NOTES: CRYPTOGRAPHY, COMPLEXITY, AND INFORMATION THEORY

HANI T. DAWOUD

Contents

Date: July 18, 2016.

1. Entropy

Shannon Entropy. The bigger the *surprise*, the more you *learn*: when p is small, $\log(1/p)$ is large.
Min-Entropy. Think of X on $\{0,1\}^n$ having min-entropy $k < n$ as a generalization of X being uniformly distributed on a set $S \subset \{0,1\}^n$ with $|S| = 2^k$. In the latter case we say X is a *flat* (n,k)-source.

If a distribution has high min-entropy, every outcome has small probability. X has (high) min-entropy if it does not have very heavy points. Or, X has min-entropy k if the weight of the heaviest point in X is at most $1/2^k$.
Conditional Entropy. $H(Y_1,\ldots,Y_m|Z) = \sum_i H(Y_i|Z,Y_1,\ldots,Y_{i-1})$.

2. Negligible and Noticeable Functions

Remark 2.1. *Polynomially-bounded functions grow at a rate which is slow enough that we consider their growth "reasonable". Negligible functions vanish so quickly that we regard them as "insignificant".*

Definition 2.2 (Negligible Function)**.** Decreases faster than the reciprocal of any positive polynomial. A function $\mu : \mathbb{N} \to \mathbb{R}$ is negligible if

$$\forall c \exists n_0 \forall n \geq n_0 \mu(n) < \frac{1}{n^c}.$$

Definition 2.3 (Noticeable Function)**.** Decreases slower than the reciprocal of some positive polynomial. A function $\mu : \mathbb{N} \to \mathbb{R}$ is noticeable if

$$\exists c \exists n_0 \forall n \geq n_0 \mu(n) \geq \frac{1}{n^c}.$$

2.1. Non-negligible Function is not Necessarily Noticeable.

A non-negligible $\mu : \mathbb{N} \to \mathbb{R}$ satisfies

$$\exists c \forall n_0 \exists n \geq n_0 \mu(n) \geq \frac{1}{n^c}.$$

A noticeable function must satisfy $\mu(n) \geq 1/n^c$ for *any* $n \geq n_0$, but a non-negligible function satisfies it only for *some* $n \geq n_0$.

Example 2.4. The function

$$\mu(n) = \begin{cases} 1/2^n & \text{if } n \text{ is even} \\ 1/2^3 & \text{if } n \text{ is odd} \end{cases}$$

is non-negligible and non-noticeable.

Remark 2.5. *For a negligible function μ and a fixed polynomial p, $\mu'(n) = p(n) \cdot \mu(n)$ is negligible. That is, an event that occurs with negligible probability is highly unlikely to occur even if the experiment is repeated polynomially many times.*

3. Statistical Distance

The statistical distance between two distributions X and Y over Ω is

$$\max_{T \subseteq \Omega} |X(T) - Y(T)|.$$

T can be viewed as a statistical test: $X(T) = \mathbb{P}[X \in T] = \sum_{w \in T} \mathbb{P}[X = w]$.

Remark 3.1. *A distribution ϵ-close to uniform can be used instead of the uniform one with ϵ* damage.

4. Weaker Definitions

Easier to obtain $\equiv$ allow adversary less power $\equiv$ ask adversary hard questions.

Remark 4.1. *Sometimes weaker definition is equivalent to stronger one. For example, next-bit unpredictability is equivalent to pseudorandomness, but a generator can be proven to be pseudorandom by satisfying the weaker definition (easier to obtain) and then concluding that it has all the nice properties of the stronger one (harder to obtain).*

5. One-Way Functions

Any poly-size circuit that attempts to invert f, errs on at least poly-large fraction of the inputs.

Remark 5.1. OWF *implies length-preserving* OWF.

6. Hardcore Bits

That OWF does not necessarily hide partial information about its pre-image limits its *direct applicability* to cryptography. Can we point to partial information about x which are hard to compute? This partial information is the "hard-core" of the difficulty of inverting f.

6.1. **An Injective Function f With Hard-core Predicate b Must Be One Way.** Since f is injective, $f(X)$ hides all information about X. If f is not one way, *"the"* pre-image X can be recovered and the predicate $b(X)$ can be computed. But the argument does not hold for a many-to-one function g: if g is not one way, *"a"* pre-image X' can be recovered from $g(X)$. But if $X' \neq X$, the predicate $b(X)$ cannot necessarily be computed.

