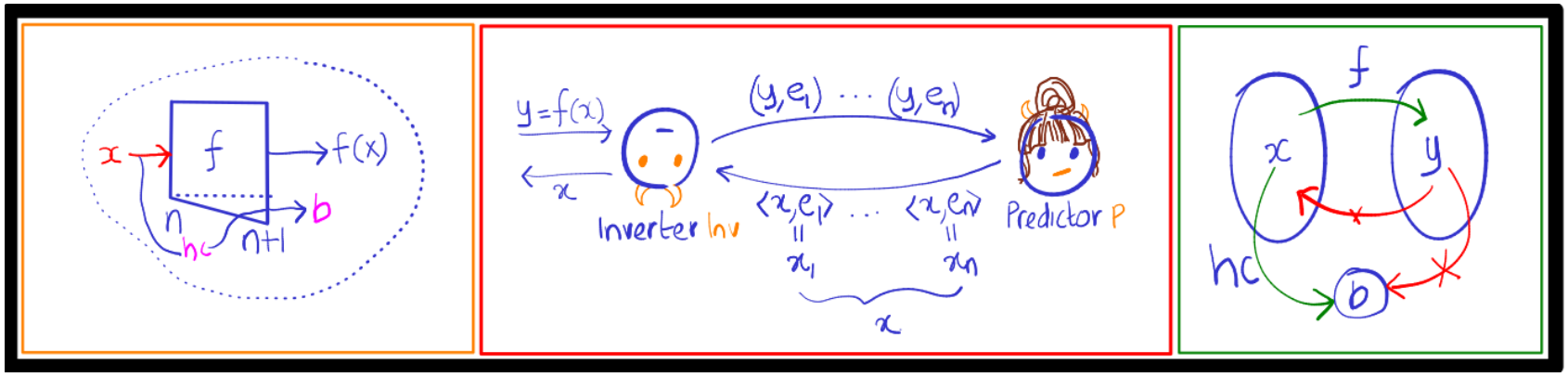



# CS783: Theoretical Foundations of Cryptography

Lecture 6 (16/Aug/24)

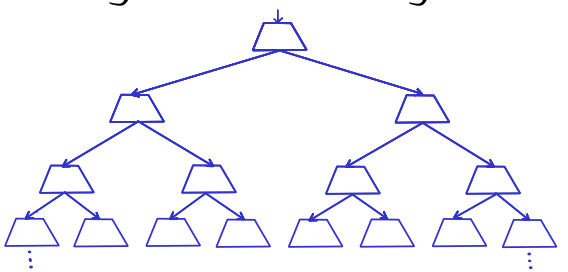
Instructor: Chethan Kamath

# Recall from Last Lecture...

- Sub-task: how to encrypt *multiple* messages using a short key

# Recall from Last Lecture...

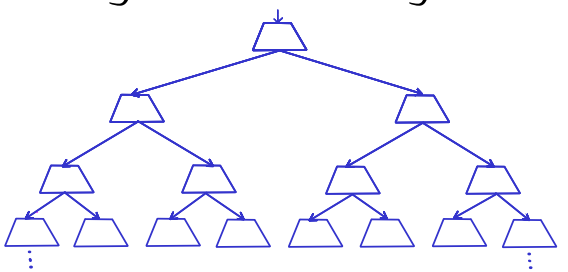
- Sub-task: how to encrypt *multiple* messages using a short key



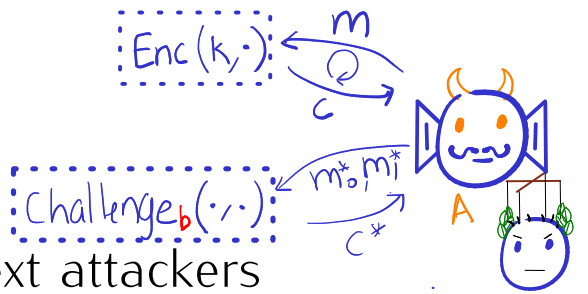
- Sub-task reduces to constructing PRF
- PRG $\rightarrow$ PRF: GGM tree-based construction
  - Proof via hybrid argument

# Recall from Last Lecture...

- Sub-task: how to encrypt *multiple* messages using a short key



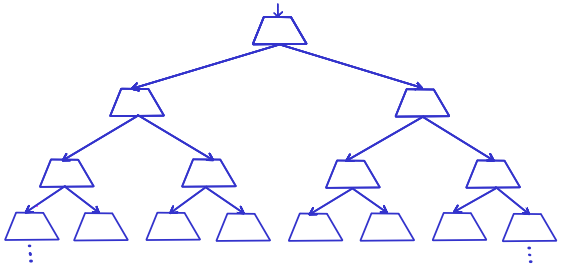
- Sub-task reduces to constructing PRF
- PRG $\rightarrow$ PRF: GGM tree-based construction
  - Proof via hybrid argument



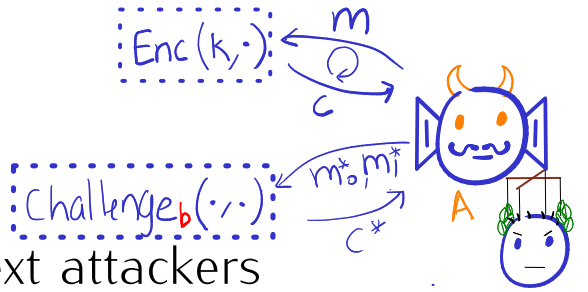

- Chosen-plaintext attack
  - PRF $\rightarrow$ SKE secret against chosen-plaintext attackers

# Recall from Last Lecture...

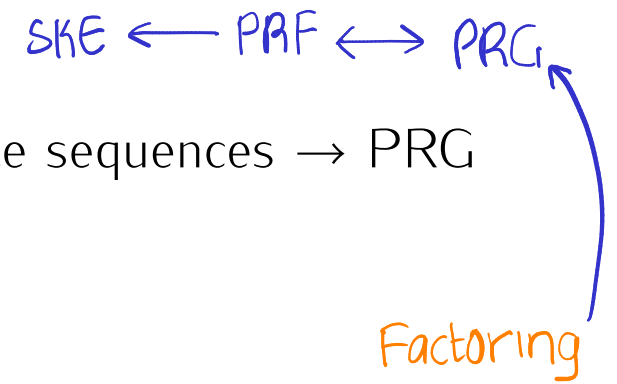
- Sub-task: how to encrypt *multiple* messages using a short key



- Sub-task reduces to constructing PRF
- PRG → PRF: GGM tree-based construction
  - Proof via hybrid argument




- Chosen-plaintext attack
  - PRF → SKE secret against chosen-plaintext attackers

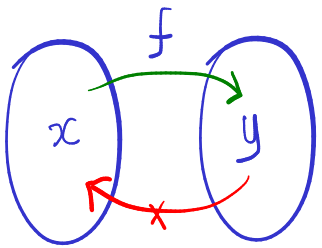



- Everything built on top of PRG
  - Only one construction yet: unpredictable sequences → PRG

⚠ All eggs in one basket

# Plan for This Lecture

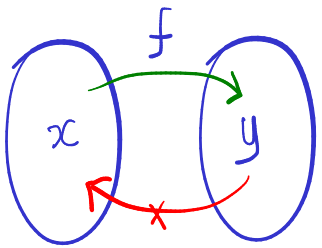
- This lecture: hardness vs. pseudo-randomness
  - Hardness in the form of one-wayness:
    - One-way function (OWF)
    - One-way permutation (OWP)

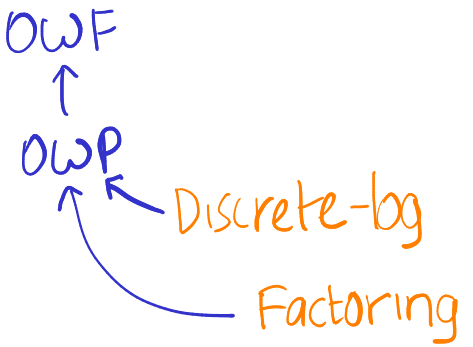



OWF
↑
OWP

# Plan for This Lecture

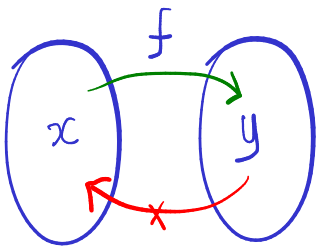
- This lecture: hardness vs. pseudo-randomness
  - Hardness in the form of one-wayness:
    - One-way function (OWF)
    - One-way permutation (OWP)
  - Several candidates for OWF and OWP

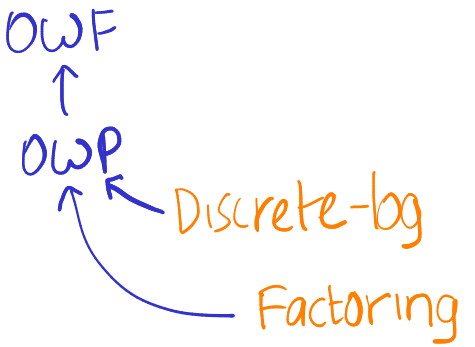


# Plan for This Lecture

- This lecture: hardness vs. pseudo-randomness
  - Hardness in the form of one-wayness:
    - One-way function (OWF)
    - One-way permutation (OWP)
  - Several candidates for OWF and OWP

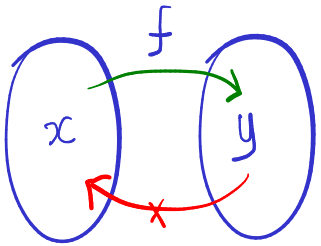


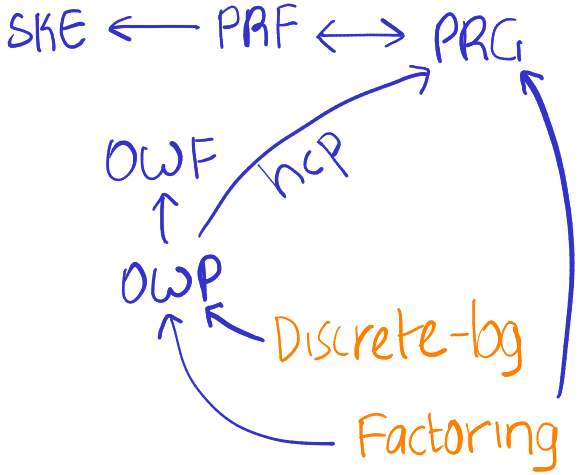




- One-wayness and hard-core predicates
  - Hard-core predicate for OWP $\rightarrow$ PRG
  - Hard-core predicate for any OWP/length-preserving OWF

# Plan for This Lecture

- This lecture: hardness vs. pseudo-randomness
  - Hardness in the form of one-wayness:
    - One-way function (OWF)
    - One-way permutation (OWP)
  - Several candidates for OWF and OWP



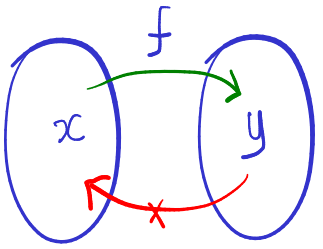




- One-wayness and hard-core predicates
  - Hard-core predicate for OWP $\rightarrow$ PRG
  - Hard-core predicate for any OWP/length-preserving OWF

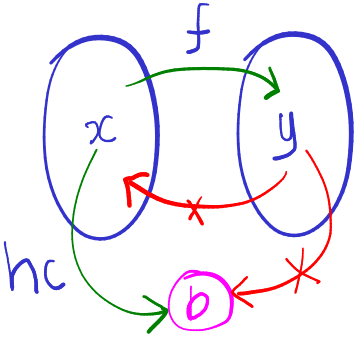
  - Corollary: OWP $\rightarrow$ PRG

# Plan for This Lecture

1. One-Way Functions and Permutations



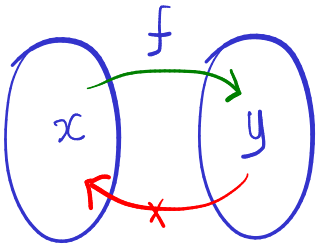

2. Hard-Core Predicate




3. Goldreich-Levin Hard-Core Predicate

# Plan for this Lecture

1. One-Way Functions and Permutations

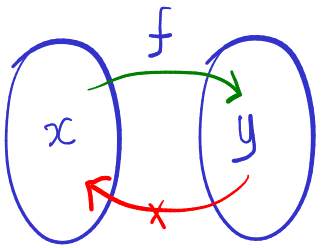



2. Hard-Core Predicate

3. Goldreich-Levin Hard-Core Predicate

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

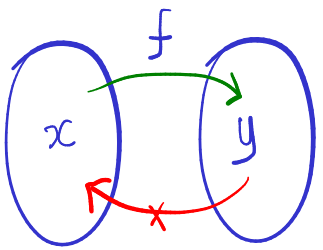


# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

"Solution"




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

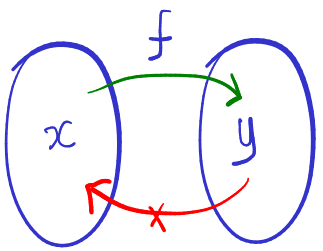
"Puzzle"

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

"Solution"




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

"Puzzle"

- What does "hard to invert" entail? Attempt :

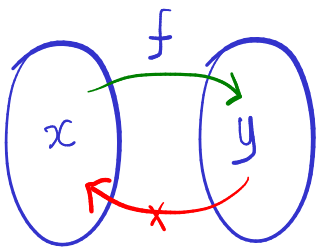
- Problem:

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

"Solution"




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

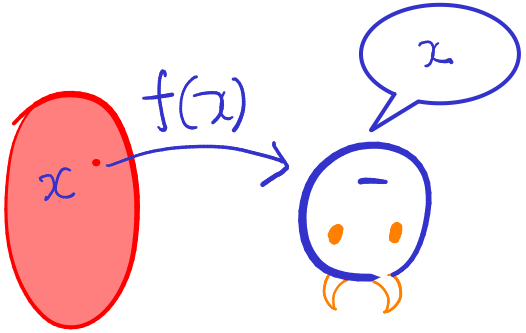
"Puzzle"

- What does "hard to invert" entail? Attempt 1 :

$\forall$ PPT inverter Inv, $\forall x$,

$$\Pr\left[\mathrm{Inv}(f(x)) = x\right]$$

is negligible.



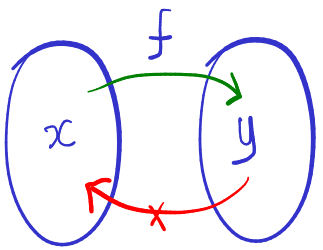

- Problem:

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

"Solution"




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

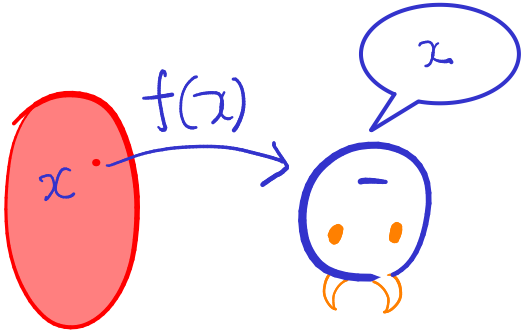
"Puzzle"

- What does "hard to invert" entail? Attempt 1 :

$\forall$ PPT inverter Inv, $\forall x$,

$$\Pr\left[\mathrm{Inv}(f(x)) = x\right]$$

is negligible.



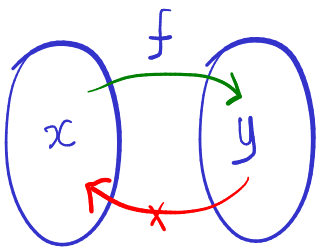

- Problem: Too much to ask (everywhere hardness)

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

"Solution"




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

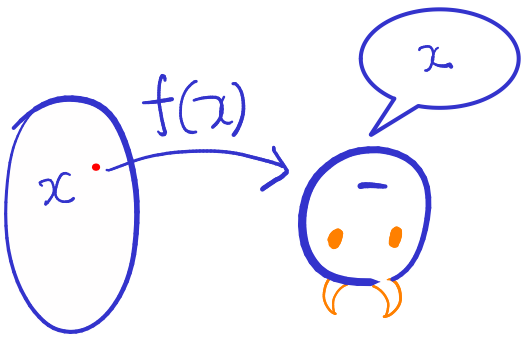
"Puzzle"

- What does "hard to invert" entail? Attempt 2:

$\forall$ PPT inverter Inv, $\exists x$,

$$\Pr\left[\mathrm{Inv}(f(x)) = x\right]$$

is negligible.



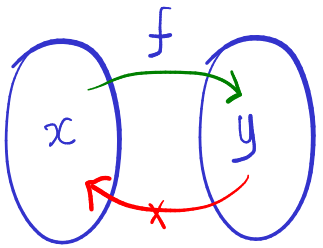

- Problem: Too much to ask (everywhere narrow)

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

"Solution"




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

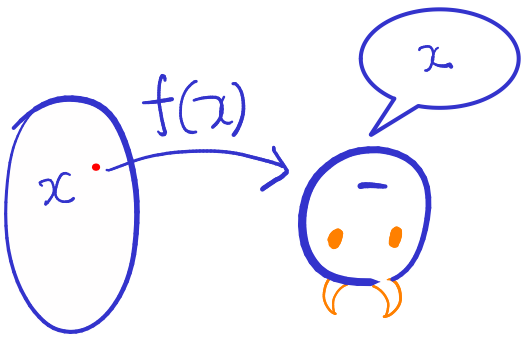
"Puzzle"

- What does "hard to invert" entail? Attempt 2:

$\forall$ PPT inverter Inv, $\exists x$,

$$\Pr\left[\mathrm{Inv}(f(x)) = x\right]$$

is negligible.



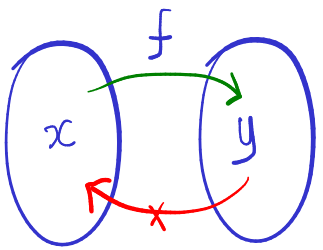

- Problem: This is not sufficient (worst-case hardness)

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

"Solution"




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

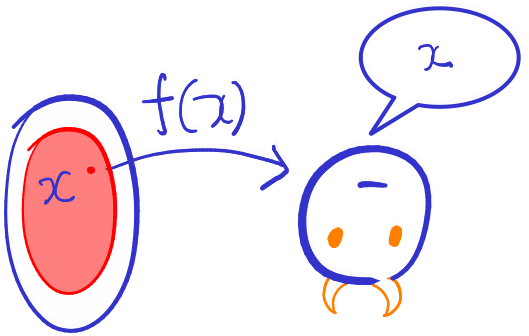
"Puzzle"

- What does "hard to invert" entail? Attempt 3:

$\forall$ PPT inverter Inv

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathrm{Inv}(f(x)) = x]$$

is negligible.



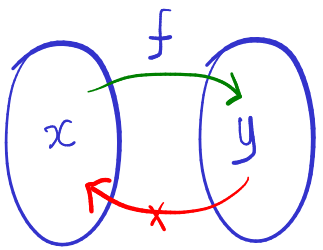

- Problem: This is not sufficient (worst-case hardness)

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

"Solution"




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

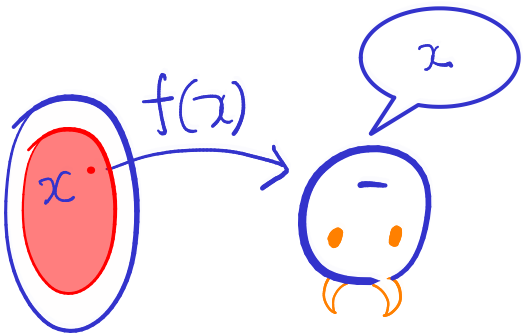
"Puzzle"

- What does "hard to invert" entail? Attempt 3:

$\forall$ PPT inverter Inv

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathrm{Inv}(f(x)) = x]$$

is negligible.



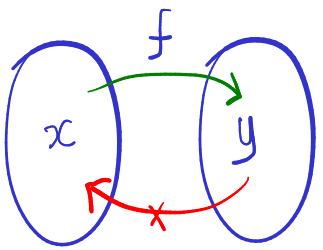

- Problem: What about $f(x) := 0^{|x|}$?

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

"Solution"




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

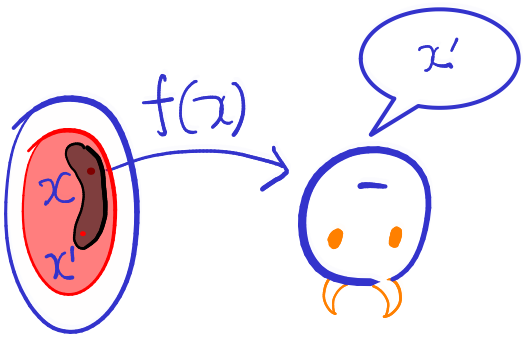
"Puzzle"

- What does "hard to invert" entail? Attempt 4 :

$\forall$ PPT inverter Inv

$$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathrm{Inv}(f(x)) \in f^{-1}(f(x))\right]$$

is negligible.



