Lecture 2: Fundamental Primitives

MIT - 6.893

Fall 2020

Henry Corrigan-Gibbs

Agenda

- Recap: Merkle puzzles & logistics ← video policies, Zoom
- Pseudorandom functions (PRFs)
  - ↳ eff & negl fns
- Pseudorandom generators (PRGs)
- One-way functions
- Stretch break
- Relations between them
  - ↳ Switching Lemma
  - ↳ GGM tree

Reminder:

HW 1 out now.
Due 9/18 @ 5pm
via Gradescope

Anonymous feedback:

Keep it coming!
Also, send me your favorite music to listen to while studying... I'll post links on Piazza.

Recap: Merkle Puzzles

Will start every lecture with one.

- Key exchange from hash fns with a "quadratic gap" between honest-party cost and attacker cost.

We use key ex. protocols with "exponential gap" (e.g. DH in suitable elliptic curve group)

Shared hash fn:

$$H : [n^2] \to \{0,1\}^n$$

Alice		Bob
$a_1, \ldots, a_n \xleftarrow{R} [n^2]$	$\xrightarrow{H(a_1), \ldots, H(a_n)}$	$b_1, \ldots, b_n \xleftarrow{R} [n^2]$
Find i, j s.t.	$\xleftarrow{H(b_1), \ldots, H(b_n)}$	
$H(a_i) = H(b_j)$		
$\downarrow$		$\downarrow$
$\boxed{a_i}$		$\boxed{b_j}$

1. Correct: * H has no collisions w.h.p.
   * A & B's hashes intersect w/ constant prob ("Birthday Bound")

2. Security: in random-oracle model.

Fundamental Primitives

- Make sure that we're all on the same page

Different crypto classes are different

- These ideas are really useful...
  come up again and again.

Primitives

- OWF : One-way function
[OWP : One-way permutation]
- PRG : Pseudorandom generator
- PRF : " function
- PRP : " permutation

Important take-aways:

Not exactly true (Merkle sigs, etc.)

- These are all "symmetric-key primitives"
- Can construct one from another (except OWP) with only polynomial loss in eff. security.

⇒ From a theoretical perspective, assuming OWF ≡ Assuming PRF

Pseudorandom functions (PRFs)

(Goldreich, Goldwasser, Micali '84, ...)

A "efficient" keyed function that "looks like" a random function to adversaries who don't know the key

$$F: \underbrace{\mathcal{K}}_{\text{key space}} \times \underbrace{\mathcal{X}}_{\text{domain}} \to \underbrace{\mathcal{Y}}_{\text{range}}$$

What does this mean formally?

Recurring theme: importance of definitions to crypto.

For an algorithm A and bit $b \in \{0,1\}$, let W_b be event that A outputs 1 in the following interaction.

Challenger $(b \in \{0,1\})$

if $b=0$: $k \xleftarrow{R} \mathcal{K}$

$b=1$: $f \xleftarrow{R} \mathrm{Funs}[\mathcal{X}, \mathcal{Y}]$

if $b=0$: $y \leftarrow F(k,x)$

$b=1$: $y \leftarrow f(x)$

Adversary

$\xleftarrow{x \in \mathcal{X}}$

$\xrightarrow{y}$

"evaluation queries"

$b' \in \{0,1\}$

Then define A's advantage in attacking PRF F as:

$$\text{PRFAdv}[A, F] := \left| \Pr[W_0] - \Pr[W_1] \right|$$

Then we say that F is a secure PRF if $\forall$ efficient algorithms A,

$$\text{PRFAdv}[A, F] < \text{"negligible"}$$

→ Need to define efficient and negligible.

Efficient and Negligible

security parameter

One view: ("asymptotic")

efficient $\equiv$ poly(λ)

too small to matter $\equiv$ negl(λ)

A fn $f(\lambda)$ is "negligible" if its inverse grows faster than any fixed polynomial.
e.g., $2^{-\lambda}$, $2^{-\sqrt{\lambda}}$, $\lambda^{-\log \lambda}$, $\lambda^{-\sqrt{\log\log \lambda}}$, ...

$\Rightarrow$ If decryption of an n-bit msg takes n^{100} time, still "efficient"

We are used to this from standard algorithms/complexity.