7. Pseudorandomness

Deterministic structures that share *some* properties of random ones. Or, efficient generation of objects that *look random* despite being constructed using little or no randomness.

7.1. **P vs NP and (Pseudo)randomness.**

Theorem 7.1 (Shannon)**.** *Random function on n variables require exponential(n) gates.*

P vs. **NP**: Exhibit a hard function! Explicitly find a function that *looks* random from the viewpoint of hardness to compute (e.g., do TSP, Clique, ..., etc, require exponential(n) gates?)

8. Computational Secrecy

8.1. **Tiny Probability.** If security fails with probability 2^{-60}, shall we be concerned? If you're concerned, then there are bigger problems to worry about than failure of your scheme, such as being struck by lightening!

Something that occurs with probability 2^{-60}/sec is expected to occur once every 100 billion years. So if encryption scheme with security 2^{-60} is used to encrypt a message each second, then the scheme will leak information to the attacker only once every 100 billion years.

In practice, $\varepsilon \geq 1/2^{30}$ is non-negligible: likely to happen over 1GB of data. But $\varepsilon \leq 1/2^{80}$ is negligible: won't happen over life of key.

8.2. **Bounded Attackers and Brute Force.** Assume one key is tested per clock cycle.

- Desktop computer $\approx 2^{57}$ keys/year
- Supercomputer $\approx 2^{80}$ keys/year
- Supercomputer since Big Bang $\approx 2^{112}$ keys
- **Modern key space:** 2^{128} keys or more!

So, restricting attention to attackers who can try 2^{112} keys is reasonable. We don't need to worry about attackers who can test 2^{256} keys, simply because there is no way that such an attacker can possibly exist in our physical universe.

Example 8.1 (Asymptotic Computational Security)**.** Consider a scheme where the best attack is brute-force. If A runs in time $t(n)$, then $\mathbb{P}\left[\mathsf{PrivK}_{A,\Pi} = 1\right] \leq 1/2 + t(n)/2^n$. The scheme is indistinguishable: for any polynomial t, the function $t(n)/2^n$ is negligible.

Remark 8.2. *In general, encryption does not hide the plaintext length. So, in indistinguishability experiment, the attacker picks two messages of the same length. Otherwise it would be easy to succeed for trivial reasons.*

9. Block Ciphers

Practical constructions of *candidate* pseudorandom permutations.

No asymptotics: $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^m$, where n is the *key length* and m is the *block length*.

It must be hard to distinguish F_k from uniform $f \in \mathsf{Perm}_m$ *even* for attackers running in time $\approx 2^n$. That is, the best attack in terms of distinguishing F_k from uniform permutation should be an exhaustive key search attack where the attacker tries to see whether any key k makes a match with the input/output that it sees.

9.1. **AES.** Block length is 128 bits, and key length is 128, 192, or 256 bits. Conjectured to be block cipher!

10. Symmetric Encryption

10.1. **High Level View.**

(Theory)	OWF	$\to$	PRP/PRF	$\to$ Encryption, MAC
(90s/00s)	Blockcipher (assumed PRP)	$\to$	Encryption, MAC	$\to$ Authenticated-Encryption
(Contemporary)	Blockcipher (assumed PRP)	$\to$	Tweakable PRP	$\to$ Authenticated-Encryption, FPE

10.2. **Pseudorandom Functions (PRFs).** Examples:

- CBC-mode using suitable block cipher.
- HMAC using suitable hash function.
- Theoretical constructions from any OWF.

10.2.1. *Security.* Concrete security is suitable for practical implementation: proving *low* advantage for *large* resources is translated as providing security.

10.3. **Pseudorandom Permutations (PRPs).** A **block cipher** is just a keyed family of permutations for which enciphering and deciphering are efficient.

- AES.
- 3DES.
- Luby-Rackoff using random functions.

10.4. **PRP/PRF Switching Lemma.** A PRP for large n is already good PRF (and visa versa).

Theorem 10.1. *Fix n. Then for any adversary $\mathcal{A}$ making q oracle queries*

$$\left|\mathbb{P}\left[\mathsf{PRP0}_n^{\mathcal{A}} \to 1\right] - \mathbb{P}\left[\mathsf{PRF0}_n^{\mathcal{A}} \to 1\right]\right| \leq \frac{q^2}{2^n} \tag{10.1}$$

Loosely speaking, we can think of good block ciphers as either random functions or random permutations (when secretly keyed).

