Lecture 14: One-way Functions

Objective

- In this lecture we shall introduce the notion of one-way functions,
- We shall consider the construction of one-way functions based on the assumption that factorization of the product of two large primes is difficult,
- Furthermore, we shall also consider one-way function construction based on other assumptions like hardness of discrete logarithm, finding square root, and elliptic curve cryptography,
- Finally, we shall also learn about hardness amplification of "weak" one-way functions into one-way functions.

Intuition:One-way Functions

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 $\mathrm{P} = \mathrm{NP}$ then one-way functions cannot exist (see appendix). So, proving that a particular function f is a one-way function will demonstrate that $\mathrm{P} \neq \mathrm{NP}$, 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

- Suppose $f\colon D \to R$ be a function, where D is the domain and R is the range of the function f
- We consider the game between honest challenger and an adversary
  - The honest challenger samples $x \xleftarrow{\$} D$ and computer $z = f(x)$. The honest challenger presents the challenge $z \in R$ to the adversary.
  - The adversary on input $z \in R$, outputs $x' \in D$
  - The adversary wins the game if $f(x') = z$
- Note that we do not insist on finding $x' = x$. The adversary wins if she finds *any* pre-image of z, that is, any $x' \in D$ such that $f(x') = z$ The pre-image chosen by the adversary need not be exactly the same as the pre-image chosen by the honest challenger.

Hardness of Inverting a Function II

- Note that if an adversary has unbounded computational power, she can evaluate f for every entry in D and check which one of them gives z. So, it is necessary to restrict the adversary to being *computationally bounded* or an *efficient algorithm*.
- The probability that an adversary $\mathcal{A}$ wins in the game against the honest challenger $\mathcal{H}$ is succinctly expressed below

$$\mathbb{P}\left[f(x') = z \colon x \xleftarrow{\$} D, z = f(x), x' = \mathcal{A}(z)\right] \leqslant \text{small}$$

- The function f is said to be one-way if for any computationally bounded adversary $\mathcal{A}$ the probability above is "small"
- In this lecture we shall see how we can use "hardness of factorization" to construct a one-way function

- Think: Why did we not insist on $x' = x$ for successful inversion of the function f?

- Let $\mathcal{P}_n$ represent the set of prime numbers in the set $[2^{n-1}, 2^n - 1]$. That is, $\mathcal{P}_n$ consists of prime numbers that need exactly n bits in their binary representation.
- Let us consider the function $f : \mathcal{P}_n \times \mathcal{P}_n \to \{0,1\}^{2n}$ defined by $f(x, y) = x \cdot y$, where $\cdot$ represents the integer multiplication of the prime numbers $x, y \in \mathcal{P}_n$
- The *hardness of factorization* assumption says that f is a one-way function
- Remarks.
  - Given z, an adversary can win by either presenting (p, q) or (q, p) such that $p \cdot q = z$.
  - Note that we are not saying whether it is easy or difficult to factorize other composite numbers. We are only saying that "factorizing numbers that are product of two large prime numbers" is difficult

Hardness of Factorization II

- In this course, we shall represent this by saying that, for all computationally bounded adversary $\mathcal{A}$, we have

$$\mathbb{P}\left[p' \cdot q' = z \colon (p, q) \xleftarrow{\$} \mathcal{P}_n \times \mathcal{P}_n, z = p \cdot q, (p', q') = \mathcal{A}(z)\right] \approx 0$$

  In a more advanced course, we shall say that the probability of successfully inverting is *negligible* in n, and define "negligible functions" formally. For this course, we shall use ≈ 0 notation.

Another One-way Function

- Suppose we define $g\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ by $g(x,y) = x \cdot y$.
- Think: Why isn't g a one-way function?
- However, we shall prove that g is a *weak* form of one-way function. Suppose $(x,y) \in \mathcal{P}_n \times \mathcal{P}_n$, then we know that g is hard to invert. The probability that $(x,y) \in \mathcal{P}_n \times \mathcal{P}_n$ is presented below

$$\begin{aligned}\mathbb{P}\left[(x,y) \in \mathcal{P}_n \times \mathcal{P}_n\right] &= \frac{|\mathcal{P}_n|}{2^n} \times \frac{|\mathcal{P}_n|}{2^n} \\ &\approx \left(\frac{2^{n-1}/n}{2^n}\right)^2 \\ &= \frac{1}{4n^2}\end{aligned}$$

- So, basically we conclude the following
  - In $1-\frac{1}{4n^2}$ fraction of the inputs, we do not have any assurance about how hard it is to invert the function g
  - In $\frac{1}{4n^2}$ fraction of the inputs, the function g is hard to invert (based on hardness of factorization).
- We want to claim the following

