CSC 2426: Fundamentals of Cryptography **Fall 2023**

Lecture 1: Introduction, Negligible Functions

Instructor: Akshayaram Srinivasan *Scribe: Ziyang Jin*

Date: 2023-09-11

1.1 What is cryptography?

Modern cryptography is about building secure systems that hide private information from adversaries.

Example 1. Suppose we have two people, Alice and Bob. Alice tries to send a secret message m to Bob through a public channel. Eve, the adversary, has access to all communication sent between Alice and Bob through the channel. Cryptography can be used to ensure that Eve learns no information about m.

Example 2. Alice tries to send a message m to Bob through a public channel. This time, Eve not only has access to m through the public channel, but she can also tamper the message m. How does Bob know that the message he gets is not tampered? Cryptography can be used to protect the integrity of messages.

Example 1 and 2 are about protecting the data. In addition to this, cryptography is also concerned about protecting secrecy and integrity of computation. See examples 3 and 4 below.

Example 3. Alice has some data stored by a cloud service provider (say AWS, Azure, Google Cloud, etc), and she wants to compute some function f on the data stored in the cloud. However, Alice neither wants the cloud service provider to learn any useful information about the data nor she wants to download the complete data; she only wants to learn the output of the computation. How can Alice achieve it?

Example 4. Alice asks the cloud service provider to compute some function f on the data stored in the cloud. How does Alice make sure that function f is computed correctly?

1.2 What is this course about?

This course talks about the theoretical foundations of cryptography. In this course, we will:

- Develop a mathematical framework that allows us to precisely define the properties to be satisfied by secure systems.
- Look at constructions of secure systems.
- Prove that these systems satisfy the required properties.

1.3 Negligible function

Definition. Given a function $\mu : \mathbb{N} \to [0, 1]$. We say μ is *negligible* if for all polynomials p, there exists

$n_0 \in \mathbb{N}$, such that $\forall n \geq n_0, \mu(n) \leq \frac{1}{p(n)}$.

Let's consider a couple of functions and see if they are negligible or non-negligible.

- $\mu_1(n) = \frac{1}{n^2}$ is not negligible since we can take $p(n) = n^3$ and for all $n \geq 1$, $\mu(n) = \frac{1}{n^2} \geq \frac{1}{n^3} = \frac{1}{p(n)}$.
- $\mu_2(n) = 2^{-n}$ is negligible since for any polynomial $p(n)$, there always exists n_0 such that $\forall n \geq n_0, \mu_2(n) \leq \frac{1}{p(n)}$. This is because $\mu_2(n)$ is exponential, so it is asymptotically smaller than any inverse polynomial $1/p(n)$.

Theorem. If μ_1, μ_2 are negligible functions, then $\mu = \mu_1 + \mu_2$ is also negligible.

Proof: For any polynomial $q(n)$, consider $2q(n)$, which is also a polynomial. Since μ_1 is negligible, there exists a $n_1 \in \mathbb{N}$ such that $\forall n \geq n_1$, $\mu_1(n) \leq \frac{1}{2q(n)}$. Since μ_2 is negligible, there exists a $n_2 \in \mathbb{N}$ such that $\forall n \geq n_1$, $\mu_1(n) \leq \frac{1}{2q(n)}$. Let $n_0 = \max(n_1, n_2)$. Then we have for all $n \geq n_0$, $\mu = \mu_1 + \mu_2 \leq \frac{1}{2q(n)} + \frac{1}{2q(n)} = \frac{1}{q(n)}$, which proves that μ is negligible. ■

Theorem. If $\mu_1, \mu_2, ..., \mu_c$ are negligible functions for some $c \in \mathbb{N}$, then $\mu = \mu_1 + \mu_2 + ... + \mu_c$ is negligible.

The proof is similar. For any polynomial $q(n)$, instead of consider $2q(n)$, we just need to consider $cq(n)$, and take $n_0 = \max(n_1, ..., n_c)$.

1.4 One-way functions

Informally speaking, a Probabilistic Polynomial-time Turing machine (abbr. PPT) $M(x; r)$ is a Turing machine that takes x as input and random variable r (can think of it as random coins), and runs in polynomial time and outputs a random variable.

Definition 1.1 *A non-uniform probabilistic polynomial-time Turing machine (abbr. nuPPT) is a set of infinite P.P.T's* $\mathcal{A} = \{M_1, M_2, ..., M_n, ...\}$ *where we use* M_i *to compute the output on inputs of length* i.

Definition 1.2 *Let* $f = \{f_n : \{0,1\}^{k(n)} \to \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$ *be a family of functions. We say* f *is a one-way function if:*

- ***Easy to evaluate:*** *There exists a polynomial algorithm that for every* $n \in \mathbb{N}$ *takes* $x \in \{0,1\}^{k(n)}$ *and outputs* $f_n(x)$.
- ***Hard to Invert:*** *For all non-uniform PPT* $\mathcal{A}$,

$$\Pr_{x \leftarrow \{0,1\}^{k(n)}}[\mathcal{A}(1^{k(n)}, f_n(x)) = y \textit{ s.t. } f(y) = f(x)] \textit{ is negligible}$$

The first condition ensures that f is easy to compute. The second condition says that given the output of f on a randomly chosen input ($x \leftarrow \{0,1\}^{k(n)}$ means x is sampled from the set of all binary strings of size $k(n)$), it is hard to invert f.

Remark 1.3 *We need* $1^{k(n)}$ *as an additional input to* $\mathcal{A}$ *to ensure that* $\mathcal{A}$ *has enough time to write down a pre-image. In case* $m(n)$ *is very small, like* $\log n$, *if we only allow* $\mathcal{A}$ *to run in time polynomial to* $m(n)$, *then* $\mathcal{A}$ *might not even have enough time to write down the entire input* x.