5.10.2 Optimal Asymmetric Encryption Padding

In this section, we describe a provably secure public-key cryptosystem in which a ciphertext reveals no information about the plaintext. The system is based on an arbitrary *trapdoor one-way permutation*, which is a (bijective) trapdoor one-way function from a set X to itself. If $f : X \rightarrow X$ is a trapdoor one-way permutation, then the inverse permutation is denoted, as usual, by f^{-1}. f is the encryption function, and f^{-1} is the decryption function of a public-key cryptosystem. In the case of the *RSA Cryptosystem*, we would take $n = pq$, $X = \mathbb{Z}_n$, $f(x) = x^b \bmod n$ and $f^{-1}(x) = x^a \bmod n$, where $ab \equiv 1 \pmod{\phi(n)}$. Cryptosystem 5.3 employs a trapdoor one-way permuation f, as well as a certain random function, G.

Cryptosystem 5.3: *Semantically Secure Public-key Cryptosystem*

Let m, k be positive integers; let $\mathcal{F}$ be a family of trapdoor one-way permutations such that $f : (\mathbb{Z}_2)^k \rightarrow (\mathbb{Z}_2)^k$ for all $f \in \mathcal{F}$; and let $G : (\mathbb{Z}_2)^k \rightarrow (\mathbb{Z}_2)^m$ be a "random" function. Let $\mathcal{P} = (\mathbb{Z}_2)^m$ and $\mathcal{C} = (\mathbb{Z}_2)^k \times (\mathbb{Z}_2)^m$, and define

$$\mathcal{K} = \{(f, f^{-1}, G) : f \in \mathcal{F}\}.$$

For $K = (f, f^{-1}, G)$, let $r \in (\mathbb{Z}_2)^k$ be chosen randomly, and define

$$e_K(x) = (y_1, y_2) = (f(r), G(r) + x),$$

where $x, y_1 \in (\mathbb{Z}_2)^k$, $y_2 \in (\mathbb{Z}_2)^m$. Further, define

$$d_K(y_1, y_2) = G(f^{-1}(y_1)) + y_2$$

($y_1 \in (\mathbb{Z}_2)^k$, $y_2 \in (\mathbb{Z}_2)^m$). The functions f and G are public; the function f^{-1} is secret.

First, we observe that Cryptosystem 5.3 is quite efficient: it requires little additional computation when compared to the underlying public-key cryptosystem based on f. In practice, the function G could be built from a secure hash function in a very efficient manner. The main drawback of Cryptosystem 5.3 is the data expansion: m bits of plaintext are encrypted to form $k + m$ bits of ciphertext. If f is based on the RSA encryption function, for example, then it will be necessary to take $k \geq 1024$ in order for the system to be secure.

It is easy to see that there must be some data expansion in any semantically secure cryptosystem due to the fact that encryption is randomized. However, there are more efficient schemes that are still provably secure. The most of important of these, *Optimal Asymmetric Encryption Padding*, will be discussed later in this section (it is presented as Cryptosystem 5.4). We begin with Cryptosystem 5.3, however, because it is conceptually simpler and easier to analyze.

An intuitive argument that Cryptosystem 5.3 is semantically secure goes as follows: in order to determine any information about the plaintext x, we need to have some information about $G(r)$. Assuming that G is random, the only way to ascertain any information about the value of $G(r)$ is to first compute $r = f^{-1}(y_1)$. (It is not sufficient to compute some partial information about r due to the fact that G is a random function. It is necesary to have complete information about r in order to obtain any information about $G(r)$.) However, if f is one-way, then r cannot be computed in a reasonable amount of time by an adversary who does not know the trapdoor, f^{-1}.

The preceding argument might be fairly convincing, but it is not a proof. If we are going to massage this argument into a proof, we need to describe a reduction, from the problem of computing information about plaintexts in this new cryptosystem, to the problem of inverting the function f. One suitable way to formalize the idea of computing information about plaintexts is to consider the the problem of Plaintext Distinguishability, which is defined as follows:

Problem 5.3:	Plaintext Distinguishability
Instance:	A trapdoor permutation $f : X \rightarrow X$; two plaintexts $x_1, x_2 \in X$; and a ciphertext $y = f(x_i)$, where $i \in \{1, 2\}$.
Question:	Is $i = 1$?

Problem 5.3 is of course trivial if the encryption function f is deterministic, since it suffices to compute $f(x_1)$ and $f(x_2)$ and see which one yields the ciphertext y. However, when f is randomized, as in Cryptosystem 5.3, it may not be feasible to solve Problem 5.3 if there are suffiently many possible encryptions of a given plaintext.

We are going to describe a reduction that is more general than the Turing reduction considered previously. We will assume the existence of an algorithm DISTINGUISH that solves the problem of Plaintext Distinguishability for two fixed plaintexts x_1 and x_2, and then modify this algorithm in such a way that we obtain an algorithm to invert f. The algorithm DISTINGUISH need not be a "perfect" algorithm; we will only require that it gives the right answer with some probability $1/2 + \epsilon$, where $\epsilon > 0$ (i.e., it is more accurate than a random guess of "1" or "2").

