CS 6222 Grad Crypto September 2, 2025

Lecture 3: Efficiency and Computational Indistinguishability

Lecturer: Jack Doerner *Scribe: Christopher Asuncion*

1 Topics Covered

- Efficient Computation
- Computational Indistinguishability
- Applications of Computational Indistinguishability

In a previous lecture, we introduced the one-time pad encryption scheme:

Definition 1 (One Time Pad (OTP) for n-Bit Messsages) *Let* $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^n$. *The* One-Time Pad *encryption scheme is defined as follows:*

$\mathsf{Gen} : 1^n \mapsto k : k \leftarrow \{0,1\}^n$

$\mathsf{Enc} : k, m \mapsto m \oplus k$

$\mathsf{Dec} : k, c \mapsto c \oplus k$

We also proved Shannon's theorem, which tells us that in order to achieve perfect secrecy, an encryption scheme *must* have $\mathcal{K} \geq \mathcal{M}$, and therefore one-time pad is optimal for this notion. We also observed that the definition of perfect secrecy places no restrictions whatsoever upon the power of the adversary. In today's lecture, we will investigate how the definition of security for encryption can be weakened in order to permit shorter keys, by restricting what kind of adversaries we consider, while still capturing all adversaries that we believe exist in the real world. Toward this end, imagine that we had function $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ for some polynomial ℓ.[1] such that if G is provided a truly random n-bit string as input, then its output "looks random" to any real-world adversary. Intuitively, we could use such a function to encrypt $\ell(n)$-bit messages using n-bit keys:

$\mathsf{Enc}' : k, m \mapsto m \oplus G(k)$

$\mathsf{Dec}' : k, c \mapsto c \oplus G(k)$

In order to maintain correctness, G must be deterministic. This implies that the image of G is much smaller than the set of all $\ell(n)$-bit random strings, and therefore an unbounded adversary can distinguish a random $\ell(n)$-bit string from a random output of G with non-negligible advantage just by checking whether the $\ell(n)$-bit string in question is in the image of G or not. This kind of attack might require enumerating all possible outputs of G,

[1] In order for G to be useful, we will need $\ell(n) > n$.

which isn't very realistic, since there are exponentially many of them. We wish to develop a quantification for adversaries that rules out this kind of exponential-time brute-force attack, while still capturing any kind of *efficient* statistical test that a realistic adversary would be able to perform.

It's worth noting that even if an adversary cannot perform the aforementioned exponential-time attack, it can still guess a *few* preimages of G, and if it gets lucky, one of them might correspond to the string it wishes to distinguish. This kind of lucky-guessing attack is acceptable if we can tune the parameters so that the chance of any realistic adversary succeeding is very small.

Thus, in this lecture, we will answer the following questions:

1. How do we define "efficient" computation and "efficient" adversaries, in contrast to inefficient ones that can perform exponential-time attacks?
2. What does it mean for a string to "look" random to an adversary, without *being* random?
3. What constitutes a sufficiently "small" probability of adversarial success?

2 Efficient Computation

Definition 2 (Runtime and Probabilistic Polynomial Time) *A (possibly randomized) Turing Machine A runs in time $T(n)$ if $\forall x \in \{0,1\}^*$ and for every random tape, $A(x)$ halts for $T(|x|)$ steps at most. If T is a polynomial (that is, $\exists c, n_0$ such that $\forall n \geq n_0$, $T(n) \leq n^c$) we say that A is* Probabilistic Polynomial Time (PPT)*, or simply Polynomial Time if A is deterministic.*

Definition 3 (Efficient Computation) *An algorithm is said informally to be* efficient *if it runs in probabilistic polynomial time.*

The choice to identify polynomial time with *efficiency* or real-world feasibility is ultimately an arbitrary one, but we find it to be *useful* because:

1. Polynomial time is closed under composition.
2. Polynomial time is insensitive to representations of the algorithm. This means that no matter how we represent the algorithm, whether it be with Turing Machines, C programs, circuits, or some other means, the conversion between the representations only affects the runtime by a polynomial factor. This is a corollary of the Extended Church-Turing Thesis[2] [BV97, Coo22].
3. Based on our human experiences, algorithms that are not computable in polynomial time quickly become infeasible to compute as their input size grows. To compute polynomial-time algorithms on large inputs, it is usually possible to buy sufficient computing resources (or at least conceivable to build sufficient computing resources, someday), whereas this is not true for super-polynomial-time algorithms.

[2] Also known as the Complexity-Theoretic Church-Turing Thesis, or the Feasibility Thesis.

3 Computational Indistinguishability

Definition 4 (Non-Uniform Probabilistic Polynomial Time) *A* non-uniform probabilistic polynomial time (NUPPT) *algorithm A is an infinite sequence of randomized Turing Machines $A = \{A_1, A_2, \dots\}$ such that $\forall n \in \mathbb{N}$, each machine A_n has a run time upper-bounded by some constant T_n, and there exists a polynomial p such that $|A_n| \leq p(n)$ and $T_n \leq p(n)$. That is, both the description length and the runtime of A_n are bounded by p. For a NUPPT A, we often write $A(x)$ to mean $A_{|x|}(x)$.*

Note 1 (Advice) *NUPPT machines compute the complexity class P/poly, and are equivalent to families of polynomial-sized circuits, and to uniform turing machines that receive polynomially-bounded* advice *[Aut25]. In the context of adversarial machines, this advice can be thought of as encoding any a-priori information[3] that the adversary might use to its advantage in the game that it plays; for example, the factorizations of some large biprimes.*

