8 Security against Chosen Plaintext Attacks

We've already seen a definition that captures security of encryption when an adversary is allowed to see just one ciphertext encrypted under the key. Clearly a more useful scheme would guarantee security against an adversary who sees an *unlimited* number of ciphertexts.

Fortunately we have arranged things so that we get the "correct" security definition when we modify the earlier definition in a natural way. We simply let the libraries choose a (secret) key for encryption and allow the calling program to request any number of plaintexts to be encrypted under that key. More formally:

Definition 8.1 (CPA security)

*Let Σ be an encryption scheme. We say that Σ has **security against chosen-plaintext attacks (CPA security)** if $\mathcal{L}^{\Sigma}_{\text{cpa-L}} \approx \mathcal{L}^{\Sigma}_{\text{cpa-R}}$, where:*

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$

$k \leftarrow \Sigma.\text{KeyGen}$

$\textsc{challenge}(m_L, m_R \in \Sigma.\mathcal{M})$:
  $c := \Sigma.\text{Enc}(k, m_L)$
  return c

$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$

$k \leftarrow \Sigma.\text{KeyGen}$

$\textsc{challenge}(m_L, m_R \in \Sigma.\mathcal{M})$:
  $c := \Sigma.\text{Enc}(k, m_R)$
  return c

CPA security is often called "IND-CPA" security, meaning "indistinguishability of ciphertexts under chosen-plaintext attack."

8.1 Implications of CPA Security

CPA security is a deceptively simple definition. In this section we will discuss some of the implications of the definition.

CPA Security Protects All Partial Information

A reasonable (informal) security definition for encryption is that "the ciphertext should not leak any partial information about the plaintext." For example, an attacker should not be able to guess, say, the least significant bit or the language of the plaintext, or whether the plaintext is written in ALL-CAPS, etc.

Looking at the CPA security definition, it may not be immediately clear that it ensures protection against partial information leakage. Luckily it does, as we will see from the following illustration.

Let's consider the simple example of an adversary who wishes to guess the least significant bit of a plaintext. Of course, this is already possible with probability 1/2, without even looking at the plaintext. If an adversary can guess with probability signficantly better than 1/2, however, we would accuse the encryption scheme of somehow leaking the least significant bit of the plaintext.

Consider an encryption scheme Σ, and let $\mathsf{lsb}(m)$ denote the least significant bit of a string m. Suppose that there exists an efficient algorithm $\mathcal{B}$ that can guess $\mathsf{lsb}(m)$ given only an encryption of m. That is:

$$\forall m \in \Sigma.\mathcal{M} : \Pr_{k \leftarrow \Sigma.\mathcal{K}}[\mathcal{B}(\Sigma.\mathsf{Enc}(k,m)) = \mathsf{lsb}(m)] \geqslant 1/2 + \epsilon.$$

We will show that ϵ must be negligible if Σ has CPA security. That is, $\mathcal{B}$ cannot guess $\mathsf{lsb}(m)$ much better than chance.

Consider the following distinguisher:

$\mathcal{A}$
choose arbitrary $m_0 \in \Sigma.\mathcal{M}$ with $\mathsf{lsb}(m_0) = 0$
choose arbitrary $m_1 \in \Sigma.\mathcal{M}$ with $\mathsf{lsb}(m_1) = 1$
$c := \textsc{query}(m_0, m_1)$
return $\mathcal{B}(c)$

.

What happens when this distinguisher is linked to the libraries that define CPA security?

- In $\mathcal{A} \diamond \mathcal{L}^{\Sigma}_{\text{cpa-L}}$, the ciphertext c encodes plaintext m_0, whose least significant bit is 0. So the probability that $\mathcal{B}$ will output 1 is at most $1/2 - \epsilon$.
- In $\mathcal{A} \diamond \mathcal{L}^{\Sigma}_{\text{cpa-R}}$, the ciphertext c encodes plaintext m_1, whose least significant bit is 1. So the probability that $\mathcal{B}$ will output 1 is at least $1/2 + \epsilon$.

This distinguisher is efficient and its advantage is at least $|(1/2 + \epsilon) - (1/2 - \epsilon)| = 2\epsilon$. But if Σ is CPA-secure, this advantage must be negligible. Hence ϵ is negligible.