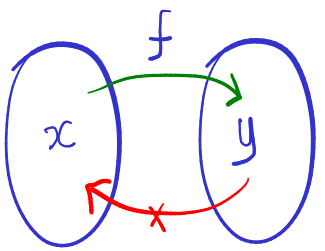

- Problem:

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function $f$ that is "hard to invert"

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

"Solution"




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

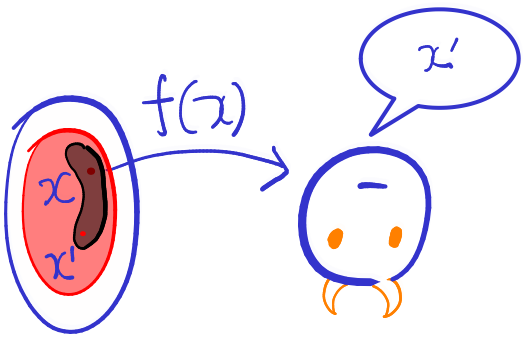
"Puzzle"

- What does "hard to invert" entail? Attempt 4 :

$\forall$ PPT inverter Inv

$$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathrm{Inv}(f(x)) \in f^{-1}(f(x))\right]$$

is negligible.




- Problem: ?

# Let's Define One-Way Functions...

- Intuitively: "easy to compute" function that is "hard to invert"

**Defintion 1 (One-way function (OWF))**

*A function family* $f := \left\{ f_n : \{0,1\}^n \to \{0,1\}^{m(n)} \right\}_{n \in \mathbb{N}}$ *is one-way if*

- *there exists an efficient algorithm* $\mathsf{F}$ *such that* $\forall x : \mathsf{F}(x) = f(x)$
- *for all PPT inverters* $\mathsf{Inv}$*, the following is negligible:*

$$p(n) := \Pr_{x \leftarrow \{0,1\}^n}[\mathsf{Inv}(f_n(x)) \in f_n^{-1}(f_n(x))]$$

# Let's Define One-Way Functions...

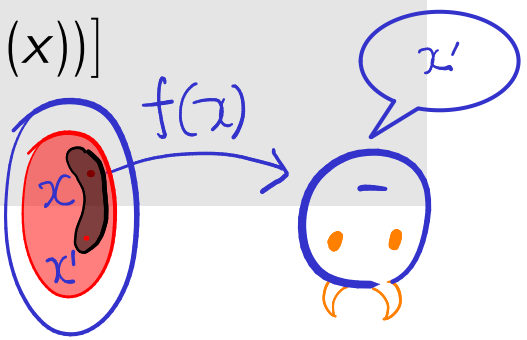
- Intuitively: "easy to compute" function that is "hard to invert"

**Defintion 1 (One-way function (OWF))**

*A function family* $f := \left\{ f_n : \{0,1\}^n \to \{0,1\}^{m(n)} \right\}_{n \in \mathbb{N}}$ *is one-way if*

- *there exists an efficient algorithm* $\mathsf{F}$ *such that* $\forall x : \mathsf{F}(x) = f(x)$
- *for all PPT inverters* $\mathsf{Inv}$*, the following is negligible:*

$$p(n) := \Pr_{x \leftarrow \{0,1\}^n}[\mathsf{Inv}(f_n(x)) \in f_n^{-1}(f_n(x))]$$



# Let's Define One-Way Functions...

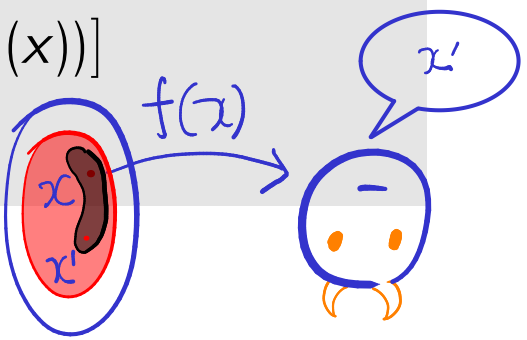
- Intuitively: "easy to compute" function that is "hard to invert"

**Defintion 1 (One-way function (OWF))**

*A function family* $f := \left\{f_n : \{0,1\}^n \to \{0,1\}^{m(n)}\right\}_{n\in\mathbb{N}}$ *is one-way if*

- *there exists an efficient algorithm* $\mathsf{F}$ *such that* $\forall x : \mathsf{F}(x) = f(x)$
- *for all PPT inverters* $\mathsf{Inv}$*, the following is negligible:*

$$p(n) := \Pr_{x\leftarrow\{0,1\}^n}[\mathsf{Inv}(f_n(x)) \in f_n^{-1}(f_n(x))]$$




- Length-preserving OWF: $m(n) = n$
- One-way permutation: $f$ is length-preserving and *bijective*

# Let's Define One-Way Functions...

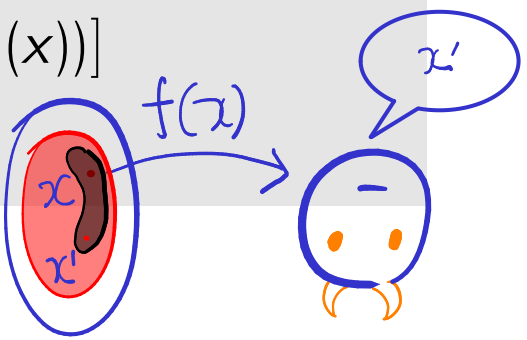
- Intuitively: "easy to compute" function that is "hard to invert"

**Defintion 1 (One-way function (OWF))**

*A function family* $f := \left\{ f_n : \{0,1\}^n \to \{0,1\}^{m(n)} \right\}_{n \in \mathbb{N}}$ *is one-way if*

- *there exists an efficient algorithm* $\mathsf{F}$ *such that* $\forall x : \mathsf{F}(x) = f(x)$
- *for all PPT inverters* $\mathsf{Inv}$*, the following is negligible:*

$$p(n) := \Pr_{x \leftarrow \{0,1\}^n}[\mathsf{Inv}(f_n(x)) \in f_n^{-1}(f_n(x))]$$




- Length-preserving OWF: $m(n) = n$
- One-way permutation: $f$ is length-preserving and *bijective*
- Convenient to consider "collection" of OWF:

$$\left\{ f_I : \mathcal{D}_I \to \mathcal{R}_I \right\}_{I \subseteq \{0,1\}^*}$$

# OWF or Not?

Some generic constructions:

1. $f_1(x) := f(x)0^{|x|}$, where $f$ is a OWF
2. $f_2(x_1x_2) := x_1 f(x_2)$, where $f$ is a OWF
3. $f_3(x_1x_2) := x_1 f(x_1x_2)$, where $f$ is a OWF
4. $f_4(x) := G(x)$, where $G$ is a PRG

# OWF or Not?

Some generic constructions:

1. $f_1(x) := f(x)0^{|x|}$, where $f$ is a OWF
2. $f_2(x_1x_2) := x_1 f(x_2)$, where $f$ is a OWF
3. $f_3(x_1x_2) := x_1 f(x_1x_2)$, where $f$ is a OWF
4. $f_4(x) := G(x)$, where $G$ is a PRG

A concrete construction:

4. $f_5(x_1x_2) := x_1 \cdot x_2$, where $x_1$ and $x_2$ are parsed as integers

# OWF or Not?

Some generic constructions:

1. $f_1(x) := f(x)0^{|x|}$, where $f$ is a OWF
2. $f_2(x_1x_2) := x_1 f(x_2)$, where $f$ is a OWF
3. $f_3(x_1x_2) := x_1 f(x_1x_2)$, where $f$ is a OWF
4. $f_4(x) := G(x)$, where $G$ is a PRG

A concrete construction:

4. $f_5(x_1x_2) := x_1 \cdot x_2$, where $x_1$ and $x_2$ are parsed as integers
   - "Weakly" one-way since primes are dense enough

# OWF or Not?

Some generic constructions:

1. $f_1(x) := f(x)0^{|x|}$, where $f$ is a OWF
2. $f_2(x_1x_2) := x_1 f(x_2)$, where $f$ is a OWF
3. $f_3(x_1x_2) := x_1 f(x_1x_2)$, where $f$ is a OWF
4. $f_4(x) := G(x)$, where $G$ is a PRG

A concrete construction:

4. $f_5(x_1x_2) := x_1 \cdot x_2$, where $x_1$ and $x_2$ are parsed as integers
   - "Weakly" one-way since primes are dense enough

## Exercise 1

1. *Show using security reduction that $f_1$, $f_2$ and $f_4$ are OWFs*
2. *Come up $f$s such that $f_3$ i) remains one-way and ii) becomes invertible*

# OWF Collection or Not?

large ↖ ↗ constant in $\mathbb{Z}_p^*$

1 Multiplication modulo prime $p$: $f_{p,a}(x) := ax \bmod p$

# OWF Collection or Not?

large ↑ constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime $p$: $f_{p,a}(x) := ax \bmod p$
2. Squaring modulo prime $p$: $f_p(x) := x^2 \bmod p$

# OWF Collection or Not?

large ↑ constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime $p$: $f_{p,a}(x) := ax \bmod p$
2. Squaring modulo prime $p$: $f_p(x) := x^2 \bmod p$
3. Exponentiation modulo prime $p$: $f_{p,g}(x) := g^x \bmod p$ → "generator"

# OWF Collection or Not?

large ↑ constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime $p$: $f_{p,a}(x) := ax \bmod p$

2. Squaring modulo prime $p$: $f_p(x) := x^2 \bmod p$

3. Exponentiation modulo prime $p$: $f_{p,g}(x) := g^x \bmod p$
   - Inversion is the discrete logarithm problem: believed hard

"generator"

4. Squaring modulo composite $N = pq$: $f_N(x) := x^2 \bmod N$

large primes

# OWF Collection or Not?

large ↑ constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime $p$: $f_{p,a}(x) := ax \bmod p$
2. Squaring modulo prime $p$: $f_p(x) := x^2 \bmod p$
3. Exponentiation modulo prime $p$: $f_{p,g}(x) := g^x \bmod p$ → "generator"
   - Inversion is the discrete logarithm problem: believed hard

~OWP!

4. Squaring modulo composite $N = pq$: $f_N(x) := x^2 \bmod N$
   - Inversion as hard as factoring $N$! → large primes
5. Matrix multiplication modulo prime $p$: $f_{\bar{A}}(x) := x^T \bar{A} \bmod p$ → $n \times m$ matrix over $\mathbb{Z}_p^*$

# OWF Collection or Not?

large ↑ constant in $\mathbb{Z}_p^*$

1. Multiplication modulo prime $p$: $f_{p,a}(x) := ax \bmod p$
2. Squaring modulo prime $p$: $f_p(x) := x^2 \bmod p$ → "generator"
3. Exponentiation modulo prime $p$: $f_{p,g}(x) := g^x \bmod p$
   - Inversion is the discrete logarithm problem: believed hard
4. Squaring modulo composite $N = pq$: $f_N(x) := x^2 \bmod N$
   - Inversion as hard as factoring $N$! → large primes
5. Matrix multiplication modulo prime $p$: $f_{\bar{A}}(x) := x^T \bar{A} \bmod p$
   - Inversion easy by Gaussian elimination → $n \times m$ matrix over $\mathbb{Z}_p^*$
   - But, inversion seems hard if we add small noise to the output

LWE: coming up in Lecture 10!

**Exercise 2**

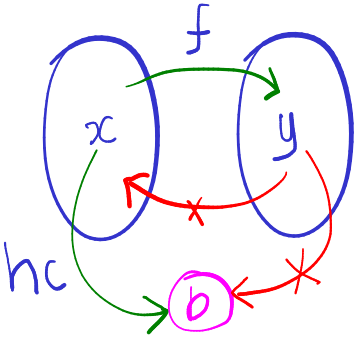
*Show that taking square root modulo $N$ is equivalent to factoring $N$. (Hint: use the identity $x^2 - y^2 = (x + y)(x - y) \bmod N$)*

# Plan for this Lecture

1 One-Way Functions and Permutations




2 Hard-Core Predicate




3 Goldreich-Levin Hard-Core Predicate

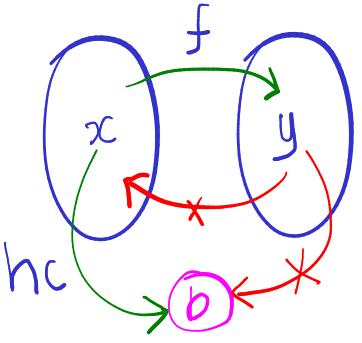
# Plan for this Lecture




1 One-Way Functions and Permutations

"unpredictable"

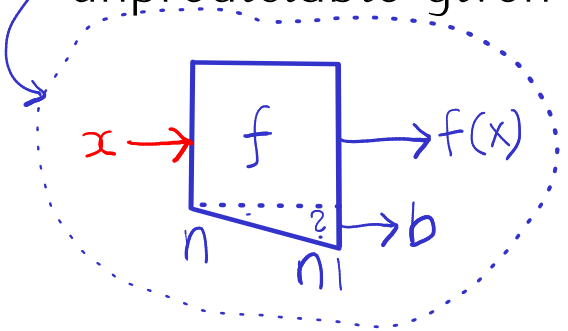
2 Hard-Core Predicate




3 Goldreich-Levin Hard-Core Predicate

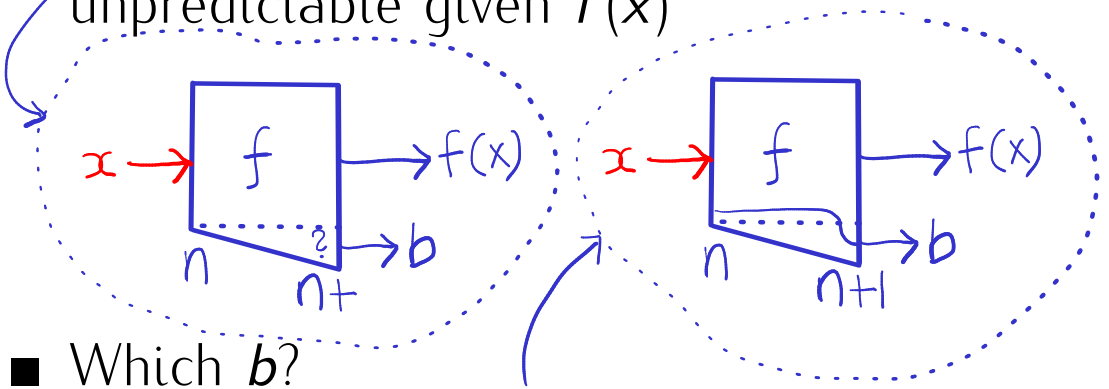
# Let's Try to "Extract" a Hard-Core Bit

- Our goal: length-preserving OWF/OWP $f \to$ PRG $\mathsf{G}$
- Our template: $\mathsf{G}(x) := f(x)b$ for some bit $b \Rightarrow b$ must be unpredictable given $f(x)$



# Let's Try to "Extract" a Hard-Core Bit

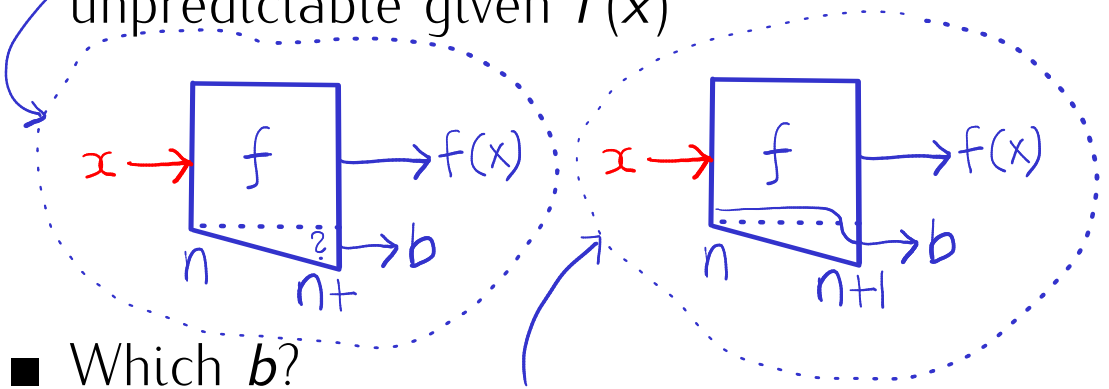
- Our goal: length-preserving OWF/OWP $f \to$ PRG $\mathsf{G}$
- Our template: $\mathsf{G}(x) := f(x)b$ for some bit $b \Rightarrow b$ must be unpredictable given $f(x)$




- Which $b$?
  - What about least significant bit (LSB) $b := x_n$?
    - LSB is unpredictable (e.g.) for squaring function modulo $N$

# Let's Try to "Extract" a Hard-Core Bit

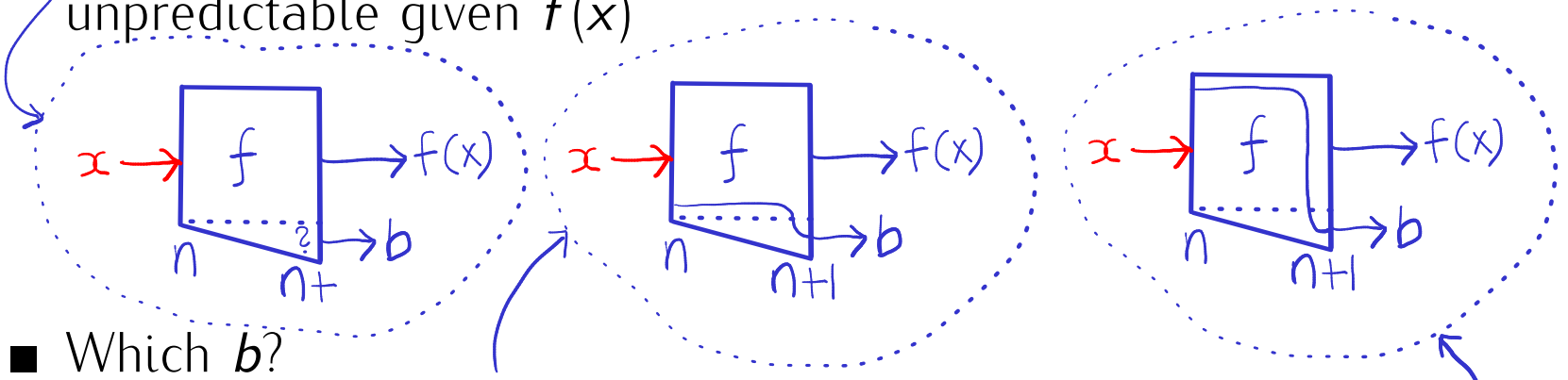
- Our goal: length-preserving OWF/OWP $f \to$ PRG $\mathsf{G}$
- Our template: $\mathsf{G}(x) := f(x)b$ for some bit $b \Rightarrow b$ must be unpredictable given $f(x)$




- Which $b$?
  - What about least significant bit (LSB) $b := x_n$?
    - LSB is unpredictable (e.g.) for squaring function modulo $N$
    - Problem: $x_n$ maybe be leaked in some other $f$s

# Let's Try to "Extract" a Hard-Core Bit

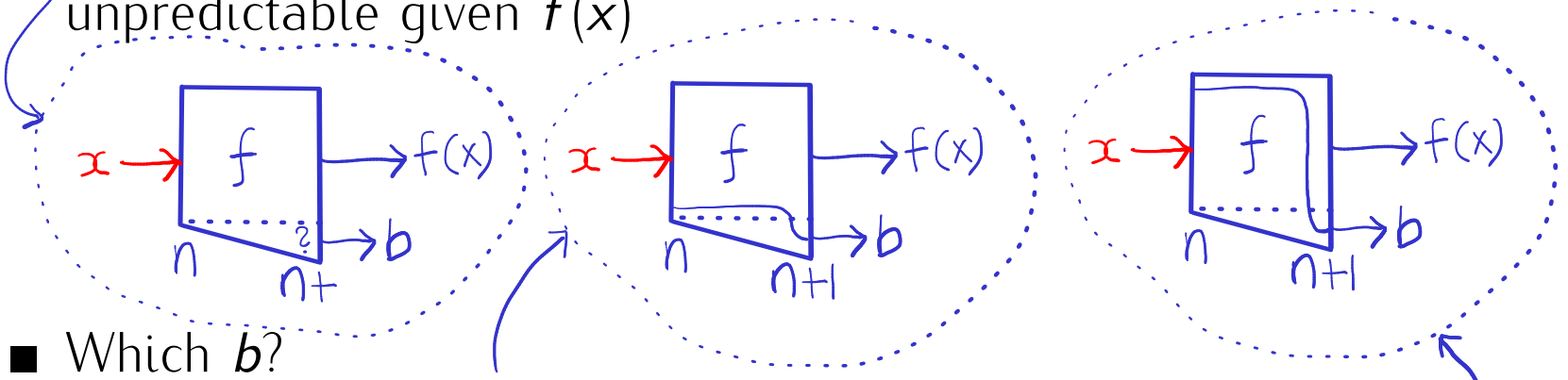
- Our goal: length-preserving OWF/OWP $f \to$ PRG $\mathsf{G}$
- Our template: $\mathsf{G}(x) := f(x)b$ for some bit $b \Rightarrow b$ must be unpredictable given $f(x)$




