| | |
|---|---|
| **6.875J/18.425J Cryptography and Cryptanalysis** | October 24, 2001 |
| Lecture 13 | |
| *Instructor: Anna Lysyanskaya* | *Scribed by: Marshall Tappen* |

# 1 PRF definition

A random function is one whose output is fixed at random. A pseudorandom function, intuitively, is one that is not really random, but you cannot tell. This notion was introduced by Goldreich, Goldwasser, and Micali, who also gave a construction.

Let $\mathcal{F} : D \mapsto R$ be a function; let $A(\cdot)$ be and algorithm. By $A^{\mathcal{F}}(\cdot)$ we denote an algorithm that makes oracle calls to the function $\mathcal{F}$. More precisely, we may imagine that $A$ has an additional tape $Q$ (for "query"), on which it can write queries to $\mathcal{F}$, and as soon as it is done writing such a query $q$, an answer to it, namely $\mathcal{F}(q)$ magically appears on this tape.

**Definition 1** *A family of functions $F.(\cdot)$ is pseudorandom if $F$ is efficiently computable and for a randomly chosen $S$, $F_S$ cannot be distinguished from a random function. More precisely, let $\rho(k, l(k), m(k))$ denote a family of random functions from $\{0,1\}^{l(k)}$ to $\{0,1\}^{m(k)}$. $F.(\cdot)$ determines a family of pseudorandom functions from $l(k)$ bits to $m(k)$ bits if $\forall$ PPTF $\{A_k\}$, there exists a negligible function $\nu(k)$ such that*

$$\begin{aligned} \Pr[S \leftarrow \{0,1\}^k; O_0 = F_S \quad &; \\ R \leftarrow \rho(k, l(k), m(k)); O_1 = R \quad &; \\ b \leftarrow \{0,1\}; b' \leftarrow A_k^{O_b}(1^k) \quad &: \quad b = b'] \leq 1/2 + \nu(k) \end{aligned}$$

Intuitively, the Turing machine for $F$ and the random $k$ bit seed implicitly defines a string of length $2^{l(k)} \cdot m(k)$.

# 2 Remarks about PRFs

## 2.1 From one-bit PRF to PRFs of arbitrary length

Let $\{F_S\}$ from $l(|S|)$ bits to 1 bit be a PRF family. (Note that a function can only be a PRF if it i chosen from a family. It it was fixed, it would be easy to tell).

$F(s)$ defines a string of length of length $2^{l(k)}$. In this string, the bit at position $x$ is equal to $F(x)$ ($F_S$'s output is 1-bit). So to find $F(15)$, look at the bit in position 15.

We can use this string to get 2-bit out. In this case:

$$\begin{aligned} F'_S(1) &= F_S(1) \circ F_S(2) \\ F'_S(3) &= F_S(5) \circ F_S(6) \end{aligned}$$

In this case, $F_S$ is a string of length $2^{l(k)-1} \cdot m$, $(m = 2)$.

In this fashion, we can construct a PRF from approximately $l(k)$ bits (maybe a bit fewer than $l(k)$ bits) to $m(|S|)$ bits.

Formally, have

$$F_S : \{0,1\}^{l(s)} \to \{0,1\}$$

and we want make

$$F_S : \{0,1\}^{l(s)-small} \to \{0,1\}^{m(k)}$$

where $m(k)$ is an arbitrary polynomial and $l(k)$ is a polynomial.

$$\begin{aligned} F'_S(x) &= F_S(x_1 \circ x_2 \circ \ldots \circ x_{l'(k)} \circ \underbrace{0 \circ 0 \circ \ldots \circ 0}_{\log_2 m(k) \text{ bits}}) \\ &\quad \circ F_S(x_1 \circ x_2 \circ \ldots \circ x_{l'(k)} \circ \underbrace{0 \circ 0 \circ \ldots \circ 1}_{\log_2 m(k) \text{ bits}}) \\ &\quad \vdots \\ &\quad \circ F_S(x_1 \circ x_2 \circ \ldots \circ x_{l'(k)} \circ \underbrace{1 \circ 1 \circ \ldots \circ 1}_{\log_2 m(k) \text{ bits}}) \end{aligned}$$

where $l'(k) = l(k) - \lceil \log_2 m(k) \rceil$.

(Can you $F'_S$ be a PRF even when the underlying $F_S$ is not a PRF? Yes: if $m(k)$ is not a power of two, then some bits of $F_S$ are never supposed to be queried. There are $2^{\lceil \log_2 m(k) \rceil} - m$ such wasted positions per each query to $F'_S$. A contrived example for an $F_S$ that is not a PRF yielding an $F'_S$ that *is* a PRF, could be created where all of these positions are zero.)

