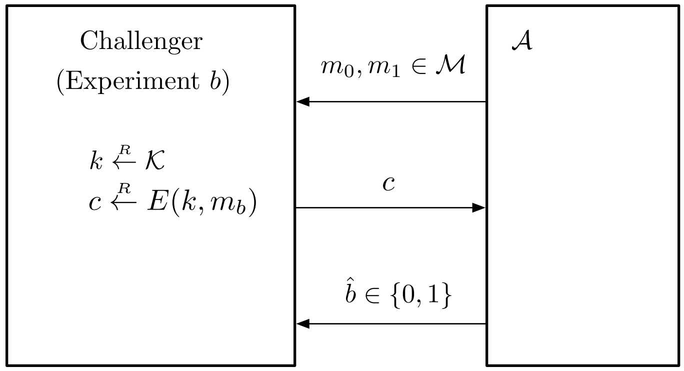

Figure 2.1: Experiment b of Attack Game 2.1

Attack Game 2.1 (semantic security). For a given cipher $\mathcal{E} = (E, D)$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, and for a given adversary $\mathcal{A}$, we define two experiments, Experiment 0 and Experiment 1. For $b = 0, 1$, we define

Experiment b:

- The adversary computes $m_0, m_1 \in \mathcal{M}$, of the same length, and sends them to the challenger.
- The challenger computes $k \xleftarrow{R} \mathcal{K}$, $c \xleftarrow{R} E(k, m_b)$, and sends c to the adversary.
- The adversary outputs a bit $\hat{b} \in \{0, 1\}$.

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define $\mathcal{A}$'s **semantic security advantage** with respect to $\mathcal{E}$ as

$$\mathsf{SSadv}[\mathcal{A}, \mathcal{E}] := \Big| \Pr[W_0] - \Pr[W_1] \Big|. \quad \square$$

Note that in the above game, the events W_0 and W_1 are defined with respect to the probability space determined by the random choice of k, the random choices made (if any) by the encryption algorithm, and the random choices made (if any) by the adversary. The value $\mathsf{SSadv}[\mathcal{A}, \mathcal{E}]$ is a number between 0 and 1.

See Fig. 2.1 for a schematic diagram of Attack Game 2.1. As indicated in the diagram, $\mathcal{A}$'s "output" is really just a final message to the challenger.

Definition 2.2 (semantic security). *A cipher $\mathcal{E}$ is **semantically secure** if for all efficient adversaries $\mathcal{A}$, the value $\mathsf{SSadv}[\mathcal{A}, \mathcal{E}]$ is negligible.*

As a formal definition, this is not quite complete, as we have yet to define what we mean by "messages of the same length", "efficient adversaries", and "negligible". We will come back to this shortly.

Let us relate this formal definition to the discussion preceding it. Suppose that the adversary $\mathcal{A}$ in Attack Game 2.1 is deterministic. First, the adversary computes in a deterministic fashion

Chapter 3

Stream ciphers

In the previous chapter, we introduced the notions of perfectly secure encryption and semantically secure encryption. The problem with perfect security is that to achieve it, one must use very long keys. Semantic security was introduced as a weaker notion of security that would perhaps allow us to build secure ciphers that use reasonably short keys; however, we have not yet produced any such ciphers. This chapter studies one type of cipher that does this: the stream cipher.

3.1 Pseudo-random generators

Recall the one-time pad. Here, keys, messages, and ciphertexts are all L-bit strings. However, we would like to use a key that is much shorter. So the idea is to instead use a short, ℓ-bit "seed" s as the encryption key, where ℓ is much smaller than L, and to "stretch" this seed into a longer, L-bit string that is used to mask the message (and unmask the ciphertext). The string s is stretched using some efficient, deterministic algorithm G that maps ℓ-bit strings to L-bit strings. Thus, the key space for this modified one-time pad is $\{0,1\}^\ell$, while the message and ciphertext spaces are $\{0,1\}^L$. For $s \in \{0,1\}^\ell$ and $m, c \in \{0,1\}^L$, encryption and decryption are defined as follows:

$$E(s,m) := G(s) \oplus m \quad \text{and} \quad D(s,c) := G(s) \oplus c.$$

This modified one-time pad is called a **stream cipher**, and the function G is called a **pseudo-random generator**.