- Which $b$?
  - What about least significant bit (LSB) $b := x_n$?
    - LSB is unpredictable (e.g.) for squaring function modulo $N$
    - Problem: $x_n$ maybe be leaked in some other $f$s
  - Maybe there is some bit $x_i$ that is not leaked?
    - MSB is unpredictable (e.g) for exponentiation function modulo $p$

# Let's Try to "Extract" a Hard-Core Bit

- Our goal: length-preserving OWF/OWP $f \rightarrow$ PRG $\mathsf{G}$
- Our template: $\mathsf{G}(x) := f(x)b$ for some bit $b \Rightarrow b$ must be unpredictable given $f(x)$




- Which $b$?
  - What about least significant bit (LSB) $b := x_n$?
    - LSB is unpredictable (e.g.) for squaring function modulo $N$
    - Problem: $x_n$ maybe be leaked in some other $f$s
  - Maybe there is some bit $x_i$ that is not leaked?
    - MSB is unpredictable (e.g) for exponentiation function modulo $p$
    - Problem: There exist OWFs such that each bit is predictable with a non-negligible probability

**Exercise 3**

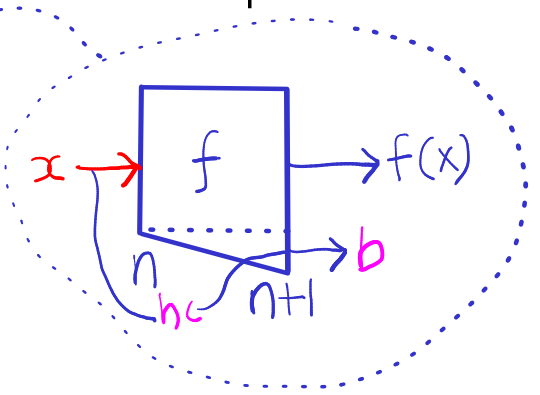
*Come up with such a OWF. What about such a OWP?*

# Therefore, Hard-core Predicates...

- Takeaway: cannot just output some bit of the input

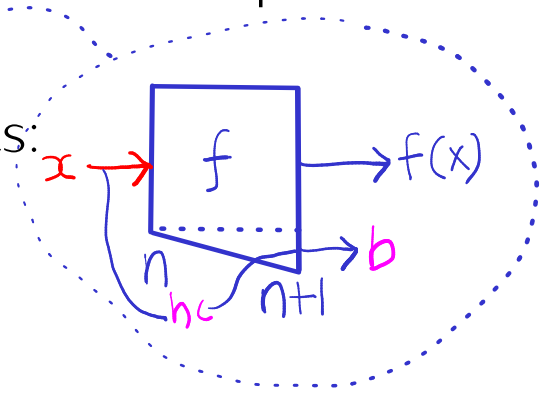
# Therefore, Hard-core Predicates...

- Takeaway: cannot just output some bit of the input
- What about a predicate of the input?



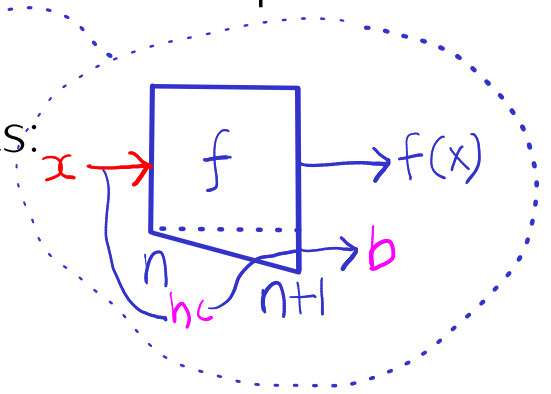
# Therefore, Hard-core Predicates...

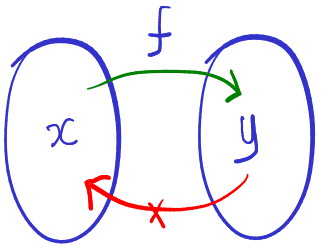
- Takeaway: cannot just output some bit of the input
- What about a predicate of the input?
- Intuitively, we need a predicate that is:
  - Easy to compute given $x$
  - Hard to predict given $f(x)$



# Therefore, Hard-core Predicates...

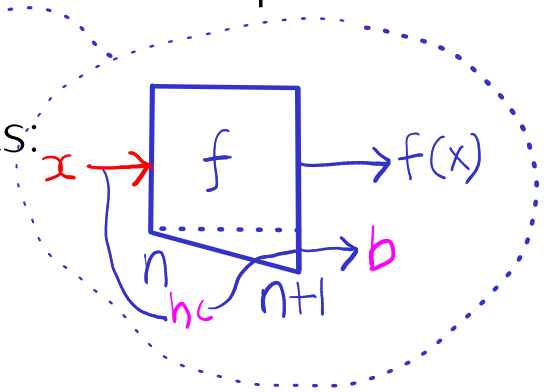
- Takeaway: cannot just output some bit of the input
- What about a predicate of the input?
- Intuitively, we need a predicate that is:
  - Easy to compute given $x$
  - Hard to predict given $f(x)$

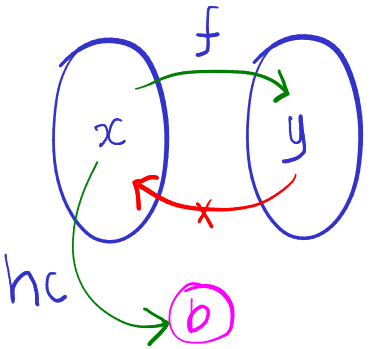


# Therefore, Hard-core Predicates...

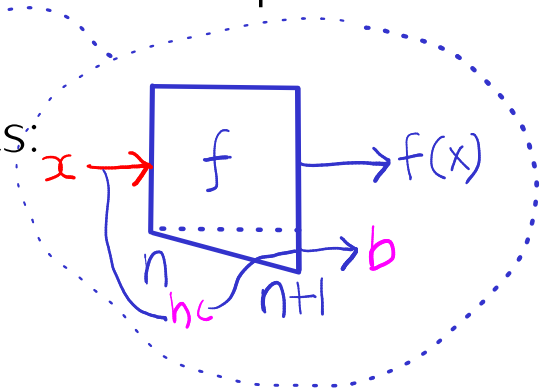
- Takeaway: cannot just output some bit of the input
- What about a predicate of the input?
- Intuitively, we need a predicate that is:
  - Easy to compute given $x$
  - Hard to predict given $f(x)$

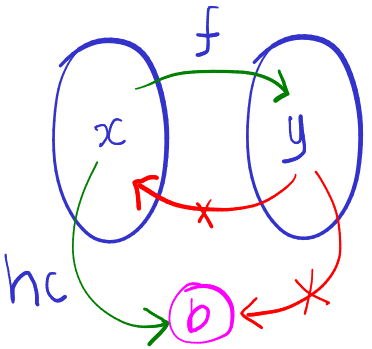


# Therefore, Hard-core Predicates...

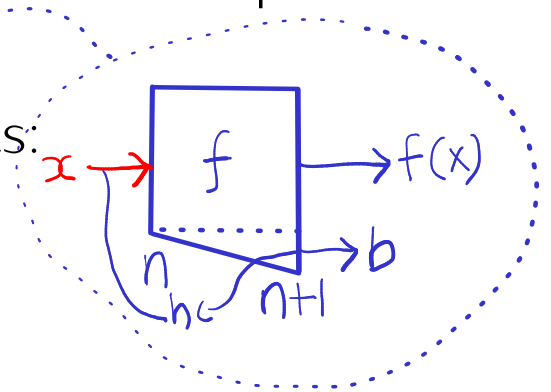
- Takeaway: cannot just output some bit of the input
- What about a predicate of the input?
- Intuitively, we need a predicate that is:
  - Easy to compute given $x$
  - Hard to predict given $f(x)$

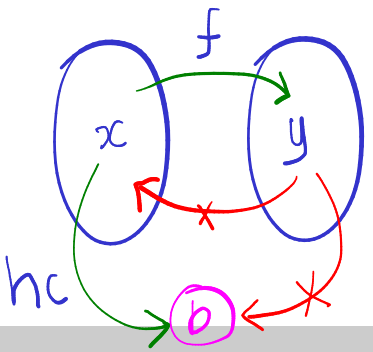


# Therefore, Hard-core Predicates...

- Takeaway: cannot just output some bit of the input
- What about a predicate of the input?
- Intuitively, we need a predicate that is:
  - Easy to compute given $x$
  - Hard to predict given $f(x)$




**Defintion 2**

*A predicate* $hc : \{0,1\}^n \to \{0,1\}$ *is hard-core for a function family* $f_n : \{0,1\}^n \to \{0,1\}^m$*, if for every PPT predictor* $\mathsf{P}$*, the following is negligible*

$$\delta(n) := \Pr_{x \leftarrow \{0,1\}^n}[\mathsf{P}(f(x)) = hc(x)] - 1/2$$

# Therefore, Hard-core Predicates...

**Defintion 2**

*A predicate* $hc : \{0,1\}^n \to \{0,1\}$ *is hard-core for a function family* $f_n : \{0,1\}^n \to \{0,1\}^m$*, if for every PPT predictor* P*, the following is negligible*

$$\delta(n) := \Pr_{x \leftarrow \{0,1\}^n}[\mathsf{P}(f(x)) = hc(x)] - 1/2$$

- For "lossy" functions (e.g., $f_n(x) = 0^n$), unpredictability stems from information loss
- For "non-lossy" functions (e.g., OWP), unpredictability is computational and stems from one-wayness

# Hard-Core Predicate $\rightarrow$ Pseudo-Random Generator...

**Theorem 1**

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $\mathsf{G}(x) := f(x)\mathbf{hc}(x)$ is a PRG.*

# Hard-Core Predicate → Pseudo-Random Generator...

**Theorem 1**

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $\mathsf{G}(x) := f(x)\mathbf{hc}(x)$ is a PRG.*

**Proof sketch.**  Idea: use next-bit unpredictability.

# Hard-Core Predicate → Pseudo-Random Generator...

**Theorem 1**

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)\mathbf{hc}(x)$ is a PRG.* ⇒ G is next-bit unpredictable

**Proof sketch.**  Idea: use next-bit unpredictability.

Goal: ∃ predictor $P'$ for hc ⇐ ∃ next-bit predictor $P$ for G

# Hard-Core Predicate → Pseudo-Random Generator...

**Theorem 1**

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)\mathbf{hc}(x)$ is a PRG.* ⇒ G is next-bit unpredictable

**Proof sketch.** Idea: use next-bit unpredictability.

Goal: ∃ predictor P' for hc ⇐ ∃ next-bit predictor P for G

Predictor P'

NB Predictor P

# Hard-Core Predicate → Pseudo-Random Generator...

## Theorem 1

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)\mathbf{hc}(x)$ is a PRG.* ⇒ G is next-bit unpredictable

## Proof sketch.

Idea: use next-bit unpredictability.

Goal: ∃ predictor P' for hc ⇐ ∃ next-bit predictor P for G

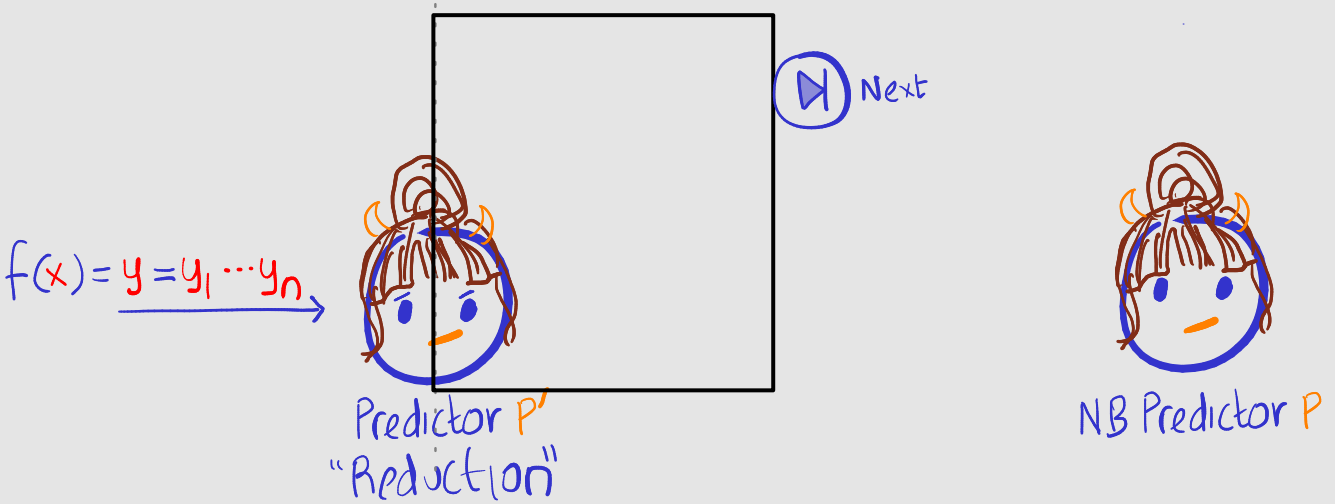
# Hard-Core Predicate → Pseudo-Random Generator...

## Theorem 1

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)\mathbf{hc}(x)$ is a PRG.* ⇒ G is next-bit unpredictable

## Proof sketch. Idea: use next-bit unpredictability.

Goal: ∃ predictor P' for hc ⇐ ∃ next-bit predictor P for G



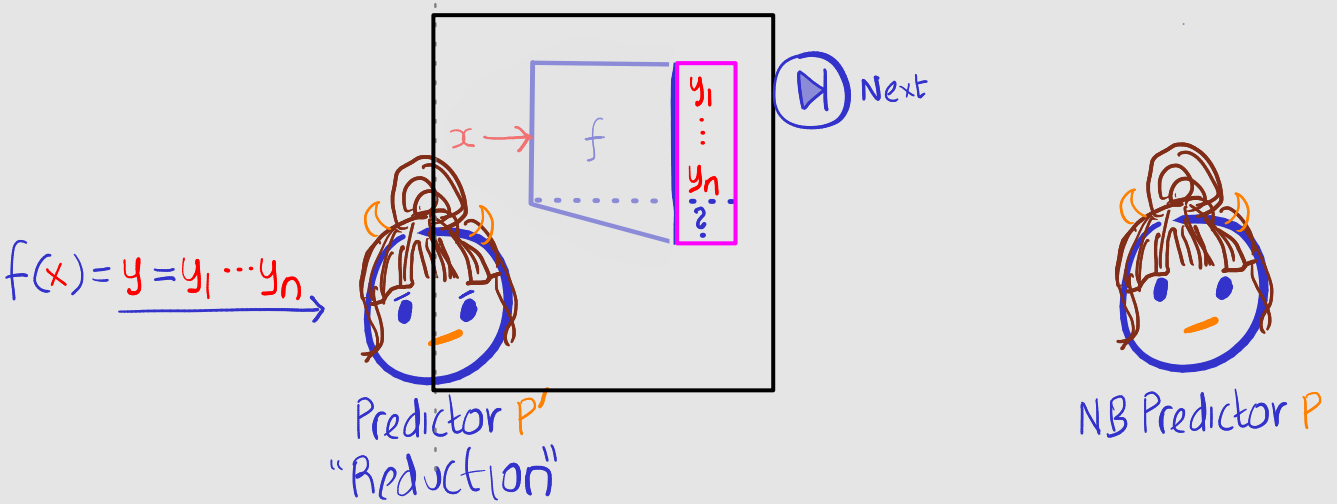
# Hard-Core Predicate → Pseudo-Random Generator...

## Theorem 1

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)\mathbf{hc}(x)$ is a PRG.* ⇒ G is next-bit unpredictable

## Proof sketch. Idea: use next-bit unpredictability.

Goal: ∃ predictor P' for hc ⇐ ∃ next-bit predictor P for G



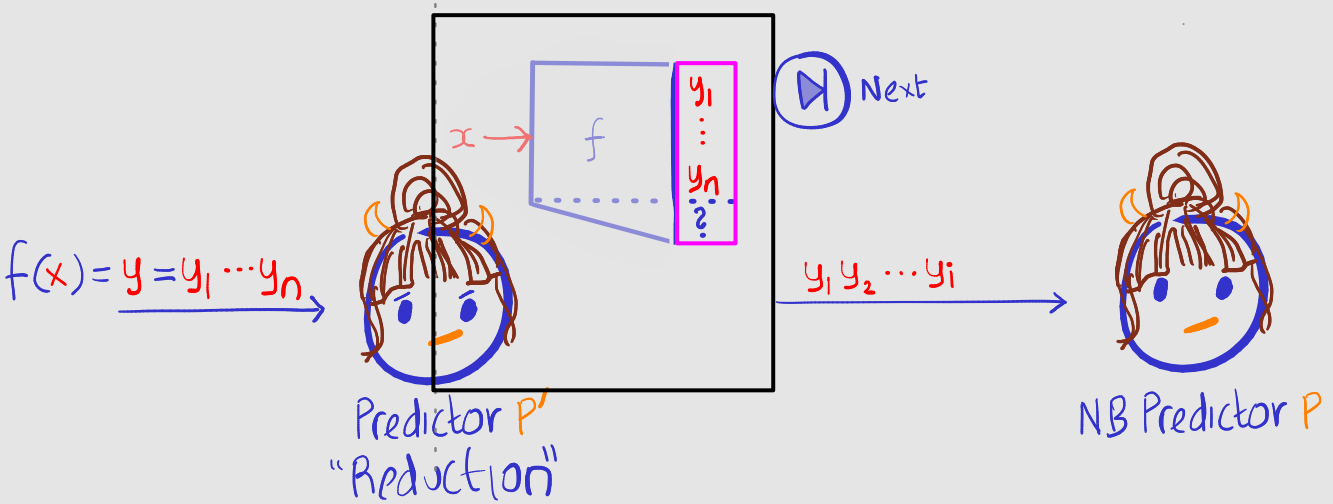
# Hard-Core Predicate → Pseudo-Random Generator...

## Theorem 1

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)\mathbf{hc}(x)$ is a PRG.* ⇒ G is next-bit unpredictable

## Proof sketch. Idea: use next-bit unpredictability.

Goal: ∃ predictor P' for hc ⇐ ∃ next-bit predictor P for G



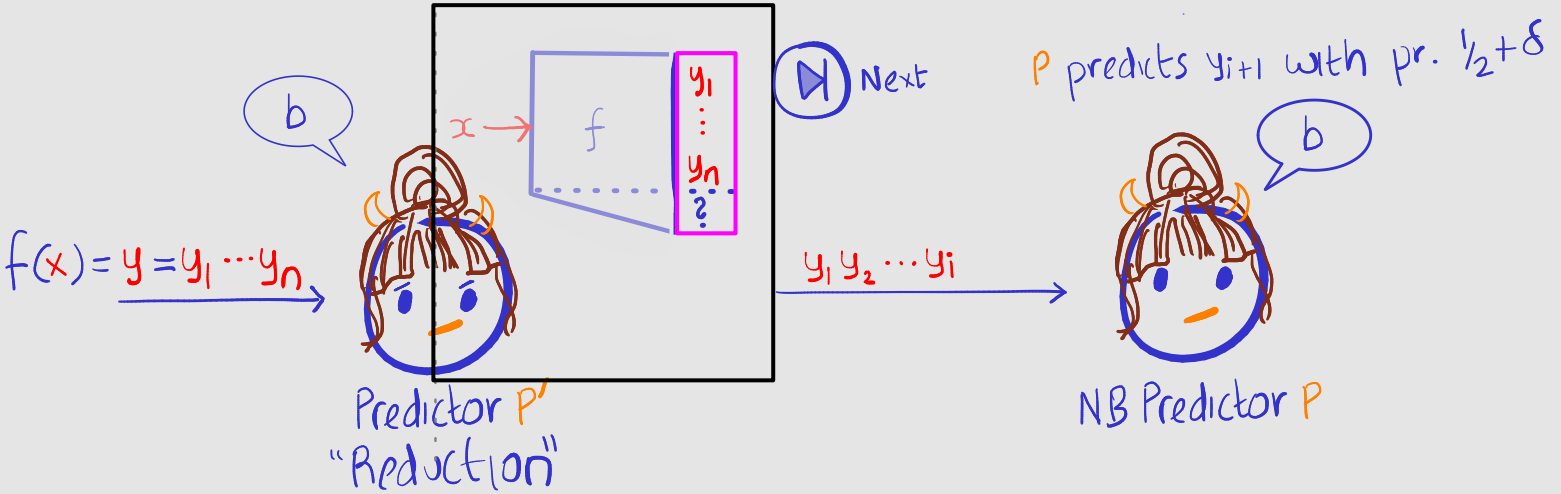
# Hard-Core Predicate → Pseudo-Random Generator...