\+ Simplifies analysis
Allows us to ignore low-level impl details

− We care about security & efficiency for specific small choices of sec param λ

→ For this to make sense, we need to parameterize all sets and algs by λ.

$\mathcal{K} \rightarrow \mathcal{K}_\lambda$
$\mathcal{X} \rightarrow \mathcal{X}_\lambda$
$\mathcal{Y} \rightarrow \mathcal{Y}_\lambda$
$A \rightarrow A(1^\lambda)$ or A_λ

Danger! There are lots of subtleties here. See Goldreich for details.

Theory/Practice gap: We need PRF defined $\forall \lambda \in \mathbb{N}$.
But for AES cipher $\lambda \in \{128, 192, 256\}$.

Another view: ("concrete") efficient $\equiv$ < 10 ms

too small to matter $\equiv$ $< 2^{-128}$

\+ Precisely captures running times and attack success prob. for real primitives (AES 128)

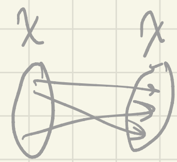
\- Complicates analysis... more quantities to track. Sensitive to model

It is worthwhile to understand and appreciate both perspectives.

(True for most things in life...)

Pseudorandom permutation (PRP)

a.k.a "block cipher"



key space, domain

$$P: \mathcal{K} \times \mathcal{X} \to \mathcal{X}$$

1) **Permutation**

$\forall$ keys $k \in \mathcal{K}$ $P(k, \cdot)$ is a permutation on $\mathcal{X}$

2) **Eff. inversion given key**

There is an efficient alg $P^{-1}: \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ s.t.

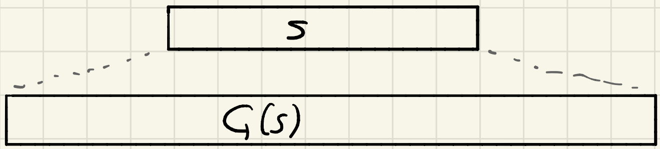
$$\forall k \in \mathcal{K} \quad \forall x \in \mathcal{X} \quad P^{-1}(k, P(k,x)) = x$$

Security is as in PRF game, but adapted to account for these two changes.

Example: AES block cipher.

Pseudorandom generators (PRG)

Recall: Stretch a short seed into a long "random-looking" output.



"security parameter" (think: length of keys)

$s \in \{0,1\}^{\lambda}$

$G(s) \in \{0,1\}^{\ell(\lambda)}$

$[\ell(\lambda) > \lambda]$

Again, need G to be efficient.

Q: How would you define PRGAdv?

Terminology: The "stretch" of a PRG is $\ell(\lambda) - \lambda$. So a PRG with 1-bit stretch maps a λ-bit seed to a $(\lambda+1)$-bit output.

Example: AES in countermode

$$k \longmapsto E(k, \text{"0"}) \,\|\, E(k, \text{"1"}) \,\|\, E(k, \text{"2"})$$

One-way Functions (OWF)

note the hyphen!

An efficient fn $f: \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$ that's "hard to invert" on random inputs.

Could be easy to invert at many points though.

Security: $\forall$ eff algs A

Think about how to specify quantifiers here.

$$\Pr[\underbrace{f(A(f(x))) = f(x)}_{\text{Accounts for the fact that } f(x) \text{ can have many inverses}} : x \xleftarrow{R} \mathcal{X}_\lambda] < \text{negl}(\lambda).$$

Examples:

$f_{SHA256}(x) := SHA256(x)$ $\quad \mathcal{X} = \{0,1\}^{256}$

$f_{factor}((p,q)) := p \cdot q.$ $\quad \mathcal{X} = \{$ pairs (p,q) of λ-bit primes

$\vdots$

One-way permutation (OWF) is a OWF where $f: \mathcal{X} \to \mathcal{X}$ defines a permutation.

Example: $f_{Dlog}(x) := g^x \bmod p$ for suitable $g, p \ldots$

Formalizing this is a little annoying.

→ If you throw a rock, you'll hit a candidate OWF. In contrast, we have very few candidate OWPs.

