Chapter 4

Symmetric Encryption

The symmetric setting considers two parties who share a key and will use this key to imbue communicated data with various security attributes. The main security goals are privacy and authenticity of the communicated data. The present chapter looks at privacy. A later chapter looks at authenticity. Chapters 3 and 4 describe tools we shall use here.

4.1 Symmetric encryption schemes

The primitive we will consider is called an *encryption scheme.* Such a scheme specifies an *encryption algorithm*, which tells the sender how to process the plaintext using the key, thereby producing the ciphertext that is actually transmitted. An encryption scheme also specifies a *decryption algorithm*, which tells the receiver how to retrieve the original plaintext from the transmission while possibly performing some verification, too. Finally, there is a *key-generation algorithm*, which produces a key that the parties need to share. The formal description follows.

Definition 4.1.1 A *symmetric encryption scheme* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms, as follows:

- The randomized *key generation* algorithm $\mathcal{K}$ returns a string K. We let $\mathsf{Keys}(\mathcal{SE})$ denote the set of all strings that have non-zero probability of being output by $\mathcal{K}$. The members of this set are called *keys*. We write $K \xleftarrow{\$} \mathcal{K}$ for the operation of executing $\mathcal{K}$ and letting K denote the key returned.
- The *encryption* algorithm $\mathcal{E}$, which might be randomized or stateful, takes a key $K \in \mathsf{Keys}(\mathcal{SE})$ and a *plaintext* $M \in \{0,1\}^*$ to return a *ciphertext* $C \in \{0,1\}^* \cup \{\perp\}$. We write $C \xleftarrow{\$} \mathcal{E}_K(M)$ for the operation of executing $\mathcal{E}$ on K and M and letting C denote the ciphertext returned.
- The deterministic *decryption* algorithm $\mathcal{D}$ takes a key $K \in \mathsf{Keys}(\mathcal{SE})$ and a ciphertext $C \in \{0,1\}^*$ to return some $M \in \{0,1\}^* \cup \{\perp\}$. We write $M \leftarrow \mathcal{D}_K(C)$ for the operation of executing $\mathcal{D}$ on K and C and letting M denote the message returned.

The scheme is said to provide *correct decryption* if for any key $K \in \mathsf{Keys}(\mathcal{SE})$, any sequence of messages $M_1, \ldots, M_q \in \{0,1\}^*$, and any sequence of ciphertexts $C_1 \xleftarrow{\$} \mathcal{E}_K(M_1), C_2 \xleftarrow{\$} \mathcal{E}_K(M_2), \ldots, C_q \xleftarrow{\$} \mathcal{E}_K(M_q)$ that may arise in encrypting $M_1, \ldots, M_q$, it is the case that $\mathcal{D}_K(C_i) = M_i$ for each $C_i \neq \perp$.

The key-generation algorithm, as the definition indicates, is randomized. It takes no inputs. When it is run, it flips coins internally and uses these to select a key K. Typically, the key is just a random string of some length, in which case this length is called the *key length* of the scheme. When two parties want to use the scheme, it is assumed they are in possession of a key K generated via $\mathcal{K}$.

How they came into joint possession of this key K in such a way that the adversary did not get to know K is not our concern here, and will be addressed later. For now we assume the key has been shared.

Once in possession of a shared key, the sender can run the encryption algorithm with key K and input message M to get back a string we call the ciphertext. The latter can then be transmitted to the receiver.

The encryption algorithm may be either randomized or stateful. If randomized, it flips coins and uses those to compute its output on a given input K, M. Each time the algorithm is invoked, it flips coins anew. In particular, invoking the encryption algorithm twice on the same inputs may not yield the same response both times.

We say the encryption algorithm is *stateful* if its operation depends on a quantity called the *state* that is initialized in some pre-specified way. When the encryption algorithm is invoked on inputs K, M, it computes a ciphertext based on K, M and the current state. It then updates the state, and the new state value is stored. (The receiver does not maintain matching state and, in particular, decryption does not require access to any global variable or call for any synchronization between parties.) Usually, when there is state to be maintained, the state is just a counter. If there is no state maintained by the encryption algorithm the encryption scheme is said to be *stateless*.

The encryption algorithm might be both randomized and stateful, but in practice this is rare: it is usually one or the other but not both.

When we talk of a *randomized symmetric encryption scheme* we mean that the encryption algorithm is randomized. When we talk of a *stateful symmetric encryption scheme* we mean that the encryption algorithm is stateful.

The receiver, upon receiving a ciphertext C, will run the decryption algorithm with the same key used to create the ciphertext, namely compute $\mathcal{D}_K(C)$. The decryption algorithm is neither randomized nor stateful.

Many encryption schemes restrict the set of strings that they are willing to encrypt. (For example, perhaps the algorithm can only encrypt plaintexts of length a positive multiple of some block length n, and can only encrypt plaintexts of length up to some maximum length.) These kinds of restrictions are captured by having the encryption algorithm return the special symbol $\perp$ when fed a message not meeting the required restriction. In a stateless scheme, there is typically a set of strings $\mathcal{M}$, called the *plaintext space*, such that

$$M \in \mathcal{M} \text{ iff } \Pr[K \xleftarrow{\$} \mathcal{K};\ C \xleftarrow{\$} \mathcal{E}_K(M) :\ C \neq \perp] \ = \ 1$$

In a stateful scheme, whether or not $\mathcal{E}_K(M)$ returns $\perp$ depends not only on M but also possibly on the value of the state variable. For example, when a counter is being used, it is typical that there is a limit to the number of encryptions performed, and when the counter reaches a certain value the encryption algorithm returns $\perp$ no matter what message is fed to it.

The correct decryption requirement simply says that decryption works: if a message M is encrypted under a key K to yield a ciphertext C, then one can recover M by decrypting C under K. This holds, however, only if $C \neq \perp$. The condition thus says that, for each key $K \in \mathsf{Keys}(\mathcal{SE})$ and message $M \in \{0,1\}^*$, with probability one over the coins of the encryption algorithm, either the latter outputs $\perp$ or it outputs a ciphertext C which upon decryption yields M. If the scheme is stateful, this condition is required to hold for every value of the state.

Correct decryption is, naturally, a requirement before one can use a symmetric encryption scheme in practice, for if this condition is not met, the scheme fails to communicate information accurately. In analyzing the security of symmetric encryption schemes, however, we will see that it is sometimes useful to be able to consider ones that do not meet this condition.

4.2 Some symmetric encryption schemes

We now provide a few examples of encryption schemes. We stress that not all of the schemes that follow are *secure* encryption schemes. Some are secure and some are not, as we will see later. All the schemes here satisfy the correct decryption requirement.

4.2.1 The one-time-pad encryption scheme

We begin with the classical one-time-pad.

Scheme 4.2.1 [One-time-pad encryption] The one-time-pad encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is stateful and deterministic. The key-generation algorithm simply returns a random k-bit string K, where the key-length k is a parameter of the scheme, so that the key space is $\mathsf{Keys}(\mathcal{SE}) = \{0,1\}^k$. The encryptor maintains a counter ctr which is initially zero. The encryption and decryption algorithms operate as follows:

algorithm $\mathcal{E}_K(M)$	algorithm $\mathcal{D}_K(\langle ctr, C\rangle)$
Let static $ctr \leftarrow 0$	Let $m \leftarrow \lvert M\rvert$
Let $m \leftarrow \lvert M\rvert$	if $ctr + m > k$ then return $\perp$
if $ctr + m > k$ then return $\perp$	$M \leftarrow C \oplus K[ctr+1 \,..\, ctr+m]$
$C \leftarrow M \oplus K[ctr+1 \,..\, ctr+m]$	return M
$ctr \leftarrow ctr + m$	
return $\langle ctr - m, C\rangle$	

Here $X[i \,..\, j]$ denotes the i-th through j-th bit of the binary string X. By $\langle ctr, C\rangle$ we mean a string that encodes the number ctr and the string C. The most natural encoding is to encode ctr using some fixed number of bits, at least $\lg k$, and to prepend this to C. Conventions are established so that every string Y is regarded as encoding some ctr, C for some ctr, C. The encryption algorithm XORs the message bits with key bits, starting with the key bit indicated by one plus the current counter value. The counter is then incremented by the length of the message. Key bits are not reused, and thus if not enough key bits are available to encrypt a message, the encryption algorithm returns $\perp$. Note that the ciphertext returned includes the value of the counter. This is to enable decryption. (Recall that the decryption algorithm, as per Definition 4.1.1, must be stateless and deterministic, so we do not want it to have to maintain a counter as well.) ▮

4.2.2 Some modes of operation

The following schemes rely either on a family of permutations (i.e., a blockcipher) or a family of functions. Effectively, the mechanisms spell out how to use the blockcipher to encrypt. We call such a mechanism a *mode of operation* of the blockcipher. For these schemes it is convenient to assume that the length of the message to be encrypted is a positive multiple of a block length associated to the family. In practice, one could pad the message appropriately so that the padded message always had length a positive multiple of the block length, and apply the encryption algorithm to

algorithm $\mathcal{E}_K(M)$
  $M[1] \cdots M[m] \leftarrow M$
  for $i \leftarrow 1$ to m do
    $C[i] \leftarrow E_K(M[i])$
  $C \leftarrow C[1] \cdots C[m]$
  return C

---

algorithm $\mathcal{D}_K(C)$
  $C[1] \cdots C[m] \leftarrow C$
  for $i \leftarrow 1$ to m do
    $M[i] \leftarrow E_K^{-1}(C[i])$
  $M \leftarrow M[1] \cdots M[m]$
  return M

Figure 4.1: ECB mode.

the padded message. The padding function should be injective and easily invertible. In this way you would create a new encryption scheme.

With a block length n understood, we will denote by

$$X[i] \dots X[m] \leftarrow X$$

the operation of parsing string X into $m - i + 1$ blocks, each block of length n. Here $i \le m$ and X is assumed to have length $(m - i + 1) \cdot n$. Thus, $X[j]$ consists of bits $(j - i)n + 1$ to $(j - i + 1)n$ of X, for $i \le j \le m$.

The first scheme we consider is ECB (Electronic Codebook Mode), whose security is considered in Section 4.5.1.

Scheme 4.2.2 [ECB mode] Let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Operating it in ECB (Electronic Code Book) mode yields a stateless symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key-generation algorithm simply returns a random key for the blockcipher, meaning it picks a random string $K \xleftarrow{\$} \mathcal{K}$ and returns it. The encryption and decryption algorithms are depicted in Fig. 4.1. Notice that this time the encryption algorithm did not make any random choices. (That does not mean it is not, technically, a randomized algorithm; it is simply a randomized algorithm that happened not to make any random choices.) ▌

The next scheme, cipher-block chaining (CBC) with random initial vector, is the most popular block-cipher mode of operation, used pervasively in practice.

Scheme 4.2.3 [CBC$ mode] Let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Operating it in CBC mode with random IV yields a stateless symmetric encryption scheme, $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key generation algorithm simply returns a random key for the blockcipher, $K \xleftarrow{\$} \mathcal{K}$. The encryption and decryption algorithms are depicted in Fig. 4.2. The IV ("initialization vector") is $C[0]$, which is chosen at random by the encryption algorithm. This choice is made independently each time the algorithm is invoked. ▌

For the following schemes it is useful to introduce some notation. With n fixed, we let $\langle i \rangle$ denote the n-bit string that is the binary representation of integer i mod 2^n. If we use a number $i \ge 0$ in

algorithm $\mathcal{E}_K(M)$
$M[1] \cdots M[m] \leftarrow M$
$C[0] \xleftarrow{\$} \{0,1\}^n$
for $i = 1, \ldots, m$ do
  $C[i] \leftarrow E_K(M[i] \oplus C[i-1])$
return C
	algorithm $\mathcal{D}_K(C)$
	$C[0] \cdots C[m] \leftarrow C$
	for $i = 1, \ldots, m$ do
	  $M[i] \leftarrow E_K^{-1}(C[i]) \oplus C[i-1]$
	return M

Figure 4.2: CBC$ mode.

algorithm $\mathcal{E}_K(M)$
$M[1] \cdots M[m] \leftarrow M$
$C[0] \leftarrow ctr$
for $i = 1, \ldots, m$ do
  $C[i] \leftarrow E_K(M[i] \oplus C[i-1])$
return C
	algorithm $\mathcal{D}_K(C)$
	$C[0] \cdots C[m] \leftarrow C$
	for $i = 1, \ldots, m$ do
	  $M[i] \leftarrow E_K^{-1}(C[i]) \oplus C[i-1]$
	return M

Figure 4.3: CBCC mode.

a context for which a string $I \in \{0,1\}^n$ is required, it is understood that we mean to replace i by $I = [i]_n$. The following is a counter-based version of CBC mode, whose security is considered in Section 4.5.3.