The trickiest part of this endeavor is to define precisely what is meant by G being a "random function". Because G must be completely specified as public part of the cryptosystem, it is clearly not a random function. However, the reduction will assume that G is a random function. It must be understood that the reduction (i.e., the proof of security) is taking place in an idealized model in which we hypothesize that G behaves like a random function. In particular, we will assume that no information about any value $G(r)$ can be determined without actually computing the function $G(r)$, even if many other values $G(r_1), G(r_2), \ldots$ have already been computed. In practice, however, because G must be completely specified, we cannot rule out the possibility of an attack on the cryptosystem that somehow takes into account the way that the function G is defined and infers some information about a value $G(r)$ without actually computing it.

We are going to prove that Cryptosystem 5.3 is semantically secure in this idealized model, which is known as the *random oracle model*. The main features of this model, and the reduction we are going to describe, are as follows.

1. We assume the existence of two plaintexts x_1 and x_2, and an algorithm DISTINGUISH that solves the Plaintext Distinguishability problem for these two given plaintexts with probability $1/2 + \epsilon$.
2. The function G is assumed to be a random function, so the algorithm DISTINGUISH cannot determine any information about a value $G(r)$ except by calling the function G with input r (therefore the function r is treated as an oracle, and since its output is required to be random, it is termed a *random oracle*).
3. We construct a new algorithm INVERT, by modifying the algorithm DISTINGUISH, which will invert randomly chosen elements y with probability bounded away from 0 (i.e., given a value $y = f(x)$ where x is chosen randomly, the algorithm INVERT will find x with some specified probability).
4. The algorithm INVERT will replace the random oracle by a specific function that we will describe, all of whose outputs are random numbers.

The algorithm INVERT is presented as Algorithm 5.14.

Algorithm 5.14: INVERT(y)

```
external f
global RList, GList, ℓ
procedure G(r)
  i ← 1
  found ← false
  while i ≤ ℓ and not found
     do { if RList[i] = r
        {   then found ← true
        {   else i ← i + 1
  if found
     then return (GList[i])
  if f(r) = y
     then { let j ∈ {1, 2} be chosen at random
          { g ← y_2 − x_j
     else let g be chosen at random
  ℓ ← ℓ + 1
  RList[ℓ] ← r
  GList[ℓ] ← g
  return (g)

main
  y_1 ← y
  choose y_2 at random
  ℓ ← 0
  insert the code for DISTINGUISH(x_1, x_2, (y_1, y_2)) here
  for i ← 1 to ℓ
     do { if f(RList[i]) = y
        {   then return (RList[i])
  return ("failure")
```

The input to INVERT is the element y to be inverted; the objective is to output $f^{-1}(y)$. INVERT begins by constucting a ciphertext (y_1, y_2) in which $y_1 = y$, and y_2 is random. Observe that, for any possible plaintext $x_0 \in X$, (y_1, y_2) is a valid encryption of x_0 if and only if $G(f^{-1}(y_1)) = y_2 + x_0$. In particular, (y_1, y_2) will be a valid encryption of either of x_1 or x_2, provided that $G(f^{-1}(y_1))$ is defined appropriately.

INVERT runs the algorithm DISTINGUISH on the ciphertext (y_1, y_2), attempting to determine if it is an encryption of x_1 or of x_2. DISTINGUISH will query the oracle at various points during its execution. We replace the oracle by a function G that outputs values in exactly the same way as a random oracle, as follows:

1. G maintains a list, denoted $RList$, of all inputs r for which the ora-

cle is queried during the execution of DISTINGUISH; and the corresponding list, denoted $GList$, of outputs $G(r)$.

2. If an input r satisfies $f(r) = y$, then $G(r)$ is defined so that (y_1, y_2) is a valid encryption of one of x_1 or x_2 (chosen at random), as described above.
3. If the oracle was previously queried with input r, then $G(r)$ is already defined.
4. Otherwise, the value for $G(r)$ is chosen randomly.

Eventually, the algorithm DISTINGUISH will terminate with an answer "1" or "2", which may or may not be correct. At this point, the algorithm INVERT examines the list $RList$ to see if any r in the $RList$ satisfies $f(r) = y$. If this r is found, then it is the desired value $f^{-1}(y)$, and the algorithm INVERT succeeds (INVERT fails if $f^{-1}(y)$ is not discovered in $RList$).

It is in fact possible to make algorithm INVERT more efficient by observing that the function G checks to see if $y = f(r)$ for every r that it is queried with. Once it is discovered, within the function G, that $y = f(r)$, we can terminate the algorithm INVERT immediately, returning the value r as its output. It is not necessary to keep running the algorithm DISTINGUISH to its conclusion. However, the analysis of the success probability, which we are going to do next, is a bit easier to understand for Algorithm 5.14 as we have presented it. (The reader might want to verify that the abovementioned modification of INVERT will not change its success probability.)