10.5. **PRPs from PRFs.**

- Cool theory (shuffling viewpoint on block ciphers).
- Tweakable PRPs trivial given good PRP-from-PRF construction.
- check this one from the video...

11. Non-Uniform Polynomial Time

A stronger (and actually unrealistic) model of efficient computation that is mainly used for the negative way, namely, for saying that *even* such machines cannot do something. Specifically, even if the adversary employs such a machine, it cannot cause harm.

12. Non-Deterministic Machines

An unrealistic model that is useful for talking about languages in **NP**.

13. On Cryptography and Hard Instances

(The introduction of "Generating Hard Instances of Lattice Problems" by Ajtai)
(The Introduction of "Perfect Structure on the Edge of Chaos" by Bitansky, Paneth, and Wichs)

14. Pseudoentropy

Definition 14.1 (Pseudoentropy)**.** X has pseudoentropy at least k if it is computationally indistinguishable from Y with $H(Y) \geq k$.

Remark 14.2. *Consider $A \sim U_n$. Then, $H(A) = n$, and by definition, A has pseudoentropy at least n. In fact, by definition, any X has pseudoentropy at least $H(X)$. We can take Y to be a separate, independent random variable that has the same distribution as X. But the interesting case is when the pseudoentropy of X is* strictly *larger than $H(X)$. That is, when X has pseudoentropy at least $H(X) + \Delta$, where $\Delta > 0$ is the entropy* gap*. Thus, we can say a random variable X has pseudoentropy if it is computationally indistinguishable from another random variable of* higher *Shannon entropy.*

$A \sim U_n$ has pseudoentropy exactly *n because it is uniform over $\{0,1\}^n$. An example of a random variable with a* gap *between its real and pseudoentropy is the output of a PRG. Let $G : \{0,1\}^n \to \{0,1\}^m$ be a PRG. Then, $H(G(U_n)) \leq n$, but $G(U_n)$ has pseudoentropy $m > n$ by the PRG definition.*

Definition 14.3 (Conditional Pseudoentropy)**.** Let (X, B) be jointly distributed. B has conditional pseudoentropy at least k given X if (X, B) is computationally indistinguishable from (X, C) with $H(C|X) \geq k$.

Remark 14.4. *A useful generalization of pseudoentropy. If B has pseudoentropy at least k given X, then (X, B) has pseudoentropy at least $H(X) + k$. (The pseudoentropy of X, which is at least $H(X)$, plus the pseudoentropy of B given X, which is given to be at least k.) But the converse is not necessarily true: consider X having pseudoentropy at least $H(X) + k$ on its own, with B completely determined by X.*

Definition 14.5 (Next-Block Pseudoentropy)**.** $X = (X_1, \ldots, X_m)$ has next-block pseudoentropy at least k if $\exists Y = (Y_1, \ldots, Y_m)$, jointly distributed with X, such that:

(1) $(X_1, X_2, \ldots, X_{i-1}, X_i) \equiv_c (X_1, X_2, \ldots, X_{i-1}, Y_i), \forall i$.
(2) $\sum_i H(Y_i|X_1, \ldots, X_{i-1}) \geq k$.

Remark 14.6. *How hard is it to predict X_i from $X_1, X_2, \ldots, X_{i-1}$ (for $i \leftarrow [m]$). Next-bit pseudoentropy is a special case where each block is one bit. This captures peudoentropy from the perspective of an online adversary who gets the bits one at a time from left to right, instead of getting them all at once. This is a relaxation of next-bit* unpredictability*. It is known that*

$$(X_1, \ldots, X_n) \equiv_c (R_1, \ldots, R_n) \iff (X_1, X_2, \ldots, X_{i-1}, X_i) \equiv_c (X_1, X_2, \ldots, X_{i-1}, R_i)\, \forall i,$$

where R_i's are uniform independent bits. So, instead of X_i being pseudorandom *given the previous bits, this definition requires that X_i has* pseudoentropy *given the previous bits.*

The next-bit pseudoentropy of a random variable can be much larger than its pseudoentropy. If $G : \{0,1\}^n \to \{0,1\}^m$ is a PRG, then $\langle G(U_n), U_n \rangle$ has next-bit pseudoentropy at least $m > n$, but has pseudoentropy n. This is because the online adversary is weaker and hence easier to fool.

The case of one block ($m = 1$) amounts to the definition of a pseudoentropy.

