Hard Core Predicates

601.642/442: Modern Cryptography

Fall 2022

Last Time

- Proof via Reduction: $f_\times$ is a weak OWF
- Amplification: From weak to strong OWFs

Today

- Hard Core Predicate
- 1-bit stretch PRGs from hard core predicate.

What OWFs Hide

- OWFs guarantee that $f(x)$ hides x but nothing more!

What OWFs Hide

- OWFs guarantee that $f(x)$ hides x but nothing more!
  - E.g., it may not hide first bit of x,

What OWFs Hide

- OWFs guarantee that $f(x)$ hides x but nothing more!
  - E.g., it may not hide first bit of x,
  - Or even first half bits of x

What OWFs Hide

- OWFs guarantee that $f(x)$ hides x but nothing more!
  - E.g., it may not hide first bit of x,
  - Or even first half bits of x
- In fact: if $\mathbf{a}(x)$ is any non-trivial information about x, we don't know if $f(x)$ will hide it (except when $\mathbf{a}(x) = x$)

What OWFs Hide

- OWFs guarantee that $f(x)$ hides x but nothing more!
  - E.g., it may not hide first bit of x,
  - Or even first half bits of x
- In fact: if $\mathbf{a}(x)$ is any non-trivial information about x, we don't know if $f(x)$ will hide it (except when $\mathbf{a}(x) = x$)

Is there any non-trivial (non-identity) function of x, even 1 bit, that OWFs hide?

Hard Core Predicate

- A **hard core predicate** for a OWF f

Hard Core Predicate

- A **hard core predicate** for a OWF f
  - is a function over its inputs $\{x\}$

Hard Core Predicate

- A **hard core predicate** for a OWF f
  - is a function over its inputs $\{x\}$
  - its output is a single bit (called "hard core bit")

Hard Core Predicate

- A **hard core predicate** for a OWF f
  - is a function over its inputs $\{x\}$
  - its output is a single bit (called "hard core bit")
  - it can be easily computed given x

Hard Core Predicate

- A **hard core predicate** for a OWF f
  - is a function over its inputs $\{x\}$
  - its output is a single bit (called "hard core bit")
  - it can be easily computed given x
  - but "hard to compute" given only $f(x)$

Hard Core Predicate

- A **hard core predicate** for a OWF f
  - is a function over its inputs $\{x\}$
  - its output is a single bit (called "hard core bit")
  - it can be easily computed given x
  - but "hard to compute" given only $f(x)$
- <u>Intuition</u>: f may leak many bits of x but it does not leak the hard-core bit.
- In other words, learning the hardcore bit of x, even given $f(x)$, is "as hard as" inverting f itself.

Hard Core Predicate

- A **hard core predicate** for a OWF f
  - is a function over its inputs $\{x\}$
  - its output is a single bit (called "hard core bit")
  - it can be easily computed given x
  - but "hard to compute" given only $f(x)$
- Intuition: f may leak many bits of x but it does not leak the hard-core bit.
- In other words, learning the hardcore bit of x, even given $f(x)$, is "as hard as" inverting f itself.
- Think: What does "hard to compute" mean for a single bit?

Hard Core Predicate

- A **hard core predicate** for a OWF f
  - is a function over its inputs $\{x\}$
  - its output is a single bit (called "hard core bit")
  - it can be easily computed given x
  - but "hard to compute" given only $f(x)$
- Intuition: f may leak many bits of x but it does not leak the hard-core bit.
- In other words, learning the hardcore bit of x, even given $f(x)$, is "as hard as" inverting f itself.
- Think: What does "hard to compute" mean for a single bit?
  - you can always guess the bit with probability 1/2.

Hard Core Predicate: Definition

- Hard-core bit cannot be learned or “predicted” or “computed” with probability $> \frac{1}{2} + \nu(|x|)$ even given $f(x)$ (where ν is a negligible function)

Hard Core Predicate: Definition

- Hard-core bit cannot be learned or "predicted" or "computed" with probability $> \frac{1}{2} + \nu(|x|)$ even given $f(x)$ (where ν is a negligible function)

Definition (Hard Core Predicate)

A predicate $h : \{0,1\}^* \to \{0,1\}$ is a hard-core predicate for $f(\cdot)$ if h is efficiently computable given x and there exists a negligible function ν s.t. for every non-uniform PPT adversary $\mathcal{A}$ and $\forall n \in \mathbb{N}$:

$$\Pr\left[x \leftarrow \{0,1\}^n : \mathcal{A}(1^n, f(x)) = h(x)\right] \leqslant \frac{1}{2} + \nu(n).$$

