Foundation of Cryptography, Introduction

Adminstration + Introduction

Benny Applebaum & Iftach Haitner, Tel Aviv University
(Slightly edited by Ronen Shaltiel, all errors are by Ronen Shaltiel)

University of Haifa.

2018

Part I

Administration and Course Overview

Section 1

Administration

Important Details

1. There will be a final exam.

Important Details

1. There will be a final exam.
2. Course website: Can be reached from Ronen's homepage.

Course Prerequisites

1. Computational Models
2. Probability theory.

Course Material

1. Books:
   1.1 Oded Goldreich. Foundations of Cryptography.
   1.2 Jonathan Katz and Yehuda Lindell. An Introduction to Modern Cryptography.
2. Lecture notes
   2.1 Ran Canetti `www.cs.tau.ac.il/~canetti/f08.html`
   2.2 Yehuda Lindell `u.cs.biu.ac.il/~lindell/89-856/main-89-856.html`
   2.3 Luca Trevisan `www.cs.berkeley.edu/~daw/cs276/`
   2.4 Salil Vadhan `people.seas.harvard.edu/~salil/cs120/`
   2.5 Benny Applebaum and Iftach Haitner `http://moodle.tau.ac.il/2016/course/view.php?id=368416201`

Section 2

Course Topics

Course Topics

Basic primitives in cryptography (i.e., one-way functions, pseudorandom generators and zero-knowledge proofs).

- Focus on *formal* definitions and *rigorous* proofs.
- The goal is not studying some list, but to understand cryptography.
- Start with “what is security?”
- Only then do we ask how to achieve it.
- Start from the bottom and work our way up.

Part II

Foundation of Cryptography

Some stories and motivation

- Encryption (symmetric and public-key).
- Coin tossing over the phone (impossible information theoretically, but possible against poly-time adversaries).

Section 3

Cryptography and Computational Hardness

Cryptography and Computational Hardness

1. What is Cryptography?

Cryptography and Computational Hardness

1. What is Cryptography?
2. Hardness assumptions, why do we need them?

Cryptography and Computational Hardness

1. What is Cryptography?
2. Hardness assumptions, why do we need them?
3. Does $\mathcal{P} \neq \mathcal{NP}$ suffice?

Cryptography and Computational Hardness

1. What is Cryptography?
2. Hardness assumptions, why do we need them?
3. Does $\mathcal{P} \neq \mathcal{NP}$ suffice?

   $\mathcal{NP}$: all (languages) $L \subset \{0,1\}^*$ for which there exists a polynomial-time algorithm V and (a polynomial) $p \in \mathrm{poly}$ such that the following hold:

Cryptography and Computational Hardness

1. What is Cryptography?
2. Hardness assumptions, why do we need them?
3. Does $\mathcal{P} \neq \mathcal{NP}$ suffice?

   $\mathcal{NP}$: all (languages) $L \subset \{0,1\}^*$ for which there exists a polynomial-time algorithm V and (a polynomial) $p \in \mathrm{poly}$ such that the following hold:

   3.1 $V(x, w) = 0$ for any $x \notin L$ and $w \in \{0,1\}^*$

Cryptography and Computational Hardness

1. What is Cryptography?
2. Hardness assumptions, why do we need them?
3. Does $\mathcal{P} \neq \mathcal{NP}$ suffice?

   $\mathcal{NP}$: all (languages) $L \subset \{0,1\}^*$ for which there exists a polynomial-time algorithm V and (a polynomial) $p \in \text{poly}$ such that the following hold:

   3.1 $V(x, w) = 0$ for any $x \notin L$ and $w \in \{0,1\}^*$

   3.2 for any $x \in L$, $\exists w \in \{0,1\}^*$ with $|w| \leq p(|x|)$ and $V(x, w) = 1$

Cryptography and Computational Hardness

1. What is Cryptography?
2. Hardness assumptions, why do we need them?
3. Does $\mathcal{P} \neq \mathcal{NP}$ suffice?

   $\mathcal{NP}$: all (languages) $L \subset \{0,1\}^*$ for which there exists a polynomial-time algorithm V and (a polynomial) $p \in \mathrm{poly}$ such that the following hold:

   3.1 $V(x, w) = 0$ for any $x \notin L$ and $w \in \{0,1\}^*$

   3.2 for any $x \in L$, $\exists w \in \{0,1\}^*$ with $|w| \leq p(|x|)$ and $V(x, w) = 1$

   $\mathcal{P} \neq \mathcal{NP}$: i.e., $\exists L \in \mathcal{NP}$, such that for any polynomial-time algorithm A, $\exists x \in \{0,1\}^*$ with $\mathsf{A}(x) \neq 1_L(x)$

Cryptography and Computational Hardness

1. What is Cryptography?
2. Hardness assumptions, why do we need them?
3. Does $\mathcal{P} \neq \mathcal{NP}$ suffice?

$\mathcal{NP}$: all (languages) $L \subset \{0,1\}^*$ for which there exists a polynomial-time algorithm V and (a polynomial) $p \in \mathrm{poly}$ such that the following hold:

3.1 $V(x,w) = 0$ for any $x \notin L$ and $w \in \{0,1\}^*$

3.2 for any $x \in L$, $\exists w \in \{0,1\}^*$ with $|w| \leq p(|x|)$ and $V(x,w) = 1$

