Adaptively-Sound SNARGs for NP from Indistinguishability Obfuscation

David Wu

based on joint works with Brent Waters

Succinct Non-Interactive Arguments (SNARGs)

NP relation $\mathcal{R}$ (with related language $\mathcal{L}$)

common reference string (crs)

prover

$$\xrightarrow{(x, \pi)}$$

verifier

Completeness: *Honest prover convinces honest verifier of true statements*

$$\forall (x, w) \in \mathcal{R} : \Pr[\mathrm{Verify}(\mathrm{crs}, x, \pi) = 1 : \pi \leftarrow \mathrm{Prove}(\mathrm{crs}, x, w)] = 1$$

Succinctness: *Proof is much shorter than sending NP witness*

$$|\pi| = \mathrm{poly}(\lambda, \log|\mathcal{R}|)$$

Succinct Non-Interactive Arguments (SNARGs)

NP relation $\mathcal{R}$ (with related language $\mathcal{L}$)

prover

$$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$$

$$(x, \pi)$$

verifier

Soundness: *Efficient prover should not be able to convince verifier of a false statement*

Notion should be **adaptive**: prover can choose which statement it proves **after** it sees the CRS

Succinct Non-Interactive Arguments (SNARGs)

NP relation $\mathcal{R}$ (with related language $\mathcal{L}$)

prover

$x \in \{0,1\}^n$ $\longrightarrow$

$\text{crs} \leftarrow \text{Setup}(1^\lambda)$ $\longleftarrow$

π $\longrightarrow$

verifier

Soundness: *Efficient prover should not be able to convince verifier of a false statement*

Non-adaptive soundness: relaxation where prover has to declare the statement **before** seeing the CRS

Succinct Non-Interactive Arguments (SNARGs)

NP relation $\mathcal{R}$ (with related language $\mathcal{L}$)

prover

$x \in \{0,1\}^n$

$\text{crs} \leftarrow \text{Setup}(1^\lambda)$

π

verifier

Soundness: *Efficient prover should not be able to convince verifier of a false statement*

Non-adaptive soundness $\not\Rightarrow$ adaptive soundness (via complexity leveraging)

Complexity leveraging: $|\pi| = \text{poly}(\lambda, n)$ **Our goal:** $\text{poly}(\lambda, \log|\mathcal{R}|)$

SNARGs for NP

Constructions in idealized models

Random oracle model [Mic94, Val08, BCS16, BBHR19, CMS19, COS20, CY21, …]

Generic (or algebraic) group model [Gro16, GWC19, MBKM19, CHMMVW20, Lip24, DMS24, …]

Constructions from knowledge assumptions

[Gro10, BCCT12, GGPR13, BCIOP13, BCPR14, BISW17, ACLMT22, CLM23, …]

Non-adaptively-sound SNARG for NP from falsifiable assumptions

Sahai-Waters [SW14]: non-adaptively-sound SNARG for NP from indistinguishability obfuscation and one-way functions

Jain-Lin-Sahai [JLS21, JLS22]: indistinguishability obfuscation from falsifiable assumptions

Adaptively-sound SNARGs for NP from falsifiable assumptions?

The Gentry-Wichs Separation

"Adaptively-sound SNARGs for NP cannot be reduced to falsifiable assumptions in a black-box manner"

Does **not** rule out reductions that are able to decide the NP relation

Strategy: rely on sub-exponential hardness

- Adversary running in $2^{\lambda^{\varepsilon}}$ time succeeds with negligible advantage
- Suppose NP relation can be decided in time 2^{n^c} for some constant $c > 0$
- Instantiate the scheme with security parameter $\lambda > n^{c/\varepsilon}$

Reductions of $i\mathcal{O}$ to falsifiable assumptions run in time $2^{\Omega(|\text{input}|)}$

In Sahai-Waters: obfuscated programs take statement x and witness w as input, so reductions run in time $2^{\Omega(|x|+|w|)}$ and the Gentry-Wichs separation does not apply

The Gentry-Wichs Separation

"Adaptively-sound SNARGs for NP cannot be reduced to falsifiable assumptions in a black-box manner"

Does **not** rule out reduc

Strategy: rely on sub-ex

- Adversary running i
- Suppose NP relation can be decided in time 2^{n^c} for som … onstant $c > 0$
- Instantiate the scheme with security parameter $\lambda > n^{c/\varepsilon}$

Challenge: The size of the proof cannot grow polynomially with n

*Can we offload the **entire** cost of complexity leveraging (i.e., the use of sub-exponential hardness) to the CRS?*

Reductions of $i\mathcal{O}$ to falsifiable assumptions run in time $2^{\Omega(|\text{input}|)}$

In Sahai-Waters: obfuscated programs take statement x and witness w as input, so reductions run in time $2^{\Omega(|x|+|w|)}$ and the Gentry-Wichs separation does not apply

Recent Progress in Adaptive Soundness

[WW24a]: Adaptively-sound SNARGs for NP from sub-exponentially-secure $i\mathcal{O}$, sub-exponentially-secure one-way functions, and re-randomizable one-way functions (e.g., from discrete log / factoring)

[MPV24]: Sahai-Waters SNARG (from sub-exponentially-secure $i\mathcal{O}$, sub-exponentially-secure one-way functions) is adaptively sound in the designated-verifier model

[WZ24]: Adaptively-sound SNARGs for NP from sub-exponentially-secure $i\mathcal{O}$, sub-exponentially-secure one-way functions, and lossy functions (e.g., includes LWE)

[WW24b]: Adaptively-sound SNARGs for NP from sub-exponentially-secure $i\mathcal{O}$, and sub-exponentially-secure one-way functions

This Talk

[WW24a]: Adaptively-sound SNARGs for NP from sub-exponentially-secure $i\mathcal{O}$, sub-exponentially-secure one-way functions, and re-randomizable one-way functions (e.g., from discrete log / factoring)

[MPV24]: Sahai-Waters SNARG (from sub-exponentially-secure $i\mathcal{O}$, sub-exponentially-secure one-way functions) is adaptively sound in the designated-verifier model

[WZ24]: Adaptively-sound SNARGs for NP from sub-exponentially-secure $i\mathcal{O}$, sub-exponentially-secure one-way functions, and lossy functions (e.g., includes LWE)

[WW24b]: Adaptively-sound SNARGs for NP from sub-exponentially-secure $i\mathcal{O}$, and sub-exponentially-secure one-way functions

The Sahai-Waters SNARG

CRS contains **two** obfuscated programs

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 1$, output $\pi = \mathrm{PRF}(k, x)$
- Otherwise, output $\perp$

$\mathrm{Verify}(x, \pi)$:

- If $f(\pi) = f\big(\mathrm{PRF}(k, x)\big)$, output 1
- Otherwise, output 0

$\mathrm{PRF}(k, x)$ is a signature on the statement (technically, a MAC)

Check $f(\pi) = f\big(\mathrm{PRF}(k, x)\big)$ instead of $\pi = \mathrm{PRF}(k, x)$ to facilitate punctured programming proof

- $\mathcal{R}$ is an NP relation (fixed)
- PRF is a (puncturable) pseudorandom function
- f is a one-way function
- PRF key k hard-wired inside both programs

The Sahai-Waters SNARG

CRS contains **two** obfuscated programs

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 1$, output $\pi = \mathrm{PRF}(k, x)$
- Otherwise, output $\perp$

$\mathrm{Verify}(x, \pi)$:
- If $f(\pi) = f\big(\mathrm{PRF}(k, x)\big)$, output 1
- Otherwise, output 0

- $\mathcal{R}$ is an NP relation (fixed)
- PRF is a (puncturable) pseudorandom function
- f is a one-way function
- PRF key k hard-wired inside both programs

Will rely on indistinguishability obfuscation

if $C_0 \equiv C_1$, then $i\mathcal{O}(C_0) \approx i\mathcal{O}(C_1)$

Obfuscations of two functionally-equivalent programs are computationally indistinguishable

