One-Way Functions and Hard-Core Predicates (单向函数与硬核谓词)

Sheng Zhong   Yuan Zhang

Computer Science and Technology Department
Nanjing University

Outline

What are one-way functions (OWFs)?

- A **one-way function (单向函数)** is a function that satisfies:
  1. easy to compute.
  2. hard to invert.
- The existence of OWF implies the existence of many other useful concepts, including *PRG*, *PRF*, *CCA-secure private-key encryption scheme*, *MAC*, etc.

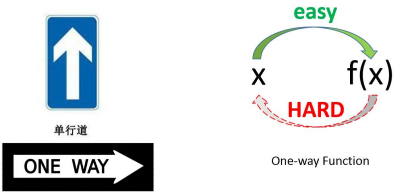

图 1: "One way" and "One-way function"

Defining the one-way function

DEFINITION 7.1

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is **one-way** if the following two conditions hold:

1. Easy to compute: There exists a PPT algorithm M_f computing f; that is, $M_f(x) = f(x)$ for all x.
2. Hard to invert: For every PPT algorithm $\mathcal{A}$, there is a negligible function *negl* such that

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n} \left[\mathcal{A}(1^n, f(x)) \in f^{-1}(f(x))\right] \leq negl(n).$$

About the "hardness"

Hard to invert: For every PPT algorithm $\mathcal{A}$, there is a negligible function *negl* such that

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n} [\mathcal{A}(1^n, f(x)) \in f^{-1}(f(x))] \leq negl(n).$$

- ("Hard-on-the-average") We consider uniformly random input x, and ask the adversary to invert $f(x)$ (rather than choosing an arbitrary y from the range of f and asking to compute $f^{-1}(y)$).
- ("One preimage is enough") We measure the probability that the adversary outputs one preimage of $f(x)$ only. (We don't require it outputs x, or all preimages.)

The experimental definition

We can encapsulate all details into the following experiment to define OWF.

The inverting experiment $Invert_{\mathcal{A},f}(n)$

1. Choose uniform $x \in \{0,1\}^n$, and compute $y := f(x)$.
2. $\mathcal{A}$ is given y as input, and outputs x'.
3. The output of the experiment is defined to be 1 if $f(x') = y$, and 0 otherwise.

Now, the Hard to invert property requires for every PPT adversary $\mathcal{A}$, there is a negligible function *negl* such that

$$Pr[Invert_{\mathcal{A},f}(n) = 1] \leq negl(n).$$

The existence of OWF

Regarding the existence of OWF, we have

- It is NOT known weather OWF exist or not, although we have several candidates that we have not found PPT solver for them so far.
- We assume or hypothesize these candidates are OWFs unless we find evidences to show they are not.

Candidate 1: the integer factorization

Q1: Factorize 16.
A1: $16 = 2 * 2 * 2 * 2$

Q2: Factorize 121.
A2: 121 $=$ 11*11

Q3: Factorize 221.
A3: 221 $=$13*17

Q4: Let x and y be two prime integers of equal bit-length n. Let

$$f_{mult}(x, y) = x \cdot y.$$

Factorize $f_{mult}(x, y)$.

Candidate 2: the subset-sum problem

Let $x_1, \ldots, x_n$ be n integers of equal bit-length n. Let $I \subseteq \{1, 2, \ldots, n\}$ be an index set. Let

$$f_{ss}(x_1, \ldots, x_n, I) = \sum_{i \in I} x_i.$$

To invert f_{ss} requires to output the index set I given $f_{ss}(x_1, \ldots, x_n, I)$.

- It is known that the subset-sum problem is NP-complete in the worst case.
- For f_{ss} to be one-way, we need it to be hard on the average.
- We conjecture this f_{ss} in the above form is a OWF due to the fact that no known PPT algorithms solve the random "high-density"[1] instance of the subset-sum problem.

[1] high density requires the length of integers approx. equals their total number.

