CS-E4340 Cryptography: Exercise Sheet 2

—One-Way Functions & Pseudorandom Generators—

Submission Deadline: September 19, 11:30 via MyCourses

Each exercise can give up to two participation points, 2 for a mostly correct solution and 1 point for a good attempt. Overall, the exercise sheet gives at most 4 participation points. We encourage to **choose** those exercises which **interesting** and/or adequately challenging to you.

Exercise Sheet 2 is intended to help...

(a) ...understand the *definition* of pseudorandom generators (PRGs).
(b) ...understand the *relation* between PRGs and OWFs.
(c) ...develop intuition about the difficulty of *constructing* hardcore bits.
(d) ...continue familiarizing with the idea of *generic counterexamples*.
(e) ...familiarize yourself with proofs via *transformations* (which we will later call *reductions*).

Exercise 1 shows that a PRG might leak half of its input and thus, just like one-way functions, is not guaranteed to hide its input.

Ex. 2 & Ex. 3 show that PRGs are a strictly stronger notion than OWFs. Namely, Exercise 3 shows that every PRG is a OWF, while Exercise 2 shows that not every OWF is a PRG.

Counterexamples & Adversary transformation Exercise 1 and Exercise 2 help practice the notion of generic counterexamples, whereas Exercise 3 helps practice the notion of transformation of one adversary into another.

Ex. 4 & Ex. 5 aim to help understand the notion of universal hardcore bits.

Exercise 6 is advanced; you are asked to prove that the Goldreich-Levin hard-core bit is, indeed, a hard-core bit.

Exercise 7 is experimental—note that we do not know whether it has a solution.

Exercise 1 (PRGs can leak half their input). Let $f : \{0,1\}^* \to \{0,1\}^*$ be a PRG. We define

$$g_f(x) = f(x_\ell)||x_r$$

Here, x_ℓ consists of the first $\lceil |x|/2 \rceil$ bits of x and x_r consists of the last $\lfloor |x|/2 \rfloor$ bits of x, i.e., $x = x_\ell||x_r$.

Task: Prove via reduction that if f is a PRG, then g_f is a PRG, too.

Solution 1. We begin by noting that if f is polynomial time computable, then g_f is polynomial time computable. Next, for length-expansion, suppose f is a PRG with stretch $s(\lambda) > 0, \forall \lambda \in \mathbb{N}$. Then, denote by $s'(\lambda)$, the length expansion of g_f. By construction, we have

$$s'(|x|) = s(|x_\ell|) > 0$$

Security. We now assess the pseudorandomness property of g_f. We will do this via a reduction argument, transforming a distinguisher for g_f into a distinguisher for f. This is sufficient, since by assumption f is a PRG, and hence, such distinguisher does not exist for f. This then leads to a contradiction to the existence of a distinguisher for g_f in the first place.

Formally, reduction is a technique that transforms any probabilistic polynomial time (PPT) algorithm (distinguisher) $\mathcal{A}$ against g_f into a new PPT algorithm (distinguisher) $\mathcal{R}^{\mathcal{A}}$ against f such that the following holds: if

$$\mathbf{Adv}^{\mathsf{PRG}}_{g,\mathcal{A}}(n) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{PRG},0}_{g,s',\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{PRG},1}_{s',\mathcal{A}}(1^\lambda) = 1\right]\right|$$

is non-negligible, then

$$\mathbf{Adv}^{\mathsf{PRG}}_{f,\mathcal{R}^{\mathcal{A}}}(n) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{PRG},0}_{f,s,\mathcal{R}^{\mathcal{A}}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{PRG},1}_{s,\mathcal{R}^{\mathcal{A}}}(1^\lambda) = 1\right]\right|$$

is non-negligible.

Reduction: Consider the following distinguisher $\mathcal{R}^{\mathcal{A}}$, where y is the output of $f(x_\ell)$ for some x_ℓ:

$\mathcal{R}^{\mathcal{A}}(y, 1^n)$

$y' \leftarrow\$ \{0,1\}^n$

$z \leftarrow \mathcal{A}(y||y', 1^{2n})$

return z

We now need to argue about two aspects of the reduction $\mathcal{R}^{\mathcal{A}}$:

- Efficiency: if $\mathcal{A}$ is PPT, then also $\mathcal{R}^{\mathcal{A}}$ is PPT.
- Success Probability: if

  $$\mathbf{Adv}^{\mathsf{PRG}}_{g,\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{PRG},0}_{g,s',\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{PRG},1}_{s',\mathcal{A}}(1^\lambda) = 1\right]\right|$$

  is non-negligible, then

  $$\mathbf{Adv}^{\mathsf{PRG}}_{f,\mathcal{R}^{\mathcal{A}}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{PRG},0}_{f,s,\mathcal{R}^{\mathcal{A}}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{PRG},1}_{s,\mathcal{R}^{\mathcal{A}}}(1^\lambda) = 1\right]\right|$$

  is non-negligible.