If $\ell < L$, then by Shannon's Theorem, this stream cipher cannot achieve perfect security; however, if G satisfies an appropriate security property, then this cipher is semantically secure. Suppose s is a random ℓ-bit string and r is a random L-bit string. Intuitively, if an adversary cannot effectively tell the difference between $G(s)$ and r, then he should not be able to tell the difference between this stream cipher and a one-time pad; moreover, since the latter cipher is semantically secure, so should be the former. To make this reasoning rigorous, we need to formalize the notion that an adversary cannot "effectively tell the difference between $G(s)$ and r."

An algorithm that is used to distinguish a pseudo-random string $G(s)$ from a truly random string r is called a **statistical test**. It takes a string as input, and outputs 0 or 1. Such a test is called **effective** if the probability that it outputs 1 on a pseudo-random input is significantly different than the probability that it outputs 1 on a truly random input. Even a relatively small difference in probabilities, say 1%, is considered significant; indeed, even with a 1% difference, if we can obtain a few hundred independent samples, which are either all pseudo-random or all truly random, then we will be able to infer with high confidence whether we are looking at pseudo-random strings or at truly random strings. However, a non-zero but negligible difference in probabilities, say 2^{-100}, is not helpful.

How might one go about designing an effective statistical test? One basic approach is the following: given an L-bit string, calculate some statistic, and then see if this statistic differs greatly from what one would expect if the string were truly random.

For example, a very simple statistic that is easy to compute is the number k of 1's appearing in the string. For a truly random string, we would expect $k \approx L/2$. If the PRG G had some bias towards either 0-bits or 1-bits, we could effectively detect this with a statistical test that, say, outputs 1 if $|k - 0.5L| < 0.01L$, and otherwise outputs 0. This statistical test would be quite effective if the PRG G did indeed have some significant bias towards either 0 or 1.

The test in the previous example can be strengthened by considering not just individual bits, but pairs of bits. One could break the L-bit string up into $\approx L/2$ bit pairs, and count the number k_{00} of pairs 00, the number k_{01} of pairs 01, the number k_{10} of pairs 10, and the number k_{11} of pairs 11. For a truly random string, one would expect each of these numbers to be $\approx L/2 \cdot 1/4 = L/8$. Thus, a natural statistical test would be one that tests if the distance from $L/8$ of each of these numbers is less than some specified bound. Alternatively, one could sum up the squares of these distances, and test whether this sum is less than some specified bound — this is the classical χ-squared test from statistics. Obviously, this idea generalizes from pairs of bits to tuples of any length.

There are many other simple statistics one might check. However, simple tests such as these do not tend to exploit deeper mathematical properties of the algorithm G that a malicious adversary may be able to exploit in designing a statistical test specifically geared towards G. For example, there are PRG's for which the simple tests in the previous two paragraphs are completely ineffective, but yet are completely predictable, given sufficiently many output bits; that is, given a prefix of $G(s)$ of sufficient length, the adversary can compute all the remaining bits of $G(s)$, or perhaps even compute the seed s itself.

Our definition of security for a PRG formalizes the notion that there should be no effective (and efficiently computable) statistical test.

3.1.1 Definition of a pseudo-random generator

A **pseudo-random generator**, or **PRG** for short, is an efficient, deterministic algorithm G that, given as input a **seed** s, computes an output r. The seed s comes from a finite **seed space** $\mathcal{S}$ and the output r belongs to a finite **output space** $\mathcal{R}$. Typically, $\mathcal{S}$ and $\mathcal{R}$ are sets of bit strings of some prescribed length (for example, in the discussion above, we had $\mathcal{S} = \{0,1\}^\ell$ and $\mathcal{R} = \{0,1\}^L$). We say that G is a PRG defined over $(\mathcal{S}, \mathcal{R})$.

Our definition of security for a PRG captures the intuitive notion that if s is chosen at random from $\mathcal{S}$ and r is chosen at random from $\mathcal{R}$, then no efficient adversary can effectively tell the difference between $G(s)$ and r: the two are **computationally indistinguishable**. The definition is formulated as an attack game.