Non-Adaptive Soundness for Sahai-Waters

CRS contains **two** obfuscated programs

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 1$, output $\pi = \mathrm{PRF}(k, x)$
- Otherwise, output $\perp$

$\mathrm{Verify}(x, \pi)$:

- If $f(\pi) = f\big(\mathrm{PRF}(k, x)\big)$, output 1
- Otherwise, output 0

- $\mathcal{R}$ is an NP relation (fixed)
- PRF is a (puncturable) pseudorandom function
- f is a one-way function
- PRF key k hard-wired inside both programs

Assume PRF is puncturable

PRF key k

Punctured key $k^{(x^*)}$

Correctness: $\forall x \neq x^*\colon\ \mathrm{PRF}(k, x) = \mathrm{PRF}\big(k^{(x^*)}, x\big)$

Security: $\mathrm{PRF}(k, x^*)$ is pseudorandom given $k^{(x^*)}$

Non-Adaptive Soundness for Sahai-Waters

Non-adaptive soundness: adversary commits to statement x^* at the beginning

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 1$, output $\pi = \mathrm{PRF}(k, x)$
- Otherwise, output $\perp$

$\mathrm{Verify}(x, \pi)$:

- If $f(\pi) = f\big(\mathrm{PRF}(k, x)\big)$, output 1
- Otherwise, output 0

Real programs

Non-Adaptive Soundness for Sahai-Waters

Non-adaptive soundness: adversary commits to statement x^* at the beginning

$\text{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 1$, output $\pi = \text{PRF}(k, x)$
- Otherwise, output $\bot$

$\text{Verify}(x, \pi)$:
- If $f(\pi) = f\big(\text{PRF}(k, x)\big)$, output 1
- Otherwise, output 0

Real programs

$i\mathcal{O}$ security

Relies on the fact that $x^* \notin \mathcal{L}$

$\text{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 1$, output $\pi = \text{PRF}\big(k^{(x^*)}, x\big)$
- Otherwise, output $\bot$

$\text{Verify}(x, \pi)$:
- If $x = x^*$ and $f(\pi) = f(y^*)$, output 1
- If $x \neq x^*$ and $f(\pi) = f\big(\text{PRF}(k^{(x^*)}, x\big)$, output 1
- Otherwise, output 0

Replace k with punctured key $k^{(x^*)}$ and hard-code $y^* = \text{PRF}(k, x^*)$

Non-Adaptive Soundness for Sahai-Waters

Non-adaptive soundness: adversary commits to statement x^* at the beginning

Prove(x, w):

- If $\mathcal{R}(x, w) = 1$, output $\pi = \mathrm{PRF}\left(k^{(x^*)}, x\right)$
- Otherwise, output $\bot$

Verify(x, π):

- If $x = x^*$ and $f(\pi) = f(y^*)$, output 1
- If $x \neq x^*$ and $f(\pi) = f\left(\mathrm{PRF}(k^{(x^*)}, x\right)$, output 1
- Otherwise, output 0

Sample $y^* \leftarrow \{0,1\}^\lambda$

PRF security

Relies on the fact that $x^* \notin \mathcal{L}$

Prove(x, w):

- If $\mathcal{R}(x, w) = 1$, output $\pi = \mathrm{PRF}\left(k^{(x^*)}, x\right)$
- Otherwise, output $\bot$

Verify(x, π):

- If $x = x^*$ and $f(\pi) = f(y^*)$, output 1
- If $x \neq x^*$ and $f(\pi) = f\left(\mathrm{PRF}(k^{(x^*)}, x\right)$, output 1
- Otherwise, output 0

Replace k with punctured key $k^{(x^*)}$ and hard-code $y^* = \mathrm{PRF}(k, x^*)$

Non-Adaptive Soundness for Sahai-Waters

Non-adaptive soundness: adversary commits to statement x^* at the beginning

$\text{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 1$, output $\pi = \text{PRF}\left(k^{(x^*)}, x\right)$
- Otherwise, output $\perp$

$\text{Verify}(x, \pi)$:
- If $x = x^*$ and $f(\pi) = f(y^*)$, output 1
- If $x \neq x^*$ and $f(\pi) = f\left(\text{PRF}(k^{(x^*)}, x\right)$, output 1
- Otherwise, output 0

Sample $y^* \leftarrow \{0,1\}^\lambda$

To win, adversary must produce π such that $f(\pi) = f(y^*)$ where y^* is uniform!

Such an adversary breaks security of the one-way function!

Understanding Sahai-Waters

CRS contains **two** obfuscated programs

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 1$, output $\pi = \mathrm{PRF}(k, x)$
- Otherwise, output $\perp$

$\mathrm{Verify}(x, \pi)$:
- If $f(\pi) = f\big(\mathrm{PRF}(k, x)\big)$, output 1
- Otherwise, output 0

Key properties:
- Proof in Sahai-Waters is a **preimage** of a one-way function
- Non-adaptive adversary tells us **where** the adversary will invert (i.e., the point x^*)
- Reduction embeds a fresh OWF challenge at x^*, so successful adversary breaks OWF

Difficulty with adaptive security: Reduction does not know where the adversary will invert, so **where** do we embed the challenge to the one-way function?

Adaptive SNARG Blueprint

CRS contains **two** obfuscated programs

$\text{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 1$, output $\pi = \text{PRF}(k, x)$
- Otherwise, output $\perp$

$\text{Verify}(x, \pi)$:
- If $f(\pi) = f\big(\text{PRF}(k, x)\big)$, output 1
- Otherwise, output 0

Our approach: embed a one-way function challenge on **all** inputs, so no matter where adversary inverts, reduction is successful

Difficulty with adaptive security: Reduction does not know where the adversary will invert, so **where** do we embed the challenge to the one-way function?

Skipping to the End…

Sahai-Waters (non-adaptively sound)

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 1$, output $\pi = \mathrm{PRF}(k, x)$
- Otherwise, output $\perp$

$\mathrm{Verify}(x, \pi)$:

- If $f(\pi) = f(\mathrm{PRF}(k, x))$, output 1
- Otherwise, output 0

This talk (adaptively sound)

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = (b, \mathrm{PRF}(k_b, x))$

$\mathrm{Verify}(x, \pi)$:

- Parse $\pi = (b, y)$
- If $y = \mathrm{PRF}(k_b, x)$, output 1
- Otherwise, output 0

Difficulty with adaptive security: Reduction does not know where the adversary will invert, so **where** do we embed the challenge to the one-way function?

Skipping to the End…

Sahai-Waters (non-adaptively sound)

Prove(x, w):
- If $\mathcal{R}(x, w) = 1$, output $\pi = \mathrm{PRF}($
- Otherwise, output $\bot$

Verify(x, π):
- If $f(\pi) = f\big(\mathrm{PRF}(k, x)\big)$, output 1
- Otherwise, output 0

Not a big edit distance!

This talk (adaptively sound)

Prove(x, w):
- $\mathcal{R}(x, w) = 0$, output $\bot$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- tput $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

erify(x, π):
- Parse $\pi = (b, y)$
- $y = \mathrm{PRF}(k_b, x)$, output 1
- Otherwise, output 0

Difficulty with adaptive security: Reduction does not know where the adversary will invert, so **where** do we embed the challenge to the one-way function?

Adaptive SNARG Blueprint

CRS contains **two** obfuscated programs

$\text{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 1$, output $\pi = \text{PRF}(k, x)$
- Otherwise, output $\perp$

$\text{Verify}(x, \pi)$:

- If $f(\pi) = f\big(\text{PRF}(k, x)\big)$, output 1
- Otherwise, output 0

Our approach: embed a one-way function challenge on **all** inputs, so no matter where adversary inverts, reduction is successful

Attempt 1: Use a single challenge $y^* \leftarrow \{0,1\}^\lambda$

Difficulty with adaptive security: Reduction does not know where the adversary will invert, so **where** do we embed the challenge to the one-way function?