Efficiency This is clear. $\mathcal{R}^{\mathcal{A}}$ simply runs $\mathcal{A}$ and additionally samples an n bit long value y', which is a process that can be computed in polynomial time.

Success Probability

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{PRG}}_{g,s,\mathcal{R}^{\mathcal{A}}}(\lambda) &= \Pr_{x\leftarrow\$\{0,1\}^{\lambda}}\left[1 \leftarrow \mathcal{R}^{\mathcal{A}}(f(x),1^{\lambda})\right] - \Pr_{z\leftarrow\$\{0,1\}^{\lambda+s(\lambda)}}\left[1 \leftarrow \mathcal{R}^{\mathcal{A}}(z,1^{\lambda})\right] \\
&= \Pr_{x\leftarrow\$\{0,1\}^{\lambda},y'\leftarrow\$\{0,1\}^{\lambda}}\left[1 \leftarrow \mathcal{A}(f(x)||y',1^{2\lambda})\right] \\
&\quad - \Pr_{z\leftarrow\$\{0,1\}^{\lambda+s(\lambda)},y'\leftarrow\$\{0,1\}^{\lambda}}\left[1 \leftarrow \mathcal{A}(z||y',1^{2\lambda})\right] \\
&= \Pr_{x\leftarrow\$\{0,1\}^{\lambda},y'\leftarrow\$\{0,1\}^{\lambda}}\left[1 \leftarrow \mathcal{A}(g(x||y'),1^{2\lambda})\right] \\
&\quad - \Pr_{z'\leftarrow\$\{0,1\}^{2\lambda+s'(2\lambda)}}\left[1 \leftarrow \mathcal{A}(z',1^{2\lambda})\right] \\
&= \mathbf{Adv}^{\mathsf{PRG}}_{g,s,\mathcal{A}}(2\lambda)
\end{aligned}$$

1. In the first equality, we plug-in the definition of $\mathbf{Adv}^{\mathsf{PRG}}_{g,s,\mathcal{R}^{\mathcal{A}}}(\lambda)$.
2. In the second equality, we plug-in the definition of the reduction $\mathcal{R}^{\mathcal{A}}$. Note that, for the probability analysis, we also need to take into account the internal randomness of $\mathcal{R}^{\mathcal{A}}$.
3. Finally, for the first term of the third equality, we use the definition of g_f as given in the exercise description. For the second term, we use the fact that $s'(2\lambda) = s(\lambda)$. Thus, now if $\mathbf{Adv}^{\mathsf{PRG}}_{g,s,\mathcal{A}}$ is non-negligible, then $\mathbf{Adv}^{\mathsf{PRG}}_{g,s,\mathcal{R}^{\mathcal{A}}}$ is also non-negligible.

Exercise 2 (Some OWFs are not PRGs)**.** Assume the existence of length-preserving one-way functions.

Task: Show that there exists a length-expanding one-way function h which is not a PRG.

Solution 2. Let f be a length-preserving one-way function. Define

$$h_f(x) := 1||f(x)$$

We observe first that h_f indeed satisfies the length-expanding condition for being PRG with $s(\lambda) = 1$. This is since it expands a length λ string into a $\lambda + s(\lambda) = \lambda + 1$ length string. Therefore, to show that h_f is not a PRG it remains to assess its pseudorandomness property.

We obtain a PRG candidate which returns values of the form $h_f(x) = 1||f(x)$. In the following, we denote the output of h_f as z. Consider now the following adversary $\mathcal{A}$:

$\mathcal{A}\left(z, 1^{\lambda}\right)$
if $z_1 = 1$
  return 1
else
  return 0

Now, let us analyze the advantage of $\mathcal{A}$.

$$\mathbf{Adv}^{\mathsf{PRG}}_{h,\mathcal{A}}(1^{\lambda}) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{PRG},0}_{g,s,\mathcal{A}}(1^{\lambda}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{PRG},1}_{s,\mathcal{A}}(1^{\lambda}) = 1\right]\right|$$

For the first probability on the left, observe that

$$\Pr_{x \leftarrow\$ \{0,1\}^\lambda}\left[1 \leftarrow\$ \mathcal{A}(h(x), 1^\lambda)\right] = 1$$

since for all inputs $x \in \{0,1\}^\lambda$ we know that $z_1 = h(x)_1 = 1$. On the other hand, for the second probability on the left we have

$$\Pr_{y \leftarrow\$ \{0,1\}^{\lambda + s(\lambda)}}\left[1 \leftarrow\$ \mathcal{A}(y, 1^\lambda)\right] = \frac{1}{2}$$

because $z_1 = y_1 = 1$ for half of the elements in $\{0,1\}^{\lambda+1}$. Hence, plugging in these probabilities we yield

$$\mathbf{Adv}^{\mathsf{PRG}}_{h,\mathcal{A}}(1^\lambda) = |1 - \frac{1}{2}| = \frac{1}{2}$$

which is non-negligible. This shows the claim.

