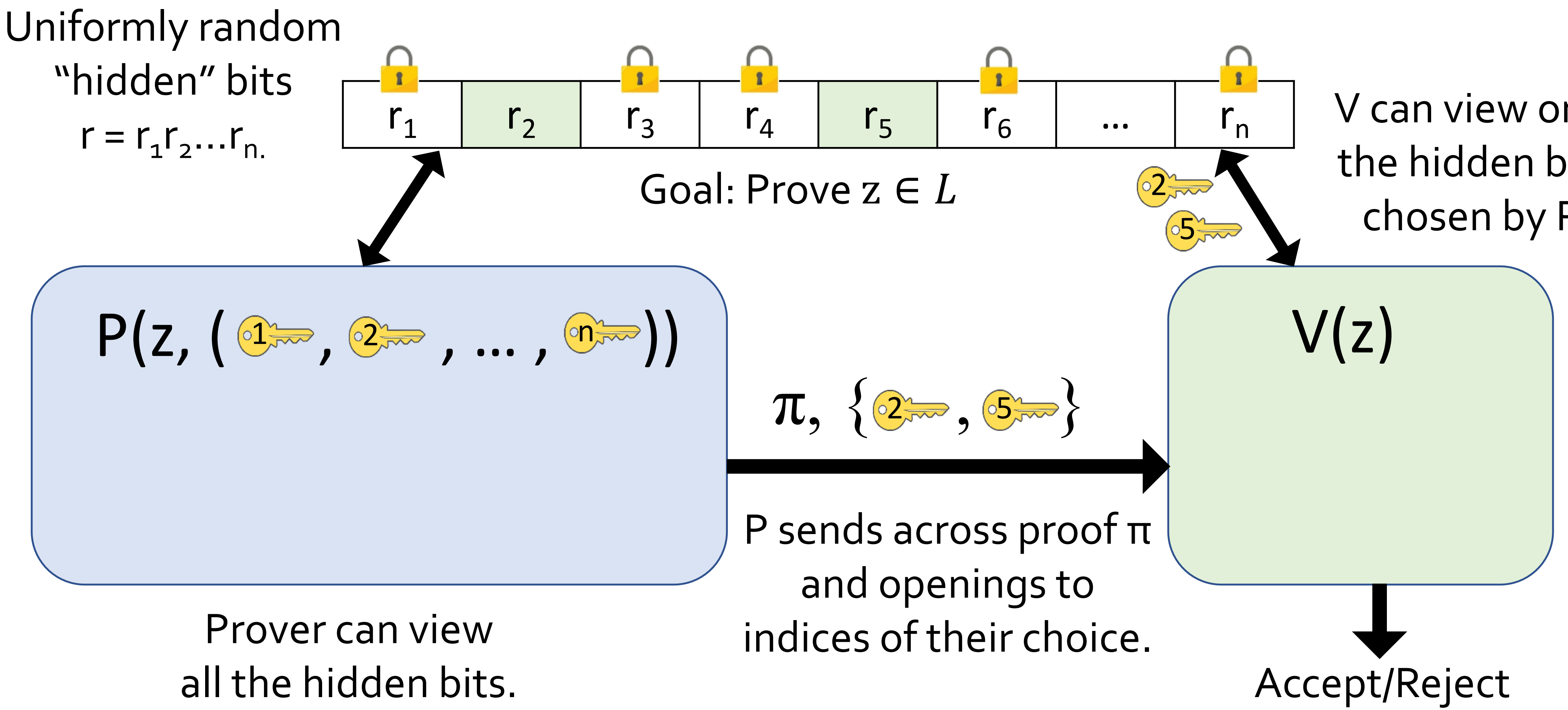
NIZK Proofs in Hidden Bits Model



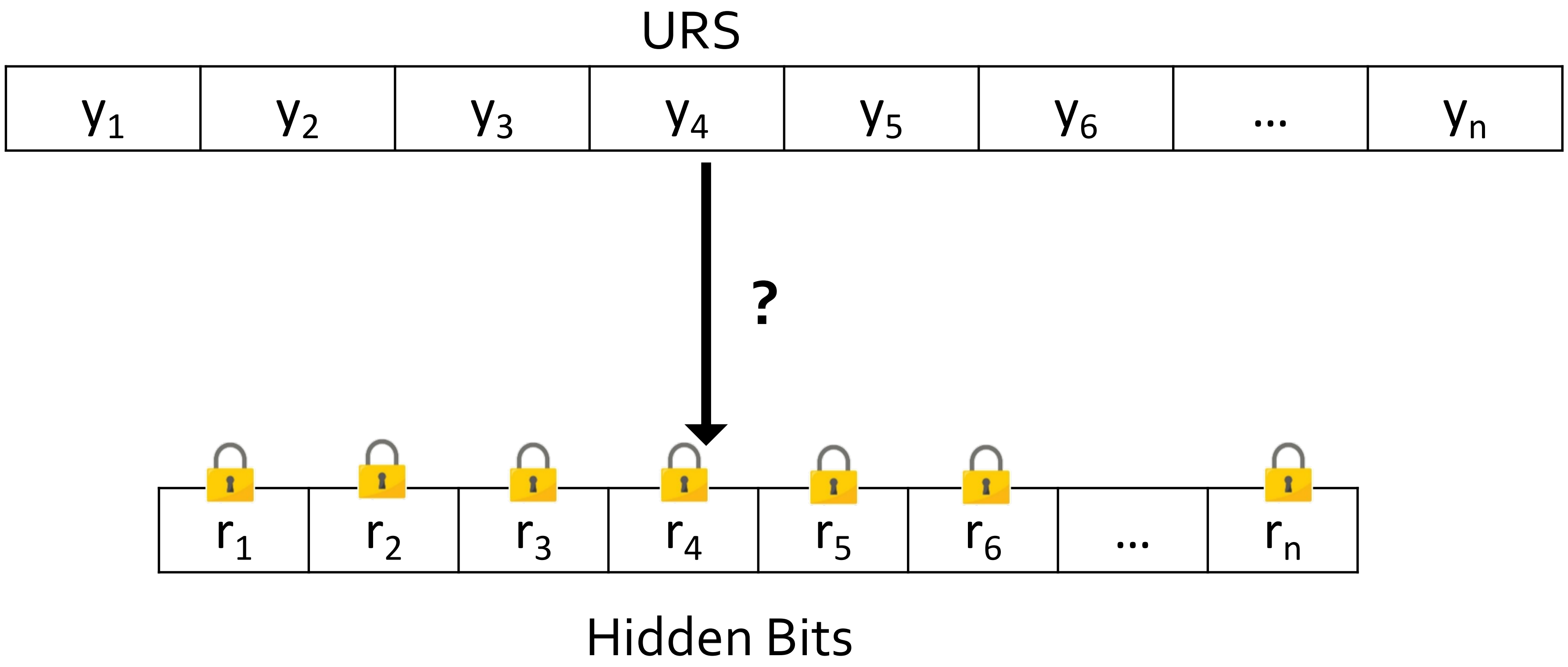
NIZK Proofs in Hidden Bits Model



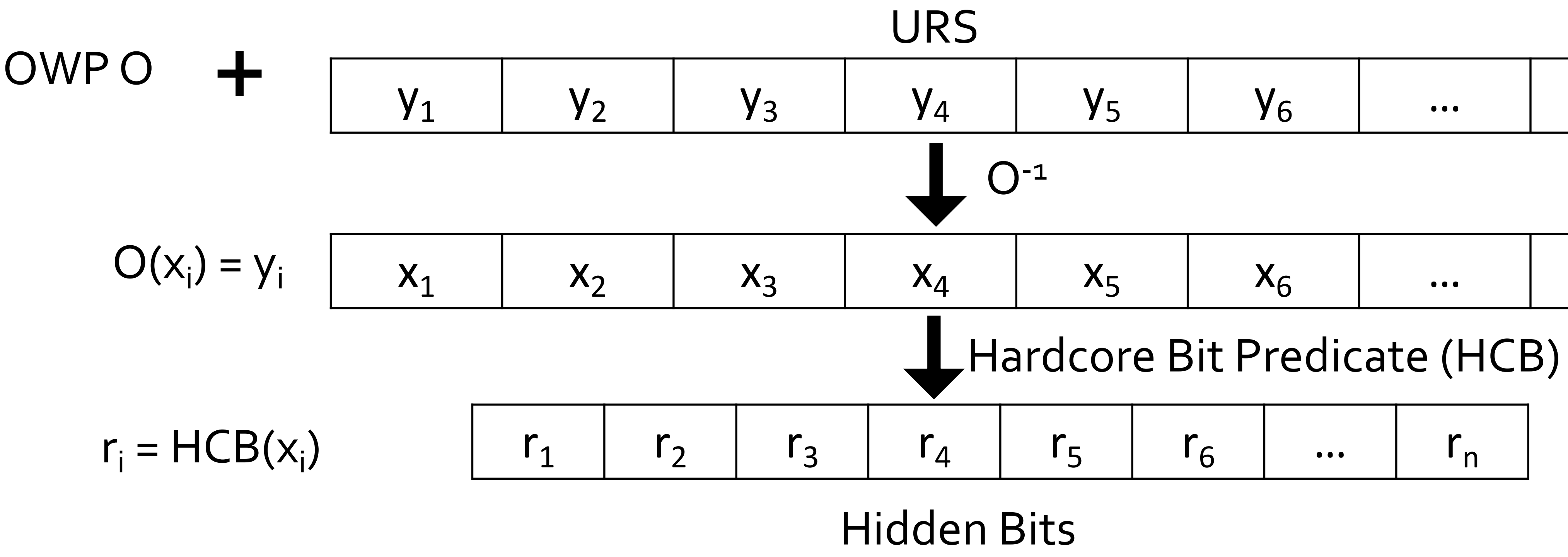
NIZK Proofs in Hidden Bits Model



Instantiating the HB model with URS!



Instantiating the HB model with URS!



Instantiating the HB model with URS!

OWP O **+** URS

y_1	y_2	y_3	y_4	y_5	y_6	…	y_n

$\downarrow O^{-1}$

$O(x_i) = y_i$

x_1	x_2	x_3	x_4	x_5	x_6	…	x_n

$\downarrow$ Hardcore Bit Predicate (HCB)

$r_i = HCB(x_i)$

r_1	r_2	r_3	r_4	r_5	r_6	…	r_n

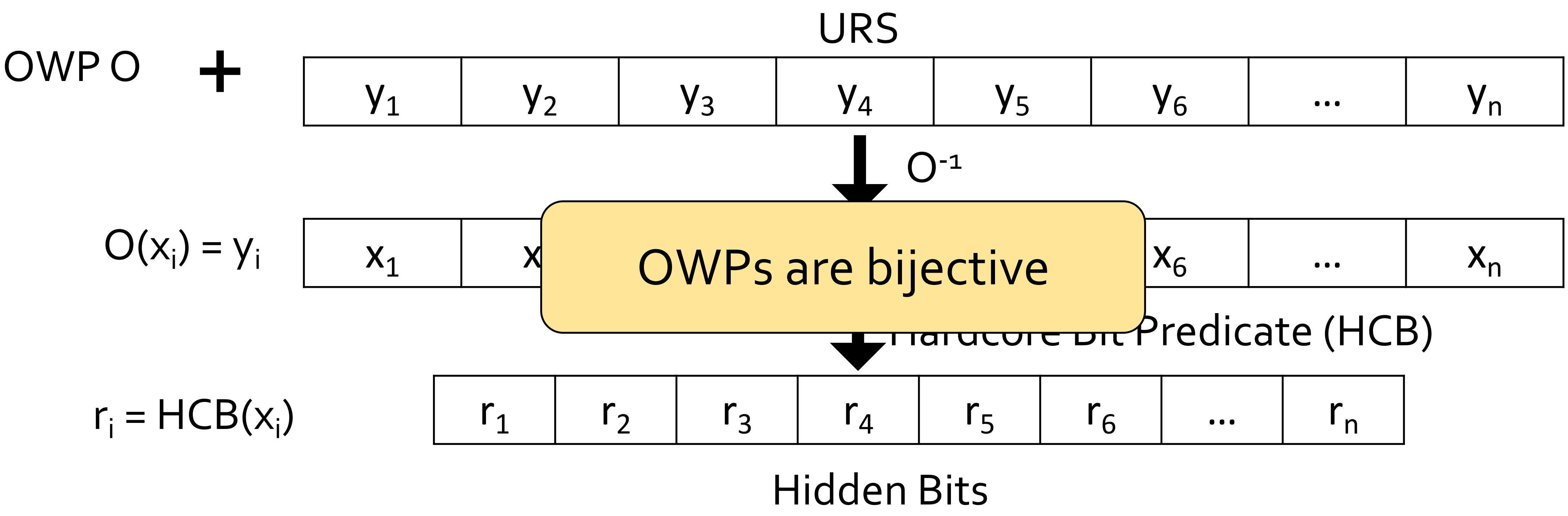
Hidden Bits

O hard to invert:
V can't learn r_i from just y_i.

= x_i

- (Unbounded) P can compute x_i by brute force.
- V can check that $y_i = O(x_i)$ and compute $r_i = HCB(x_i)$.

Instantiating the HB model with URS!



O hard to invert: V can't learn r_i from just y_i.

= x_i

- (Unbounded) P can compute x_i by brute force.
- V can check that $y_i = O(x_i)$ and compute $r_i = HCB(x_i)$.

Instantiating the HB model with Random Oracle and URS?

Random Oracle O +

URS

y_1	y_2	y_3	y_4	y_5	y_6	…	y_n

$\downarrow O^{-1}$

$O(x_i) = y_i$

x_1	-	x_3	x_4	-	x_6	…	x_n

$\downarrow$ Hardcore Bit Predicate (HCB)

$r_i = HCB(x_i)$

r_1	?	r_3	r_4	?	r_6	…	r_n

Hidden Bits

Problem: y_i might not have a preimage.
Lose completeness!

Instantiating the HB model with Random Oracle and URS?

Random Oracle O **+** URS

y_1	y_2	y_3	y_4	y_5	y_6	…	y_n

$\downarrow O^{-1}$

$O(x_i) = y_i$

x_1	-	x_{31}, x_{32}	x_4	-	x_{61}, x_{62}, x_{63}	…	x_n

$\downarrow$ Hardcore Bit Predicate (HCB)

$r_i = HCB(x_i)$

