As indicated by the previous example, we can view the security parameter as a mechanism that allows the honest parties to "tune" the security of a scheme to some desired level. (Increasing the security parameter also increases the time required to run the scheme, as well as the length of the key, so the honest parties will want to set the security parameter as small as possible subject to defending against the class of attacks they are concerned about.) Viewing the security parameter as the key length, this corresponds roughly to the fact that the time required for an exhaustive-search attack grows exponentially in the length of the key. The ability to "increase security" by increasing the security parameter has important practical ramifications, since it enables honest parties to defend against increases in computing power. The following example gives a sense of how this might play out in practice.

Example 3.3

Let us see the effect that the availability of faster computers might have on security in practice. Say we have a cryptographic scheme in which the honest parties run for $10^6 \cdot n^2$ cycles, and for which an adversary running for $10^8 \cdot n^4$ cycles can succeed in "breaking" the scheme with probability at most $2^{-n/2}$. (The numbers are intended to make calculations easier, and are not meant to correspond to any existing cryptographic scheme.)

Say all parties are using 2 GHz computers and the honest parties set $n = 80$. Then the honest parties run for $10^6 \cdot 6400$ cycles, or 3.2 seconds, and an adversary running for $10^8 \cdot (80)^4$ cycles, or roughly 3 weeks, can break the scheme with probability only 2^{-40}.

Say 8 GHz computers become available, and all parties upgrade. Honest parties can increase n to 160 (which requires generating a fresh key) and maintain a running time of 3.2 seconds (i.e., $10^6 \cdot 160^2$ cycles at $8 \cdot 10^9$ cycles/second). In contrast, the adversary now has to run for over 8 million seconds, or more than 13 weeks, to achieve a success probability of 2^{-80}. The effect of a faster computer has been to make the adversary's job *harder*. ◇

Even when using the asymptotic approach it is important to remember that, ultimately, when a cryptosystem is deployed in practice a concrete security guarantee will be needed. (After all, one must decide on some value of n.) As the above examples indicate, however, it is generally the case that an asymptotic security claim can be translated into a concrete security bound for any desired value of n.

The Asymptotic Approach in Detail

We now discuss more formally the notions of "polynomial-time algorithms" and "negligible success probabilities."

Efficient algorithms. We have defined an algorithm to be efficient if it runs in polynomial time. An algorithm A runs in polynomial time if there exists a

polynomial p such that, for every input $x \in \{0,1\}^*$, the computation of $A(x)$ terminates within at most $p(|x|)$ steps. (Here, $|x|$ denotes the length of the string x.) As mentioned earlier, we are only interested in adversaries whose running time is polynomial in the security parameter n. Since we measure the running time of an algorithm in terms of the length of its input, we sometimes provide algorithms with the security parameter written in unary (i.e., as 1^n, or a string of n ones) as input. Parties (or, more precisely, the algorithms they run) may take other inputs besides the security parameter—for example, a message to be encrypted—and we allow their running time to be polynomial in the (total) length of their inputs.

By default, we allow all algorithms to be probabilistic (or randomized). Any such algorithm may "toss a coin" at each step of its execution; this is a metaphorical way of saying that the algorithm can access an unbiased random bit at each step. Equivalently, we can view a randomized algorithm as one that, in addition to its input, is given a uniformly distributed *random tape* of sufficient length[1] whose bits it can use, as needed, throughout its execution.

We consider randomized algorithms by default for two reasons. First, randomness is essential to cryptography (e.g., in order to choose random keys and so on) and so honest parties must be probabilistic; given this, it is natural to allow adversaries to be probabilistic as well. Second, randomization is practical and—as far as we know—gives attackers additional power. Since our goal is to model *all* realistic attacks, we prefer a more liberal definition of efficient computation.

Negligible success probability. A negligible function is one that is asymptotically smaller than any inverse polynomial function. Formally:

DEFINITION 3.4 *A function f from the natural numbers to the non-negative real numbers is* negligible *if for every positive polynomial p there is an N such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.*

For shorthand, the above is also stated as follows: for every polynomial p and *all sufficiently large values of* n it holds that $f(n) < \frac{1}{p(n)}$. An equivalent formulation of the above is to require that for all constants c there exists an N such that for all $n > N$ it holds that $f(n) < n^{-c}$. We typically denote an arbitrary negligible function by negl.

Example 3.5
The functions $2^{-n}, 2^{-\sqrt{n}}$, and $n^{-\log n}$ are all negligible. However, they approach zero at very different rates. For example, we can look at the minimum value of n for which each function is smaller than $1/n^5$:

[1] If the algorithm in question runs for $p(n)$ steps on inputs of length n, then a random tape of length $p(n)$ is sufficient since the attacker can read at most one random bit per time step.

1. Solving $2^{-n} < n^{-5}$ we get $n > 5 \log n$. The smallest integer value of n for which this holds is $n = 23$.
2. Solving $2^{-\sqrt{n}} < n^{-5}$ we get $n > 25 \log^2 n$. The smallest integer value of n for which this holds is $n \approx 3500$.
3. Solving $n^{-\log n} < n^{-5}$ we get $\log n > 5$. The smallest integer value of n for which this holds is $n = 33$.

From the above you may have the impression that $n^{-\log n}$ approaches zero more quickly than $2^{-\sqrt{n}}$. However, this is incorrect; for all $n > 65536$ it holds that $2^{-\sqrt{n}} < n^{-\log n}$. Nevertheless, this does show that for values of n in the hundreds or thousands, an adversarial success probability of $n^{-\log n}$ is preferable to an adversarial success probability of $2^{-\sqrt{n}}$. ♢

A technical advantage of working with negligible success probabilities is that they obey certain closure properties. The following is an easy exercise.

PROPOSITION 3.6 *Let negl_1 and negl_2 be negligible functions. Then,*

1. *The function negl_3 defined by $\mathsf{negl}_3(n) = \mathsf{negl}_1(n) + \mathsf{negl}_2(n)$ is negligible.*
2. *For any positive polynomial p, the function negl_4 defined by $\mathsf{negl}_4(n) = p(n) \cdot \mathsf{negl}_1(n)$ is negligible.*

The second part of the above proposition implies that if a certain event occurs with only negligible probability in a certain experiment, then the event occurs with negligible probability even if the experiment is repeated polynomially many times. (This relies on the union bound; see Proposition A.7.) For example, the probability that n fair coin flips all come up "heads" is negligible. This means that even if we repeat the experiment of flipping n coins polynomially many times, the probability that *any* of those experiments result in n heads is still negligible.

A corollary of the second part of the above proposition is that if a function g is *not* negligible, then neither is the function $f(n) \stackrel{\text{def}}{=} g(n)/p(n)$ for any positive polynomial p.

Asymptotic Security: A Summary

Any security definition consists of two parts: a definition of what is considered a "break" of the scheme, and a specification of the power of the adversary. The power of the adversary can relate to many issues (e.g., in the case of encryption, whether we assume a ciphertext-only attack or a chosen-plaintext attack). However, when it comes to the *computational* power of the adversary, we will from now on model the adversary as efficient and thus only consider adversarial strategies