7 Pseudorandom Permutations

A pseudorandom function behaves like a randomly chosen function. In particular, a randomly chosen function need not have a mathematical inverse. But very often in cryptography it is convenient to have cryptographically interesting algorithms which can be efficiently inverted.

A **pseudorandom permutation (PRP)** is essentially a PRF that is invertible (when given the key). Since the PRP is known to be a permutation, we do not require it to be indistinguishable from a randomly chosen function. Rather, we require it to be indistinguishable from a randomly chosen *permutation.*

7.1 Definitions

Definition 7.1 (PRP syntax)

*Let $F, F^{-1} : \{0,1\}^\lambda \times \{0,1\}^{blen} \to \{0,1\}^{blen}$ be deterministic and efficiently computable functions. Then F is a **pseudorandom permutation (PRP)** if for all keys $k \in \{0,1\}^\lambda$, and for all $x \in \{0,1\}^{blen}$, we have:*

$$F^{-1}(k, F(k,x)) = x.$$

*That is, if $F^{-1}(k,\cdot)$ is the inverse of $F(k,\cdot)$. In particular, this implies that $F(k,\cdot)$ is a permutation on $\{0,1\}^{blen}$ for every key k. We refer to blen as the **blocklength** of F and any element of $\{0,1\}^{blen}$ as a **block**.*

Outside of the world of academic cryptography, pseudorandom permutations are typically called **block ciphers**. We will use both terms interchangeably.

As mentioned above, our security definition will require a PRP to be indistinguishable from a randomly chosen *permutation* on $\{0,1\}^{blen}$, rather than a randomly chosen *function* from $\{0,1\}^{blen}$ to itself. In practical terms, we must modify the libraries that define PRF security.

As with PRFs, we could consider sampling a permutation on $\{0,1\}^{blen}$ uniformly at random, but this requires exponential time when n is large. Instead, we will sample the truth table of a random permutation on-the-fly, and store the mappings in an associative array T in case a query is repeated. This is the same approach taken in our PRF definition.

However, we must ensure that we always sample truth table values that are consistent with a permutation. It suffices to ensure that the function is *injective.* Injectivity is guaranteed by making sure that no two inputs map to the same output. Concretely, when we sample the value of the function at input x, we choose it uniformly among the *set of values not used as function outputs so far.* This ensures that the library's behavior is consistent with a random permutation. The formal details are below:

Definition 7.2 (PRP security)

Let $F : \{0,1\}^\lambda \times \{0,1\}^{blen} \to \{0,1\}^{blen}$ be a deterministic function. For an associative array T, define:

$$\text{range}(T) \overset{\text{def}}{=} \{v \mid \exists x : T[x] = v\}.$$

We say that F is a ***secure PRP*** *if* $\mathcal{L}^F_{\text{prp-real}} \approx \mathcal{L}^F_{\text{prp-rand}}$, *where:*

```
$\mathcal{L}^F_{\text{prp-real}}$
$k \leftarrow \{0,1\}^\lambda$

QUERY($x \in \{0,1\}^{blen}$):
  return $F(k,x)$
```

```
$\mathcal{L}^F_{\text{prp-rand}}$
$T$ := empty assoc. array

QUERY($x \in \{0,1\}^{blen}$):
  if $T[x]$ undefined:
    $T[x] \leftarrow \{0,1\}^{blen} \setminus \text{range}(T)$
  return $T[x]$
```

The changes from the PRF definition are highlighted in yellow. In particular, the $\mathcal{L}_{\text{prp-real}}$ and $\mathcal{L}_{\text{prf-real}}$ libraries are identical.

7.2 Switching Lemma

We have modified $\mathcal{L}_{\text{prf-rand}}$ to provide access to a random permutation rather than a random function. How significant is the change that we made?

Consider the $\mathcal{L}_{\text{prf-rand}}$ library implementing a function whose domain is $\{0,1\}^\lambda$. This domain is so large that a calling program can see only a *negligible fraction* of the function's behavior. In that case, it seems unlikely that the calling program could tell the difference between a random *permutation* and a random (arbitrary) *function*.

More formally, we expect that $\mathcal{L}_{\text{prf-rand}}$ (which realizes a random function) and $\mathcal{L}_{\text{prp-rand}}$ (which realizes a random permutation) will be indistinguishable, when the parameters are such that their interfaces are the same ($blen = in = out = \lambda$). This is indeed true, as the following lemma demonstrates.

