Winter 2018 CS 485/585 Introduction to Cryptography

Lecture 4

Portland State University — *Jan. 18, 2018*

Lecturer: Fang Song

Draft note. Version: January 25, 2018. Email fang.song@pdx.edu for comments and corrections.

Agenda

- (Last time) Computational Secrecy;
- OTP-PRG: Proof by Reduction
- Block cipher

Logistics

- Quiz 1 coming Tuesday. Covers all lectures so far, but mostly on last three.
- Note card soliciting questions. Q& A last 20 minutes.

Proof of PRG-OTP

Recall the scheme, definitions of PRG and comp. secrecy.

Proof. Assume that there is an PPT adversary $\mathcal{A}$ such that

$$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1/2 + \epsilon(n), \text{ for } \epsilon(n) \geq 1/p(n)\,,$$

some polynomial p. We construct a distinguisher

Distinguisher D: Given input string $w \in \{0,1\}^{\ell(n)}$

1. Run $\mathcal{A}(1^n)$ to obtain a pair of messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$
2. Choose $b \leftarrow \{0,1\}$. Set $c := w \oplus m_b$.
3. Give c to $\mathcal{A}$ and obtain b'. Output 1 if $b' = b$, and 0 otherwise.

We will demontrate that

$$\left|\Pr[D(w) = 1 : w \leftarrow \{0,1\}^{\ell(n)}] - \Pr[D(w) = 1 : w = G(s), s \leftarrow \{0,1\}^n]\right| \geq \epsilon(n)\,,$$

which shows a contradiction since it is supposed to be negligible assuming G is a PRG.

- $w \leftarrow \{0,1\}^{\ell(n)}$. If we look at what $\mathcal{A}$ sees in the reduction, the challenge cipher $c := w \oplus m_b$ is exactly what one gets from OTP with uniformly (long) random key w. We know that $\Pr[b' = b] = 1/2$ by the perfect secrecy of OTP, and therefore $\Pr[D(w) = 1 : w \leftarrow \{0,1\}^{\ell(n)}] = 1/2$.
- $w = G(s), s \leftarrow \{0,1\}^n$. Then $\mathcal{A}$ is playing exaclty the indistinguishablity game for Π, and hence $\Pr[D(w) = 1 : w = G(s), s \leftarrow \{0,1\}^n] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] = 1/2 + \epsilon(n)$.

  Therefore,

$$\left|\Pr[D(w) = 1 : w \leftarrow \{0,1\}^{\ell(n)}] - \Pr[D(w) = 1 : w = G(s), s \leftarrow \{0,1\}^n]\right| = \epsilon(n) \geq 1/p(n)\,.$$

□

A few remarks:

- A stream cipher is basically a PRG but runs in a continuous mode. We do not get into the details, and just identify stream ciphers with PRGs.
- We will often say an output string of a PRG a pseudorandom string, although this is totally nonsense. Similarly, it makes no sense to say any fixed string is "random". Rather, we mean that the string is generated according to some distribution (pseudorandom or uniformly random).
- Some basic questions: given a PRG with 1 bit surplus, how to get more? (Think about it, and we will come back to it in the future)

Constructions of Stream ciphers.

- In theory. A beautiful theory. One-way functions (OWF) to PRG (will see).
- In practice. Block cipher (counter mode) recommended for security. But efficient ad hoc designs exist. Linear-feedback shift register (LFSR); a popular one: RC4 by Ron Rivest: WEP (Wired Equivalent Privacy), WPA (Wireless Protected Access), vulnerability of RC4 renders WEP completely broken.

Block ciphers

Now we come to the other main type of private-key ciphers: *block ciphers*. Block ciphers are the work horse in cryptography (espeically in private-key cryptography). Many examples will come soon.

The abstraction of block ciphers is the so called *pseudorandom permutations* (PRPs). Without the restriction of being permutations, we have the core primitive called *pseudorandom functions* (PRFs). PRF is

a generaliztion of PRGs. Instead of considering "random-looking" strings, we cosider "random-looking" functions. Roughly, it means that it behaves the same as a truly random function, at least as far as an efficient observer is concerned. Once again, it makes no sense to say any *fixed* function is pseudorandom (or random in any sense). It refers to a *distribution* on functions. But before we talk about pseudorandom functions, let's be clear what we mean by a *truly random function*.

Random functions. Consider $\mathcal{F} := \{f : \{0,1\}^n \to \{0,1\}^n\}$ be the collection of all possible functions that maps n-bit strings to n-bit strings. Similar to what we mean by a random string, a random function is just a sample from $\mathcal{F}$ uniformly at random. How many are there? Well if we think about the truth table of a function $f : \{0,1\}^n \to \{0,1\}^n$, for each input string x there are 2^n possible output strings we can map it to. Therefore

$$|\mathcal{F}| = \underbrace{2^n \cdot 2^n \cdot \ldots \cdot 2^n}_{2^n \text{ times}} = 2^{n2^n}.$$

This means you need $\Omega(\log |\mathcal{F}|) = \Omega(n2^n)$ bits to sample or write down the description of a random function.

There is a more intuitive and operational perspective of thinking about a random function. Imagine yourself as a machine implementing a random function, when an input x comes, you can just sample a $y \leftarrow \{0,1\}^n$ uniformly at random. The only thing you need to make sure is maintain consistency, i.e., give the same answer if an input comes again. This can be done by keeping a lookup table for instance. This sample "on-the-fly" viewpoint will be useful for a lot of the analysis in the future.

Pseudorandom functions. So how do we define random-looking functions? We consider a collection of functions $F := \{f_k : \{0,1\}^n \to \{0,1\}^n\} \subseteq \mathcal{F}$ indexed by $k \in \mathcal{K}$. We assume $\mathcal{K} = \{0,1\}^n$ too for simplicity. [1] F (a set of functions) can be equally viewed as a keyed function $F : (k, x) \mapsto f_k(x)$.