$\mathcal{P} \neq \mathcal{NP}$: i.e., $\exists L \in \mathcal{NP}$, such that for any polynomial-time algorithm A, $\exists x \in \{0,1\}^*$ with $\mathsf{A}(x) \neq 1_L(x)$

polynomial-time algorithms: an algorithm A runs in polynomial-time, if $\exists p \in \mathrm{poly}$ such that the running time of $A(x)$ is bounded by $p(|x|)$ for any $x \in \{0,1\}^*$

Cryptography and Computational Hardness

1. What is Cryptography?
2. Hardness assumptions, why do we need them?
3. Does $\mathcal{P} \neq \mathcal{NP}$ suffice?

   $\mathcal{NP}$: all (languages) $L \subset \{0,1\}^*$ for which there exists a polynomial-time algorithm V and (a polynomial) $p \in \mathsf{poly}$ such that the following hold:

   3.1 $V(x,w) = 0$ for any $x \notin L$ and $w \in \{0,1\}^*$

   3.2 for any $x \in L$, $\exists w \in \{0,1\}^*$ with $|w| \leq p(|x|)$ and $V(x,w) = 1$

   $\mathcal{P} \neq \mathcal{NP}$: i.e., $\exists L \in \mathcal{NP}$, such that for any polynomial-time algorithm A, $\exists x \in \{0,1\}^*$ with $\mathsf{A}(x) \neq 1_L(x)$

   polynomial-time algorithms: an algorithm A runs in polynomial-time, if $\exists p \in \mathsf{poly}$ such that the running time of $A(x)$ is bounded by $p(|x|)$ for any $x \in \{0,1\}^*$
4. Problems: hard on the average. No known solution

Cryptography and Computational Hardness

1. What is Cryptography?
2. Hardness assumptions, why do we need them?
3. Does $\mathcal{P} \neq \mathcal{NP}$ suffice?

   $\mathcal{NP}$: all (languages) $L \subset \{0,1\}^*$ for which there exists a polynomial-time algorithm V and (a polynomial) $p \in \mathrm{poly}$ such that the following hold:

   3.1 $V(x,w) = 0$ for any $x \notin L$ and $w \in \{0,1\}^*$

   3.2 for any $x \in L$, $\exists w \in \{0,1\}^*$ with $|w| \leq p(|x|)$ and $V(x,w) = 1$

   $\mathcal{P} \neq \mathcal{NP}$: i.e., $\exists L \in \mathcal{NP}$, such that for any polynomial-time algorithm A, $\exists x \in \{0,1\}^*$ with $\mathsf{A}(x) \neq 1_L(x)$

   polynomial-time algorithms: an algorithm A runs in polynomial-time, if $\exists p \in \mathrm{poly}$ such that the running time of $A(x)$ is bounded by $p(|x|)$ for any $x \in \{0,1\}^*$
4. Problems: hard on the average. No known solution
5. One-way functions: an efficiently computable function that no efficient algorithm can invert.

Part III

Notation

Notation I

- For $t \in \mathbb{N}$, let $[t] := \{1, \ldots, t\}$.
- Given a string $x \in \{0,1\}^*$ and $0 \leq i < j \leq |x|$, let $x_{i,\ldots,j}$ stands for the substring induced by taking the $i, \ldots, j$ bit of x (i.e., $x[i] \ldots, x[j]$).
- Given a function f defined over a set $\mathcal{U}$, and a set $\mathcal{S} \subseteq \mathcal{U}$, let $f(\mathcal{S}) := \{f(x) \colon x \in \mathcal{S}\}$, and for $y \in f(\mathcal{U})$ let $f^{-1}(y) := \{x \in \mathcal{U} \colon f(x) = y\}$.
- poly stands for the set of all polynomials.
- The worst-case running-time of a *polynomial-time algorithm* on input x, is bounded by $p(|x|)$ for some $p \in \mathrm{poly}$.
- A function is *polynomial-time computable*, if there exists a polynomial-time algorithm to compute it.
- PPT stands for probabilistic polynomial-time algorithms.
- A function $\mu \colon \mathbb{N} \mapsto [0,1]$ is negligible, denoted $\mu(n) = \mathrm{neg}(n)$, if for any $p \in \mathrm{poly}$ there exists $n' \in \mathbb{N}$ with $\mu(n) \leq 1/p(n)$ for any $n > n'$.

Distribution and random variables I

- The support of a distribution P over a finite set $\mathcal{U}$, denoted $\mathsf{Supp}(P)$, is defined as $\{u \in \mathcal{U} : P(u) > 0\}$.
- Given a distribution P and en event E with $\Pr_P[E] > 0$, we let $(P \mid E)$ denote the conditional distribution P given E (i.e., $(P \mid E)(x) = \frac{P(x) \wedge E}{\Pr_P[E]}$).
- For $t \in \mathbb{N}$, let let U_t denote a random variable uniformly distributed over $\{0,1\}^t$.
- Given a random variable X, we let $x \leftarrow X$ denote that x is distributed according to X (e.g., $\Pr_{x \leftarrow X}[x = 7]$).
- Given a final set $\mathcal{S}$, we let $x \leftarrow \mathcal{S}$ denote that x is uniformly distributed in $\mathcal{S}$.
- We use the convention that when a random variable appears twice in the same expression, it refers to a *single* instance of this random variable. For instance, $\Pr[X = X] = 1$ (regardless of the definition of X).

Distribution and random variables II

- Given distribution P over $\mathcal{U}$ and $t \in \mathbb{N}$, we let P^t over $\mathcal{U}^t$ be defined by $D^t(x_1, \ldots, x_t) = \Pi_{i \in [t]} D(x_i)$.
- Similarly, given a random variable X, we let X^t denote the random variable induced by t independent samples from X.