Proving Computation Integrity

CS 276: Introduction to Cryptography

Sanjam Garg

April 8, 2026

Overview

1. NIZKs from Trapdoor Permutations

2. NIZK for NP: Graph Hamiltonicity

NIZK: Definition

Definition 1 (NIZK Proof System)

A NIZK for language L is efficient algorithms (K, P, V):

1. **Key generation**: $\sigma \leftarrow K(1^k)$ generates the CRS.
2. **Prover**: $\pi \leftarrow P(\sigma, x, \omega)$ produces a proof (given witness ω).
3. **Verifier**: $V(\sigma, x, \pi) \in \{0, 1\}$ accepts or rejects.

NIZK: Properties

Completeness

$\forall x \in L,\ \forall \omega \in R_L(x)$:

$$\Pr[\sigma \leftarrow K(1^k),\ \pi \leftarrow P(\sigma, x, \omega) : V(\sigma, x, \pi) = 1] = 1.$$

NIZK: Properties

Completeness

$\forall\, x \in L,\ \forall\, \omega \in R_L(x)$:

$$\Pr[\sigma \leftarrow K(1^k),\ \pi \leftarrow P(\sigma, x, \omega) : V(\sigma, x, \pi) = 1] = 1.$$

Non-adaptive soundness

$\forall\, x \notin L$:

$$\Pr[\sigma \leftarrow K(1^k) : \exists\, \pi \text{ s.t. } V(\sigma, x, \pi) = 1] = \mathsf{negl}(k).$$

The cheating prover must fix x **before** seeing σ.

NIZK: Properties

Completeness

$\forall\, x \in L,\ \forall\, \omega \in R_L(x)$:

$$\Pr[\sigma \leftarrow K(1^k),\ \pi \leftarrow P(\sigma, x, \omega) : V(\sigma, x, \pi) = 1] = 1.$$

Non-adaptive soundness

$\forall\, x \notin L$:

$$\Pr[\sigma \leftarrow K(1^k) : \exists\, \pi \text{ s.t. } V(\sigma, x, \pi) = 1] = \mathsf{negl}(k).$$

The cheating prover must fix x **before** seeing σ.

Adaptive soundness (stronger)

$$\Pr[\sigma \leftarrow K(1^k) : \exists\, (x, \pi) \text{ s.t. } x \notin L \text{ and } V(\sigma, x, \pi) = 1] = \mathsf{negl}(k).$$

The cheating prover may choose x **after** seeing σ.

NIZK: Zero-Knowledge Property

Non-adaptive zero-knowledge

$\exists$ PPT simulator S such that $\forall\, x \in L$, $\omega \in R_L(x)$, the following are computationally indistinguishable:

Real:

1. $\sigma \leftarrow K(1^k)$
2. $\pi \leftarrow P(\sigma, x, \omega)$
3. Output (σ, π)

Simulated:

1. $(\sigma, \pi) \leftarrow S(1^k, x)$
2. Output (σ, π)

NIZK: Zero-Knowledge Property

Non-adaptive zero-knowledge

$\exists$ PPT simulator S such that $\forall\, x \in L$, $\omega \in R_L(x)$, the following are computationally indistinguishable:

Real:

1. $\sigma \leftarrow K(1^k)$
2. $\pi \leftarrow P(\sigma, x, \omega)$
3. Output (σ, π)

Simulated:

1. $(\sigma, \pi) \leftarrow S(1^k, x)$
2. Output (σ, π)

Key point

The simulator produces **both** σ and π together. If S could not generate σ, the definition would be trivial (verifier could run S itself instead of interacting with P).

Adaptive ZK

Adaptive zero-knowledge

$\exists$ PPT simulator (S_1, S_2) such that $\forall$ PPT $\mathcal{A}$, the following are computationally indistinguishable:

Real:

1. $\sigma \leftarrow K(1^k)$
2. $(x, \omega) \leftarrow \mathcal{A}(\sigma)$ with $(x, \omega) \in R_L$
3. $\pi \leftarrow P(\sigma, x, \omega)$
4. Output (σ, x, π)

Simulated:

1. $(\sigma, \tau) \leftarrow S_1(1^k)$
2. $(x, \omega) \leftarrow \mathcal{A}(\sigma)$
3. $\pi \leftarrow S_2(\sigma, x, \tau)$
4. Output (σ, x, π)