Adaptive SNARG Blueprint

CRS contains **two** obfuscated programs

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x$ **Ignore for now!** $\mathrm{RF}(k, x)$
- Otherwise, output $\perp$

$\mathrm{Verify}(x, \pi)$:

- If $f(\pi) = f(y^*)$, output 1
- Otherwise, output 0

Our approach: embed a one-way function challenge on **all** inputs, so no matter where adversary inverts, reduction is successful

Attempt 1: Use a single challenge $y^* \leftarrow \{0,1\}^\lambda$

Not indistinguishable from real verification program (where there are many distinct targets)

Difficulty with adaptive security: Reduction does not know where the adversary will invert, so **where** do we embed the challenge to the one-way function?

Adaptive SNARG Blueprint

CRS contains **two** obfuscated programs

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x$ **Ignore for now!** $\mathrm{RF}(k, x)$
- Otherwise, output $\perp$

$\mathrm{Verify}(x, \pi)$:
- If $f(\pi) = f(y^*)$, output 1
- Otherwise, output 0

Our approach: embed a one-way function challenge on **all** inputs, so no matter where adversary inverts, reduction is successful

Rerandomizable one-way function:

$$\mathrm{Rerand}(y^*; r) \rightarrow \tilde{y}$$

- Distribution of $\tilde{y}$ identical to fresh challenge
- Solution to $\tilde{y}$ implies solution for y

Difficulty with adaptive security: Reduction does not know where the adversary will invert, so **where** do we embed the challenge to the one-way function?

Adaptive SNARG Blueprint

CRS contains **two** obfuscated programs

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x$ **Ignore for now!** $\mathrm{RF}(k, x)$
- Otherwise, output $\bot$

$\mathrm{Verify}(x, \pi)$:
- If $f(\pi) = f(y^*)$, output 1
- Otherwise, output 0

Our approach: embed a one-way function challenge on **all** inputs, so no matter where adversary inverts, reduction is successful

Rerandomizable one-way function:

$$\mathrm{Rerand}(y^*; r) \to \tilde{y}$$

- Distribution of $\tilde{y}$ identical to fresh challenge
- Solution to $\tilde{y}$ implies solution for y^*

Construction from discrete log:
- Discrete log problem: given $y^* = g^x$, find x
- $\mathrm{Rerand}(y^*; r)$: Output $y^* \cdot g^r$
- Given z where $g^z = y^* \cdot g^r$ and r, recover $x = z - r$

Suffices to have **perfect** random self-reduction

Adaptive SNARG Blueprint

CRS contains **two** obfuscated programs

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x$ **Ignore for now!** $\mathrm{RF}(k, x)$
- Otherwise, output $\perp$

$\mathrm{Verify}(x, \pi)$:

- If $f(\pi) = f\left(\mathrm{Rerand}\left(y^*; \mathrm{PRF}(k, x)\right)\right)$, output 1
- Otherwise, output 0

Our approach: embed a one-way function challenge on **all** inputs, so no matter where adversary inverts, reduction is successful

Attempt 2: Use a different re-randomized challenge on every input

Proof on **any** statement yields a solution to f

Problem: how does the honest prover algorithm construct proofs?

Difficulty with adaptive security: Reduction does not know where the adversary will invert, so **where** do we embed the challenge to the one-way function?

The Two-Challenge Approach

CRS contains **two** obfuscated programs

$\text{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \text{PRF}(k_{\text{sel}}, x)$
- Output $\pi = \big(b, \text{PRF}(k_b, x)\big)$

$\text{Verify}(x, \pi)$:

- Parse $\pi = (b, y)$
- If $f(y) = f\big(\text{PRF}(k_b, x)\big)$, output 1
- Otherwise, output 0

Our approach: embed a one-way function challenge on **all** inputs, so no matter where adversary inverts, reduction is successful

Key idea: Every statement will be associated with **two** challenges and prover program will output solution to one of them

Selector $\text{PRF}(k_{\text{sel}}, \cdot)$ chooses bit $b \in \{0,1\}$

Both $\big(0, \text{PRF}(k_0, x)\big)$ and $\big(1, \text{PRF}(k_1, x)\big)$ are valid proofs, and prover program outputs **one** of them (determined by selector PRF)

Proving Adaptive Security

x_1
- $\mathrm{PRF}(k_0, x_1)$
- $\mathrm{PRF}(k_1, x_1)$

x_2
- $\mathrm{PRF}(k_0, x_2)$
- $\mathrm{PRF}(k_1, x_2)$

$\vdots$

x_N
- $\mathrm{PRF}(k_0, x_N)$
- $\mathrm{PRF}(k_1, x_N)$

Statements

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$, output $\bot$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Adversary wins if it outputs $x, \pi = (b, y)$ where $f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$

Proving Adaptive Security

x_1 → $f(\mathrm{PRF}(k_0, x_1))$
x_1 → $f(\mathrm{PRF}(k_1, x_1))$

x_2 → $f(\mathrm{PRF}(k_0, x_2))$
x_2 → $f(\mathrm{PRF}(k_1, x_2))$

⋮

x_N → $f(\mathrm{PRF}(k_0, x_N))$
x_N → $f(\mathrm{PRF}(k_1, x_N))$

Verification targets

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Adversary wins if it outputs $x, \pi = (b, y)$ where $f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$

Take any false statement $x \notin \mathcal{L}$

By PRF security, the value of $\mathrm{PRF}(k_{\mathrm{sel}}, x)$ is pseudorandom

If adversary produces a proof $\pi = (b, y)$ on x, then
$$\Pr[b = \mathrm{PRF}(k_{\mathrm{sel}}, x)] \approx 1/2$$
Otherwise, adversary distinguishes $\mathrm{PRF}(k_{\mathrm{sel}}, x)$

Proving Adaptive Security

x_1: $f(\mathrm{PRF}(k_0, x_1))$, $f(\mathrm{PRF}(k_1, x_1))$

x_2: $f(\mathrm{PRF}(k_0, x_2))$, $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

x_N: $f(\mathrm{PRF}(k_0, x_N))$, $f(\mathrm{PRF}(k_1, x_N))$

Verification targets

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Adversary wins if it outputs $x, \pi = (b, y)$ where $f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$

Ta

B

If

Ot

Consider adaptive soundness game where adversary wins only when the adversary outputs a statement x and a proof where $\pi = (b, y)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Only decreases adversary's advantage by factor of 2

Step 1: Only Accept an Off-Path Proof

x_1: $f(\mathrm{PRF}(k_0, x_1))$, $f(\mathrm{PRF}(k_1, x_1))$

x_2: $f(\mathrm{PRF}(k_0, x_2))$, $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

x_N: $f(\mathrm{PRF}(k_0, x_N))$, $f(\mathrm{PRF}(k_1, x_N))$

Verification targets

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Adversary only wins if it outputs x, b, y where
$f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Step 1: Only Accept an Off-Path Proof

Formally:

Game_0: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\text{Verify}(x, \pi) = 1$

Game_1: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\text{Verify}(x, \pi) = 1$ and $b \neq F(k_{\text{sel}}, x)$

Claim: $\Pr[\text{Game}_1 = 1] \geq \frac{1}{2} \cdot \Pr[\text{Game}_0 = 1] - \text{negl}(\lambda)$

Define event E_i to be the event that prover chooses statement $i \in \{0,1\}^n$

$$\Pr[\text{Game}_1 = 1] = \sum_{i \in \{0,1\}^n} \Pr[\text{Game}_1 = 1 \wedge \text{E}_i] \qquad \Pr[\text{Game}_0 = 1] = \sum_{i \in \{0,1\}^n} \Pr[\text{Game}_0 = 1 \wedge \text{E}_i]$$

$\text{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \text{PRF}(k_{\text{sel}}, x)$
- Output $\pi = (b, \text{PRF}(k_b, x))$

$\text{Verify}(x, \pi)$:

- Parse $\pi = (b, y)$
- If $f(y) = f(\text{PRF}(k_b, x))$, output 1
- Otherwise, output 0

Step 1: Only Accept an Off-Path Proof

Formally:

Game_0: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\text{Verify}(x, \pi) = 1$

Game_1: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\text{Verify}(x, \pi) = 1$ and $b \neq F(k_{\text{sel}}, x)$

Claim: $\Pr[\text{Game}_1 = 1] \geq \frac{1}{2} \cdot \Pr[\text{Game}_0 = 1] - \text{negl}(\lambda)$

Define event E_i to be the event that prover chooses statement $i \in \{0,1\}^n$

$$\Pr[\text{Game}_1 = 1] = \sum_{i \in \{0,1\}^n} \Pr[\text{Game}_1 = 1 \wedge \text{E}_i] \qquad \Pr[\text{Game}_0 = 1] = \sum_{i \in \{0,1\}^n} \Pr[\text{Game}_0 = 1 \wedge \text{E}_i]$$

Suffices to show that for all $i \in \{0,1\}^n$:

$$\Pr[\text{Game}_1 = 1 \wedge \text{E}_i] \geq \frac{1}{2} \cdot \Pr[\text{Game}_0 = 1 \wedge \text{E}_i] - \frac{1}{2^n} \cdot \text{negl}(\lambda)$$

Will require sub-exponential hardness!

Step 1: Only Accept an Off-Path Proof

Formally:

Game_0: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\text{Verify}(x, \pi) = 1$

Game_1: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\text{Verify}(x, \pi) = 1$ and $b \neq F(k_{\text{sel}}, x)$

Claim: $\Pr[\text{Game}_1 = 1] \geq \frac{1}{2} \cdot \Pr[\text{Game}_0 = 1] - \text{negl}(\lambda)$

Define event E_i to be the event that prover chooses statement $i \in \{0,1\}^n$

$$\Pr[\text{Game}_1 = 1] = \sum_{i \in \{0,1\}^n} \Pr[\text{Game}_1 = 1 \wedge \text{E}_i] \qquad \Pr[\text{Game}_0 = 1] = \sum_{i \in \{0,1\}^n} \Pr[\text{Game}_0 = 1 \wedge \text{E}_i]$$

Suffices to show that for all $i \in \{0,1\}^n$:

$$\Pr[\text{Game}_1 = 1 \wedge \text{E}_i] \geq \frac{1}{2} \cdot \Pr[\text{Game}_0 = 1 \wedge \text{E}_i] - \frac{1}{2^n} \cdot \text{negl}(\lambda)$$

Observe: If $i \in \mathcal{L}$, then $\Pr[\text{Game}_1 = 1 \wedge \text{E}_i] = 0 = \Pr[\text{Game}_0 = 1 \wedge \text{E}_i]$

Step 1: Only Accept an Off-Path Proof

Formally:

Game_0: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$

Game_1: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$ and $b \neq F(k_{\mathrm{sel}}, x)$

Claim: for all $i \notin \mathcal{L}$: $\Pr[\mathrm{Game}_1 = 1 \wedge \mathrm{E}_i] \geq \frac{1}{2} \cdot \Pr[\mathrm{Game}_0 = 1 \wedge \mathrm{E}_i] - \frac{1}{2^n} \cdot \mathrm{negl}(\lambda)$

$\mathrm{Hyb}_{i,0}$ for $i \notin \mathcal{L}$

$$\Pr\left[\mathrm{Hyb}_{i,0} = 1\right] = \Pr[\mathrm{Game}_0 = 1 \wedge \mathrm{E}_i]$$

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \left(b, \mathrm{PRF}(k_b, x)\right)$

Prover wins if it outputs x, b, y where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$ and $x = i$

Step 1: Only Accept an Off-Path Proof

Formally:

Game_0: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$

Game_1: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$ and $b \neq F(k_{\mathrm{sel}}, x)$

Claim: for all $i \notin \mathcal{L}$: $\Pr[\mathrm{Game}_1 = 1 \wedge \mathrm{E}_i] \geq \frac{1}{2} \cdot \Pr[\mathrm{Game}_0 = 1 \wedge \mathrm{E}_i] - \frac{1}{2^n} \cdot \mathrm{negl}(\lambda)$

$\mathrm{Hyb}_{i,0}$ for $i \notin \mathcal{L}$

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

$i\mathcal{O}$

$\mathrm{Hyb}_{i,1}$ for $i \notin \mathcal{L}$

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$ or $x = i$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}\left(k_{\mathrm{sel}}^{(i)}, x\right)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Prover wins if it outputs x, b, y where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$ and $x = i$

Step 1: Only Accept an Off-Path Proof

- $\Pr\left[\mathrm{Hyb}_{i,0} = 1\right] = \Pr[\mathrm{Game}_0 = 1 \wedge \mathrm{E}_i]$
- $\Pr\left[\mathrm{Hyb}_{i,1} = 1\right] \geq \Pr\left[\mathrm{Hyb}_{i,0} = 1\right] - 2^{-n} \cdot \mathrm{negl}(\lambda)$ (sub-exponential security of $i\mathcal{O}$)

$\mathrm{Hyb}_{i,0}$ for $i \notin \mathcal{L}$

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \left(b, \mathrm{PRF}(k_b, x)\right)$

$i\mathcal{O}$ $\Rightarrow$

$\mathrm{Hyb}_{i,1}$ for $i \notin \mathcal{L}$

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$ or $x = i$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}\left(k_{\mathrm{sel}}^{(i)}, x\right)$
- Output $\pi = \left(b, \mathrm{PRF}(k_b, x)\right)$

Prover wins if it outputs x, b, y where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$ and $x = i$

Step 1: Only Accept an Off-Path Proof

- $\Pr[\mathrm{Hyb}_{i,0} = 1] = \Pr[\mathrm{Game}_0 = 1 \wedge \mathrm{E}_i]$
- $\Pr[\mathrm{Hyb}_{i,1} = 1] \geq \Pr[\mathrm{Hyb}_{i,0} = 1] - 2^{-n} \cdot \mathrm{negl}(\lambda)$ (sub-exponential security of $i\mathcal{O}$)
- $\Pr[\mathrm{Hyb}_{i,2} = 1] = \frac{1}{2} \cdot \Pr[\mathrm{Hyb}_{i,1} = 1]$

$\mathrm{Hyb}_{i,2}$ for $i \notin \mathcal{L}$

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$ or $x = i$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}\left(k_{\mathrm{sel}}^{(i)}, x\right)$
- Output $\pi = \left(b, \mathrm{PRF}(k_b, x)\right)$

$\mathrm{Hyb}_{i,1}$ for $i \notin \mathcal{L}$

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$ or $x = i$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}\left(k_{\mathrm{sel}}^{(i)}, x\right)$
- Output $\pi = \left(b, \mathrm{PRF}(k_b, x)\right)$

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

Prover wins if it outputs x, b, y where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$ and $x = i$ and $b \neq b'$

Step 1: Only Accept an Off-Path Proof

- $\Pr\left[\mathrm{Hyb}_{i,0} = 1\right] = \Pr[\mathrm{Game}_0 = 1 \wedge \mathrm{E}_i]$
- $\Pr\left[\mathrm{Hyb}_{i,1} = 1\right] \geq \Pr\left[\mathrm{Hyb}_{i,0} = 1\right] - 2^{-n} \cdot \mathrm{negl}(\lambda)$ (sub-exponential security of $i\mathcal{O}$)
- $\Pr\left[\mathrm{Hyb}_{i,2} = 1\right] = \frac{1}{2} \cdot \Pr\left[\mathrm{Hyb}_{i,1} = 1\right]$
- $\Pr\left[\mathrm{Hyb}_{i,3} = 1\right] \geq \Pr\left[\mathrm{Hyb}_{i,2} = 1\right] - 2^{-n} \cdot \mathrm{negl}(\lambda)$ (sub-exponential security of PRF)
- $\Pr\left[\mathrm{Hyb}_{i,3} = 1\right] = \Pr[\mathrm{Game}_1 = 1 \wedge \mathrm{E}_i]$