Stretch Break

Connections

$A \to B$ means that A exists $\Rightarrow$ B exists.

$$\begin{tikzcd}
\text{PRPs} \arrow[r, bend left, "\text{``switching lemma''}"] & \text{PRFs} \arrow[l, bend left, "\text{Luby-Rackoff (``Feistel network'')}"] \arrow[r, bend left, "\text{counter mode}"] & \text{PRGs} \arrow[l, bend left, "\text{GGM tree}"] \arrow[r, bend left, "\text{immediate}"] & \text{OWFs} \arrow[l, bend left, "\text{G.L., HILL, ...}"]
\end{tikzcd}$$

MIT grads

MIT profs

MIT postdoc

MIT grads

Small version on your homework

Surprise 1: Can build signatures (seems like a pub-key primitive) from OWFs.
So think: "pub key crypto" = key exchange and above.

Surprise 2: Good evidence that cannot build collision resistant hashes from OWFs.

Theory vs. Practice:

While these connections are really important for understanding the fundamental power of these primitives, we typically in practice construct whatever we need (PRP, ...) directly under an ad-hoc assumption.

Exception: Accelerated AES hardware gives a incentive to build everything from PRPs.

Remember:

$P = NP \Rightarrow \nexists$ PRGs $\Rightarrow$ None of these primitives exist.

$\exists$ PRGs $\Rightarrow P \neq NP$

Important: We must make assumptions ($P \neq NP$ and it seems much more) to get PRGs, since we don't even yet know whether $P \neq NP$.

Also:

$P \neq NP \overset{?}{\Rightarrow} \exists$ PRGs

We have no idea. See Impagliazzo's "Five Worlds" paper.

From PRP to PRF ("Switching Lemma")

[Boneh · Shoup Thm 4.4]

Idea

A secure PRP is also a secure PRF.

Thm Let P be a PRP with domain $\mathcal{X}$. Then for all eff advs $\mathcal{A}$ making at most Q queries to it challenger:

$$\left|\text{PRPAdv}[\mathcal{A},P] - \text{PRFAdv}[\mathcal{A},P]\right| \leq \frac{Q^2}{|\mathcal{X}|}.$$

$\Rightarrow$ If $\mathcal{A}$ is efficient then $Q = \text{poly}(\lambda)$.

If $\mathcal{X}_\lambda = \{0,1\}^\lambda$ (for example) $\frac{Q^2}{2^\lambda} = \frac{\text{poly}(\lambda)}{2^\lambda} = \text{negl}(\lambda)$.

$\Rightarrow$ Eff adv can't distinguish PRP from PRF.

When the switching lemma breaks down...

* AES is a block cipher with $\mathcal{X} = \{0,1\}^{128}$.
* AES counter mode uses AES as a PRF to generate a pseudorandom "one-time pad"

msg			
$\oplus$			
$E(k,0)$	$E(k,1)$	....	$E(k,L)$
=			
ciphertext			

Problem: If msg length $L \approx 2^{64}$, the switching lemma breaks down. ("Sweet32 attack") 3DES

$\Rightarrow$ Security guarantees break down.

$\Rightarrow$ Not only that but semantic security breaks. AND adv can even recover blocks of msg.

Attack: Ask for encryption of a) random string of length 2^{70}
b) 0s " " "

If ct has repeated blocks $\Rightarrow$ random
no repeated blocks $\Rightarrow$ probably all-zero string.

$\rightarrow$ Example of why concrete view is useful for analyzing security of practical protocols.

From PRGs to PRFs ("GGM Tree")

- Recall PRG $G: \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ stretches a λ-bit seed into a 2λ-bit "random-looking" output.

- A PRF seems much more complicated. A keyed fn

$$F: \underbrace{\{0,1\}^\lambda}_{\text{key}} \times \underbrace{\{0,1\}^\lambda}_{\text{input}} \to \underbrace{\{0,1\}^\lambda}_{\text{output}}$$