Scheme 4.2.4 [CBCC mode] Let E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Operating it in CBC mode with counter IV yields a stateful symmetric encryption scheme, $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key generation algorithm simply returns a random key for the blockcipher, $K \xleftarrow{\$} \mathcal{K}$. The encryptor maintains a counter *ctr* which is initially zero. The encryption and decryption algorithms are depicted in Fig. 4.3. The IV ("initialization vector") is $C[0]$, which is set to the current value of the counter. The counter is then incremented each time a message is encrypted. The counter is a static variable, meaning that its value is preserved across invocations of the encryption algorithm. ∎

The CTR (counter) modes that follow are not much used, to the best of our knowledge, but perhaps wrongly so. We will see later that they have good privacy properties. In contrast to CBC, the encryption procedure is parallelizable, which can be exploited to speed up the process in the presence of hardware support. It is also the case that the methods work for strings of arbitrary bit lengths, without doing anything "special" to achieve this end. There are two variants of CTR mode, one random and the other stateful, and, as we will see later, their security properties are different. For security analyses see Section 4.7 and Section 4.10.1.

Scheme 4.2.5 [CTR$ mode] Let F: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a family of functions. (Possibly a blockcipher, but not necessarily.) Then CTR mode over F with a random starting point is a probabilistic, stateless symmetric encryption scheme, $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key-generation algorithm simply returns a random key for E. The encryption and decryption algorithms are depicted in Fig. 4.4. The starting point $C[0]$ is used to define a sequence of values on which F_K is applied to produce a "pseudo one-time pad" to which the plaintext is XORed. The starting point $C[0]$

algorithm $\mathcal{E}_K(M)$
$M[1] \cdots M[m] \leftarrow M$
$C[0] \xleftarrow{\$} \{0,1\}^n$
for $i = 1, \ldots, m$ do
    $P[i] \leftarrow F_K(C[0] + i)$
    $C[i] \leftarrow P[i] \oplus M[i]$
return C

algorithm $\mathcal{D}_K(C)$
$C[0] \cdots C[m] \leftarrow C$
for $i = 1, \ldots, m$ do
    $P[i] \leftarrow F_K(C[0] + i)$
    $M[i] \leftarrow P[i] \oplus C[i]$
return M

Figure 4.4: CTR$ mode using a family of functions F: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. This version of counter mode is randomized and stateless.

algorithm $\mathcal{E}_K(M)$
$M[1] \cdots M[m] \leftarrow M$
$C[0] \leftarrow$ ctr
for $i = 1, \ldots, m$ do
    $P[i] \leftarrow F_K(ctr + i)$
    $C[i] \leftarrow P[i] \oplus M[i]$
ctr $\leftarrow$ ctr + m
return C

algorithm $\mathcal{D}_K(C)$
$C[0] \cdots C[m] \leftarrow C$
ctr $\leftarrow C[0]$
for $i = 1, \ldots, m$ do
    $P[i] \leftarrow F_K(ctr + i)$
    $M[i] \leftarrow P[i] \oplus C[i]$
return M

Figure 4.5: CTRC mode using a family of functions F: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. This version of counter mode uses stateful (but deterministic) encryption.

chosen by the encryption algorithm is a random n-bit string. To add an n-bit string $C[0]$ to an integer i—when we write $F_K(R + i)$—convert the n-bit string $C[0]$ into an integer in the range $[0 \,..\, 2^n - 1]$ in the usual way, add this number to i, take the result modulo 2^n, and then convert this back into an n-bit string. Note that the starting point $C[0]$ is included in the ciphertext, to enable decryption. ▮

We now give the counter-based version of CTR mode.

Scheme 4.2.6 [CTRC mode] Let F: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a family of functions (possibly a blockcipher, but not necessarily). Operating it in CTR mode with a counter starting point is a stateful symmetric encryption scheme, $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, which we call CTRC. The key-generation algorithm simply returns a random key for F. The encryptor maintains a counter *ctr* which is initially zero. The encryption and decryption algorithms are depicted in Fig. 4.5. Position index *ctr* is not allowed to wrap around: the encryption algorithm returns $\perp$ if this would happen. The position index is included in the ciphertext in order to enable decryption. The encryption algorithm updates the position index upon each invocation, and begins with this updated value the next time it is invoked. ▮

We will return to the security of these schemes after we have developed the appropriate notions.

4.3 Issues in privacy

Let us fix a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Two parties share a key K for this scheme, this key having being generated as $K \xleftarrow{\$} \mathcal{K}$. The adversary does not a priori know K. We now want to explore the issue of what the privacy of the scheme might mean. For this chapter, security *is* privacy, and we are trying to get to the heart of what security is about.

The adversary is assumed able to capture any ciphertext that flows on the channel between the two parties. It can thus collect ciphertexts, and try to glean something from them. Our first question is: what exactly does "glean" mean? What tasks, were the adversary to accomplish them, would make us declare the scheme insecure? And, correspondingly, what tasks, were the adversary unable to accomplish them, would make us declare the scheme secure?

It is easier to think about *insecurity* than security, because we can certainly identify adversary actions that indubitably imply the scheme is insecure. So let us begin here.

For example, if the adversary can, from a few ciphertexts, derive the underlying key K, it can later decrypt anything it sees, so if the scheme allowed easy key recovery from a few ciphertexts it is definitely insecure.

Now, the mistake that is often made is to go on to reverse this, saying that if key recovery is hard, then the scheme is secure. This is certainly not true, for there are other possible weaknesses. For example, what if, given the ciphertext, the adversary could easily recover the plaintext M without finding the key? Certainly the scheme is insecure then too.

So should we now declare a scheme secure if it is hard to recover a plaintext from the ciphertext? Many people would say yes. Yet, this would be wrong too.

One reason is that the adversary might be able to figure out *partial information* about M. For example, even though it might not be able to recover M, the adversary might, given C, be able to recover the first bit of M, or the sum of all the bits of M. This is not good, because these bits might carry valuable information.

For a concrete example, say I am communicating to my broker a message which is a sequence of "buy" or "sell" decisions for a pre-specified sequence of stocks. That is, we have certain stocks, numbered 1 through m, and bit i of the message is 1 if I want to buy stock i and 0 otherwise. The message is sent encrypted. But if the first bit leaks, the adversary knows whether I want to buy or sell stock 1, which may be something I don't want to reveal. If the sum of the bits leaks, the adversary knows how many stocks I am buying.

Granted, this might not be a problem at all if the data were in a different format. However, making assumptions, or requirements, on how users format data, or how they use it, is a bad and dangerous approach to secure protocol design. An important principle of good cryptographic design is that the encryption scheme should provide security regardless of the format of the plaintext. Users should not have to worry about the how they format their data: they format it as they like, and encryption should provide privacy nonetheless.

Put another way, as designers of security protocols, we should not make assumptions about data content or formats. Our protocols must protect any data, no matter how formatted. We view it as the job of the protocol designer to ensure this is true.

At this point it should start becoming obvious that there is an infinite list of insecurity properties, and we can hardly attempt to characterize security as their absence. We need to think about security in a different and more direct way and arrive at some definition of it.

This important task is surprisingly neglected in many treatments of cryptography, which will provide you with many schemes and attacks, but never actually define the goal by saying what an encryption scheme is actually trying to achieve and when it should be considered secure rather than merely not known to be insecure. This is the task that we want to address.

One might want to say something like: the encryption scheme is secure if given C, the adversary has no idea what M is. This however cannot be true, because of what is called *a priori* information. Often, something about the message is known. For example, it might be a packet with known headers. Or, it might be an English word. So the adversary, and everyone else, has some information about the message even before it is encrypted.

We want schemes that are secure in the strongest possible natural sense. What is the best we could hope for? It is useful to make a thought experiment. What would an "ideal" encryption be like? Well, it would be as though some angel took the message M from the sender and delivered it to the receiver, in some magic way. The adversary would see nothing at all. Intuitively, our goal is to approximate this as best as possible. We would like encryption to have the properties of ideal encryption. In particular, no partial information would leak.

We do deviate from the ideal in one way, though. Encryption is not asked to hide the length of the plaintext string. This information not only can leak but is usually supposed to be known to the adversary a priori.

As an example, consider the ECB encryption scheme of Scheme 4.2.2. Given the ciphertext, can an eavesdropping adversary figure out the message? It is hard to see how, since it does not know K, and if F is a "good" blockcipher, then it ought to have a hard time inverting F_K without knowledge of the underlying key. Nonetheless this is not a good scheme. Consider just the case $n = 1$ of a single block message. Suppose a missile command center has just two messages, 1^n for *fire* and 0^n for *don't fire*. It keeps sending data, but always one of these two. What happens? When the first ciphertext C_1 goes by, the adversary may not know what is the plaintext. But then, let us say it sees a missile taking off. Now, it knows the message M_1 underlying C_1 was 1^n. But then it can easily decrypt all subsequent messages, for if it sees a ciphertext C, the message is 1^n if $C = C_1$ and 0^n if $C \neq C_1$.

In a secure encryption scheme, it should not be possible to relate ciphertexts of different messages of the same length in such a way that information is leaked.

Not allowing message-equalities to be leaked has a dramatic implication. Namely, *encryption must be probabilistic* or *depend on state information*. If not, you can always tell if the same message was sent twice. Each encryption must use fresh coin tosses, or, say, a counter, and an encryption of a particular message may be different each time. In terms of our setup it means $\mathcal{E}$ is a *probabilistic* or *stateful* algorithm. That's why we defined symmetric encryption schemes, above, to allow these types of algorithms.

The reason this is dramatic is that it goes in many ways against the historical or popular notion of encryption. Encryption was once thought of as a code, a fixed mapping of plaintexts to ciphertexts. But this is not the contemporary viewpoint. A single plaintext should have many possible ciphertexts (depending on the random choices or the state of the encryption algorithm). Yet it must be possible to decrypt. How is this possible? We have seen several examples above.

One formalization of privacy is what is called *perfect security*, an information-theoretic notion introduced by Shannon and showed by him to be met by the one-time pad scheme, and covered in Chapter 2. Perfect security asks that regardless of the computing power available to the adversary, the ciphertext provides it no information about the plaintext beyond the a priori information it had prior to seeing the ciphertext. Perfect security is a very strong attribute, but achieving it requires a key as long as the total amount of data encrypted, and this is not usually practical. So here we look at a notion of *computational security*. The security will only hold with respect to adversaries of limited computing power. If the adversary works harder, she can figure out more, but a "feasible" amount of effort yields no noticeable information. This is the important notion for us and will be used to analyze the security of schemes such as those presented above.

4.4 Indistinguishability under chosen-plaintext attack

Having discussed the issues in Section 4.3 above, we will now distill a formal definition of security.

4.4.1 Definition

The basic idea behind indistinguishability (or, more fully, *left-or-right indistinguishability under a chosen-plaintext attack*) is to consider an adversary (not in possession of the secret key) who chooses two messages of the same length. Then one of the two messages is encrypted, and the ciphertext is given to the adversary. The scheme is considered secure if the adversary has a hard time telling which of the two messages was the one encrypted.

We will actually give the adversary a little more power, letting her choose a whole sequence of pairs of equal-length messages. Let us now detail the game.

The adversary chooses a sequence of pairs of messages, $(M_{0,1}, M_{1,1}), \ldots, (M_{0,q}, M_{1,q})$, where, in each pair, the two messages have the same length. We give to the adversary a sequence of ciphertexts $C_1, \ldots, C_q$ where either (1) C_i is an encryption of $M_{0,i}$ for all $1 \le i \le q$ or, (2) C_i is an encryption of $M_{1,i}$ for all $1 \le i \le q$. In doing the encryptions, the encryption algorithm uses the same key but fresh coins, or an updated state, each time. The adversary gets the sequence of ciphertexts and now it must guess whether $M_{0,1}, \ldots, M_{0,q}$ were encrypted or $M_{1,1}, \ldots, M_{1,q}$ were encrypted.