r_1	?	r_{31} or r_{32}	r_4	?	r_{61} or r_{62} or r_{63}	…	r_n

Hidden Bits

Problem: y_i might not have a preimage.
Lose completeness!

Problem: y_i might have multiple preimages.
P can pick whichever he wants so r_i not uniformly random. Lose soundness!

Instantiating the HB model with Random Oracle and URS?

Random Oracle O +

URS

y_1	y_2	y_3	y_4	y_5	y_6	…	y_n

$O(x_i) = y_i$

x_1	$_{61}, x_{62}, x_{63}$	…	x_n

> **New Idea:**
> Set $r_i = 1$ dependent on whether or not y_i has a preimage.

t Predicate (HCB)

$r_i = HCB(x_i)$

r_1	?	r_{31} or r_{32}	r_4	?	r_{61} or r_{62} or r_{63}	…	r_n

Hidden Bits

Problem: y_i might not have a preimage.
Lose completeness!

Problem: y_i might have multiple preimages.
P can pick whichever he wants so r_i not uniformly random. Lose soundness!

Instantiating the HB model with Random Oracle and URS?

Random Oracle O +

URS

y_1	y_2	y_3	y_4	y_5	y_6	…	y_n

$$r_i = \begin{cases} 0 & if\ \exists x_i, O(x_i) = y_i \\ 1 & else \end{cases}$$

r_1	r_2	r_3	r_4	r_5	r_6	…	r_n

Hidden Bits

Instantiating the HB model with Random Oracle and URS?

Random Oracle O **+**

URS

y_1	y_2	y_3	y_4	y_5	y_6	…	y_n

$$r_i = \begin{cases} 0 & if\ \exists x_i, O(x_i) = y_i \\ 1 & else \end{cases}$$

r_1	r_2	r_3	r_4	r_5	r_6	…	r_n

Hidden Bits

V can't easily determine if y_i has a preimage or not.

Instantiating the HB model with Random Oracle and URS?

Random Oracle O **+**

URS

y_1	y_2	y_3	y_4	y_5	y_6	…	y_n

$$r_i = \begin{cases} 0 & if\ \exists x_i, O(x_i) = y_i \\ 1 & else \end{cases}$$

r_1	r_2	r_3	r_4	r_5	r_6	…	r_n

Hidden Bits

V can't easily determine if y_i has a preimage or not.