Adaptive ZK

Adaptive zero-knowledge

$\exists$ PPT simulator (S_1, S_2) such that $\forall$ PPT $\mathcal{A}$, the following are computationally indistinguishable:

Real:

1. $\sigma \leftarrow K(1^k)$
2. $(x, \omega) \leftarrow \mathcal{A}(\sigma)$ with $(x, \omega) \in R_L$
3. $\pi \leftarrow P(\sigma, x, \omega)$
4. Output (σ, x, π)

Simulated:

1. $(\sigma, \tau) \leftarrow S_1(1^k)$
2. $(x, \omega) \leftarrow \mathcal{A}(\sigma)$
3. $\pi \leftarrow S_2(\sigma, x, \tau)$
4. Output (σ, x, π)

Here τ is local state ("trapdoor") passed from S_1 to S_2: the simulator sets up σ in a way that lets it later produce fake proofs for any x chosen by $\mathcal{A}$.

Trapdoor One-Way Permutations

Definition 2 (Trapdoor OWP)

A collection $\{f_i : D_i \to D_i\}_{i \in I}$ where $D_i \subset \{0,1\}^{|i|}$ with:

1. $\exists$ PPT G: $G(1^k)$ outputs (i, t_i) where $i \in I \cap \{0,1\}^k$.
2. Easy to sample from D_i given i.
3. f_i is easy to compute but hard to invert.
4. f_i is a permutation.
5. $\exists$ PPT A: $A(i, y, t_i) \in f_i^{-1}(y)$ (efficient inversion with trapdoor t_i).

Trapdoor One-Way Permutations

Definition 2 (Trapdoor OWP)

A collection $\{f_i : D_i \to D_i\}_{i \in I}$ where $D_i \subset \{0,1\}^{|i|}$ with:

1. $\exists$ PPT G: $G(1^k)$ outputs (i, t_i) where $i \in I \cap \{0,1\}^k$.
2. Easy to sample from D_i given i.
3. f_i is easy to compute but hard to invert.
4. f_i is a permutation.
5. $\exists$ PPT A: $A(i, y, t_i) \in f_i^{-1}(y)$ (efficient inversion with trapdoor t_i).

Example: RSA

$G(1^k) = ((N, e), d)$ with $N = pq$ (primes), $\gcd(e, \phi(N)) = 1$, $d = e^{-1} \bmod \phi(N)$.

$$F_{N,e}(x) = x^e \bmod N, \qquad A((N, e), y, d) = y^d \bmod N.$$

Correctness: $(x^e)^d = x \pmod N$.

NIZK in the Hidden-Bit Model

Definition 3 (Hidden-Bit Model (HBM))

A NIZK in the HBM for statement x (witness ω) is (K_H, P_H, V_H):

1. $r \leftarrow K_H(1^k)$: generates a hidden random string (ℓ bits).
2. $(I, \phi) \leftarrow P_H(r, x, \omega)$: prover reads all of r, selects indices $I \subseteq [\ell]$ to reveal, and produces proof ϕ.
3. $V_H(I, \{r_i\}_{i \in I}, x, \phi)$: verifier sees only revealed bits, accepts or rejects.

Must satisfy completeness, soundness, and zero-knowledge.

NIZK in the Hidden-Bit Model

Definition 3 (Hidden-Bit Model (HBM))

A NIZK in the HBM for statement x (witness ω) is (K_H, P_H, V_H):

1. $r \leftarrow K_H(1^k)$: generates a hidden random string (ℓ bits).
2. $(I, \phi) \leftarrow P_H(r, x, \omega)$: prover reads all of r, selects indices $I \subseteq [\ell]$ to reveal, and produces proof ϕ.
3. $V_H(I, \{r_i\}_{i \in I}, x, \phi)$: verifier sees only revealed bits, accepts or rejects.

Must satisfy completeness, soundness, and zero-knowledge.

Intuition

The prover can selectively reveal parts of a random string — like clouds obscuring a random sky, where the prover chooses which clouds to clear.

HBM $\Rightarrow$ Standard Model via Trapdoor OWPs

Theorem 4

Given a NIZK (P_H, V_H) in the HBM, we can construct a NIZK (P, V) in the CRS model using trapdoor one-way permutations.

HBM $\Rightarrow$ Standard Model via Trapdoor OWPs

Theorem 4

Given a NIZK (P_H, V_H) in the HBM, we can construct a NIZK (P, V) in the CRS model using trapdoor one-way permutations.