To further empower the adversary, we let it choose the sequence of message pairs via a *chosen plaintext attack*. This means that the adversary chooses the first pair, then receives C_1, then chooses the second pair, receives C_2, and so on. (Sometimes this is called an *adaptive* chosen-plaintext attack, because the adversary can adaptively choose each query in a way responsive to the earlier answers.)

Let us now formalize this. We fix some encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. It could be either stateless or stateful. We consider an adversary A. It is a program which has access to an oracle that we call LR (left or right) oracle. A can provide as input any pair of equal-length messages. The oracle will return a ciphertext. We will consider two possible ways in which this ciphertext is computed by the oracle, corresponding to two possible "worlds" in which the adversary "lives". In the "right" world, the oracle, given query M_0, M_1, runs $\mathcal{E}$ with key K and input M_1 to get a ciphertext C which it returns. In the "left" world, the oracle, given M_0, M_1, runs $\mathcal{E}$ with key K and input M_0 to get a ciphertext C which it returns. The problem for the adversary is, after talking to its oracle for some time, to tell which of the two oracles it was given. The formalization uses the game $\text{Left}_{\mathcal{SE}}$ and $\text{Right}_{\mathcal{SE}}$ of Fig. 4.6. The game begins by picking at random encryption key K. (The key is not returned to the adversary.) Each game then defines a **LR** oracle, to which the adversary may make multiple queries, each query being a pair of equal-length strings. The adversary outputs a bit which become the output of the game.

First assume the given symmetric encryption scheme $\mathcal{SE}$ is stateless. The oracle, in either world, is probabilistic, because it calls the encryption algorithm. Recall that this algorithm is probabilistic. Above, when we say $C \xleftarrow{\$} \mathcal{E}_K(M_b)$, it is implicit that the oracle picks its own random coins and uses them to compute ciphertext C.

The random choices of the encryption function are somewhat "under the rug" here, not being explicitly represented in the notation. But these random bits should not be forgotten. They are central to the meaningfulness of the notion and the security of the schemes.

If the given symmetric encryption scheme $\mathcal{SE}$ is stateful, the oracles, in either world, become stateful, too. (Think of a subroutine that maintains a "static" variable across successive calls.) An oracle begins with a state value initialized to a value specified by the encryption scheme. For

Game $\text{Left}_{\mathcal{SE}}$	Game $\text{Right}_{\mathcal{SE}}$
procedure Initialize	**procedure** Initialize
$K \xleftarrow{\$} \mathcal{K}$	$K \xleftarrow{\$} \mathcal{K}$
procedure LR(M_0, M_1)	**procedure LR**(M_0, M_1)
Return $C \xleftarrow{\$} \mathcal{E}_K(M_0)$	Return $C \xleftarrow{\$} \mathcal{E}_K(M_1)$

Figure 4.6: Games used to define IND-CPA.

example, in CTRC mode, the state is an integer *ctr* that is initialized to 0. Now, each time the oracle is invoked, it computes $\mathcal{E}_K(M_b)$ according to the specification of algorithm $\mathcal{E}$. The algorithm may, as a side-effect, update the state, and upon the next invocation of the oracle, the new state value will be used.

The following definition associates to a symmetric encryption scheme $\mathcal{SE}$ and an adversary A a pair of experiments, one capturing each of the worlds described above. The adversary's advantage, which measures its success in breaking the scheme, is the difference in probabilities of the two games returning the bit one.

Definition 4.4.1 Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, and let A be an algorithm that has access to an oracle. We consider two games as described in Fig. 4.6. The *ind-cpa advantage* of A is defined as

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) = \Pr\left[\text{Right}^{A}_{\mathcal{SE}} \Rightarrow 1\right] - \Pr\left[\text{Left}^{A}_{\mathcal{SE}} \Rightarrow 1\right]$$

As the above indicates, the choice of which world we are in is made just once, at the beginning, before the adversary starts to interact with the oracle. In the left world, *all* message pairs sent to the oracle are answered by the oracle encrypting the left message in the pair, while in the right world, all message pairs are answered by the oracle encrypting the right message in the pair. The choice of which does not flip-flop from oracle query to oracle query.

If $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A)$ is small (meaning close to zero), it means that A is outputting 1 about as often in world 0 as in world 1, meaning it is not doing a good job of telling which world it is in. If this quantity is large (meaning close to one—or at least far from zero) then the adversary A is doing well, meaning our scheme $\mathcal{SE}$ is not secure, at least to the extent that we regard A as "reasonable."

Informally, for symmetric encryption scheme $\mathcal{SE}$ to be secure against chosen plaintext attack, the ind-cpa advantage of an adversary must be small, no matter what strategy the adversary tries. However, we have to be realistic in our expectations, understanding that the advantage may grow as the adversary invests more effort in its attack. Security is a measure of how large the advantage of the adversary might when compared against the adversary's resources.

We consider an encryption scheme to be "secure against chosen-plaintext attack" if an adversary restricted to using "practical" amount of resources (computing time, number of queries) cannot obtain "significant" advantage. The technical notion is called left-or-right indistinguishability under chosen-plaintext attack, denoted IND-CPA.

We discuss some important conventions regarding the resources of adversary A. The *running time* of an adversary A is the worst case execution time of A over all possible coins of A and all conceivable oracle return values (including return values that could never arise in the experiments used to define the advantage). Oracle queries are understood to return a value in unit time, but it takes the adversary one unit of time to read any bit that it chooses to read. By convention, the

Game $\text{Guess}_{\mathcal{SE}}$

procedure Initialize
$K \xleftarrow{\$} \mathcal{K}$; $b \xleftarrow{\$} \{0,1\}$

procedure LR(M_0, M_1)
return $C \xleftarrow{\$} \mathcal{E}_K(M_b)$

procedure Finalize(b')
return $(b = b')$

Figure 4.7: Game used to redefine IND-CPA.

running time of A also includes the size of the code of the adversary A, in some fixed RAM model of computation. This convention for measuring time complexity is the same as used in other parts of these notes, for all kinds of adversaries.

Other resource conventions are specific to the IND-CPA notion. When the adversary asks its left-or-right encryption oracle a query (M_0, M_1) we say that length of this query is $\max(|M_0|, |M_1|)$. (This will equal $|M_0|$ since we require that any query consist of equal-length messages.) The total length of queries is the sum of the length of each query. We can measure query lengths in bits or in blocks, with block having some understood number of bits n.

The resources of the adversary we will typically care about are three. First, its time-complexity, measured according to the convention above. Second, the number of oracle queries, meaning the number of message pairs the adversary asks of its oracle. These messages may have different lengths, and our third resource measure is the sum of all these lengths, denoted μ, again measured according to the convention above.

4.4.2 Alternative interpretation

Let us move on to describe a somewhat different interpretation of indistinguishability. Why is $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A)$ called the "advantage" of the adversary? We can view the task of the adversary as trying to guess which world it is in. A trivial guess is for the adversary to return a random bit. In that case, it has probability 1/2 of being right. Clearly, it has not done anything damaging in this case. The advantage of the adversary measures how much better than this it does at guessing which world it is in, namely the excess over 1/2 of the adversary's probability of guessing correctly. In this subsection we will see how the above definition corresponds to this alternative view, a view that lends some extra intuition to the definition and is also useful in later usages of the definition.

Proposition 4.4.2 Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, and let A be an algorithm that has access to an oracle that takes input a pair of strings and returns a string. We consider the game as described in Fig. 4.7. Then

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) = 2 \cdot \Pr\left[\text{Guess}^{A}_{\mathcal{SE}} \Rightarrow \mathsf{true}\right] - 1 \ .$$

In the above game, the **LR** oracle provided to an adversary A is as in $\text{Left}_{\mathcal{SE}}$ if $b = 0$ and as in $\text{Right}_{\mathcal{SE}}$ if $b = 1$, where the bit b is chosen at random. A eventually outputs a bit b', its guess as to the value of b. Thus,

$$\Pr\left[\text{Guess}^{A}_{\mathcal{SE}} \Rightarrow \mathsf{true}\right]$$

is the probability that A correctly guesses which world it is in. The probability is over the initial choice of world as given by the bit b, the choice of K, the random choices of $\mathcal{E}_K(\cdot)$ if any, and the coins of A if any. This value is 1/2 when the adversary deserves no advantage, since one can guess b correctly by a strategy as simple as "always answer zero" or "answer with a random bit." The

"advantage" of A can thus be viewed as the excess of this probability over 1/2, which, re-scaled, is

$$2 \cdot \Pr\left[\mathbf{Exp}^{\text{ind-cpa-cg}}_{\mathcal{SE}}(A) = 1\right] - 1 \ .$$

The Proposition says that this rescaled advantage is exactly the same measure as before.

Proof of Proposition 4.4.2: We let $\Pr[\cdot]$ be the probability of event "$\cdot$" in the game $\text{Guess}^A_{\mathcal{SE}}$, and refer below to quantities in this game. The claim of the Proposition follows by a straightforward calculation:

$$\begin{aligned}
&\Pr\left[\text{Guess}^A_{\mathcal{SE}} \Rightarrow \mathsf{true}\right] \\
&= \Pr[b = b'] \\
&= \Pr[b = b' \mid b = 1] \cdot \Pr[b = 1] + \Pr[b = b' \mid b = 0] \cdot \Pr[b = 0] \\
&= \Pr[b = b' \mid b = 1] \cdot \frac{1}{2} + \Pr[b = b' \mid b = 0] \cdot \frac{1}{2} \\
&= \Pr[b' = 1 \mid b = 1] \cdot \frac{1}{2} + \Pr[b' = 0 \mid b = 0] \cdot \frac{1}{2} \\
&= \Pr[b' = 1 \mid b = 1] \cdot \frac{1}{2} + (1 - \Pr[b' = 1 \mid b = 0]) \cdot \frac{1}{2} \\
&= \frac{1}{2} + \frac{1}{2} \cdot (\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \left(\Pr\left[\text{Right}^A_{\mathcal{SE}} \Rightarrow 1\right] - \Pr\left[\text{Left}^A_{\mathcal{SE}} \Rightarrow 1\right]\right) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \ .
\end{aligned}$$

We began by expanding the quantity of interest via standard conditioning. The term of 1/2 in the third line emerged because the choice of b is made at random. In the fourth line we noted that if we are asking whether $b = b'$ given that we know $b = 1$, it is the same as asking whether $b' = 1$ given $b = 1$, and analogously for $b = 0$. In the fifth line and sixth lines we just manipulated the probabilities and simplified. The next line is important; here we observed that the conditional probabilities in question are exactly the probabilities that A returns 1 in the games of Definition 4.4.1. ▮

4.4.3 Why is this a good definition?

Our thesis is that we should consider an encryption scheme to be "secure" if and only if it is IND-CPA secure, meaning that the above formalization captures our intuitive sense of privacy, and the security requirements that one might put on an encryption scheme can be boiled down to this one.

But why? Why does IND-CPA capture "privacy"? This is an important question to address and answer.

In particular, here is one concern. In Section 4.3 we noted a number of security properties that are necessary but not sufficient for security. For example, it should be computationally infeasible for an adversary to recover the key from a few plaintext-ciphertext pairs, or to recover a plaintext from a ciphertext.

A test of our definition is that it implies the necessary properties that we have discussed, and others. For example, a scheme that is secure in the IND-CPA sense of our definition should also be, automatically, secure against key-recovery or plaintext-recovery. Later, we will prove such things,

and even stronger things. For now, let us continue to get a better sense of how to work with the definition by using it to show that certain schemes are insecure.

4.5 Example chosen-plaintext attacks

We illustrate the use of our IND-CPA definition in finding attacks by providing an attack on ECB mode, and also a general attack on deterministic, stateless schemes.

4.5.1 Attack on ECB

Let us fix a blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. The ECB symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ was described as Scheme 4.2.2. Suppose an adversary sees a ciphertext $C = \mathcal{E}_K(M)$ corresponding to some random plaintext M, encrypted under the key K also unknown to the adversary. Can the adversary recover M? Not easily, if E is a "good" blockcipher. For example if E is AES, it seems quite infeasible. Yet, we have already discussed how infeasibility of recovering plaintext from ciphertext is not an indication of security. ECB has other weaknesses. Notice that if two plaintexts M and M' agree in the first block, then so do the corresponding ciphertexts. So an adversary, given the ciphertexts, can tell whether or not the first blocks of the corresponding plaintexts are the same. This is loss of partial information about the plaintexts, and is not permissible in a secure encryption scheme.