## 3 The GGM construction of PRFs from PRGs

(Due to Goldreich, Goldwasser, and Micali)

Suppose $G$ is a PRG that expands $k$ random bits into $2k$ pseudorandom bits. Then here is a tree-like construction of a function, $F_S$, from $\{0,1\}^k$ to $\{0,1\}^k$ with a $k$-bit seed:

Since $G(x)$ is length $2k$, we can divide it in half and create $G_0$ and $G_1$. $G_0(x)$ is the first $k$ bits of $G(x)$ and $G_1(x)$ is the last $k$ bits of $G(x)$

Let $S \leftarrow \{0,1\}^k$ be the random seed and $x = x_1 \circ x_2 \circ \ldots \circ x_k$ . Then $F_s(x) \triangleq l_x$ where

$$\begin{aligned} l_x &= l_{x_1 \circ x_2 \circ \ldots \circ x_k} \\ &= G_{x_k}(l_{x_1 \circ x_2 \circ \ldots \circ x_{k-1}}) \\ &= G_{x_k} G_{x_{k-1}}(l_{x_1 \circ x_2 \circ \ldots \circ x_{k-2}}) \end{aligned}$$

$$\begin{aligned} l_0 &= G_0(S) \\ l_1 &= G_1(S) \end{aligned}$$

See Figure 3 for a graphical representation of this process. At each level of the tree, the pseudorandomness doubles and the string that represents this function has length $2^k \cdot k$.

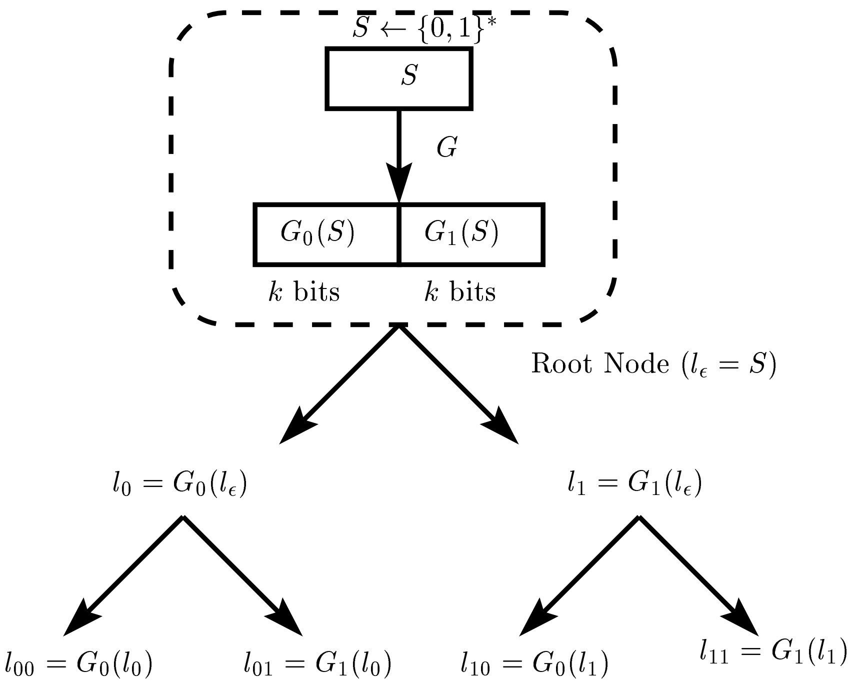


Figure 1: The tree used to create a PRF from a PRG

### 3.1 Why is $F_s$ a PRF?

We can show that $F_s$ is a PRF using a hybrid argument. The hybrids cannot be generated horizontally because there would be an exponentially large number of hybrids, but we can generate them vertically.

The hybrids are defined in this way:

> $0^{th}$ hybrid: $F_S$ for some random $S$.
>
> $i^{th}$ hybrid: Everything on level $i$ is random. On all levels $j > i$, labels are derived from labels in level $i$.
>
> $k^{th}$ hybrid: Some random function.

**Claim:** If $A_k$ distinguishes $F_S$ from a random function $R$, then, for some $i$, it can distinguish $F_S^i$ from $F_S^{i+1}$.

This claim is true because there are only a polynomial number of hybrids and because of the claim we proved the first time that hybrids were introduced. From the problem sets, we also know that we can choose $i$ at random.

