B504/I538:
Introduction to Cryptography

Spring 2017 • Lecture 14
(2017-02-23)
Assignment 3 is due on Tuesday!
(2017-02-28)
(Last day for help is tomorrow!!)

Ryan Henry
Pseudorandom objects

- Pseudorandom generators
  - A function that “mimics” the uniform random variable on a larger sample space

- Pseudorandom functions
  - A keyed function that “mimics” the uniform random variable on the set of all functions with the same domain/range (when the key is uniformly distributed)

- Pseudorandom permutations
  - A keyed function that “mimics” the random variable on the set of all permutations on its range (when the key is uniformly distributed)
Pseudorandom objects

Given PRGs, PRFs, and (strong) PRPs, we have seen how to build:

1. Stream ciphers
   - just replace the OTP pad with a PRG keystream
2. Block ciphers
   - use a strong PRP (or, sometimes, a PRF) and a mode of operation
3. Hash functions
   - use a PRF in the Davies-Meyer construction to get compression function
   - use Merkle-Damgård construction build hash from compression function
4. MAC schemes
   - use block cipher in NMAC construction or hash function in HMAC construction
Pseudorandom objects

Q: Do PRGs, PRFs, or PRPs really exist?
A: Errr... umm, well... maybe!
   - Proving (or disproving) their existence would require a massive breakthrough in complexity theory
   - Several “candidates” are believed (or, at least, hoped) to be PR* s

- So what now?
  - Study minimal assumptions under which PRGs, PRFs, and PRPs exist.
Minimal primitive: One-way functions

- Intuitively, a one-way function (OWF) is a function that is easy to compute but hard to invert. A “simpler” and (presumably) “easier” definition to satisfy:

Thm (informal): OWFs exist if and only if PRGs, PRFs, and PRPs all exist.
- PRP/PRF switching lemma: Every PRP is also a PRF
- Luby-Rackoff: Given any PRF, it is possible to construct a PRP
- #5 on Assignment 2: Given any PRF, it is possible to construct a PRG
- Goldreich-Levin: Given any PRG, it is possible to construct a PRF

Corollary: OWFs exist if and only if all crypto discussed so far is possible!
OWF inversion game

Challenger (C)

\[ x \in_R \{0, 1\}^s \]
\[ y := f(x) \]

Inverter (A)

Let \( E \) be the event that \( f(x') = y \)

Define A's advantage to be

\[ \text{Adv}^f^{-1}(A) := \Pr[E] \]

Note: The input \( x \) must be uniformly distributed --- it may be easy to invert \( y = f(x) \) for some inputs \( x \)
One-way function

**Defn:** A function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \) is a (strong) one-way function (OWF) if it is

1. **easy to compute:** There exists an efficient algorithm that, on input \( x \in \{0, 1\}^* \), outputs \( f(x) \), and

2. **hard to invert:** For every PPT algorithm \( A \), there exists a negligible function \( \varepsilon : \mathbb{N} \rightarrow \mathbb{R}^+ \) such that \( \text{Adv}^f(A) \leq \varepsilon(s) \).
Recall: Preimage resistance

Let $E$ be the event that $H(k, m) = y$

Define A's advantage to be $\text{Adv}_{\text{preimage}}(A) := \Pr[E]$

- Preimage finding game: hard to invert arbitrary output
- OWF inversion game: hard to invert image of a random input

(Why is this distinction important?)
Primes and composites

- A positive integer \( p \) is a prime if it has no non-trivial divisors.
- A positive integer \( n \) is composite if it is not a prime. Equivalently: \( n \) is composite if it is a product of primes.
- E.g.: 2, 3, 5, 7, and 11 are all primes.

\[
4 = 2^2, \quad 6 = 2 \cdot 3, \quad 8 = 2^3, \quad 9 = 3^2 \text{ and } 10 = 2 \cdot 5 \text{ are all composite.}
\]
Primes and composites

