9.1 Pseudorandom Functions

We now give an abstraction describing a main security goal of block ciphers. In this definition we will need the concept of an oracle for an algorithm. You can think about oracles intuitively as “subroutines” that an algorithm can call. When \( \mathcal{A} \) is an oracle algorithm and \( O_1 \) is a function, we write \( \mathcal{A}^{O_1} \) for \( \mathcal{A} \) connected to the oracle (subroutine) \( O_1 \). If \( O_2 \) is another oracle, we write \( \mathcal{A}^{O_2} \) for \( \mathcal{A} \) connected to \( O_2 \), and so on. A key point in this formalism is that \( \mathcal{A} \) can only observe the input/output behavior of its oracle, and not the code implementing the oracle. So if \( O_1 \) and \( O_2 \) are the same function, then \( \mathcal{A}^{O_1} \) and \( \mathcal{A}^{O_2} \) will behave exactly the same.

The following definition also needs the concept of a random function \( f : \{0,1\}^\ell \rightarrow \{0,1\}^\ell \). You can think of such a random variable as a table listing an output for every possible input, and such that the outputs are all uniformly random and independent. For a concrete example with \( \ell = 3 \), we might have \( f \) represented by the table:

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>100</td>
</tr>
<tr>
<td>001</td>
<td>011</td>
</tr>
<tr>
<td>010</td>
<td>000</td>
</tr>
<tr>
<td>011</td>
<td>101</td>
</tr>
<tr>
<td>100</td>
<td>011</td>
</tr>
<tr>
<td>101</td>
<td>111</td>
</tr>
<tr>
<td>110</td>
<td>001</td>
</tr>
<tr>
<td>111</td>
<td>110</td>
</tr>
</tbody>
</table>

So “picking a random function” means filling in the right side of the table with uniformly random and independent entries. These entries may repeat, but the outcome of one entry does not influence the outcome of any other entry.
Example 9.1. Let \( f \) be a random function from \( \{0,1\}^{128} \) to \( \{0,1\}^{128} \). Then the following hold:

\[
\Pr[ f(0^{128}) = 0^{128} ] = \frac{1}{2^{128}},
\]

\[
\Pr[ f(0^{128}) \text{ starts with } 0 ] = \frac{1}{2},
\]

\[
\Pr[ f(0^{128}) = f(1^{128}) ] = \frac{1}{2^{128}},
\]

\[
\Pr[ f(0^{128}) \oplus f(1^{128}) = 1^{128} ] = \frac{1}{2^{128}}.
\]

These can be proved rigorously by the independence and uniformity of \( f(0^{128}) \) and \( f(1^{128}) \) (or, one could actually count the fraction of all possible functions for which these hold).

Things are not always so simple. For instance,

\[
\Pr[ f( f(0^{128}) ) = 0^{128} ] = \frac{1}{2^{128}} + (1 - \frac{1}{2^{128}}) \frac{1}{2^{128}},
\]

which can be seen by applying the law of total probability to split up the probability into the cases where \( f(0^{128}) = 0^{128} \) and \( f(0^{128}) \neq 0^{128} \).

Exercise 9.1. Let \( \text{Func}(\{0,1\}^\ell,\{0,1\}^\ell) \) be the set of all functions from \( \{0,1\}^\ell \) to \( \{0,1\}^\ell \). This set has size \( 2^{2\ell^2} \). Using the table point of view on functions, or another method, explain this formula.

We can now state the definition, and afterwards interpret it.

Definition 9.1. Let \( E : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell \) and \( A \) be an adversary. Let \( K \) be uniform on \( \{0,1\}^n \), and let \( f \) be a random function from \( \{0,1\}^\ell \) to \( \{0,1\}^\ell \). We define the pseudorandom function (PRF) distinguishing advantage of \( A \) against \( E \) to be

\[
\text{Adv}_E^{\text{prf}}(A) = \left| \Pr[ A^{E(K,\cdot)} = 1 ] - \Pr[ A^{f(\cdot)} = 1 ] \right|.
\]

An adversary \( A \) can be judged according to its runtime, the number of queries it issues, and its advantage. We won’t distinguish much between runtime and queries, but in practice an adversary will usually have far more computation time than it does queries, because the queries need to be run by the parties under attack. They won’t be enabling anything like \( 2^{80} \) queries, but the adversary might have that much computation. If the advantage of every reasonable \( A \) is small (say \( 1/2^{64} \) or \( 1/2^{128} \), depending on the application), we informally say that \( E \) is a good PRF.

