PSEUDO-RANDOM FUNCTIONS CONCEPTS

Overview

Pseudo-Random Functions (PRFs) generalize the notion of pseudo-random generators, a pseudo-random function is like random-looking functions as opposed to random-looking strings generated in pseudo-random generators.

Here we consider the pseudorandomness of distribution on functions and this type of distribution is introduced through keyed functions.

$$F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$$

Prf is of the form F given above where the first input is called the key and is mostly represented by k.

A Pseudorandom function is said to be efficient if there is a polynomial-time algorithm that computes F (k, x)

More formally it is defined as below

DEFINITION 3.24 *An efficient, length preserving, keyed function* $F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ *is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers* D, *there is a negligible function* negl *such that:*

$$\left| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \right| \leq \mathsf{negl}(n),$$

where the first probability is taken over uniform choice of $k \in \{0,1\}^n$ *and the randomness of* D, *and the second probability is taken over uniform choice of* $f \in \mathsf{Func}_n$ *and the randomness of* D.

Proof:

We stress that D is *not* given the key k (in the same way that D is not given the seed when defining a pseudorandom generator). It is meaningless to require that F_k "look random" if k is known, since given k it is trivial to distinguish an oracle for F_k from an oracle for f. (All the distinguisher has to do is query the oracle at any point x to obtain the answer y, and compare this to the result $y' := F_k(x)$ that it computes itself using the known value k. An oracle for F_k will return $y = y'$, while an oracle for a random function will return $y = y'$ only with probability 2^{-n}.) This means that if k is revealed, any claims about pseudorandomness no longer hold.

Code construction:

In the main program, a length preserving pseudorandom function has been implemented.

In a length preserving function for some key k $\in \{0,1\}^n$ the function F_K maps n bit input to n bit output.

The construction of the given Pseudo-Random function from a pseudo-random function is formally defined as follows.

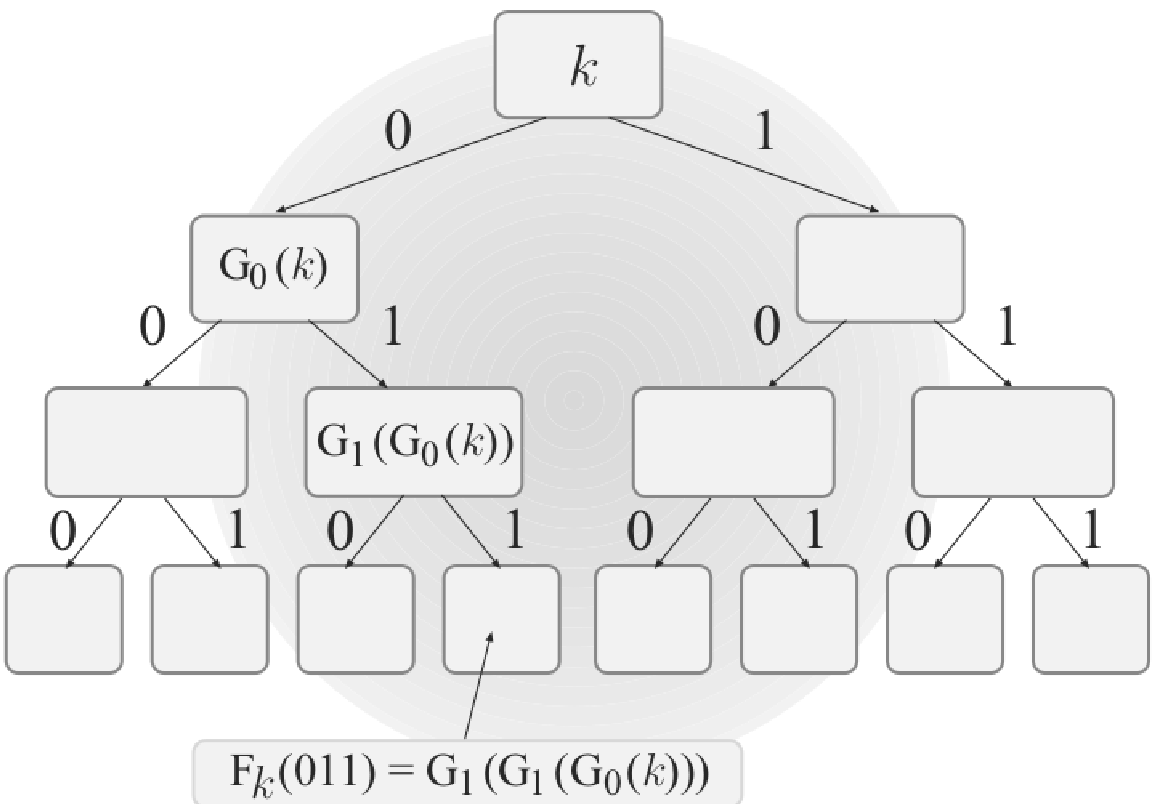
> ***CONSTRUCTION 8.20***
>
> Let G be a pseudorandom generator with expansion factor $\ell(n) = 2n$, and define G_0, G_1 as in the text. For $k \in \{0,1\}^n$, define the function $F_k : \{0,1\}^n \to \{0,1\}^n$ as:
>
> $$F_k(x_1 x_2 \cdots x_n) = G_{x_n}\left(\cdots (G_{x_2}(G_{x_1}(k)))\cdots\right).$$

Here G_0 and G_1 are two invocations of pseudo-random functions on any input. G_0 outputs the value after taking first n bits out of a 2n bit binary string whereas G_1 takes the later n bits.

