Notes for Lecture 11

Scribed by Luca, posted March 10, 2009

Summary

Today we begin a tour of the theory of one-way functions and pseudorandomness.

The highlight of the theory is a proof that if *one-way functions* exist (with good asymptotic security) then pseudorandom permutations exist (with good asymptotic security). We have seen that pseudorandom permutations suffice to do encryption and authentication with extravagantly high levels of security (respectively, CCA security and existential unforgeability under chosen message attack), and it is easy to see that if one-way functions do not exist, then every encryption and authentication scheme suffers from a total break.

Thus the conclusion is a strong "dichotomy" result, saying that either cryptography is fundamentally impossible, or extravagantly high security is possible.

Unfortunately the proof of this result involves a rather inefficient reduction, so the concrete parameters for which the dichotomy holds are rather unrealistic. (One would probably end up with a system requiring gigabyte-long keys and days of processing time for each encryption, with the guarantee that if it is not CCA secure then every 128-bit key scheme suffers a total break.) Nonetheless it is one of the great unifying achievements of the asymptotic theory, and it remains possible that a more effective proof will be found.

In this lecture and the next few ones we shall prove the weaker statement that if *one-way permutations* exist then pseudorandom permutations exist. This will be done in a series of four steps each involving reasonable concrete bounds. A number of combinatorial and number-theoretic problems which are believed to be intractable give us highly plausible candidate one-way permutations. Overall, we can show that if any of those well-defined and well-understood problems are hard, then we can get secure encryption and authentication with schemes that are slow but not entirely impractical. If, for example, solving discrete log with a modulus of the order of $2^{1,000}$ is hard, then there is a CCA-secure encryption scheme requiring a $4,000$-bit key and fast enough to carry email, instant messages and probably voice communication. (Though probably too slow to encrypt disk access or video playback.)

1 One-way Functions and One-way Permutations

A one-way function f is a function such that, for a random x, it is hard to find a pre-image of $f(x)$.

Definition 1 (One-way Function) *A function $f : \{0,1\}^n \to \{0,1\}^m$ is (t,ϵ)-one way if for every algorithm A of complexity $\leq t$ we have*

$$\mathop{\mathbb{P}}_{x\sim\{0,1\}^n}[A(f(x)) = x' : f(x) = f(x')] \leq \epsilon$$

In the asymptotic theory, one is interested in one-way functions that are defined for all input lengths and are efficiently computable. Recall that a function $\nu : \mathbb{N} \to \mathbb{R}$ is called *negligible* if for every polynomial p we have $\lim_{n\to\infty} \nu(n)/p(n) = 0$.

Definition 2 (One-way Function – Asymptotic Definition) *A function $f : \{0,1\}^* \to \{0,1\}^*$ is one-way if*

1. *f is polynomial time computable and*
2. *for every polynomial $p()$ there is a negligible function ν such that for all large enough n the function $f_n(x) := (n, f(x))$ is $(t(n), \nu(n))$-one way.*

Example 3 (Subset Sum) *On input $x \in \{0,1\}^n$, where $n = k \cdot (k+1)$, $SS_k(x)$ parses x as a sequence of k integers, each k-bit long, plus a subset $I \subseteq \{1,\ldots,k\}$.*

The output is

$$SS_k(x_1,\ldots,x_k,I) := x_1,\ldots,x_k,\sum_{i\in I} x_i$$

Some variants of subset-sum have been broken, but it is plausible that SS_k is a (t,ϵ)-one way function with t and $1/\epsilon$ super-polynomial in k, maybe even as large as $2^{k^{\Omega(1)}}$.

Exercise 1 *Let $f : \{0,1\}^n \to \{0,1\}^m$ be a (t,ϵ)-secure one-way function. Show that*

$$\frac{t}{\epsilon} \leq O((m+n)\cdot 2^n)$$

Definition 4 (One-way Permutation) *If $f : \{0,1\}^n \to \{0,1\}^n$ is a bijective (t,ϵ)-one way function, then we call f a (t,ϵ)-one-way permutation.*

If f is an (asymptotic) one-way function, and for every n f is a bijection from $\{0,1\}^n$ into $\{0,1\}^n$, then we say that f is an (asymptotic) one-way permutation.

There is a non-trivial general attack against one-way permutations.

Exercise 2 *Let $f : \{0,1\}^n \to \{0,1\}^m$ be a (t,ϵ)-secure one-way permutation. Show that*

$$\frac{t^2}{\epsilon} \leq O((m+n)^2 \cdot 2^n)$$

This means that we should generally expect the input length of a secure one-way permutation to be at least 200 bits or so. (Stronger attacks than the generic one are known for the candidates that we shall consider, and their input length is usually 1000 bits or more.)