Hard Core Predicate: Definition

- Hard-core bit cannot be learned or “predicted” or “computed” with probability $> \frac{1}{2} + \nu(|x|)$ even given $f(x)$ (where ν is a negligible function)

Definition (Hard Core Predicate)

A predicate $h : \{0,1\}^* \rightarrow \{0,1\}$ is a hard-core predicate for $f(\cdot)$ if h is efficiently computable given x and there exists a negligible function ν s.t. for every non-uniform PPT adversary $\mathcal{A}$ and $\forall n \in \mathbb{N}$:

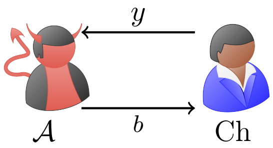
$$\Pr\left[x \leftarrow \{0,1\}^n : \mathcal{A}(1^n, f(x)) = h(x)\right] \leqslant \frac{1}{2} + \nu(n).$$

Hard Core Predicate: Game Based Definition

It is also instructive to think of that definition in this game-based form.

Hard Core Predicate: Game Based Definition

It is also instructive to think of that definition in this game-based form.

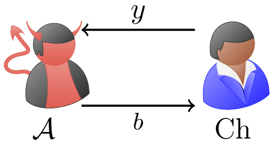


$$x \xleftarrow{\$} \{0,1\}^n$$
$$y = f(x)$$

$$\boxed{\mathcal{A} \text{ wins if } h(x) = b}$$

Hard Core Predicate: Game Based Definition

It is also instructive to think of that definition in this game-based form.



$$x \xleftarrow{\$} \{0,1\}^n$$
$$y = f(x)$$

$$\boxed{\mathcal{A} \text{ wins if } h(x) = b}$$

We want that for all n.u. PPT adversary $\mathcal{A}$, the adversary wins with probability only at most negligible more than 1/2.

$$\Pr[\mathcal{A} \text{ wins}] \leqslant \frac{1}{2} + \nu(n).$$

Hard Core Predicate: Construction

- Can we construct hard-core predicates for general OWFs f?

Hard Core Predicate: Construction

- Can we construct hard-core predicates for general OWFs f?
- Define $\langle x, r \rangle$ to be the **inner product** function mod 2. I.e:,

$$\langle x, r \rangle = \Big(\sum_i x_i r_i\Big) \mod 2$$

Hard Core Predicate: Construction

- Can we construct hard-core predicates for general OWFs f?
- Define $\langle x, r \rangle$ to be the **inner product** function mod 2. I.e:,

$$\langle x, r \rangle = \Big(\sum_i x_i r_i\Big) \mod 2$$

Theorem (Goldreich-Levin)

Let f be a OWF. Define function

$$g(x, r) = (f(x), r)$$

where $|x| = |r|$. Then g is a OWF and

$$h(x, r) = \langle x, r \rangle$$

is a hard-core predicate for f

Proof?

- Proof via Reduction?

Proof?

- Proof via Reduction?
- **Main challenge**: Adversary $\mathcal{A}$ for h only outputs 1 bit. Need to build an inverter $\mathcal{B}$ for f that outputs n bits.

Warmup Proof (1)

- Assumption: Given $g(x, r) = (f(x), r)$, adversary $\mathcal{A}$ **always** (i.e., with probability 1) outputs $h(x, r)$ correctly

Warmup Proof (1)

- Assumption: Given $g(x, r) = (f(x), r)$, adversary $\mathcal{A}$ **always** (i.e., with probability 1) outputs $h(x, r)$ correctly
- Inverter $\mathcal{B}$:

Warmup Proof (1)

- Assumption: Given $g(x,r) = (f(x), r)$, adversary $\mathcal{A}$ **always** (i.e., with probability 1) outputs $h(x,r)$ correctly
- Inverter $\mathcal{B}$:
  - Compute $x_i^* \leftarrow \mathcal{A}(f(x), e_i)$ for every $i \in [n]$ where:

$$e_i = (\underbrace{0,\ldots,0}_{(i-1)\text{-times}}, 1, \ldots, 0)$$

Warmup Proof (1)