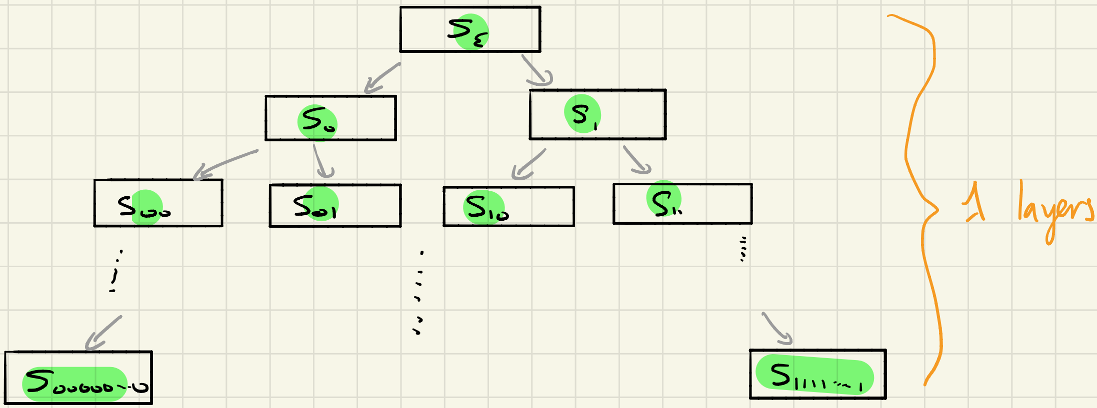
$\Rightarrow$ Not obvious that can build PRF from PRG.

Why discuss this?

1. General crypto literacy.
2. Idea comes up in many recent constructions ("puncturable PRFs", ...)
3. A really nice trick to know.

GGM Tree

Idea: Use PRG to label each node of a depth-λ tree with a λ-bit pseudorandom string.



Labels are defined inductively

s_ε = random λ-bit string (PRF Key)

For any $\sigma \in \bigcup_{i=0}^{\lambda-1} \{0,1\}^i$

$$s_\sigma, \quad (s_{\sigma 0} \| s_{\sigma 1}) \leftarrow G(s_\sigma)$$

Then $F(k, \sigma) :=$ label on node s_σ.

Efficient since can compute labels on any path down to a leaf using λ PRG invocations.

Other properties to notice

- Delegation: Can produce a PRF key $\hat{k}$ that allows evaluating $F(k,x)$ on a inputs x with a certain prefix (e.g. "01")

  Application: Selective decryption

- Puncturing (Sahai, Waters, ...): Can produce a PRF key that allows evaluating $F(k,x)$ at all $x \in \{0,1\}^n$ except a special point x^*.

  Application: Advanced crypto tools (iO) also some applications to PIR

Proof uses a hybrid argument. ← Goldreich Thm 3.6.6.

Idea: * Replace labels at each level of tree by truly random labels (one level at a time.
* Argue that any adv that can distinguish can break PRG security.

$$\left\{\begin{matrix}\text{Real}\\ \text{tree}\end{matrix}\right\} \approx_c \left\{\begin{matrix}\text{Real tree}\\ \text{w/ level 1}\\ \text{labels random}\end{matrix}\right\} \approx_c \left\{\begin{matrix}\text{real tree w/}\\ \text{level 1+2 labels}\\ \text{random}\end{matrix}\right\} \approx_c \cdots \approx_c \left\{\begin{matrix}\text{tree w/}\\ \text{random}\\ \text{labels}\end{matrix}\right\}$$

Only increases adv's advantage by negl amount ..... λ steps can only increase adv by $\lambda \cdot \text{negl}(\lambda) = \text{negl}(\lambda)$ amount.

The End

Extra material on PRGs

Formally $G_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$

$G = \{G_\lambda : \lambda \in \mathbb{N}\}$

"Family" so that G is defined $\forall \lambda$.

Defn A PRG $G = \{G_\lambda : \lambda \in \mathbb{N}\}$ is secure if for all probabilistic algs A running in time $\mathrm{poly}(\lambda)$

$$\Big| \Pr[A(G(s)) = 1 : s \xleftarrow{R} \{0,1\}^\lambda] - \Pr[A(r) = 1 : r \xleftarrow{R} \{0,1\}^{\ell(\lambda)}] \Big| < \mathrm{negl}(\lambda).$$

"negligible"
Inverse grows faster than any fixed polynomial.

Equivalent formalization:

Distributions

$$\{G(s) : s \xleftarrow{R} \{0,1\}^\lambda\} \overset{c}{\approx} \{r : r \xleftarrow{R} \{0,1\}^{\ell(\lambda)}\}$$

Distributions (implicitly parameterized by $\lambda \in \mathbb{N}$) are "computationally indistinguishable."

No $\mathrm{poly}(\lambda)$-time prob alg distinguish them.

Should be a surprise that PRGs even exist.