## Theorem 1

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)\mathbf{hc}(x)$ is a PRG.* ⇒ G is next-bit unpredictable

## Proof sketch. Idea: use next-bit unpredictability.

Goal: ∃ predictor P' for hc ⇐ ∃ next-bit predictor P for G



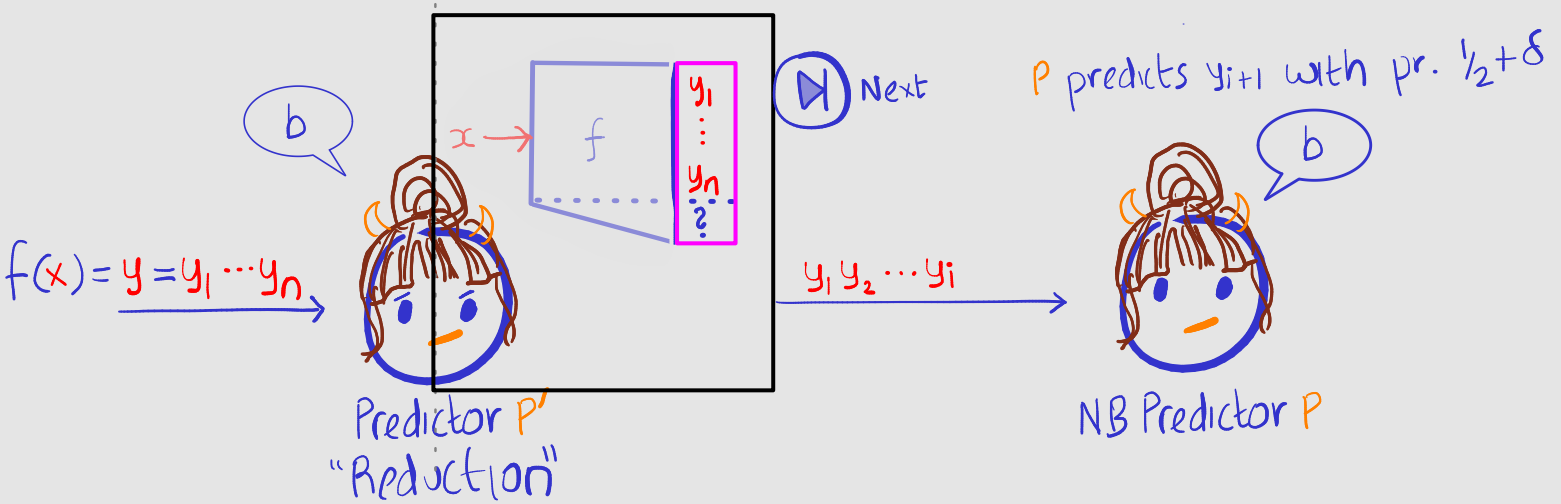
# Hard-Core Predicate → Pseudo-Random Generator...

## Theorem 1

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)\mathbf{hc}(x)$ is a PRG.* ⇒ G is next-bit unpredictable

## Proof sketch. Idea: use next-bit unpredictability.

Goal: ∃ predictor P' for hc ⇐ ∃ next-bit predictor P for G




i) P can only be exploited if i=n

# Hard-Core Predicate → Pseudo-Random Generator

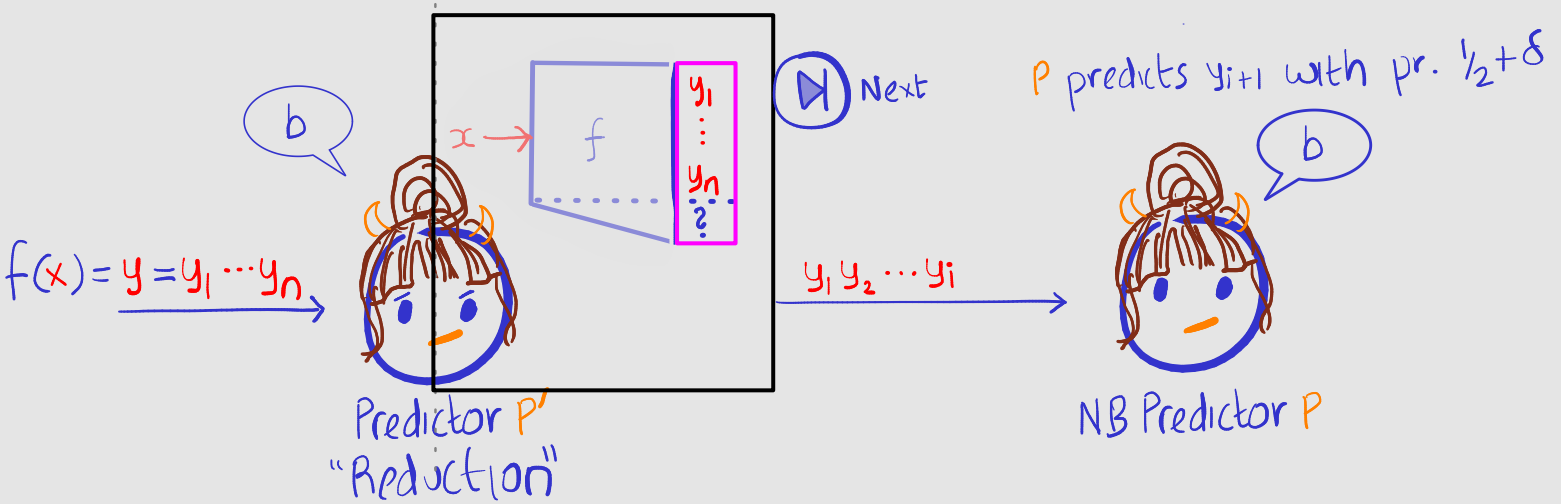
## Theorem 1

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)\mathbf{hc}(x)$ is a PRG.* ⇒ G is next-bit unpredictable

## Proof sketch.

Idea: use next-bit unpredictability.

Goal: ∃ predictor P' for hc ⇐ ∃ next-bit predictor P for G




1) P can only be exploited if $i=n$ ?
2) P has no advantage in predicting $y_1 \cdots y_n$!

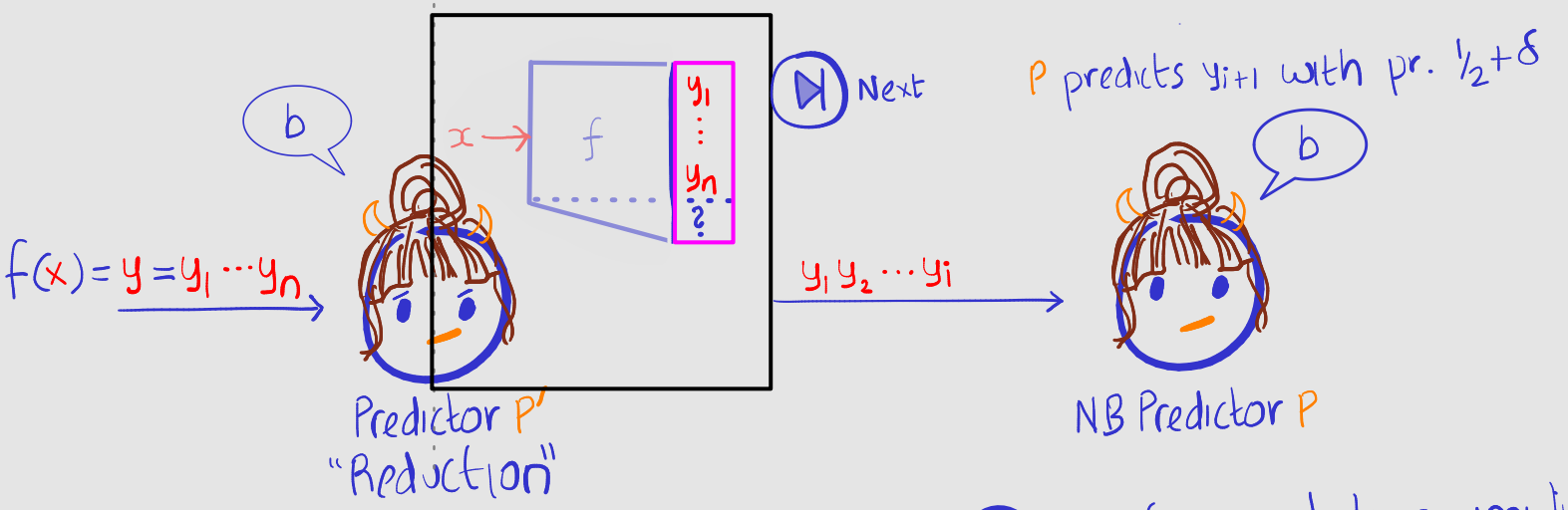
# Hard-Core Predicate → Pseudo-Random Generator

## Theorem 1

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)\mathbf{hc}(x)$ is a PRG.* ⇒ G is next-bit unpredictable

## Proof sketch. Idea: use next-bit unpredictability.

Goal: ∄ predictor P' for hc ⇐ ∄ next-bit predictor P for G




1) P can only be exploited if $i=n$
2) P has no advantage in predicting $y_1 \ldots y_n$!

? → f permutation implies $y_1 \ldots y_n$ is uniformly random

# Hard-Core Predicate $\rightarrow$ Pseudo-Random Generator...

**Theorem 1**

*Let $f$ be a OWP and $\mathbf{hc}$ be a hard-core predicate for $f$. Then the PRG $\mathsf{G}(x) := f(x)\mathbf{hc}(x)$ is a PRG.*

# Hard-Core Predicate $\rightarrow$ Pseudo-Random Generator...

**Theorem 1**

*Let $\mathsf{f}$ be a OWP and $\mathsf{hc}$ be a hard-core predicate for $\mathsf{f}$. Then the PRG $\mathsf{G}(x) := \mathsf{f}(x)\mathsf{hc}(x)$ is a PRG.*

**Exercise 4**

*What happens if we use a length-preserving OWF instead of OWP?*

# Hard-Core Predicate → Pseudo-Random Generator...

## Theorem 1

*Let $f$ be a OWP and $hc$ be a hard-core predicate for $f$. Then the PRG $G(x) := f(x)hc(x)$ is a PRG.*

## Exercise 4

*What happens if we use a length-preserving OWF instead of OWP?*

## Corollary 2

- *We get PRG from hardness of following problems:*
  - *Exponentiation modulo prime $p$: $f_{p,g}(x) := g^x \bmod p$*
    - *Inversion hard by discrete logarithm assumption*
  - *Squaring modulo composite $N = pq$: $f_N(x) := x^2 \bmod N$*
    - *Inversion hard by factoring assumption*

# Plan for this Lecture




1 One-Way Functions and Permutations




2 Hard-Core Predicate

3 Goldreich-Levin Hard-Core Predicate

# Hard-core Predicate for "Any" OWP

- We want: one hard-core predicate that works for every OWP

# Hard-core Predicate for "Any" OWP

- We want: one hard-core predicate that works for every OWP
  
  This is not quite possible

**Exercise 5**

*Show that one function* $\mathbf{hc} : \{0,1\}^n \to \{0,1\}$ *cannot be hard-core predicate for all one-way functions.*

# Hard-core Predicate for "Any" OWP

- We want: one hard-core predicate that works for every OWP

   This is not quite possible

**Exercise 5**

*Show that one function $hc : \{0,1\}^n \to \{0,1\}$ cannot be hard-core predicate for all one-way functions.*

- Compromise:
  1. OWP $\to$ "leaky" OWP
     - For a OWP $f$, the "leaky" OWP is $f'(x, r) := (f(x), r)$.
  2. One hard-core predicate that works for every "leaky" OWP

# Hard-core Predicate for "Any" OWP

- We want: one hard-core predicate that works for every OWP
  - ⚠ This is not quite possible

**Exercise 5**

*Show that one function $hc : \{0,1\}^n \to \{0,1\}$ cannot be hard-core predicate for all one-way functions.*

- Compromise:
  1. OWP $\to$ "leaky" OWP
     - For a OWP $f$, the "leaky" OWP is $f'(x, r) := (f(x), r)$.
  2. One hard-core predicate that works for every "leaky" OWP

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $hc(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

↝ leaky OWP

↑ inner product modulo 2

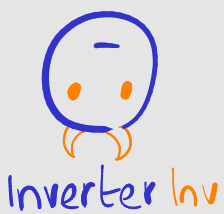
$\sum_{i \in [1,n]} x_i r_i \bmod 2$

# Let's First Prove the Simplest Case: Perfect Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. $\exists$ Inverter $\mathbf{Inv} \Leftarrow \exists$ Perfect Predictor $\mathbf{P}$.**




# Let's First Prove the Simplest Case: Perfect Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

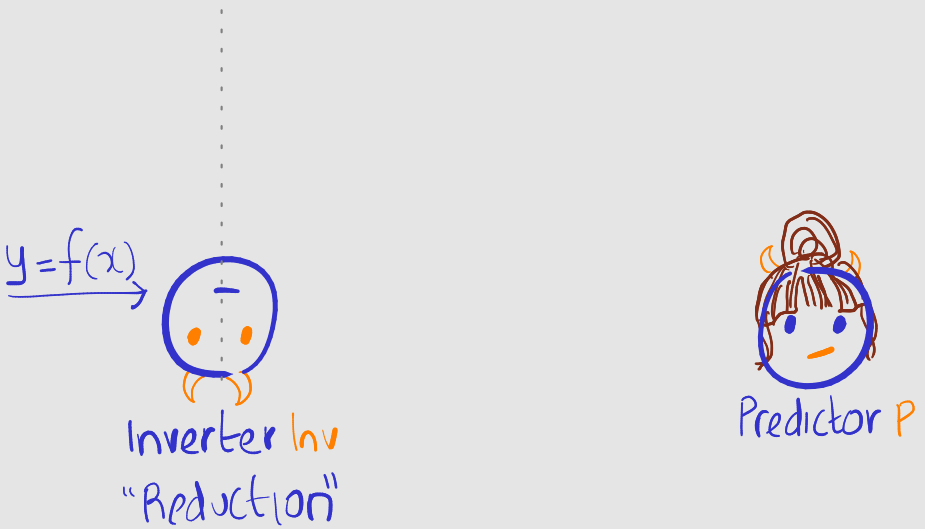
**Proof. $\exists$ Inverter $\mathbf{Inv} \Leftarrow \exists$ Perfect Predictor $\mathbf{P}$.**


# Let's First Prove the Simplest Case: Perfect Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. $\exists$ Inverter $\mathbf{Inv} \Leftarrow \exists$ Perfect Predictor $\mathbf{P}$.**

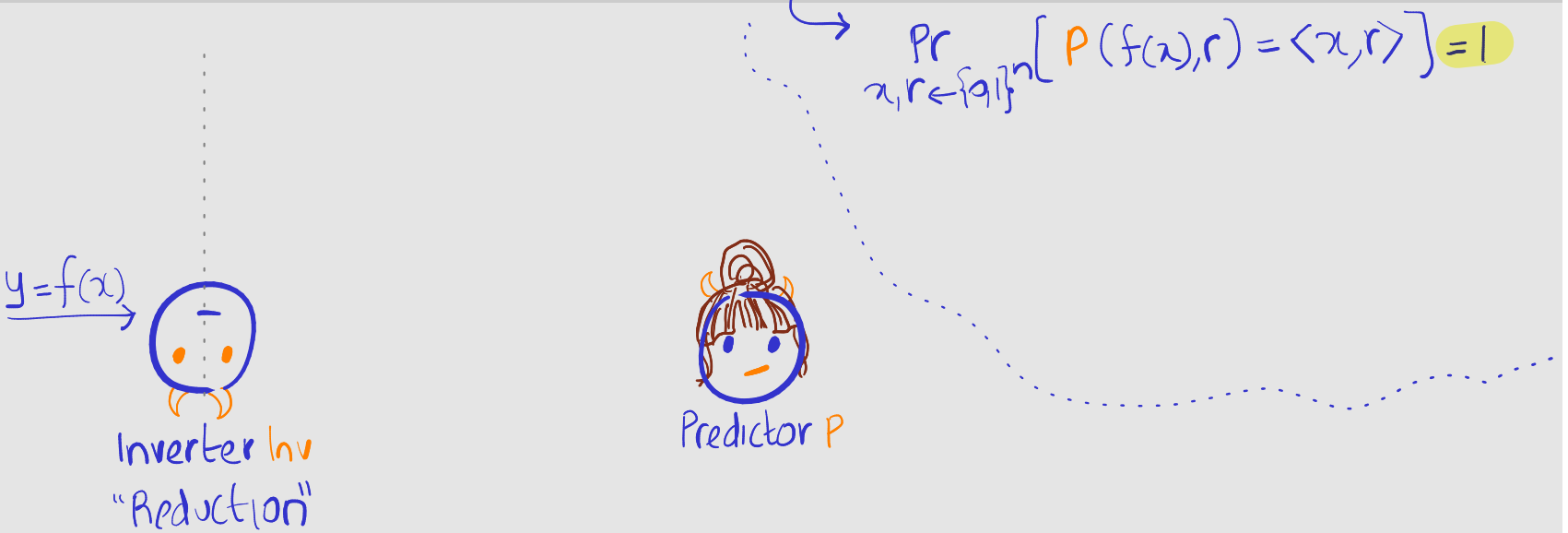


# Let's First Prove the Simplest Case: Perfect Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x),r)$. Then $\mathbf{hc}(x,r) := \langle x,r\rangle_2$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter $\mathbf{Inv} \Leftarrow \exists$ Perfect Predictor $\mathbf{P}$.

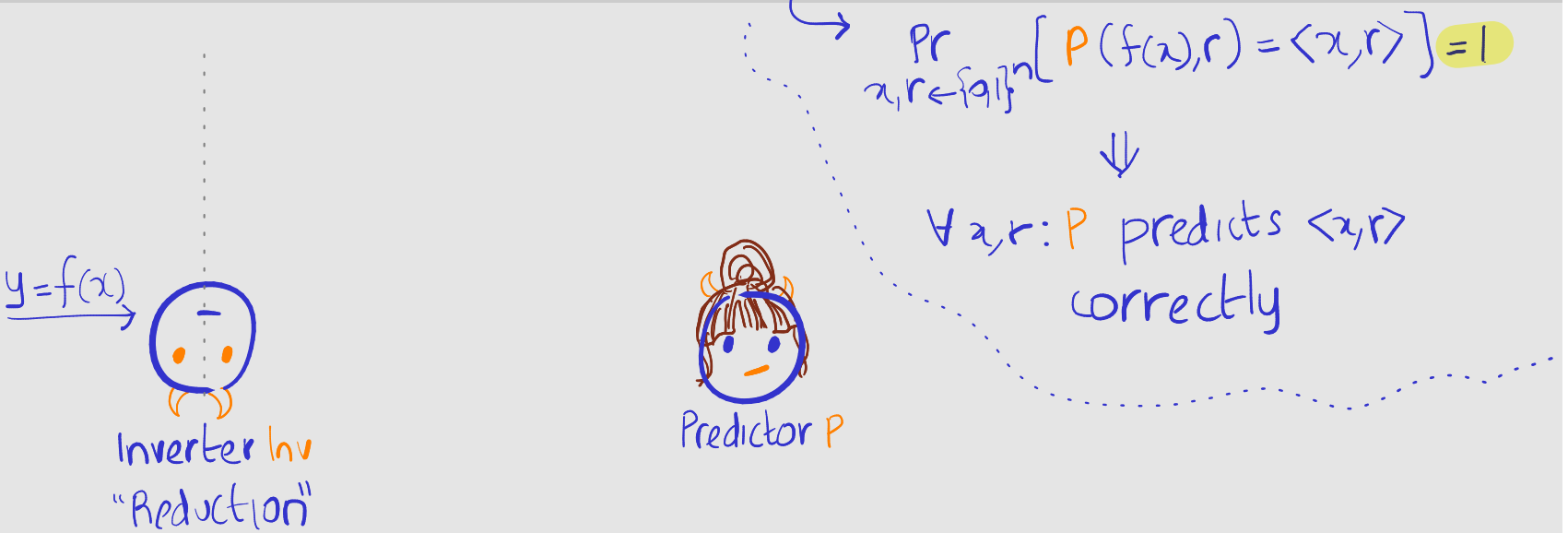


# Let's First Prove the Simplest Case: Perfect Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $\mathbf{hc}(x,r) := \langle x, r \rangle$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter $\mathbf{Inv} \Leftarrow \exists$ Perfect Predictor $\mathbf{P}$.