Lemma 7.3 (PRP switching) *Let* $\mathcal{L}_{\text{prf-rand}}$ *and* $\mathcal{L}_{\text{prp-rand}}$ *be defined as in Definitions 6.1 & 7.2, with parameters* $in = out = blen = \lambda$. *Then* $\mathcal{L}_{\text{prf-rand}} \approx \mathcal{L}_{\text{prp-rand}}$.

Proof Recall the replacement-sampling lemma, Lemma 4.7, which showed that the following libraries are indistinguishable:

```
$\mathcal{L}_{\text{samp-L}}$
SAMP():
  $r \leftarrow \{0,1\}^\lambda$
  return $r$
```

```
$\mathcal{L}_{\text{samp-R}}$
$R := \emptyset$

SAMP():
  $r \leftarrow \{0,1\}^\lambda \setminus R$
  $R := R \cup \{r\}$
  return $r$
```

$\mathcal{L}_{\textsf{samp-L}}$ samples values with replacement, and $\mathcal{L}_{\textsf{samp-R}}$ samples values without replacement. Now consider the following library $\mathcal{L}^*$:

```
$\mathcal{L}^*$
$T$ := empty assoc. array

QUERY($x \in \{0,1\}^\lambda$):
  if $T[x]$ undefined:
    $T[x] \leftarrow$ SAMP()
  return $T[x]$
```

When we link $\mathcal{L}^* \diamond \mathcal{L}_{\textsf{samp-L}}$ we obtain $\mathcal{L}_{\textsf{prf-rand}}$ since the values in $T[x]$ are sampled uniformly. When we link $\mathcal{L}^* \diamond \mathcal{L}_{\textsf{samp-R}}$ we obtain $\mathcal{L}_{\textsf{prp-rand}}$ since the values in $T[x]$ are sampled uniformly subject to having no repeats. Then from Lemma 4.7, we have:

$$\mathcal{L}_{\textsf{prf-rand}} \equiv \mathcal{L}^* \diamond \mathcal{L}_{\textsf{samp-L}} \approx \mathcal{L}^* \diamond \mathcal{L}_{\textsf{samp-R}} \equiv \mathcal{L}_{\textsf{prp-rand}},$$

which completes the proof. ■

Discussion

- It is possible to define PRFs/PRPs and random functions over domains of any size. However, the switching lemma applies only when the domains are sufficiently large — in this case, at least λ bits long. This comes from the fact that $\mathcal{L}_{\textsf{samp-L}}$ and $\mathcal{L}_{\textsf{samp-R}}$ in the proof are indistinguishable only when dealing with long (length-λ) strings (look at the proof of Lemma 4.7 to recall why).

  The exercises ask you to show that random permutations over *small domains* can be distinguished from random (arbitrary) functions.

- The switching lemma refers only to the idealized $\mathcal{L}_{\textsf{prf-rand}}$ and $\mathcal{L}_{\textsf{prp-rand}}$ libraries. But a *secure* PRF has behavior that is indistinguishable to that of $\mathcal{L}_{\textsf{prf-rand}}$, and similarly a secure PRP to that of $\mathcal{L}_{\textsf{prp-rand}}$. That means that a secure PRF with $\textit{in} = \textit{out} = \lambda$ will have outputs that are indistinguishable from those of a secure PRP with $\textit{blen} = \lambda$.

7.3 Feistel Ciphers

A PRF takes an input x and "scrambles" it to give a pseudorandom output. A PRP asks for more: there should be a way to "unscramble" the result and recover the original x. This seems much harder! Nevertheless, in this section we will see an elegant and simple way to transform a (not necessarily invertible) PRF into a PRP!

Definition 7.4 (Feistel transform) *Let $f : \{0,1\}^n \to \{0,1\}^n$ be any function. The **Feistel transform of** f is the function $\textsc{fstl}_f : \{0,1\}^{2n} \to \{0,1\}^{2n}$ defined by:*

$$\textsc{fstl}_f(L,R) = (R, f(R) \oplus L).$$

We treat the $2n$-bit input and output of $\textsc{fstl}_f$ as a pair of n-bit inputs; thus L and R each have n bits.