Setup

Let $\mathcal{F}$ be a family of trapdoor OWPs with hard-core bit $B(\cdot)$. CRS: $\sigma = \sigma_1 \cdots \sigma_\ell$ (each $\sigma_i \in \{0,1\}^k$). Amplify HBM soundness to $\leq 2^{-2k}$ by repetition.

HBM $\Rightarrow$ CRS: Construction

Prover $P(\sigma, x, \omega)$

1. Sample trapdoor OWP: $(f, f^{-1}) \leftarrow \mathcal{F}(1^k)$.
2. Compute $\alpha_i = f^{-1}(\sigma_i)$ for all $i \in [\ell]$.
3. Compute hidden bits $r_i = B(\alpha_i)$ for all $i \in [\ell]$. Let $r := r_1 \cdots r_\ell$.
4. Run the HBM prover: $(I, \phi) \leftarrow P_H(r, x, \omega)$.
5. Send $(f, I, \{\alpha_i\}_{i \in I}, \phi)$ to verifier.

HBM $\Rightarrow$ CRS: Construction

Prover $P(\sigma, x, \omega)$

1. Sample trapdoor OWP: $(f, f^{-1}) \leftarrow \mathcal{F}(1^k)$.
2. Compute $\alpha_i = f^{-1}(\sigma_i)$ for all $i \in [\ell]$.
3. Compute hidden bits $r_i = B(\alpha_i)$ for all $i \in [\ell]$. Let $r := r_1 \cdots r_\ell$.
4. Run the HBM prover: $(I, \phi) \leftarrow P_H(r, x, \omega)$.
5. Send $(f, I, \{\alpha_i\}_{i \in I}, \phi)$ to verifier.

Verifier $V(\sigma, x, f, I, \{\alpha_i\}_{i \in I}, \phi)$

1. Confirm $f \in \mathcal{F}$ and $f(\alpha_i) = \sigma_i$ for all $i \in I$.
2. Compute revealed bits $r_i = B(\alpha_i)$ for all $i \in I$.
3. Output $V_H(I, \{r_i\}_{i \in I}, x, \phi)$.

HBM $\Rightarrow$ CRS: Completeness and Soundness

Completeness

Honest prover: $\alpha_i = f^{-1}(\sigma_i)$ uniformly random (since f is a permutation), so $r_i = B(\alpha_i)$ is unbiased. Reduces exactly to HBM distributions.

HBM $\Rightarrow$ CRS: Completeness and Soundness

Completeness

Honest prover: $\alpha_i = f^{-1}(\sigma_i)$ uniformly random (since f is a permutation), so $r_i = B(\alpha_i)$ is unbiased. Reduces exactly to HBM distributions.

Soundness

For a fixed $f = f_0$: r_i's are uniformly random, so by HBM soundness:

$$\Pr_{\sigma}[P^* \text{ cheats using } f_0] \leq 2^{-2k}.$$

A cheating P^* may choose f adversarially, but $|\mathcal{F}| = 2^k$, so by union bound:

$$\Pr_{\sigma}[\exists\, f \in \mathcal{F} \text{ s.t. } P^* \text{ cheats}] \leq 2^k \cdot 2^{-2k} = 2^{-k}.$$

HBM $\Rightarrow$ CRS: Completeness and Soundness

Completeness

Honest prover: $\alpha_i = f^{-1}(\sigma_i)$ uniformly random (since f is a permutation), so $r_i = B(\alpha_i)$ is unbiased. Reduces exactly to HBM distributions.

Soundness

For a fixed $f = f_0$: r_i's are uniformly random, so by HBM soundness:

$$\Pr_{\sigma}[P^* \text{ cheats using } f_0] \leq 2^{-2k}.$$

A cheating P^* may choose f adversarially, but $|\mathcal{F}| = 2^k$, so by union bound:

$$\Pr_{\sigma}[\exists\, f \in \mathcal{F} \text{ s.t. } P^* \text{ cheats}] \leq 2^k \cdot 2^{-2k} = 2^{-k}.$$

Important: V must verify $f \in \mathcal{F}$. If f is not a permutation, $f^{-1}(\sigma_i)$ can be multi-valued, letting the prover "explain" σ_i with different α_i's having different hard-core bits.