It is a test of our definition to see that it captures these weaknesses and also finds the scheme insecure. It does. To show this, we want to show that there is an adversary that has a high IND-CPA advantage while using a small amount of resources. We now construct such an adversary A. Remember that A is given a lr-encryption oracle $\mathbf{LR}(\cdot,\cdot)$ that takes as input a pair of messages and that returns an encryption of either the left or the right message in the pair, depending on the value of the bit b. The goal of A is to determine the value of b. Our adversary works like this:

Adversary A
  $M_1 \leftarrow 0^{2n}$; $M_0 \leftarrow 0^n \,\|\, 1^n$
  $C[1]C[2] \leftarrow \mathbf{LR}(M_0, M_1)$
  If $C[1] = C[2]$ then return 1 else return 0

Above, $X[i]$ denotes the i-th block of a string X, a block being a sequence of n bits. The adversary's single oracle query is the pair of messages M_0, M_1. Since each of them is two blocks long, so is the ciphertext computed according to the ECB scheme. Now, we claim that

$$\begin{aligned} \Pr\left[\mathrm{Right}^A_{\mathcal{SE}} \Rightarrow 1\right] &= 1 \text{ and} \\ \Pr\left[\mathrm{Left}^A_{\mathcal{SE}} \Rightarrow 1\right] &= 0\,. \end{aligned}$$

Why? You have to return to the definitions of the quantities in question. In the right world, the oracle returns $C[1]C[2] = E_K(0^n) \,\|\, E_K(0^n)$, so $C[1] = C[2]$ and A returns 1. In the left world, the oracle returns $C[1]C[2] = E_K(0^n)E_K(1^n)$. Since E_K is a permutation, $C[1] \neq C[2]$. So A returns 0 in this case.

Subtracting, we get $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) = 1 - 0 = 1$. And A achieved this advantage by making just one oracle query, whose length, which as per our conventions is just the length of M_0, is $2n$ bits. This means that the ECB encryption scheme is insecure.

As an exercise, try to analyze the same adversary as an adversary against CBC\$ or CTR modes, and convince yourself that the adversary will not get a high advantage.

There is an important feature of this attack that must be emphasized. Namely, ECB is an insecure encryption scheme *even if the underlying blockcipher E is highly secure*. The weakness is not in the tool being used (here the blockcipher) but in the manner we are using it. It is the ECB mechanism that is at fault. Even the best of tools are useless if you don't know how to properly use them.

This is the kind of design flaw that we want to be able to spot and eradicate. Our goal is to find symmetric encryption schemes that are secure as long as the underlying blockcipher is secure. In other words, the scheme has no inherent flaw; as long as you use good ingredients, the recipe will produce a good meal.

If you don't use good ingredients? Well, that is your problem. All bets are off.

4.5.2 Any deterministic, stateless schemes is insecure

ECB mode is deterministic and stateless, so that if the same message is encrypted twice, the same ciphertext is returned. It turns out that this property, in general, results in an insecure scheme, and provides perhaps a better understanding of why ECB fails. Let us state the general fact more precisely.

Proposition 4.5.1 Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a deterministic, stateless symmetric encryption scheme. Assume there is an integer m such that the plaintext space of the scheme contains two distinct strings of length m. Then there is an adversary A such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;=\; 1\,.$$

Adversary A runs in time $O(m)$ and asks just two queries, each of length m. ∎

The requirement being made on the message space is minimal; typical schemes have messages spaces containing all strings of lengths between some minimum and maximum length, possibly restricted to strings of some given multiples. Note that this Proposition applies to ECB and is enough to show the latter is insecure.

Proof of Proposition 4.5.1: We must describe the adversary A. Remember that A is given an lr-encryption oracle $\mathbf{LR}(\cdot,\cdot)$ that takes input a pair of messages and returns an encryption of either the left or the right message in the pair. The goal of A is to determine which. Our adversary works like this:

Adversary A
  Let X, Y be distinct, m-bit strings in the plaintext space
  $C_1 \leftarrow \mathbf{LR}(X, Y)$
  $C_2 \leftarrow \mathbf{LR}(Y, Y)$
  If $C_1 = C_2$ then return 1 else return 0

Now, we claim that

$$\begin{aligned} \Pr\left[\text{Right}^A_{\mathcal{SE}} \Rightarrow 1\right] &= 1 \text{ and} \\ \Pr\left[\text{Left}^A_{\mathcal{SE}} \Rightarrow 1\right] &= 0\,. \end{aligned}$$

Why? In the right world, the oracle returns $C_1 = \mathcal{E}_K(Y)$ and $C_2 = \mathcal{E}_K(Y)$, and since the encryption function is deterministic and stateless, $C_1 = C_2$, so A returns 1. In the left world, the oracle returns

$C_1 = \mathcal{E}_K(X)$ and $C_2 = \mathcal{E}_K(Y)$, and since it is required that decryption be able to recover the message, it must be that $C_1 \neq C_2$. So A returns 0.

Subtracting, we get $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) = 1 - 0 = 1$. And A achieved this advantage by making two oracle queries, each of whose length, which as per our conventions is just the length of the first message, is m bits. ▮

4.5.3 Attack on CBC encryption with counter IV

Let us fix a blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding counter-based version of the CBC encryption mode described in Scheme 4.2.4. We show that this scheme is insecure. The reason is that the adversary can predict the counter value.

To justify our claim of insecurity, we present an adversary A. As usual it is given an lr-encryption oracle $\mathbf{LR}(\cdot,\cdot)$ and wants to determine whether the left or right message is being encrypted. Our adversary works like this:

Adversary A
$M_{0,1} \leftarrow 0^n$; $M_{1,1} \leftarrow 0^n$
$M_{0,2} \leftarrow 0^n$; $M_{1,2} \leftarrow 0^{n-1}1$
$C_1 \xleftarrow{\$} \mathbf{LR}(M_{0,1}, M_{1,1})$
$C_2 \xleftarrow{\$} \mathbf{LR}(M_{0,2}, M_{1,2})$
If $C_1 = C_2$ then return 1 else return 0

We claim that

$$\Pr\left[\text{Right}^A_{\mathcal{SE}} \Rightarrow 1\right] = 1 \text{ and}$$
$$\Pr\left[\text{Left}^A_{\mathcal{SE}} \Rightarrow 1\right] = 0\,.$$

Why? First consider the left world,. In that case $C_1[0] = 0$ and $C_2[0] = 1$ and $C_1[1] = E_K(0)$ and $C_2[1] = E_K(1)$ and so $C_1 \neq C_2$ and A returns 0. On the other hand, if we are in the right world, then $C_1[0] = 0$ and $C_2[0] = 1$ and $C_1[1] = E_K(0)$ and $C_2[1] = E_K(0)$, so A returns 1.

Subtracting, we get $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) = 1 - 0 = 1$, showing that A has a very high advantage. Moreover, A is practical, using very few resources. So the scheme is insecure.

4.6 Semantic security

In this section we describe an alternative notion of encryption-scheme security, *semantic security*, again under a chosen-plaintext attack. We will abbreviate this notion as SEM-CPA. It captures the idea that a secure encryption scheme should hide all information about an unknown plaintext. This definition may match our intuition about what secure encryption ought to achieve better than does IND-CPA. We then show that IND-CPA implies SEM-CPA. (In fact, they are equivalent.) By showing that our IND-CPA notion implies SEM-CPA we gain confidence that our definition appropriately models privacy.

Semantic security, which was introduced by Goldwasser and Micali for public-key encryption, transfers the intuition of Shannon's notion of security to a setting where security is not absolute but dependent on the computational effort made by an adversary. Shannon says that an encryption scheme is secure if *that which can be determined about a plaintext from its ciphertext can be*

determined in the absence of the ciphertext. Semantic security asks that *that which can be efficiently computed about some plaintexts from their ciphertexts can be computed, just as easily, in the absence of those ciphertexts.*

Our formalization allows an adversary to choose a message space $\mathcal{M}$ from which messages may be drawn, and to specify a function f on messages. Messages M and M' are drawn independently and at random from the message space $\mathcal{M}$. We consider two worlds. In the first, the adversary will attempt to compute $f(M)$ given an encryption of M. In the second, the adversary will attempt to compute $f(M)$ given an encryption of M' (that is, the adversary is given no information related to M). The scheme is secure if it succeeds about as often in the second game as the first, no matter what (reasonable) f and $\mathcal{M}$ the adversary selects.

To make our definition as general as possible, we will actually let the adversary choose, in sequence, message spaces $\mathcal{M}_1, \ldots, \mathcal{M}_q$. From each message space $\mathcal{M}_i$ draw the message M_i at random, and then let C_i be a random encryption of M_i. Adaptively querying, the adversary obtains the vector of ciphertexts $(C_1, \ldots, C_q)$. Now the adversary tries to find a function f such that it can do a good job at predicting $f(M_1, \ldots, M_q)$. Doing a good job means predicting this value significantly better than how well the adversary would predict it had it been given *no* information about $M_1, \ldots, M_q$: each C_i was not the encryption of M_i but the encryption of a random point M'_i from $\mathcal{M}_i$. The formal definition now follows.

Definition 4.6.1 [Semantic security] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, and let A be an algorithm that has access to an oracle. We consider the following experiments:

Experiment $\mathbf{Exp}^{\text{ss-cpa-1}}_{\mathcal{SE}}(A)$	Experiment $\mathbf{Exp}^{\text{ss-cpa-0}}_{\mathcal{SE}}(A)$
$K \xleftarrow{\$} \mathcal{K}$, $s \xleftarrow{\$} \varepsilon$	$K \xleftarrow{\$} \mathcal{K}$, $s \xleftarrow{\$} \varepsilon$
for $i \leftarrow 1$ to q do	for $i \leftarrow 1$ to q do
$(\mathcal{M}_i, s) \xleftarrow{\$} A(s)$	$(\mathcal{M}_i, s) \xleftarrow{\$} A(s)$
$M_i, M'_i \xleftarrow{\$} \mathcal{M}_i$	$M_i, M'_i \xleftarrow{\$} \mathcal{M}_i$
if $\lvert M_i\rvert \neq \lvert M'_i\rvert$ then $M_i \leftarrow M'_i \leftarrow \varepsilon$	if $\lvert M_i\rvert \neq \lvert M'_i\rvert$ then $M_i \leftarrow M'_i \leftarrow \varepsilon$
$C_i \xleftarrow{\$} \mathcal{E}_K(M_i)$; $s \leftarrow \langle s, C_i\rangle$	$C_i \xleftarrow{\$} \mathcal{E}_K(M'_i)$; $s \leftarrow \langle s, C_i\rangle$
$(f, Y) \xleftarrow{\$} A(s)$	$(f, Y) \xleftarrow{\$} A(s)$
return $f(M_1, \ldots, M_q) = Y$	return $f(M_1, \ldots, M_q) = Y$

The SEM-CPA advantage of A is defined as

$$\mathbf{Adv}^{\text{sem-cpa}}_{\mathcal{SE}}(A) \quad = \quad \Pr\left[\mathbf{Exp}^{\text{ss-cpa-1}}_{\mathcal{SE}}(A) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^{\text{ss-cpa-0}}_{\mathcal{SE}}(A) \Rightarrow 1\right] . \blacksquare$$

In the definition above, each experiment initializes its oracle by choosing a random key K. A total of q times, the adversary chooses a message space $\mathcal{M}_i$. The message space is specified by an always-halting probabilistic algorithm, written in some fixed programming language. The code for this algorithm is what the adversary actually outputs. Each time the message space is output, two random samples are drawn from this message space, M_i and M'_i. We expect that M_i and M'_i to have the same length, and if they don't we "erase" both strings. The encryption of one of these messages will be returned to the adversary. Which string gets encrypted depends on the experiment: M_i for experiment 1 and M'_i for experiment 0. By f we denote a deterministic function. It is described by an always-halting program and, as before, it actually the program for f that the adversary outputs. By Y we denote a string. The string s represents saved state that the adversary may wish to retain.

In speaking of the running time of A, we include, beyond the actual running time, the maximal time to draw two samples from each message space $\mathcal{M}$ that A outputs, and we include the maximal

time to compute $f(M_1, \ldots, M_q)$ over any vector of strings. In speaking of the length of A's queries we sum, over all the message spaces output by A, the maximal length of a string M output with nonzero probability by $\mathcal{M}$, and we sum also over the lengths of the encodings of each messages space, function f, and string Y output by A.