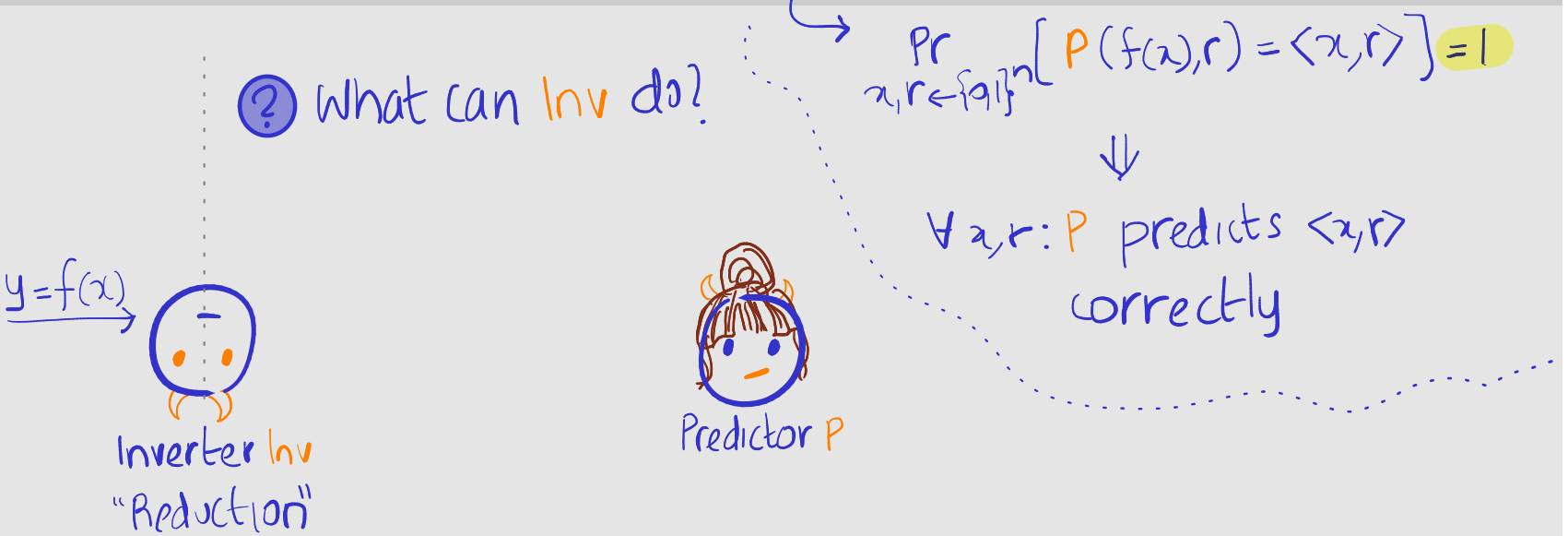


# Let's First Prove the Simplest Case: Perfect Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $\mathbf{hc}(x,r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter **Inv** $\Leftarrow \exists$ Perfect Predictor **P**.

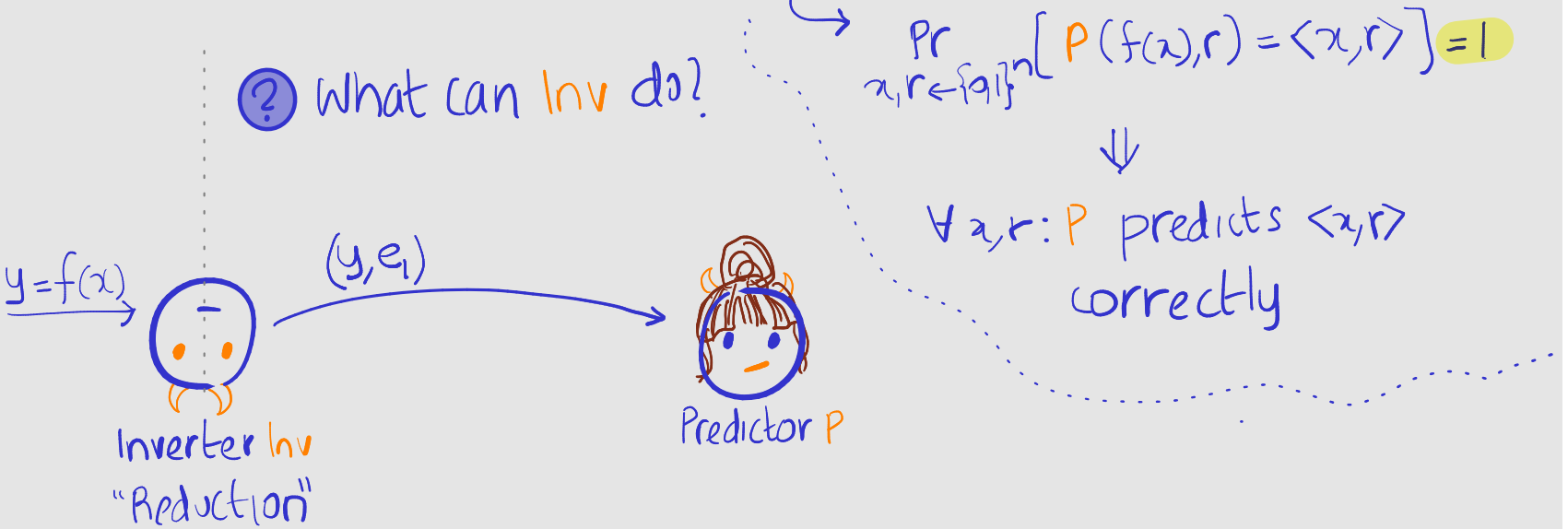


# Let's First Prove the Simplest Case: Perfect Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $\mathbf{hc}(x,r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter $\mathbf{Inv} \Leftarrow \exists$ Perfect Predictor $\mathbf{P}$.

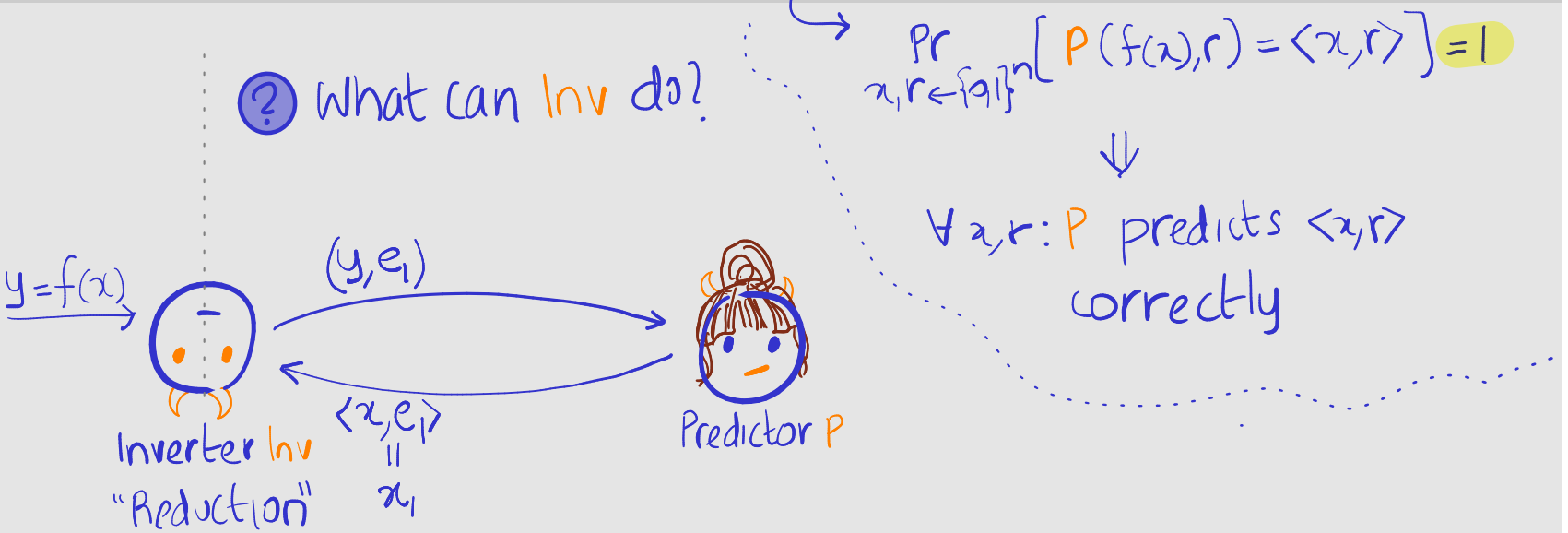


# Let's First Prove the Simplest Case: Perfect Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $\mathbf{hc}(x,r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter $\mathbf{Inv} \Leftarrow \exists$ Perfect Predictor $\mathbf{P}$.

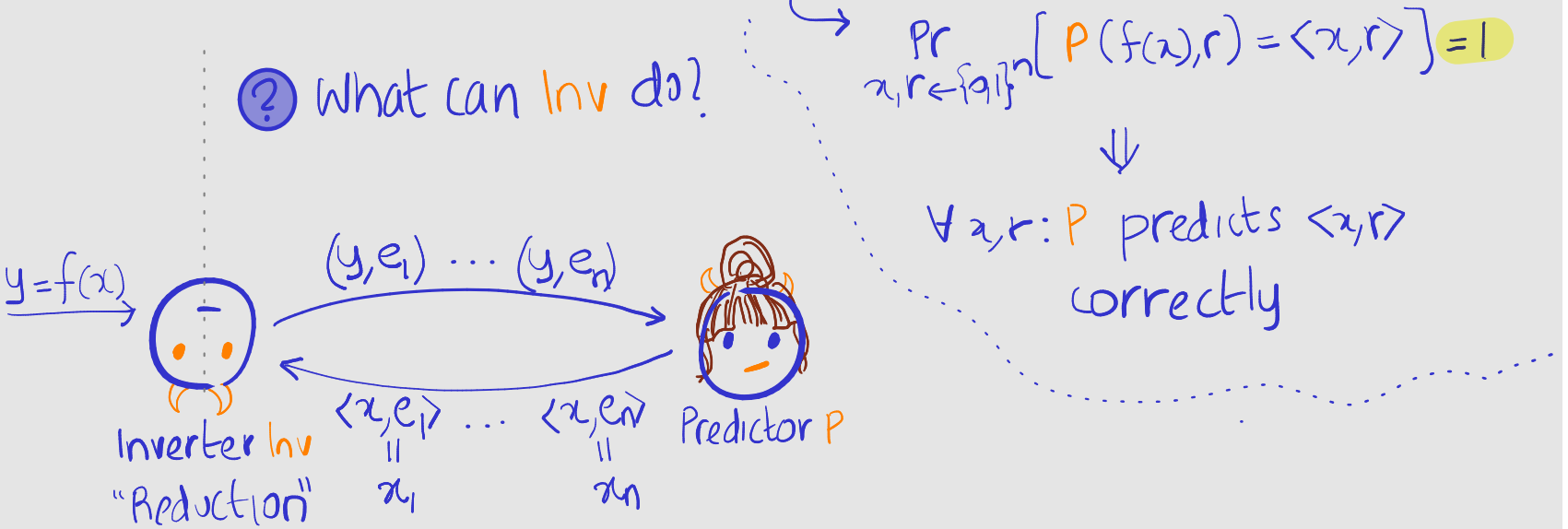


# Let's First Prove the Simplest Case: Perfect Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $\mathbf{hc}(x,r) := \langle x, r \rangle$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter $\mathbf{Inv} \Leftarrow \exists$ Perfect Predictor $\mathbf{P}$.



# Let's First Prove the Simplest Case: Perfect Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

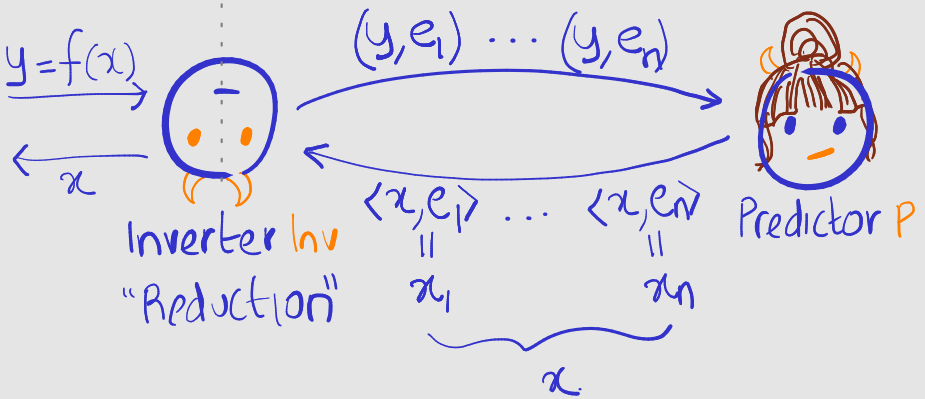
*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $\mathbf{hc}(x,r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter **Inv** $\Leftarrow \exists$ Perfect Predictor **P**.

$\Pr_{x,r \leftarrow \{0,1\}^n}[P(f(x),r) = \langle x,r \rangle] = 1$

$\Downarrow$

$\forall x,r$: P predicts $\langle x,r \rangle$ correctly



# Next Let's Consider "Good" Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter $\mathsf{Inv} \Leftarrow \exists$ Good Predictor $\mathsf{P}$.

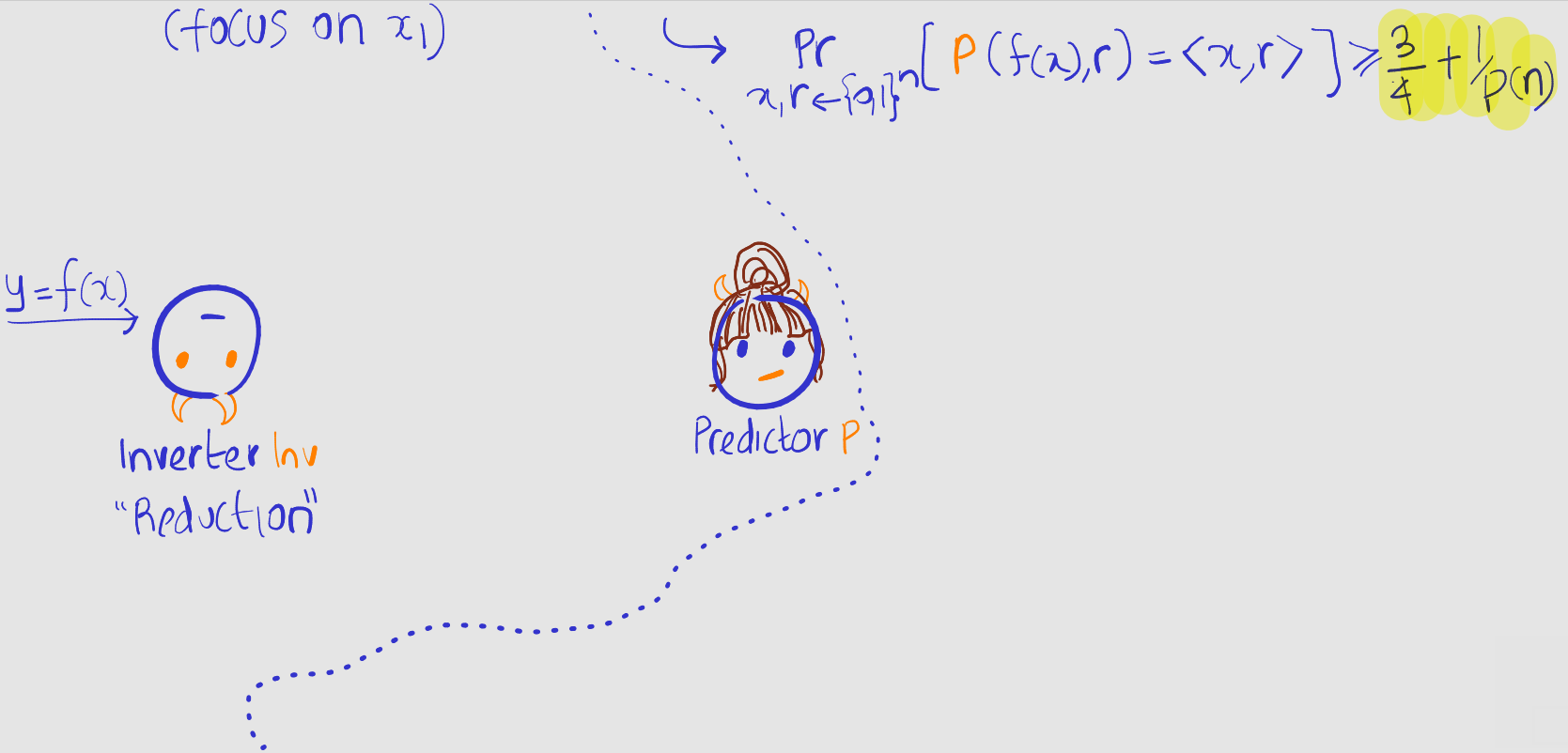
(focus on $x_1$)

# Next Let's Consider "Good" Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter $\mathsf{Inv} \Leftarrow \exists$ Good Predictor $\mathsf{P}$.

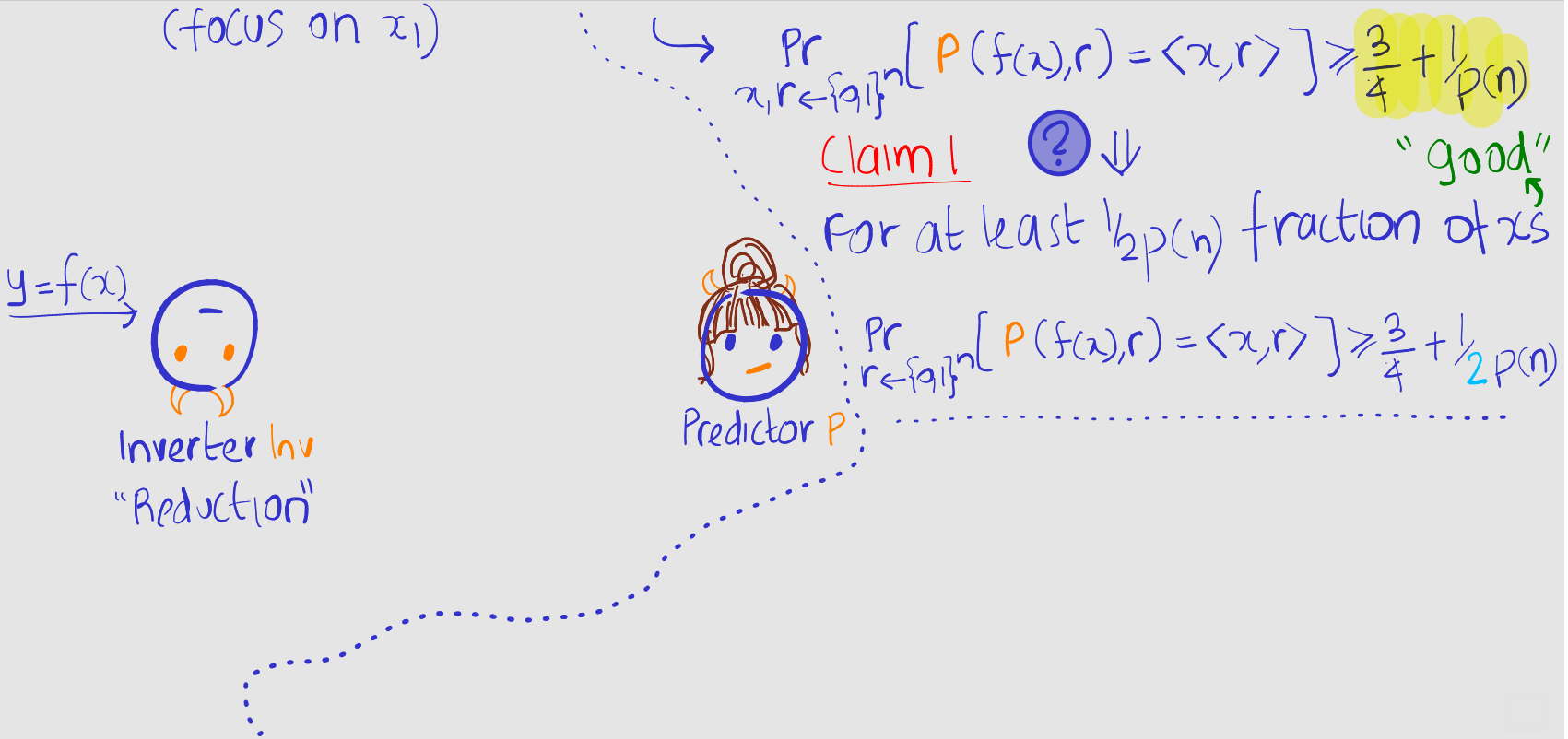


# Next Let's Consider "Good" Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $hc(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. ∃ Inverter Inv ⇐ ∃ Good Predictor P.**