$\mathrm{Hyb}_{i,2}$ for $i \notin \mathcal{L}$

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$ or $x = i$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}\left(k_{\mathrm{sel}}^{(i)}, x\right)$
- Output $\pi = \left(b, \mathrm{PRF}(k_b, x)\right)$

$\xrightarrow{\mathrm{PRF}}$

$\mathrm{Hyb}_{i,3}$ for $i \notin \mathcal{L}$

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$ or $x = i$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}\left(k_{\mathrm{sel}}^{(i)}, x\right)$
- Output $\pi = \left(b, \mathrm{PRF}(k_b, x)\right)$

Prover wins if it outputs x, b, y where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$ and $x = i$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, i)$

Step 1: Only Accept an Off-Path Proof

- $\Pr[\mathrm{Hyb}_{i,0} = 1] = \Pr[\mathrm{Game}_0 = 1 \wedge \mathrm{E}_i]$
- $\Pr[\mathrm{Hyb}_{i,1} = 1] \geq \Pr[\mathrm{Hyb}_{i,0} = 1] - 2^{-n} \cdot \mathrm{negl}(\lambda)$ (sub-exponential security of $i\mathcal{O}$)
- $\Pr[\mathrm{Hyb}_{i,2} = 1] = \frac{1}{2} \cdot \Pr[\mathrm{Hyb}_{i,1} = 1]$
- $\Pr[\mathrm{Hyb}_{i,3} = 1] \geq \Pr[\mathrm{Hyb}_{i,2} = 1] - 2^{-n} \cdot \mathrm{negl}(\lambda)$ (sub-exponential security of PRF)
- $\Pr[\mathrm{Hyb}_{i,3} = 1] = \Pr[\mathrm{Game}_1 = 1 \wedge \mathrm{E}_i]$

Formally:

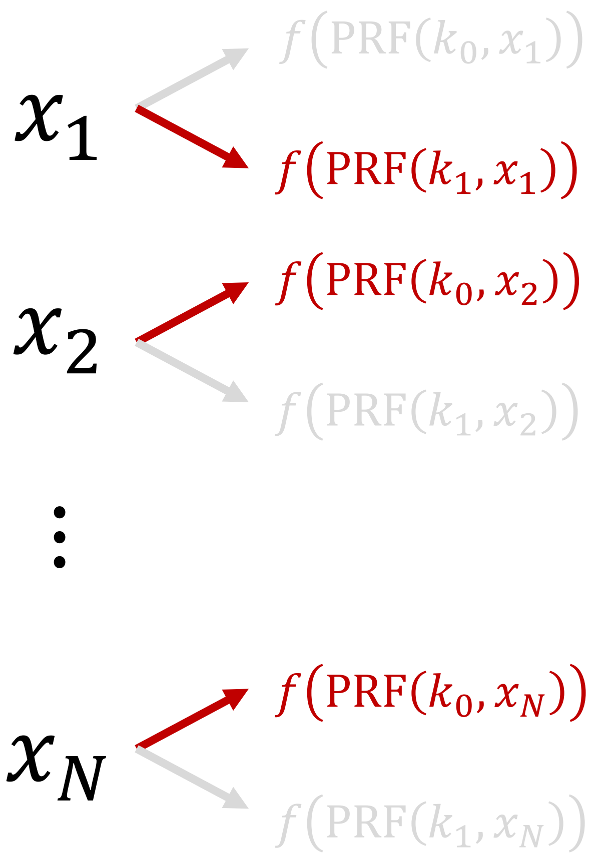
Game_0: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$

Game_1: Prover wins if it outputs $x, \pi = (b, y)$ where $x \notin \mathcal{L}$ and $\mathrm{Verify}(x, \pi) = 1$ and $b \neq F(k_{\mathrm{sel}}, x)$

Claim: for all $i \notin \mathcal{L}$: $\Pr[\mathrm{Game}_1 = 1 \wedge \mathrm{E}_i] \geq \frac{1}{2} \cdot \Pr[\mathrm{Game}_0 = 1 \wedge \mathrm{E}_i] - \frac{1}{2^n} \cdot \mathrm{negl}(\lambda)$

Therefore: $\Pr[\mathrm{Game}_1 = 1] \geq \frac{1}{2} \cdot \Pr[\mathrm{Game}_0 = 1] - \mathrm{negl}(\lambda)$

Step 1: Only Accept an Off-Path Proof



Verification targets

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Adversary only wins if it outputs x, b, y where
$f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Observation: Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

Value is pseudorandom!

Step 2: Change the Off-Path Targets

x_1: $f(\mathrm{PRF}(k_0, x_1))$, $f(\mathrm{PRF}(k_1, x_1))$

x_2: $f(\mathrm{PRF}(k_0, x_2))$, $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

x_N: $f(\mathrm{PRF}(k_0, x_N))$, $f(\mathrm{PRF}(k_1, x_N))$

Verification targets

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Adversary only wins if it outputs x, b, y where
$f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Observation: Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

Switch "off-path" verification targets to be

$$f\big(\mathrm{PRF}(k_b, x)\big) \approx \mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_b, x)\big)$$

Step 2: Change the Off-Path Targets

Formally argued using $N = 2^n$ hybrids

x_1: $f(\mathrm{PRF}(k_0, x_1))$, $\mathrm{Rerand}(y^*; \mathrm{PRF}(k_1, x_1))$

x_2: $f(\mathrm{PRF}(k_0, x_2))$, $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

x_N: $f(\mathrm{PRF}(k_0, x_N))$, $f(\mathrm{PRF}(k_1, x_N))$

Verification targets

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Adversary only wins if it outputs x, b, y where
$f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Observation: Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

Switch "off-path" verification targets to be

$$f\big(\mathrm{PRF}(k_b, x)\big) \approx \mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_b, x)\big)$$

Step 2: Change the Off-Path Targets

Formally argued using $N = 2^n$ hybrids

x_1 → $f(\mathrm{PRF}(k_0, x_1))$

x_1 → $\mathrm{Rerand}(y^*; \mathrm{PRF}(k_1, x_1))$

x_2 → $\mathrm{Rerand}(y^*; \mathrm{PRF}(k_0, x_2))$

x_2 → $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

x_N → $f(\mathrm{PRF}(k_0, x_N))$

x_N → $f(\mathrm{PRF}(k_1, x_N))$

Verification targets

Every statement has **two** possible proofs:

one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Adversary only wins if it outputs x, b, y where

$f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Observation: Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

Switch "off-path" verification targets to be

$$f\big(\mathrm{PRF}(k_b, x)\big) \approx \mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_b, x)\big)$$

Step 2: Change the Off-Path Targets

Formally argued using $N = 2^n$ hybrids

x_1: $f\big(\mathrm{PRF}(k_0, x_1)\big)$ / $\mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_1, x_1)\big)$

x_2: $\mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_0, x_2)\big)$ / $f\big(\mathrm{PRF}(k_1, x_2)\big)$

$\vdots$

x_N: $\mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_0, x_N)\big)$ / $f\big(\mathrm{PRF}(k_1, x_N)\big)$

Verification targets

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Adversary only wins if it outputs x, b, y where
$f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Observation: Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

Switch "off-path" verification targets to be

$$f\big(\mathrm{PRF}(k_b, x)\big) \approx \mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_b, x)\big)$$

Step 2: Change the Off-Path Targets

x_1: $f(\mathrm{PRF}(k_0, x_1))$; $\mathrm{Rerand}(y^*; \mathrm{PRF}(k_1, x_1))$

x_2: $\mathrm{Rerand}(y^*; \mathrm{PRF}(k_0, x_2))$; $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

x_N: $\mathrm{Rerand}(y^*; \mathrm{PRF}(k_0, x_N))$; $f(\mathrm{PRF}(k_1, x_N))$

Verification targets

Every statement has **two** possible proofs:

one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

$\mathrm{Verify}(x, \pi)$:

- Parse $\pi = (b, y)$
- Output 1 if
  - $b = \mathrm{PRF}(k_{\mathrm{sel}}, x)$ and $f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$
  - $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$ and $f(y) = f\Big(\mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_b, x)\big)\Big)$
- Otherwise, output 0

y^* is a random instance for the OWF

Step 2: Change the Off-Path Targets

x_1: $f(\mathrm{PRF}(k_0, x_1))$, $\mathrm{Rerand}(y^*; \mathrm{PRF}(k_1, x_1))$

x_2: $\mathrm{Rerand}(y^*; \mathrm{PRF}(k_0, x_2))$, $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

x_N: $\mathrm{Rerand}(y^*; \mathrm{PRF}(k_0, x_N))$, $f(\mathrm{PRF}(k_1, x_N))$

Verification targets

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = (b, \mathrm{PRF}(k_b, x))$

Adversary only wins if it outputs x, b, y where
$f(y) = f(\mathrm{PRF}(k_b, x))$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

$\Downarrow$

Adversary only wins if it outputs x, b, y where
$f(y) = f\left(\mathrm{Rerand}(y^*; \mathrm{PRF}(k_b, x))\right)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Step 2: Change the Off-Path Targets

x_1

$f\left(\mathrm{PRF}(k_0, x_1)\right)$

$\mathrm{Rerand}\left(y^*; \mathrm{PRF}(k_1, x_1)\right)$

x_2

$\mathrm{Rerand}\left(y^*; \mathrm{PRF}(k_0, x_2)\right)$

$f\left(\mathrm{PRF}(k_1, x_2)\right)$

$\vdots$

x_N

$\mathrm{Rerand}\left(y^*; \mathrm{PRF}(k_0, x_N)\right)$

$f\left(\mathrm{PRF}(k_1, x_N)\right)$

Verification targets

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \left(b, \mathrm{PRF}(k_b, x)\right)$

Adversary only wins if it outputs x, b, y where
$f(y) = f\left(\mathrm{PRF}(k_b, x)\right)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

By the rerandomization property, any such y yields a preimage of the challenge y^*

where
$f(y) = f\left(\mathrm{Rerand}\left(y^*; \mathrm{PRF}(k_b, x)\right)\right)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Step 2: Change the Off-Path Targets

x_1

$f(\mathrm{PRF}(k_0, x_1))$

$\mathrm{Rerand}(y^*; \mathrm{PRF}(k_1, x_1))$

x_2

$\mathrm{Rerand}(y^*; \mathrm{PRF}(k_0, x_2))$

$f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

x_N

$\mathrm{Rerand}(y^*; \mathrm{PRF}(k_0, x_N))$

$f(\mathrm{PRF}(k_1, x_N))$

Verification targets

Every statement has **two** possible proofs:
one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Final proof is a bit and a single preimage of the OWF:
$\mathrm{poly}(\lambda)$ bits, independent of n

CRS size is $\mathrm{poly}(\lambda, n)$ – necessary to absorb the exponential security loss incurred by the $N = 2^n$ hybrids

Avoiding Rerandomization

Previous approach needed the OWF to be statistically rerandomizable

Rerandomizability seems to be an *algebraic* property (not known how to build from $i\mathcal{O}$ and OWFs)

Waters-Zhandry [WZ24]: Can relax rerandomizable PRF to a lossy function

Lossy functions also not known from $i\mathcal{O}$ and OWFs

Can we get adaptive soundness just from $i\mathcal{O}$ and OWFs?

Avoiding Rerandomization

$x_1 \to f(\mathrm{PRF}(k_0, x_1))$ / $f(\mathrm{PRF}(k_1, x_1))$

$x_2 \to f(\mathrm{PRF}(k_0, x_2))$ / $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

$x_N \to f(\mathrm{PRF}(k_0, x_N))$ / $f(\mathrm{PRF}(k_1, x_N))$

Verifier program checks if

$$f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$$

What if we just planted the ***same*** challenge everywhere?

$x_1 \to f(\mathrm{PRF}(k_0, x_1))$ / $\mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_1, x_1)\big)$

$x_2 \to \mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_0, x_2)\big)$ / $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

$x_N \to \mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_0, x_N)\big)$ / $f(\mathrm{PRF}(k_1, x_N))$

Verifier program checks if

$$f(y) = f\left(\mathrm{Rerand}\big(y^*; \mathrm{PRF}(k_b, x)\big)\right)$$

Avoiding Rerandomization

x_1 → $f\big(\mathrm{PRF}(k_0, x_1)\big)$, $f\big(\mathrm{PRF}(k_1, x_1)\big)$

x_2 → $f\big(\mathrm{PRF}(k_0, x_2)\big)$, $f\big(\mathrm{PRF}(k_1, x_2)\big)$

$\vdots$

x_N → $f\big(\mathrm{PRF}(k_0, x_N)\big)$, $f\big(\mathrm{PRF}(k_1, x_N)\big)$

Verifier program checks if
$f(y) = f\big(\mathrm{PRF}(k_b, x)\big)$

What if we just planted the ***same*** challenge everywhere?

Can no longer argue indistinguishability

x_1 → $f\big(\mathrm{PRF}(k_0, x_1)\big)$, $f(y^*)$

x_2 → $f(y^*)$, $f\big(\mathrm{PRF}(k_1, x_2)\big)$

$\vdots$

x_N → $f(y^*)$, $f\big(\mathrm{PRF}(k_1, x_N)\big)$

Verifier program checks if
$f(y) = f(y^*)$

Avoiding Rerandomization

x_1 → $f(\mathrm{PRF}(k_0, x_1))$
x_1 → $f(\mathrm{PRF}(k_1, x_1))$

x_2 → $f(\mathrm{PRF}(k_0, x_2))$
x_2 → $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

x_N → $f(\mathrm{PRF}(k_0, x_N))$
x_N → $f(\mathrm{PRF}(k_1, x_N))$

Verifier program checks if
$f(y) = f(\mathrm{PRF}(k_b, x))$

What if we just planted the ***same*** challenge everywhere?

Can no longer argue indistinguishability

x_1 → $f(\mathrm{PRF}(k_0, x_1))$
x_1 → $f(y^*)$

x_2 → $f(y^*)$
x_2 → $f(\mathrm{PRF}(k_1, x_2))$

$\vdots$

Need a different way to embed $f(y^*)$

Before: PRF outputs the OWF challenge so we need a different challenge for each statement

New approach: use PRF to **blind** a **single** OWF challenge

Avoiding Rerandomization

x_1 $y = \mathrm{PRF}(k_0, x_1)$ $y = \mathrm{PRF}(k_1, x_1)$

x_2 $y = \mathrm{PRF}(k_0, x_2)$ $y = \mathrm{PRF}(k_1, x_2)$

$\vdots$

x_N $y = \mathrm{PRF}(k_0, x_N)$ $y = \mathrm{PRF}(k_1, x_N)$

y is a valid proof for x_i if it corresponds to one of the two paths

consider adversary successful only if it provides "off-path" target

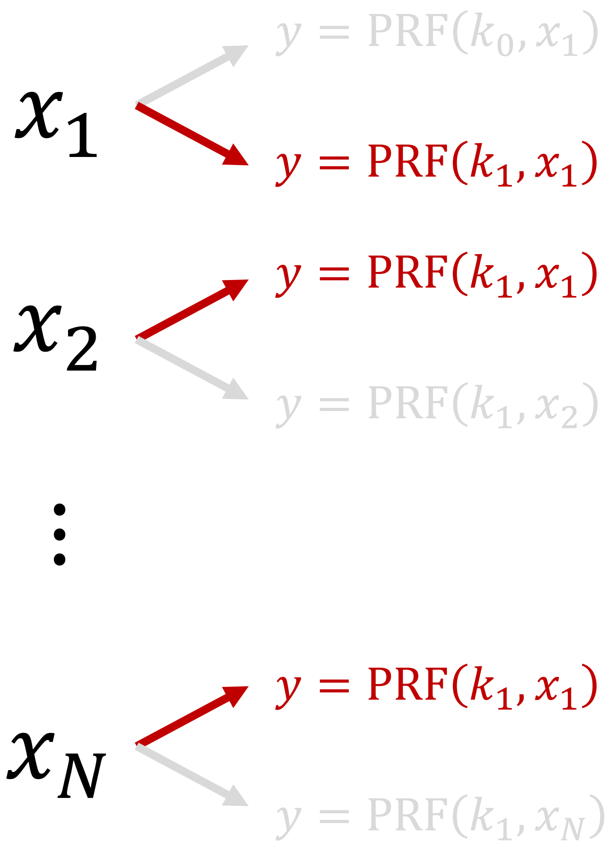
x_1 $y = \mathrm{PRF}(k_0, x_1)$ $y = \mathrm{PRF}(k_1, x_1)$