- Assumption: Given $g(x, r) = (f(x), r)$, adversary $\mathcal{A}$ **always** (i.e., with probability 1) outputs $h(x, r)$ correctly
- Inverter $\mathcal{B}$:
  - Compute $x_i^* \leftarrow \mathcal{A}(f(x), e_i)$ for every $i \in [n]$ where:

$$e_i = (\underbrace{0, \ldots, 0}_{(i-1)\text{-times}}, 1, \ldots, 0)$$

  - Output $x^* = x_1^* \ldots x_n^*$

Warmup Proof (2)

- Assumption: Given $g(x, r) = (f(x), r)$, adversary $\mathcal{A}$ outputs $h(x, r)$ with probability $3/4 + \varepsilon(n)$ (over choices of (x, r))

Warmup Proof (2)

- Assumption: Given $g(x, r) = (f(x), r)$, adversary $\mathcal{A}$ outputs $h(x, r)$ with probability $3/4 + \varepsilon(n)$ (over choices of (x, r))
- **Main Problem:** Adversary may not work on "improper" inputs (e.g., $r = e_i$ as in previous case)

Warmup Proof (2)

- Assumption: Given $g(x, r) = (f(x), r)$, adversary $\mathcal{A}$ outputs $h(x, r)$ with probability $3/4 + \varepsilon(n)$ (over choices of (x, r))
- **Main Problem:** Adversary may not work on "improper" inputs (e.g., $r = e_i$ as in previous case)
- **Main Idea:** Split each query into two queries s.t. each query individually looks random

Warmup Proof (2)

- Assumption: Given $g(x,r) = (f(x), r)$, adversary $\mathcal{A}$ outputs $h(x,r)$ with probability $3/4 + \varepsilon(n)$ (over choices of (x,r))
- **Main Problem:** Adversary may not work on "improper" inputs (e.g., $r = e_i$ as in previous case)
- **Main Idea:** Split each query into two queries s.t. each query individually looks random
- Inverter $\mathcal{B}$:

Warmup Proof (2)

- Assumption: Given $g(x, r) = (f(x), r)$, adversary $\mathcal{A}$ outputs $h(x, r)$ with probability $3/4 + \varepsilon(n)$ (over choices of (x, r))
- **Main Problem:** Adversary may not work on "improper" inputs (e.g., $r = e_i$ as in previous case)
- **Main Idea:** Split each query into two queries s.t. each query individually looks random
- Inverter $\mathcal{B}$:
  - Let $a := \mathcal{A}(f(x), e_i + r)$ and $b := \mathcal{A}(f(x), r)$, for $r \xleftarrow{\$} \{0, 1\}^n$

Warmup Proof (2)

- Assumption: Given $g(x,r) = (f(x), r)$, adversary $\mathcal{A}$ outputs $h(x,r)$ with probability $3/4 + \varepsilon(n)$ (over choices of (x,r))
- **Main Problem:** Adversary may not work on "improper" inputs (e.g., $r = e_i$ as in previous case)
- **Main Idea:** Split each query into two queries s.t. each query individually looks random
- Inverter $\mathcal{B}$:
  - Let $a := \mathcal{A}(f(x), e_i + r)$ and $b := \mathcal{A}(f(x), r)$, for $r \xleftarrow{\$} \{0,1\}^n$
  - Compute $c := a \oplus b$

Warmup Proof (2)

- Assumption: Given $g(x,r) = (f(x), r)$, adversary $\mathcal{A}$ outputs $h(x,r)$ with probability $3/4 + \varepsilon(n)$ (over choices of (x,r))
- **Main Problem:** Adversary may not work on "improper" inputs (e.g., $r = e_i$ as in previous case)
- **Main Idea:** Split each query into two queries s.t. each query individually looks random
- Inverter $\mathcal{B}$:
  - Let $a := \mathcal{A}(f(x), e_i + r)$ and $b := \mathcal{A}(f(x), r)$, for $r \xleftarrow{\$} \{0,1\}^n$
  - Compute $c := a \oplus b$
  - $c = x_i$ with probability at least $\frac{1}{2} + \varepsilon$ (Union Bound)

Warmup Proof (2)