Surprisingly, even though f need not be invertible, the Feistel transform of f is *always* invertible! See for yourself. Define $\textsc{fstl}_f^{-1}(X,Y) = (Y \oplus f(X), X)$. Then,

$$\begin{aligned}\textsc{fstl}_f^{-1}(\textsc{fstl}_f(L,R)) &= \textsc{fstl}_f^{-1}(R, f(R) \oplus L)\\ &= \big((f(R) \oplus L) \oplus f(R), R\big)\\ &= (L,R).\end{aligned}$$

In fact, $\textsc{fstl}_f$ and its inverse are essentially mirror images of each other: see Figures 7.1 and 7.2. The similarity of $\textsc{fstl}_f$ and $\textsc{fstl}_f^{-1}$ makes hardware and software implementations much simpler, as many components can be reused.

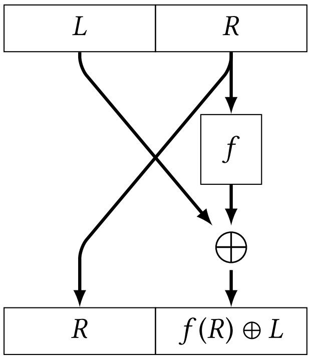

Figure 7.1: *Feistel transform of f.*

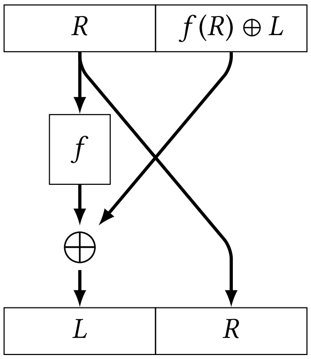

Figure 7.2: *Inverse Feistel transform of f.*

Feistel Ciphers

A **Feistel cipher** is a block cipher built by composing several Feistel transforms. For example, let $\circ$ denote composition of functions, so $f \circ g$ denotes the function $x \mapsto f(g(x))$. Then

$$\textsc{fstl}_{f_3} \circ \textsc{fstl}_{f_2} \circ \textsc{fstl}_{f_1}$$

is a 3-round Feistel cipher whose **round functions** are f_1, f_2, and f_3. Its inverse would then be:

$$\textsc{fstl}_{f_1}^{-1} \circ \textsc{fstl}_{f_2}^{-1} \circ \textsc{fstl}_{f_3}^{-1}.$$

Note how the round functions are reversed.

It is useful for the round functions of a Feistel cipher to be different. But rather than using totally unrelated functions, a typical Feistel cipher uses a single **keyed round function**; that is, the round function is the same for each round, but it takes an extra argument (a **round key**) which is different in each round.

For example, $F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ has the syntax of a keyed round function, where the first argument of F is the round key. For each round key k, the function $F(k,\cdot)$ maps $\{0,1\}^n$ to itself, so $F(k,\cdot)$ is a suitable round function. The following is a 3-round keyed Feistel cipher with (keyed) round function F (also Figures 7.3 & 7.4):

$$\textsc{fstl}_{F(k_3,\cdot)} \circ \textsc{fstl}_{F(k_2,\cdot)} \circ \textsc{fstl}_{F(k_1,\cdot)}.$$

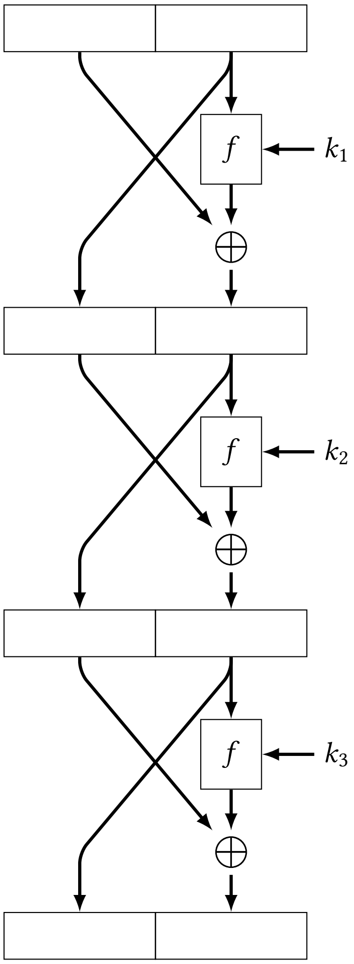

Figure 7.3: *A typical 3-round Feistel cipher with keyed round function. The key schedule is (k_1,k_2,k_3).*