14.1. Pseudoentropy from OWF.

Theorem 14.7 (HILL)**.** *$W = \langle f(X), H, H(X)_1, \ldots, H(X)_J \rangle$ has pseudoentropy $\geq H(W) + \omega(\log n)/n$, where $H : \{0,1\}^n \to \{0,1\}^n$ is a certain kind of hash function, $X \leftarrow \{0,1\}^n$, and $J \leftarrow \{1, \ldots, n\}$.*

Intuition. The choice of J is guessing the log of the number of pre-images of $f(X)$. Or how much entropy there's in X given $f(X)$. If guessed correctly, the first few bits will be statistically close to uniform by the Leftover Hash Lemma (we're hashing out the information-theoretic randomness in X). Taking J to be $\log n$ bits larger, then by Goldreich-Levin theorem (hashing X) we *hope* to get $\log n$ pseudorandom bits from the computational hardness of finding any pre-image of $f(X)$. You loose $1/n$ because of guessing how many pre-images there are (the guess of J). There's only one value of small number of values ($\log n/n$) where you get a gap (where you get the computational entropy).

14.2. VZ Construction.

The goal is to show $\langle f(U_n), U_n \rangle$ is a next-bit pseudoentropy generator. Let (X, B) be jointly distributed. Intuitively,

B has high **pseudoentropy** *given X iff B is hard to* **predict** *from X.*

But one difficulty in characterizing this intuition is that *pseudoentropy* and *unpredictability* may occur for information-theoretic reasons, i.e., $H(B|X) > 0$. Consider $B \sim U_1$ that is independent of X. Then

(1) B has 1 bit of **pseudoentropy** given X[1]
(2) B cannot be **predicted** better than random guessing form X.

This characterization doesn't separate information-theoretic and computational hardness. So, how to characterize the computational randomness in B?

(1) For **pseudoentropy**, simply subtract $H(B|X)$. (So this side of characterization is easy.)
(2) For **unpredictability**, consider feasibility of **sampling** $B|_{X=x}$ given a sample $x \sim X$.

Then,

(1) **Pseudoentropy** becomes 0.
(2) **Sampling** is easy (it's the uniform distribution), in contrast to predicting B from X.

Now, this alternative characterization separates information-theoretic from computational hardness. Using that, VZ prove the following characterization of pseudoentropy:

Pseudoentropy *and* **hardness of sampling** *are equivalent.*

15. Probability Ensemble

A collection of probability distributions $\{\pi_k\}_{k\in K}$ where for every $k \in K$, π_k is distributed on $\{0,1\}^k$.

A PPT test T distinguishes between two probability ensembles $\{\pi_k\}$ and $\{\pi'_k\}$ iff

$$\left| \mathop{\mathbb{P}}_{x\sim\pi_k}[T(x)=1] - \mathop{\mathbb{P}}_{x\sim\pi'_k}[T(x)=1] \right| > \frac{1}{k^c},$$

for some $c > 0$ and infinitely many k's.

Remark 15.1. *The test T above is* uniform. *The same c for all k's. Indistinguishability can be defined by requiring that probability ensembles be indistinguishable by* nonuniform *polynomial test.*

16. Pseudorandomness

16.1. **Kolmogorov-randomness.** A string s is Kolmogorov-random if $|s|$ equals the length of the shortest sampler of s. Hence, a string s is considered Kolmogorov-random if it does *not* possess a *simple* explanation.

But, it can not be determined whether or not a given string is Kolmogorov-random.

16.2. **Two Approaches.** The Kolmogorov-randomness approach is *ontological*. The pseudorandomness approach is *behavioristic*. (Instead of considering the *explanation* for a phenomenon, consider the phenomenon's effect on the environment.)

17. k-wise Independence

A probability distribution over $\{0,1\}^n$ is k-wise independent if its restriction to any k coordinates is uniform. Such distribution looks uniform *locally* to an observer of only k coordinates, despite possibly being far from uniform *globally*.

17.1. **Min-entropy.** k-wise independence of a distribution implies that the distribution has min-entropy at least k. It's easy to come up with a distribution that has min-entropy k but is nowhere near k-wise independent.

