

## Notes for Lecture 11 (Draft)

### 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  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  is  $(t, \epsilon)$ -one way if for every algorithm  $A$  of complexity  $\leq t$  we have

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

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  $\nu : \mathbb{N} \rightarrow \mathbb{R}$  is called *negligible* if for every polynomial  $p$  we have  $\lim_{n \rightarrow \infty} \nu(n)/p(n) = 0$ .

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

1.  $f$  is polynomial time computable and
2. for every polynomial  $p(\cdot)$  there is a negligible function  $\nu$  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  $I \subseteq \{1, \dots, k\}$ .

The output is

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

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

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

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

**Definition 4 (One-way Permutation)** If  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is a bijective  $(t, \epsilon)$ -one way function, then we call  $f$  a  $(t, \epsilon)$ -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  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  be a  $(t, \epsilon)$ -secure one-way permutation. Show that

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

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  $\{1, \dots, p-1\}$  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  $2^{O((\log p)^{1/3})}$ . It is plausible that for most  $p$  and most  $g$  the discrete logarithm is a  $(t, \epsilon)$  one way permutation with  $t$  and  $\epsilon^{-1}$  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  $\ell$  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  $\ell$  bits into  $2\ell$ , 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  $P : \{0, 1\}^n \rightarrow \{0, 1\}$  is  $(t, \epsilon)$ -hard core for a permutation  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$  if for every algorithm  $A$  of complexity  $\leq t$

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

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

**Exercise 3** Suppose that  $P$  is a  $(t, \epsilon)$ -hard core predicate for a permutation  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ , 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.

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

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

(where  $\langle x, r \rangle := \sum_i x_i r_i \bmod 2$ .) Then there is an algorithm  $A'$  of complexity at most  $O(t\epsilon^{-2}n^{O(1)})$  such that

$$\mathbb{P}_x[A'(f(x)) = x] \geq \frac{\epsilon}{2}$$