Example 5 (Modular Exponentiation) *Let p be a prime, and $\mathbb{Z}_p^*$ be the group whose elements are $\{1,\ldots,p-1\}$ and whose operation is multiplication* mod$p$*. It is a fact (which we shall not prove) that $\mathbb{Z}_p^*$ is cyclic, meaning that there is an element g such that the mapping*

$$EXP_{g,p}(x) := g^x \bmod p$$

is a permutation on $\mathbb{Z}_p^*$. Such an element g is called a generator, and in fact most elements of $\mathbb{Z}_p^*$ are generators. $EXP_{g,p}$ is conjectured to be one-way for most choices of p and g.

The problem of inverting $EXP_{g,p}$ is called the discrete logarithm problem.

The best known algorithm for the discrete logarithm is conjectured to run in time $2^{O((logp)^{1/3})}$. It is plausible that for most p and most g the discrete logarithm is a (t,ϵ) one way permutation with t and ϵ^{-1} of the order of $2^{(\log p)^{\Omega(1)}}$.

Problems like exponentiation do not fit well in the asymptotic definition, because of the extra parameters g, p. (Technically, they do not fit our definitions at all because the input is an element of $\mathbb{Z}_p^*$ instead of a bit string, but this is a fairly trivial issue of data representation.) This leads to the definition of *family of one-way functions (and permutations).*

2 A Preview of What is Ahead

Our proof that a pseudorandom permutation can be constructed from any one-way permutation will proceed via the following steps:

1. We shall prove that for any one-way permutation f we can construct a *hard-core predicate* P, that is a predicate P such that $P(x)$ is easy to compute given x, but it is hard to compute given $f(x)$.

2. From a one-way function with a hard-core predicate, we shall show how to construct a pseudorandom generator with *one-bit expansion*, mapping ℓ bits into $\ell + 1$.

3. From a pseudorandom generator with one-bit expansion, we shall show how to get generators with essentially *arbitrary expansion*.

4. From a length-doubling generator mapping ℓ bits into 2ℓ, we shall show how to get *pseudorandom functions*.

5. For a pseudorandom function, we shall show how to get *pseudorandom permutations*.

3 Hard-Core Predicate

Definition 6 (Hard-Core Predicate) *A boolean function $P : \{0,1\}^n \to \{0,1\}$ is (t, ϵ)-hard core for a permutation $f : \{0,1\}^n \to \{0,1\}^n$ if for every algorithm A of complexity $\leq t$*

$$\mathop{\mathbb{P}}_{x \sim \{0,1\}^n}[A(f(x)) = P(x)] \leq \frac{1}{2} + \epsilon$$

Note that only one-way permutations can have efficiently computable hard-core predicates.

Exercise 3 *Suppose that P is a (t, ϵ)-hard core predicate for a permutation $f : \{0,1\}^n \to \{0,1\}^n$, and P is computable in time r. Show that f is $(t - r, 2\epsilon)$-one way.*

It is known that if $Exp_{g,p}$ is one-way, then every bit of x is hard-core.

Our first theorem will be that a random XOR is hard-core for every one-way permutation.

We will use the following notation for "inner product" modulo 2:

$$\langle x, r \rangle := \sum_i x_i r_i \bmod 2 \tag{1}$$

Theorem 7 (Goldreich and Levin) *Suppose that A is an algorithm of complexity t such that*

$$\mathop{\mathbb{P}}_{x,r}[A(f(x),r) = \langle x, r\rangle] \geq \frac{1}{2} + \epsilon \tag{2}$$

Then there is an algorithm A' of complexity at most $O(t\epsilon^{-2}n^{O(1)})$ such that

$$\mathop{\mathbb{P}}_{x}[A'(f(x)) = x] \geq \Omega(\epsilon)$$

We begin by establishing the following weaker result.