x_2 $y = \mathrm{PRF}(k_0, x_2)$ $y = \mathrm{PRF}(k_1, x_2)$

$\vdots$

x_N $y = \mathrm{PRF}(k_0, x_N)$ $y = \mathrm{PRF}(k_1, x_N)$

Avoiding Rerandomization



"off-path" verification targets

First, rewrite $y = \mathrm{PRF}(k_b, x_i)$ as

$$y \oplus \mathrm{PRF}(k_b, x_i) \oplus y^* = y^*$$

Avoiding Rerandomization

x_1 $y = \mathrm{PRF}(k_0, x_1)$

$y \oplus \mathrm{PRF}(k_1, x_1) \oplus y^* = y^*$

x_2 $y \oplus \mathrm{PRF}(k_0, x_2) \oplus y^* = y^*$

$y = \mathrm{PRF}(k_1, x_2)$

$\vdots$

x_N $y \oplus \mathrm{PRF}(k_0, x_N) \oplus y^* = y^*$

$y = \mathrm{PRF}(k_1, x_N)$

"off-path" verification targets

First, rewrite $y = \mathrm{PRF}(k_b, x_i)$ as

$$y \oplus \mathrm{PRF}(k_b, x_i) \oplus y^* = y^*$$

Adversary only wins if it outputs x, b, y where $y \oplus \mathrm{PRF}(k_b, x) \oplus y^* = y^*$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

By punctured PRF security:

$$\mathrm{PRF}(k_b, x) \oplus y^* \approx \mathrm{PRF}(k_b, x)$$

Avoiding Rerandomization

x_1

$y = \mathrm{PRF}(k_0, x_1)$

$y \oplus \mathrm{PRF}(k_1, x_1) = y^*$

x_2

$y \oplus \mathrm{PRF}(k_0, x_2) \oplus y^* = y^*$

$y = \mathrm{PRF}(k_1, x_2)$

$\vdots$

x_N

$y \oplus \mathrm{PRF}(k_0, x_N) \oplus y^* = y^*$

$y = \mathrm{PRF}(k_1, x_N)$

"off-path" verification targets

First, rewrite $y = \mathrm{PRF}(k_b, x_i)$ as

$$y \oplus \mathrm{PRF}(k_b, x_i) \oplus y^* = y^*$$

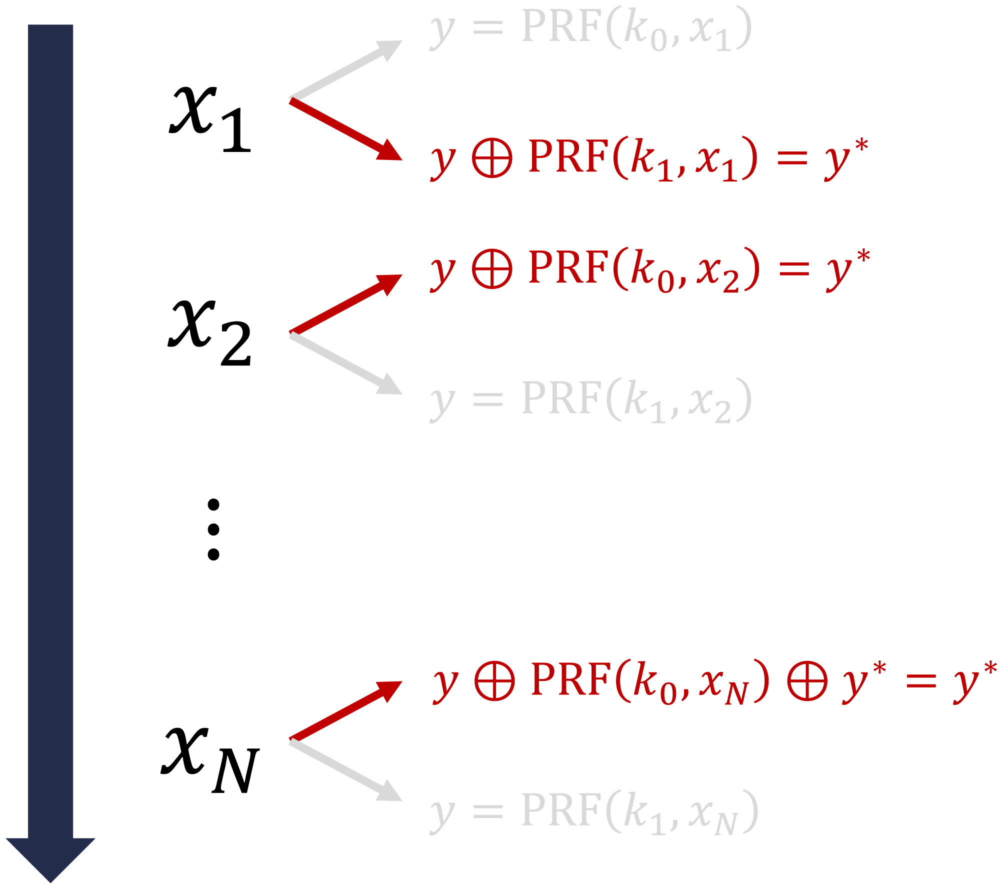
Adversary only wins if it outputs x, b, y where $y \oplus \mathrm{PRF}(k_b, x) \oplus y^* = y^*$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

By punctured PRF security:

$$\mathrm{PRF}(k_b, x) \oplus y^* \approx \mathrm{PRF}(k_b, x)$$

Avoiding Rerandomization



“off-path” verification targets

First, rewrite $y = \mathrm{PRF}(k_b, x_i)$ as

$$y \oplus \mathrm{PRF}(k_b, x_i) \oplus y^* = y^*$$

Adversary only wins if it outputs x, b, y where $y \oplus \mathrm{PRF}(k_b, x) \oplus y^* = y^*$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

By punctured PRF security:

$$\mathrm{PRF}(k_b, x) \oplus y^* \approx \mathrm{PRF}(k_b, x)$$

Avoiding Rerandomization

x_1: $y = \mathrm{PRF}(k_0, x_1)$; $y \oplus \mathrm{PRF}(k_1, x_1) = y^*$

x_2: $y \oplus \mathrm{PRF}(k_0, x_2) = y^*$; $y = \mathrm{PRF}(k_1, x_2)$

$\vdots$

x_N: $y \oplus \mathrm{PRF}(k_0, x_N) = y^*$; $y = \mathrm{PRF}(k_1, x_N)$

"off-path" verification targets

First, rewrite $y = \mathrm{PRF}(k_b, x_i)$ as

$$y \oplus \mathrm{PRF}(k_b, x_i) \oplus y^* = y^*$$

Adversary only wins if it outputs x, b, y where
$y \oplus \mathrm{PRF}(k_b, x) \oplus y^* = y^*$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