Attack Game 3.1 (PRG). For a given PRG G, defined over $(\mathcal{S}, \mathcal{R})$, and for a given adversary $\mathcal{A}$, we define two experiments, Experiment 0 and Experiment 1. For $b = 0, 1$, we define:

Experiment b:

- The challenger computes $r \in \mathcal{R}$ as follows:

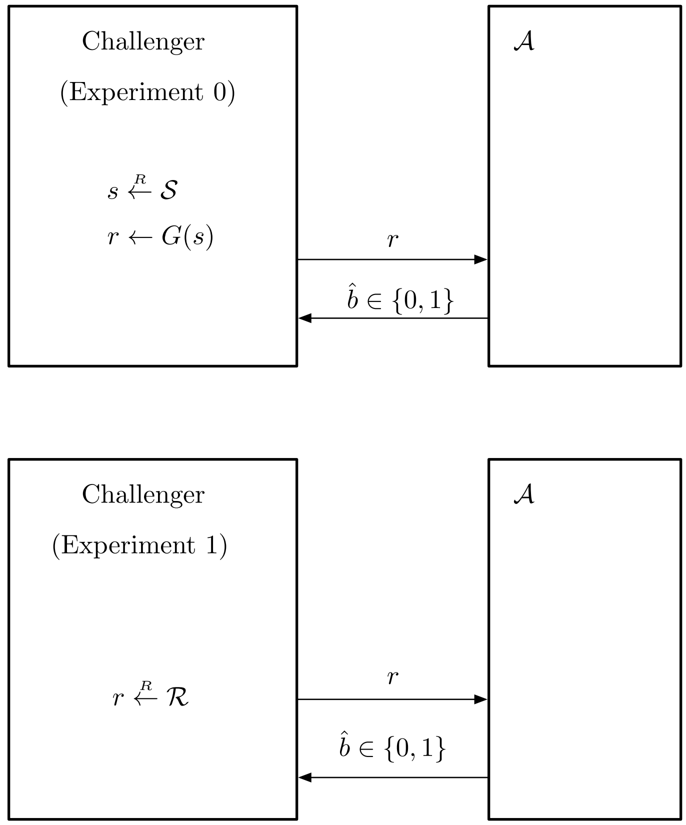

Figure 3.1: Experiments 0 and 1 of Attack Game 3.1

- if $b = 0$: $s \xleftarrow{R} \mathcal{S}$, $r \leftarrow G(s)$;
- if $b = 1$: $r \xleftarrow{R} \mathcal{R}$.

and sends r to the adversary.

- Given r, the adversary computes and outputs a bit $\hat{b} \in \{0,1\}$.

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define $\mathcal{A}$'s **advantage** with respect to G as

$$\text{PRG}\mathsf{adv}[\mathcal{A}, G] := \Big|\Pr[W_0] - \Pr[W_1]\Big|. \quad \square$$

The attack game is illustrated in Fig. 3.1.

Definition 3.1 (secure PRG). *A PRG G is* ***secure*** *if the value* PRG$\mathsf{adv}[\mathcal{A}, G]$ *is negligible for all efficient adversaries $\mathcal{A}$.*

As discussed in Section 2.2.5, Attack Game 3.1 can be recast as a "bit guessing" game, where instead of having two separate experiments, the challenger chooses $b \in \{0,1\}$ at random, and then runs Experiment b against the adversary $\mathcal{A}$. In this game, we measure $\mathcal{A}$'s *bit-guessing advantage*

$\text{PRGadv}^*[\mathcal{A}, G]$ as $|\Pr[\hat{b} = b] - 1/2|$. The general result of Section 2.2.5 (namely, (2.11)) applies here as well:

$$\text{PRGadv}[\mathcal{A}, G] = 2 \cdot \text{PRGadv}^*[\mathcal{A}, G]. \tag{3.1}$$

We also note that a PRG can only be secure if the cardinality of the seed space is super-poly (see Exercise 3.5).