We now proceed to compute a lower bound on the success probability of the algorithm INVERT. We do this by examining the success probability of DISTINGUISH, which is assumed to be at least $1/2 + \epsilon$. We calculate the success probability of DISTINGUISH, conditioned on whether (or not) $f^{-1}(y) \in RList$:

$$\begin{aligned}
&p(\text{DISTINGUISH succeeds}) = \\
&\quad p(\text{DISTINGUISH succeeds} \mid f^{-1}(y) \in RList)\, p(f^{-1}(y) \in RList) + \\
&\quad p(\text{DISTINGUISH succeeds} \mid f^{-1}(y) \notin RList)\, p(f^{-1}(y) \notin RList).
\end{aligned}$$

It is clear that

$$p(\text{DISTINGUISH succeeds} \mid f^{-1}(y) \notin RList) = 1/2,$$

because there is no way to distinguish an encryption of x_1 from an encryption of x_2 if the value of $G(f^{-1}(y))$ is not determined. Now, using the fact that

$$p(\text{DISTINGUISH succeeds} \mid f^{-1}(y) \in RList) \leq 1,$$

we obtain the following:

$$\frac{1}{2} + \epsilon \leq p(\text{DISTINGUISH succeeds})$$

$$\leq p(f^{-1}(y) \in RList) + \frac{1}{2}\, p(f^{-1}(y) \notin RList)$$
$$\leq p(f^{-1}(y) \in RList) + \frac{1}{2}.$$

Therefore, it follows that

$$p(f^{-1}(y) \in RList) \geq \epsilon.$$

Since

$$p(\text{INVERSE succeeds}) = p(f^{-1}(y) \in RList),$$

it follows that

$$p(\text{INVERSE succeeds}) \geq \epsilon.$$

It is also worthwhile to consider the running time of INVERT as compared to that of DISTINGUISH. Suppose that t_1 is the running time of DISTINGUISH, t_2 is the time required to evaluate the function f, and q denotes the number of oracle queries made by DISTINGUISH. Then it is not difficult to see that the running time of INVERT is $t_1 + O(q^2 + qt_2)$.

We close this section by presenting a more efficient provably secure cryptosystem.

Cryptosystem 5.4: *Optimal Asymmetric Encryption Padding*

Let m, k be positive integers with $m < k$, and let $k_0 = k - m$. Let $\mathcal{F}$ be a family of trapdoor one-way permutations such that $f : (\mathbb{Z}_2)^k \rightarrow (\mathbb{Z}_2)^k$ for all $f \in \mathcal{F}$. Let $G : (\mathbb{Z}_2)^{k_0} \rightarrow (\mathbb{Z}_2)^m$ and let $H : (\mathbb{Z}_2)^m \rightarrow (\mathbb{Z}_2)^{k_0}$ be "random" functions. Define $\mathcal{P} = (\mathbb{Z}_2)^m$, $\mathcal{C} = (\mathbb{Z}_2)^k$, and define

$$\mathcal{K} = \{(f, f^{-1}, G, H) : f \in \mathcal{F}\}.$$

For $K = (f, f^{-1}, G, H)$, let $r \in (\mathbb{Z}_2)^{k_0}$ be chosen randomly, and define

$$e_K(x) = f(y_1 \parallel y_2),$$

where

$$y_1 = x + G(r)$$

and

$$y_2 = r + H(x + G(r))),$$

$x, y_1 \in (\mathbb{Z}_2)^m$, $y_2 \in (\mathbb{Z}_2)^{k_0}$, and "$\|$" denotes concatenation of vectors. Further, define

$$f^{-1}(y) = x_1 \parallel x_2,$$

where $x_1 \in (\mathbb{Z}_2)^m$ and $x_2 \in (\mathbb{Z}_2)^{k_0}$. Then define

$$r = x_2 + H(x_1)$$

and

$$d_K(y) = G(r) + x_1.$$

The functions f, G and H are public; the function f^{-1} is secret.

In Cryptosystem 5.4, it suffices to take k_0 to be large enough that 2^{k_0} is an infeasibly large running time. $k_0 = 128$ should suffice for most applications. The length of a ciphertext in Cryptosystem 5.4 exceeds the length of a plaintext by k_0 bits, so the data expansion is considerably less than Cryptosystem 5.3. The security proof for Cryptosystem 5.4 is more complicated, however.

The adjective "optimal" in Cryptosystem 5.4 refers to the message expansion. Observe that each plaintext has 2^{k_0} possible valid encryptions. One way of solving the problem of Plaintext Distinguishability would be simply to compute all the possible of encryptions of one of the two given plaintexts, say x_1, and check to see if the given ciphertext y is obtained. The complexity of this algorithm is 2^{k_0}. It is therefore clear that the message expansion of the cryptosystem must be at least as big as the logarithm to the base 2 of the amount of computation time of an algorithm that solves the Plaintext Distinguishability problem.