Candidate 3: the discrete-logarithm problem

Let *Gen* be a PPT-algorithm that, on input 1^n, outputs an n-bit prime p along with a random element $g \in \{2, 3, \ldots, p-1\}$. Define

$$f_{p,g}(x) = g^x \mod p.$$

1. One-Way Functions and One-Way Function Families

2. OWF candidates

3. Hard-core predicates
   - Hard-core predicates

4. Constructing PRGs with One-Way Function

OWF could reveal information about its input

Q: $f(x)$ is OWF $\Rightarrow$ No adversary can compute x from $f(x)$ (except for a negligible probability) $\overset{?}{\Rightarrow}$ "nothing about x can be determined from $f(x)$".

A: Not necessarily.

Q: A counter-example?
A: Assume $g(x)$ is an OWF with input length $n/2$. Construct $f(x_1||x_2) = x_1||g(x_2)$. It is easy to see $f(x)$ is an OWF. But $f(x)$ leaks the first half of its input.

Hard-core predicate

We use hard-core predicate (硬核谓词) to model the information that the function $f(x)$ hides:

DEFINITION 7.4

A function $hc : \{0,1\}^* \to \{0,1\}$ is a **hard-core predicate of a function** f if hc can be computed in polynomial time, and for every PPT adversaries $\mathcal{A}$ there is a negligible function $negl$ such that

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n} [\mathcal{A}(1^n, f(x)) = hc(x)] \le \frac{1}{2} + negl(n),$$

where the probability is taken over the uniform choice of x in $\{0,1\}^n$ and the randomness of $\mathcal{A}$.

- hc(x) can be efficiently computed given x.
- The above definition does NOT require f(x) to be one-way.

A simple constructing idea that does not work

Let $x_1, \ldots, x_n$ denote x's bits. Define $hc(x) = \oplus_{i=1}^{n} x_i$.

Q: Is $hc(x)$ always a hard-core predicate given $f(x)$ is one-way?
A: No.

Q: Counter-example?
A: Let $g(x)$ be one-way. Let $f(x) = (g(x), \oplus_{i=1}^{n} x_i)$. It is easy to see $f(x)$ is one-way.

A trivial hard-core predicate

Let $x_1, \ldots, x_n$ denote x's bits. Define $f(x)$ be the function that drops the last bit, i.e.

$$f(x) = x_1 \cdots x_{n-1}.$$

Let $hc(x) = x_n$.

- $hc(x)$ is a hard-core predicate, and $hc(x)$ is NOT leaked from $f(x)$.
- Clearly, f is not one-way.

A hard-core predicate for any OWF

THEOREM 7.5 Goldreich-Levin Theorem

Assume one-way function (resp. permutation) exists. Then there exists a one-way function (resp. permutation) g and a hard-core predicate hc of g.

- Specifically, given OWF f, g and hc can be constructed as follows:

$$g(x, r) = (f(x), r) \text{ for } |x| = |r|,$$

and

$$hc(x, r) = \oplus_{i=1}^{n} x_i \cdot r_i.$$

- It essentially states if f is a OWF, then $f(x)$ hides the the exclusive-or of a random subset of the bits of x.

Constructing PRG with minimal expansion

THEOREM 7.19

Let f be a one-way permutation and let hc be a hard-core predicate of f. Then $G(s) = f(s)||hc(s)$ is a PRG with expansion factor $l(n) = n + 1$.

- $f(s)$ is uniformly random given s is random and f is a permutation.
- $hc(s)$ "looks" random to any PPT adversaries even when they can see $f(s)$.

Increasing the expansion factor

THEOREM 7.20

If there exists a pseudo-random generator G with expansion factor $n+1$, then for any polynomial *poly* there exists a pseudo-random generator $\hat{G}$ with expansion factor $poly(n)$.

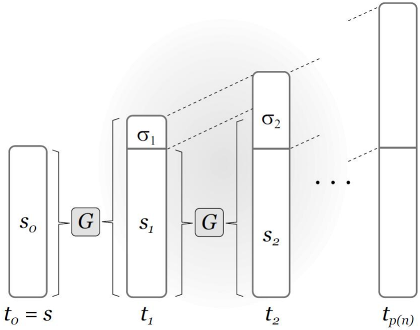

FIGURE 7.1: Increasing the expansion of a pseudorandom generator.