Exercise 3 (PRGs are OWFs)**.** Let g be a pseudorandom generator with $s(\lambda) := \lambda$, i.e., for all $x \in \{0,1\}^\lambda$, we have $|g(x)| = 2\lambda$.

Task: Prove that g is also a one-way function.

Solution 3. Let g be a pseudorandom generator with $|g(x)| = 2|x| = 2\lambda$. Let $\mathcal{A}$ be any PPT algorithm which tries to invert g. To show that g is one way, we need to prove that

$$\mathbf{Win}^{\mathsf{OW}}_{g,\mathcal{A}}(\lambda) = \Pr_{x \leftarrow\$ \{0,1\}^\lambda}\left[\mathcal{A}\left(g(x), 1^\lambda\right) \in g^{-1}(g(x))\right]$$

is negligible in λ.

Let $\mathcal{R}^{\mathcal{A}}$ be an algorithm, which with input $\left(g(x), 1^{|x|}\right)$ computes $x' = \mathcal{A}\left(g(x), 1^{|x|}\right)$ and returns 1, if and only if $g(x') = g(x)$. Formally, $\mathcal{R}^{\mathcal{A}}$ is defined as follows:

$\mathcal{R}^{\mathcal{A}}\left(g(x), 1^\lambda\right)$
$x' \leftarrow \mathcal{A}\left(g(x), 1^\lambda\right)$
if $g(x') = g(x)$
  return 1
else
  return 0

Let us analyze the distinguishing advantage of $\mathcal{R}^{\mathcal{A}}$ against randomly chosen $g(x)$ and randomly chosen x':

- Let $x \in \{0,1\}^\lambda$. Then $g(x)$ is always in the image of g, or in other words, x is always in the preimage of $g(x)$. Since $\mathcal{R}^{\mathcal{A}}$ returns 1 if and only if $\mathcal{A}\left(g(x), 1^\lambda\right) \in g^{-1}(g(x))$, the probability that $\mathcal{R}^{\mathcal{A}}$ returns 1 is the same as the probability that $\mathcal{A}$ finds an inverse. In terms of probability, this means

$$\Pr_{x \leftarrow\$ \{0,1\}^\lambda}\left[\mathcal{R}^{\mathcal{A}}\left(g(x), 1^\lambda\right) = 1\right] = \Pr_{x \leftarrow\$ \{0,1\}^\lambda}\left[\mathcal{A}\left(g(x), 1^\lambda\right) \in g^{-1}(g(x))\right].$$

– Since $|g(x)| = 2\lambda$, g maps a set of $|\{0,1\}^\lambda| = 2^\lambda$ strings to a set of $|\{0,1\}^{2\lambda}| = 2^{2\lambda}$ strings. Hence, we get

$$\begin{aligned}
&\Pr_{x' \leftarrow\$ \{0,1\}^{2\lambda}} \left[\mathcal{R}^{\mathcal{A}}\left(x', 1^\lambda\right) = 1\right] \\
=& \underbrace{\Pr_{x' \leftarrow\$ \{0,1\}^{2\lambda}} \left[\mathcal{R}^{\mathcal{A}}\left(x', 1^\lambda\right) = 1 | x' \in g(\{0,1\}^\lambda)\right]}_{\leq 1} \underbrace{\Pr_{x' \leftarrow\$ \{0,1\}^{2\lambda}} [x' \in g(\{0,1\}^\lambda)]}_{\leq 2^\lambda / 2^{2\lambda}} \\
&+ \underbrace{\Pr_{x' \leftarrow\$ \{0,1\}^{\lambda}} \left[\mathcal{R}^{\mathcal{A}}\left(x', 1^\lambda\right) = 1 | x' \notin g(\{0,1\}^n)\right]}_{=0} \Pr_{x' \leftarrow\$ \{0,1\}^{2\lambda}} [x' \notin g(\{0,1\}^\lambda)] \\
\leq& 2^{-\lambda},
\end{aligned}$$

which is negligible in λ.

