Pseudorandomness - I

601.642/442: Modern Cryptography

Fall 2019

Randomness

- Your computer needs "randomness" for many tasks every day!
- Examples:
  - encrypting a session-key for an SSL connection (login)
  - encrypting your hard-drive for secure backup
- How does your computer generate this randomness?
  - true randomness is difficult to get
  - often, a lot of it is required (e.g. disk encryption)

Randomness

- Common sources of randomness:
  - key-strokes
  - mouse movement
  - power consumption
  - ...
- These processes can only produce so much true randomness

Fundamental Question

Can we "expand" few random bits into many random bits?

- Many heuristic approaches; good in many cases, e.g., primality testing
- But not good for cryptography, such as for data encryption
- For crypto, need bits that are "as good as truly random bits"

Pseudorandomness

- Suppose you have n uniformly random bits: $x = x_1 \| \ldots \| x_n$
- Find a **deterministic** (polynomial-time) algorithm G such that:
  - $G(x)$ outputs a $n+1$ bits: $y = y_1 \| \ldots \| y_{n+1}$
  - y looks "as good as" a truly random string $r = r_1 \| \ldots \| r_{n+1}$
- $G : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ is called a **pseudorandom generator** (PRG)
- Think: What does "as good as truly random" mean?

As good as truly random

- Should have no obvious patterns
- Pass **all** statistical tests that a truly random string would pass
  - Number of 0's and 1's roughly the same
  - ...
- Main Idea: No efficient test can tell $G(x)$ and r apart!
- Distributions:

$$\left\{x \leftarrow \{0,1\}^n : G(x)\right\} \quad \text{and} \quad \left\{r \leftarrow \{0,1\}^{n+1} : r\right\}$$

are **"computationally indistinguishable"**

Roadmap

- New crypto language: Computational Indistinguishability
- Defining Pseudorandomness using the above
- A complete test for pseudorandom distributions: Next-bit prediction
- Pseudorandom Generators
  - Small expansion
  - Arbitrary (polynomial) expansion

Distributions & Ensembles

- Distribution: X is a distribution over sample space $\mathcal{S}$ if it assigns probability p_s to the element $s \in \mathcal{S}$ s.t. $\sum_s p_s = 1$

Definition

A sequence $\{X_n\}_{n \in \mathbb{N}}$ is called an ensemble if for each $n \in \mathbb{N}$, X_n is a probability distribution over $\{0,1\}^*$.

- Generally, X_n will be a distribution over the sample space $\{0,1\}^{\ell(n)}$ (where $\ell(\cdot)$ is a polynomial)

Computational Indistinguishability

- Captures what it means for two distributions X and Y to "look alike" to *any* efficient test
- Efficient test = efficient computation = non-uniform PPT
- No **non-uniform PPT** "distinguisher" algorithm D can tell them apart
- i.e. "behavior" of D on X and Y is the same
- Think: How to formalize?

Computational Indistinguishability

- Scoring system: Give D a sample of X:
  - If D say "Sample is from X" it gets $+1$ point
  - If D say "Sample is from Y" it gets -1 point
- D's output can be encoded using just one bit:
  1 ="Sample is from X" and 0 ="Sample is from Y"
- Want: Average score of D on X and Y should be roughly same

$$\Pr\left[x \leftarrow X; D(1^n, x) = 1\right] \approx \Pr\left[y \leftarrow Y; D(1^n, y) = 1\right] \implies$$

$$\left|\Pr\left[x \leftarrow X; D(1^n, x) = 1\right] - \Pr\left[y \leftarrow Y; D(1^n, y) = 1\right]\right| \leqslant \mu(n).$$

Computationally Indistinguishability: Definition

Definition (Computationally Indistinguishability)

Two ensembles of probability distributions $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are said to be *computationally indistinguishable* if for every non-uniform PPT D there exists a negligible function $\nu(\cdot)$ s.t.:

$$\left|\Pr\left[x \leftarrow X_n; D(1^n, x) = 1\right] - \Pr\left[y \leftarrow Y_n; D(1^n, y) = 1\right]\right| \leqslant \nu(n).$$

Properties of Computational Indistinguishability

- Notation: $\{X_n\} \approx_c \{Y_n\}$ means computational indistinguishability
- **Closure:** If we apply an efficient operation on X and Y, they remain indistinguishable. That is, $\forall$ non-uniform-PPT M

$$\{X_n\} \approx_c \{Y_n\} \implies \{M(X_n)\} \approx_c M\{Y_n\}$$

  Proof Idea: If not, D can use M to tell them apart!

- **Transitivity:** If X, Y are computationally indistinguishable, and Y, Z are computationally indistinguishable; then X, Z are also computationally indistinguishable.

Generalizing Transitivity: Hybrid Lemma

Lemma (Hybrid Lemma)

Let $X^1, \ldots, X^m$ *be distribution ensembles for* $m = \text{poly(n)}$. *If for every* $i \in [m-1]$, X^i *and* X^{i+1} *are computationally indistinguishable, then* X^1 *and* X^m *are computationally indistinguishable.*

Used in most crypto proofs!