Another way to look at this example is: if the ciphertexts of an encryption scheme leak some partial information about plaintexts, then it is possible to break CPA security. Simply challenge the CPA libraries with two plaintexts whose partial information is different. Then detecting the partial information will tell you which library you are linked to.

Hopefully you can see that there was nothing special about least-significant bits of plaintexts here. Any partial information can be used for the attack.

We could also imagine expanding the capabilities of $\mathcal{B}$. Suppose $\mathcal{B}$ could determine some partial information about a ciphertext by also asking for chosen plaintexts to be encrypted under the same key. Since this extra capability can be done within the CPA libraries, we can still use such a $\mathcal{B}$ to break CPA security if $\mathcal{B}$ is successful.

Impossibility of Deterministic Encryption

We will now explore a not-so-obvious side effect of CPA security, which was first pointed out by Goldwasser & Micali[1]: CPA-secure encryption **cannot be deterministic.** By that,

[1] Shafi Goldwasser & Sivlio Micali: *Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information.* 1982. This paper along with another one led to a Turing Award for the authors.

we mean that calling $\mathsf{Enc}(k,m)$ twice with the same key and same plaintext *must* lead to different outputs, if the scheme is CPA secure.

To see why, consider the following distinguisher $\mathcal{A}$:

$\mathcal{A}$
arbitrarily choose *distinct* plaintexts $x, y \in \mathcal{M}$
$c_1 := \text{QUERY}(x,x)$
$c_2 := \text{QUERY}(x,y)$
return $c_1 \stackrel{?}{=} c_2$

When executed as $\mathcal{A} \diamond \mathcal{L}^{\Sigma}_{\text{cpa-L}}$, we see that c_1 and c_2 will be encryptions of the same plaintext x. When executed as $\mathcal{A} \diamond \mathcal{L}^{\Sigma}_{\text{cpa-R}}$, those ciphertexts will be encryptions of different plaintexts x and y.

Now, suppose the encryption algorithm Enc is a **deterministic** function of the key and the plaintext. When linked to $\mathcal{L}_{\text{cpa-L}}$, we must have $c_1 = c_2$, so $\mathcal{A}$ will always output 1. When linked to $\mathcal{L}_{\text{cpa-R}}$, we must always have $c_1 \neq c_2$ (since otherwise correctness would be violated). Hence, we have described a distinguisher with advantage 1; the scheme is not CPA-secure.

So if deterministic encryption schemes are not secure, *what information is leaking* exactly? We can figure it out by comparing the differences between $\mathcal{A} \diamond \mathcal{L}_{\text{cpa-L}}$ and $\mathcal{A} \diamond \mathcal{L}_{\text{cpa-R}}$. The distinguisher that we constructed is able to tell whether two ciphertexts encrypt the same plaintext. In short, it is able to perform *equality tests* on encrypted data. Ciphertexts are leaking an equality predicate.

We are only now seeing this subtlety arise because this is the first time our security definition allows an adversary to see multiple ciphertexts encrypted under the same key.

This example illustrates a fundamental property of CPA security:

> *If an encryption scheme is CPA-secure, then calling the* Enc *algorithm twice with the same plaintext and key* must *result in different ciphertexts. (For if not, the attacker $\mathcal{A}$ above violates CPA security.)*

We can use this observation as a kind of sanity check. Any time an encryption scheme is proposed, you can ask whether it has a deterministic encryption algorithm. If so, then the scheme cannot be CPA-secure.

Later in this chapter we will see how to construct an encryption scheme whose encryption algorithm is *randomized* and that achieves CPA security.

8.2 Pseudorandom Ciphertexts

You may have noticed a common structure in previous security proofs for encryption (Theorem 2.8 & Claim 5.4). We start with a library in which plaintext m_L was being encrypted. Through a sequence of hybrids we arrive at a hybrid library in which ciphertexts are chosen uniformly at random. This is typically the "half-way point" of the proof, since the steps used before/after this step are mirror images of each other (and with m_R in place of m_L).