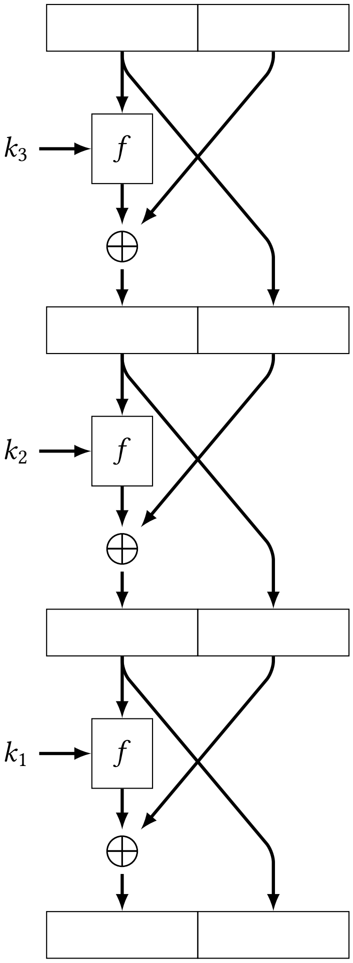

Figure 7.4: *The inverse of the Feistel cipher in Figure 7.3. Note that the key schedule is reversed.*

The sequence of keys k_1,k_2,k_3 is called the **key schedule** of the cipher. To invert this Feistel cipher, we take its mirror image and reverse the key schedule. Below is a procedural way to compute and invert an r-round keyed Feistel cipher C, with key schedule $k_1,\ldots,k_r$ and keyed round function F:

```
$C\big((k_1,\ldots,k_r),(L,R)\big)$:
  $V_{-1} := L$; $V_0 := R$
  for $i = 1$ to $r$:
    $V_i := F(k_i,V_{i-1}) \oplus V_{i-2}$
  return $(V_{r-1},V_r)$
```

```
$C^{-1}\big((k_1,\ldots,k_r),(L,R)\big)$:
  $V_{r-1} := L$; $V_r := R$
  for $i = r$ downto 1:
    $V_{i-2} := F(k_i,V_{i-1}) \oplus V_i$
  return $(V_{-1},V_0)$
```

Interestingly, when F is a secure PRF and k_1,k_2,k_3 are chosen uniformly and independently, such a 3-round Feistel cipher is a pseudorandom permutation (when the blocklength is large enough). In the exercises, you will show that 1 and 2 rounds do not lead to a secure PRP.

Theorem 7.5 (Luby-Rackoff) *Let $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a secure PRF with in = out = λ, and define $F^* : \{0,1\}^{3\lambda} \times \{0,1\}^{2\lambda} \to \{0,1\}^{2\lambda}$ as:*

$$F^*\Big((k_1,k_2,k_3),(L,R)\Big) \overset{\text{def}}{=} \mathit{FSTL}_{F(k_3,\cdot)}(\mathit{FSTL}_{F(k_2,\cdot)}(\mathit{FSTL}_{F(k_1,\cdot)}(L,R))).$$

Then F^* is a secure PRP.

to-do *The proof is just slightly trickier than what I would prefer including in the notes. I will think more about finding helpful indistinguishability lemmas that make the proof more manageable.*

The PRP in question has the following form:

```
$F^*\Big((k_1,k_2,k_3),(L,R)\Big)$:
  $X := F(k_1,R) \oplus L$
  $Y := F(k_2,X) \oplus R$
  $Z := F(k_3,Y) \oplus X$
  return $(Y,Z)$
```

The idea behind the proof is to show that F^* is a secure PRF. Then, because its blocklength is large enough, the switching lemma shows that it is also a secure PRP. The main idea is:

- *We can apply the PRF security to the 3 different PRF instances. The result is $T_1[R]$, $T_2[X]$ and $T_3[Y]$ in place of the PRF calls, where T_i are the truth tables of distinct random functions (sampled on the fly).*

- *Now it suffices to show that the calling program will never find distinct (L,R) pairs that lead to duplicate X or Y values. As long as that's the case, $T_2[X]$ and $T_3[Y]$ will always be uniformly random for distinct inputs to F^*.*

- *It is possible to show that the calling program cannot find (L,R) pairs that lead to the same X value. This is a quite subtle part, since it depends on X not being visible to the calling program.*

- *Given that X is distinct for all distinct inputs, it can then be shown that Y is distinct as well.*

Instantiations

The Data Encryption Standard (DES) was the most commonly used block cipher, until it was phased out in favor of the newer Advanced Encryption Standard (AES). DES was designed as a standard Feistel network with 16 rounds. Its blocklength is 64 bits, so the input/output length of the round function is 32 bits.