3.1.2 Mathematical details

Just as in Section 2.3, we give here more of the mathematical details pertaining to PRGs. Just like Section 2.3, this section may be safely skipped on first reading with very little loss in understanding.

First, we state the precise definition of a PRG, using the terminology introduced in Definition 2.9.

Definition 3.2 (pseudo-random generator). *A **pseudo-random generator** consists of an algorithm G, along with two families of spaces with system parameterization P:*

$$\mathbf{S} = \{\mathcal{S}_{\lambda,\Lambda}\}_{\lambda,\Lambda} \quad \textit{and} \quad \mathbf{R} = \{\mathcal{R}_{\lambda,\Lambda}\}_{\lambda,\Lambda},$$

such that

1. *$\mathbf{S}$ and $\mathbf{R}$ are efficiently recognizable and sampleable.*
2. *Algorithm G is an efficient deterministic algorithm that on input λ, Λ, s, where $\lambda \in \mathbb{Z}_{\geq 1}$, $\Lambda \in \text{Supp}(P(\lambda))$, and $s \in \mathcal{S}_{\lambda,\Lambda}$, outputs an element of $\mathcal{R}_{\lambda,\Lambda}$.*

Next, Definition 3.1 needs to be properly interpreted. First, in Attack Game 3.1, it is to be understood that for each value of the security parameter λ, we get a different probability space, determined by the random choices of the challenger and the random choices of the adversary. Second, the challenger generates a system parameter Λ, and sends this to the adversary at the very start of the game. Third, the advantage $\text{PRGadv}[\mathcal{A}, G]$ is a function of the security parameter λ, and security means that this function is a negligible function.

3.2 Stream ciphers: encryption with a PRG

Let G be a PRG defined over $(\{0,1\}^\ell, \{0,1\}^L)$; that is, G stretches an ℓ-bit seed to an L-bit output. The **stream cipher $\mathcal{E} = (E, D)$ constructed from** G is defined over $(\{0,1\}^\ell, \{0,1\}^{\leq L}, \{0,1\}^{\leq L})$; for $s \in \{0,1\}^\ell$ and $m, c \in \{0,1\}^{\leq L}$, encryption and decryption are defined as follows: if $|m| = v$, then

$$E(s, m) := G(s)[0 \mathinner{..} v-1] \ \oplus \ m,$$

and if $|c| = v$, then

$$D(s, c) := G(s)[0 \mathinner{..} v-1] \ \oplus \ c.$$

As the reader may easily verify, this satisfies our definition of a cipher (in particular, the correctness property is satisfied).

Note that for the purposes of analyzing the semantic security of $\mathcal{E}$, the length associated with a message m in Attack Game 2.1 is the natural length $|m|$ of m in bits. Also, note that if v is much smaller than L, then for many practical PRGs, it is possible to compute the first v bits of $G(s)$ much faster than actually computing all the bits of $G(s)$ and then truncating.

The main result of this section is the following:

Theorem 3.1. *If G is a secure PRG, then the stream cipher $\mathcal{E}$ constructed from G is a semantically secure cipher.*

In particular, for every SS adversary $\mathcal{A}$ that attacks $\mathcal{E}$ as in Attack Game 2.1, there exists a PRG adversary $\mathcal{B}$ that attacks G as in Attack Game 3.1, where $\mathcal{B}$ is an elementary wrapper around $\mathcal{A}$, such that

$$\mathrm{SS}\mathsf{adv}[\mathcal{A}, \mathcal{E}] = 2 \cdot \mathrm{PRG}\mathsf{adv}[\mathcal{B}, G]. \tag{3.2}$$

Proof idea. The basic idea is to argue that we can replace the output of the PRG by a truly random string, without affecting the adversary's advantage by more than a negligible amount. However, after making this replacement, the adversary's advantage is zero. □

Proof. Let $\mathcal{A}$ be an efficient adversary attacking $\mathcal{E}$ as in Attack Game 2.1. We want to show that $\mathrm{SS}\mathsf{adv}[\mathcal{A}, \mathcal{E}]$ is negligible, assuming that G is a secure PRG. It is more convenient to work with the bit-guessing version of the SS attack game. We prove:

$$\mathrm{SS}\mathsf{adv}^*[\mathcal{A}, \mathcal{E}] = \mathrm{PRG}\mathsf{adv}[\mathcal{B}, G] \tag{3.3}$$

for some efficient adversary $\mathcal{B}$. Then (3.2) follows from Theorem 2.10. Moreover, by the assumption that G is a secure PRG, the quantity $\mathrm{PRG}\mathsf{adv}[\mathcal{B}, G]$ must be negligible, and so the quantity $\mathrm{SS}\mathsf{adv}[\mathcal{A}, \mathcal{E}]$ is negligible as well.