We emphasize that the above would seem to be an exceptionally strong notion of security. We have given the adversary the ability to choose the message spaces from which each message will be drawn. We have let the adversary choose the partial information about the messages that it finds convenient to predict. We have let the adversary be fully adaptive. We have built in the ability to perform a chosen-message attack (simply by producing an algorithm $\mathcal{M}$ that samples one and only one point). Despite all this, we now show that security in the indistinguishability sense implies semantic security.

Theorem 4.6.2 [IND-CPA $\Rightarrow$ SEM-CPA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let A be an adversary (for attacking the SEM-CPA security of $\mathcal{SE}$) that runs in time at most t and asks at most q queries, these queries totaling at most μ bits. Then there exists and adversary B (for attacking the IND-CPA security of $\mathcal{SE}$) that achieves advantage

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(B) \;\geq\; \mathbf{Adv}^{\text{sem-cpa}}_{\mathcal{SE}}(A)$$

and where B runs in time $t + O(\mu)$ and asks at most q queries, these queries totaling μ bits.

Proof: The adversary B, which has oracle g, is constructed as follows.

algorithm B^g
  $s \xleftarrow{\$} \varepsilon$
  for $i \leftarrow 1$ to q do
    $(\mathcal{M}_i, s) \xleftarrow{\$} A(s)$
    $M_i, M'_i \xleftarrow{\$} \mathcal{M}_i$
    if $|M_i| \neq |M'_i|$ then $M_i \leftarrow M'_i \leftarrow \varepsilon$
    $C_i \leftarrow g(M'_i, M_i)$, $s \leftarrow \langle s, C_i \rangle$
  $(f, Y) \xleftarrow{\$} A(s)$
  if $f(M_1, \ldots, M_q) = Y$ then return 1 else return 0

Suppose first that g is instantiated by a right encryption oracle—an oracle that returns $C \xleftarrow{\$} \mathcal{E}_K(M)$ in response to a query (M', M). Then the algorithm above coincides with experiment $\mathbf{Exp}^{\text{ss-cpa-1}}_{\mathcal{SE}}(A)$. Similarly, if g is instantiated by a left encryption oracle—the oracle it returns $C \xleftarrow{\$} \mathcal{E}_K(M')$ in response to a query (M', M)—then the algorithm above coincides with experiment $\mathbf{Exp}^{\text{ss-cpa-0}}_{\mathcal{SE}}(A)$. It follows that $\mathbf{Adv}^{\text{sem-cpa}}_{\mathcal{SE}}(B) = \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A)$. To complete the theorem, note that B's running time is A's running time plus $O(\mu)$ and B asks a total of q queries, these having total length at most the total length of A's queries, under our convention. ▮

4.7 Security of CTR modes

Recall that the CTR (counter) mode of operation of a family of functions comes in two variants: the randomized (stateless) version CTRC of Scheme 4.2.5, and the counter-based (stateful) mechanism CTR$ of Scheme 4.2.6. Both modes achieve indistinguishability under a chosen-plaintext attack, but, interestingly, the quantitative security is a little different. The difference springs from the fact

that CTRC achieves *perfect* indistinguishability if one uses a random function in the role of the underlying F_K—but CTR$ would not achieve perfect indistinguishability even then, because of the possibility that collisions would produce "overlaps" in the pseudo-one-time pad.

We will state the main theorems about the schemes, discuss them, and then prove them. For the counter version we have:

Theorem 4.7.1 [Security of CTRC mode] Let $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a family of functions and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding CTRC symmetric encryption scheme as described in Scheme 4.2.6. Let A be an adversary (for attacking the IND-CPA security of $\mathcal{SE}$) that runs in time at most t and asks at most q queries, these totaling at most σ n-bit blocks. Then there exists an adversary B (attacking the PRF security of F) such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;\leq\; 2 \cdot \mathbf{Adv}^{\text{prf}}_{F}(B)\,.$$

Furthermore B runs in time at most $t' = t + O(q + n\sigma)$ and asks at most $q' = \sigma$ oracle queries. ▌

Theorem 4.7.2 [Security of CTR$ mode] Let $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding CTR$ symmetric encryption scheme as described in Scheme 4.2.5. Let A be an adversary (for attacking the IND-CPA security of $\mathcal{SE}$) that runs in time at most t and asks at most q queries, these totaling at most σ n-bit blocks. Then there exists an adversary B (attacking the PRF security of F) such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;\leq\; \mathbf{Adv}^{\text{prf}}_{F}(B) + \frac{0.5\,\sigma^2}{2^n}\,.$$

Furthermore B runs in time at most $t' = t + O(q + n\sigma)$ and asks at most $q' = \sigma$ oracle queries. ▌

The above theorems exemplify the kinds of results that the provable-security approach is about. Namely, we are able to provide provable guarantees of security of some higher level cryptographic construct (in this case, a symmetric encryption scheme) based on the assumption that some building block (in this case an underlying block) is secure. The above results are the first example of the "punch-line" we have been building towards. So it is worth pausing at this point and trying to make sure we really understand what these theorems are saying and what are their implications.

If we want to entrust our data to some encryption mechanism, we want to know that this encryption mechanism really provides privacy. If it is ill-designed, it may not. We saw this happen with ECB. Even if we used a secure blockcipher, the flaws of ECB mode make it an insecure encryption scheme.

Flaws are not apparent in CTR at first glance. But maybe they exist. It is very hard to see how one can be convinced they do *not* exist, when one cannot possible exhaust the space of all possible attacks that could be tried. Yet this is exactly the difficulty that the above theorems circumvent. They are saying that CTR mode *does not have design flaws*. They are saying that as long as you use a good blockcipher, you are *assured* that nobody will break your encryption scheme. One cannot ask for more, since if one does not use a good blockcipher, there is no reason to expect security of your encryption scheme anyway. We are thus getting a conviction that *all attacks fail* even though we do not even know exactly how these attacks might operate. That is the power of the approach.

Now, one might appreciate that the ability to make such a powerful statement takes work. It is for this that we have put so much work and time into developing the definitions: the formal notions of security that make such results meaningful. For readers who have less experience with definitions, it is worth knowing, at least, that the effort is worth it. It takes time and work to understand the notions, but the payoffs are big: you get significant guarantees of security.

How, exactly, are the theorems saying this? The above discussion has pushed under the rug the quantitative aspect that is an important part of the results. It may help to look at a concrete example.

Example 4.7.3 Let us suppose that F is the blockcipher AES, so that $n = 128$. Suppose I want to encrypt $q = 2^{30}$ messages, each being one kilobyte (2^{13} bits) long. I am thus encrypting a total of 2^{43} bits, which is to say $\sigma = 2^{36}$ blocks. (This is about one terabyte). Can I do this securely using CTR\$? Let A be an adversary attacking the privacy of my encryption. Theorem 4.7.2 says that there exists a B satisfying the stated conditions. How large can $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{AES}}(B)$ be? It makes $q = 2^{36}$ queries, and it is consistent with our state of knowledge of the security of AES to assume that such an adversary cannot do better than mount a birthday attack, meaning its advantage is no more than $q^2/2^{128}$. Under such an assumption, the theorem tells us that $\mathbf{Adv}^{\text{rnd-cpa}}_{\mathcal{SE}}(A)$ is at most $\sigma^2/2^{128} + 0.5\,\sigma^2/2^{128} = 1.5\,2^{72}/2^{128} \leq 1/2^{55}$. This is a very small number indeed, saying that our encryption is secure, at least under the assumption that the best attack on the PRF security of AES is a birthday attack. Note however that if we encrypt 2^{64} blocks of data, our provable-security bound becomes meaningless. ▮

The example illustrates how to use the theorems to figure out how much security you will get from the CTR encryption scheme in a given application.

Note that as per the above theorems, encrypting more than $\sigma = 2^{n/2}$ blocks of data with CTR\$ is not secure regardless of the quality of F as a PRF. On the other hand, with CTRC, it might be secure, as long as F can withstand σ queries. This is an interesting and possibly useful distinction. Yet, in the setting in which such modes are usually employed, the distinction all but vanishes. For usually F is a blockcipher and in that case, we know from the birthday attack that the prf-advantage of B may itself be as large as $\Theta(\sigma^2/2^n)$, and thus, again, encrypting more than $\sigma = 2^{n/2}$ blocks of data is not secure. However, we might be able to find or build function families F that are not families of permutations and preserve PRF security against adversaries making more than $2^{n/2}$ queries.

Proof of Theorem 4.7.1: We consider the games G_0, G_1, G_2 of Fig. 4.8. We now build prf-adversary B so that

$$\Pr\left[G_0^A \Rightarrow 1\right] - \Pr\left[G_1^A \Rightarrow 1\right] = \mathbf{Adv}^{\mathrm{prf}}_F(B). \tag{4.1}$$

Additionally, B's resource usage (running time and query count) is as claimed in the theorem.

Remember that B has access to an oracle $\mathbf{Fn}$: $\{0,1\}^n \to \{0,1\}^n$. B runs A, answering any queries that A makes to its **LR** oracle via the subroutine LRSIM. This subroutine uses B's own oracle $\mathbf{Fn}$.

Adversary B $b \xleftarrow{\$} \{0,1\}$; $ctr \leftarrow 0$; $b' \xleftarrow{\$} A^{\mathrm{LRSIM}}$ If $(b = b')$ then return 1 Else return 0	**subroutine** LRSIM(M_0, M_1) $C[0] \leftarrow ctr$; $m \leftarrow \|M_b\|_n$ for $i = 1, \ldots, m$ do $T[ctr + i] \leftarrow \mathbf{Fn}(ctr + i)$ $C[i] \leftarrow T[ctr + i] \oplus M_b[i]$ $ctr \leftarrow ctr + m$ return C

If $\mathbf{Fn} = F_K$ then B is providing A the environment of game G_0 so

$$\Pr\left[\mathrm{Real}_F^B \Rightarrow 1\right] = \Pr\left[G_0^A \Rightarrow 1\right]$$

Game G_0

procedure Initialize
$K \xleftarrow{\$} \{0,1\}^k; b \xleftarrow{\$} \{0,1\}$
$ctr \leftarrow 0$

procedure LR(M_0, M_1)
$C[0] \leftarrow ctr; m \leftarrow \|M_b\|_n$
for $i = 1, ..., m$ do
  $T[ctr + i] \leftarrow F_K(ctr + i)$
  $C[i] \leftarrow T[ctr + i] \oplus M_b[i]$
$ctr \leftarrow ctr + m$
return C

procedure Finalize(b')
return $(b = b')$

Game G_1

procedure Initialize
$b \xleftarrow{\$} \{0,1\}; ctr \leftarrow 0$

procedure LR(M_0, M_1)
$C[0] \leftarrow ctr; m \leftarrow \|M_b\|_n$
for $i = 1, ..., m$ do
  $T[ctr + i] \xleftarrow{\$} \{0,1\}^n$
  $C[i] \leftarrow T[ctr + i] \oplus M_b[i]$
$ctr \leftarrow ctr + m$
return C

procedure Finalize(b')
return $(b = b')$

Game G_2

procedure Initialize
$b \xleftarrow{\$} \{0,1\}; ctr \leftarrow 0$

procedure LR(M_0, M_1)
$C[0] \leftarrow ctr; m \leftarrow \|M_0\|_n$
for $i = 1, ..., m$ do
  $C[i] \xleftarrow{\$} \{0,1\}^n$
return C

procedure Finalize(b')
return $(b = b')$

Figure 4.8: Games used to prove Theorem 4.7.1.