Claim

$$\mathbb{P}\left[\mathcal{A} \textit{ factorizes } z\right] \lesssim 1-\frac{1}{4n^2}$$

- We proceed as follows

$$\begin{aligned}\mathbb{P}\left[\mathcal{A} \text{ factorizes } z\right] &= \mathbb{P}\left[\mathcal{A} \text{ factorizes } z, (x,y) \notin \mathcal{P}_n \times \mathcal{P}_n\right] \\ &\quad + \mathbb{P}\left[\mathcal{A} \text{ factorizes } z, (x,y) \in \mathcal{P}_n \times \mathcal{P}_n\right]\end{aligned}$$

- We bound each of these two terms separately

$$\begin{aligned}&\mathbb{P}\left[\mathcal{A} \text{ factorizes } z, (x,y) \notin \mathcal{P}_n \times \mathcal{P}_n\right]\\ =&\mathbb{P}\left[\mathcal{A} \text{ factorizes } z | (x,y) \notin \mathcal{P}_n \times \mathcal{P}_n\right] \mathbb{P}\left[(x,y) \notin \mathcal{P}_n \times \mathcal{P}_n\right]\\ \leqslant& 1 \cdot \left(1 - \frac{1}{4n^2}\right) = 1 - \frac{1}{4n^2}\end{aligned}$$

- The other term is

$$\begin{aligned}&\mathbb{P}\left[\mathcal{A} \text{ factorizes } z, (x,y) \in \mathcal{P}_n \times \mathcal{P}_n\right]\\ =&\mathbb{P}\left[\mathcal{A} \text{ factorizes } z | (x,y) \in \mathcal{P}_n \times \mathcal{P}_n\right] \mathbb{P}\left[(x,y) \in \mathcal{P}_n \times \mathcal{P}_n\right]\\ \lesssim& 0 \cdot \frac{1}{4n^2} = 0\end{aligned}$$

- So, overall, we get

$$\mathbb{P}\left[\mathcal{A} \text{ factorizes } z\right] \lesssim \left(1 - \frac{1}{4n^2}\right) + 0 = 1 - \frac{1}{4n^2}$$

- Intuitive Conclusion: If there exists a "dense" set of inputs for which the function g is hard to invert, then the function g is "slightly" hard to invert on average!
- The next question is: Can we "amplify this nugget of hardness" in the function g to get a one-way function?

- Suppose $g^{(k)}\colon \left(\{0,1\}^n \times \{0,1\}^n\right)^k \to \left(\{0,1\}^{2n}\right)^k$ defined as follows

$$g^{(k)}(x_1, y_1, x_2, y_2, \ldots, x_k, y_k) = (x_1 \cdot y_1, x_2 \cdot y_2, \ldots, x_k \cdot y_k)$$

- Note that an adversary who inverts $g^{(k)}$ factorizes every $z_i = x_i \cdot y_i$, where $1 \leqslant i \leqslant n$
- From the previous claim, we have, for all $1 \leqslant i \leqslant k$

$$\mathbb{P}\left[\mathcal{A} \text{ factorizes } z_i\right] \lesssim 1 - \frac{1}{4n^2}$$

- So, the probability that $\mathcal{A}$ inverts all z_i, where $1 \leqslant i \leqslant k$, is

$$\mathbb{P}\left[\mathcal{A} \text{ factorizes } z_1, \ldots, z_k\right] \lesssim \left(1 - \frac{1}{4n^2}\right)^k \leqslant \exp(-k/4n^2)$$

- Note that is we use $k = 4n^2 t$, then the probability that any efficient adversary $\mathcal{A}$ inverts $g^{(k)}$ is $\lesssim \exp(-t)$

Second Candidate: Discrete Log is Hard

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

Third 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 $y = x^2 \mod N$.
  (Hint: First, give efficient algorithm for square-root over prime-order fields. Then use Chinese remainder theorem.)

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 $f\colon \{0,1\}^n \rightarrow \{0,1\}^n$ 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 assign $= \{x_{i_1} \mapsto b_1, x_{i_2} \mapsto b_2, \ldots, x_{i_k} \mapsto b_k\}$, we define $\varphi(x)|_{\mathsf{assign}}$ by setting the values of $x_{i_1}, \ldots, x_{i_k}$ as $b_1, \ldots, b_k$ in $\varphi(x)$ and simplifying
- Again, if P = NP and f is efficiently computable, then it is efficient to find whether $\varphi(x)|_{\mathsf{assign}}$ 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 P $=$ 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 $P + Q$