### 3.2 Using $A_k$ to distinguish $G(\cdot)$ from a random string

There are two important claims:

1. $A_k$ can be used to distinguish $p(k)$ random $2k$-bit strings from $p(k)$ pseudorandom $2k$-bit strings. (Here $p(k)$ is an upperbound on the number of queries that $A_k$ makes).

2. If $A_k$ can distinguish $p(k)$ random strings from $p(k)$ pseudorandom strings, then it can tell if a given string is random or pseudorandom.

Claim 2 can be shown with a hybrid argument and will not be proved here.

Claim 1 can be show by creating a reduction.

The goal is to create a reduction $R$, which given strings $S_1, \ldots S_{p(k)}$, produces $F_S^i$ if the $S_j$'s are random or produces $F_s^{i+1}$ if they are pseudorandom.

On input $S_1, \ldots S_{p(k)}$, the steps of the reduction are:

On query $x_l$

- If $x_l$ has already been queried, then return the previous answer generated.
- Otherwise, find the longest common prefix between $x_l$ and some $x_m$, $m < l$, and call this prefix $y_{l,m}$.
  * If $|y_{l,m}| \geq i+1$ compute $F(x_l)$ by deriving it from $l_{y_{l,m}}$.
  * If $|y_{l,m}| = i$ Do the same.
  * Otherwise, take the next unused $S_j$, let $y_l$ be the first $i$ bits of $x_l$, let $b_l$ be the $i+1$'st bit of $x_l$. Then $l_{y_l b_l} = S_j^{b_l}$. Where $S_j^0$ is the first $k$ bits of $S_j$ and $S_j^1$ is the last $k$ bits of $S_j$.

A graphical example of this process can be seen in Figure 3.2. In this case $F(0001)$ is queried, followed by $F(1101)$.

**Claim:** If $S_j$'s are random, resulting $F$ is an $F^{i+1}$. If $S_j$'s are pseudorandom, then $F$ is an $F_S^i$.

This can be seen by examining the construction and the definition of the hybrids.

So, if $A_k$ tells apart the $i^{th}$ and $i+1$st hybrids, it tells us whether the $S_j$'s are random or pseudorandom.

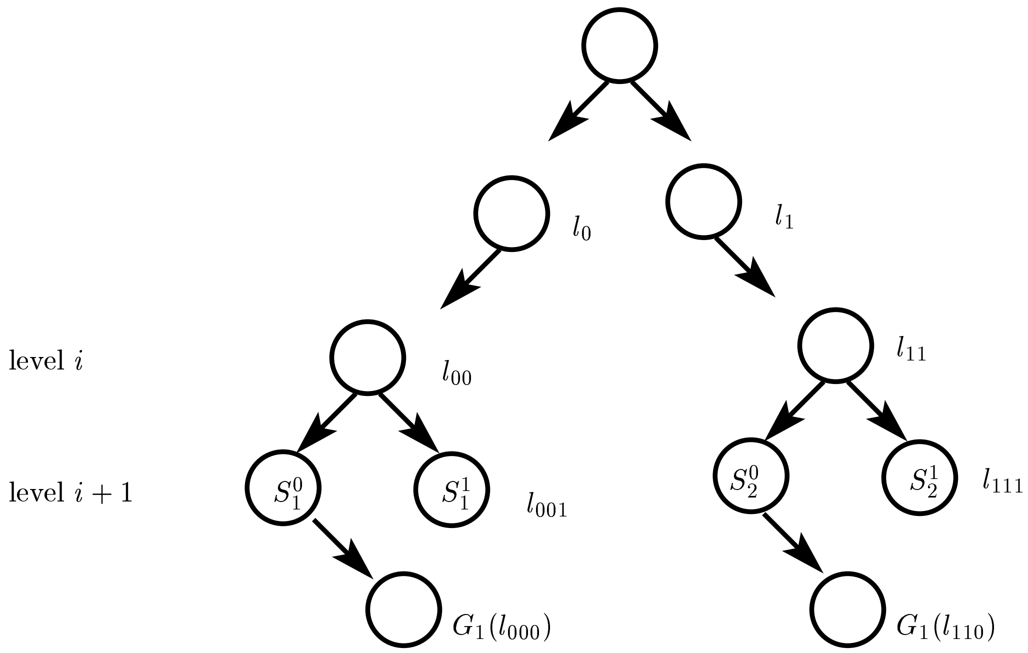


Figure 2: An example of the Reduction $R$.