Theorem 8 (Goldreich and Levin – Weak Version) *Suppose that A is an algorithm of complexity t such that*

$$\mathop{\mathbb{P}}_{x,r}[A(f(x),r) = \langle x, r\rangle] \geq \frac{15}{16} \tag{3}$$

Then there is an algorithm A' of complexity at most $O(tn\log n + n^2\log n)$ such that

$$\mathop{\mathbb{P}}_{x}[A'(f(x)) = x] \geq \frac{1}{3}$$

Before getting into the proof of Theorem 8, it is useful to think of the "super-weak" version of the Goldreich-Levin theorem, in which the right-hand-side in (3) is 1. Then inverting f is very easy. Call $e_i \in \{0,1\}^n$ the vector that has 1 in the i-th position and zeroes everywhere else, thus $\langle x, e_i\rangle = x_i$. Now, given $y = f(x)$ and an algorithm A for which the right-hand-side of (3) is 1, we have $x_i = A(y, e_i)$ for every i, and so we can compute x given $f(x)$ via n invocations of A. In order to prove the Goldreich-Levin theorem we will do something similar, but we will have to deal with the fact that we only have an algorithm that approximately computes inner products.

We derive the Weak Goldreich-Levin Theorem from the following reconstruction algorithm.

Lemma 9 (Goldreich-Levin Algorithm – Weak Version) *There is an algorithm GLW that given oracle access to a function $H : \{0,1\}^n \to \{0,1\}^n$ such that, for some $x \in \{0,1\}^n$,*

$$\mathop{\mathbb{P}}_{r\sim\{0,1\}^n}[H(r) = \langle x, r\rangle] \geq \frac{7}{8}$$

runs in time $O(n^2\log n)$, makes $O(\log n)$ queries into H, and with $1-o(1)$ probability outputs x.

Before proving Lemma 9, we need to state the following version of the *Chernoff Bound*.

Lemma 10 (Chernoff Bound) *Let $X_1, \ldots, X_n$ be mutually independent 0/1 random variables. Then, for every $\epsilon > 0$, we have*

$$\mathbb{P}\left[\sum_{i=1}^{n} X_i > \mathbb{E}\sum_{i=1}^{n} X_i + \epsilon n\right] \leq e^{-2\epsilon^2 n} \tag{4}$$

PROOF: We only give a sketch. Let $Y_i := X_i - \mathbb{E}\, X_i$. Then we want to prove that

$$\mathbb{P}\left[\sum_{i} Y_i > \epsilon n\right] \leq e^{-2\epsilon^2 n}$$

For every fixed λ, Markov's inequality gives us

$$\begin{aligned}\mathbb{P}\left[\sum_{i} Y_i > \epsilon n\right] &= \mathbb{P}\left[e^{\lambda \sum_i Y_i} > e^{\lambda \epsilon n}\right] \\ &\leq \frac{\mathbb{E}\, e^{\lambda \sum_i Y_i}}{e^{\lambda \epsilon n}}\end{aligned}$$

We can use independence to write

$$\mathbb{E}\, e^{\lambda \sum_i Y_i} = \prod_i \mathbb{E}\, e^{\lambda Y_i}$$

and some calculus shows that for every Y_i we have

$$\mathbb{E}\, e^{\lambda Y_i} \leq e^{\lambda^2/8}$$

So we get

$$\mathbb{P}\left[\sum_{i} Y_i > \epsilon n\right] \leq e^{\lambda^2 n/8 - \lambda \epsilon n} \tag{5}$$

Equation (5) holds for every $\lambda > 0$, and in particular for $\lambda := 4\epsilon$ giving us

$$\mathbb{P}\left[\sum_{i} Y_i > \epsilon n\right] \leq e^{-2\epsilon^2 n}$$

as desired. □

We can proceed with the design and the analysis of the algorithm of Lemma 9.

PROOF:[Of Lemma 9] The idea of the algorithm is that we would like to compute $\langle x, e_i \rangle$ for $i = 1, \ldots, n$, but we cannot do so by simply evaluating $H(e_i)$, because it is entirely possible that H is incorrect on those inputs. If, however, we were just interested in computing $\langle x, r \rangle$ for a random r, then we would be in good shape, because $H(r)$ would be correct with resonably large probability. We thus want to *reduce the task of computing $\langle x, y \rangle$ on a specific y, to the task of computing $\langle x, r \rangle$ for a random r.* We can do so by observing the following identity: for every y and every r, we have

$$\langle x, y \rangle = \langle x, r + y \rangle - \langle x, r \rangle$$

where all operations are mod 2. (And bit-wise, when involving vectors.) So, in order to compute $\langle x, y \rangle$ we can pick a random r, and then compute $H(r + y) - H(r)$. If r is uniformly distributed, then $H(r + y)$ and $H(r)$ are uniformly distributed, and we have

$$\begin{aligned}
& \mathbb{P}_{r \sim \{0,1\}^n} [H(r + y) - H(r) = \langle x, y \rangle] \\
\geq & \mathbb{P}_{r \sim \{0,1\}^n} [H(r + y) = \langle x, r + y \rangle \wedge H(r) = \langle x, r \rangle] \\
= & 1 - \mathbb{P}_{r \sim \{0,1\}^n} [H(r + y) \neq \langle x, r + y \rangle \vee H(r) \neq \langle x, r \rangle] \\
\geq & 1 - \mathbb{P}_{r \sim \{0,1\}^n} [H(r + y) \neq \langle x, r + y \rangle] - \mathbb{P}_{r \sim \{0,1\}^n} [H(r) \neq \langle x, r \rangle] \\
\geq & \frac{3}{4}
\end{aligned}$$