 = (preimage x_i) or ("has no preimage")

- (Unbounded) P can compute x_i by brute force.
- V can check that $y_i = O(x_i)$.

Instantiating the HB model with Random Oracle and URS?

Random Oracle O **+**

URS

y_1	y_2	y_3	y_4	y_5	y_6	…	y_n

$$r_i = \begin{cases} 0 & if\ \exists x_i, O(x_i) = y_i \\ 1 & else \end{cases}$$

y_i has no preimage:
Prover must set r_i =1

r_1	r_2	r_3	r_4	r_5	r_6	…	r_n

Hidden Bits

V can't easily determine if y_i has a preimage or not.

= (preimage x_i) or ("has no preimage")

- (Unbounded) P can compute x_i by brute force.
- V can check that y_i = $O(x_i)$.

Instantiating the HB model with Random Oracle and URS?

Random Oracle O **+**

URS

y_1	y_2	y_3	y_4	y_5	y_6	…	y_n

$$r_i = \begin{cases} 0 & if\ \exists x_i, O(x_i) = y_i \\ 1 & else \end{cases}$$

r_1	r_2	r_3	r_4	r_5	r_6	…	r_n

Hidden Bits

Problem: P can claim y_i has no preimage even when it does!

V can't easily determine if y_i has a preimage or not.

= (preimage x_i) or $\boxed{\text{("has no preimage")}}$

- (Unbounded) P can compute x_i by brute force.
- V can check that y_i = $O(x_i)$.

Instantiating the HB model with Random Oracle and URS?

Random Oracle O **+**

URS

y_1	y_2	y_3	y_4	y_5	y_6	…	y_n

$O(x_i) = y_i$

Problem: P can claim y_i has no preimage even when it does!

Solution:
Define new model that captures this behavior.

r_n

Hidden Bits

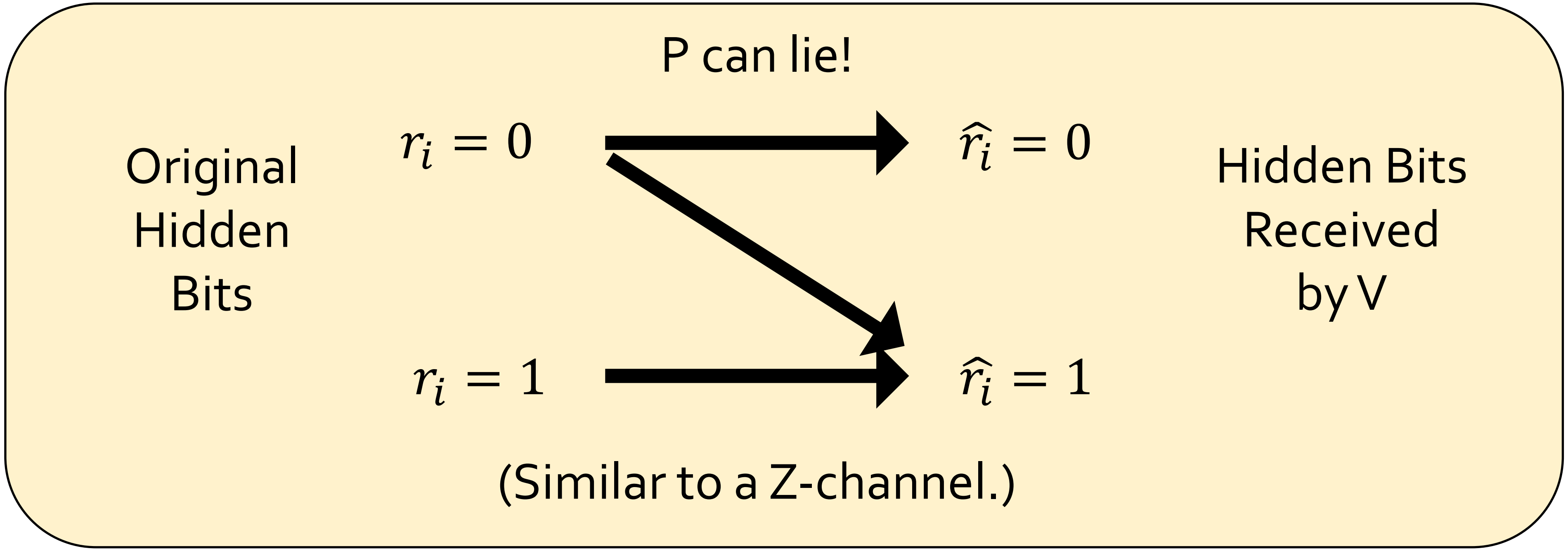
V can't easily determine if y_i has a preimage or not.

i = (preimage x_i) or ("has no preimage")

- (Unbounded) P can compute x_i by brute force.
- V can check that $y_i = O(x_i)$.

NIZK Proofs in Z-Tamperable Hidden Bits Model

- Same as Hidden Bits model except that P can lie about r_i if $r_i = 0$.
  - Captures ability of dishonest P to lie by saying “has no preimage” when there is actually a preimage.
  - Honest P never lies about r_i.



NIZK Proofs in Z-Tamperable Hidden Bits Model

- Observation: P can't lie too much.
  - V can run statistical tests on distribution of r to see if there are too many 1's.

NIZK Proofs in Z-Tamperable Hidden Bits Model

- Observation: P can't lie too much.
  - V can run statistical tests on distribution of r to see if there are too many 1's.

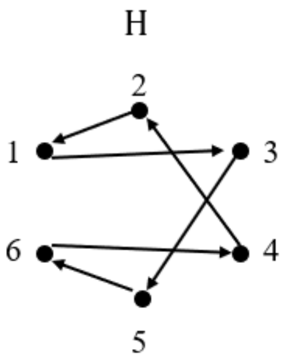
- Key Idea: Add careful statistical tests to construction of NIZK proofs in the (regular) Hidden Bits model [FLS90].

  - Step 1: Carefully change parameters to make bad behavior more detectable.
  - Step 2: This requires statistical tests.
  - Step 3: Our analysis shows that any significant amount of cheating using the ZHB model will be caught with high probability.

Warmup: Prove that G is Hamiltonian.

Assume: Hidden bit string r represents adjacency matrix of cycle graph H.

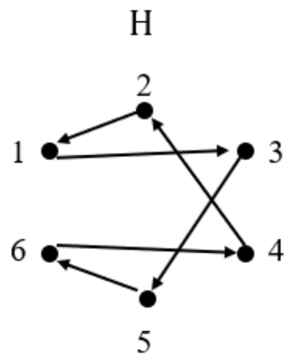
	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	1	0	0	0	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0



Warmup: Prove that G is Hamiltonian.

Assume: Hidden bit string r represents adjacency matrix of cycle graph H.

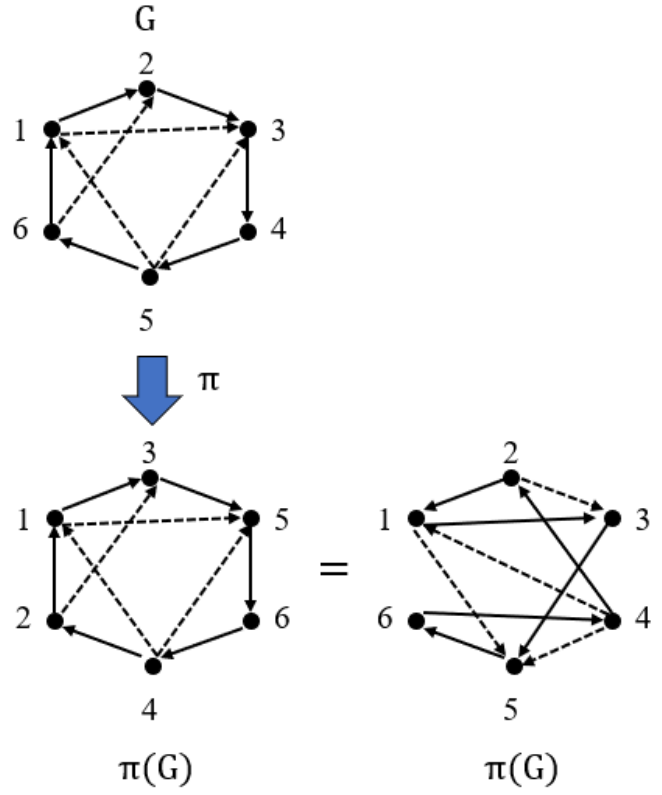
	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	1	0	0	0	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0



1. P finds permutation π such that $\pi(C_G) = H$ where C_G is Hamiltonian cycle of G.

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	1	0	1	0	0	1
6	1	1	0	0	0	0

	1	2	3	4	5	6
1	0	0	1	0	1	0
2	1	0	1	0	0	0
3	0	0	0	0	1	0
4	1	1	0	0	1	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0

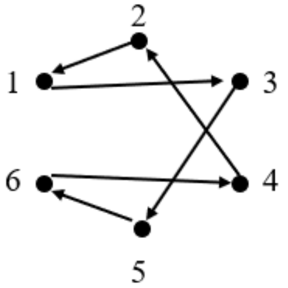


Warmup: Prove that G is Hamiltonian.