In that case we think of \( E \) as “essentially looking like a random function,” and all of the properties of random functions are thus inherited by \( E \). So \( E(K, x) \) and \( E(K, x') \) should look like totally random, independent strings, when \( x \neq x' \), even when they differ only by one bit. If not, then there would be an adversary with good PRF advantage.

Example 9.2. Suppose \( E : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128} \) satisfies \( E(k,0^{128}) = 0^{128} \) for every \( k \in \{0,1\}^{128} \). We show that \( E \) is not a good PRF. Consider \( A^O \) that queries \( 0^{128} \) to its oracle \( O \) and calls the response \( y \). If \( y = 0^{128} \) then \( A \) outputs 1, else 0. Then

\[
\Pr[ A^{E(K,\cdot)} = 1 ] = 1
\]

and

\[
\Pr[ A^{f(\cdot)} = 1 ] = \frac{1}{2^{128}},
\]
the latter because a random function satisfies \( f(0^{128}) = 0^{128} \) with probability \( \frac{1}{2^{128}} \). Thus

\[
\text{Adv}^\text{prf}_E(A) = 1 - \frac{1}{2^{128}},
\]

which is high.

Note that we had to define \( A \) with respect to a generic oracle \( \mathcal{O} \); We don’t get to say how \( A \) works with \( E(K, \cdot) \) and \( f(\cdot) \) separately, since \( A \) only gets to see their input/output behavior. More complicated patterns can be found with more clever attacks.

Example 9.3. Define \( E : \{0, 1\}^{128} \times \{0, 1\}^{128} \rightarrow \{0, 1\}^{128} \) by \( E(k, x) = k \oplus x \). We show that \( E \) is not a good PRF. Consider \( A^{\mathcal{O}} \) that fixes any \( x_1 \neq x_2 \in \{0, 1\}^{128} \), queries \( y_1 \leftarrow \mathcal{O}(x_1) \), and \( y_2 \leftarrow \mathcal{O}(x_2) \). If \( y_1 \oplus y_2 = x_1 \oplus x_2 \) then \( A \) outputs 1, else it outputs 0.

\[
\Pr[A^{E(k, \cdot)} = 1] = 1
\]

and

\[
\Pr[A^{f(\cdot)} = 1] \Pr[f(x_1) \oplus f(x_2) = x_1 \oplus x_2] = \frac{1}{2^{128}},
\]

the latter because a random function satisfies that equation with probability \( \frac{1}{2^{128}} \) (because \( f(x_1), f(x_2) \) are uniform and independent). Thus

\[
\text{Adv}^\text{prf}_E(A) = 1 - \frac{1}{2^{128}},
\]

which is high. Note that we needed two queries to break this one – It’s actually impossible to break it in one query!

Finally we address the need for oracles in the definition. It would be simpler to, say, provide a function table (like the one we diagrammed above) to \( A \), and ask it if the table represents \( E(K, \cdot) \) or \( f \). But this table would gigantic for the sizes we care about: \( 2^{128} \cdot 128 \) bits for AES. But no algorithm with realistic resources could even read its input in that case. We resort to oracles to allow “selective” access to parts of the function without paying a huge runtime.

### 9.2 Birthday Attacks

You may have noticed that in the definition of PRF advantage, when the oracle is \( E(K, \cdot) \), then the adversary will never see a repeated output for two distinct inputs. But when the oracle is \( f(\cdot) \) then two distinct inputs will produce the same output with non-zero probability. This suggests the following adversary, which attacks \( E : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^k \) and issues some number \( q \leq 2^\ell \) queries:

<table>
<thead>
<tr>
<th>Adversary ( \mathcal{A}^{\mathcal{O}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize an empty set ( S )</td>
</tr>
<tr>
<td>For ( i = 1, \ldots, q ):</td>
</tr>
<tr>
<td>\hspace{1em} Let ( x_i \in {0, 1}^\ell ) be the ( \ell )-bit encoding of ( i )</td>
</tr>
<tr>
<td>\hspace{1em} Query ( y_i \leftarrow \mathcal{O}(x_i) )</td>
</tr>
<tr>
<td>\hspace{1em} If ( y_i \in S ): Output 1</td>
</tr>
<tr>
<td>\hspace{1em} Add ( y_i ) to ( S )</td>
</tr>
<tr>
<td>Output 0</td>
</tr>
</tbody>
</table>
The set $S$ could be implemented with a hash table or another data structure that allows for fast membership testing and fast addition.

Let’s find the advantage of this adversary. First it is simple to show that

$$\Pr[A_E(K, \cdot) = 1] = 0$$

because the $y_i$ will all be distinct points, and hence will never be in $S$ in the for loop. This is because the $y_i$ are the outputs of a block cipher applied with the same key to distinct inputs. On the other hand,

$$\Pr[A_f(\cdot) = 1]$$

is less obvious to calculate. We will need the following interesting and well-known “birthday bound”.