So consider the adversary $\mathcal{A}$'s attack of $\mathcal{E}$ in the bit-guessing version of Attack Game 2.1. In this game, $\mathcal{A}$ presents the challenger with two messages m_0, m_1 of the same length; the challenger then chooses a random key s and a random bit b, and encrypts m_b under s, giving the resulting ciphertext c to $\mathcal{A}$; finally, $\mathcal{A}$ outputs a bit $\hat{b}$. The adversary $\mathcal{A}$ wins the game if $\hat{b} = b$.

Let us call this **Game 0.** The logic of the challenger in this game may be written as follows:

Upon receiving $m_0, m_1 \in \{0,1\}^v$ from $\mathcal{A}$, for some $v \leq L$, do:
$\quad b \overset{\scriptscriptstyle R}{\leftarrow} \{0,1\}$
$\quad s \overset{\scriptscriptstyle R}{\leftarrow} \{0,1\}^\ell$, $r \leftarrow G(s)$
$\quad c \leftarrow r[0 \,..\, v-1] \oplus m_b$
$\quad$send c to $\mathcal{A}$.

Game 0 is illustrated in Fig. 3.2.

Let W_0 be the event that $\hat{b} = b$ in Game 0. By definition, we have

$$\mathrm{SS}\mathsf{adv}^*[\mathcal{A}, \mathcal{E}] = |\Pr[W_0] - 1/2|. \tag{3.4}$$

Next, we modify the challenger of Game 0, obtaining a new game, called **Game 1**, which is exactly the same as Game 0, except that the challenger uses a truly random string in place of a pseudo-random string. The logic of the challenger in Game 1 is as follows:

Upon receiving $m_0, m_1 \in \{0,1\}^v$ from $\mathcal{A}$, for some $v \leq L$, do:
$\quad b \overset{\scriptscriptstyle R}{\leftarrow} \{0,1\}$
$\quad r \overset{\scriptscriptstyle R}{\leftarrow} \{0,1\}^L$
$\quad c \leftarrow r[0 \,..\, v-1] \oplus m_b$
$\quad$send c to $\mathcal{A}$.

