CS 427, Assignment 3

Cody Malick
malickc@oregonstate.edu

February 2, 2017

1

a

The PRG seems to be secure. While the results of y never change, the randomness of the first λ bits are still uniformly random, as it has been given that $G()$ is secure. The only difference, then, would be that instead of $H(s)$ returning a string of $0, 1^{3\lambda}$, it returns $0, 1^{2\lambda}$. Incidentally, this is identical behavior to the function we were provided, $G()$. We can prove that this is secure by showing that $H(s)$ is indistinguishable from a secure PRG:

$$\begin{array}{l} H(s): \\ \quad x := G(s) \\ \quad y := G(0^{\lambda}) \\ \quad return(x||y) \end{array}$$

We can first abstract $x := G(s)$ because $G()$ is secure. We can replace that with a random distribution of $x \leftarrow 0, 1^{\lambda}$. This does not affect the calling program as it is simply the result of $G(s)$:

$$\begin{array}{l} H(s): \\ \quad \boldsymbol{x \leftarrow \{0,1\}^{2\lambda}} \\ \quad y := G(0^{\lambda}) \\ \quad return(x||y) \end{array}$$

Next, I claim we can remove the constant string from the library, as it would not change the probability of the resulting function. In the former case, we had a result of:

$\frac{1}{2^{2\lambda}}c$

Where c is an unchanging constant, the resulting value of y. As we call this function an arbitrarily large number of times, the unchanging value of c would have no effect on the resulting probability function. Therefore:

$$\begin{array}{l} H(s): \\ \quad x \leftarrow \{0,1\}^{2\lambda} \\ \quad return(\boldsymbol{x}) \end{array}$$

As expected, this ends up being identical to $G()$, and therefore, $H()$ is indistinguishable from $G()$, and a secure PRG.

b

There is a problem with this PRG, in that you are always xoring a constant value, $G(0^\lambda)$. We can take advantage of this in an attack, by running two different values through the PRG x_1 and x_2, then xoring those two values together. The resulting string is the values of $G(x_1), G(x_2)$ xored together. With that strategy, an arbitrary calling program can always check the resulting string:

$$
\begin{array}{l}
P():\\
\quad x_1, x_2 \leftarrow \{0,1\}^\lambda\\
\quad y_1 := H(x_1)\\
\quad y_2 := H(x_2)\\
\quad if(G(x_1) \bmod G(x_2)) == (y_1 \bmod y_2))\ \{\\
\quad\quad return\ true\\
\quad \}\\
\quad return\ false
\end{array}
$$

With this setup, $P()$ will return true with a probability of 1, while a secure PRG of this type should return $\frac{1}{2^{2\lambda}}$

c

This is similar to the first problem, in that it simply adds a constant string to the resulting random string from $G()$:

$$
\begin{array}{l}
H(s):\\
\quad x := G(s)\\
\quad return(s||x)
\end{array}
$$

We can first abstract $x := G(s)$ because $G()$ is secure. We can replace that with a random distribution of $x \leftarrow 0, 1^\lambda$. This does not affect the calling program as it is simply the result of $G(s)$:

$$
\begin{array}{l}
H(s):\\
\quad \boldsymbol{x \leftarrow \{0,1\}^{2\lambda}}\\
\quad return(s||x)
\end{array}
$$

Next, I claim we can remove the constant string from the library, as it would not change the probability of the resulting function. In the former case, we had a result of:

$\frac{1}{2^{2\lambda}}c$

Where c is an unchanging constant, the resulting value of y. As we call this function an arbitrarily large number of times, the unchanging value of c would have no effect on the resulting probability function. Therefore:

$$
\begin{array}{l}
H(s):\\
\quad x \leftarrow \{0,1\}^{2\lambda}\\
\quad return(\boldsymbol{x})
\end{array}
$$

As before, the resulting function is indistinguishable to $G()$ in probability.

2

This is a relatively simplistic proof. Our end goal is simply to show that we can freely exchange G_1 and G_2. Starting with $\mathcal{L}^{G_1}_{which-prg}$:

$$\begin{array}{l} QUERY():\\ \qquad s \leftarrow \{0,1\}^{\lambda} \\ \qquad return\ G_1(s) \end{array}$$

My first step would be abstract away the call to G_1 into a seperate function:

$$\begin{array}{l} QUERY():\\ \qquad s \leftarrow \{0,1\}^{\lambda} \\ \qquad \boldsymbol{z := \mathcal{F}(s)} \\ \qquad return\ \boldsymbol{z} \end{array}$$

Where $\mathcal{F}$:

$$\begin{array}{l} \mathcal{F}(s):\\ \qquad z := G_1(s) \\ \qquad return\ z \end{array}$$

We can substitute the call to G_1 with a call to G_2 as they both have equal input and output ranges:

$$\begin{array}{l} \mathcal{F}(s):\\ \qquad z := \boldsymbol{G_2(s)} \\ \qquad return\ z \end{array}$$

With that changed, we can substitute the resulting function back into $QUERY()$:

$$\begin{array}{l} QUERY():\\ \qquad s \leftarrow \{0,1\}^{\lambda} \\ \qquad \boldsymbol{z := G_2(s)} \\ \qquad return\ z \end{array}$$

With that in place, we've shown that $\mathcal{L}^{G_1}_{which-prg}$ is indistinguishable from $\mathcal{L}^{G_2}_{which-prg}$