**Theorem 1.** Let $N, q \geq 1$ be integers, and let $Col(N, q)$ be the probability that there is a repeated value (a collision) in $q$ independent uniform samples from a set of size $N$. Then

$$1 - \exp\left(-\frac{q(q-1)}{2N}\right) \leq Col(N, q) \leq \frac{0.5q(q-1)}{N}.$$  

If $q \leq \sqrt{2N}$, then this implies

$$0.3\frac{q(q-1)}{N} \leq Col(N, q) \leq 0.5\frac{q(q-1)}{N}.$$  

Above $\exp(x) = e^x$ is the usual exponential function from calculus.

**Exercise 9.2.** Prove the upper bound of the theorem via a union bound.


What is surprising about this bound is that the probability of a collision is approximately $q^2/N$ and not $q/N$. In particular, once $q$ is only a little larger than $\sqrt{N}$, the probability jumps up exponentially close 1 and is therefore almost certain. This results in some surprising phenomena, like the result of the following exercise

**Exercise 9.3.** Use the theorem to find the minimum number $q$ of people required to have a $1/2$ or greater chance of two people having the same birthday. Assume birthdays are uniformly random and independent values amongst the 365 days of the year.

Returning to our adversary, and assuming $q \leq \sqrt{2 \cdot 2^\ell}$, we have

$$\Pr[A_f(\cdot) = 1] \geq 0.3\frac{q(q-1)}{2^\ell},$$

because the $y_i$ values in the algorithm are all uniform and independent samples from a set of size $2^\ell$ when the oracle is a random function. Thus

$$\text{Adv}_{E}^{\text{prf}}(A) = \left| \Pr[A_E(K, \cdot) = 1] - \Pr[A_f(\cdot) = 1] \right|$$

$$= \left| 0 - \Pr[A_f(\cdot) = 1] \right|$$

$$= \Pr[A_f(\cdot) = 1]$$

$$\geq 0.3\frac{q(q-1)}{2^\ell}.$$
Thus, for \(\ell\)-bit outputs, we only need \(q \approx 2^{\ell/2}\), not \(q \approx 2^\ell\) queries, to have good PRF advantage! For AES, this means about \(2^{64}\) queries suffice, which is astronomically smaller than \(2^{128}\).

These attacks are called “birthday attacks,” and are avoidable if one insists on working with a block cipher (as we do). Thus we’ll aim for a block cipher to have only this defect, and remain a good PRF until about \(2^{\ell/2}\) queries are issued.

### 9.3 Chosen-Plaintext Security

We now extend the one-time CPA definition to many queries.

**Definition 9.2.** Let \(\Pi = (\text{Enc}, \text{Dec})\) be a randomized encryption scheme with key-space \(K\), message-space \(M\), randomness-space \(R\), and ciphertext-space \(C\). We assume that the message-space \(M\) is a set of bit strings, i.e. \(M \subseteq \{0,1\}^*\). Let \(A\) be an algorithm. Define algorithm \(\text{Expt}_\Pi^{\text{CPA}}(A)\) as

\[
\text{Alg Expt}_\Pi^{\text{CPA}}(A) \\
01 \text{ Pick } k \leftarrow K, b \leftarrow \{0,1\} \\
02 \text{ Run } A^{\text{LR}_k,b(\cdot, \cdot)}, \text{ where the oracle is given below. Eventually } A \text{ halts with output } \hat{b} \\
03 \text{ If } \hat{b} = b: \text{ Output } 1 \\
06 \text{ Else: Output } 0
\]

**Oracle LR\(_{k,b}(m_0,m_1)\)**

If \(m_0, m_1\) are not the same length: Return \(\perp\)

- Pick \(r \leftarrow R\)
- Compute \(c \leftarrow \text{Enc}(k, m_b, r)\)
- Return \(c\)

Define the CPA advantage of \(A\) against \(\Pi\) as

\[
\text{Adv}_\Pi^{\text{CPA}}(A) = \left| \Pr[\text{Expt}_\Pi^{\text{CPA}}(A) = 1] - \frac{1}{2} \right|
\]

The definition for the stateful version of the scheme is the same, except the oracle works as follows:

**Oracle LR\(_{k,b}(m_0,m_1)\)**

If \(m_0, m_1\) are not the same length: Return \(\perp\)

- Compute \((c, s') \leftarrow \text{Enc}(k, m_b, s)\)
- Overwrite the current state \(s\) with \(s'\)
- Return \(c\)