If **Fn** is random then B is providing A the environment of game G_1 so

$$\Pr\left[\mathrm{Rand}_F^B \Rightarrow 1\right] = \Pr G_1^A \Rightarrow 1$$

Thus

$$\begin{aligned}\mathbf{Adv}_F^{\mathrm{prf}}(B) &= \Pr\left[\mathrm{Real}_F^B \Rightarrow 1\right] - \Pr\left[\mathrm{Rand}_F^B \Rightarrow 1\right] \\ &= \Pr\left[G_0^A \Rightarrow 1\right] - \Pr\left[G_1^A \Rightarrow 1\right]\end{aligned}$$

which proves equation (4.1). Obviously,

$$\Pr[G_0^A \Rightarrow 1] = \Pr[G_1^A \Rightarrow 1] + \Pr[G_0^A \Rightarrow 1] - \Pr[G_1^A \Rightarrow 1]$$

So, using equation (4.1) we have

$$\begin{aligned}\mathbf{Adv}_{\mathcal{SE}}^{\text{ind-cpa}}(A) &= 2 \cdot \Pr[G_0^A \Rightarrow 1] - 1 \\ &= 2 \cdot \Big(\Pr[G_1^A \Rightarrow 1] + \mathbf{Adv}_F^{\mathrm{prf}}(B) \Big) - 1 \\ &= 2 \cdot \mathbf{Adv}_F^{\mathrm{prf}}(B) + 2 \Pr[G_1^A \Rightarrow 1] - 1\end{aligned}$$

We now claim that

$$\Pr[G_1^A \Rightarrow 1] = \frac{1}{2}.$$

Combining this with the above we have

Game G_0

procedure Initialize
$K \xleftarrow{\$} \{0,1\}^k$; $b \xleftarrow{\$} \{0,1\}$; $S \leftarrow \emptyset$

procedure LR(M_0, M_1)
$m \leftarrow \|M_b\|_n$; $C[0] \xleftarrow{\$} \{0,1\}^n$
for $i = 1, ..., m$ do
  $P \leftarrow C[i-1] \oplus M_b[i]$
  if $P \in S$ then
    $\mathsf{T}[P] \leftarrow E_K(P)$
  $C[i] \leftarrow T[P_{j,i}]$
$S \leftarrow S \cup \{P\}$
return C

procedure Finalize(b')
return $(b = b')$

Game G_1

procedure Initialize
$b \xleftarrow{\$} \{0,1\}$; $S \leftarrow \emptyset$

procedure LR(M_0, M_1)
$m \leftarrow \|M_b\|_n$; $C[0] \xleftarrow{\$} \{0,1\}^n$
for $i = 1, ..., n$ do
  $P \leftarrow C[i-1] \oplus M_b[i]$
  If $P \notin S$ then $\mathsf{T}[P] \xleftarrow{\$} \{0,1\}^n$
  $C[i] \leftarrow \mathsf{T}[P]$
$S \leftarrow S \cup \{P\}$
return C

procedure Finalize(b')
return $(b = b')$

Figure 4.9: Games used to prove Theorem 4.8.1.

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;\leq\; 2 \cdot \mathbf{Adv}^{\text{prf}}_{F}(B).$$

The above claim is justified by observing that

$$\Pr\left[G_1^A \Rightarrow 1\right] = \Pr\left[G_2^A \Rightarrow 1\right] \text{ and } \Pr\left[G_2^A \Rightarrow 1\right] = \tfrac{1}{2}.$$

∎

4.8 Security of CBC with a random IV

In this section we show that CBC encryption using a random IV is IND-CPA secure as long as E is a blockcipher that is a secure PRF or PRP. Namely we show:

Theorem 4.8.1 [Security of CBC$ mode] Let E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding CBC$ symmetric encryption scheme as described in Scheme 4.2.5. Let A be an adversary (for attacking the IND-CPA security of $\mathcal{SE}$) that runs in time at most t and asks at most q queries, these totaling at most σ n-bit blocks. Then there exists an adversary B (attacking the PRF security of E) such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;\leq\; 2 \cdot \mathbf{Adv}^{\text{prf}}_{E}(B) + \frac{\sigma^2}{2^n}\,.$$

Furthermore B runs in time at most $t' = t + O(q + n\sigma)$ and asks at most $q' = \sigma$ oracle queries. ∎

Proof: The proof uses games G_0, G_1 of Fig. 4.9 and games G_2, G_3, G_4 of Fig. 4.10. Then we have

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) = 2 \cdot \Pr\left[G_0^A \Rightarrow \mathsf{true}\right] - 1.$$

But

$$\Pr\left[G_0^A \Rightarrow \mathsf{true}\right] = \Pr\left[G_1^A \Rightarrow \mathsf{true}\right] + \left(\Pr\left[G_0^A \Rightarrow \mathsf{true}\right] - \Pr\left[G_1^A \Rightarrow \mathsf{true}\right]\right)$$

We now build prf-adversary B so that

$$\Pr\left[G_0^A \Rightarrow \mathsf{true}\right] - \Pr\left[G_1^A \Rightarrow \mathsf{true}\right] \leq \mathbf{Adv}_E^{\mathrm{prf}}(B).$$

Additionally, B's resource usage (query count and running time) is as claimed in the theorem. The prf-adversary B works as follows:

Adversary B
$b \xleftarrow{\$} \{0,1\}$; $S \leftarrow \emptyset$
$b' \xleftarrow{\$} A^{\mathbf{LR}}$
if $(b = b')$ then return 1
else return 0

subroutine LRSIM(M_0, M_1)
$m \leftarrow \|M_b\|_n$; $C[0] \xleftarrow{\$} \{0,1\}^n$
for $i = 1, ..., m$ do
  $P \leftarrow C[i-1] \oplus M_b[i]$
  if $P \notin S$ then $T[P] \leftarrow \mathbf{Fn}(P)$
  $C[i] \leftarrow T[P]$
$S \leftarrow S \cup \{P\}$
return C

It is easy to see that

$$\begin{aligned}\Pr\left[\mathrm{Real}_E^B \Rightarrow 1\right] &= \Pr\left[G_0^A \Rightarrow \mathsf{true}\right] \\ \Pr\left[\mathrm{Rand}_E^B \Rightarrow 1\right] &= \Pr\left[G_1^A \Rightarrow \mathsf{true}\right]\end{aligned}$$

So we have

$$\Pr\left[G_0^A \Rightarrow \mathsf{true}\right] - \Pr\left[G_1^A \Rightarrow \mathsf{true}\right] = \Pr\left[\mathrm{Real}_E^B \Rightarrow 1\right] - \Pr\left[\mathrm{Rand}_E^B \Rightarrow 1\right] \leq \mathbf{Adv}_E^{\mathrm{prf}}(B).$$

We are going to prove that $\Pr\left[G_1^A \Rightarrow \mathsf{true}\right] \leq \frac{1}{2} + \sigma^2 \cdot 2^{-n-1}$.

Assuming this for now, we will have

$$\begin{aligned}\mathbf{Adv}_{\mathcal{SE}}^{\text{ind-cpa}}(A) &\leq 2 \cdot \left(\frac{1}{2} + \frac{\sigma^2}{2^{n+1}}\right) - 1 + 2\mathbf{Adv}_E^{\mathrm{prf}}(B) \\ &= \frac{\sigma^2}{2^n} + 2\mathbf{Adv}_E^{\mathrm{prf}}(B)\end{aligned}$$

In the following, we use games G_2, G_3, G_4 of Fig. 4.10.

$$\begin{aligned}\Pr[G_1^A \Rightarrow \mathsf{true}] &= \Pr[G_2^A \Rightarrow \mathsf{true}] \\ &= \Pr[G_3^A \Rightarrow \mathsf{true}] + (\Pr[G_2^A \Rightarrow \mathsf{true}] - \Pr[G_3^A \Rightarrow \mathsf{true}])\end{aligned}$$

We will show that

- $\Pr[G_3^A \Rightarrow \mathsf{true}] = \frac{1}{2}$

Game $\boxed{G_2}$, G_3

procedure Initialize
$b \xleftarrow{\$} \{0,1\}$; $S \leftarrow \emptyset$

procedure LR(M_0, M_1)
$m \leftarrow \|M_b\|_n$; $C[0] \xleftarrow{\$} \{0,1\}^n$
for $i = 1, ..., n$ do
  $P \leftarrow C[i-1] \oplus M_b[i]$
  $C[i] \xleftarrow{\$} \{0,1\}^n$
  If $P \in S$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{C[i] \leftarrow \mathsf{T}[P]}$
  $\mathsf{T}[P] \leftarrow C[i]$
$S \leftarrow S \cup \{P\}$
return C

procedure Finalize(b')
return $(b = b')$

Game G_4

procedure Initialize
$j \leftarrow 0$; $S \leftarrow \emptyset$

procedure LR(M_0, M_1)
$m \leftarrow \|M_0\|_n$
for $i = 1, ..., m$ do
  $P \xleftarrow{\$} \{0,1\}^n$
  $C[i] \xleftarrow{\$} \{0,1\}^n$
  If $P \in S$ then $\mathsf{bad} \leftarrow \mathsf{true}$
$S \leftarrow S \cup \{P\}$

procedure Finalize(b')
return $(b = b')$

Figure 4.10: Games used to prove Theorem 4.8.1.

- $\Pr[G_2^A \Rightarrow \mathsf{true}] - \Pr[G_3^A \Rightarrow \mathsf{true}] \leq \frac{\sigma^2}{2^{n+1}}$

Ciphertext C in G_3 is always random, independently of b, so

$$\Pr\left[G_3^A \Rightarrow \mathsf{true}\right] = \frac{1}{2}.$$

G_2 and G_3 are identical-until-bad, so Fundamental Lemma implies

$$\Pr\left[G_2^A \Rightarrow \mathsf{true}\right] - \Pr\left[G_3^A \Rightarrow \mathsf{true}\right] \leq \Pr\left[G_3^A \text{ sets } \mathsf{bad}\right].$$

$$\Pr\left[G_3^A \text{ sets } \mathsf{bad}\right] = \Pr\left[G_4^A \text{ sets } \mathsf{bad}\right]$$

The ℓ-th time the if-statement is executed, it has probability

$$\frac{\ell - 1}{2^n}$$

of setting bad. Thus

$$\begin{aligned}\Pr\left[G_4^A \text{ sets } \mathsf{bad}\right] &\leq \sum_{\ell=1}^{\sigma} \frac{\ell - 1}{2^n} \\ &= \frac{\sigma(\sigma-1)}{2^{n+1}} \\ &\leq \frac{\sigma^2}{2^{n+1}}\,.\end{aligned}$$

The theorem follows. ▮

We have so far shown that the IND-CPA advantage of CBC$ falls off by an amount that is at most quadratic in the number of blocks, σ, asked by the adversary. We can also give a matching attack, showing that there actually *is* an adversary that obtains advantage of about $\sigma^2/2^n$. This tells us that our security result is *tight*—there is no possibility of making the bound significantly better. It means that we have arrived at reasonably precise understanding of the security of CBC encryption with a random IV.

Proposition 4.8.2 Let $n \geq 1$, let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a function family, and let $\sigma \in [0 \,..\, \sqrt{2}\, 2^{n/2} - 1]$. Then there is an adversary A that asks a single query, the query consisting of σ blocks, runs in time $O(n\sigma \lg(\sigma))$, and achieves advantage $\mathbf{Adv}^{\text{ind-cpa}}_{\text{CBC}[E]}(A) \geq 0.15\,\sigma^2/2^n$ and

Proof: The adversary A sets $L \leftarrow 0^{n\sigma}$, chooses $R \xleftarrow{\$} \{0,1\}^{n\sigma}$, and asks its oracle the query (L, R), receiving in response a ciphertext C that it partitions into $\sigma + 1$ n-bit blocks, $C_0 C_1 \ldots C_\sigma$. If there is an $i, I \in [0 \,..\, \sigma]$ such that $i < I$ and $C_i = C_I$ then A selects the lexicographically first such (i, I) and answers 1 (for "right oracle") if $C_{i+1} \neq C_{I+1}$. (In this case the adversary has found a "proof" that the oracle is a right oracle.) In all other cases the adversary outputs 0 (for "left oracle").