Assume: Hidden bit string r represents adjacency matrix of cycle graph H.

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	1	0	0	0	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0

H



Reveal non-edges of π(G)

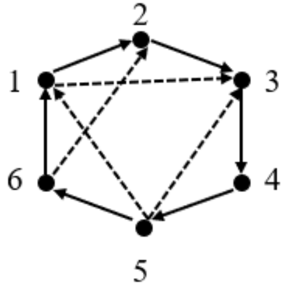
	1	2	3	4	5	6
1	0	0		0		0
2		0		0	0	0
3	0	0	0	0		0
4			0	0		0
5	0	0	0	0	0	
6	0	0	0		0	0

2. Show that H is a subgraph of π(G) by opening non-edges of π(G) in H to 0.

1. P finds permutation π such that $\pi(C_G) = H$ where C_G is Hamiltonian cycle of G.

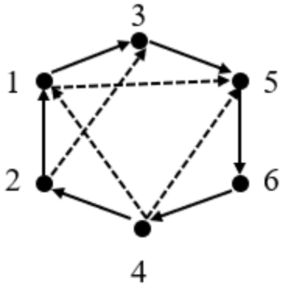
	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	1	0	1	0	0	1
6	1	1	0	0	0	0

G



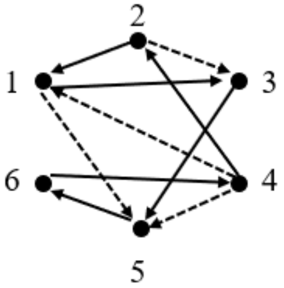
π

	1	2	3	4	5	6
1	0	0	1	0	1	0
2	1	0	1	0	0	0
3	0	0	0	0	1	0
4	1	1	0	0	1	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0



π(G)

=



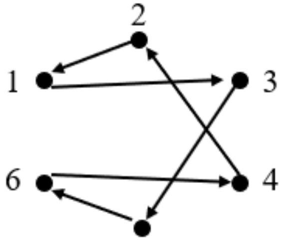
π(G)

Warmup: Prove that G is Hamiltonian.

Assume: Hidden bit string r represents adjacency matrix of cycle graph H.

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	1	0	0	0	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0

H



Reveal non-edges of π(G)

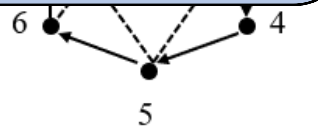
	1	2	3	4	5	6
1	0	0		0		0
2		0		0	0	0
3	0	0	0	0		0
4			0	0		0
5	0	0	0	0	0	
6	0	0	0		0	0

- P sends proof: (π, openings)
- V checks that all non-edges of π(G) in H opened to 0.

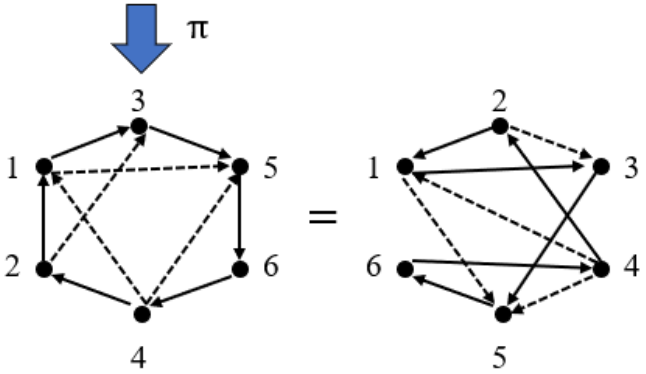
2. Show that H is a subgraph of π(G) by opening non-edges of π(G) in H to 0.

1. P finds permutation π such that $\pi(C_G) = H$ where C_G is Hamiltonian cycle of G.

4	0	0	0	0	1	0
5	1	0	1	0	0	1
6	1	1	0	0	0	0



	1	2	3	4	5	6
1	0	0	1	0	1	0
2	1	0	1	0	0	0
3	0	0	0	0	1	0
4	1	1	0	0	1	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0



π(G) π(G)

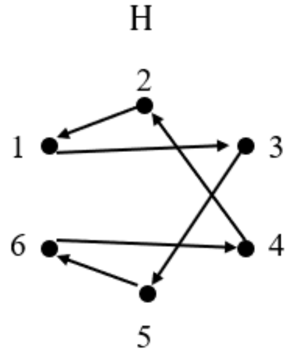
Warmup: Prove that G is Hamiltonian.

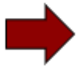
Assume: Hidden bit string r represents adjacency m

Completeness: Always exists a permutation that works if G is Hamiltonian.

1. P finds permutation π such that $\pi(C_G) = H$ where C_G is Hamiltonian cycle of G.

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	1	0	0	0	0
				0	0	1
				1	0	0

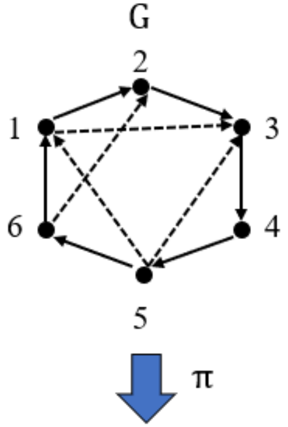




Reveal non-edges of π(G)

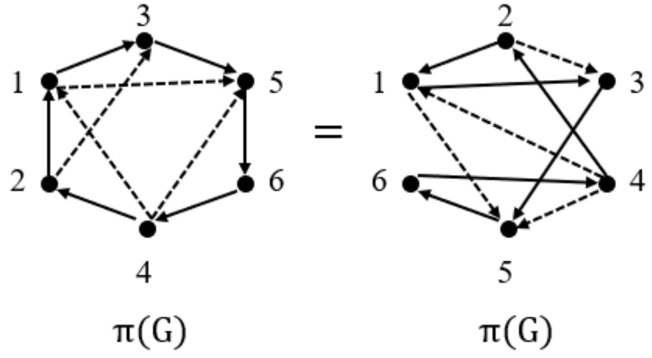
	1	2	3	4	5	6
1	0	0		0		0
2		0		0	0	0
3	0	0	0	0		0
4			0	0		0
5	0	0	0	0	0	
6	0	0	0		0	0

			3	4	5	6
1	0	1	1	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	1	0	1	0	0	1
6	1	1	0	0	0	0



2. Show that H is a subgraph of π(G) by opening non-edges of π(G) in H to 0.

	1	2	3	4	5	6
1	0	0	1	0	1	0
2	1	0	1	0	0	0
3	0	0	0	0	1	0
4	1	1	0	0	1	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0

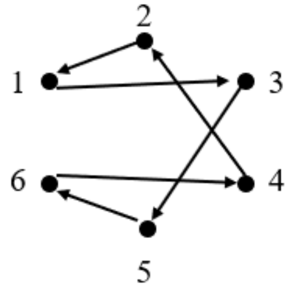


Warmup: Prove that G is Hamiltonian.

Assume: Hidden bit string r represents adjacency matrix of cycle graph H.

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	1	0	0	0	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0

H

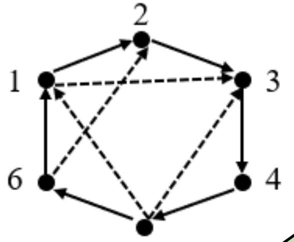


Reveal non-edges of π(G)

	1	2	3	4	5	6
1	0	0		0		0
2		0		0	0	0
3	0	0	0	0		0
4			0	0		0
5	0	0	0	0	0	
6	0	0	0		0	0

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	1	0	1	0	0	1
6	1	1	0	0	0	0

G



2. Show that H is a subgraph of π(G) by opening non-edges of π(G) in H to 0.

1. P finds permutation π such that $\pi(C_G) = H$ where C_G is Hamiltonian cycle of G.

	1	2	3	4	5	6
1	0	0	1	0	1	0
2	1	0	1	0	0	0
3	0	0	0	0	1	0
4	1	1	0	0	1	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0

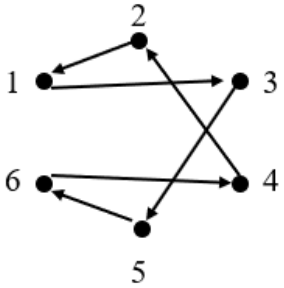
Soundness: V knows that every non-edge in G corresponds to a non-edge in H => every edge in H corresponds to an edge in G => G must have a Hamiltonian cycle.

