

# **Part I**

## **Fundamental Issues**

# Preliminaries

## 2.1 Data Hiding Framework

A typical data hiding framework is illustrated in Fig. 2.1. Starting with an original copy of digital media, or *original media* ( $I_0$ ) in short <sup>1</sup>, an embedding module puts in it a set of secondary data to be embedded ( $b$ ), which is referred as *embedded data*. The output of the embedding module is perceptually identical to the original but with data hidden in. It is often referred as *marked media* ( $I_1$ ). The difference between  $I_1$  and  $I_0$  is referred as *embedding distortion*, i.e., the distortion introduced by the embedding process.

In most cases, the hidden data is a collection of bits, which, depending on the application, may come from an encoded character string, from a pattern, from some executable agents, or others. The bit-by-bit accuracy in extracting these hidden data from the marked media is desirable in these cases and is the focus of this thesis. The embedded data may form a perceptual source itself, such as the application of “image in image” and “video in video” [69]. Some moderate decay of the hidden data is tolerable in this case.

---

<sup>1</sup> $I_0$  is also commonly referred as *host media* or *cover media*.

The marked media may be subjected to various kinds of processing and attacks before feeding into a detector. The input media to the detector is referred as *test media* ( $I_2$ ), and the difference between  $I_2$  and  $I_1$  is referred as *noise*. The data extracted from  $I_2$  is referred as *extracted data* ( $\hat{b}$ ). In such applications as ownership protection, fingerprinting<sup>2</sup> and access control, accurate decoding of hidden data from distorted test media is preferred. They are commonly referred as *robust data hiding / watermarking*. In other applications such as authentication and annotation, robustness against processing and attacks are not a principal requirement in general. We will discuss the specific design requirement in the later chapters.

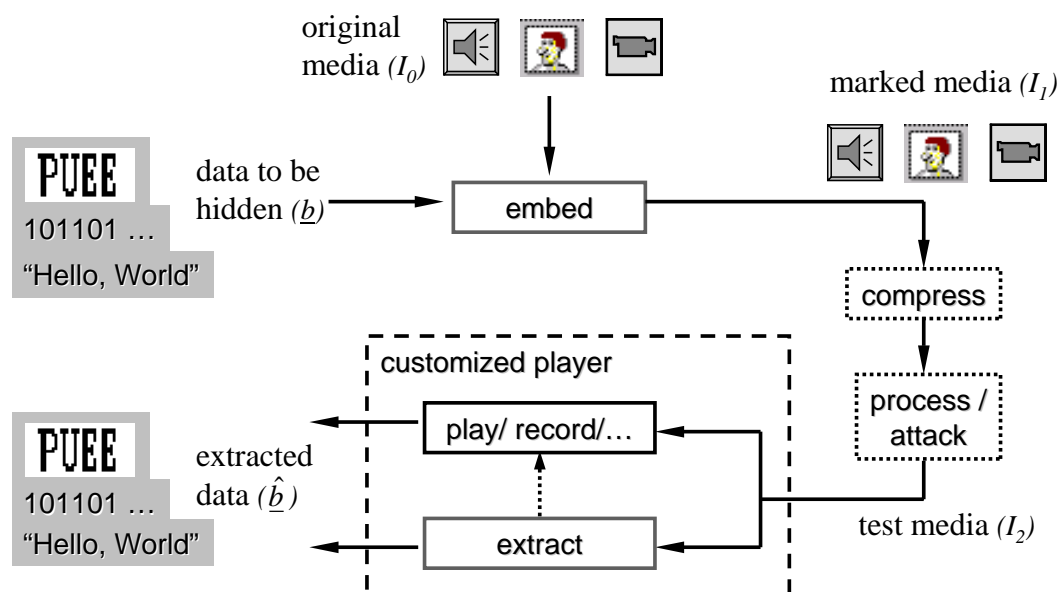


Figure 2.1: General framework of data hiding systems

<sup>2</sup>The *fingerprinting* here refers to the application where different labels are embedded in copies of the same media content before distributing to multiple recipients and the hidden labels are used for tracing each recipient. See also the discussion in Chapter 1.

## 2.2 Key Elements and A Layered View

Data hiding can be viewed as a communication problem, in which the hidden data is to be delivered and the media host serves as a carrier or as part of the channel. Techniques such as matched filtering, spread spectrum communication, modulation, and error correction coding are widely used in data hiding. In addition, a layered structure helps to prioritize and compartmentalize various design issues.

The key elements in many data hiding systems include:

1. A perceptual model that ensures imperceptibility;
2. How to embed one bit;
3. How to embed multiple bits via modulation/multiplexing techniques;
4. How to embed in those parts of cover media which are difficult to embed, or more generally, how to handle uneven embedding capacity;
5. How to enhance robustness and security;
6. What data to embed.

The elements can be viewed in layers, analogous to the layered structure in network communication (Fig. 2.2). The “physical layer” of data hiding deals with how one or multiple bits are embedded imperceptibly in the original media. This layer has three key elements, namely, (1) the mechanism for embedding one bit, (2) the perceptual model to ensure imperceptibility, and (3) the modulation/multiplexing techniques for hiding multiples bits. Protocols for achieving additional functionalities are built on top of this “physical layer”, for example, to handle uneven embedding capacity, to enhance robustness and approach capacity via error correction coding, and to

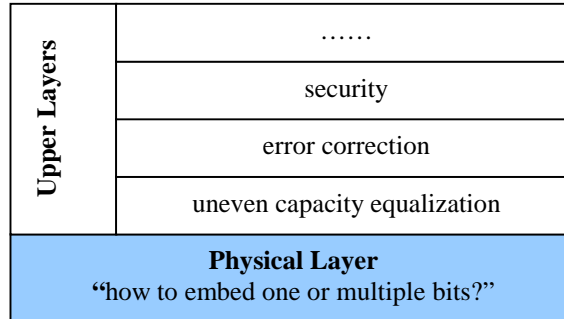


Figure 2.2: Layered structure of a data hiding system.

incorporate additional security measures. In the remaining chapters of Part I, we shall use data hiding in images as an example to discuss several elements in detail.