- Which if the following are prime and which are composite?
  - 41 prime!
  - 51 composite! \( (51 = 17 \cdot 3) \)
  - 81 composite! \( (81 = 9^2) \)
  - 551 composite! \( (551 = 29 \cdot 19) \)
  - 24 281 prime!
  - 76 151 composite! \( (76 151 = 271 \cdot 281) \)

Fun fact: It is easy to determine if a number is prime, but it is (apparently) hard to compute its prime factors!
A candidate OWF: multiplication

- Let Primes[s] denote the set of primes less than or equal to $2^s$
- Let $f: \text{Primes}[s] \times \text{Primes}[s] \to \mathbb{N}$ be the multiplication function: $f(p, q) = p \cdot q$
  - Schoolbook algorithm: $O(s^2)$
  - Fast Fourier transform: $O(s \lg n \lg \lg s)$
- Computing the inverse $f^{-1}(p \cdot q)$ requires finding $p$ and $q$
  - Trial division: $O\left(2^{2s}/2s\right)$
  - Quadratic sieve: $O\left(2^{(2s \lg 2s)^{1/2}}\right)$
  - General number field sieve: $O\left(e^{(64/9)(2s)^{1/3} (\lg 2s)^{2/3}}\right)$

each of these has polynomial running time!

each of these has super-polynomial running time!

fastest known algorithm

Ryan Henry
Random inputs versus random outputs

- Recall similarity between preimage resistance game and OWF inversion game
  - Preimage resistance game: Invert an arbitrary output
  - OWF inversion game: Invert a random input

- If we choose $p \in \mathbb{R} \text{Primes}[s]$ followed by $q \in \mathbb{R} \text{Primes}[2s]$ subject to $\lg pq = 2s$, then $\lg p \approx \lg q \approx s$ with very high probability, in which case $f$ appears to be hard to invert.

Q: If we choose $n \in \mathbb{R} \{pq, \ p \text{ and } q \text{ are prime and } \lg pq = 2s\}$, how hard is it hard to compute $f^{-1}(n)$?

A: Not very hard at all (on average)!
Random inputs versus random outputs

Q: If we choose \( n \in \mathbb{R} \{ pq \mid p \text{ and } q \text{ are prime and } \log pq = 2s \} \), how hard is it hard to compute \( f^{-1}(n) \)?

Prime number theorem: The number of primes less than \( 2^s \) is about \( \frac{2^s}{s \cdot \ln 2} \).

- Each number in the above set is the product of a \( k \)-bit prime and a \((2s-k)\)-bit prime.
- About \( \frac{2^{2s-1}((2s-1) \ln 2)}{(2s-1) \ln 2} \) primes have \( 2s-1 \) or fewer bits, and about \( \frac{2^{2s-2}((2s-2) \ln 2)}{(2s-2) \ln 2} \) have \( 2s-2 \) or fewer bits; thus, about half of the possible outputs of \( f \) are divisible by 2 or 3.
- About 99% are of all outputs are divisible by a prime less than 128.
- More generally, a fraction about \( \frac{(2s-k)}{(2s-1)} 2^k \) are divisible by a prime less than \( 2^k \).
**Weak one-way function**

**Defn:** A function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a weak one-way function (weak OWF) if it is

1. **easy to compute:** There exists an efficient algorithm that, on input $x \in \{0, 1\}^*$, outputs $f(x)$, and

2. **occasionally hard to invert:** For every PPT algorithm $A$, there exists a noticeable function $\mu : \mathbb{N} \rightarrow \mathbb{R}^+$ such that $\text{Adv}^{f^{-1}}(A) \geq 1 - \mu(s)$.

**One-way function:** PPT attackers almost always fail to invert

**Weak one-way function:** PPT attackers occasionally fail to invert
Strong OWFs versus weak OWFs

Q: Why should we care about weak OWFs?
A: Weak OWFs are presumably easier to construct.

Thm (informal): If weak OWFs exist, then so do strong OWFs.