Now we are ready to estimate the distinguishing advantage of $\mathcal{R}^{\mathcal{A}}$:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{PRG}}_{g,\mathcal{R}^{\mathcal{A}}}(\lambda) &= \left| \Pr_{x \leftarrow\$ \{0,1\}^\lambda} [\mathcal{R}^{\mathcal{A}}(g(x), 1^\lambda) = 1] - \Pr_{x' \leftarrow\$ \{0,1\}^{2\lambda}} [\mathcal{R}^{\mathcal{A}}(x', 1^\lambda) = 1] \right| \\
&\overset{(*)}{\geq} \Pr_{x \leftarrow\$ \{0,1\}^n} \left[\mathcal{A}\left(g(x), 1^{|x|}\right) \in g^{-1}(g(x))\right] - 2^{-\lambda}. \\
&\iff \Pr_{x \leftarrow\$ \{0,1\}^n} \left[\mathcal{A}\left(g(x), 1^{|x|}\right) \in g^{-1}(g(x))\right] \leq \mathsf{Adv}^{\mathsf{PRG}}_{\mathsf{g},\mathcal{R}^{\mathcal{A}}}(\mathsf{n}) + 2^{-\lambda}.
\end{aligned}$$

Since g is a pseudorandom generator, the distinguishing advantage is negligible in n. Hence, $\mathbf{Win}^{\mathsf{OW}}_{g,\mathcal{A}}(\lambda)$ is bounded from above by the sum of two negligible functions, which is also negligible (see Exercise Sheet 3). Therefore $\mathbf{Win}^{\mathsf{OW}}_{g,\mathcal{A}}(\lambda)$ is negligible, and we can conclude that g is a one-way function.

$(*)$: This inequality follows from the reverse triangle inequality, since probabilites are non-negative by definition.

Exercise 4. (Universal Hardcore Bit I) A universal hardcore bit is a function $b : \{0,1\}^* \to \{0,1\}$ such that b is a hardcore bit for *all* one-way functions f. This and the next exercise show that such a universal hardcore bit cannot exist. Namely: Assume towards contradiction that there exists a universal hardcore bit b. Assume that there exists a one-way function f. Consider the one-way function $h_{b,f}$ defined as $h_{b,f}(x) := f(x)||b(x)$.

Task: Show via reduction that if f is a one-way function, then $h_{b,f}$ is a one-way function, too.

Solution 4. We provide the solution for Exercise 4 together with the solution for Exercise 5 below.

Exercise 5. (Universal Hardcore Bit II) As in the previous exercise, consider a (candidate) universal hardcore bit $b : \{0,1\}^* \to \{0,1\}$ and a one-way function f. Again, we look at the one-way function $h_{b,f}$ defined as $h_{b,f}(x) := f(x)||b(x)$.

Task: Show that b is not a hardcore bit for $h_{b,f}$.

Solution 5. We provide the solution for both Exercises 4 and 5 together.

Assume, towards contradiction, that b is a universal hard-core bit. Assume there exists a one-way function f. We first need to show that the function $h_{b,f}$ is also a one way function.

Proof. To show that the one-wayness of $h_{b,f}$ follows from the one-wayness of f, we assume towards contradiction that there exists an adversary against $h_{b,f}$ and then transform it into an adversary against f. In other words, given $\mathcal{A}_h$ against $h_{b,f}$, we construct $\mathcal{A}_f$ such that

$$\mathbf{Win}^{\mathsf{OW}}_{h_{b,f},\mathcal{A}_f} \leq 2\mathbf{Win}^{\mathsf{OW}}_{f,\mathcal{A}_h}$$

Consider the reduction:

$\underline{\mathcal{A}_f(y, 1^{|y|})}$
$d \leftarrow\$ \{0,1\}$
$x \leftarrow\$ \mathcal{A}_h(y||d, 1^{|y|})$
return x

Note that $\mathcal{A}_f$ runs in polynomial time, since $\mathcal{A}_h$ runs in poly-time.

Probability Analysis. First note that

$$\begin{aligned} h^{-1}_{b,f}(h_{b,f}(x)) &= \{y : h_{b,f}(y) = h_{b,f}(x)\} \\ &= \{y : f(y)||b(y) = f(x)||b(x)\} \\ &\subseteq \{y : f(y) = f(x)\} = f^{-1}(f(x)). \end{aligned}$$

which helps us in our analysis:

$$\begin{aligned} \mathbf{Win}^{\mathsf{OW}}_{f,\mathcal{A}_f} &= \Pr_{x\leftarrow\$\{0,1\}^\lambda}\left[\mathcal{A}_f(f(x), 1^{|x|}) \in f^{-1}(f(x))\right] \\ &= \Pr_{x\leftarrow\$\{0,1\}^\lambda, d\leftarrow\$\{0,1\}}\left[\mathcal{A}_h(f(x)||d, 1^\lambda) \in f^{-1}(f(x))\right] \\ &\geq \Pr_{x\leftarrow\$\{0,1\}^\lambda, d\leftarrow\$\{0,1\}}\left[\mathcal{A}_h(f(x)||d, 1^\lambda) \in h^{-1}_{b,f}(h_{b,f}(x))\right] \\ &\geq \Pr_{x\leftarrow\$\{0,1\}^\lambda}\left[\mathcal{A}_h(f(x)||b(x), 1^\lambda) \in h^{-1}_{b,f}(h_{b,f}(x))\right] \cdot \underbrace{\Pr_{d\leftarrow\$\{0,1\}}[d = b(x)]}_{=1/2} \\ &= \frac{1}{2}\Pr_{x\leftarrow\$\{0,1\}^\lambda}\left[\mathcal{A}_h(h_{b,f}(x)), 1^\lambda) \in h^{-1}_{b,f}(h_{b,f}(x))\right] \\ &= \frac{1}{2}\mathbf{Win}^{\mathsf{OW}}_{h_{b,f},\mathcal{A}_h} \end{aligned}$$

Here we first plug in the definition of $\mathcal{A}_f$, then estimate from above since $b(x)$ is always either 0 or 1 (we move to a rarer case, thus the probability becomes smaller or equal).

Now, we have shown that $\mathbf{Win}^{\mathsf{OW}}_{h_{b,f},\mathcal{A}_f} \leq \frac{1}{2}\mathbf{Win}^{\mathsf{OW}}_{f,\mathcal{A}_h}$. Since f is one-way, $\mathbf{Win}^{\mathsf{OW}}_{f,\mathcal{A}_f}$ is negligible. Since $\frac{1}{2}\mathbf{Win}^{\mathsf{OW}}_{h_{b,f},\mathcal{A}_h}$ is smaller (or equal) to a negligible function, $\frac{1}{2}\mathbf{Win}^{\mathsf{OW}}_{h_{b,f},\mathcal{A}_h}$ is negligible. Since negligible function multiplied by a constant remains negligible, then also $2\frac{1}{2}\mathbf{Win}^{\mathsf{OW}}_{h_{b,f},\mathcal{A}_h} = \mathbf{Win}^{\mathsf{OW}}_{h_{b,f},\mathcal{A}_h}$ is negligible. Hence, $h_{b,f}$ is one-way. □

Now, we build an efficient adversary $\mathcal{A}$ such that

$$\mathbf{Win}^{\mathsf{HB}}_{h_{b,f},b,\mathcal{A}}(\lambda) = \big| \Pr_{x \leftarrow\$ \{0,1\}^n}[1 \leftarrow\$ \mathcal{A}(h_{b,f}(x), b(x), 1^n)] - \Pr_{x \leftarrow\$ \{0,1\}^n, z \leftarrow\$ \{0,1\}}[1 \leftarrow\$ \mathcal{A}(h_{b,f}(x), z, 1^n)] \big|$$

is non-negligible.

We recall that in the security definition of the hardcore bit, an adversary receives the output of the one way function (denoted as y) as input together with a value b, which is either a hardcore bit or a bit chosen uniformly at random. Now, by definition, the last bit of $h_{b,f}(x) = b(x)$. Then, we can choose, for example, an adversary $\mathcal{A}(y, b, 1^n)$ that compares the last bit of y and b and returns 1 if they are equal and 0 else. Now, $\mathbf{Win}^{\mathsf{HB}}_{h_{b,f},b,\mathcal{A}}(\lambda) = |1 - 1/2| = 1/2$, which is non-negligible. But this is a contradiction, since b was supposed to be a universal hard-core bit.

Hence, a universal hard-core bit cannot exist.

Exercise 6. (Goldreich-Levin Hard-Core Bit) Let f_{base} be a polynomial-time computable function, and let f be defined (on even-lengths inputs) as $f(x||r) := f_{\text{base}}(x)||r$, where $|x| = |r|$. Assume that there is a PPT algorithm $\mathcal{A}$ using $p(n)$ random bits such that

$$\Pr[\mathcal{A}(f(x||r), 1^n) = b_{GL}(x, r)] = 1, \tag{1}$$

where the probability is over sampling $x \leftarrow\$ \{0,1\}^n$, $r \leftarrow\$ \{0,1\}^n$ and the randomness of the adversary $\mathcal{A}$.