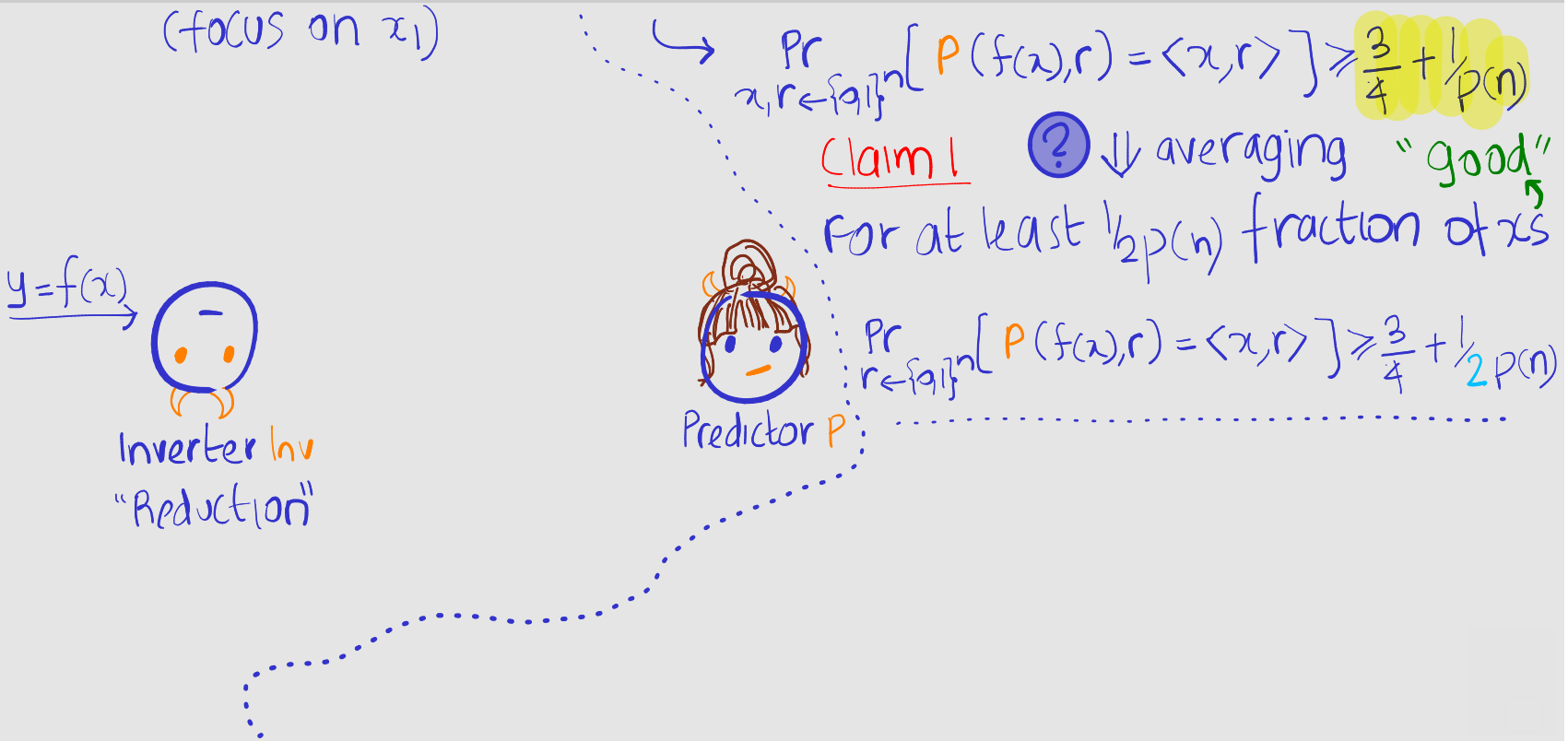


# Next Let's Consider "Good" Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $hc(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. $\exists$ Inverter Inv $\Leftarrow \exists$ Good Predictor P.**

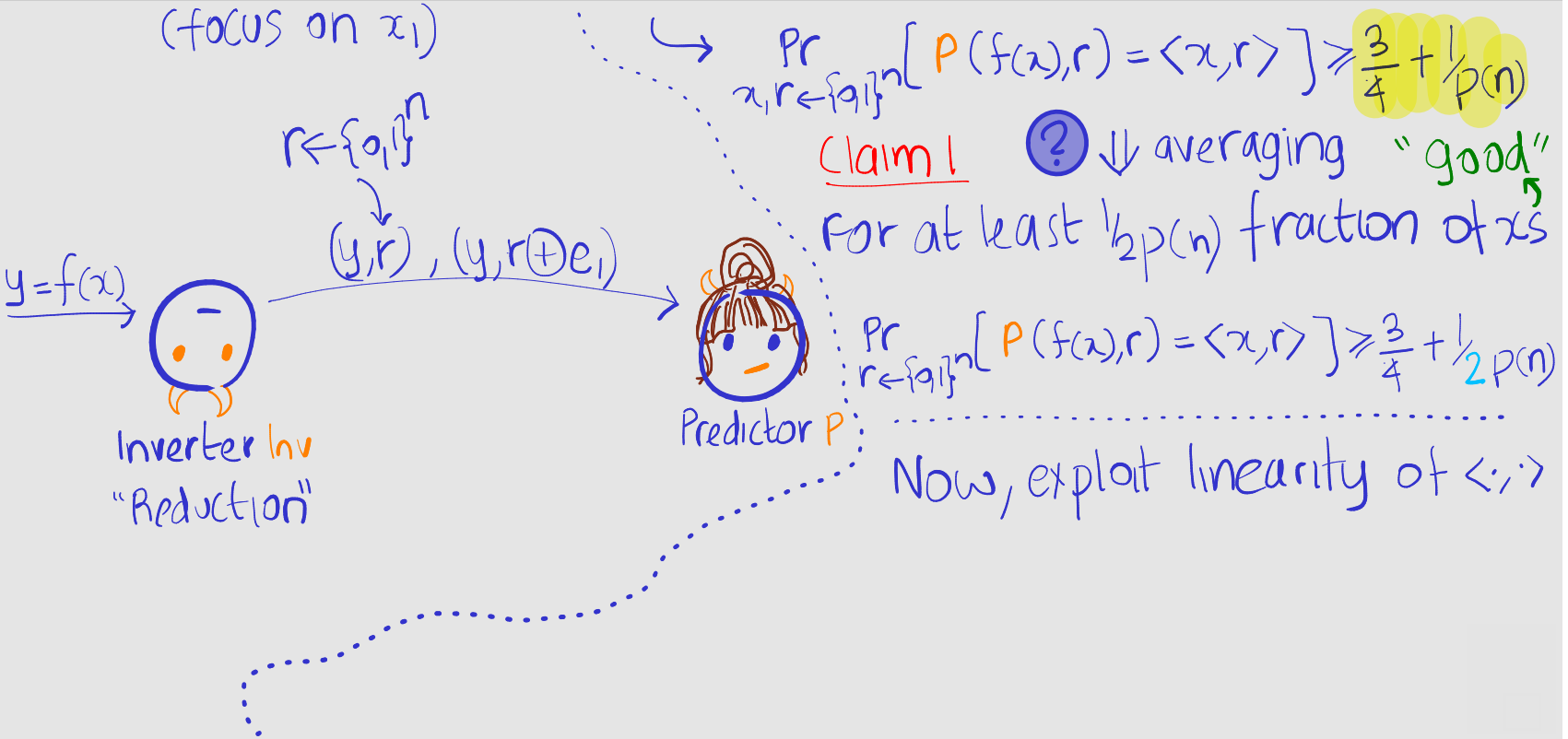


# Next Let's Consider "Good" Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. ∃ Inverter Inv ⇐ ∃ Good Predictor P.**

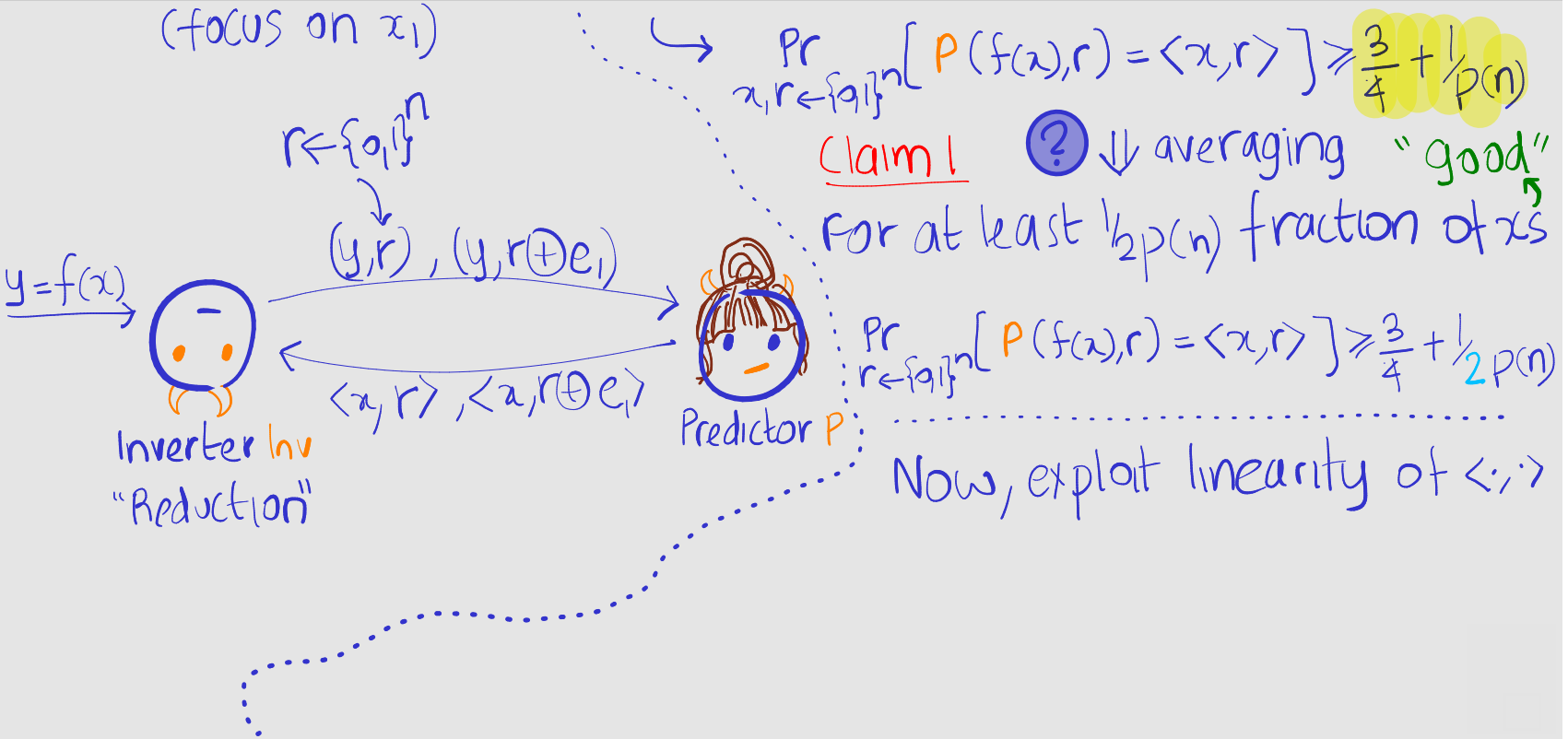


# Next Let's Consider "Good" Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $hc(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. ∃ Inverter Inv ⇐ ∃ Good Predictor P.**

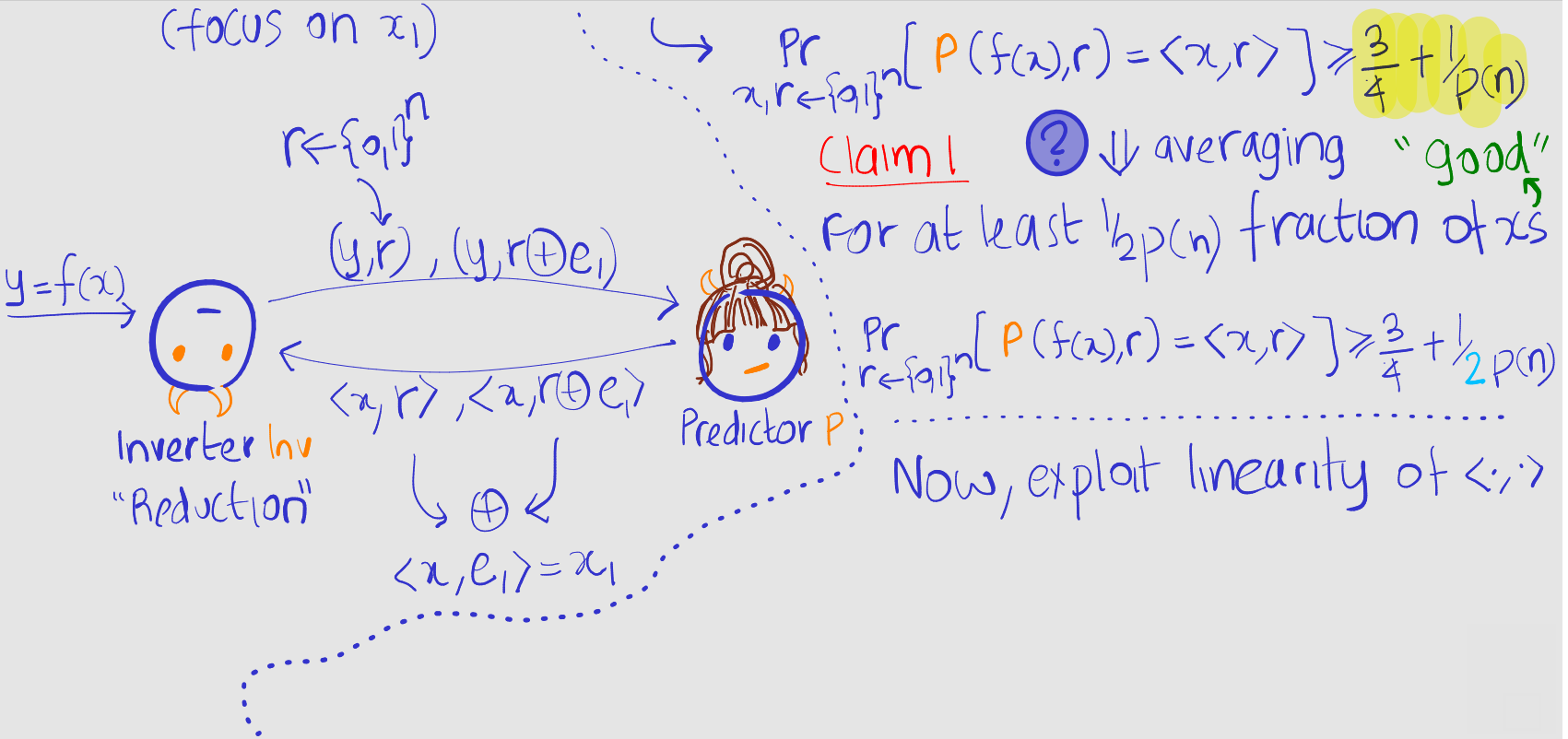


# Next Let's Consider "Good" Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $hc(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. ∃ Inverter Inv ⇐ ∃ Good Predictor P.**

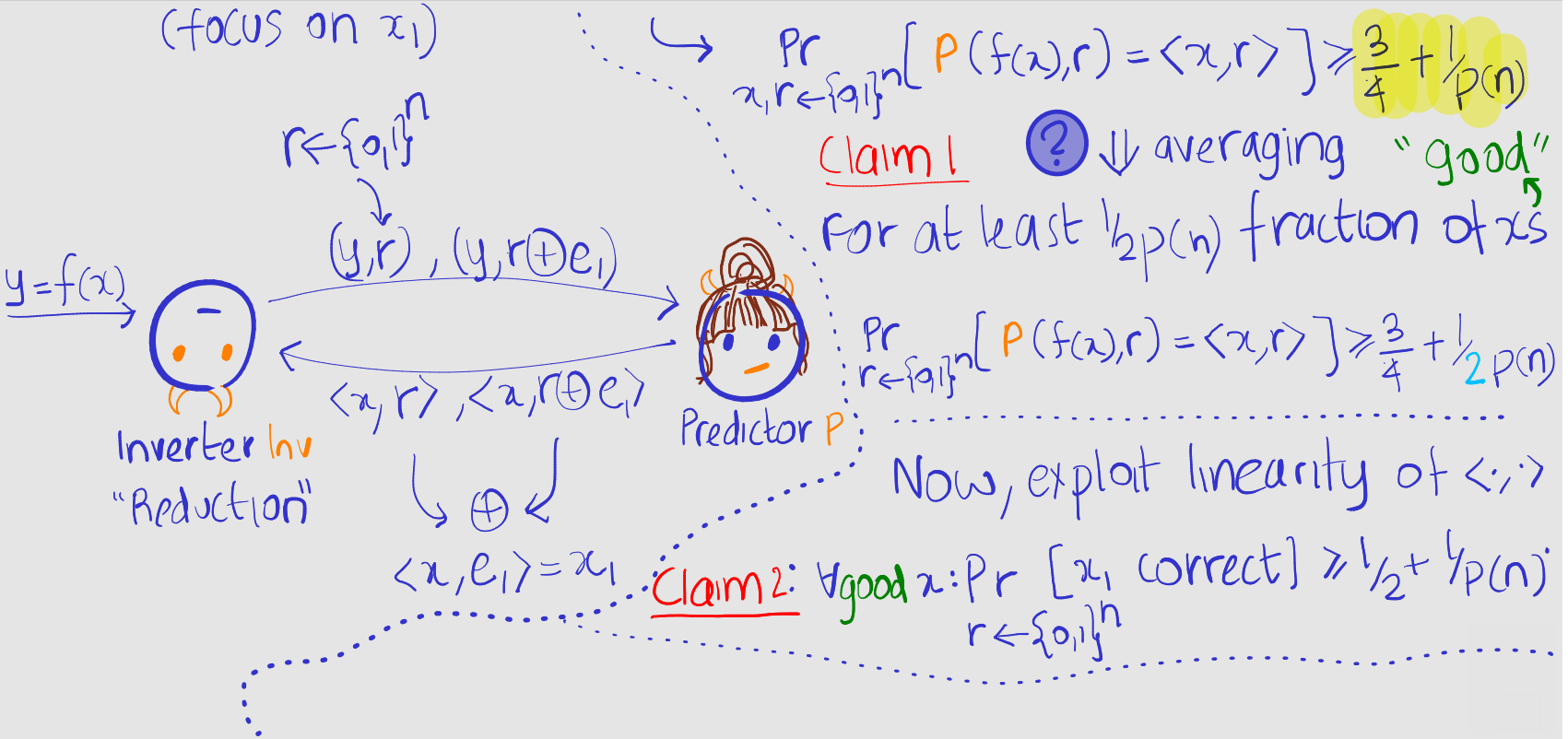


# Next Let's Consider "Good" Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $hc(x,r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. ∃ Inverter Inv ⇐ ∃ Good Predictor P.**