Warmup: Prove that G is Hamiltonian.

Assume: Hidden bit string r represents adjacency matrix of cycle graph H.

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	1	0	0	0	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0

H

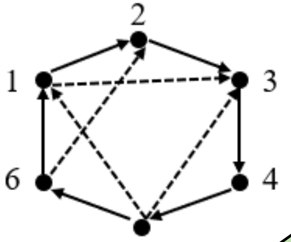


Reveal non-edges of π(G)

	1	2	3	4	5	6
1	0	0		0		0
2		0		0	0	0
3	0	0	0	0		0
4			0	0		0
5	0	0	0	0	0	
6	0	0	0		0	0

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	1	0	1	0	0	1
6	1	1	0	0	0	0

G



2. Show that H is a subgraph of π(G) by opening non-edges of π(G) in H to 0.

1. P finds permutation π such that $\pi(C_G)$ = H where

Ham

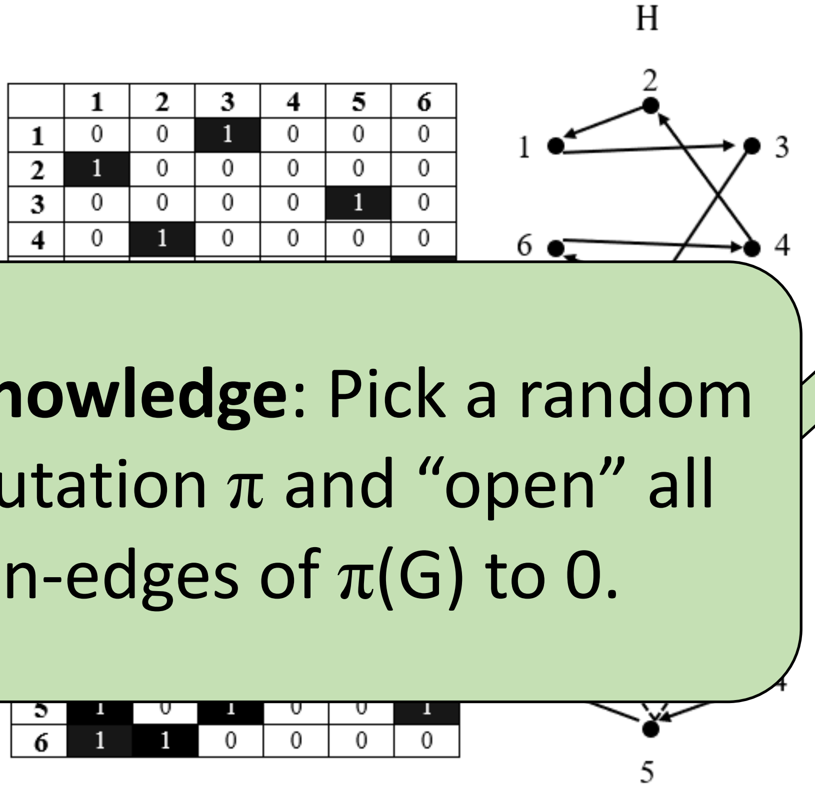
5	6
1	0
0	
1	0
0	1
0	0

Also works in **Z-Tamperable Hidden Bits** model since flipping 0's to 1's only adds edges to H!

Soundness: V knows that every non-edge in G corresponds to a non-edge in H => every edge in H corresponds to an edge in G => G must have a Hamiltonian cycle.

Warmup: Prove that G is Hamiltonian.

Assume: Hidden bit string r represents adjacency matrix of cycle graph H.



Zero Knowledge: Pick a random permutation π and "open" all non-edges of $\pi(G)$ to 0.

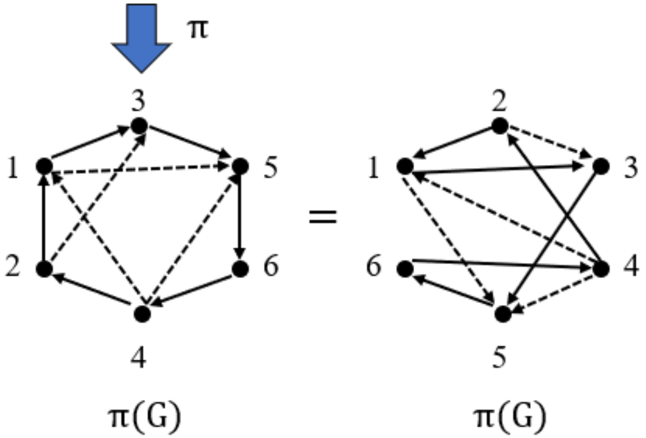
Reveal non-edges of $\pi(G)$

	1	2	3	4	5	6
1	0	0		0		0
2		0		0	0	0
3	0	0	0	0		0
4			0	0		0
5	0	0	0	0	0	
6	0	0	0		0	0

2. Show that H is a subgraph of $\pi(G)$ by opening non-edges of $\pi(G)$ in H to 0.

1. P finds permutation π such that $\pi(C_G)$ = H where C_G is Hamiltonian cycle of G.

	1	2	3	4	5	6
1	0	0	1	0	1	0
2	1	0	1	0	0	0
3	0	0	0	0	1	0
4	1	1	0	0	1	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0



[FLS90] NIZK Proofs in Hidden Bits Model

- Warmup: Assume hidden bit string r is a random cycle graph.
  - Works in Z-Tamperable Hidden Bits Model!

[FLS90] NIZK Proofs in Hidden Bits Model

- Warmup: Assume hidden bit string r is a random cycle graph.
  - Works in Z-Tamperable Hidden Bits Model!
- What if r is not a cycle?
  - Random $n \times n$ graph unlikely to be a cycle.
  - [FLS90] Use r to sample graphs such that w.h.p. at least one is a cycle graph.

[FLS90] Sampling Cycle Graphs

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.

$M^{(i)}$

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

$S^{(i)}$

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	1	0	0	0	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0

[FLS90] Sampling Cycle Graphs

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

$M^{(i)}$

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

$S^{(i)}$

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	1	0	0	0	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0

Reveal rows and columns not in $S^{(i)}$, and reveal all non-edges of $\pi(G)$ in $S^{(i)}$

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0		0	0	0	0		0	0
0		0	0	0		0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0		0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0			0	0	0	0	0	0	0		0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0		0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0		0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Reveal non-edges of $\pi(G)$

	1	2	3	4	5	6
1	0	0		0		0
2		0		0	0	0
3	0	0	0	0		0
4			0	0		0
5	0	0	0	0	0	
6	0	0	0		0	0

