Lecture 19: Candidate One-way Functions

Recall: OWF

Intuition: OWF

A function $f\colon \{0,1\}^n \to \{0,1\}^n$ is a one-way function if

1. The function f is easy to evaluate, and
2. The function f is difficult is hard to invert

- We believe certain functions are one-way functions
- If P $=$ NP then one-way functions cannot exist (see appendix). So, proving that a particular function f is a one-way function will demonstrate that P $\neq$ NP, which we believe is a very difficult problem to resolve
- So, based on our current knowledge in mathematics, we have invested faith in believing that a few specially designed functions are one-way functions

First Candidate: Discrete Log is Hard

- Let $(G, \times)$ be a group and g be a generator. That is, $G = \{g^0, g^1, g^2, \dots, g^{K-1}\}$
- Let $f\colon \{0, \dots, K-1\} \to G$ be defined as follows

$$f(x) = g^x$$

- Think: Why is this function efficient to evaluate?
- It is believed that there exists group G where f is hard to invert
- Clarification: We are not saying that f is hard to invert in any group G. There are special groups G where f is believed to be hard to invert
- Note that the inversion problem asks you to find the "logarithm," given y find x such that $g^x = y$. This is known as the discrete logarithm problem

Second Candidate: Finding Square-root is Hard

- Let p and q be n-bit prime numbers
- Let $N = pq$
- Rabin's function is defined as follows

$$f(x) = x^2 \mod N$$

- Think: Why is this function efficient to evaluate?
- It is believed that finding square-roots mod N is hard when N is the product of two large primes
- Think: How can you invert Rabin's function if you know the factorization of N. That is, given p and q, how can you efficiently compute x' such that $(x')^2 \mod N = y$, where $y = x^2 \mod N$

Third Candidate: Factorization is Hard

- Let $\mathcal{P}_n$ be the set of prime numbers that require n-bit for their binary representation (i.e., the primes in the range $\{2^{n-1}, \ldots, 2^n - 1\}$). For example, $\mathcal{P}_4 = \{11, 13\}$
- Consider the function $f : \mathcal{P}_n \times \mathcal{P}_n \to \mathbb{N}$

$$f(x, y) = xy$$

- Think: Why is this function efficient to compute?
- Assuming that the factorization of product of large prime numbers is difficult, this function is hard to invert

Fourth Candidate: Elliptic Curves

- Elliptic curves are sets of pairs of elements x, y in a field that satisfy the equation $y = x^3 + ax + b$, for some suitably chosen values of a, b
- There is a definition of "point addition" over an elliptic curve, i.e., given two points P and Q on the curve, we can suitably define a point $P + Q$ on the curve
- Given a point P on the elliptic curve, we can add $\overbrace{P + P + \cdots + P}^{x\text{-times}}$ and represent the resulting point as xP
- Then the following function is believed to be one-way for suitable elliptic curves

$$f(x, P) = (P, xP)$$

- Think: Can you connect this assumption to the discrete log problem?

One-way Permutations

Definition

A function $f\colon \{0,1\}^n \to \{0,1\}^n$ is a one-way permutation if it is a one-way function and the function f is a bijection

We introduce this primitive because the construction of pseudorandom generators from one-way permutations is significantly more intuitive than the construction of pseudorandom generators from OWF

Appendix: Efficient Inversion of Efficiently Computable Functions I

We shall show the following result

Theorem

Let $f: \{0,1\}^n \rightarrow \{0,1\}^n$ *be a function that can be efficiently computed. If* $\mathrm{P} = \mathrm{NP}$ *then there exists an efficient algorithm to find an inverse* x' *of* y*, where* $y = f(x)$ *for some* $x \in \{0,1\}^n$

Appendix: Efficient Inversion of Efficiently Computable Functions II

Before we begin the proof of the theorem, let me emphasize that there is always an inefficient algorithm to find x', an inverse of y

Invert-Ineffcient (y):

1. For $x' \in \{0,1\}^n$: If $f(x') == y$, then return x'
2. Return -1

This is an inefficient algorithm to compute an inverse of $y = f(x)$

Appendix: Efficient Inversion of Efficiently Computable Functions III

Let us prove the theorem now. First, let us introduce a few notations.

- Recall $f: \{0,1\}^n \to \{0,1\}^n$ is the function
- Let $\varphi(x)$ be a 3-SAT formula that tests whether $f(x) = y$ or not. That is, $\varphi(x)$ evaluates to true if and only if $f(x) = y$.
- If f can be evaluated in polynomial time, then the size of $\varphi(x)$ is polynomial in n
- If $\mathrm{P} = \mathrm{NP}$ then we can efficiently determine: Is $\varphi(x)$ satisfiable or not

Appendix: Efficient Inversion of Efficiently Computable Functions IV

Let us introduce the notion of a partial assignment of variables $\{x_1, x_2, \ldots, x_n\}$

- Consider the following example.

$$\varphi(x) = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

- The formula "$\varphi(x)$ under the restriction $x_i \mapsto b$," is obtained by substituting b as the value of x_i in the formula $\varphi(x)$ and simplifying. For example, "$\varphi(x)$ under the restriction $x_1 \mapsto 0$" is the following formula

$$\begin{aligned}\varphi(x)|_{x_1 \mapsto 0} &= (0 \vee x_2 \vee \neg x_3) \wedge (\neg 0 \vee x_2 \vee x_3) \\ &= (0 \vee x_2 \vee \neg x_3) \wedge (1 \vee x_2 \vee x_3) \\ &= (x_2 \vee \neg x_3)\end{aligned}$$

Appendix: Efficient Inversion of Efficiently Computable Functions V

- Given a set of partial assignments $\text{assign} = \{x_{i_1} \mapsto b_1, x_{i_2} \mapsto b_2, \dots, x_{i_k} \mapsto b_k\}$, we define $\varphi(x)|_{\text{assign}}$ by setting the values of $x_{i_1}, \dots, x_{i_k}$ as $b_1, \dots, b_k$ in $\varphi(x)$ and simplifying
- Again, if $\text{P} = \text{NP}$ and f is efficiently computable, then it is efficient to find whether $\varphi(x)|_{\text{assign}}$ is satisfiable or not

Appendix: Efficient Inversion of Efficiently Computable Functions VI

Now consider the following algorithm. We will construct a solution $x_1 x_2 \dots x_n$ such that $f(x_1 x_2 \dots x_n) = y$ one bit at a time.

Find_Inverse(y):

1. Let $\varphi(x)$ be the 3-SAT formula mentioned above
2. If $\varphi(x)$ is not satisfiable, then return -1
3. assign $= \emptyset$
4. For $i = 1$ to n:
    1. result = Test whether "$\varphi(x)|_{\text{assign} \cup \{x_i \mapsto 0\}}$" is satisfiable or not
    2. If result == true: assign = assign $\cup \{x_i \mapsto 0\}$
    3. Else: assign = assign $\cup \{x_i \mapsto 1\}$
5. Return assign

Note that this is an efficient algorithm to compute an inverse of y if f can be computed efficiently and $\mathrm{P} = \mathrm{NP}$

Appendix: Defining Addition on Elliptic Curves

- Consider the field $(\mathbb{R}, +, \times)$
- Let us consider the plot of the curve $y^2 = x^3 + ax + b$ (in this example, we have $a = -2$ and $b = 4$)
- Given two points P and Q on the curve, draw the line through them and find R', the third intersection point of the line with the curve
- Reflect R' on the X-axis to obtain the point R
- We define the point R as the sum $P + Q$