[1] Here F is length-preserving. But in general the domain and codomain need not be the same.

$$F : \{0,1\}^{\ell_{\text{key}}(n)} \times \{0,1\}^{\ell_{\text{in}}(n)} \to \{0,1\}^{\ell_{\text{out}}(n)}.$$

- Security parameter n.
- $\ell_{\text{key}}(n), \ell_{\text{in}}(n), \ell_{\text{out}}(n)$: the key length, input length and output length.
- **length-preserving**: assuming $\ell_{\text{key}}(n) = \ell_{\text{in}}(n) = \ell_{\text{out}}(n) = n$.
- **efficient**: for each k, there is an deterministic algorithm that on input x computes $F(k, x)$ in polynomial-time.

- Notation. For each k, $F_k := F(k, \cdot)$ determines a function $\{0,1\}^n \to \{0,1\}^n$

How do we formalize "random-looking" for a function? How about requiring no efficient distinguisher can tell apart f_k for a random k from a truly random function $f \leftarrow \mathcal{F}$? But how do we specify the function to a distinguisher D? It's just too long ($\Omega(n2^n)$) to write down an arbitrary function in $\mathcal{F}$, and a poly-time D can only see a tiny piece of it.

Instead we consider giving a distinguisher *oracle* access of function f. $D^{f(\cdot)}$ means a oracle algorithm, where D can make multiple queries x_i and get responses $y_i := f(x_i)$ during its execution.

Definition 1. Let $F = \{f_k\} \subseteq \mathcal{F}$, F is a pseudorandom function if

- F is efficient, i.e., $f_k(x)$ can be computed by a deterministic polynomial-time (in n) algorithm.
- for any PPT distinguisher D,

$$\left|\Pr[D^{f_k(\cdot)}(1^n) = 1 : k \leftarrow \{0,1\}^n] - \Pr[D^{f(\cdot)}(1^n) = 1 : f \leftarrow \mathcal{F}]\right| \leq \mathrm{negl}(n)\,,$$

Discussion on the definition.

- $\mathcal{O}(\cdot)$ usually denotes an unspecified oracle.
- k is unknown to distinguisher, otherwise trivial to distinguish. PPT D can make poly-many queries to $\mathcal{O}(\cdot)$ at most.
- PRFs against unbounded adversaries *impossible*.
- An insecure example: $F_k(x) = k \oplus x$.

Pesudorandom permutations. Pseudorandom permutations are a special case of PRFs. Let $\Pi := \{\pi \in S_{\{0,1\}^n}\}$ be the set of permutations on n-bit strings. We can define efficient, length-preserving, keyed *permutations* $F = \{f_k\} \subseteq \Pi$ similarly, and we write $f_k^{-1}(\cdot)$ as its inverse permutation[2]. We define PRP as one that is indistinguishable from a truly random permutation (of which you should be able to figure out the meaning).

[2] Here we require that f_k and f_k^{-1} both can be computed efficiently.

Definition 2. Let F be an efficient, length-preserving, keyed *permutation*, F is a pseudorandom permutation if for any PPT distinguisher D,

$$\left|\Pr[D^{f_k(\cdot)}(1^n) = 1 : k \leftarrow \{0,1\}^n] - \Pr[D^{\pi(\cdot)}(1^n) = 1 : \pi \leftarrow \Pi]\right| \leq \mathrm{negl}(n)\,,$$

We said earlier PRPs are a special case of PRF. But is this indeed the case? Is a PRP necessarily a PRF? (WHY NON-TRIVIAL?) The answer is yes as long as the domain of PRP is sufficiently large (super-polynomial).

Proposition 3 ([KL: KL-Prop.3.17]). *If F is a PRP and $\ell_{\text{in}}(n) \geq n$, then F is also a PRF.*

The critical observation is that a random permutation and a random function are identical unless a collision occurs in the random function. But how many samples are needed to see a collision on average? **FS NOTE**: Exercise

The Proposition relies on the following claim.

Lemma 4. *For any D making at most q queries*

$$\left|\Pr[D^{f(\cdot)}(1^n) = 1 : f \leftarrow \mathcal{F}] - \Pr[D^{\pi(\cdot)}(1^n) = 1 : \pi \leftarrow \Pi]\right| \leq O(q^2/2^n)\,.$$

Intuitively, this holds because as long as you don't see a *collision*, i.e. $x \neq x'$ with $f(x) = f(x')$, a RF and RP behave identically. We will defer further discussion when we study hash functions.

Block ciphers are secure instances of PRPs of some fixed key-length (e.g., 128-bit). Why call it a block cipher? It works on data blocks of a fixed length, called the block-length, say 128-bits.

Immediate applications of PRFs/PRPs

- PRG from PRF (PRP). Let $F = \{f_k\}$ be a PRF or PRP.

  $$G(s) := f_s(0), f_s(1), \ldots, f_s(k)\,.$$

  This is a secure PRG (HW 2 [KL:]). As an immediate consequence, we can implement a stream cipher from a block cipher (The resulting cipher is sometimes called *deterministic counter mode* if it is implemented by a block cipher). How about the converse? PRF from PRG? Future lecture, stay tuned!

- A computationally secret encryption from PRP[3]: (G, E, D)
  - $G : k \leftarrow \{0,1\}^n$.
  - $E : c := F_k(m)$.
  - $D : m := F_k^{-1}(c)$.

[3] Exercise: prove it

Constructions of Block ciphers

- In theory: beautiful constructions from (OWF to) PRG to PRF to PRP.
- In practice: next time.