[FLS90] Sampling Cycle Graphs

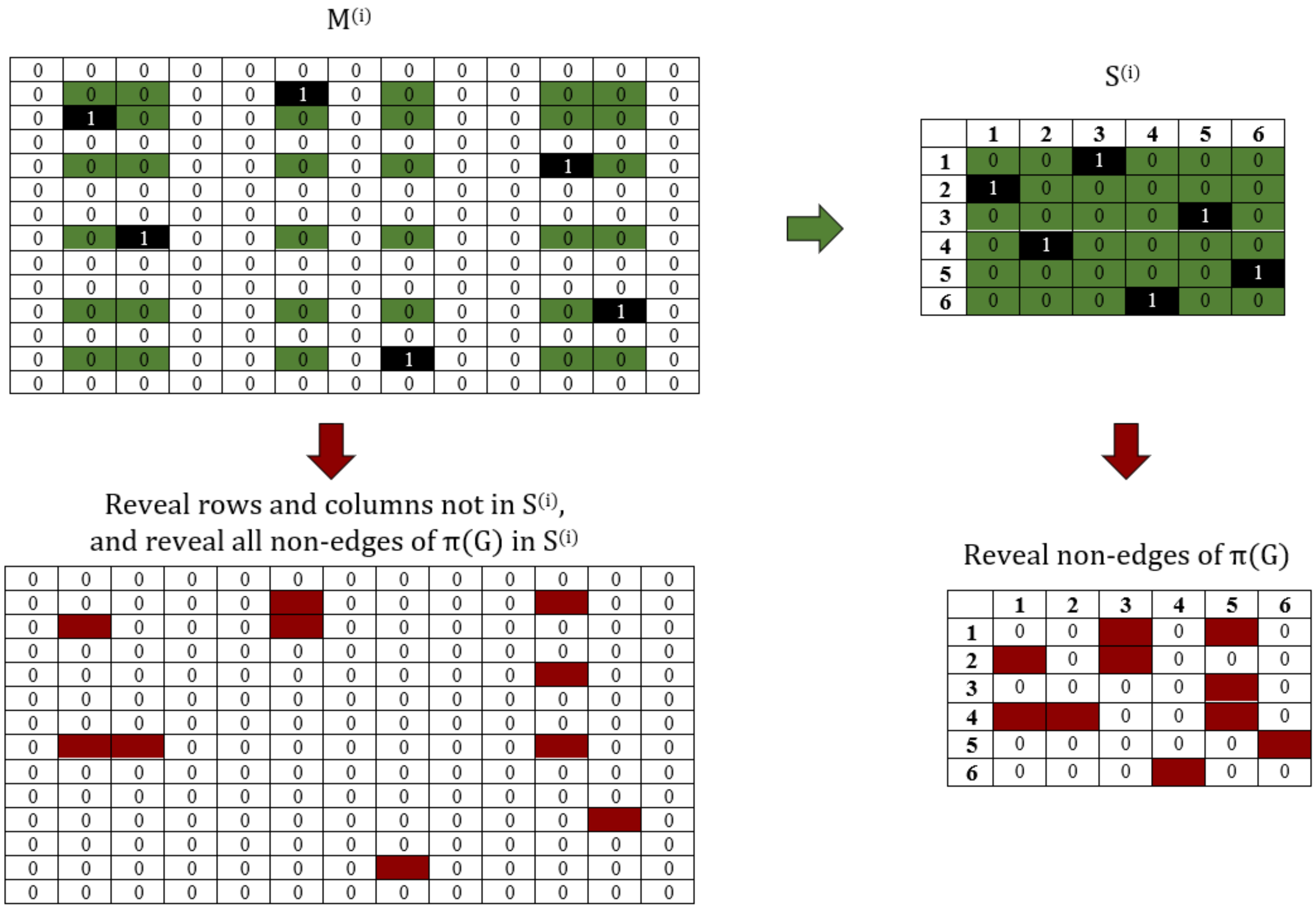
Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.



[FLS90] Sampling Cycle Graphs

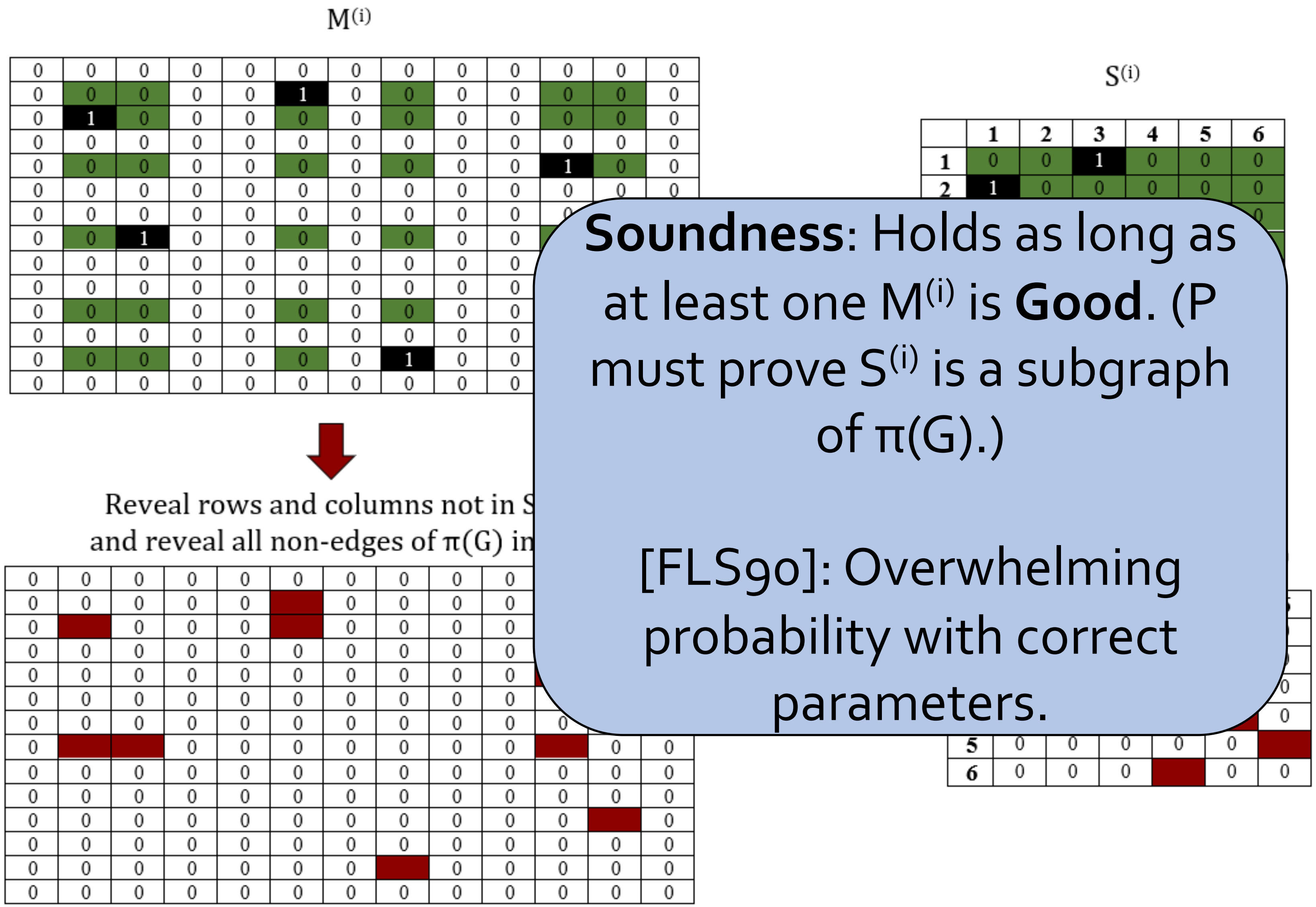
Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.



Soundness: Holds as long as at least one $M^{(i)}$ is **Good**. (P must prove $S^{(i)}$ is a subgraph of $\pi(G)$.)

[FLS90]: Overwhelming probability with correct parameters.