CPA security demands that encryptions of m_L are indistinguishable from encryptions of m_R. But is not important that these ciphertext distributions are anything in particular; it's only important that they are indistinguishable for all plaintexts. But in the schemes that we've seen (and indeed, in most schemes that we will see in the future), those distributions happen to look like the uniform distribution. Of course, we have a word for "a distribution that looks like the uniform distribution" — *pseudorandom.*

We can formalize what it means for an encryption scheme to have pseudorandom ciphertexts, and prove that pseudorandom ciphertexts imply CPA security. Then in the future it is enough to show that new encryption schemes have pseudorandom ciphertexts. These proofs will typically be about half as long, since won't have to give a sequence of hybrids whose second half is the "mirror image" of the first half. The idea of getting to a half-way point and then using mirror-image steps is captured once and for all in the proof that pseudorandom ciphertexts implies CPA security.

Definition 8.2 (CPA$ security) *Let Σ be an encryption scheme. We say that Σ has **pseudorandom ciphertexts in the presence of chosen-plaintext attacks (CPA$ security)** if $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$, where:*

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$

$k \leftarrow \Sigma.\mathsf{KeyGen}$

$\textsc{challenge}(m \in \Sigma.\mathcal{M})$:
  $c := \Sigma.\mathsf{Enc}(k, m)$
  return c

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$

$\textsc{challenge}(m \in \Sigma.\mathcal{M})$:
  $c \leftarrow \Sigma.\mathcal{C}$
  return c

This definition is also called "IND$-CPA", meaning "indistinguishable from random under chosen plaintext attacks."

Claim 8.3 *Let Σ be an encryption scheme. If Σ has CPA$ security, then it also has CPA security.*

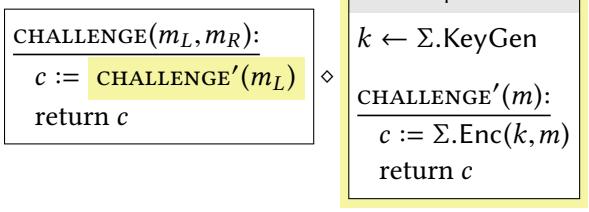
Proof We want to prove that $\mathcal{L}^{\Sigma}_{\text{cpa-L}} \approx \mathcal{L}^{\Sigma}_{\text{cpa-R}}$, *using* the assumption that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$. The sequence of hybrids follows:

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$:

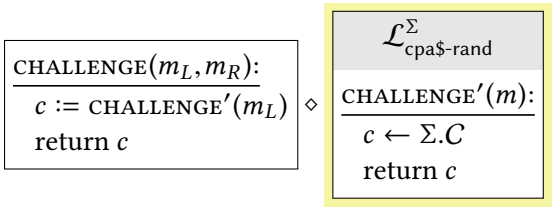
$k \leftarrow \Sigma.\mathsf{KeyGen}$

$\textsc{challenge}(m_L, m_R)$:
  $c := \Sigma.\mathsf{Enc}(k, m_L)$
  return c

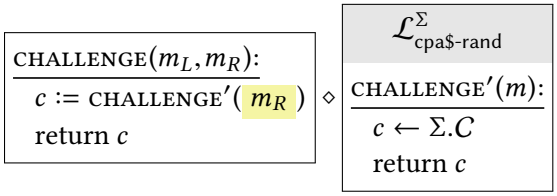
The starting point is $\mathcal{L}^{\Sigma}_{\text{cpa-L}}$, as expected.



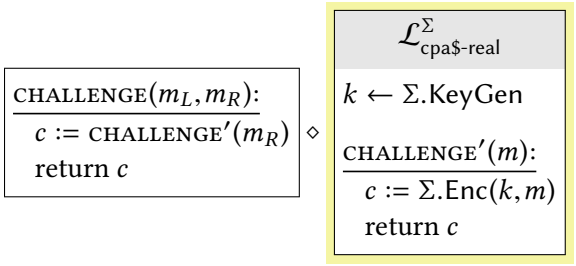
It may look strange, but we have further factored out the call to Enc into a subroutine. It's no accident that the subroutine exactly matches $\mathcal{L}_{\text{cpa\$-real}}$. Since the two libraries have a slight mismatch in their interface (CHALLENGE in $\mathcal{L}_{\text{cpa-L}}$ takes two arguments while CHALLENGE′ in $\mathcal{L}_{\text{cpa\$-real}}$ takes one), the original library still "makes the choice" of which of m_L, m_R to encrypt.



