Hard Core Predicate

601.642/442: Modern Cryptography

Fall 2019
Proof via Reduction: $f_\times$ is a weak OWF

Amplification: From weak to strong OWFs
Today - Part I

- What do OWFs Hide?
- Hard Core Predicate
- Concluding Remarks on OWFs
What OWFs Hide

- The concept of OWFs is simple and concise
- But OWFs often not very useful by themselves
- It only guarantees 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 $a(x)$ is any non-trivial information about $x$, we don’t know if $f(x)$ will hide it (except when $a(x) = x$)

Is there any non-trivial (non-identity) function of $x$, even 1 bit, that OWFs hide?
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 \( \nu \) 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 \( \nu \) s.t. for every non-uniform PPT adversary \( A \) and \( \forall n \in \mathbb{N} \):

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

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

**Theorem (Goldreich-Levin)**

Let $f$ be a OWF (OWP). Define function

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

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

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

is a hard-core predicate for $f$
Proof via Reduction?

**Main challenge**: Adversary $A$ for $h$ only outputs 1 bit. Need to build an inverter $B$ for $f$ that outputs $n$ bits.
Assumption: Given $g(x, r) = (f(x), r)$, adversary $A$ always (i.e., with probability 1) outputs $h(x, r)$ correctly.

Inverter $B$:
- Compute $x_i^* \leftarrow A(f(x), e_i)$ for every $i \in [n]$ where:
  \[
e_i = (0, \ldots, 0, 1, \ldots, 0)^{(i-1)}-\text{times}
  \]
- Output $x^* = x_1^* \ldots x_n^*$
Assumption: Given $g(x, r) = (f(x), r)$, adversary $A$ 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 $B$:
- Let $a := A(f(x), e_i + r)$ and $b := A(f(x), r)$, for $r \leftarrow \{0, 1\}^n$
- Compute $c := a \oplus b$
- $c = x_i$ with probability $\frac{1}{2} + \varepsilon$ (Union Bound)
- Repeat and take majority to obtain $x_i^*$ s.t. $x_i^* = x_i$ with prob. $1 - \text{negl}(n)$
- Output $x^* = x_1^* \ldots x_n^*$
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 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!