[FLS90] Sampling Cycle Graphs

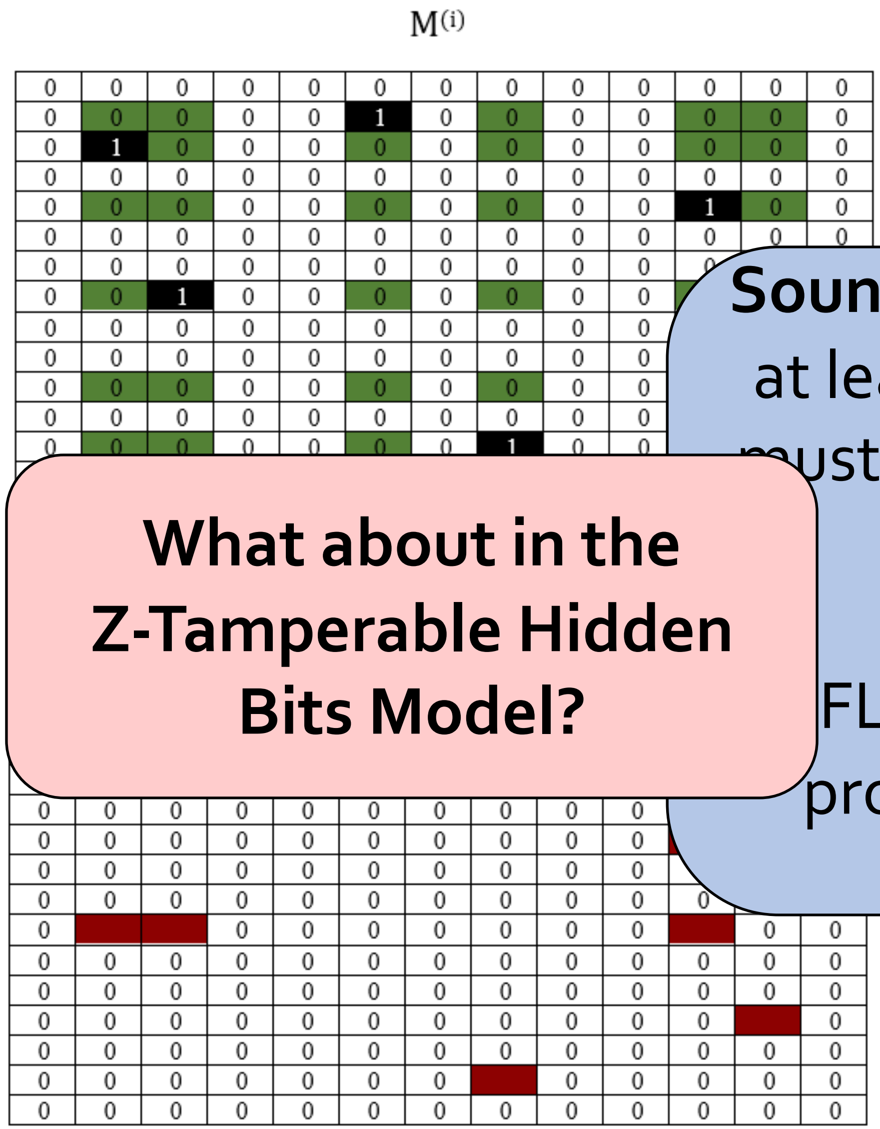
Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

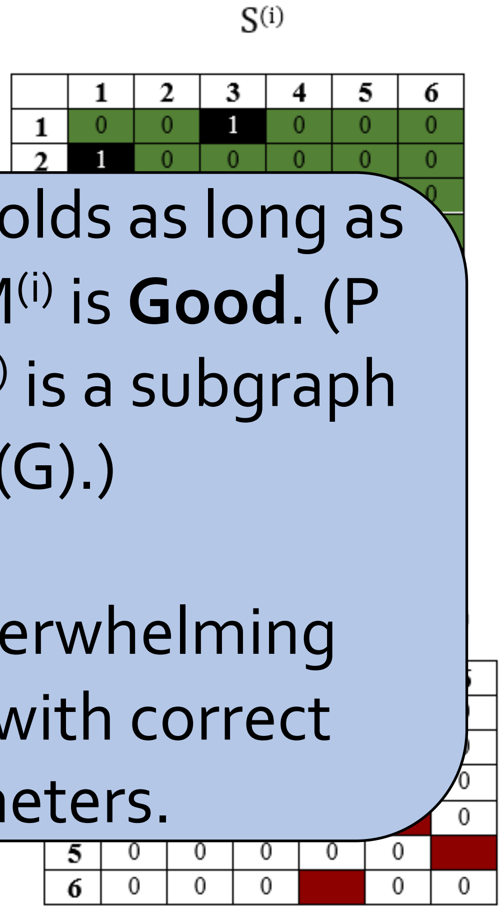
Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.





What about in the Z-Tamperable Hidden Bits Model?

Soundness: Holds as long as at least one $M^{(i)}$ is **Good**. (P must prove $S^{(i)}$ is a subgraph of $\pi(G)$.)

[FLS90]: Overwhelming probability with correct parameters.

[FLS90] Sampling Cycle Graphs

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Recall: P can add '1's.

[FLS90] Sampling Cycle Graphs

Sample $n^c \times n^c$ matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Recall: P can add '1's.

Problem: P can turn $M^{(i)}$ from **Good** to **Bad** by adding '1's.

[FLS90] Sampling Cycle Graphs

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Recall: P can add '1's.

Case 1: Good $M^{(i)}$
- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Problem: P can turn $M^{(i)}$ from **Good** to **Bad** by adding '1's.

Case 2: Bad $M^{(i)}$
- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Problem: P can pretend a **Bad** $M^{(i)}$ is **Good** as long as it contains a subgraph of $\pi(G)$.

[FLS90] Sampling Cycle Graphs

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Recall: P can add '1's but cannot remove them.

Problem: P can turn $M^{(i)}$ from **Good** to **Bad** by adding '1's.

P can only add '1's: All such $M^{(i)}$ have at least n+1 '1's.

Problem: P can pretend a **Bad** $M^{(i)}$ is **Good** as long as it contains a subgraph of π(G).

[FLS90] Sampling Cycle Graphs

Sample $n^c \times n^c$ matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$
- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$
- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Recall: P can add '1's but cannot remove them.

Problem: P can turn $M^{(i)}$ from **Good** to **Bad** by adding '1's.

P can only add '1's: All such $M^{(i)}$ have at least n+1 '1's.

Problem: P can pretend a **Bad** $M^{(i)}$ is **Good** as long as it contains a subgraph of $\pi(G)$.

'1's of $M^{(i)}$ must be contained in an $n \times n$ submatrix.

[FLS90] Sampling Cycle Graphs

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Key Insight: To cheat, P must convert every **Good** $M^{(i)}$ to **Bad** $M^{(i)}$

Problem: P can turn $M^{(i)}$ from **Good** to **Bad** by adding ‘1’s.

P can only add ‘1’s: All such $M^{(i)}$ have at least n+1 ‘1’s.

Problem: P can pretend a **Bad** $M^{(i)}$ is **Good** as long as it contains a subgraph of π(G).

‘1’s of $M^{(i)}$ must be contained in an $n \times n$ submatrix.

[FLS90] Sampling Cycle Graphs

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$
- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$
- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Key Insight: If c is large, matrices become very sparse => Most matrices with at least n+1 '1's, do not fit all these '1's into an $n \times n$ submatrix!

Key Insight: To cheat, P must convert every **Good** $M^{(i)}$ to **Bad** $M^{(i)}$

Problem: P can turn $M^{(i)}$ from **Good** to **Bad** by adding '1's.

P can only add '1's: All such $M^{(i)}$ have at least n+1 '1's.

Problem: P can pretend a **Bad** $M^{(i)}$ is **Good** as long as it contains a subgraph of π(G).

'1's of $M^{(i)}$ must be contained in an $n \times n$ submatrix.

[FLS90] Sampling Cycle Graphs

Sample $n^c \times n^c$ matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$
- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$
- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Solution: V checks for expected number of matrices with at least n+1 '1's.

Problem: P can turn $M^{(i)}$ from **Good** to **Bad** by adding '1's.

P can only add '1's: All such $M^{(i)}$ have at least n+1 '1's.

Problem: P can pretend a **Bad** $M^{(i)}$ is **Good** as long as it contains a subgraph of π(G).

'1's of $M^{(i)}$ must be contained in an $n \times n$ submatrix.

[FLS90] Sampling Cycle Graphs

Sample $n^c \times n^c$ matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$
- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$
- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Solution: V checks for expected number of matrices with at least n+1 ‘1’s.

Problem: P can turn $M^{(i)}$ from **Good** to **Bad** by adding ‘1’s.

Cheating P must add all **Good** $M^{(i)}$ to count.

Problem: P can pretend a **Bad** $M^{(i)}$ is **Good** as long as it contains a subgraph of π(G).

Not enough **Bad** matrices that fit in an $n \times n$ submatrix to make up for it.

[FLS90] Sampling Cycle Graphs

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Solution: V checks for expected number of matrices with at least n+1 '1's.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Problem: P can turn $M^{(i)}$ from **Good** to **Bad** by adding '1's.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Problem: P can pretend a **Bad** $M^{(i)}$ is **Good** as long as it contains a subgraph of $\pi(G)$.

Soundness in Z-Tamperable Hidden Bits Model!

NIZK Proofs in Z-Tamperable Hidden Bits Model

- Warmup: Assume hidden bit string r is a random cycle graph.
  - Works in Z-Tamperable Hidden Bits Model!

- What if r is not a cycle?
  - Random $n \times n$ graph unlikely to be a cycle.
  - [FLS90] Use r to sample graphs such that w.h.p. at least one is a cycle graph.
  - **Our Work:** Increase sparsity of matrices and add statistical checks to ensure that P must use at least one cycle graph.