The adversary described asks asks a single query of σ blocks and, using standard data structure techniques, it runs in time $O(n\sigma \lg(\sigma))$. It remains to calculate the adversary's advantage, $\mathbf{Adv}^{\text{prp}}_E(A) = \Pr[A^{\text{Right}(\cdot,\cdot)} \Rightarrow 1] - \Pr[A^{\text{Left}(\cdot,\cdot)} \Rightarrow 1]$. The second summand is zero since when A is given a left encryption-oracle that oracle is encrypting the zero-string and any time $C_i = C_I$ we *must* have that $C_{i+1} = C_{I+1}$ as well. Thus

$$\begin{aligned}
\mathbf{Adv}^{\text{prp}}_E(A) &= \Pr[A^{\text{Right}(\cdot,\cdot)} \Rightarrow 1] \\
&= \Pr[R \xleftarrow{\$} \{0,1\}^{n\sigma};\ K \xleftarrow{\$} \mathcal{K};\ \text{IV} \xleftarrow{\$} \{0,1\}^n;\ C \xleftarrow{\$} \text{CBC}^{\text{IV}}_K(R) : \\
&\qquad \exists i < I \text{ s.t. } C_i = C_I \text{ and } C_{i+1} \neq C_{I+1} \text{ on the first such } (i, I)]
\end{aligned}$$

By the structure of CBC mode with a random IV it is easy to see that that when you encrypt a random string $R \in \{0,1\}^{n\sigma}$ you get a random string $C \in \{0,1\}^{n(\sigma+1)}$. To see this, note that to make block C_i, for $i \geq 1$, you xor the random block R_i with C_i and apply the blockcipher. The random block R_i is independent of C_i—it wasn't even consulted in making C_i—and it is independent of all of $C_0, \ldots, C_{i-1}$, too. The image of a uniformly selected value under a permutation is uniform. The very first block of ciphertext, C_0, is uniform. This makes the entire string $C_0 C_1 \cdots C_\sigma$ uniform. So the probability in question is

$$\begin{aligned}
\mathbf{Adv}^{\text{prp}}_E(A) &= \Pr[C \xleftarrow{\$} \{0,1\}^{n(\sigma+1)} : \\
&\qquad \exists i < I \text{ s.t. } C_i = C_I \text{ and } C_{i+1} \neq C_{I+1} \text{ on the first such } (i, I)]
\end{aligned}$$

Now the birthday bound (Appendix **??**, Theorem **??**) tells us that the probability there will be an $i < I$ such that $C_i = C_I$ is at least $C(2^n, \sigma + 1) \geq 0.3\sigma^2/2^n$. When there is such an i, I and we fix the lexicographically first such i, I, note that C_{I+1} is still uniform and independent of C_{i+1}. Independence is assured because C_{I+1} is obtained as $E_K(R_{I+1} \oplus C_I)$ for a permutation E_K and a uniform random value R_{I+1} that is independent of C_I and C_{i+1}. Because of this probabilistic independence, the probability of the conjunct is just the product of the probabilities and we have that

$$\mathbf{Adv}^{\text{prp}}_E(A) \;\geq\; 0.3\,\sigma^2/2^n \cdot (1 - 2^{-n}) \;\geq\; 0.15\,\sigma^2/2^n$$

completing the proof. ▮

4.9 Indistinguishability under chosen-ciphertext attack

So far we have considered privacy under chosen-plaintext attack. Sometimes we want to consider privacy when the adversary is capable of mounting a stronger type of attack, namely a chosen-ciphertext attack. In this type of attack, an adversary has access to a decryption oracle. It can feed this oracle a ciphertext and get back the corresponding plaintext.

How might such a situation arise? One situation one could imagine is that an adversary at some point gains temporary access to the equipment performing decryption. It can feed the equipment ciphertexts and see what plaintexts emerge. (We assume it cannot directly extract the key from the equipment, however.)

If an adversary has access to a decryption oracle, security at first seems moot, since after all it can decrypt anything it wants. To create a meaningful notion of security, we put a restriction on the use of the decryption oracle. To see what this is, let us look closer at the formalization. As in the case of chosen-plaintext attacks, we consider two worlds. In the left world, *all* message pairs sent to the oracle are answered by the oracle encrypting the left message in the pair, while in the right world, all message pairs are answered by the oracle encrypting the right message in the pair. The choice of which does not flip-flop from oracle query to oracle query.

The adversary's goal is the same as in the case of chosen-plaintext attacks: it wants to figure out which world it is in. There is one easy way to do this. Namely, query the lr-encryption oracle on two distinct, equal length messages M_0, M_1 to get back a ciphertext C, and now call the decryption oracle on C. If the message returned by the decryption oracle is M_0 then the adversary is in world 0, and if the message returned by the decryption oracle is M_1 then the adversary is in world 1. The restriction we impose is simply that this call to the decryption oracle is not allowed. More generally, call a query C to the decryption oracle *illegitimate* if C was previously returned by the lr-encryption oracle; otherwise a query is *legitimate.* We insist that only legitimate queries are allowed. In the formalization below, the experiment simply returns 0 if the adversary makes an illegitimate query. (We clarify that a query C is legitimate if C is returned by the lr-encryption oracle *after* C was queried to the decryption oracle.)

This restriction still leaves the adversary with a lot of power. Typically, a successful chosen-ciphertext attack proceeds by taking a ciphertext C returned by the lr-encryption oracle, modifying it into a related ciphertext C', and querying the decryption oracle with C'. The attacker seeks to create C' in such a way that its decryption tells the attacker what the underlying message M was. We will see this illustrated in Section 4.10 below.

The model we are considering here might seem quite artificial. If an adversary has access to a decryption oracle, how can we prevent it from calling the decryption oracle on certain messages? The restriction might arise due to the adversary's having access to the decryption equipment for a limited period of time. We imagine that after it has lost access to the decryption equipment, it sees some ciphertexts, and we are capturing the security of these ciphertexts in the face of previous access to the decryption oracle. Further motivation for the model will emerge when we see how encryption schemes are used in protocols. We will see that when an encryption scheme is used in many authenticated key-exchange protocols the adversary effectively has the ability to mount chosen-ciphertext attacks of the type we are discussing. For now let us just provide the definition and exercise it.

Definition 4.9.1 Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, let A be an algorithm that has access to two oracles, and let b be a bit. We consider the following experiment:

Game $\text{Left}_{\mathcal{SE}}$	Game $\text{Right}_{\mathcal{SE}}$
procedure Initialize	**procedure Initialize**
$K \xleftarrow{\$} \mathcal{K}$	$K \xleftarrow{\$} \mathcal{K}$
procedure LR(M_0, M_1)	**procedure LR**(M_0, M_1)
Return $C \xleftarrow{\$} \mathcal{E}_K(M_0)$	Return $C \xleftarrow{\$} \mathcal{E}_K(M_1)$
procedure Dec(C)	**procedure Dec**(C)
return $M \leftarrow \mathcal{D}_K(C)$	return $M \leftarrow \mathcal{D}_K(C)$

The *IND-CCA advantage* of A is defined as

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}}(A) \quad = \quad \Pr\left[\text{Right}^A_{\mathcal{SE}} \Rightarrow 1\right] - \Pr\left[\text{Left}^A_{\mathcal{SE}} \Rightarrow 1\right] \ .$$

The conventions with regard to resource measures are the same as those used in the case of chosen-plaintext attacks. In particular, the length of a query M_0, M_1 to the lr-encryption oracle is defined as the length of M_0.

We consider an encryption scheme to be "secure against chosen-ciphertext attack" if a "reasonable" adversary cannot obtain "significant" advantage in distinguishing the cases $b = 0$ and $b = 1$ given access to the oracles, where reasonable reflects its resource usage. The technical notion is called indistinguishability under chosen-ciphertext attack, denoted IND-CCA.

4.10 Example chosen-ciphertext attacks

Chosen-ciphertext attacks are powerful enough to break all the standard modes of operation, even those like CTR and CBC that are secure against chosen-plaintext attack. The one-time pad scheme is also vulnerable to a chosen-ciphertext attack: our notion of perfect security only took into account chosen-plaintext attacks. Let us now illustrate a few chosen-ciphertext attacks.

4.10.1 Attacks on the CTR schemes

Let $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a family of functions and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the associated CTR\$ symmetric encryption scheme as described in Scheme 4.2.5. The weakness of the scheme that makes it susceptible to a chosen-ciphertext attack is the following. Say $\langle r, C\rangle$ is a ciphertext of some n-bit message M, and we flip bit i of C, resulting in a new ciphertext $\langle r, C'\rangle$. Let M' be the message obtained by decrypting the new ciphertext. Then M' equals M with the i-th bit flipped. (You should check that you understand why.) Thus, by making a decryption oracle query of $\langle r, C'\rangle$ one can learn M' and thus M. In the following, we show how this idea can be applied to break the scheme in our model by figuring out in which world an adversary has been placed.

Proposition 4.10.1 Let $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a family of functions and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding CTR\$ symmetric encryption scheme as described in Scheme 4.2.5. Then

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}}(t, 1, n, 1, 2n) \quad = \quad 1$$

for $t = O(n)$ plus the time for one application of F.

The advantage of this adversary is 1 even though it uses hardly any resources: just one query to each oracle. That is clearly an indication that the scheme is insecure.

Proof of Proposition 4.10.1: We will present an adversary algorithm A, having time-complexity t, making 1 query to its lr-encryption oracle, this query being of length n, making 1 query to its decryption oracle, this query being of length $2n$, and having

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}}(A) \;=\; 1\ .$$

The Proposition follows.

Remember that the lr-encryption oracle $\mathcal{E}_K(\mathbf{LR}(\cdot,\cdot,b))$ takes input a pair of messages, and returns an encryption of either the left or the right message in the pair, depending on the value of b. The goal of A is to determine the value of b. Our adversary works like this:

Adversary $A^{\mathbf{LR}(\cdot,\cdot),\mathbf{Dec}(\cdot)}$
$M_0 \leftarrow 0^n$; $M_1 \leftarrow 1^n$
$\langle r, C\rangle \leftarrow \mathbf{LR}(M_0, M_1)$
$C' \leftarrow C \oplus 1^n$
$M \leftarrow \mathbf{Dec}(\langle r, C'\rangle)$
If $M = M_0$ then return 1 else return 0

The adversary's single lr-encryption oracle query is the pair of distinct messages M_0, M_1, each one block long. It is returned a ciphertext $\langle r, C\rangle$. It flips the bits of C to get C' and then feeds the ciphertext $\langle r, C\rangle$ to the decryption oracle. It bets on world 1 if it gets back M_0, and otherwise on world 0. Notice that $\langle r, C'\rangle \neq \langle r, C\rangle$, so the decryption query is legitimate. Now, we claim that

$$\begin{aligned}\Pr\left[\text{Right}^A_{\mathcal{SE}}\Rightarrow 1\right] &= 1\\ \Pr\left[\text{Left}^A_{\mathcal{SE}}\Rightarrow 1\right] &= 0\ .\end{aligned}$$

Hence $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) = 1 - 0 = 1$. And A achieved this advantage by making just one lr-encryption oracle query, whose length, which as per our conventions is just the length of M_0, is n bits, and just one decryption oracle query, whose length is $2n$ bits (assuming an encoding of $\langle r, X\rangle$ as $n+|X|$-bits). So $\mathbf{Adv}^{\text{pr-cpa}}_{\mathcal{SE}}(t, 1, n, 1, 2n) = 1$.

Why are the two equations claimed above true? You have to return to the definitions of the quantities in question, as well as the description of the scheme itself, and walk it through. In world 1, meaning $b = 1$, let $\langle r, C\rangle$ denote the ciphertext returned by the lr-encryption oracle. Then

$$C \;=\; F_K(r+1) \oplus M_1 \;=\; F_K(r+1) \oplus 1^n\ .$$

Now notice that

$$\begin{aligned}M &= \mathcal{D}_K(\langle r, C'\rangle)\\ &= F_K(r+1) \oplus C'\\ &= F_K(r+1) \oplus C \oplus 1^n\\ &= F_K(r+1) \oplus (F_K(r+1) \oplus 1^n) \oplus 1^n\\ &= 0^n\\ &= M_0\ .\end{aligned}$$

Thus, the decryption oracle will return M_0, and A will return 1. In world 0, meaning $b = 0$, let $\langle r, C[1]\rangle$ denote the ciphertext returned by the lr-encryption oracle. Then

$$C \;=\; F_K(r+1) \oplus M_0 \;=\; F_K(r+1) \oplus 0^n \;.$$

Now notice that

$$\begin{aligned} M &= \mathcal{D}_K(\langle r, C'\rangle) \\ &= F_K(r+1) \oplus C' \\ &= F_K(r+1) \oplus C \oplus 1^n \\ &= F_K(r+1) \oplus (F_K(r+1) \oplus 0^n) \oplus 1^n \\ &= 1^n \\ &= M_1 \;. \end{aligned}$$

Thus, the decryption oracle will return M_1, and A will return 0, meaning will return 1 with probability zero. ▮

An attack on CTRC (cf. Scheme 4.2.6) is similar, and is left to the reader.