We have replaced $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$ with $\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$. By our assumption, the change is indistinguishable.

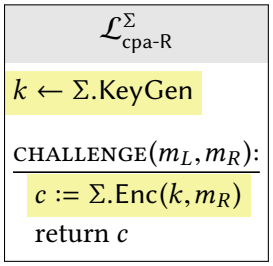


We have changed the argument being passed to CHALLENGE′. This has no effect on the library's behavior since CHALLENGE′ completely ignores its argument in these hybrids.



The mirror image of a previous step; we replace $\mathcal{L}_{\text{cpa\$-rand}}$ with $\mathcal{L}_{\text{cpa\$-real}}$.

$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$:



The $\mathcal{L}_{\text{cpa\$-real}}$ library has been inlined, and the result is $\mathcal{L}^{\Sigma}_{\text{cpa-R}}$.

The sequence of hybrids shows that $\mathcal{L}^{\Sigma}_{\text{cpa-L}} \approx \mathcal{L}^{\Sigma}_{\text{cpa-R}}$, as desired. ■

8.3 CPA-Secure Encryption from PRFs

CPA security presents a significant challenge; its goals seem difficult to reconcile. On the one hand, we need an encryption method that is randomized, so that each plaintext m is mapped to a large number of potential ciphertexts. On the other hand, the decryption method must be able to recognize all of these various ciphertexts as being encryptions of m.

Fortunately, both of these problems can be solved by an appropriate use of PRFs. Intuitively, the only way to encrypt a plaintext is to mask it with an *independent*, pseudorandom one-time pad. Both the sender and receiver must be able to derive these one-time pads from their (short) shared encryption key. Furthermore, they should be able to derive *any polynomial in* λ number of these one-time pads, in order to send any number of messages.

A PRF is then a good fit for the problem. A short PRF key can be used to derive an *exponential* amount of pseudorandom outputs ($F(k,x_1)$, $F(k,x_2)$, ...) which can be used as one-time pads. The only challenge is therefore to decide which PRF outputs to use. The most important factor is that these one-time pads should never be repeated, as we are aware of the pitfalls of reusing one-time pads.

One way to use the PRF is with a counter as its input. That is, the ith message is encrypted using $F(k,i)$ as a one-time pad. While this works, it has the downside that it *requires both sender and receiver to maintain state.*

The approach taken below is to simply choose a value r at random and use $F(k,r)$ as the one-time pad to mask the plaintext. If r is also sent as part of the ciphertext, then the receiver can also compute $F(k,r)$ and recover the plaintext. Furthermore, a value r would be repeated (and hence its mask $F(k,r)$ would be reused) only with negligible probability. It turns out that this construction does indeed achieve CPA security (more specifically, CPA$ security):

Construction 8.4 *Let F be a secure PRF with $in = \lambda$. Define the following encryption scheme based on F:*

$\mathcal{K} = \{0,1\}^\lambda$
$\mathcal{M} = \{0,1\}^{out}$
$\mathcal{C} = \{0,1\}^\lambda \times \{0,1\}^{out}$

KeyGen:
$k \leftarrow \{0,1\}^\lambda$
return k

Enc(k,m):
$r \leftarrow \{0,1\}^\lambda$
$x := F(k,r) \oplus m$
return (r,x)

Dec$(k,(r,x))$:
$m := F(k,r) \oplus x$
return m

Correctness of the scheme can be verified by inspection.

Claim 8.5 *Construction 8.4 has CPA$ security if F is a secure PRF.*

Proof Note that we are proving that the scheme has pseudorandom ciphertexts (CPA$ security). This implies that the scheme has standard CPA security as well. The sequence of hybrids is shown below:

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$:

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$

$k \leftarrow \{0,1\}^{\lambda}$

CHALLENGE(m):
$r \leftarrow \{0,1\}^{\lambda}$
$x := F(k,r) \oplus m$
return (r,x)

The starting point is $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$. The details of Σ have been inlined.