The function can be viewed as a binary tree of depth n where each node contains an n bit value and key k is the root node and for all non-leaf node with value v there are two child the left child has value $G_0(v)$ and right child has a value $G_1(v)$

The result F_k (x) for $x = x_1 \cdot \cdot \cdot x_n$ is defined to be the value on the leaf node
reached by traversing the tree according to the bits of x, where x i = 0 means
"go left" and $x_i = 1$ means "go right."

The sample construction is shown below.



Proof:

THEOREM 8.21 *If G is a pseudorandom generator with expansion factor $\ell(n) = 2n$, then Construction 8.20 is a pseudorandom function.*

PROOF We first show that for any polynomial t it is infeasible to distinguish $t(n)$ uniform $2n$-bit strings from $t(n)$ pseudorandom strings; i.e., for any polynomial t and any PPT algorithm A, the following is negligible:

$$\left| \Pr\left[A\left(r_1 \| \cdots \| r_{t(n)}\right) = 1\right] - \Pr\left[A\left(G(s_1) \| \cdots \| G(s_{t(n)})\right) = 1\right] \right|,$$

where the first probability is over uniform choice of $r_1, \ldots, r_{t(n)} \in \{0,1\}^{2n}$, and the second probability is over uniform choice of $s_1, \ldots, s_{t(n)} \in \{0,1\}^n$.

The proof is by a hybrid argument. Fix a polynomial t and a PPT algorithm A, and consider the following algorithm A':

Distinguisher A':
A' is given as input a string $w \in \{0,1\}^{2n}$.

1. Choose uniform $j \in \{1, \ldots, t(n)\}$.
2. Choose uniform, independent values $r_1, \ldots, r_{j-1} \in \{0,1\}^{2n}$ and $s_{j+1}, \ldots, s_{t(n)} \in \{0,1\}^n$.
3. Output $A\left(r_1 \| \cdots \| r_{j-1} \| \, w \, \| G(s_{j+1}) \| \cdots \| G(s_{t(n)})\right)$.

For any n and $0 \leq i \leq t(n)$, let G_n^i denote the distribution on strings of length $2n \cdot t(n)$ in which the first i "blocks" of length $2n$ are uniform and the remaining $t(n) - i$ blocks are pseudorandom. Note that $G_n^{t(n)}$ corresponds to the distribution in which all $t(n)$ blocks are uniform, while G_n^0 corresponds to the distribution in which all $t(n)$ blocks are pseudorandom. That is,

$$\begin{aligned} &\left| \Pr_{y \leftarrow G_n^{t(n)}}[A(y) = 1] - \Pr_{y \leftarrow G_n^0}[A(y) = 1] \right| \qquad (8.11) \\ &= \left| \Pr\left[A\left(r_1 \| \cdots \| r_{t(n)}\right) = 1\right] - \Pr\left[A\left(G(s_1) \| \cdots \| G(s_{t(n)})\right) = 1\right] \right| \end{aligned}$$

,

Say A' chooses $j = j^*$. If its input w is a uniform $2n$-bit string, then A is run on an input distributed according to $G_n^{j^*}$. If, on the other hand, $w = G(s)$ for uniform s, then A is run on an input distributed according to $G_n^{j^*-1}$. This means that

$$\Pr_{r \leftarrow \{0,1\}^{2n}}[A'(r) = 1] = \frac{1}{t(n)} \cdot \sum_{j=1}^{t(n)} \Pr_{y \leftarrow G_n^j}[A(y) = 1]$$

and

$$\Pr_{s \leftarrow \{0,1\}^{n}}[A'(G(s)) = 1] = \frac{1}{t(n)} \cdot \sum_{j=0}^{t(n)-1} \Pr_{y \leftarrow G_n^j}[A(y) = 1].$$

Therefore,

$$\begin{aligned} &\left| \Pr_{r \leftarrow \{0,1\}^{2n}}[A'(r) = 1] - \Pr_{s \leftarrow \{0,1\}^{n}}[A'(G(s)) = 1] \right| \\ &= \frac{1}{t(n)} \cdot \left| \Pr_{y \leftarrow G_n^{t(n)}}[A(y) = 1] - \Pr_{y \leftarrow G_n^0}[A(y) = 1] \right|. \end{aligned} \tag{8.12}$$

Since G is a pseudorandom generator and A' runs in polynomial time, we know that the left-hand side of Equation (8.12) must be negligible; because

$t(n)$ is polynomial, this implies that the left-hand side of Equation (8.11) is negligible as well.