HBM $\Rightarrow$ CRS: Zero-Knowledge (Hybrid Argument)

Hybrids (differences from previous in red)

- H_0 Real protocol: sample σ, compute $\alpha_i = f^{-1}(\sigma_i)$, $r_i = B(\alpha_i)$, run P_H.
- H_1 Sample α_i randomly, set $\sigma_i = f(\alpha_i)$. Identical distribution (f is a permutation).
- H_2 Sample $r_i \leftarrow \{0,1\}$ first, then α_i with $B(\alpha_i) = r_i$. Identical (reordering).
- H_3 For $i \notin I$: $\sigma_i \leftarrow \{0,1\}^k$ uniformly (instead of $f(\alpha_i)$). Indistinguishable by hard-core bit security.
- H_4 Replace P_H with HBM simulator S_H. Indistinguishable by HBM zero-knowledge.

HBM $\Rightarrow$ CRS: Zero-Knowledge (Hybrid Argument)

Hybrids (differences from previous in red)

- H_0 Real protocol: sample σ, compute $\alpha_i = f^{-1}(\sigma_i)$, $r_i = B(\alpha_i)$, run P_H.
- H_1 Sample α_i randomly, set $\sigma_i = f(\alpha_i)$. Identical distribution (f is a permutation).
- H_2 Sample $r_i \leftarrow \{0,1\}$ first, then α_i with $B(\alpha_i) = r_i$. Identical (reordering).
- H_3 For $i \notin I$: $\sigma_i \leftarrow \{0,1\}^k$ uniformly (instead of $f(\alpha_i)$). Indistinguishable by hard-core bit security.
- H_4 Replace P_H with HBM simulator S_H. Indistinguishable by HBM zero-knowledge.

H_4 is the NIZK simulator: it does **not** use the witness. So (P, V) is zero-knowledge. $\square$

$H_2 \approx_c H_3$: Why Unrevealed Bits Look Random

Key claim

For a fixed known bit r: $\{f(B^{-1}(r))\} \approx_c \{f(B^{-1}(\bar{r}))\}$, where randomness is over sampling B^{-1}.

$H_2 \approx_c H_3$: Why Unrevealed Bits Look Random

Key claim

For a fixed known bit r: $\{f(B^{-1}(r))\} \approx_c \{f(B^{-1}(\bar{r}))\}$, where randomness is over sampling B^{-1}.

Proof

Distinguishing the above $\equiv$ guessing the hard-core bit from $f(\alpha)$. Therefore:

$$\{f(B^{-1}(r))\} \approx_c \mathcal{U}_k$$

since $\mathcal{U}_k$ can be generated by sampling random b, then outputting $f(B^{-1}(b))$ (using that f is a permutation).

$H_2 \approx_c H_3$: Why Unrevealed Bits Look Random

Key claim

For a fixed known bit r: $\{f(B^{-1}(r))\} \approx_c \{f(B^{-1}(\bar{r}))\}$, where randomness is over sampling B^{-1}.

Proof

Distinguishing the above $\equiv$ guessing the hard-core bit from $f(\alpha)$. Therefore:

$$\{f(B^{-1}(r))\} \approx_c \mathcal{U}_k$$

since $\mathcal{U}_k$ can be generated by sampling random b, then outputting $f(B^{-1}(b))$ (using that f is a permutation).

This justifies replacing $\sigma_i = f(\alpha_i)$ with random for $i \notin I$ in hybrid H_3.

Graph Hamiltonicity

Definition 5 (Hamiltonian Graph)

A graph $G = (V, E)$ with $|V| = n$ is Hamiltonian if there exist distinct $x_1, \ldots, x_n \in V$ such that $(x_i, x_{i+1}) \in E$ for all i, and $(x_n, x_1) \in E$.

Graph Hamiltonicity

Definition 5 (Hamiltonian Graph)

A graph $G = (V, E)$ with $|V| = n$ is Hamiltonian if there exist distinct $x_1, \ldots, x_n \in V$ such that $(x_i, x_{i+1}) \in E$ for all i, and $(x_n, x_1) \in E$.

Fact

Hamiltonicity is **NP-complete**. A NIZK for Hamiltonicity $\Rightarrow$ NIZK for all of NP.

Graph Hamiltonicity

Definition 5 (Hamiltonian Graph)

A graph $G = (V, E)$ with $|V| = n$ is Hamiltonian if there exist distinct $x_1, \ldots, x_n \in V$ such that $(x_i, x_{i+1}) \in E$ for all i, and $(x_n, x_1) \in E$.

Fact

Hamiltonicity is **NP-complete**. A NIZK for Hamiltonicity $\Rightarrow$ NIZK for all of NP.

