University of Illinois, Urbana Champaign
CS 598DK Special Topics in Cryptography

Instructor: Dakshita Khurana
Scribe: Muhammad Haris
Date: August 28, 2019

LECTURE
2

One Way Functions, Hard Core Bit - I

First, we will understand the notion of negligible functions, which intuitively are functions that are "too small to matter" from the point of view of theoretical cryptography. We will also look at the class of adversaries against which we will typically require provable security: non-uniform probabilistic polynomial time Turing Machines.

Next, we will initiate our study of cryptographic hardness by looking at the simplest cryptographic primitive: a one-way function. We will see how a one-way function is defined, and show that the existence of one-way functions implies that $\mathsf{P} \neq \mathsf{NP}$. This means that we can only conjecture that one-way functions exist, *actually proving* that one-way functions exist is at least as hard as proving that $\mathsf{P} \neq \mathsf{NP}$.

Finally, we will introduce the concept of a hard-core bit of a one-way function, which offers a way to condense all the hardness of a one-way function into the unpredictability of a single bit.

2.1 Preliminaries

2.1.1 Negligible Function

Definition 2.1. A function $\mathsf{v}(\cdot)$ is negligible if $\forall \mathsf{c} \geq 0, \mathsf{c} \in \mathbb{N}, \exists \mathsf{k}_0 (\in \mathbb{N})$ such that $\forall \mathsf{k} (\in \mathbb{N}) \geq \mathsf{k}_0$,

$$|\mathsf{v}(\mathsf{k})| < \frac{1}{\mathsf{k}^{\mathsf{c}}} \text{ or } \mathsf{v}(\mathsf{k}) = \frac{1}{\mathsf{k}^{\mathsf{w}(1)}}$$

In words, the function $v : \mathbb{N} \to \mathbb{N}$ is called *negligible* if it approaches zero faster than the reciprocal of any polynomial. That is for every $\mathsf{c} \in \mathbb{N}$ there is an integer k_0 such that $|\mathsf{v}(\mathsf{k})| < \mathsf{k}^{-\mathsf{c}}$ for all $\mathsf{k} \geq \mathsf{k}_0$.

Notation: We will use $\mathtt{negl}(\mathsf{k})$ to denote any function that is asymptotically smaller than the reciprocal of every polynomial.

Examples:

- $\frac{1}{2^{\mathsf{k}}} = \mathtt{negl}(\mathsf{k})$, because $\frac{1}{2^{\mathsf{k}}}$ approach zero faster than the reciprocal of any polynomial.
- $\frac{1}{\mathsf{k}^3} \neq \mathtt{negl}(\mathsf{k})$, because it does not approach zero faster than $\frac{1}{\mathsf{k}^4}$.

Properties:

- Let $v_1(k), v_2(k)$ are $\mathtt{negl}(\mathsf{k})$, then $v_1(k) + v_2(k) = \mathtt{negl}(\mathsf{k})$
- Suppose $\mathsf{poly}(\mathsf{k})$ represents a polynomial function in k, then $\mathsf{poly}(\mathsf{k}) * \mathtt{negl}(\mathsf{k}) = \mathtt{negl}(\mathsf{k})$

Remark 2.2. From theoretical crypto point of view, negligble reflects "too small to matter".

2.1.2 Turing Machines

Definition 2.3. Probabilistic Polynomial Time (PPT): A Turing machine $\mathcal{A}$ is a PPT Turing machine if $\exists \mathsf{c} \in \mathbb{N}$ such that $\forall x, \mathcal{A}(x)$ halts in $|x|^c$ steps (where $|x|$ denotes the size of x, equivalently the number of bits in the binary representation of x).

A non-uniform PPT Turing Machine (nuPPT) is allowed to specify different machines for different input lengths, as opposed a single machine that must work for any input length:

Definition 2.4. Non Uniform Probabilistic Polynomial Time (nuPPT): A nuPPT machine is a sequence of probabilistic Turing machines $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, ..)$ such that $\exists \mathsf{c} \in \mathbb{N}$ such that $\forall \mathsf{x}, \mathcal{A}_{|\mathsf{x}|}(\mathsf{x})$ halts in $(|x|^c)$ steps.

2.2 One Way Functions

Intuitively, a one-way function is a function f that is easy to compute, but hard to invert. We formalize this by saying that there cannot exist a machine that can invert f in polynomial time.

Definition 2.5. (Attempt 1)
A function $\mathsf{f} : \{0,1\}^* \to \{0,1\}^*$ is one way function if

- f is **easy** to compute, where **easy** means computable by uniform **PPT** machine.
- $\forall$nuPPT machines (adversaries) $\mathcal{A}, \exists$ a negligible function $\mathsf{v}_{\mathcal{A}}$ such that: $\mathsf{Pr}_{x \xleftarrow{\$} \{0,1\}}[\mathcal{A}(f(x)) \in f^{-1}(f(x))] = \mathsf{v}_{\mathcal{A}}(k)$

Note that in this definition, $f^{-1}(y)$ is the set of all values of x such that $f(x) = y$. The notation $x \xleftarrow{\$} \{0,1\}^*$ means that x is drawn uniformly at random from the set $\{0,1\}^*$.

Problem- input size of $\mathcal{A}$: The above definition has a problem, the nuPPT machine $\mathcal{A}$(adversary) is only given $f(x)$ as input. If size of $f(x)$ is much smaller than x, the adversary only gets time polynomial in $|f(x)|$ in order to invert f to recover an k bit value of x. This is especially problematic if $|f(x|$ is exponentially smaller than $|x|$.