Task: Construct a PPT algorithm $\mathcal{R}$ that uses $\mathcal{A}$ to invert f_{base} on even-length inputs.

Hint. The algorithm $\mathcal{R}$ uses $\mathcal{A}$ to recover x bitwise.

Advanced Task (optional): Does your reduction $\mathcal{R}$ still work if

$$\Pr[\mathcal{A}(f(x||r), 1^n) = b_{GL}(x, r)] = 1 - \mathsf{negl}(n). \tag{2}$$

If so, analyse why. If no, explain why not. Can you come up with a reduction which works in the case (2), where the success probability is only $1 - \mathsf{negl}(n)$ instead of 1?

Solution 6. Let e^n_i denote an n-bit string whose ith bit is 1 and all other bits are 0. When taking the inner product of e^n_i and some other n-bit string z, then one obtains the ith bit $z[i]$ of z. If one has an algorithm that computes those bits for the first half of a pre-image of a value y under f, then one can recover the first half of a pre-image bit-by-bit. As f leaks the second half of its input anyway, one can then combine the first and second half. We now make this high-level discussion rigorous.

Reduction: Let e^n_i denote an n-bit string such that the jth bit $e^n_i[j]$ is 1 if $j = i$ and 0, else. Note that in the first line of the reduction below, we perform the parsing by counting the last n bits. This is, because we do not necessarily know the output length of what f produces. However, we know that r is exactly n bits long.

$\mathcal{R}^{\mathcal{A}}\left(y, 1^{2n}\right)$

$r \leftarrow y[|y| - n + 1..|y|]$

$w \leftarrow y[1..|y| - n]$

`For` i `from` 1 `to` n `do:`

  $z[i] \leftarrow \mathcal{A}\left(w||e_i^n, 1^n\right)$

return $z||r$

We now analyze the probability that $\mathcal{R}$ inverts f given an adversary $\mathcal{A}$ for which the equation in the exercise description holds.

$$
\begin{aligned}
&\Pr\left[\mathcal{R}^{\mathcal{A}}(f(x||r), 1^{2n}) \in f^{-1}(f(x||r))\right] \\
=&\Pr\left[\mathcal{R}^{\mathcal{A}}(f_{base}(x)||r, 1^{2n}) \in f^{-1}(f_{base}(x)||r)\right] \\
=&\Pr\left[\mathcal{R}^{\mathcal{A}}(f_{base}(x)||r, 1^{2n}) \in f_{base}^{-1}(f_{base}(x)) \times \{r\}\right] \\
=&\Pr\left[\mathcal{A}(f_{base}(x)||e_1^n, 1^{2n})||..||\mathcal{A}(f_{base}(x)||e_n^n, 1^{2n})||r \in f_{base}^{-1}(f_{base}(x)) \times \{r\}\right] \\
=&\Pr\left[b(x||e_1^n)||..||b(x||e_n^n)||r \in f_{base}^{-1}(f_{base}(x)) \times \{r\}\right] \\
=&\Pr\left[x[1]||..||x[n]||r \in f_{base}^{-1}(f_{base}(x)) \times \{r\}\right] \\
=&\Pr\left[x||r \in f_{base}^{-1}(f_{base}(x)) \times \{r\}\right] \\
=&1
\end{aligned}
$$

For the sake of clarification, we have dropped all the subscripts from the probabilities, since they are all taken over independently sampling $x \leftarrow\$ \{0,1\}^n$ and $r \leftarrow\$ \{0,1\}^n$.

1. In the second step, we plug in the definition of f.
2. In the third step, we use the definition of f to describe the pre-image set more concisely. Namely, $x'||r'$ is a pre-image of $f_{base}(x)||r$ if $r' = r$ and $f_{base}(x') = f_{base}(x)$, i.e., $x' \in f_{base}^{-1}(f_{base}(x))$.
3. In the fourth step, we now plug-in the definition of the reduction $\mathcal{R}$ that outputs the second half of its input $f_{base}(x)||r$ as the second half of its output and computes the other bits $z[i]$ of the output by running $\mathcal{A}$ on $(f_{base}(x)||e_i^n)$.
4. In the fifth step, we use that (1) holds for $\mathcal{A}$ and thus $\mathcal{A}(f_{base}(x)||e_1^n, 1^{2n}) = b(x||e_1^n)$.
5. In the sixth step, we use the definition of the Goldreich-Levin hardcore bit (we argue about it shortly).
6. In the seventh step, we replace the notation of the concatenation of the individual bits of x by x, and the eighth step follows since x is a pre-image of $f_{base}(x)$.