CHALLENGE(m):
$r \leftarrow \{0,1\}^{\lambda}$
$z :=$ QUERY(r)
$x := z \oplus m$
return (r,x)

$\diamond$

$\mathcal{L}^{F}_{\text{prf-real}}$

$k \leftarrow \{0,1\}^{\lambda}$

QUERY(r):
return $F(k,r)$

The statements pertaining to the PRF have been factored out into a subroutine, matching $\mathcal{L}^{F}_{\text{prf-real}}$.

CHALLENGE(m):
$r \leftarrow \{0,1\}^{\lambda}$
$z :=$ QUERY(r)
$x := z \oplus m$
return (r,x)

$\diamond$

$\mathcal{L}^{F}_{\text{prf-rand}}$

$T :=$ empty

QUERY(r):
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{out}$
return $T[r]$

We have replaced $\mathcal{L}^{F}_{\text{prf-real}}$ with $\mathcal{L}^{F}_{\text{prf-rand}}$. From the PRF security of F, these two hybrids are indistinguishable.

At this point in the proof, it seems like we are done. Ciphertexts have the form (r,x), where r is chosen uniformly and x is the result of encrypting the plaintext with what appears to be a one-time pad. Looking more carefuly, however, the "one-time" pad is $T[r]$ — a value that could potentially be used more than once if r is ever repeated!

Put differently, a PRF gives independently random(-looking) outputs on *distinct inputs*. But in our current hybrid there is no guarantee that PRF inputs are distinct! Our proof must explicitly contain reasoning about why PRF inputs are unlikely to be repeated. We do so by appealing to the sampling-with-replacement lemma of Lemma 4.10.

We first factor out the sampling of r values into a subroutine. The subroutine corresponds to the $\mathcal{L}_{\text{samp-L}}$ library of Lemma 4.10:

CHALLENGE(m):
$r \leftarrow$ SAMP()
$z :=$ QUERY(r)
$x := z \oplus m$
return (r,x)

$\diamond$

$\mathcal{L}^{F}_{\text{prf-rand}}$

$T :=$ empty

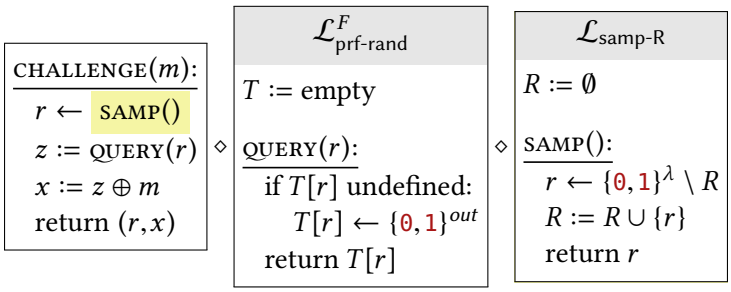
QUERY(r):
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{out}$
return $T[r]$

$\diamond$

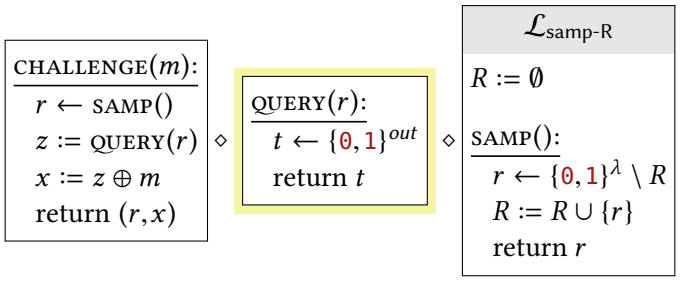
$\mathcal{L}_{\text{samp-L}}$

SAMP():
$r \leftarrow \{0,1\}^{\lambda}$
return r

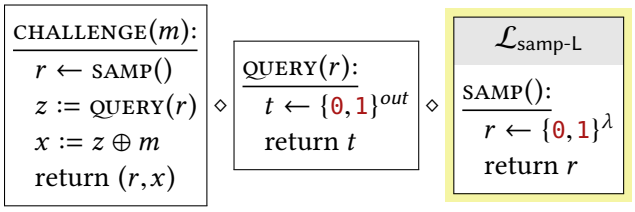
Next, $\mathcal{L}_{\text{samp-L}}$ is replaced by $\mathcal{L}_{\text{samp-R}}$. By Lemma 4.10, the difference is indistinguishable:



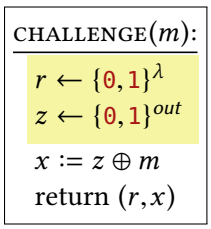
By inspection, the arguments to QUERY are *guaranteed* to never repeat in the previous hybrid, so the $\mathcal{L}_{\text{prf-rand}}$ library can be greatly simplified. In particular, the if-condition in $\mathcal{L}_{\text{prf-rand}}$ is always true. Simplifying has no effect on the library's output behavior:



Now we are indeed using unique one-time pads to mask the plaintext. We are in much better shape than before. Recall that our goal is to arrive at a hybrid in which the outputs of CHALLENGE are chosen uniformly. These outputs include the value r, but now r is no longer being chosen uniformly! We must revert r back to being sampled uniformly, and then it is a simple matter to argue that the outputs of CHALLENGE are generated uniformly.



As promised, $\mathcal{L}_{\text{samp-R}}$ has been replaced by $\mathcal{L}_{\text{samp-L}}$. The difference is indistinguishable due to Lemma 4.10.



All of the subroutine calls have been inlined; no effect on the library's output behavior.

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$:

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$

CHALLENGE(m):
$r \leftarrow \{0,1\}^{\lambda}$
$x \leftarrow \{0,1\}^{out}$
return (r, x)

We have applied the one-time pad rule with respect to variables z and x. In the interest of brevity we have omitted some familiar steps (factor out, replace library, inline) that we have seen several times before. The resulting library is precisely $\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$ since it samples uniformly from $\Sigma.C = \{0,1\}^{\lambda} \times \{0,1\}^{out}$.

The sequence of hybrids shows that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$, so Σ has pseudorandom ciphertexts. ■

Exercises

to-do *Add problems about composing CPA-secure schemes. Or perhaps these can go in the chapter about CCA security where they provide a good contrast.*

8.1. Let Σ be an encryption scheme, and suppose there is a program $\mathcal{A}$ that recovers the key from a chosen plaintext attack. More precisely, $\Pr[\mathcal{A} \diamond \mathcal{L}^{\Sigma}_{\text{cpa}}$ outputs $k]$ is non-negligible, where $\mathcal{L}^{\Sigma}_{\text{cpa}}$ is defined as:

$\mathcal{L}^{\Sigma}_{\text{cpa}}$

$k \leftarrow \Sigma.\text{KeyGen}$

CHALLENGE($m \in \Sigma.\mathcal{M}$):
$c := \Sigma.\text{Enc}(k, m)$
return c

Prove that if such an $\mathcal{A}$ exists, then Σ does not have CPA security. Use $\mathcal{A}$ as a subroutine in a distinguisher that violates the CPA security definition.

In other words, CPA security implies that it should be hard to determine the key from seeing encryptions of chosen plaintexts.

8.2. Let Σ be an encryption scheme with CPA\$ security. Let Σ' be the encryption scheme defined by:

$$\Sigma'.\text{Enc}(k, m) = 00 \| \Sigma.\text{Enc}(k, m)$$

The decryption algorithm in Σ' simply throws away the first two bits of the ciphertext and then calls Σ.Dec.

(a) Does Σ' have CPA\$ security? Prove or disprove (if disproving, show a distinguisher and calculate its advantage).

(b) Does Σ' have CPA security? Prove or disprove (if disproving, show a distinguisher and calculate its advantage).

8.3. Suppose a user is using Construction 8.4 and an adversary observes two ciphertexts that have the same r value.

(a) What exactly does the adversary learn about the plaintexts in this case?

(b) How do you reconcile this with the fact that in the proof of Claim 8.5 there is a hybrid where r values are *never* repeated?

8.4. Let F be a secure PRP with blocklength $blen = \lambda$. Below are several encryption schemes, each with $\mathcal{K} = \mathcal{M} = \{0,1\}^\lambda$ and $C = (\{0,1\}^\lambda)^2$. For each one:

- Give the corresponding Dec algorithm.
- State whether the scheme has CPA security. (Assume KeyGen samples the key uniformly from $\{0,1\}^\lambda$.) If so, then give a security proof. If not, then describe a successful adversary and compute its distinguishing bias.