Proof (sketch): Assume PPT attacker loses OWF inversion game for $f$ with noticeable probability $\mu(s)$. Construct a new function

$$f'(x_1 \parallel x_2 \parallel \ldots \parallel x_{p(s)}) := f(x_1) \parallel f(x_2) \parallel \ldots \parallel f(x_{p(s)})$$

for some polynomial $p(s) \gg \mu(s)$.

Now apply Chernoff's bound to conclude that $f'$ is a strong OWF.
Existence of OWFs

Q: Do OWFs exist?
A: Nobody knows for sure...

Thm (informal): If OWFs exist, then $P \neq NP$

- $P =$ class of decision problems solvable by PPT algorithms
- $NP =$ class of decision problems with PPT checkable proofs
  - Where $P = NP$ is THE major open problem in computer science
  - Most experts believe $P \neq NP$
  - Clay Mathematics Institute offers $1$ million case prize for a proof
Hard-core predicates

- Strong OWFs are hard to invert in their entirety
- Want to say: \( f(x) \) reveals "nothing" about \( x \)
  
  Q: Do OWFs satisfy this requirement?
  
  A: In general, NO! (But why?)
    - Suppose \( g \) is an OWF, then it is easy to prove that \( f(x_1 || x_2) = x_1 || g(x_2) \) is also an OWF!

- A relaxation: Can we say \( f(x) \) reveals "nothing" about \( h(x) \), for some particular function \( h \) that depends on \( f \) but not \( x \)?
**Hard-core predicates**

- Let $h: \{0, 1\}^* \rightarrow \{0, 1\}$ be an efficiently computable function
  - Think of $h(x)$ as indicating whether $x$ has some property ($h(x)=1$) or not ($h(x)=0$)
- Intuitively, we call $h$ a hard-core predicate for $f$ if $f(x)$ reveals nothing about $h(x)$

Let $E$ be the event that $h(x) = b$

Define A's advantage to be $\text{Adv}^{h,f}(A) := | \Pr[E] - 1/2 |$
Hard-core predicates

**Defn:** Let $f: \{0,1\}^* \to \{0,1\}^*$ and let $h: \{0,1\}^* \to \{0,1\}$ be an efficiently computable Boolean-valued function. Then $h$ is a hard-core predicate for $f$ if, for every PPT algorithm $A$, there exists a negligible function $\varepsilon: \mathbb{N} \to \mathbb{R}^+$ such that $\text{Adv}^h_f(A) \leq \varepsilon(s)$.

- $h$ is easy to compute from $x$ but hard to predict from $f(x)$
- Equivalently: $h(x)$ looks random given $f(x)$
- If $h(x)$ equal some bit of $x$, then we call $h$ a hard-core bit for $f$
Hard-core predicate examples

Let $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be an OWF and define $h(x) = \bigoplus_{i=1}^{||x||} x_i$

Q: Is $h$ a hard-core predicate for $f$?

A: In general, NO! (If $g$ is a OWF, then $f(x) := g(x) \ll \bigoplus_{i=1}^{||x||} x_i$ is an OWF for which $h(x)$ is not hard-core!)

Let $g : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be the function that just “drops” the lsb of its input and define $h(x) := \text{lsb}(x)$

Q: Is $h$ a hard-core predicate for $f$?

A: Yes! (But not a very useful/interesting one...)

Ryan Henry
Goldreich-Levin Theorem

Thm: If there exists an OWF, then there exists a pair of functions \((g, h)\) such that \(g\) is an OWF and \(h\) is a hard-core predicate for \(g\).

Specifically, if \(f\) is an OWF, then the function

\[
g(x \parallel r) := f(x) \parallel r \text{ with } |x| = |r| \text{ is an OWF}
\]

and

\[
h(x) = \bigoplus_{i=1}^{\frac{|x|}{d}} (x_i \cdot r_i)
\]

is a hard-core predicate for \(f\).

Note: Goldreich-Levin does \textit{not} claim that every OWF has a hard-core predicate!
That's all for today, folks!