18. Uniform vs Non-uniform Machines

Recall $\mathbf{BPP} \in \mathbf{P}/_{\text{poly}}$. So, anything a *probabilistic polynomial-time* machine can do, a *non-uniform polynomial-time* machine can also do. Thus, non-uniform adversaries are stronger than probabilistic polynomial-time ones. It is not clear whether adversaries should be modeled as probabilistic polynomial-time or non-uniform polynomial-time (or whether this makes any difference). The tradeoff however is clear: security guarantees against non-uniform adversaries are stronger, but almost always rely on stronger hardness assumptions. Also, proofs of security for probabilistic polynomial-time adversaries hold also for non-uniform polynomial-time adversaries. So, *uniform* proofs of security are preferable, where they are known.

[1]It also has 1 bit of Shannon entropy. As mentioned previously, a random variable has pseudoentropy equals to its real entropy, just by definition. Generally, pseudoentropy can be larger than real entropy, but B here is the uniform distribution over 1-bit strings, and so B can't have larger pseudoentropy than the length of 1-bit string.

18.1. **Definitions.** Uniform: $\mathcal{A}(1^n)$ for any PPT $\mathcal{A}$. Non-uniform: $\mathcal{C}_n(1^n)$ for any polynomial-size family of circuits $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$.

19. Why Attacker Success Represented By Negligible

For example, OWF requires that for every PPT adversary $\mathcal{A}$, every positive polynomial, and all sufficiently large n, the probability of inverting the OWF is less than $1/p(n)$. One way to read that is: it is useless to apply $\mathcal{A}$ for polynomially many times to guess the inverse.

20. BPP = P?

One way to resolve this question in the positive is first to show that the number of random bits for *any* **BPP** algorithm can be reduced from poly(n) to $O(\log n)$, and then eliminate the randomness entirely by *enumeration*. This can be achieved by having a *deterministic* function G that stretches a seed of $O(\log n)$ truly random bits into poly(n) *pseudorandom* bits.

21. Kolmogorov Complexity

A string x *looks random* if it is incompressible, i.e., cannot be generated by a Turning machine with a description of length less than $|x|$. An appealing aspect of this notion is that it makes sense of randomness in a *fixed* string, rather than a distribution. Unfortunately, it does not give us what we need. If a function G is computable, then all of its outputs have Kolmogorov complexity $O(\log n)$. (Just hardwire the seed into the TM computing G.) So the outputs are very compressible.

22. Computational Indistinguishability

One perspective comes from the definition of statistical distance

$$\Delta(X,Y) \triangleq \max_T \left|\mathbb{P}\left[X \in T\right] - \mathbb{P}\left[Y \in T\right]\right|.$$

With computational indistinguishability, the max is restricted to be taken only over *efficient* statistical tests T, i.e., those for which membership can be efficiently tested.

23. Computing Distributions

Given a source of random bits (coins), output a sample with a given distribution D.

Example 23.1. Compute $D = \left\{(\text{"0"}, \frac{1}{2}), (\text{"1"}, \frac{1}{4}), (\text{"2"}, \frac{1}{4})\right\}$ via a random walk on a binary tree using at most two coins.

24. Distributional One-Way Function

A OWF is easy to compute but hard to invert even on average.

When the OWF is a permutation, inversion is unambiguous: given $f(x)$ the inverter must find x.

When f is many-to-one, the inversion meaning is less clear: if $f(x) = x'$ where x' is all but the last bit of x, no inverter succeeds with probability greater than $1/2$.

The standard definition of OWF handles this problem by having the inverter succeed if it finds any y such that $f(y) = f(x)$.

However, this allows inverters that might never guess the specific x used to generate $f(x)$. For example, let $g(x)$ be any OWP and define $f(x,y)$ to be x if $y = 0$ and $g(x)$ otherwise. Then on input $z = f(x,y)$, the algorithm that outputs $(z,0)$ always finds some inverse of f but an atypical one. The f defined has a lot of "one-wayness" via g although it is not at all one-way by the standard definition.

A *distributional one-way function* is a weaker notion that captures the "one-wayness" of this f: f is distributionally one-way if given $f(x)$ it is hard to randomly and uniformly generate a preimage of $f(x)$.

Remark 24.1. *The standard definition of* OWF *makes it hard to achieve one-wayness even with a seemingly hard construction. So it's stronger, or harder to achieve. The distributional one-wayness is a weaker requirement. It is easier to achieve, since it puts more restrictions on the inverter.*

In general, the more restrictions you put on the adversary, the weaker is your notion of security and hence the easier it is to achieve but it does not withstand strong (unrestricted) adversaries. If you define the inverter task as catching the sun, then any *function achieves whatever you dream of achieving.*