A concrete example is a function $f(x) = \mathsf{size}(x)$ (where $\mathsf{size}(x)$ is a function that just outputs the number of bits in x). Intuitively, this shouldn't be a one-way function since it is pretty easy to invert. But, this function satisfies Definition 2.6!

The reason is that by Definition 2.4, the nuPPT machine $\mathcal{A}$ can only run in time polynomial in the size of its input. The input to $\mathcal{A}$ is $f(x)$, so the *size* of input to $\mathcal{A}$ is $\mathsf{size}(f(x))$. Since $f(x) = \mathsf{size}(x)$, $\mathsf{size}(f(x)) = \log(\mathsf{size}(x)) = \log\log x$. Therefore $\mathcal{A}$ is allowed only $\mathsf{poly}(\log\log x)$ steps. But $\mathcal{A}$ needs at least $\log x$ steps to write down all the bits in x – and therefore no nuPPT machine can even write down all the bits in x!

In order to fix this issue, when x is sampled as a k bit string, the adversary is given 1^k (k repetition of 1 bit) as input. That is, the nuPPT adversary gets $f(x)$ and 1^k as input. So attacker gets time polynomial in $|f(x)| + k$ to invert $f(x)$. The resulting (more accurate) definition of one-way functions is below.

Definition 2.6. (Final)
A function $\mathsf{f} : \{0,1\}^* \to \{0,1\}^*$ is a one way function if

- f is **easy** to compute, where **easy** means computable by uniform **PPT** machine.
- $\forall \mathsf{nuPPT}$ machines (adversaries) $\mathcal{A}, \exists$ negligible function $\mathsf{v}_{\mathcal{A}}$ such that: $\Pr_{x \xleftarrow{\$} \{0,1\}^k}[\mathcal{A}(1^k, f(x)) \in f^{-1}(f(x))] = \mathsf{v}_{\mathcal{A}}(k)$

2.2.1 Candidate One-Way Functions

One candidate one-way function can be based on the conjectured hardness of factoring. The function can be defined as follows: f uses its input x as randomness to sample two primes p and q, and outputs their product $p \cdot q$. We refer the reader to Bellare-Goldwasser for a detailed exposition and additional candidates.

2.2.2 One-Way Functions Imply $\mathsf{P} \neq \mathsf{NP}$

Claim 2.7. *If one way functions exist, then* $\mathsf{P} \neq \mathsf{NP}$.

Proof. We will prove this claim by contradiction. Towards that end, suppose the claim is not true. This means that one-way functions exist and $\mathsf{P} = \mathsf{NP}$.

Fix any one-way function f. Define

$$\mathsf{L}_f = \{(x, y, k) | \exists z : f(x||z) = y \text{ and } (\mathsf{size}(x) + \mathsf{size}(z)) = k\}.$$

Clearly, L is an NP language. By assumption, since $\mathsf{P} = \mathsf{NP}$, $L \in \mathsf{P}$. Then, there exists a uniform PPT adversary $\mathcal{A}$ that uses the polynomial time algorithm deciding membership in L, to invert the one way function f as follows. $\mathcal{A}(1^k, y)$ checks if $(0, y, k) \in L$. If true, it sets $x[1]$ (the first bit of x) to 0, and otherwise sets $x[1] = 1$. Next, it checks if $(x[1]||0, y, k) \in L$. If true, it sets $x[2]$ to 0, and otherwise $x[2] = 1$. It continues to guess x one bit at a time, appending it to string x' and sending (x', y, k) to $\mathsf{L_f}$, if $\mathsf{L_f}$ outputs 1, then the guessed bit is correct (part of x) and otherwise the flipped bit is the correct bit of x. At the end of this process, $\mathcal{A}$ outputs x, which is the inverse of y, contradicting the fact that f is a one-way function.

Therefore, if one-way functions exist, then $\mathsf{P} \neq \mathsf{NP}$. □

2.3 Hard Code Bit

Before looking at hard core bits, we ask the question: does the output of a one-way function hide all meaningful information about its input?

Question: If $f(x)$ is a OWF then is $g(x) = x[1]||f(x)$ also a OWF?
Answer: $g(x)$ is OWF function because the adversary still needs to invert $f(x)$ to learn x. The first bit of x does not reveal information about other bits of x since x is sampled uniformly at random.

This illustrates that the output of a one-way function (eg, the function $g(\cdot)$ in the example above) could reveal information about certain bits of x. But can we define some Boolean predicate of x that *is completely hidden* by the output of the one-way function?

It turns out that we can, and the resulting predicate is called a hard-core bit/hard-core predicate. Informally, a hardcore predicate is a function B such that even given $f(x)$, guessing $B(x)$ is as hard as finding x.

Formally, we will define:

DEFINITION 2.8. $\mathsf{P} : \{0,1\}^* \to \{0,1\}$ is a hard-core predicate of a OWF function $\mathsf{g(x)}$ if

- $\mathsf{P(x)}$ is PPT computable.
- $\forall \mathsf{nuPPT}\mathcal{A}$,

$$\mathsf{Pr}_{\mathsf{x} \xleftarrow{\$} \{0,1\}^{\mathsf{k}}}[\mathcal{A}(\mathsf{g(x)}) = \mathsf{P(x)}] \leq \frac{1}{2} + \mathsf{negl(k)}$$

Next Class: We will better understand the definition of a hard core predicate, and look at non-interactive commitment schemes.

Acknowledgement

These scribe notes were prepared by editing a light modification of the template designed by Alexander Sherstov.