Matrix representations

- **Adjacency matrix** M_G: $M_G[i, j] = 1$ iff $(i, j) \in E$.
- **Permutation matrix**: boolean $n \times n$ matrix with exactly one 1 per row and column. There are $n!$ of these.
- **Cycle matrix**: permutation matrix corresponding to a single Hamiltonian cycle. There are $(n - 1)!$ of these.

NIZK for Hamiltonicity in the HBM (Ideal Case)

Assumption

Suppose the hidden random string r encodes an $n \times n$ **cycle matrix** M_c.

NIZK for Hamiltonicity in the HBM (Ideal Case)

Assumption

Suppose the hidden random string r encodes an $n \times n$ **cycle matrix** M_c.

Protocol

The prover knows Hamiltonian cycle $x_1, \ldots, x_n$ in G.

1. Find bijection $\phi : V \to \{1, \ldots, n\}$ mapping the Hamiltonian cycle onto the cycle of M_c: $M_c[\phi(x_i), \phi(x_{i+1})] = 1$ for all i.
2. For all **non-edges** $(u, v) \notin E$: reveal $M_c[\phi(u), \phi(v)] = 0$.
3. Send ϕ and the revealed entries to the verifier.

NIZK for Hamiltonicity in the HBM (Ideal Case)

Assumption

Suppose the hidden random string r encodes an $n \times n$ **cycle matrix** M_c.

Protocol

The prover knows Hamiltonian cycle $x_1, \ldots, x_n$ in G.

1. Find bijection $\phi : V \to \{1, \ldots, n\}$ mapping the Hamiltonian cycle onto the cycle of M_c: $M_c[\phi(x_i), \phi(x_{i+1})] = 1$ for all i.
2. For all **non-edges** $(u, v) \notin E$: reveal $M_c[\phi(u), \phi(v)] = 0$.
3. Send ϕ and the revealed entries to the verifier.

Verifier

Check that all revealed entries are 0. This proves: every non-edge of G maps to a non-edge of M_c, so the edges of M_c (which form a Hamiltonian cycle) must all lie within G's edges.

Ideal Case: Properties

Completeness

If P knows a Hamiltonian cycle, it can always find ϕ mapping the cycle onto M_c.

Ideal Case: Properties

Completeness

If P knows a Hamiltonian cycle, it can always find ϕ mapping the cycle onto M_c.

Soundness (perfect)

If G has no Hamiltonian cycle, then for any ϕ, at least one edge of M_c maps to a non-edge of G. That entry is 1 but must be revealed as a non-edge $\Rightarrow$ verifier rejects with probability 1.

Ideal Case: Properties

Completeness

If P knows a Hamiltonian cycle, it can always find ϕ mapping the cycle onto M_c.

Soundness (perfect)

If G has no Hamiltonian cycle, then for any ϕ, at least one edge of M_c maps to a non-edge of G. That entry is 1 but must be revealed as a non-edge $\Rightarrow$ verifier rejects with probability 1.

Zero knowledge

Simulator: sample random bijection ϕ, reveal 0's for all non-edges of $\phi(G)$. The verifier only sees which positions are 0 — this reveals nothing beyond the graph structure (which is public).

From Uniform Random Bits to Cycle Matrices

Problem

In the real HBM, r is a **uniform** random string, not a cycle matrix. We need to extract a cycle matrix from random bits.

From Uniform Random Bits to Cycle Matrices

Problem

In the real HBM, r is a **uniform** random string, not a cycle matrix. We need to extract a cycle matrix from random bits.

Construction

1. Use $\ell = \lceil 3 \log_2 n \rceil \cdot n^4$ random bits. View as n^4 blocks of $\lceil 3 \log_2 n \rceil$ bits.
2. Define $r'_i = 1$ iff all bits in block i are 1. Then $\Pr[r'_i = 1] \approx 1/n^3$.
3. Arrange r' as an $n^2 \times n^2$ matrix M.
4. If M has exactly n ones, all in distinct rows and columns, extract the $n \times n$ submatrix M_c.
5. If M_c is a cycle matrix, use it. Otherwise, reveal all bits and retry.