While we have theoretical results (like the one above) about the security of Feistel ciphers with *independent* round keys, Feistel ciphers in practice typically do not use independent keys. Rather, their round keys are derived in some way from a master key. DES follows this pattern, with each 48-bit round key being derived from a 56-bit **master key**. The reason for this discrepancy stems from the fact that an r-round Feistel network with *independent* keys will have an $r\lambda$-bit master key but give only λ bits of security. In practice, it is desirable to use a small λ-bit master key but derive from it many round keys in a way that "mixes" the entire master key into all of the rounds in some way.

7.4 Strong Pseudorandom Permutations

Although every PRP has an inverse, it is not necessarily true that a PRP and its inverse have comparable security properties (the exercises explore this idea further). Yet, it is sometimes very convenient to argue about pseudorandom behavior of F^{-1}. For instance, F^{-1} is used to decrypt ciphertexts in many encryption schemes that we will see, and unpredictability in F^{-1} is useful when an adversary generates invalid ciphertexts.

It is possible to insist that both F and F^{-1} are PRPs, which would mean that $\mathcal{L}_{\textsf{prp-real}}$ and $\mathcal{L}_{\textsf{prp-rand}}$ are indistinguishable when instantiated with F and when instantiated with F^{-1}. In doing so, we have some interactions where a distinguisher queries F and other interactions where a distinguisher queries F^{-1}. A stronger requirement would be to allow the distinguisher to query both F and F^{-1} in a *single* interaction. If a PRP is indistinguishable from a random permutation under that setting, then we say it is a **strong** PRP (SPRP). In our previous example, such security would be useful to model an adversary who can see encryptions of plaintexts (where encryption involves evaluating F) and who can also ask for decryptions of invalid ciphertexts (where decryption involves evaluating F^{-1}).

The security definition for an SPRP then considers two libraries. The common interface of these two libraries provides *two* subroutines: one for forward queries and one for reverse queries. In the $\mathcal{L}_{\textsf{sprp-real}}$ library, these subroutines are implemented by calling the PRP or its inverse accordingly. In the $\mathcal{L}_{\textsf{sprp-rand}}$ library, we would like to emulate the behavior of a randomly chosen permutation that can be queried in both directions. For convenience, we therefore maintain two associative arrays to maintain the forward and backward mappings. The formal details are below:

Definition 7.6 (SPRP security)

Let $F : \{0,1\}^\lambda \times \{0,1\}^{blen} \to \{0,1\}^{blen}$ be a deterministic function. We say that F is a ***secure***

strong pseudorandom permutation (SPRP) *if* $\mathcal{L}^F_{\text{sprp-real}} \approx \mathcal{L}^F_{\text{sprp-rand}}$, *where:*

```
$\mathcal{L}^F_{\text{sprp-real}}$
$k \leftarrow \{0,1\}^\lambda$

QUERY($x \in \{0,1\}^{blen}$):
  return $F(k,x)$

INVQUERY($y \in \{0,1\}^{blen}$):
  return $F^{-1}(k,y)$
```

```
$\mathcal{L}^F_{\text{sprp-real}}$
$T, T^{-1}$ := empty assoc. arrays

QUERY($x \in \{0,1\}^{blen}$):
  if $T[x]$ undefined:
    $y \leftarrow \{0,1\}^{blen} \setminus \text{range}(T)$
    $T[x] = y$; $T^{-1}[y] = x$
  return $T[x]$

INVQUERY($y \in \{0,1\}^{blen}$):
  if $T^{-1}[y]$ undefined:
    $x \leftarrow \{0,1\}^{blen} \setminus \text{range}(T^{-1})$
    $T^{-1}[y] = x$; $T[x] = y$
  return $T^{-1}[y]$
```

Earlier we showed that a 3-round Feistel network can be used to construct a PRP from a PRF. The resulting PRP is *not* a *strong* PRP. However, a 4-round Feistel network *is* a strong PRP! We present the following theorem without proof:

Theorem 7.7 (Luby-Rackoff) *Let* $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ *be a secure PRF with* $in = out = \lambda$, *and define* $F^* : \{0,1\}^{4\lambda} \times \{0,1\}^{2\lambda} \to \{0,1\}^{2\lambda}$ *as:*

$$F^*\Big((k_1,k_2,k_3,k_4),(L,R)\Big) \overset{\text{def}}{=} \mathit{FSTL}_{F(k_4,\cdot)}\big(\mathit{FSTL}_{F(k_3,\cdot)}\big(\mathit{FSTL}_{F(k_2,\cdot)}\big(\mathit{FSTL}_{F(k_1,\cdot)}(L,R)\big)\big)\big).$$

Then F^* *is a secure* ***strong*** *PRP.*

Exercises

7.1. Let F be a secure PRP with blocklength $blen = 128$. Then for each k, the function $F(k,\cdot)$ is a permutation on $\{0,1\}^{128}$. Suppose I choose a permutation on $\{0,1\}^{128}$ uniformly at random. What is the probability that the permutation I chose agrees with a permutation of the form $F(k,\cdot)$? Compute the probability as an actual number — is it a reasonable probability or a tiny one?