- Assumption: Given $g(x,r) = (f(x), r)$, adversary $\mathcal{A}$ outputs $h(x,r)$ with probability $3/4 + \varepsilon(n)$ (over choices of (x,r))
- **Main Problem:** Adversary may not work on "improper" inputs (e.g., $r = e_i$ as in previous case)
- **Main Idea:** Split each query into two queries s.t. each query individually looks random
- Inverter $\mathcal{B}$:
  - Let $a := \mathcal{A}(f(x), e_i + r)$ and $b := \mathcal{A}(f(x), r)$, for $r \xleftarrow{\$} \{0,1\}^n$
  - Compute $c := a \oplus b$
  - $c = x_i$ with probability at least $\frac{1}{2} + \varepsilon$ (Union Bound)
  - Repeat and take majority to obtain x_i^* s.t. $x_i^* = x_i$ with prob. $1 - \mathsf{negl}(n)(n)$

Warmup Proof (2)

- Assumption: Given $g(x, r) = (f(x), r)$, adversary $\mathcal{A}$ outputs $h(x, r)$ with probability $3/4 + \varepsilon(n)$ (over choices of (x, r))
- **Main Problem:** Adversary may not work on "improper" inputs (e.g., $r = e_i$ as in previous case)
- **Main Idea:** Split each query into two queries s.t. each query individually looks random
- Inverter $\mathcal{B}$:
  - Let $a := \mathcal{A}(f(x), e_i + r)$ and $b := \mathcal{A}(f(x), r)$, for $r \xleftarrow{\$} \{0, 1\}^n$
  - Compute $c := a \oplus b$
  - $c = x_i$ with probability at least $\frac{1}{2} + \varepsilon$ (Union Bound)
  - Repeat and take majority to obtain x_i^* s.t. $x_i^* = x_i$ with prob. $1 - \mathsf{negl}(n)(n)$
  - Output $x^* = x_1^* \dots x_n^*$

Full Proof

Try on your own

Full Proof

Try on your own (or read from lecture notes)

Full Proof

Try on your own (or read from lecture notes)

- Goldreich-Levin Theorem extremely influential even outside cryptography

Full Proof

Try on your own (or read from lecture notes)

- Goldreich-Levin Theorem extremely influential even outside cryptography
- Applications to learning, list-decoding codes, extractors,...

Full Proof

Try on your own (or read from lecture notes)

- Goldreich-Levin Theorem extremely influential even outside cryptography
- Applications to learning, list-decoding codes, extractors,...
- Extremely useful tool to add to your toolkit

One-Way Functions: Remarks

- One-way functions are necessary for most of cryptography

One-Way Functions: Remarks

- One-way functions are necessary for most of cryptography
- But often not sufficient. Black-box separations known [Impagliazzo-Rudich'89]; full separations not known

One-Way Functions: Remarks

- One-way functions are necessary for most of cryptography
- But often not sufficient. Black-box separations known [Impagliazzo-Rudich'89]; full separations not known
- Additional Reading: Universal One-way Functions

One-Way Functions: Remarks

- One-way functions are necessary for most of cryptography
- But often not sufficient. Black-box separations known [Impagliazzo-Rudich'89]; full separations not known
- Additional Reading: Universal One-way Functions
  - Suppose somebody tells you that OWFs exist! E.g., they might discover a proof for it!

One-Way Functions: Remarks

- One-way functions are necessary for most of cryptography
- But often not sufficient. Black-box separations known [Impagliazzo-Rudich'89]; full separations not known
- Additional Reading: Universal One-way Functions
  - Suppose somebody tells you that OWFs exist! E.g., they might discover a proof for it!
  - But they don't tell you what this function is. E.g., even they might not know the function! They just have a proof of its existence...

One-Way Functions: Remarks

- One-way functions are necessary for most of cryptography
- But often not sufficient. Black-box separations known [Impagliazzo-Rudich'89]; full separations not known
- Additional Reading: Universal One-way Functions
  - Suppose somebody tells you that OWFs exist! E.g., they might discover a proof for it!
  - But they don't tell you what this function is. E.g., even they might not know the function! They just have a proof of its existence...
  - Can you use this fact to build an **explicit** OWF?

One-Way Functions: Remarks

- One-way functions are necessary for most of cryptography
- But often not sufficient. Black-box separations known [Impagliazzo-Rudich'89]; full separations not known
- Additional Reading: Universal One-way Functions
  - Suppose somebody tells you that OWFs exist! E.g., they might discover a proof for it!
  - But they don't tell you what this function is. E.g., even they might not know the function! They just have a proof of its existence...
  - Can you use this fact to build an **explicit** OWF?
  - Yes! Levin gives us a method!