Definition 5 (Negligible Function) *A function $\varepsilon : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is* negligible *if $\forall c\ \exists n_0$ such that $\forall n \geq n_0$, $\varepsilon(n) < \frac{1}{n^c}$. That is, negligible functions diminish faster than any inverse polynomial.*

Examples: 2^{-n}, $n^{100}2^{-n}$, $2^{-\sqrt{n}}$

The choice to use negligible functions to bound adversarial success probability is ultimately arbitrary, and as above, we make this choice because we find it useful:

1. Negligible functions are closed under addition and under multiplication by polynomials. So, for example, $\mathsf{negl}_1(n) + \mathsf{negl}_2(n)$ is negligible, and if p is any polynomial, then $p(n) \cdot \mathsf{negl}(n)$ is negligible.

2. Combining polynomial limits on the adversarial runtime with negligible bounds on adversarial advantage yields schemes that become more secure even as computation power grows. Consider the following example:[4]

   > Suppose that running some some encryption scheme honestly requires n^2 computing power. At any given moment in history, we set n so that honest users take a consistent amount of time to run (so, as computing power increases over time, we increase n to match). Now suppose that the amount of computing power available to an adversary grows quadratically with the amount of power available to honest users; not only do adversaries have more power, their power grows faster over time. Specifically, honest users can do n^2 computation and adversaries n^4 at any given moment. If the failure rate of our system is bounded by the negligible function 2^{-n}, then our encryption scheme actually becomes *more* secure as computing power increases, even though the adversaries' power is growing faster than the honest users' power.

[3] That is, information that is independent of the random coins sampled in the course of the game.
[4] This example is borrowed from [KL20].

Now consider our algorithm G again. Let U_n be the uniform distribution over the set of all n-bit strings. We wish to define what it means for $G(U_n)$ to "look like" $U_{\ell(n)}$, and more generally what it means for any distribution to "look like" another distribution. The tools we have defined above allow us to bound adversarial power and success probability *asymptotically*. Therefore we can only reason about sequences of these distributions.

Definition 6 (Ensemble of Distributions) *An* ensemble of distributions *is a sequence* $\mathcal{X} = \{X_1, X_2, \dots\}$ *such that* $\forall n \in \mathbb{N}$, X_n *is a distribution on* $\{0,1\}^*$.

Definition 7 (Computational Indistinguishability) *Let* $\mathcal{X} = \{X_1, X_2, \dots\}$ *and* $\mathcal{Y} = \{Y_1, Y_2, \dots\}$ *be ensembles such that* $\forall n \in \mathbb{N}$, X_n *and* Y_n *are distributions on* $\{0,1\}^{\ell(n)}$ *for some polynomial* ℓ. $\mathcal{X}$ *and* $\mathcal{Y}$ *are* computationally indistinguishable *if and only if* $\forall$ *NUPPT distinguishers* D, $\exists$ *a negligible function* ε *such that* $\forall n \in \mathbb{N}$:

$$|Pr[D(1^n, t) = 1 : t \leftarrow X_n] - Pr[D(1^n, t) = 1 : t \leftarrow Y_n]| < \varepsilon(n)$$

We use the notation $\mathcal{X} \approx_c \mathcal{Y}$ *to indicate that the above condition holds.*

4 Applications of Computational Indistinguishability

Finally, we can formally describe G and the distribution that it produces.

Definition 8 (Pseudorandomness) $\mathcal{X}$ *is* pseudorandom *if and only if* $\mathcal{X} \approx_c \{U_{\ell(n)}\}_{n \in \mathbb{N}}$, *where* ℓ *is some polynomial and* $U_{\ell(n)}$ *is the uniform distribution on* $\{0,1\}^{\ell(n)}$.

Definition 9 (Pseudorandom Generator) *A polynomial-time algorithm* $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ *for some polynomial* ℓ *is a* pseudorandom generator (PRG) *if and only if all of the following conditions hold:*

1. G *is deterministic*
2. $\ell(n) > n$
3. $\{G(x) : x \leftarrow U_n\}_{n \in \mathbb{N}}$ *is pseudorandom.*

Shannon's theorem tells us that $(\mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}')$ cannot possibly be perfectly secure, but we can define a basic *computational* security notion that it does satisfy, if G is a PRG.[5]

Definition 10 (EAV1-Security) *A symmetric-key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with message space* $\mathcal{M}$ *is* computationally secure against one-ciphertext eavesdropping (EAV1-Secure) *if* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *are all polynomial-time algorithms and* $\forall m_0, m_1 \in \mathcal{M}$:

$$\{\mathsf{Enc}_k(m_0) : k \leftarrow \mathsf{Gen}(1^n)\}_{n \in \mathbb{N}} \approx_c \{\mathsf{Enc}_k(m_1) : k \leftarrow \mathsf{Gen}(1^n)\}_{n \in \mathbb{N}}$$

[5] Proof of this fact is left as an exercise.

References

[Aut25] Various Authors. P/poly. `https://complexityzoo.net/Complexity_Zoo:P#ppoly`, 2025.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.

[Coo22] Stephen Cook. The p vs np problem. `https://www.claymath.org/wp-content/uploads/2022/06/pvsnp.pdf`, 2022.

[KL20] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (3rd ed.)*. 2020.