Probability Analysis (I)

Pr[exactly n ones in M]

Expected number of ones: $n^4/n^3 = n$. By Chebyshev: $\Pr[|x - n| \geq n] < 1/n$. Since $\Pr[x = k]$ is maximized at $k = n$:

$$\Pr[x = n] > \frac{1}{3n}.$$

Probability Analysis (I)

Pr[exactly n ones in M]

Expected number of ones: $n^4/n^3 = n$. By Chebyshev: $\Pr[|x - n| \geq n] < 1/n$. Since $\Pr[x = k]$ is maximized at $k = n$:

$$\Pr[x = n] > \frac{1}{3n}.$$

Pr[no row/column collision | $x = n$]

After k ones placed, probability the next avoids all occupied rows and columns is $(1 - k/n^2)^2$.

$$\prod_{k=1}^{n-1} \left(1 - \frac{k}{n^2}\right)^2 > 1 - \frac{n-1}{n} = \frac{1}{n}.$$

Probability Analysis (II)

Pr[M_c is a cycle matrix | permutation matrix]

$(n-1)!/n! = 1/n$.

Probability Analysis (II)

Pr[M_c is a cycle matrix | permutation matrix]

$(n-1)!/n! = 1/n$.

Overall success probability

$$\Pr[\text{success per trial}] > \frac{1}{3n} \cdot \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{3n^3}.$$

Probability Analysis (II)

Pr[M_c is a cycle matrix | permutation matrix]

$(n-1)!/n! = 1/n$.

Overall success probability

$$\Pr[\text{success per trial}] > \frac{1}{3n} \cdot \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{3n^3}.$$

Amplification

Repeat n^4 times:

$$\Pr[\text{at least one success}] > 1 - \left(1 - \frac{1}{3n^3}\right)^{n^4} \approx 1 - e^{-n/3} = 1 - \text{negl}(n).$$

Full HBM Proof System for Hamiltonicity

Protocol (repeated n^4 times)

For each iteration with its portion of the random string r:

1. Attempt cycle matrix extraction from r.
2. **If extraction fails**: reveal all bits in this portion. Verifier checks the extraction indeed fails.
3. **If extraction succeeds**: reveal non-used entries of M (confirming they are 0), then apply the Hamiltonian NIZK using the cycle matrix M_c.

Full HBM Proof System for Hamiltonicity

Protocol (repeated n^4 times)

For each iteration with its portion of the random string r:

1. Attempt cycle matrix extraction from r.
2. **If extraction fails**: reveal all bits in this portion. Verifier checks the extraction indeed fails.
3. **If extraction succeeds**: reveal non-used entries of M (confirming they are 0), then apply the Hamiltonian NIZK using the cycle matrix M_c.

Properties

- **Completeness**: prover succeeds whenever a cycle matrix is found.
- **Soundness**: if any iteration produces a cycle matrix, a cheating prover is caught (probability $1 - \mathrm{negl}(n)$).
- **Zero knowledge**: failed iterations reveal only random bits; successful iterations use the ZK protocol for cycle matrices.

NIZK for All of NP

Theorem 6

For any $L \in NP$, there is a NIZK proof in the HBM for L.

NIZK for All of NP

Theorem 6

For any $L \in NP$, there is a NIZK proof in the HBM for L.

Proof

Hamiltonicity is NP-complete: $\exists$ poly-time f with $x \in L \iff f(x)$ is Hamiltonian.
Prover computes $f(x)$ and runs the HBM NIZK for Hamiltonicity. $\square$

NIZK for All of NP

Theorem 6

For any $L \in NP$, there is a NIZK proof in the HBM for L.

Proof

Hamiltonicity is NP-complete: $\exists$ poly-time f with $x \in L \iff f(x)$ is Hamiltonian.
Prover computes $f(x)$ and runs the HBM NIZK for Hamiltonicity. $\square$

Theorem 7

Assuming trapdoor one-way permutations exist, for any $L \in NP$ there is a NIZK proof in the CRS model for L.

NIZK for All of NP

Theorem 6

For any $L \in NP$, there is a NIZK proof in the HBM for L.

Proof

Hamiltonicity is NP-complete: $\exists$ poly-time f with $x \in L \iff f(x)$ is Hamiltonian. Prover computes $f(x)$ and runs the HBM NIZK for Hamiltonicity. $\square$

Theorem 7

Assuming trapdoor one-way permutations exist, for any $L \in NP$ there is a NIZK proof in the CRS model for L.

Proof

Combine: (1) NIZK in HBM for all of NP (via Hamiltonicity), and (2) HBM $\Rightarrow$ CRS transformation using trapdoor OWPs. $\square$