(a)
$\mathsf{Enc}(k,m)$:
$r \leftarrow \{0,1\}^\lambda$
$z := F(k,m) \oplus r$
return (r,z)

(b)
$\mathsf{Enc}(k,m)$:
$r \leftarrow \{0,1\}^\lambda$
$s := r \oplus m$
$x := F(k,r)$
return (s,x)

(c)
$\mathsf{Enc}(k,m)$:
$r \leftarrow \{0,1\}^\lambda$
$x := F(k, r \oplus m)$
return (r,x)

(d)
$\mathsf{Enc}(k,m)$:
$r \leftarrow \{0,1\}^\lambda$
$x := F(k,r)$
$y := r \oplus m$
return (x,y)

(e)
$\mathsf{Enc}(k,m)$:
$r \leftarrow \{0,1\}^\lambda$
$x := F(k,r)$
$y := F(k,r) \oplus m$
return (x,y)

(f)
$\mathsf{Enc}(k,m)$:
$r \leftarrow \{0,1\}^\lambda$
$x := F(k,r)$
$y := r \oplus F(k,m)$
return (x,y)

(g)
$\mathsf{Enc}(k,m)$:
$s_1 \leftarrow \{0,1\}^\lambda$
$s_2 := s_1 \oplus m$
$x := F(k,s_1)$
$y := F(k,s_2)$
return (x,y)

★ (h)
$\mathsf{Enc}(k,m)$:
$r \leftarrow \{0,1\}^\lambda$
$x := F(k, m \oplus r) \oplus r$
return (r,x)

Hint: In all security proofs, use the PRP switching lemma (Lemma 7.3) since F is a PRP. Parts (b) and (c) differ by an application of Exercise 2.1.

8.5. Let Σ be an encryption scheme with plaintext space $\mathcal{M} = \{0,1\}^n$ and ciphertext space $C = \{0,1\}^n$. Prove that Σ cannot have CPA security.

Conclude that direct application of a PRP to the plaintext is not a good choice for an encryption scheme.

★ 8.6. In all of the CPA-secure encryption schemes that we'll ever see, ciphertexts are at least λ bits longer than plaintexts. This problem shows that such **ciphertext expansion** is essentially unavoidable for CPA security.

Let Σ be an encryption scheme with plaintext space $\mathcal{M} = \{0,1\}^n$ and ciphertext space $C = \{0,1\}^{n+\ell}$. Show that there exists a distinguisher that distinguishes the two CPA libraries with advantage $\Omega(1/2^\ell)$.

Hint: As a warmup, consider the case where each plaintext has *exactly* 2^ℓ possible ciphertexts. However, this need not be true in general. For the general case, choose a random plaintext m and argue that with "good probability" (that you should precisely quantify) m has at most $2^{\ell+1}$ possible ciphertexts.

8.7. Show that an encryption scheme Σ has CPA security if and only if the following two libraries are indistinguishable:

$\mathcal{L}^{\Sigma}_{\text{left}}$

$k \leftarrow \Sigma.\text{KeyGen}$

CHALLENGE($m \in \Sigma.\mathcal{M}$):
return $\Sigma.\text{Enc}(k, m)$

$\mathcal{L}^{\Sigma}_{\text{right}}$

$k \leftarrow \Sigma.\text{KeyGen}$

CHALLENGE($m \in \Sigma.\mathcal{M}$):
$m' \leftarrow \Sigma.\mathcal{M}$
return $\Sigma.\text{Enc}(k, m')$

★ 8.8. Let Σ_1 and Σ_2 be encryption schemes with $\Sigma_1.\mathcal{M} = \Sigma_2.\mathcal{M} = \{0,1\}^n$.

Consider the following approach for encrypting plaintext $m \in \{0,1\}^n$: First, secret-share m into s_1 and s_2, as in the 2-out-of-2 secret-sharing scheme in Construction 3.5. Then encrypt s_1 under Σ_1 and s_2 under Σ_2.

(a) Formally describe this encryption method, including the key generation and decryption procedure.

(b) Prove that the scheme has CPA security if **at least one of** $\{\Sigma_1, \Sigma_2\}$ has CPA security. In other words, it is not necessary that *both* Σ_1 and Σ_2 are secure.