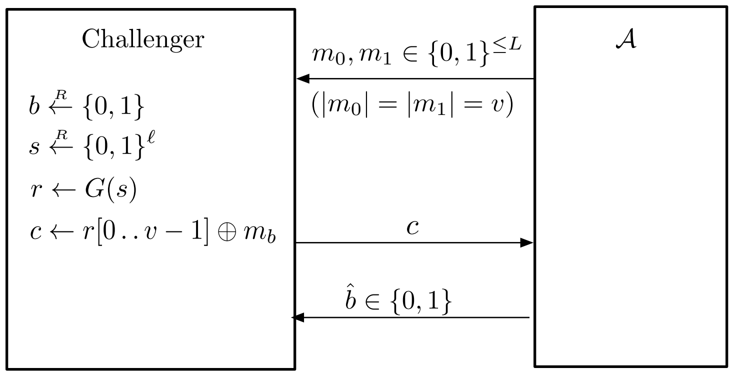

Figure 3.2: Game 0 in the proof of Theorem 3.1

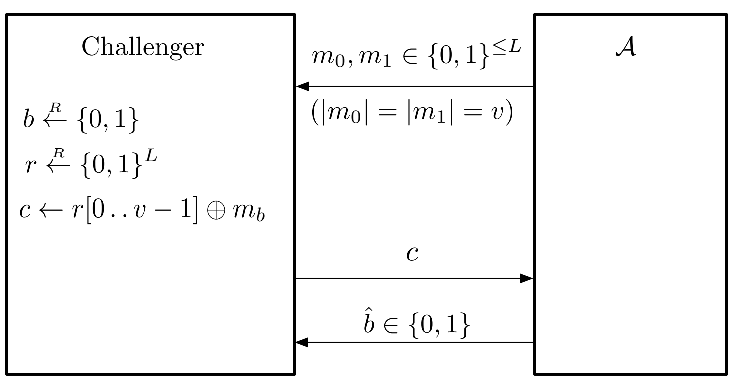

Figure 3.3: Game 1 in the proof of Theorem 3.1

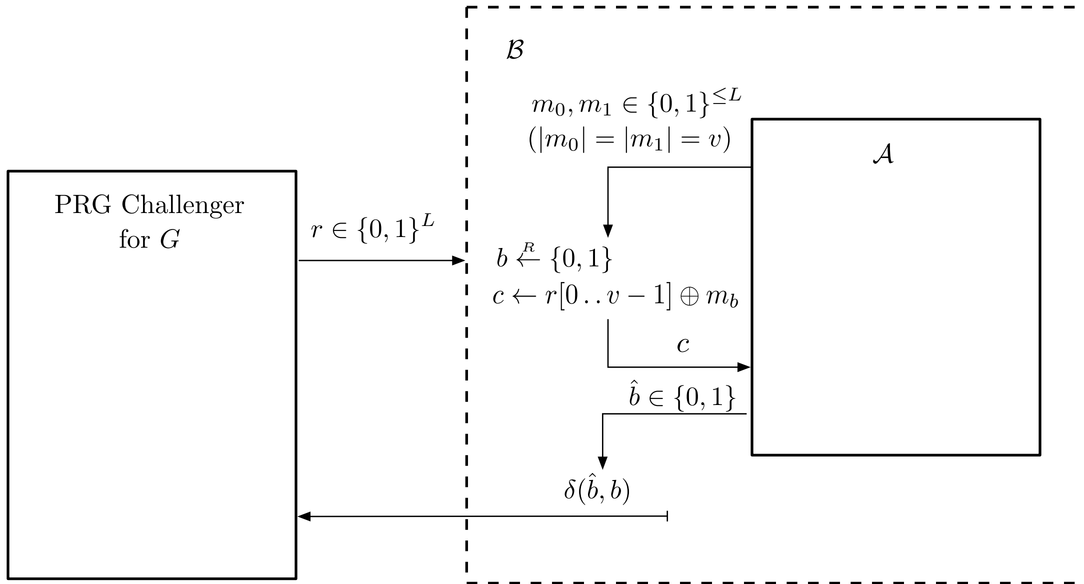

Figure 3.4: The PRG adversary $\mathcal{B}$ in the proof of Theorem 3.1

As usual, $\mathcal{A}$ outputs a bit $\hat{b}$ at the end of this game. We have highlighted the changes from Game 0 in gray. Game 1 is illustrated in Fig. 3.3.

Let W_1 be the event that $\hat{b} = b$ in Game 1. We claim that

$$\Pr[W_1] = 1/2. \tag{3.5}$$

This is because in Game 1, the adversary is attacking the variable length one-time pad. In particular, it is easy to see that the adversary's output $\hat{b}$ and the challenger's hidden bit b are independent.

Finally, we show how to construct an efficient PRG adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine, such that

$$|\Pr[W_0] - \Pr[W_1]| = \text{PRG}\mathsf{adv}[\mathcal{B}, G]. \tag{3.6}$$

This is actually quite straightforward. The logic of our new adversary $\mathcal{B}$ is illustrated in Fig. 3.4. Here, δ is defined as follows:

$$\delta(x, y) := \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y. \end{cases} \tag{3.7}$$

Also, the box labeled "PRG Challenger" is playing the role of the challenger in Attack Game 3.1 with respect to G.

In words, adversary $\mathcal{B}$, which is a PRG adversary designed to attack G (as in Attack Game 3.1), receives $r \in \{0,1\}^L$ from its PRG challenger, and then plays the role of challenger to $\mathcal{A}$, as follows:

Upon receiving $m_0, m_1 \in \{0,1\}^v$ from $\mathcal{A}$, for some $v \leq L$, do:
  $b \xleftarrow{R} \{0,1\}$
  $c \leftarrow r[0\,..\,v-1] \oplus m_b$
  send c to $\mathcal{A}$.