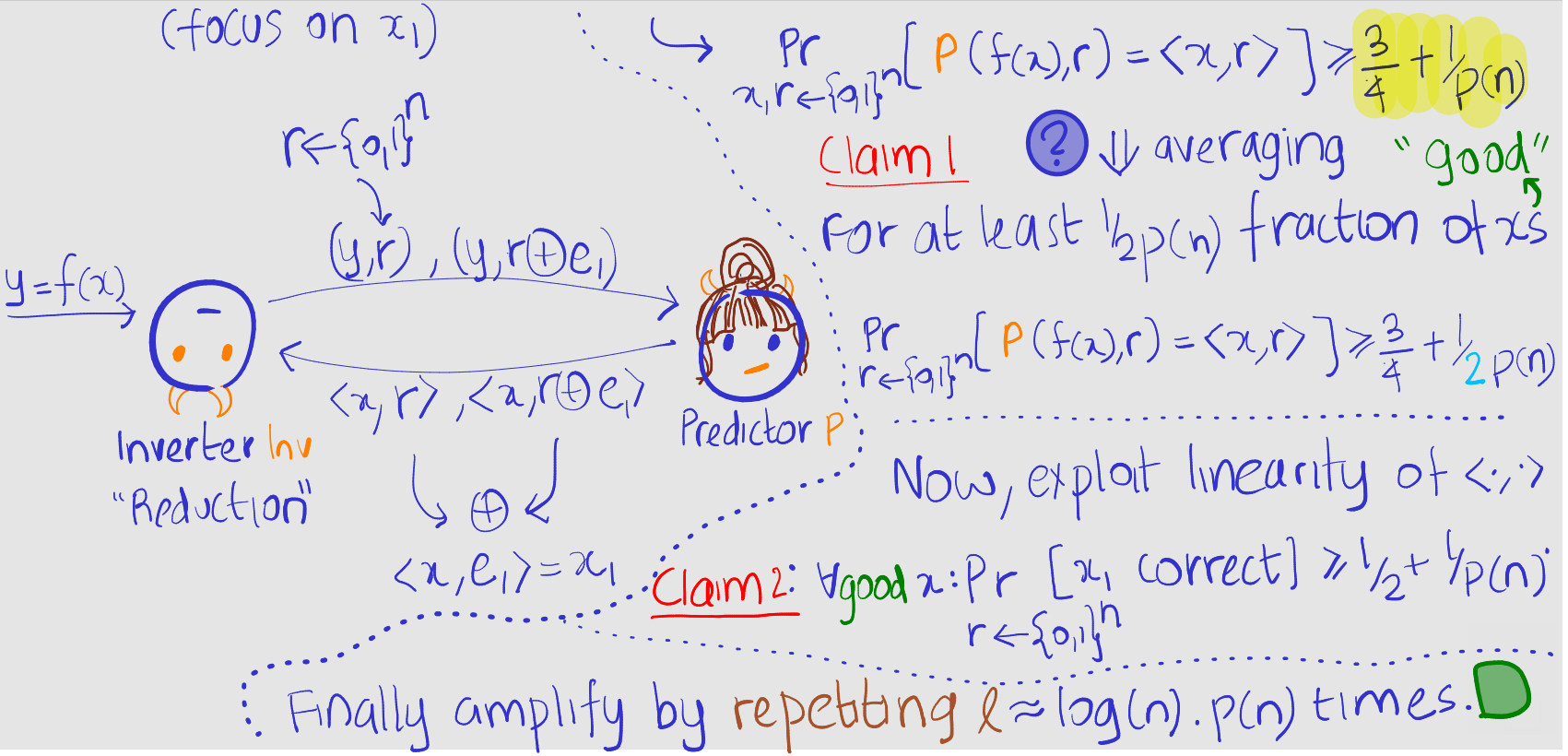


# Next Let's Consider "Good" Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $hc(x,r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. ∃ Inverter Inv ⇐ ∃ Good Predictor P.**

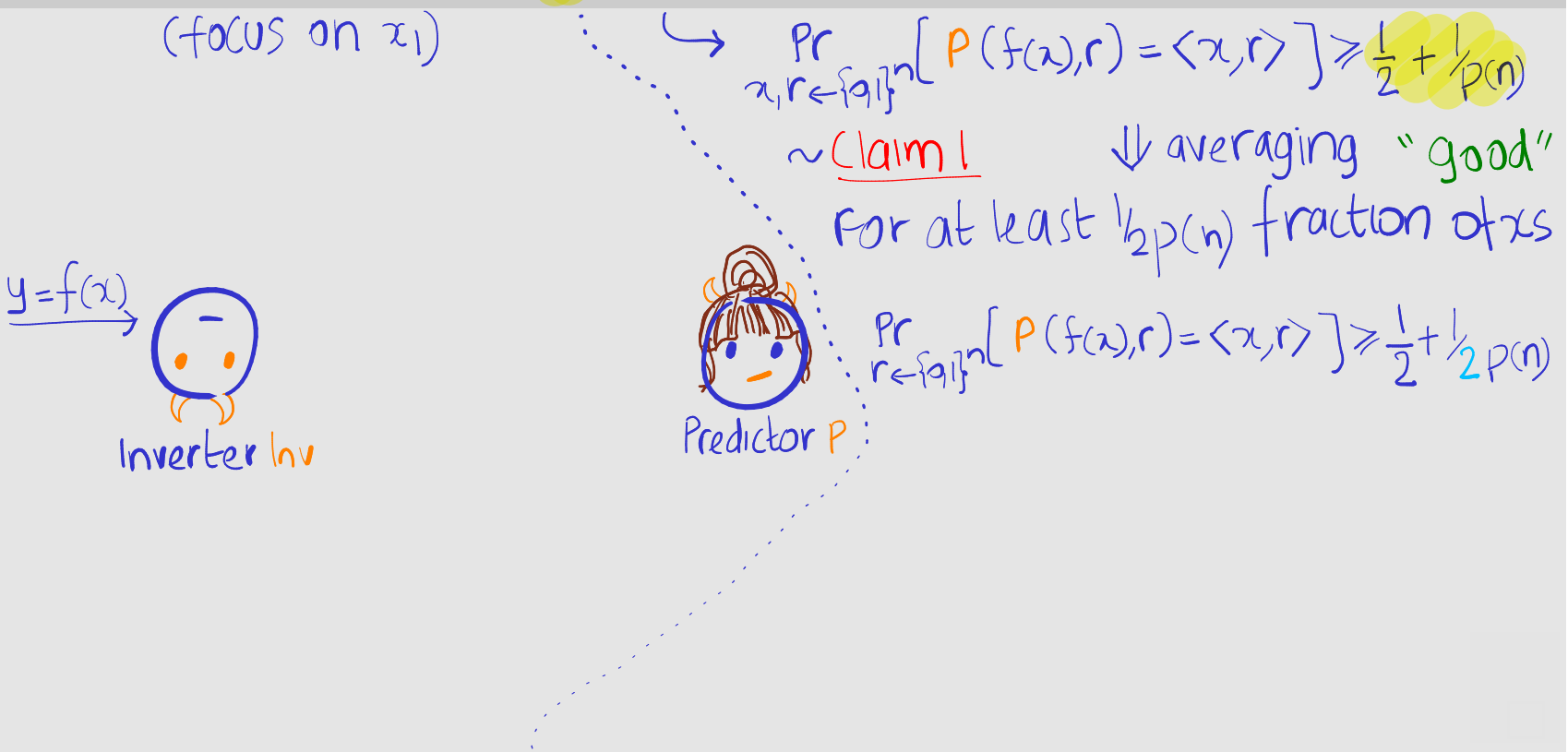


# Finally, We Consider All Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter $\mathsf{Inv} \Leftarrow \exists$ Predictor $\mathsf{P}$.



# Finally, We Consider All Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

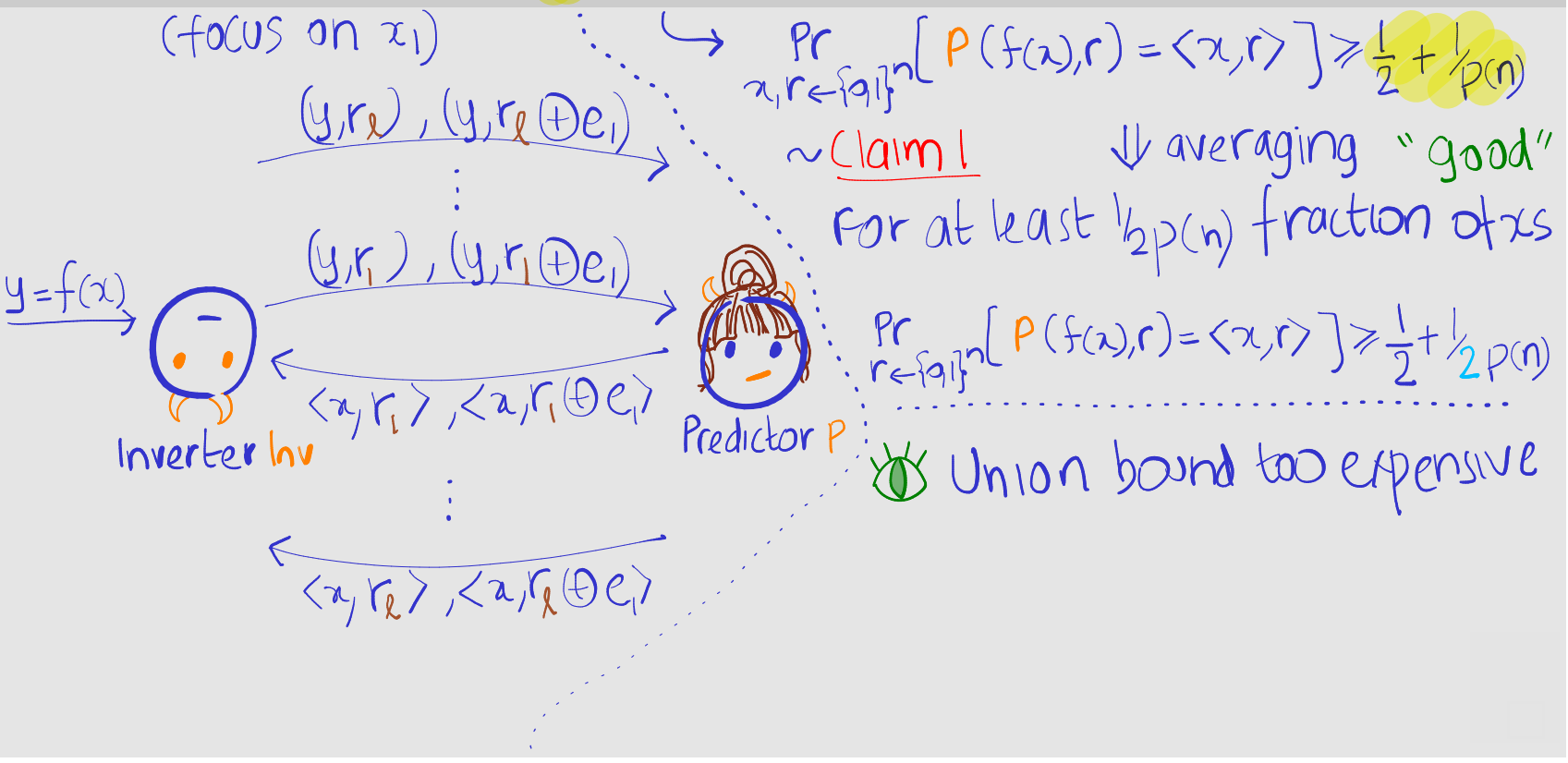
**Proof.** $\exists$ Inverter **Inv** $\Leftarrow \exists$ Predictor **P**.

# Finally, We Consider All Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $\mathbf{hc}(x,r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. $\exists$ Inverter $\mathsf{Inv} \Leftarrow \exists$ Predictor $\mathsf{P}$.**

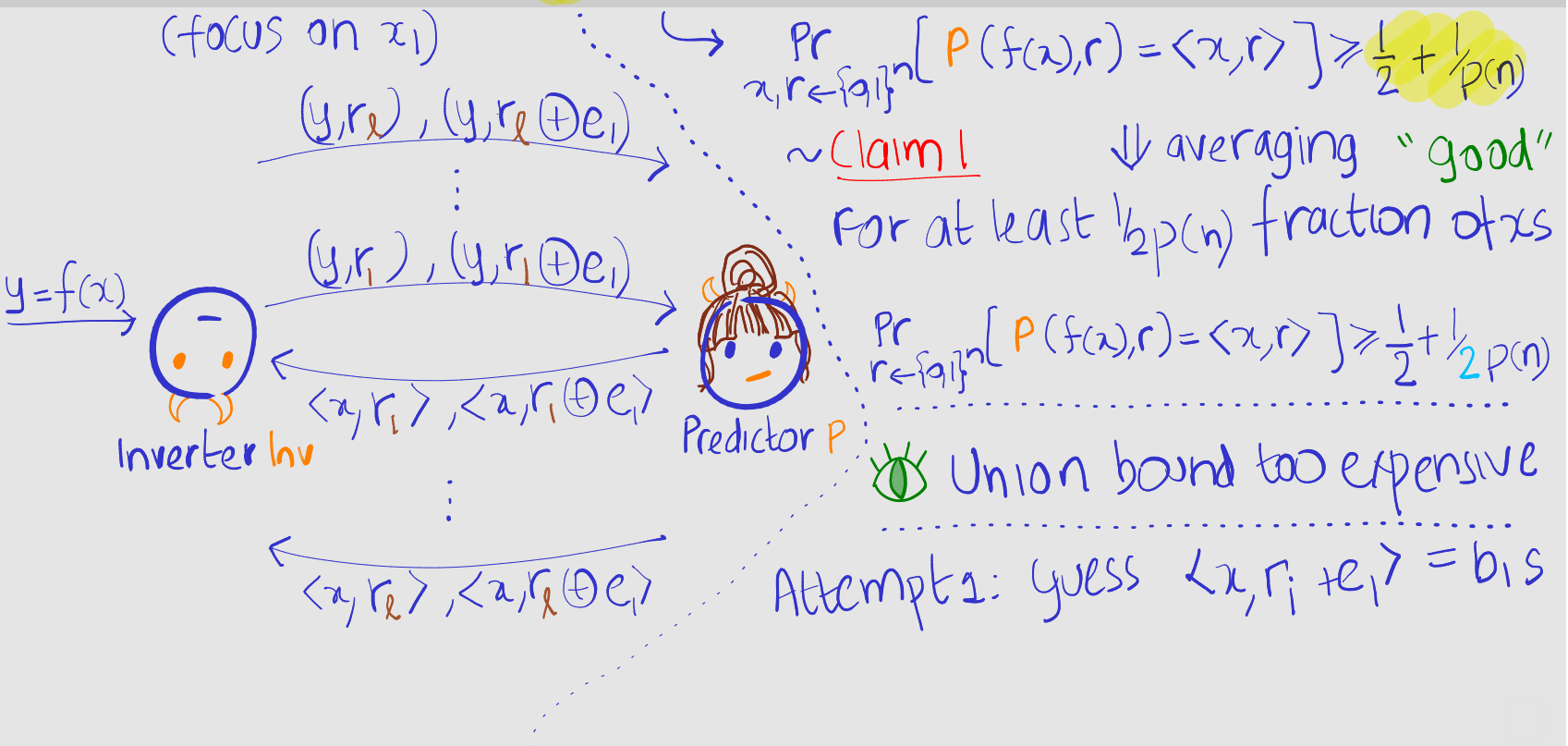


# Finally, We Consider All Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter **Inv** $\Leftarrow \exists$ Predictor **P**.

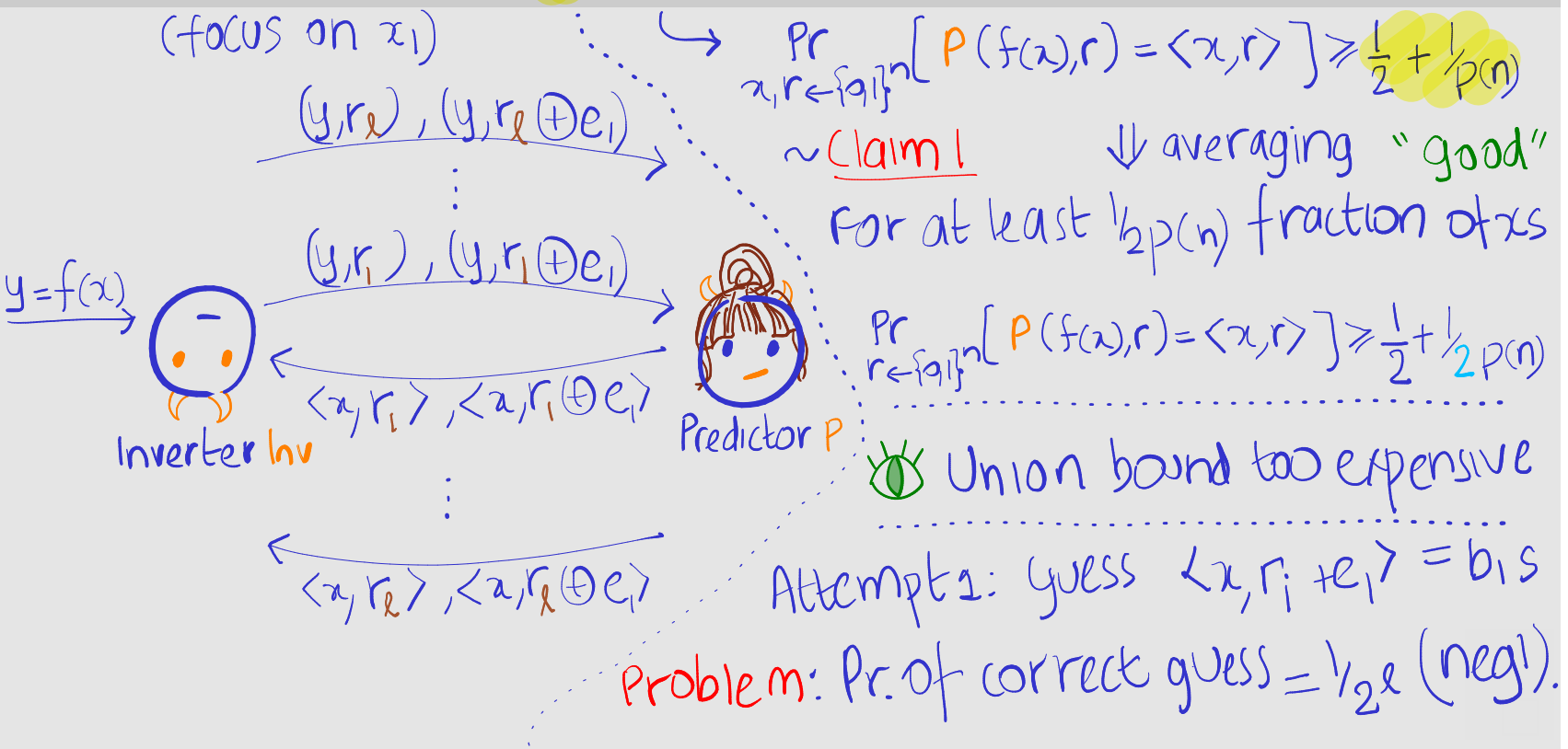


# Finally, We Consider All Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $\mathbf{hc}(x,r) := \langle x, r\rangle_2$ is a hard-core predicate for $f'$.*

**Proof.** $\exists$ Inverter **Inv** $\Leftarrow \exists$ Predictor **P**.

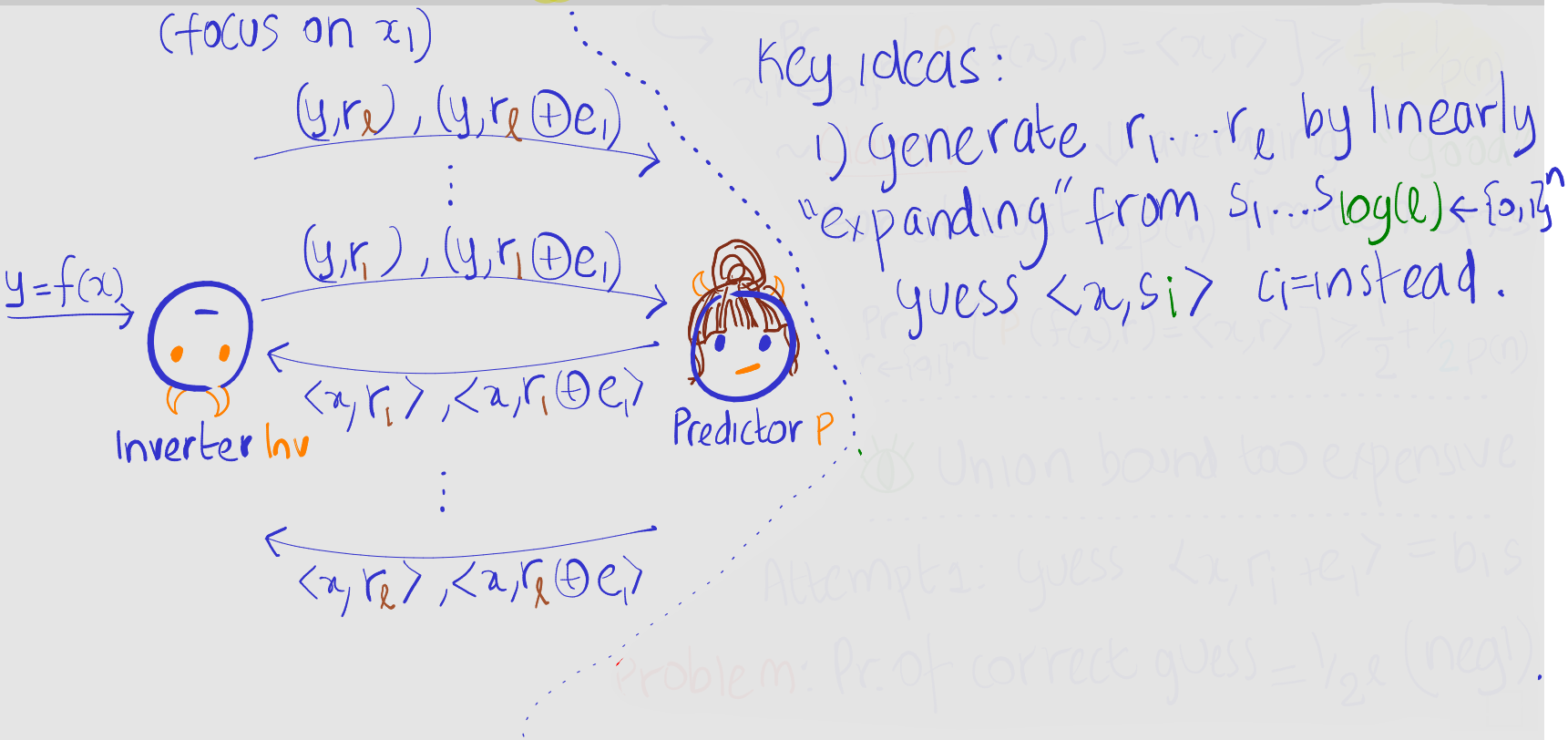


# Finally, We Consider All Predictors

**Theorem 3 (Goldreich-Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. $\exists$ Inverter $\mathsf{Inv} \Leftarrow \exists$ Predictor $\mathsf{P}$.**