4.10.2 Attack on CBC$

Let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the associated CBC$ symmetric encryption scheme as described in Scheme 4.2.3. The weakness of the scheme that makes it susceptible to a chosen-ciphertext attack is the following. Say $\langle \mathrm{IV}, C[1]\rangle$ is a ciphertext of some n-bit message M, and we flip bit i of the IV, resulting in a new ciphertext $\langle \mathrm{IV}', C[1]\rangle$. Let M' be the message obtained by decrypting the new ciphertext. Then M' equals M with the i-th bit flipped. (You should check that you understand why by looking at Scheme 4.2.3.) Thus, by making a decryption oracle query of $\langle \mathrm{IV}', C[1]\rangle$ one can learn M' and thus M. In the following, we show how this idea can be applied to break the scheme in our model by figuring out in which world an adversary has been placed.

Proposition 4.10.2 Let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding CBC$ encryption scheme as described in Scheme 4.2.3. Then

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}}(t, 1, n, 1, 2n) \;=\; 1$$

for $t = O(n)$ plus the time for one application of F.

The advantage of this adversary is 1 even though it uses hardly any resources: just one query to each oracle. That is clearly an indication that the scheme is insecure.

Proof of Proposition 4.10.2: We will present an adversary A, having time-complexity t, making 1 query to its lr-encryption oracle, this query being of length n, making 1 query to its decryption oracle, this query being of length $2n$, and having

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}}(A) \;=\; 1 \;.$$

The proposition follows.

Remember that the lr-encryption oracle $\mathcal{E}_K(\mathbf{LR}(\cdot,\cdot,b))$ takes input a pair of messages, and returns an encryption of either the left or the right message in the pair. The goal of A is to determine which. Our adversary works like this:

Adversary $A^{\mathbf{LR}(\cdot,\cdot),\mathbf{Dec}(\cdot)}$
$M_0 \leftarrow 0^n$; $M_1 \leftarrow 1^n$
$\langle \mathrm{IV}, C[1]\rangle \leftarrow \mathbf{LR}(M_0, M_1)$
$\mathrm{IV}' \leftarrow \mathrm{IV} \oplus 1^n$
$M \leftarrow \mathbf{Dec}(\langle \mathrm{IV}', C[1]\rangle)$
If $M = M_0$ then return 1 else return 0

The adversary's single lr-encryption oracle query is the pair of distinct messages M_0, M_1, each one block long. It is returned a ciphertext $\langle \mathrm{IV}, C[1]\rangle$. It flips the bits of the IV to get a new IV, IV′, and then feeds the ciphertext $\langle \mathrm{IV}', C[1]\rangle$ to the decryption oracle. It bets on world 1 if it gets back M_0, and otherwise on world 0. It is important that $\langle \mathrm{IV}', C[1]\rangle \neq \langle \mathrm{IV}, C[1]\rangle$ so the decryption oracle query is legitimate. Now, we claim that

$$\begin{aligned} \Pr\left[\mathrm{Right}^A_{\mathcal{SE}} \Rightarrow 1\right] &= 1 \\ \Pr\left[\mathrm{Left}^A_{\mathcal{SE}} \Rightarrow 1\right] &= 0\ . \end{aligned}$$

Hence $\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}}(A) = 1 - 0 = 1$. And A achieved this advantage by making just one lr-encryption oracle query, whose length, which as per our conventions is just the length of M_0, is n bits, and just one decryption oracle query, whose length is $2n$ bits. So $\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}}(t, 1, n, 1, 2n) = 1$.

Why are the two equations claimed above true? You have to return to the definitions of the quantities in question, as well as the description of the scheme itself, and walk it through. In world 1, meaning $b = 1$, the lr-encryption oracle returns $\langle \mathrm{IV}, C[1]\rangle$ with

$$C[1] \ = \ E_K(\mathrm{IV} \oplus M_1) \ = \ E_K(\mathrm{IV} \oplus 1^n)\ .$$

Now notice that

$$\begin{aligned} M &= \mathcal{D}_K(\langle \mathrm{IV}', C[1]\rangle) \\ &= E_K^{-1}(C[1]) \oplus \mathrm{IV}' \\ &= E_K^{-1}(E_K(\mathrm{IV} \oplus 1^n)) \oplus \mathrm{IV}' \\ &= (\mathrm{IV} \oplus 1^n) \oplus \mathrm{IV}'[0] \\ &= (\mathrm{IV} \oplus 1^n) \oplus (\mathrm{IV} \oplus 1^n) \\ &= 0^n \\ &= M_0\ . \end{aligned}$$

Thus, the decryption oracle will return M_0, and A will return 1. In world 0, meaning $b = 0$, the lr-encryption oracle returns $\langle \mathrm{IV}, C[1]\rangle$ with

$$C[1] \ = \ E_K(\mathrm{IV} \oplus M_0) \ = \ E_K(\mathrm{IV} \oplus 0^l)\ .$$

Now notice that

$$\begin{aligned} M &= \mathcal{D}_K(\langle \mathrm{IV}', C[1]\rangle) \\ &= E_K^{-1}(C[1]) \oplus \mathrm{IV}' \\ &= E_K^{-1}(E_K(\mathrm{IV} \oplus 0^n)) \oplus \mathrm{IV}' \\ &= (\mathrm{IV} \oplus 0^n) \oplus \mathrm{IV}'[0] \end{aligned}$$

$$\begin{aligned} &= (\mathrm{IV} \oplus 0^n) \oplus (\mathrm{IV} \oplus 1^n) \\ &= 1^n \\ &= M_1 \ . \end{aligned}$$

Thus, the decryption oracle will return M_1, and A will return 0, meaning will return 1 with probability zero. ▮

4.11 Historical notes

The pioneering work on the theory of encryption is that of Goldwasser and Micali [3], with refinements by [4, 2]. This body of work is however in the asymmetric (i.e., public key) setting, and uses the asymptotic framework of polynomial-time adversaries and negligible success probabilities. The treatment of symmetric encryption we are using is from [1]. In particular Definition 4.1.1 and the concrete security framework are from [1]. The analysis of the CTR and CBC mode encryption schemes, as given in Theorems 4.7.1, 4.7.2 and 4.8.1 is also from [1]. The analysis of the CBC mode here is new.

4.12 Problems

Problem 1 Formalize a notion of security against key-recovery for symmetric encryption schemes. Then prove that IND-CPA security implies key-recovery security.

Problem 2 Consider the following notion of indistinguishability of an encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$:

$$\mathbf{Adv}^{\text{ind0-cpa}}_{\mathcal{SE}}(A) \quad = \quad \Pr[K \xleftarrow{\$} \mathcal{K} : \ A^{\mathcal{E}_K(\cdot)} \Rightarrow 1] - \Pr[K \xleftarrow{\$} \mathcal{K} : \ A^{\mathcal{E}_K(0^{|\cdot|}} \Rightarrow 1] \ .$$

That is, a scheme in IND0-CPA secure if the encryption of every string looks like the encryption of an equal number of zeros. Here we assume that whenever M is in the message space, so is $0^{|M|}$. Prove that this notion of security is equivalent to IND-CPA security, carefully stating a pair of theorems and proving them.

Problem 3 The definition above for IND0-CPA provides the adversary with no method to get, with certitude, the encryption of a given message: when the adversary asks a query M, it might get answered with $C \xleftarrow{\$} \mathcal{E}_K(M)$ or it might get answered with $C \xleftarrow{\$} \mathcal{E}_K(0^{|M|})$. Consider providing the adversary an additional, "reference" oracle that *always* encrypts the queried string. Consider defining the corresponding advantage notion in the natural way: for an encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, let

$$\mathbf{Adv}^{\text{ind0-cpa*}}_{\mathcal{SE}}(A) \quad = \quad \Pr[K \xleftarrow{\$} \mathcal{K} : \ A^{\mathcal{E}_K(\cdot),\, \mathcal{E}_K(\cdot)} \Rightarrow 1] - \Pr[K \xleftarrow{\$} \mathcal{K} : \ A^{0^{|M|},\, \mathcal{E}_K(\cdot)} \Rightarrow 1]$$

State and prove a theorem that shows that this notion of security is equivalent to our original IND0-CPA notion (and therefore to IND-CPA).

Problem 4 Let $l \geq 1$ and $m \geq 2$ be integers, and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a given symmetric encryption scheme whose associated plaintext space is $\{0,1\}^n$, meaning one can only encrypt messages of

```
algorithm E_K^(m)(M)
    M[1]...M[m] ← M
    for i ← 1 to m do
        C[i] ← E_K(M[i])
    return C ← ⟨C[1],...,C[m]⟩

algorithm D_K^(m)(C)
    C[1]···C[m] ← C
    for i ← 1 to m do
        M[i] ← D_K(C[i])
        if M[i] = ⊥ then return ⊥
    return M ← ⟨M[1],...,M[m]⟩
```

Figure 4.11: Encryption scheme for Problem 4.

length n. In order to be able to encrypt longer messages, say ones of mn bits for some $m \geq 1$, we define a new symmetric encryption scheme $\mathcal{SE}^{(m)} = (\mathcal{K}, \mathcal{E}^{(m)}, \mathcal{D}^{(m)})$ having the same key-generation algorithm as that of $\mathcal{SE}$, plaintext space $\{0,1\}^{mn}$, and encryption and decryption algorithms as depicted in Fig. 4.11.

(a) Show that

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{SE}^{(m)}}(t, 1, mn, 1, mn) \quad = \quad 1$$

for some small t.

(b) Show that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}^{(m)}}(t, q, mnq) \quad \leq \quad \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t, mq, mnq)$$

for any t, q.

Part **(a)** says that $\mathcal{SE}^{(m)}$ is insecure against chosen-ciphertext attack. Note this is true regardless of the security properties of $\mathcal{SE}$, which may itself be secure against chosen-ciphertext attack. Part **(b)** says that if $\mathcal{SE}$ is secure against chosen-plaintext attack, then so is $\mathcal{SE}^{(m)}$.

Problem 5 The CBC-Chain mode of operation is a CBC variant in which the IV that is used for the very first message to be encrypted is random, while the IV used for each subsequent encrypted message is the last block of ciphertext that was generated. The scheme is probabilistic and stateful. Show that CBC-Chain is insecure by giving a simple and efficient adversary that breaks it in the IND-CPA sense.

Problem 6 Using the proof of Theorem 4.7.1 as a template, prove Theorem 4.7.2 assuming Lemma **??**.

Problem 7 Define a notion for indistinguishability from random bits, IND$-CPA. Your notion should capture the idea that the encryption of each message M looks like a string of random bits. Pay careful attention to the *number* of random bits that one outputs. Then formalize and prove that IND$-CPA security implies IND-CPA security—but that IND-CPA security does not imply IND$-CPA security.

Problem 8 Using a game-based argument, prove that CBC$[$\mathsf{Func}$($n$,$n$)] achieves IND$-CPA security. Assume that one encodes $\langle R, C\rangle$ as $R \,\|\, C$.

Problem 9 Devise a secure extension to CBC$ mode that allows messages of any bit length to be encrypted. Clearly state your encryption and decryption algorithm. Your algorithm should resemble CBC mode as much as possible, and should coincide with CBC mode when the message being encrypted is a multiple of the blocklength. It should increase the length of the message being encrypted by exactly n bits, where n is the length of the underlying blockcipher. How would you prove your algorithm secure?

Problem 10 An IND-CPA secure encryption scheme might not conceal identities, in the following sense: given a pair of ciphertexts C, C' for equal-length messages, it might be "obvious" if the ciphertexts were encrypted using the same random key or were encrypted using two different random keys. Give an example of a (plausibly) IND-CPA secure encryption scheme that has this is identity-revealing. Then give a definition for "identity-concealing" encryption. Your definition should imply IND-CPA security but a scheme meeting your definition can't be identity-revealing.

Bibliography

[1] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. *Proceedings of the 38th Symposium on Foundations of Computer Science*, IEEE, 1997.

[2] O. Goldreich. A uniform complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, Vol. 6, 1993, pp. 21-53.

[3] S. Goldwasser and S. Micali. Probabilistic encryption. *J. of Computer and System Sciences*, Vol. 28, April 1984, pp. 270–299.

[4] S. Micali, C. Rackoff and R. Sloan. The notion of security for probabilistic cryptosystems. *SIAM J. of Computing*, April 1988.

[5] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. *Proceedings of the 22nd Annual Symposium on the Theory of Computing*, ACM, 1990.

[6] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.