Suppose now that we pick independently several random vectors $r_1, \ldots, r_k$, and that we compute $Y_j := H(r_j + y) - H(r_j)$ for $j = 1, \ldots, k$ and we take the majority value of the Y_j as our estimate for $\langle x, y \rangle$. By the above analysis, each Y_j equals $\langle x, y \rangle$ with probability at least $3/4$; furthermore, the events $Y_j = \langle x, y \rangle$ are mutually independent. We can then invoke the Chernoff bound to deduce that the probability that the majority value is wrong is at most $e^{-k/8}$. (If the majority vote of the Y_j is wrong, it means that at least $k/2$ or the Y_j are wrong, even though the expected number of wrong ones is at most $k/4$, implying a deviation of $k/4$ from the expectation; we can invoke the Chernoff bound with $\epsilon = 1/4$.) The algorithm GLW is thus as follows:

- Algorithm GLW
- for $i := 1$ to n
  - for $j := 1$ to $16 \log n$

    * pick a random $r_j \in \{0,1\}^n$
  - $x_i := \text{majority}\{H(r_j + e_i) - H(e_i) : j = 1, \ldots, 16 \log n\}$
- return x

For every i, the probability fails to compute $\langle x, e_i \rangle = x_i$ is at most $e^{-2 \log n} = 1/n^2$. So the probability that the algorithm fails to return x is at most $1/n = o(1)$. The algorithm takes time $O(n^2 \log n)$ and makes $32n \log n$ oracle queries into H. $\square$

In order to derive Theorem 8 from Lemma 9 we will need the following variant of Markov's inequality.

Lemma 11 *Let X be a discrete bounded non-negative random variable ranging over $[0,1]$. Then for every $0 \leq t \leq \mathbb{E}\, X$,*

$$\mathbb{P}[X \geq t] \geq \frac{\mathbb{E}\, X - t}{1 - t} \tag{6}$$

PROOF: Let R be the set of values taken by X with non-zero probability. Then

$$\begin{aligned}
\mathbb{E}\, X &= \sum_{v \in R} v \cdot \mathbb{P}[X = v] \\
&= \sum_{v \in R : v < t} v \cdot \mathbb{P}[X = v] + \sum_{v \in R : v \geq t} v \cdot \mathbb{P}[X = v] \\
&\leq \sum_{v \in R : v < t} t \cdot \mathbb{P}[X = v] + \sum_{v \in R : v \geq t} 1 \cdot \mathbb{P}[X = v] \\
&= t \cdot \mathbb{P}[X < t] + \mathbb{P}[X \geq t] \\
&= t - t \cdot \mathbb{P}[X \geq t] + \mathbb{P}[X \geq t]
\end{aligned}$$

So we have $\mathbb{P}[X \geq t] \cdot (1 - t) \geq \mathbb{E}\, X - t$. $\square$

We can now prove Theorem 8.

PROOF:[Of Theorem 8] The assumption of the Theorem can be rewritten as

$$\mathop{\mathbb{E}}_{x} \left[\mathop{\mathbb{P}}_{r} [A(f(x), r) = \langle x, r \rangle] \right] \geq \frac{15}{16}$$

From Lemma 11 we have

$$\mathop{\mathbb{P}}_{x} \left[\mathop{\mathbb{P}}_{r} [A(f(x), r) = \langle x, r \rangle] \geq \frac{7}{8} \right] \geq \frac{1}{2}$$

Call an x "good" if it satisfies $\mathbb{P}_r[A(f(x), r) = \langle x, r \rangle]$.

The inverter A', on input $y = f(x)$, runs the algorithm GLW using the oracle $A'(y, \cdot)$. If x is good, then the algorithm finds x with probability at least $1 - o(1)$. At least half of the choices of x are good, so overall the algorithm inverts $f()$ on at least a $.5 - o(1) > 1/3$ fraction of inputs. The running time of the algorithm if $O(n^2 \log n)$ plus the cost of $32n \log n$ calls to $A(y, \cdot)$, each costing time t. $\square$