We now provide the missing argument for the Goldreich-Levin hardcore bit:

$$
\begin{aligned}
b(x||e_1^i) &= \bigoplus_{j=1}^{|x|} x[j] \cdot e_1^i[j] \\
&= (\bigoplus_{j=1}^{i-1} x[j] \cdot 0) \oplus x[i] \cdot 1 \oplus (\bigoplus_{j=i+1}^{|x|} x[j] \cdot 0) \\
&= 0 \oplus x[i] \oplus 0 \\
&= x[i]
\end{aligned}
$$

The first equation is the definition of the Goldreich-Levin hardcore bit. The second equation splits the $\oplus$ into the terms with $j < i$ and $j = i$ and $j > i$. For $j = i$, $e_1^i[j] = 1$ for $j < i$ and $j > i$, $e_1^i[j] = 0$. Thus, the $\bigoplus$ from $j = 1$ to $i - 1$ only sums over zeroes. We thus replace it by a 0 in the third equation. The same applies to the $\bigoplus$ from $j = i + 1$ to $|x|$.

Exercise 7 (Crocodile Experiment)**.** Give a function $f_{10} : \{1, .., 10\} \to \{1, .., 10\}$ which predicts the next number in the crocodile experiment in 80% of the data (based on the experimental data given in the materials section of the MyCourses page). (Doing this is considered solving the exercise, but here are further questions to consider if you like to:) Try to use less than 10 previous results, say, use only 5. Can you give a very simple prediction function f_3 or f_2 or even f_1 which is less accurate and takes only the last 3, 2, 1 numbers, respectively? Which reflections come to mind in terms of whether a more complex or a more simple function describes the experiment "better" (for this, we need a notion of what a "better" description is)? What is the easiest *distinguisher* which you can come up with for this experiment? Finally, assume that you are provided with a second crocodile. Can your function f_{10} be used for predicting the outputs of the second crocodile as well? On this line, is it possible to extend the function in such way that it includes a data collection loop at the beginning?

Solution 7. (*Credits to Miro Aurela for the solution!)*

The idea of this solution is to first transform the second data set to a new one, so that it can be paired with the first one. This is done by the mapping

$$x \mapsto 11 - x$$

As a result, triggering the trap on tooth number 10 would then correspond to the trigger going off on the first press, tooth number 9 would correspond to triggering on the second press and so forth. After this, we can merge all the observations to one data set.

Next, with the help of Python, we can write a script that counts the frequency of each number in $\{1, \ldots, 10\}$ for each given distinct sequence in the data set. In another words, we create a dictionary

$$\texttt{(Seq, Int)} \mapsto \texttt{Int}$$

Here the key consists of a sequence of given length together with the observed tooth trigger and the value is the amount of times it was observed.

We then define a function

$$f_n : \{1, \ldots, n\}^n \to \{1, \ldots, 10\}$$

which takes a sequence of length n and outputs a number from 1 to 10 based on the observations in the data set. The best guess for a given sequence of length n is chosen by the largest largest value for a given pair of key `(Seq, Int)`.

It was found, based on the data, that this method could not predict correctly 80% of the time, when $n = 10$ (this hints that for long sequences, predicting is hard). However, for $n = 4$, the method succeeded correctly on $\frac{231}{256} \approx 90\%$ of the observed sequences. For $n = 3$, the prediction power drops to roughly 70%.
For $n = 4$, the function

$$f_4(x) : \{1, \ldots, 10\}^4 \to \{1, \ldots, 10\}$$

is defined as follows:

$$f_4(x) = \begin{cases} 1, & x \in \big\{(6,1,1,5),(5,1,4,7),(5,7,3,4),(1,2,6,8),(2,6,8,1),(3,6,2,5),\\ & (2,5,1,2),(2,1,1,6),(1,1,6,1),(5,2,4,6),(6,1,4,5),(4,5,1,6),(4,9,2,5),\\ & (1,2,4,6),(6,1,9,2),(3,7,2,5),(2,5,1,3),(1,9,5,3),(5,3,1,5),(3,1,5,1),\\ & (4,4,5,3),(4,5,3,1),(4,2,6,3),(1,7,6,3),(5,3,1,3),(3,1,3,1),(2,9,5,2),\\ & (5,2,5,3),(1,8,5,2),(5,2,1,4),(4,2,1,2),(2,1,5,2)\big\}\\ 2, & x \in \big\{(5,2,5,9),(2,1,4,8),(7,1,3,5),(5,2,6,9),(6,9,2,5),(4,3,4,8),\\ & (7,1,4,5),(7,3,4,1),(8,1,1,5),(8,2,4,5),(5,9,3,6),(3,5,3,6),\\ & (5,1,4,6),(3,7,1,4),(4,2,6,9),(6,1,3,1),(3,4,9,3),(3,2,5,1),\\ & (5,7,3,5),(5,7,2,5),(3,6,4,9),(9,2,5,1),(4,6,1,9),(2,1,3,7),\\ & (1,1,6,3),(1,6,3,2),(1,1,6,4),(3,1,3,5),(9,5,3,5),(1,6,2,6),\\ & (6,2,9,5),(1,3,9,5),(3,1,8,5),(4,1,6,4),(6,4,2,1),(1,2,1,5)\big\}\\ 3, & x \in \big\{(1,4,7,1),(7,1,4,4),(8,2,4,6),(6,2,6,9),(6,9,3,5),(9,1,5,1),\\ & (9,2,4,1),(1,3,6,1),(2,6,1,5),(3,6,1,4),(4,3,4,9),(4,2,5,7),\\ & (5,1,6,1),(1,9,2,1),(7,2,5,1),(3,1,9,5),(5,1,1,6),(8,4,4,5),\\ & (6,4,2,6),(3,1,7,6),(7,6,3,1),(2,2,9,5),(5,2,9,5),(9,5,2,1),\\ & (9,5,2,5)\big\}\\ 4, & x \in \big\{(5,9,2,1),(4,8,3,1),(2,5,7,1),(5,7,1,4),(1,4,4,3),(3,4,8,2),\\ & (1,4,6,2),(2,1,4,9),(4,9,1,5),(5,1,3,7),(6,9,2,4),(4,1,3,6),\\ & (3,6,1,3),(3,1,2,6),(1,5,3,6),(5,3,6,1),(5,1,3,6),(9,3,2,5),\\ & (1,1,5,1),(2,5,7,3),(6,9,2,1),(1,4,8,2),(6,2,5,1),(4,6,2,1),\\ & (1,3,7,1),(6,1,4,3),(5,1,2,1),(7,2,5,2),(2,4,6,1),(6,1,3,6),\\ & (3,2,2,8),(2,2,8,4),(3,1,1,6),(8,5,2,1),(1,4,1,6)\big\}\\ 5, & x \in \big\{(4,7,1,3),(2,6,9,2),(9,2,5,2),(2,4,6,3),(5,6,1,1),(4,7,1,4),\\ & (1,4,5,2),(6,8,1,1),(2,4,5,2),(2,6,9,3),(5,3,6,2),(1,4,9,1),\\ & (1,2,6,1),(3,6,1,1),(4,9,3,2),(8,2,4,2),(7,3,5,2),(1,6,1,1),\\ & (2,5,7,2),(4,6,1,4),(6,4,9,2),(1,3,7,2),(1,3,1,9),(9,5,3,1),\\ & (2,8,4,4),(6,3,1,3),(5,2,2,9),(2,9,5,3),(3,5,2,9),(2,6,2,9)\\ & (2,1,3,9),(3,9,5,2),(5,3,1,8),(2,1,2,1),(2,1,2,9)\big\}\\ 6, & x \in \big\{(1,3,5,2),(4,8,2,4),(4,6,3,5),(3,4,1,2),(1,1,5,2),(2,5,9,3),\\ & (9,3,6,2),(9,3,5,3),(2,5,1,4),(7,1,4,2),(2,4,1,3),(1,3,1,2),\\ & (6,1,5,3),(9,2,1,1),(2,5,2,4),(1,4,5,1),(1,6,1,3),(5,1,2,4),\\ & (1,5,1,1),(5,3,1,1),(1,6,4,2),(6,3,1,7),(1,3,1,1),(1,1,6,2),\\ & (2,1,4,1)\big\}\\ 7, & x \in \big\{(8,3,1,4),(1,5,1,4),(4,5,2,5),(1,5,1,3),(2,4,2,5),(1,5,2,5),\\ & (9,2,1,3),(2,6,3,1)\big\}\\ 8, & x \in \big\{(9,2,1,4),(4,4,3,4),(4,1,2,6),(1,2,1,4),(6,3,2,2),(2,5,3,1)\big\}\\ 9, & x \in \big\{(2,5,2,5),(3,5,2,6),(1,5,2,6),(3,6,2,6),(6,2,1,4),(1,4,2,6),\\ & (1,4,3,4),(3,5,2,5),(1,3,6,4),(5,1,3,1),(3,5,2,2),(5,3,5,2),\\ & (6,2,6,2),(5,2,1,3),(5,2,1,2)\big\}\\ 10, & x \in \emptyset \end{cases}$$

Note that this solution would not be guaranteed to work in the case where we added a second crocodile to the experiment, due to the approach that we used. Also, given more data, the function can be updated accordingly to create better predictions.