Lecture 2: Pseudorandom functions

Last time: One-time pad (OTP)

Today:

1. Recap
2. Motivate and define the notion of pseudorandomness
3. Define the notion of pseudorandom functions (PRFs)
4. Show how to use a PRF to construct a reusable encryption

Recap: A symmetric encryption scheme is associated a message space M a key space K and a ciphertext space C and with two algorithms:

$$Enc: K \times M \to C$$

$$Dec: K \times C \to M$$

Correctness: For every message $m \in M$ and any key $k \in K$

$$Dec\big(k, Enc(k, m)\big) = m$$

Perfect (one-time) security:

For every m and m' in M and for every c in C

$$Pr[Enc(k,m)=c] = Pr[Enc(k,m')=c]$$

Intuitively, this means that a ciphertext gives no information about the message since it occurs with the same probability irrespective of the message.

This is a very strong definition, since it says that even if the advesary knows that the encrypted message is ether m or m', still c does not reveal which one it is.

On the other hand it is too weak since it says that the ciphertext gives no information assuming the adversary has no knowledge about the secret key!

However, a ciphertext is a function of the secret key

and hence may give information about the key.

This definition undermines the power of the adversary!

One-time Pad [Frank Miller 1882, Gilbert Vernam 1917]

The message space, key space and ciphertext space are $\{0,1\}^n$

$Enc(k,m) = k \oplus m$

$Dec(k,c) = k \oplus c$

This scheme has perfect (one-time) security since

for every m and c in $\{0,1\}^n$

$Pr[Enc(k,m)=c = 1/2^n$

Downside:

This scheme breaks down completely if 2 messages are encrypted,

since a ciphertext gives a lot of information about the secret key k

Inherent downside: The key needs to be as large as the total number of bits encrypted (if we require perfect security).

Cryptography is the art of overcoming barriers!

Idea: Let's generate more randomness from a short random key

This way, we can encrypt a long message using a short random secret key, by first "stretching" it and then using the "stretched" secret key to encrypt the long message.

It is impossible to generate randomness!

Amazing insight: We can generate **computational randomness from hardness**!

We overcome the barrier by restricting the class of adversaries that we are considering to be the class of "computationally bounded" adversaries.

Now, we don't need the key to random, rather it only needs to "**look random**".

It turns out that we can take a short random string and stretch it into one that "looks" random.

The stretched string is not random, but it is indistinguishable from random in the eyes of a computationally bounded adversary.

Question: How do we define a computationally bounded Adversary and how do we define "looks random"?

In theory: A computationally bounded adversary is one that runs in probabilistic polynomial time (PPT).

In practice: A computationally bounded adversary is one that runs in time at most, say 2^{80} or 2^{96}.

We next define what it means to "look random".

Intuitively, it means that a PPT adversary "cannot distinguish" between the "random looking" distribution and the uniform distribution.

"cannot distinguish" means it can distinguish with only negligible probability.

In practice: negligible means essentially zero (say 2^{-80} or 2^{-96}).

In theory: negligible is defined as follows:

Definition: A function $\mu: N \to N$ is said to be **negligible** if for every polynomial $p: N \to N$, there exists n_0 such that for every $n > n_0$, $\mu(n) < \frac{1}{p(n)}$.

Definition: A distribution ensemble $\{D_n\}_{n\in N}$, where D_n generates strings in $\{0,1\}^n$, is said to be **pseudorandom** if for every PPT adversary A there exists a negligible function $\mu: N \to N$ s.t.

$$\left|\Pr_{x\leftarrow D_n}[A(x)=1] - \Pr_{x\leftarrow\{0,1\}^n}[A(x)=1]\right| < \mu(n).$$

Initial Goal: Construct a function $PRG: \{0,1\}^n \to \{0,1\}^{n+1}$ s.t. the distribution $PRG(U_n)$ is pseudorandom.

Theorem: PRGs exist assuming one-way functions exist.

Intuitively, a one-way function (OWF) is a function that is easy to compute but hard to invert.

Definition: a function $f_n: \{0,1\}^n \to \{0,1\}^{m(n)}$ is said to be a one-way function (OWF) if it is easy to compute (i.e., it is computable in poly-time) and for any PPT A there exists a negligible function μ s.t. for every $n \in N$,

$$\Pr\left[A\left(f(x)\right) = x' \; s.t. \; x' \in f^{-1}\left(f(x)\right)\right] < \mu(n)$$

We strongly believe that OWFs exist.

Example: $f(x) = g^x \; mod \; p$, where p is a fixed n-bit prime number and g is a fixed element in $\{1, \dots, p-1\}$.

It is not a OWF for every p, g but it is easy to generate p, g such that $f = f_{p,g}$ is believed to be a OWF.

We don't know how to prove that it is a OWF but rather we assume it is. This is called the Discrete Log Assumption.

An idea of how PRG's are constructed:

Suppose $f: \{0,1\}^n \to \{0,1\}^n$ is a one-way permutation, such that given $f(x)$, say the first bit of x, denoted by x_1, looks random. Then, $PRG(x) = f(x)||x_1$.

If x_1 looks random given $f(x)$, then x_1 is called a hard-core predicate. It turns out that any OWF has a hard-core predicate, which is used to generate "randomness".

This shows how to generate a single bit or randomness.

We want to generate many bits or randomness

Theorem: Once we can generate one bit of randomness, we can generate as much randomness as we want (even exp. many)!

Definition: A function $f: K \times \{0,1\}^n \to \{0,1\}^m$ is said to be a **pseudorandom function** (PRF) if a PPT A cannot distinguish between given oracle access to $f(k,\cdot)$ for random $k \leftarrow K$ and oracle access to a truly random function $U: \{0,1\}^n \to \{0,1\}^m$.

Theorem: A PRF exists assuming a PRG exists (i.e., OWF exists).

This construction is inefficient in practice

Practical construction: **AES** (Advanced Encryption Standard).

This is a heuristic construction and is not provably secure under standard hardness assumptions, but it is used a lot in practice.

A PRF allows us to get reusable security!

Encrypt the $i'th$ message by $m \oplus f(k, i)$,

where k is the secret key.

This is a stateful encryption where the sender and receiver need to remember the message number i.

One can construct a stateless encryption, by using randomness:

$$Enc(k, m) = (r, f(k, r) \oplus m)$$

$$Dec\big(k, (r, c)\big) = f(k, r) \oplus c$$

It turns out that randomness is necessary to achieve (many-time, computational) security!