Finally, when $\mathcal{A}$ outputs a bit $\hat{b}$, $\mathcal{B}$ outputs the bit $\delta(\hat{b}, b)$.

Let p_0 be the probability that $\mathcal{B}$ outputs 1 when the PRG challenger is running Experiment 0 of Attack Game 3.1, and let p_1 be the probability that $\mathcal{B}$ outputs 1 when the PRG challenger is running Experiment 1 of Attack Game 3.1. By definition, $\text{PRG}\mathsf{adv}[\mathcal{B}, G] = |p_1 - p_0|$. Moreover, if the PRG challenger is running Experiment 0, then adversary $\mathcal{A}$ is essentially playing our Game 0, and so $p_0 = \Pr[W_0]$, and if the PRG challenger is running Experiment 1, then $\mathcal{A}$ is essentially playing our Game 1, and so $p_1 = \Pr[W_1]$. Equation (3.6) now follows immediately.

Combining (3.4), (3.5), and (3.6), yields (3.3). $\Box$

In the above theorem, we reduced the security of $\mathcal{E}$ to that of G by showing that if $\mathcal{A}$ is an efficient SS adversary that attacks $\mathcal{E}$, then there exists an efficient PRG adversary $\mathcal{B}$ that attacks G, such that

$$\text{SS}\mathsf{adv}[\mathcal{A}, \mathcal{E}] \leq 2 \cdot \text{PRG}\mathsf{adv}[\mathcal{B}, G].$$

(Actually, we showed that equality holds, but that is not so important.) In the proof, we argued that if G is secure, then $\text{PRG}\mathsf{adv}[\mathcal{B}, G]$ is negligible, hence by the above inequality, we conclude that $\text{SS}\mathsf{adv}[\mathcal{A}, \mathcal{E}]$ is also negligible. Since this holds for all efficient adversaries $\mathcal{A}$, we conclude that $\mathcal{E}$ is semantically secure.

Analogous to the discussion after the proof of Theorem 2.7, another way to structure the proof is by proving the contrapositive: indeed, if we assume that $\mathcal{E}$ is insecure, then there must be an efficient adversary $\mathcal{A}$ such that $\text{SS}\mathsf{adv}[\mathcal{A}, \mathcal{E}]$ is non-negligible, and the reduction (and the above inequality) gives us an efficient adversary $\mathcal{B}$ such that $\text{PRG}\mathsf{adv}[\mathcal{B}, G]$ is also non-negligible. That is, if we can break $\mathcal{E}$, we can also break G. While logically equivalent, such a proof has a different "feeling": one starts with an adversary $\mathcal{A}$ that breaks $\mathcal{E}$, and shows how to use $\mathcal{A}$ to construct a new adversary $\mathcal{B}$ that breaks G.

The reader should notice that the proof of the above theorem follows the same basic pattern as our analysis of Internet roulette in Section 2.2.4. In both cases, we started with an attack game (Fig. 2.2 or Fig. 3.2) which we modified to obtain a new attack game (Fig. 2.3 or Fig. 3.3); in this new attack game, it was quite easy to compute the adversary's advantage. Also, we used an appropriate security assumption to show that the difference between the adversary's advantages in the original and the modified games was negligible. This was done by exhibiting a new adversary (Fig. 2.4 or Fig. 3.4) that attacked the underlying cryptographic primitive (cipher or PRG) with an advantage equal to this difference. Assuming the underlying primitive was secure, this difference must be negligible; alternatively, one could argue the contrapositive: if this difference were not negligible, the new adversary would "break" the underlying cryptographic primitive.

This is a pattern that will be repeated and elaborated upon throughout this text. The reader is urged to study both of these analyses to make sure he or she completely understands what is going on.

3.3 Stream cipher limitations: attacks on the one time pad

Although stream ciphers are semantically secure, they are quite brittle and become insecure if used incorrectly.