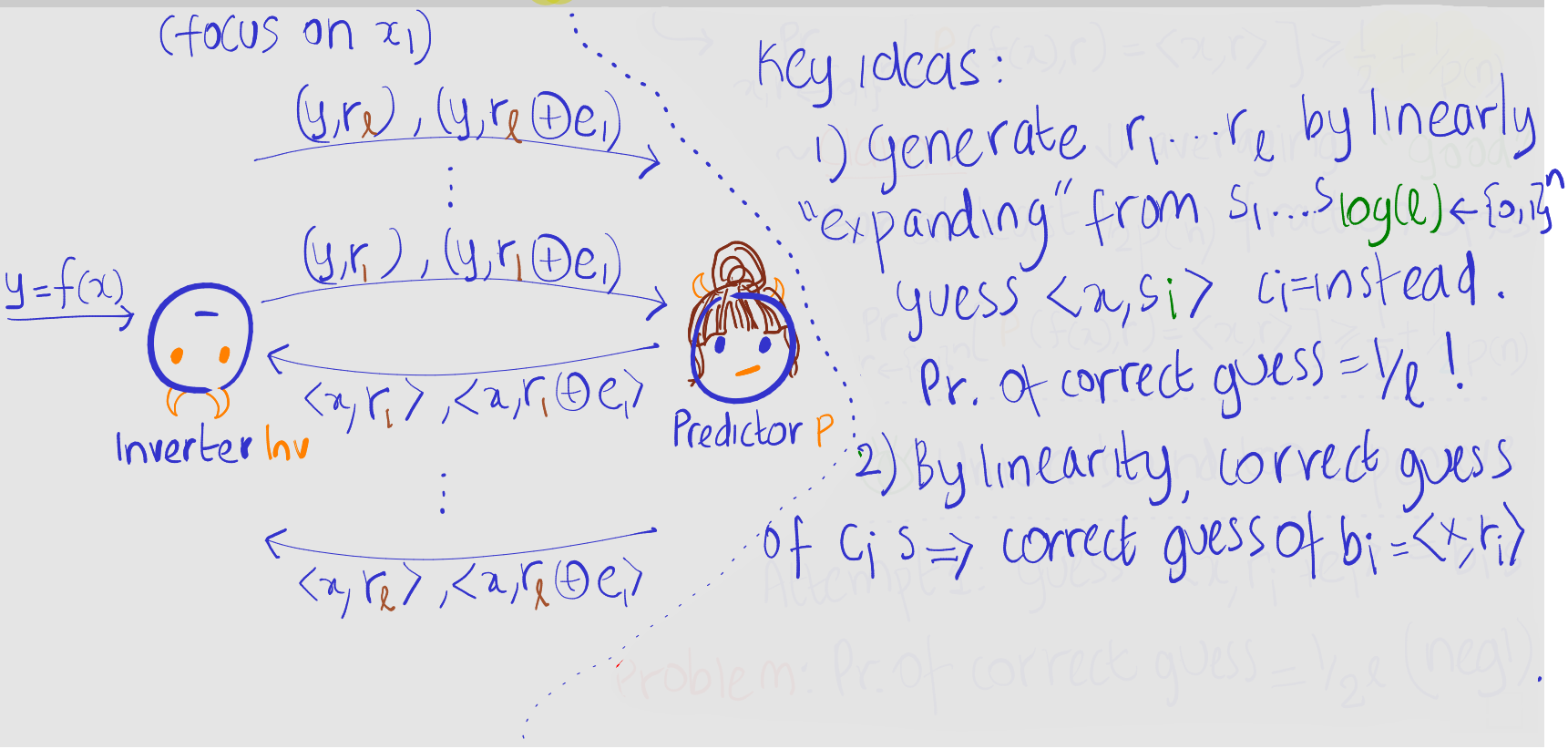


# Finally, We Consider All Predictors

**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x,r) := (f(x), r)$. Then $\mathbf{hc}(x,r) := \langle x, r\rangle_2$ is a hard-core predicate for $f'$.*

**Proof. $\exists$ Inverter $\mathsf{Inv} \Leftarrow \exists$ Predictor $\mathsf{P}$.**

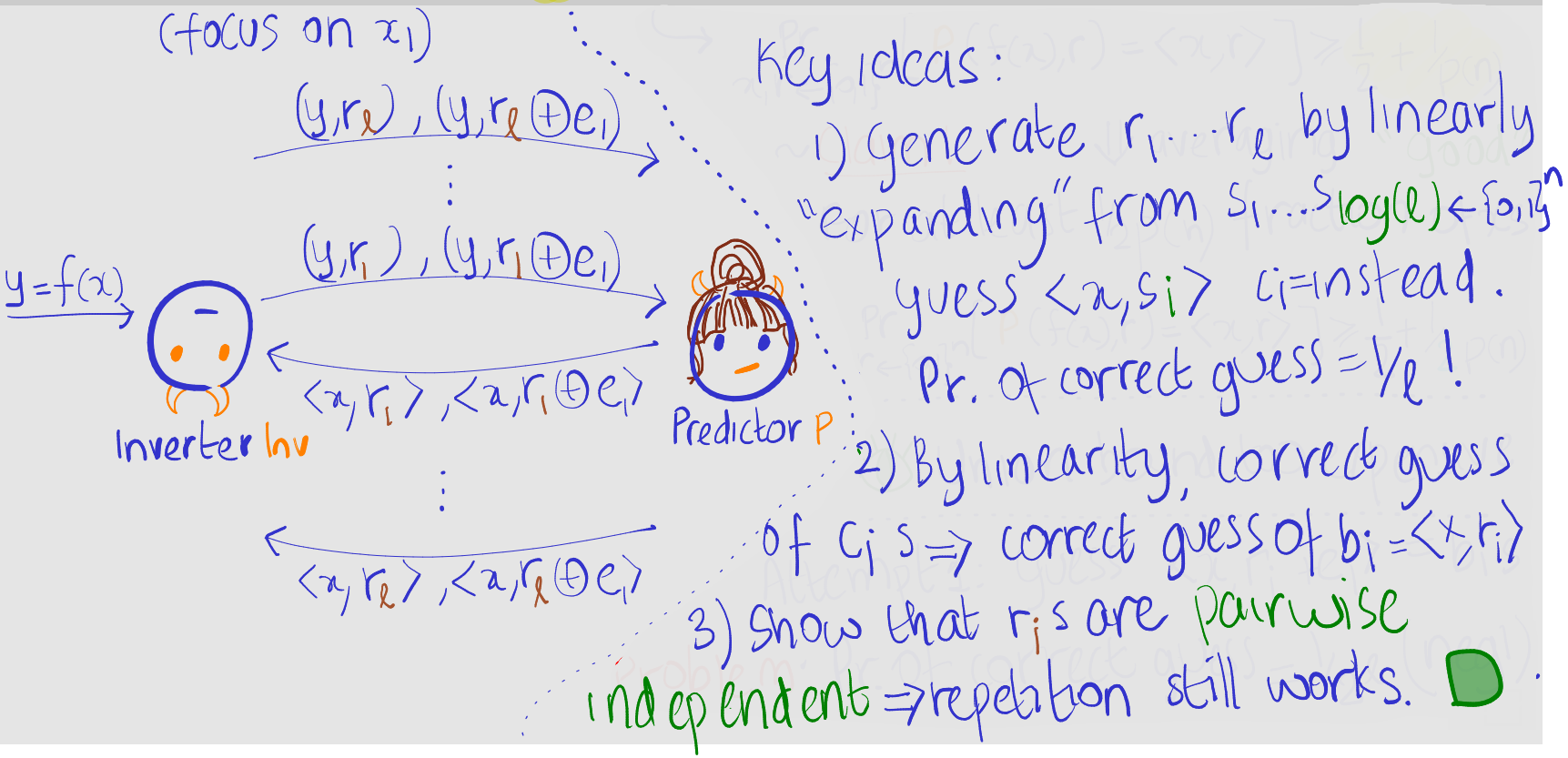


# Finally, We Consider All Predictors

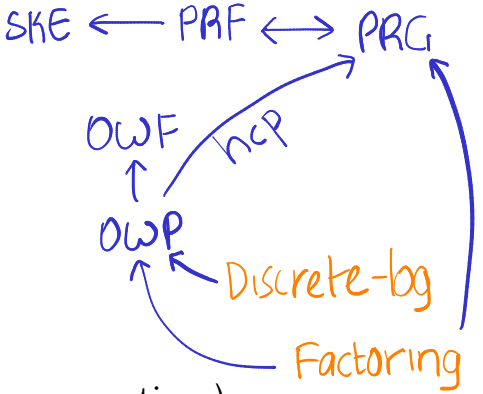
**Theorem 3 (Goldreich–Levin Theorem)**

*For a OWP $f$, let $f'(x, r) := (f(x), r)$. Then $\mathbf{hc}(x, r) := \langle x, r \rangle_2$ is a hard-core predicate for $f'$.*

**Proof. $\exists$ Inverter $\mathsf{Inv} \Leftarrow \exists$ Predictor $\mathsf{P}$.**

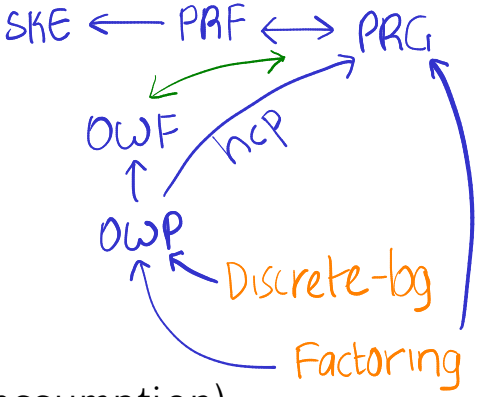


# To Recap



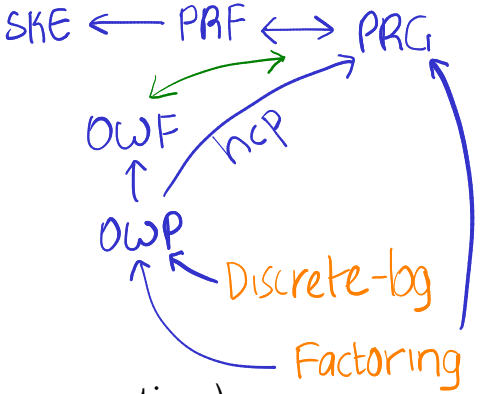

- We saw pseudo-randomness via one-wayness
  - OWP $\rightarrow$ hard-core bit $\rightarrow$ PRG
  - Several examples of OWP:
    1. Exponentiation modulo prime $p$ (discrete-log assumption)
    2. Squaring modulo composite $N = pq$ (factoring assumption)
  - Discrete-log problem/factoring $\rightarrow$ PRG

# To Recap



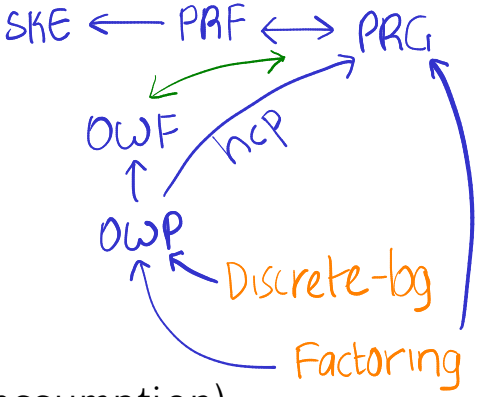

- We saw pseudo-randomness via one-wayness
  - OWP $\rightarrow$ hard-core bit $\rightarrow$ PRG
  - Several examples of OWP:
    1. Exponentiation modulo prime $p$ (discrete-log assumption)
    2. Squaring modulo composite $N = pq$ (factoring assumption)
  - Discrete-log problem/factoring $\rightarrow$ PRG
  - Requirement can be further relaxed: any OWF $\rightarrow$ PRG

# To Recap

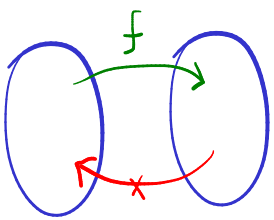


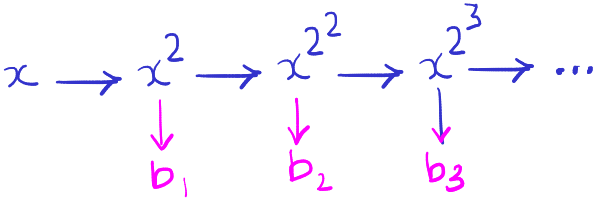

- We saw pseudo-randomness via one-wayness
  - OWP → hard-core bit → PRG
  - Several examples of OWP:
    1. Exponentiation modulo prime $p$ (discrete-log assumption)
    2. Squaring modulo composite $N = pq$ (factoring assumption)
  - Discrete-log problem/factoring → PRG
  - Requirement can be further relaxed: any OWF → PRG
- Worst-case hardness → complexity-theoretic PRG

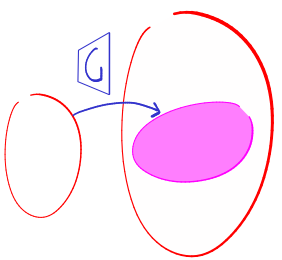
# To Recap




- We saw pseudo-randomness via one-wayness
  - OWP → hard-core bit → PRG
  - Several examples of OWP:
    1. Exponentiation modulo prime $p$ (discrete-log assumption)
    2. Squaring modulo composite $N = pq$ (factoring assumption)
  - Discrete-log problem/factoring → PRG
  - Requirement can be further relaxed: any OWF → PRG
- Worst-case hardness → complexity-theoretic PRG
- Hardness ↔ Unpredictability ↔ Pseudorandomness

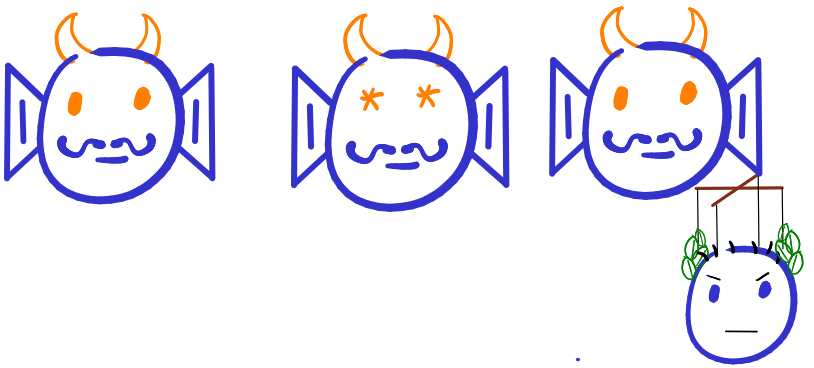


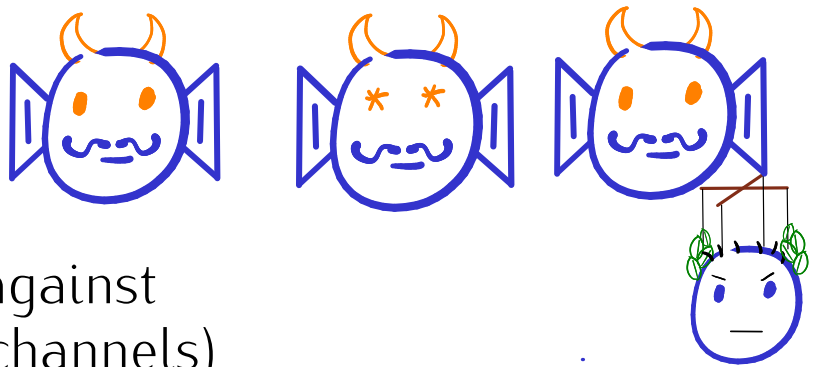
# Next Lecture

- So far: adversaries who don't tamper with ciphertext
  - Eavesdropper of various forms, chosen-plaintext attacker

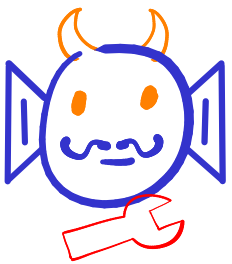


# Next Lecture

- So far: adversaries who don't tamper with ciphertext
  - Eavesdropper of various forms, chosen-plaintext attacker

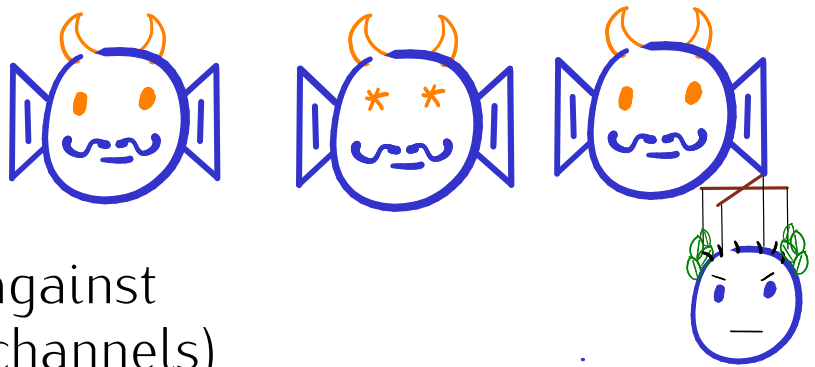


- Coming up: secret communication against *tampering* adversary (a.k.a. secure channels)
  - Authentication and integrity
  - Message authentication codes (MAC)
  - PRF $\rightarrow$ MAC

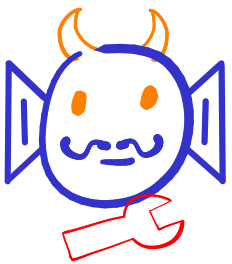


# Next Lecture

- So far: adversaries who don't tamper with ciphertext
  - Eavesdropper of various forms, chosen-plaintext attacker



- Coming up: secret communication against *tampering* adversary (a.k.a. secure channels)
  - Authentication and integrity
  - Message authentication codes (MAC)
  - PRF $\rightarrow$ MAC



More Questions?

# References

1. For a historical take on OWFs, see [DH76].
2. The construction of PRG from OWP via hard-core predicate is from [BM84]. There they rely on the discrete-log based OWP.
3. The squaring based OWP was first studied by Rabin [Rab79].
4. The Goldreich-Levin theorem is from [GL89]. The proof described here closely follows Vinod Vaikuntanathan's proof in Lecture 6 of MIT6875. See [Gol01, §2.5.2] for a formal description of the proof.
5. The construction of PRG from OWF is due to [HILL99]


Manuel Blum and Silvio Micali.
How to generate cryptographically strong sequences of pseudo-random bits.
*SIAM J. Comput.*, 13(4):850–864, 1984.


Whitfield Diffie and Martin E. Hellman.
New directions in cryptography.
*IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.


Oded Goldreich and Leonid A. Levin.
A hard-core predicate for all one-way functions.
In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.


Oded Goldreich.
*The Foundations of Cryptography - Volume 1: Basic Techniques*.
Cambridge University Press, 2001.


Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby.
A pseudorandom generator from any one-way function.
*SIAM J. Comput.*, 28(4):1364–1396, 1999.


M. O. Rabin.
Digitalized signatures and public-key functions as intractable as factorization.
Technical report, USA, 1979.