A Graduate Course in Applied Cryptography

Dan Boneh and Victor Shoup

Version 0.4, September 2017

Chapter 2.4—Edited for CIS 331 (cut short and added a note)

2.4 Mathematical details

Up until now, we have used the terms *efficient* and *negligible* rather loosely, without a formal mathematical definition:

- we required that a computational cipher have *efficient* encryption and decryption algorithms;
- for a semantically secure cipher, we required that any *efficient* adversary have a *negligible* advantage in Attack Game 2.1.

The goal of this section is to provide precise mathematical definitions for these terms. While these definitions lead to a satisfying theoretical framework for the study of cryptography as a mathematical discipline, we should warn the reader:

- the definitions are rather complicated, requiring an unfortunate amount of notation; and
- the definitions model our intuitive understanding of these terms only very crudely.

We stress that the reader may safely skip this section without suffering a significant loss in understanding. Before marching headlong into the formal definitions, let us remind the reader of what we are trying to capture in these definitions.

- First, when we speak of an efficient encryption or decryption algorithm, we usually mean one that runs very quickly, encrypting data at a rate of, say, 10–100 computer cycles per byte of data.

- Second, when we speak of an efficient adversary, we usually mean an algorithm that runs in some large, but still feasible amount of time (and other resources). Typically, one assumes that an adversary that is trying to break a cryptosystem is willing to expend many more resources than a user of the cryptosystem. Thus, 10,000 computers running in parallel for 10 years may be viewed as an upper limit on what is feasibly computable by a determined, patient, and financially well-off adversary. However, in some settings, like the Internet roulette example in Section 2.3.4, the adversary may have a much more limited amount of time to perform its computations before they become irrelevant.

- Third, when we speak of an adversary's advantage as being negligible, we mean that it is so small that it may as well be regarded as being equal to zero for all practical purposes. As we saw in the Internet roulette example, if no efficient adversary has an advantage better than 2^{-100} in Attack Game 2.1, then no player can in practice improve his odds at winning Internet roulette by more than 2^{-100} relative to physical roulette.

Even though our intuitive understanding of the term *efficient* depends on the context, our formal definition will not make any such distinction. Indeed, we shall adopt the computational complexity theorist's habit of equating the notion of an *efficient* algorithm with that of a *(probabilistic) polynomial-time* algorithm. For better and for worse, this gives us a formal framework that is independent of the specific details of any particular model of computation.

2.4.1 Negligible, super-poly, and poly-bounded functions

We begin by defining the notions of *negligible*, *super-poly*, and *poly-bounded* functions.

Intuitively, a negligible function $f : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ is one that not only tends to zero as $n \to \infty$, but does so faster than the inverse of any polynomial.

Definition 2.5. *A function $f : \mathbb{Z}_{\geq 1} \to \mathbb{R}$ is called **negligible** if for all $c \in \mathbb{R}_{>0}$ there exists $n_0 \in \mathbb{Z}_{\geq 1}$ such that for all integers $n \geq n_0$, we have $|f(n)| < 1/n^c$.*

An alternative characterization of a negligible function, which is perhaps easier to work with, is the following:

Theorem 2.11. *A function $f : \mathbb{Z}_{\geq 1} \to \mathbb{R}$ is negligible if and only if for all $c > 0$, we have*

$$\lim_{n \to \infty} f(n) n^c = 0.$$

Proof. Exercise. □

Example 2.10. Some examples of negligible functions:

$$2^{-n}, \quad 2^{-\sqrt{n}}, \quad n^{-\log n}.$$

Some examples of non-negligible functions:

$$\frac{1}{1000n^4 + n^2 \log n}, \quad \frac{1}{n^{100}}. \quad \square$$

Once we have the term "negligible" formally defined, defining "super-poly" is easy:

Definition 2.6. *A function $f : \mathbb{Z}_{\geq 1} \to \mathbb{R}$ is called **super-poly** if $1/f$ is negligible.*

Essentially, a poly-bounded function $f : \mathbb{Z}_{\geq 1} \to \mathbb{R}$ is one that is bounded (in absolute value) by some polynomial. Formally:

Definition 2.7. *A function $f : \mathbb{Z}_{\geq 1} \to \mathbb{R}$ is called **poly-bounded**, if there exists $c, d \in \mathbb{R}_{>0}$ such that for all integers $n \geq 0$, we have $|f(n)| \leq n^c + d$.*

Note that if f is a poly-bounded function, then $1/f$ is definitely *not* a negligible function. However, as the following example illustrates, one must take care not to draw erroneous inferences.

Example 2.11. Define $f : \mathbb{Z}_{\geq 1} \to \mathbb{R}$ so that $f(n) = 1/n$ for all even integers n and $f(n) = 2^{-n}$ for all odd integers n. Then f is not negligible, and $1/f$ is neither poly-bounded nor super-poly. □

2.4.2 Computational ciphers: the formalities

Now the formalities. We begin by admitting a lie: when we said a computational cipher $\mathcal{E} = (E, D)$ is defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, where $\mathcal{K}$ is the key space, $\mathcal{M}$ is the message space, and $\mathcal{C}$ is the ciphertext space, and with each of these spaces being finite sets, we were not telling the whole truth. In the mathematical model (though not always in real-world systems), we associate with $\mathcal{E}$ *families* of key, message, and ciphertext spaces, indexed by

- a **security parameter**, which is a positive integer, and is denoted by λ, and
- a **system parameter**, which is a bit string, and is denoted by Λ.

Thus, instead of just finite sets $\mathcal{K}$, $\mathcal{M}$, and $\mathcal{C}$, we have families of finite sets

$$\{\mathcal{K}_{\lambda,\Lambda}\}_{\lambda,\Lambda}, \quad \{\mathcal{M}_{\lambda,\Lambda}\}_{\lambda,\Lambda}, \quad \text{and} \quad \{\mathcal{C}_{\lambda,\Lambda}\}_{\lambda,\Lambda},$$

which for the purposes of this definition, we view as sets of bit strings (which may represent mathematical objects by way of some canonical encoding functions).

The idea is that when the cipher $\mathcal{E}$ is deployed, the security parameter λ is fixed to some value. Generally speaking, larger values of λ imply higher levels of security (i.e., resistance against adversaries with more computational resources), but also larger key sizes, as well as slower encryption and decryption speeds. Thus, the security parameter is like a "dial" we can turn, setting a trade-off between security and efficiency.

Once λ is chosen, a system parameter Λ is generated using an algorithm specific to the cipher. The idea is that the system parameter Λ (together with λ) gives a detailed description of a fixed instance of the cipher, with

$$(\mathcal{K}, \mathcal{M}, \mathcal{C}) = (\mathcal{K}_{\lambda,\Lambda},\ \mathcal{M}_{\lambda,\Lambda},\ \mathcal{C}_{\lambda,\Lambda}).$$

This one, fixed instance may be deployed in a larger system and used by many parties — the values of λ and Λ are public and known to everyone (including the adversary).

Example 2.12. Consider the additive one-time pad discussed in Example 2.4. This cipher was described in terms of a modulus n. To deploy such a cipher, a suitable modulus n is generated, and is made public (possibly just "hardwired" into the software that implements the cipher). The modulus n is the system parameter for this cipher. Each specific value of the security parameter determines the length, in bits, of n. The value n itself is generated by some algorithm that may be probabilistic and whose output distribution may depend on the intended application. For example, we may want to insist that n is a prime in some applications. □

CIS 331 note: In the book by N. Smart this is called the "Modified Shift Cipher" (Chapter 2.1)