Crypto

Preliminaries and notation

- $S \subseteq \{0,1\}^*$ defines set S as a finite subset of $\{0,1\}$ all finite-length strings.
- $x \in_R S$, R indicates x is chosen randomly / uniformly from S
- U_n x is chosen from the set of all n-bit strings
- $\mu(\cdot)$ means the negligible function can take any input. Negligible functions decrease faster than inverse polynomial as n increases
- must use positive polynomial, if not, maybe it won't be negligible. We want $\mu(n) < \frac{1}{p(n)}$
- λ is an empty string

Polynomial time, Security parameter

- Polynomial time refers to the computational complexity of an algorithm with respect to the security parameter
- it means the protocol is efficient and practical because algorithms are feasible in polynomial time
- But also secure because breaking the algorithm is infeasible
- security parameter determines level of security e.g. length of keys in bits
- an algorithm running in polynomial time in the security parameter can be expressed as a polynomial function of the security parameter. That is, there exists $p(\lambda)$ such that $\mathcal{O}(p(\lambda))$
- E.g. if $\lambda = 128$ bits runs in polynomial time, running time will be a function of λ like $\mathcal{O}\lambda^2$
- In contrast to exponential time which are $\mathcal{O}2^{\lambda}$

Uniform, Non-Uniform Non uniform algorithms aren't non-uniform randomness! Uniform algorithms don't change procedure based on the input size. Non-uniform algorithms can have logic based around the input size. Uniform algorithms have a fixed strategy, non-uniform can adapt. The non-uniformity is not about randomness but the potential for the algorithm to have different strategies for input lengths. Allowing a distinguisher algorithm D to be non-uniform means it's a powerful attacker that has different strategies for each input length (key length or message size) and can more easily distinguish.

Negligible Function A function $\mu(n)$ is negligible if it decreases faster than the inverse of any polynomial.

Definition 1. *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*

$$\forall\, n \geq n_0, \;\; \mu(n) \leq \frac{1}{p(n)}$$

Tests: are the following negligible?

1. $\mu(n) = \frac{1}{n^2}$
2. $\mu(n) = \frac{1}{2^n}$
3. $\mu(n) = \frac{1}{n!}$
4. $\mu(n) = \frac{1}{2^{-n}}$
5. $\mu(n) = \frac{1}{2^{\log(n)}}$
6. $\mu(n) = \frac{1}{n^{\log(n)}}$

Answer and Discussion

1. $\frac{1}{n^k}$ is a inverse power function, aka polynomial time decreasing function. e.g. inverse cubic $1/n^3$, inverse quartic $1/n^4$, they approach 0 as n approaches ∞. They are efficiently computable and tractable, though non-negligible.

2. $\mu(n) = \frac{1}{2^n}$ is an inverse exponential function and will always satsify the inequality because an exponential function will decrease faster than the inverse polynomial. It's non-negligible and used in crypto

3. $\mu(n) = \frac{1}{n!}$ is an inverse factorial, defined only for non-negative, it's super exponential decreasing faster than exponential and isn't seen in computer science but maybe in poisson distribution. It's non-negligible but not used

4. $\mu(n) = \frac{1}{2^{-n}}$ without calculation, this is 2^n which is exponential growth rather than decay, definately not negligible.

5. $\frac{1}{2^{\log(n)}}$ is negligible but is sub-exponential, decreasing slower than $\frac{1}{2^n}$. For any $p(n)$ we show that a large enough n satisifes our inequality. $\frac{1}{2^{\log(n)}} = \frac{1}{n^{\log(2)}}$, $n^{\log(2)} = n^{0.693}$ therefore for sufficiently large n, this satifies the inequality.

6. $\frac{1}{n^{\log(n)}}$ decreases faster than the above

log/exp rule: $x^{\log_a^{(y)}} = y^{\log_a^{(x)}}$

Theory of Computation

- a turing machine is a theoretical device that "manipulates symbols on a strip of tape according to a table of rules" simulating algorithm logic
- Includes an infinitely long tape divided into blocks, a head can read and write symbols on the tape, a state register storing the machine state, a finite table of instructions
- We use unary 1^n, a string of 1's on the security paramter tape for reasons:
  - Unary: 1^n provides unary representation of n meaning input length corresponds with n, the longest possible representation (Worst case and Lower bound)
  - in binary, n is represented in $log_2(n)$ bits. e.g. 1000 is 11111101000 = 10 bits long. In unary, 1^{1000} is a string of 1000 ones.
  - using binary, an algorithm taking time proportional to input length would run in $\mathcal{O}(log(n))$ which runs in $\mathcal{O}(n)$
- Security parameter tape is used to model how a system scales with security parameter, 1^λ is written on it e.g. string of 1's
- This means, the function is bounded by the length of the input on the security parameter tape

Distinguishing Advantage Quantifies D's ability to distinguish between X and Y when given a sample from each $\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]$. The definition states this distinguishing advantage must be $\leq$ some negligible function $\mu(n)$ for suffiently large n.

Algorithm bounds The definition of computational indistinguishability states the distinguishing advantage $\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]$ is bound by a negligible function which by definition "Given unbounded computational power" "Brute force atttack"

Computational Indistinguishability

Two probability ensembles, X, Y are computationally indistinguishabile: $X \overset{c}{\equiv} Y$ if

$$|\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]| \leq \mu(n)$$

- n is input length, the security parameter
- a is an auxillary input drawn from a binary string of any length, could be public parameters
- ensembles X, Y are computationally indistinguishable if no polynomial time algorithm D can tell them apart with greater than negligible advantage.
- D is a PPT algorithm trying to distinguish between samples from X and Y, not guessing the bit
- $D's$ output is binary $\{0, 1\}$. Output 1 means a successful distinguish

- We look at the absolute value difference in probability of D outputting 1 for X vs Y

$$X = \{X(a,n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$$

- Set $X = \{X(a,n)\}$ defines the set of random variables X
- Subscript $a \in \{0,1\}^*; n \in \mathbb{N}$ defines the indexing, that is, exactly what values of a, n can take for infinitely any element in set X. e.g. $X('01', 3)$ is the index of a random variable X where a is a binary string of any finite length "0, 1, 01, 000", n is a natural number 1,2
- $\{0,1\}^*$ is the Kleene star operation, means all finite strings, an infinite set because there's no limit to the length
- $n \in \mathbb{N}$ means n can be any natural number, also an infinite set
- indexing gives us an address or a way to talk about 1 specific random variable rather than the collection
- "probability ensemble" is a term in cryptography / probability theory referring to a collection or family of probability distributions or random variables. Used to describe systems where behaviours depend on on input length, security parameter, etc
- "ensemble" means we're dealing with a collection of probabilistic objects rather than a single fixed distribution

Non-uniformity

- D is defined above as non-uniform which increases its power to distinguish
- Why it's important for computational Indistinguishability to be non-uniform?

Order of quantifiers for computational indistinguishability Questions - What are the tapes that are referred to? - How does the definition claim uniformity? Something to do with D Di'ng the random variabnles based on the indexed input - what's the difference between leq negligible function and ¡ 1/poly(n)?