Turning to the crux of the proof, we now show that F as in Construction 8.20 is a pseudorandom function. Let D be an arbitrary PPT distinguisher that is given 1^n as input. We show that D cannot distinguish between the case when it is given oracle access to a function that is equal to F_k for a uniform k, or a function chosen uniformly from Func_n. (See Section 3.5.1.) To do so, we use another hybrid argument. Here, we define distributions over n-bit values at the leaves of a complete binary tree of depth n. By associating each leaf of these binary trees with an n-bit input as in Construction 8.20, we can equivalently view these as distributions over functions mapping n-bit inputs to n-bit outputs. For any n and $0 \leq i \leq n$, let H_n^i be the following distribution over the values at the leaves of a binary tree of depth n: first choose values for the nodes at level i independently and uniformly from $\{0,1\}^n$. Then for every node at level i or below with value k, its left child is given value $G_0(k)$ and its right child is given value $G_1(k)$. Note that H_n^n corresponds to the distribution in which all values at the leaves are chosen uniformly and independently, and thus corresponds to choosing a uniform function from Func_n, whereas H_n^0 corresponds to choosing a uniform key k in Construction 8.20 since in that case only the value at the root (at level 0) is chosen uniformly. That is,

$$\left| \Pr_{k \leftarrow \{0,1\}^n}[D^{F_k(\cdot)}(1^n) = 1] - \Pr_{f \leftarrow \mathsf{Func}_n}[D^{f(\cdot)}(1^n) = 1] \right|$$
$$= \left| \Pr_{f \leftarrow H_n^0}[D^{f(\cdot)}(1^n) = 1] - \Pr_{f \leftarrow H_n^n}[D^{f(\cdot)}(1^n) = 1] \right|. \qquad (8.13)$$

We show that Equation (8.13) is negligible, completing the proof.

Let $t = t(n)$ be a polynomial upper bound on the number of queries D makes to its oracle on input 1^n. Define a distinguisher A that tries to distinguish $t(n)$ uniform $2n$-bit strings from $t(n)$ pseudorandom strings, as follows:

Distinguisher A:
A is given as input a $2n \cdot t(n)$-bit string $w_1 \| \cdots \| w_{t(n)}$.

1. Choose uniform $j \in \{0, \ldots, n-1\}$. In what follows, A (implicitly) maintains a binary tree of depth n with n-bit values at (a subset of) the internal nodes at depth $j+1$ and below.
2. Run $D(1^n)$. When D makes oracle query $x = x_1 \cdots x_n$, look at the prefix $x_1 \cdots x_j$. There are two cases:
   - If D has never made a query with this prefix before, then use $x_1 \cdots x_j$ to reach a node v on the jth level of the tree. Take the next unused $2n$-bit string w and set the value of the left child of node v to the first half of w, and the value of the right child of v to the second half of w.

- If D has made a query with prefix $x_1 \cdots x_j$ before, then node $x_1 \cdots x_{j+1}$ has already been assigned a value.

Using the value at node $x_1 \cdots x_{j+1}$, compute the value at the leaf corresponding to $x_1 \cdots x_n$ as in Construction 8.20, and return this value to D.

3. When execution of D is done, output the bit returned by D.

A runs in polynomial time. It is important here that A does not need to store the entire binary tree of exponential size. Instead, it "fills in" the values of at most $2t(n)$ nodes in the tree.

Say A chooses $j = j^*$. Observe that:

1. If A's input is a uniform $2n \cdot t(n)$-bit string, then the answers it gives to D are distributed exactly as if D were interacting with a function chosen from distribution $H_n^{j^*+1}$. This holds because the values of the nodes at level $j^* + 1$ of the tree are uniform and independent.

2. If A's input consists of $t(n)$ pseudorandom strings—i.e., $w_i = G(s_i)$ for uniform seed s_i—then the answers it gives to D are distributed exactly as if D were interacting with a function chosen from distribution $H_n^{j^*}$. This holds because the values of the nodes at level j^* of the tree (namely, the $\{s_i\}$) are uniform and independent. (The $\{s_i\}$ are unknown to A, but that makes no difference.)

Proceeding as before, one can show that

$$\left| \Pr\left[A\left(r_1 \| \cdots \| r_{t(n)}\right) = 1\right] - \Pr\left[A\left(G(s_1) \| \cdots \| G(s_{t(n)})\right) = 1\right] \right| \tag{8.14}$$
$$= \frac{1}{n} \cdot \left| \Pr_{f \leftarrow H_n^0}[D^{f(\cdot)}(1^n) = 1] - \Pr_{f \leftarrow H_n^n}[D^{f(\cdot)}(1^n) = 1] \right|.$$

We have shown earlier that Equation (8.14) must be negligible. The above thus implies that Equation (8.13) must be negligible as well. ■