Our Results

Main Theorem

If $UP \not\subseteq RP$, then with probability 1 over the choice of a random oracle O,
$P^O \neq NP^O \cap coNP^O$

NIZK Proofs in Random Oracle Model

There exists an (unbounded-prover) NIZK proof system for NP in the random oracle model.

NIZK Proofs in URS model from δ-Dense-PRHFs

Assuming there exists a δ-Dense-PRHF,

there exists an (unbounded-prover) NIZK proof system for NP in the URS model.

Future Directions

1. Get an unconditional random oracle separation of P and $NP \cap coNP$.

2. Extend our techniques to get more separation results.

3. Instantiate a δ-Dense-PRHF from standard unstructured assumptions.

4. Build *efficient-prover* NIZK proofs from random oracles.

THANK YOU!!!

APPENDIX

Pr[M is Good]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Pr[M is Good]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

1. M has exactly n 1's.

2. These 1's form a permutation submatrix.

3. The permutation is an n-cycle.

Pr[M is Good]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

1. M has exactly n 1's.

By Chebyshev's Inequality,

$$\Pr\left[\#1's \in \left[n-\sqrt{2n}, n+\sqrt{2n}\right]\right] \geq \frac{1}{2}$$

Therefore,

$$\Pr[M\ has\ n\ 1's] \geq \frac{1}{2\sqrt{2n}} \sum_{i=n-\sqrt{2n}}^{n+\sqrt{2n}} \Pr[M\ has\ i\ 1's] \geq \frac{1}{4\sqrt{2n}}$$

2. These 1's form a permutation submatrix.

3. The permutation is an n-cycle.

Pr[M is Good]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

1. M has exactly n 1's.

By Chebyshev's Inequality,

$$\Pr\left[\#1's \in \left[n-\sqrt{2n}, n+\sqrt{2n}\right]\right] \geq \frac{1}{2}$$

Therefore,

$$\Pr[M\ has\ n\ 1's] \geq \frac{1}{2\sqrt{2n}} \sum_{i=n-\sqrt{2n}}^{n+\sqrt{2n}} \Pr[M\ has\ i\ 1's] \geq \frac{1}{4\sqrt{2n}}$$

2. These 1's form a permutation submatrix.

Affected by c!

$$\Pr[1's\ form\ a\ permutation]$$
$$\geq 1 - \Pr[two\ 1's\ in\ same\ column] - \Pr[two\ 1's\ in\ same\ row]$$
$$\geq 1 - O\left(\frac{1}{n^2}\right)$$

3. The permutation is an n-cycle.

Pr[M is Good]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$
- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$
- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

1. M has exactly n 1's.

By Chebyshev's Inequality,

$$\Pr\left[\#1's \in \left[n-\sqrt{2n}, n+\sqrt{2n}\right]\right] \geq \frac{1}{2}$$

Therefore,

$$\Pr[M\ has\ n\ 1's] \geq \frac{1}{2\sqrt{2n}} \sum_{i=n-\sqrt{2n}}^{n+\sqrt{2n}} \Pr[M\ has\ i\ 1's] \geq \frac{1}{4\sqrt{2n}}$$

2. These 1's form a permutation submatrix.

$$\Pr[1's\ form\ a\ permutation]$$
$$\geq 1 - \Pr[two\ 1's\ in\ same\ column] - \Pr[two\ 1's\ in\ same\ row]$$
$$\geq 1 - O\left(\frac{1}{n^2}\right)$$

3. The permutation is an n-cycle.

$$\Pr[\,n - cycle \mid permutation] = \frac{1}{n}$$

$\Pr[\text{M is Good}] = \Omega\left(\frac{1}{n^{1.5}}\right)$

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

1. M has exactly n 1's.

By Chebyshev's Inequality,

$$\Pr\left[\#1's \in \left[n - \sqrt{2n}, n + \sqrt{2n}\right]\right] \geq \frac{1}{2}$$

Therefore,

$$\Pr[M\ has\ n\ 1's] \geq \frac{1}{2\sqrt{2n}} \sum_{i=n-\sqrt{2n}}^{n+\sqrt{2n}} \Pr[M\ has\ i\ 1's] \geq \frac{1}{4\sqrt{2n}}$$

2. These 1's form a permutation submatrix.

$$\Pr[1's\ form\ a\ permutation]$$
$$\geq 1 - \Pr[two\ 1's\ in\ same\ column] - \Pr[two\ 1's\ in\ same\ row]$$
$$\geq 1 - O\left(\frac{1}{n^2}\right)$$

3. The permutation is an n-cycle.

$$\Pr[\,n - cycle \mid permutation] = \frac{1}{n}$$

Pr[P' can pretend Bad M is Good, and M has ≥ n+1 1's]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Pr[P' can pretend Bad M is Good, and M has $\geq$ n+1 1's]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$
- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$
- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Let p_n = Pr[M has $\geq$ n+ 1 1's]

Idea: Compute probability that a matrix with n+1 1's has

1. #1's $\leq$ 2n

2. 1's form a permutation submatrix

Pr[P' can pretend Bad M is Good, and M has $\geq$ n+1 1's]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Let p_n = Pr[M has $\geq$ n+ 1 1's]

Idea: Compute probability that a matrix with n+1 1's has

1. #1's $\leq$ 2n

2. 1's form a permutation submatrix

P can't cheat on these because 1's do not fit in an n x n submatrix!

Pr[P' can pretend Bad M is Good, and M has $\geq$ n+1 1's]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$
- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$
- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Let p_n = Pr[M has $\geq$ n+ 1 1's]

Idea: Compute probability that a matrix with n+1 1's has

1. #1's $\leq$ 2n

   By Chernoff Bound:
   $\Pr[\#1's \leq 2n \mid \#1's > n] \geq 1 - negl(n)/p_n$

2. 1's form a permutation submatrix

P can't cheat on these because 1's do not fit in an n x n submatrix!

Pr[P' can pretend Bad M is Good, and M has $\geq$ n+1 1's]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Let p_n = Pr[M has $\geq$ n+ 1 1's]

Idea: Compute probability that a matrix with n+1 1's has

1. #1's $\leq$ 2n

   By Chernoff Bound:
   $\Pr[\#1's \leq 2n \mid \#1's > n] \geq 1 - negl(n)/p_n$

2. 1's form a permutation submatrix

   Similar to before:
   $\Pr[1's\ form\ a\ permutation] \geq 1 - O\left(\frac{1}{n^2}\right)$

P can't cheat on these because 1's do not fit in an n x n submatrix!

Pr[P' can pretend Bad M is Good, and M has ≥ n+1 1's]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

Let p_n = Pr[M has ≥ n+ 1 1's]

Idea: Compute probability that a matrix with n+1 1's has $\leq \left(1 - O\left(\frac{1}{n^2}\right)\right) + negl(n)/p_n$

1. #1's ≤ 2n

By Chernoff Bound:
$\Pr[\#1's \leq 2n \mid \#1's > n] \geq 1 - negl(n)/p_n$

2. 1's form a permutation submatrix

Similar to before:

$\Pr[1's\ form\ a\ permutation] \geq 1 - O\left(\frac{1}{n^2}\right)$

P can't cheat on these because 1's do not fit in an n x n submatrix!

Pr[P' can pretend Bad M is Good, and M has ≥ n+1 1's]

Set c = 4.

Sample n^c x n^c matrices $M^{(i)}$ such that each element of $M^{(i)}$ is 1 with probability $1/n^{2c-1}$.

Case 1: Good $M^{(i)}$

- $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a cycle graph on n nodes, and $M^{(i)}$ is 0 everywhere else.
- P uses submatrix $S^{(i)}$ for protocol and reveals all other rows and columns to be 0.

Case 2: Bad $M^{(i)}$

- All other $M^{(i)}$.
- P reveals all of $M^{(i)}$ to prove it was Bad.

$$\leq p_n \cdot O\left(\frac{1}{n^2}\right)$$

Let p_n = Pr[M has ≥ n+ 1 1's]

Idea: Compute probability that a matrix with n+1 1's has

$$\leq \left(1 - O\left(\frac{1}{n^2}\right)\right) + negl(n)/p_n$$

1. #1's ≤ 2n

By Chernoff Bound:

$$\Pr[\#1's \leq 2n \mid \#1's > n] \geq 1 - negl(n)/p_n$$

2. 1's form a permutation submatrix

Similar to before:

$$\Pr[1's\ form\ a\ permutation] \geq 1 - O\left(\frac{1}{n^2}\right)$$

P can't cheat on these because 1's do not fit in an n x n submatrix!