By punctured PRF security:

$$\mathrm{PRF}(k_b, x) \oplus y^* \approx \mathrm{PRF}(k_b, x)$$

Avoiding Rerandomization

x_1
$y = \mathrm{PRF}(k_0, x_1)$
$y \oplus \mathrm{PRF}(k_1, x_1) = y^*$

x_2
$y \oplus \mathrm{PRF}(k_0, x_2) = y^*$
$y = \mathrm{PRF}(k_1, x_2)$

$\vdots$

x_N
$y \oplus \mathrm{PRF}(k_0, x_N) = y^*$
$y = \mathrm{PRF}(k_1, x_N)$

"off-path" verification targets

First, rewrite $y = \mathrm{PRF}(k_b, x_i)$ as

$$y \oplus \mathrm{PRF}(k_b, x_i) \oplus y^* = y^*$$

Adversary only wins if it outputs x, b, y where
$y \oplus \mathrm{PRF}(k_b, x) \oplus y^* = y^*$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

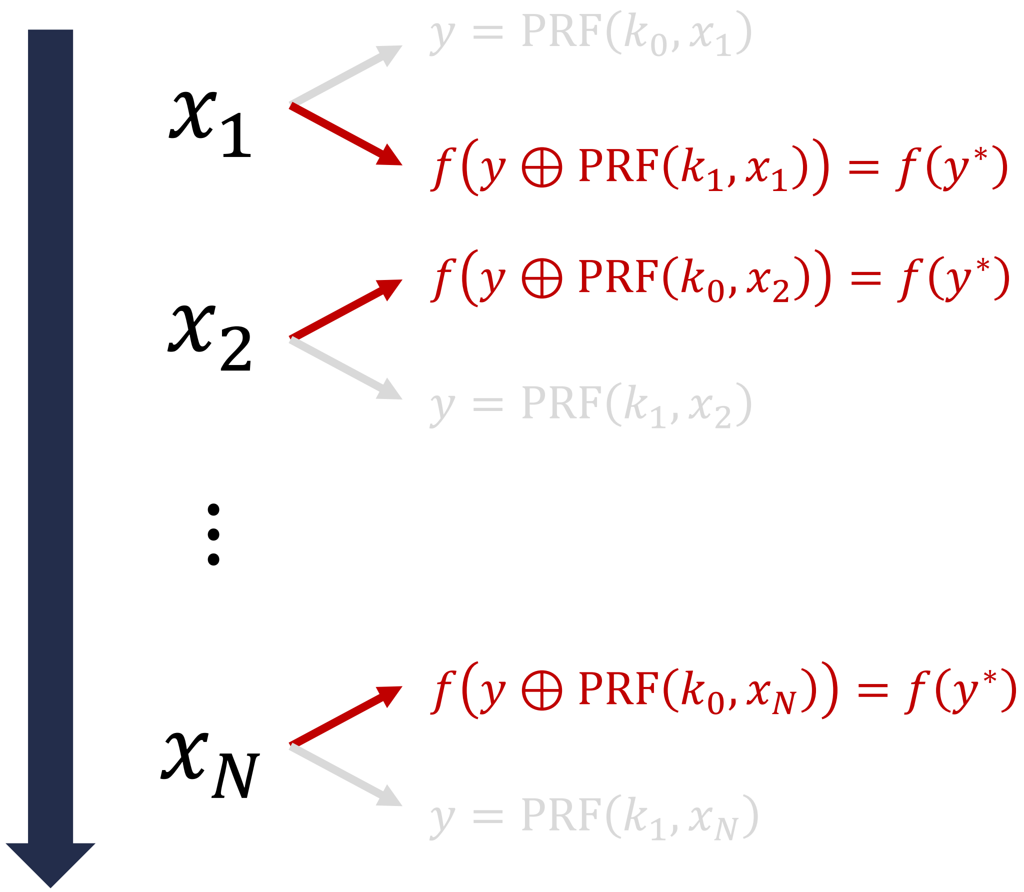
Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

By punctured PRF security:

$$\mathrm{PRF}(k_b, x) \oplus y^* \approx \mathrm{PRF}(k_b, x)$$

Let f be an injective OWF
Then $z = z' \Leftrightarrow f(z) = f(z')$

Avoiding Rerandomization



"off-path" verification targets

First, rewrite $y = \mathrm{PRF}(k_b, x_i)$ as

$$y \oplus \mathrm{PRF}(k_b, x_i) \oplus y^* = y^*$$

Adversary only wins if it outputs x, b, y where
$y \oplus \mathrm{PRF}(k_b, x) \oplus y^* = y^*$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Prover program ***never*** computes $\mathrm{PRF}(k_b, x)$

By punctured PRF security:

$$\mathrm{PRF}(k_b, x) \oplus y^* \approx \mathrm{PRF}(k_b, x)$$

Let f be an injective OWF
Then $z = z' \Leftrightarrow f(z) = f(z')$

Avoiding Rerandomization

x_1

$y = \mathrm{PRF}(k_0, x_1)$

$f\big(y \oplus \mathrm{PRF}(k_1, x_1)\big) = f(y^*)$

x_2

$f\big(y \oplus \mathrm{PRF}(k_0, x_2)\big) = f(y^*)$

$y = \mathrm{PRF}(k_1, x_2)$

$\vdots$

x_N

$f\big(y \oplus \mathrm{PRF}(k_0, x_N)\big) = f(y^*)$

$y = \mathrm{PRF}(k_1, x_N)$

"off-path" verification targets

Every statement has **two** possible proofs:

one that is output by the Prove program and one that is not

$\mathrm{Prove}(x, w)$:

- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \mathrm{PRF}(k_{\mathrm{sel}}, x)$
- Output $\pi = \big(b, \mathrm{PRF}(k_b, x)\big)$

Adversary only wins if it outputs x, b, y where

$f\big(y \oplus \mathrm{PRF}(k_b, x)\big) = f(y^*)$ and $b \neq \mathrm{PRF}(k_{\mathrm{sel}}, x)$

Adversary only wins if it outputs an **encryption** of a preimage to $f(y^*)$; reduction only needs a single instance $f(y^*)$ of the OWF!

Summary

CRS contains **two** obfuscated programs

$\text{Prove}(x, w)$:
- If $\mathcal{R}(x, w) = 0$, output $\perp$
- Compute $b \leftarrow \text{PRF}(k_{\text{sel}}, x)$
- Output $\pi = \big(b, \text{PRF}(k_b, x)\big)$

$\text{Verify}(x, \pi)$:
- Parse $\pi = (b, y)$
- If $y = \text{PRF}(k_b, x)$, output 1
- Otherwise, output 0

Scheme relies on sub-exponential secure $i\mathcal{O}$ and sub-exponential secure OWFs

Construction as described relies on **injective** one-way function

[BPW16]: $i\mathcal{O}$ + OWFs ⇒ (keyed) injective OWFs

Alternatively, observe that injective one-way function only shows up in the security proof

Suffices to build injective OWF with an *inefficient* sampler (implied by vanilla OWFs)

[see paper for details]

Summary

This work: Adaptively-sound SNARGs for NP from sub-exponentially-secure $i\mathcal{O}$ and sub-exponentially-secure one-way functions

Large CRS ($|\mathrm{crs}| = \mathrm{poly}(\lambda, |\mathcal{R}|)$), short proofs ($|\pi| = \mathrm{poly}(\lambda)$)

Reduction to falsifiable assumptions runs in time $2^{\Omega(|x|+|w|)}$

Upcoming work [DWW24]**:** fully succinct SNARGs for batch NP from sub-exponentially-secure $i\mathcal{O}$, sub-exponentially secure one-way functions, and rerandomizable one-way functions

Open problems:

- Adaptively-sound SNARGs for NP without $i\mathcal{O}$ (e.g., from LWE)?
- Non-adaptively-sound SNARGs for NP from a polynomial-time falsifiable assumption?

(or extend Gentry-Wichs to rule this out)

Thank you!