7.2. Suppose $R : \{0,1\}^n \to \{0,1\}^n$ is chosen uniformly among all such functions. What is the probability that there exists an $x \in \{0,1\}^n$ such that $R(x) = x$?

Hint: First find the probability that $R(x) \neq x$ for all x. Simplify your answer using the approximation $(1-y) \approx e^{-y}$.

7.3. In this problem, you will show that the PRP switching lemma holds only for large domains. Let $\mathcal{L}_{\text{prf-rand}}$ and $\mathcal{L}_{\text{prp-rand}}$ be as in Lemma 7.3. Choose any small value of $n = in = out$

that you like, and show that $\mathcal{L}_{\textsf{prf-rand}} \not\approx \mathcal{L}_{\textsf{prp-rand}}$ with those parameters. Describe a distinguisher and compute its advantage. *Hint:* remember that the distinguisher needs to run in polynomial time in λ, but not necessarily polynomial in n.

7.4. Let f be a (not necessarily invertible) function that takes in *in* bits of input and gives *out* bits of output. The Feistel transform introduced in the notes works *as long as that in = out.*

Describe a modification of the Feistel transform that works for round functions even where $\mathit{in} \neq \mathit{out}$, giving an invertible function with input/output length $\mathit{in} + \mathit{out}$. Be sure to show that your proposed transform is invertible!

7.5. Show that any function F that is a 1-round keyed Feistel cipher **cannot** be a secure PRP. That is, construct a distinguisher to demonstrate that $\mathcal{L}^F_{\textsf{prp-real}} \not\approx \mathcal{L}^F_{\textsf{prp-rand}}$, knowing only that F is a 1-round Feistel cipher. In particular, the purpose is to attack the Feistel transform and not the round function, so your attack should work no matter what the round function is.

7.6. Show that any function F that is a 2-round keyed Feistel cipher **cannot** be a secure PRP. As above, your distinguisher should work without knowing what the round functions are, and the attack should work with different (independent) round functions for the 2 rounds.

Hint: Make two queries, where the second query depends on the answer to the first. With carefully chosen queries, it is possible to identify a property that is always satisfied by a 2-round Feistel network but that is rarely satisfied by a random function.

7.7. Show that any function F that is a 3-round keyed Feistel cipher **cannot** be a secure *strong* PRP. As above, your distinguisher should work without knowing what the round functions are, and the attack should work with different (independent) round functions.

7.8. In this problem you will show that PRPs are hard to invert without the key (if the blocklength is large enough). Let F be a candidate PRP with blocklength $\mathit{blen} \geq \lambda$. Suppose there is a program $\mathcal{A}$ where:

$$\Pr_{y \leftarrow \{0,1\}^{\mathit{blen}}} \left[\mathcal{A}(y) \diamond \mathcal{L}^F_{\textsf{prf-real}} \text{ outputs } F^{-1}(k,y) \right] \text{ is non-negligible.}$$

In the above expression, $\mathcal{A}(y)$ indicates that $\mathcal{A}$ receives y as an input, and k refers to the private variable within $\mathcal{L}_{\textsf{prf-real}}$. So, when given the ability to evaluate F in the forward direction only (via $\mathcal{L}_{\textsf{prf-real}}$), $\mathcal{A}$ can invert a uniformly chosen block y.

Prove that if such an $\mathcal{A}$ exists, then F is not a secure PRP. Use $\mathcal{A}$ to construct a distinguisher that violates the PRP security definition. Where do you use the fact that $\mathit{blen} \geq \lambda$?

★ 7.9. Construct a PRP F such that F^{-1} is **not** a PRP.

Hint: Take an existing PRP and modify it so that $F(k,k) = 0^{\mathit{blen}}$. Argue that the result is still a PRP, and show an attack against the PRP security definition applied to F^{-1}.