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, Chapter 6 looks at authenticity, and Chapter 7 looks at providing both together. Chapters 2 and 3 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 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 $\mathcal{E}$ takes a key $K \in \mathsf{Keys}(\mathcal{SE})$ and a *plaintext* $M \in \{0,1\}^*$ to return a *ciphertext* $C \in \{0,1\}^* \cup \{\perp\}$. This algorithm might be randomized or stateful. We write $C \xleftarrow{\$} \mathcal{E}_K(M)$.

- 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 $M \leftarrow \mathcal{D}_K(C)$.

We require that for any key $K \in \mathsf{Keys}(\mathcal{SE})$ and any message $M \in \{0,1\}^*$, if $\mathcal{E}_K(M)$ returns a ciphertext $C \neq \perp$ then $\mathcal{D}_K(C) = M$. ▌

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 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 is then 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.

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. If C was an output of $\mathcal{E}_K$ on input $M \in \mathsf{Plaintexts}$ then $\mathcal{D}_K(C)$ will equal M, enabling the receiver to recover the ciphertext. The decryption algorithm might however, fail to decrypt and return the special symbol $\perp$ to indicate that it deems the ciphertext invalid.

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, called the *plaintext space*, such that $\mathcal{E}_K(M) \neq \perp$ for all K and all M in the plaintext space. 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 it is fed to the algorithm.

4.2 Example encryption schemes (ECB, CBC, CTR)

Here are a few examples of encryption schemes. We stress that not all of the schemes here are *secure* encryption schemes. Some are secure and some are not, as we will see later. We begin with the classical one-time-pad. For a security analysis of this scheme, see Chapter B.

Scheme 4.2 [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 $ctr \leftarrow 1$	Let $m \leftarrow \lvert M\rvert$
Let $m \leftarrow \lvert M\rvert$	If $ctr + m - 1 > k$ then return $\perp$
If $ctr + m - 1 > k$ then return $\perp$	$M \leftarrow C \oplus K[ctr \mathinner{..} ctr + m - 1]$
$C \leftarrow M \oplus K[ctr \mathinner{..} ctr + m - 1]$	Return M
$ctr \leftarrow ctr + m$	
Return $\langle ctr, C\rangle$	

Here $X[i \mathinner{..} j]$ 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 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, must be stateless and deterministic, so we do not want it to have to maintain a counter as well.) ■

Algorithm $\mathcal{E}_K(M)$
  If ($|M| \bmod n \neq 0$ or $|M| = 0$) then return $\perp$
  Break M into n-bit blocks $M[1] \cdots M[m]$
  For $i \leftarrow 1$ to m do
    $C[i] \leftarrow E_K(M[i])$
  EndFor
  $C \leftarrow C[1] \cdots C[m]$
  Return C

Algorithm $\mathcal{D}_K(C)$
  If ($|C| \bmod n \neq 0$ or $|C| = 0$) then return $\perp$
  Break C into n-bit blocks $C[1] \cdots C[m]$
  For $i \leftarrow 1$ to m do
    $M[i] \leftarrow E_K^{-1}(C[i])$
  EndFor
  $M \leftarrow M[1] \cdots M[m]$
  Return M

Figure 4.1: ECB mode.

The following schemes rely either on a family of permutations (i.e., a block cipher) or a family of functions. Effectively, the mechanisms spell out how to use the block cipher to encrypt. We call such a mechanism a *mode of operation* of the block cipher. For some of the 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. Accordingly, we will let the encryption algorithm returns $\perp$ if this is not the case. 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 the padded message. The padding function should be injective and easily invertible. In this way you would create a new encryption scheme.

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

Scheme 4.3 [ECB mode] Let E: $\mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher. 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 block cipher, meaning it picks a random string $K \stackrel{\$}{\leftarrow} \mathcal{K}$ and returns it. The encryption and decryption algorithms are depicted in Figure 4.1. "Break M into n-bit blocks $M[1] \cdots M[m]$" means to set $m = |M|/n$ and, for $i \in \{1, \ldots, m\}$, set $M[i]$ to the i-th n-bit block in M, that is, $(i-1)n+1$ through in of M. Similarly for breaking C into $C[1] \cdots C[m]$. Notice that this time the encryption algorithm did not make any random choices. (That does not mean it is not, technically, a

Algorithm $\mathcal{E}_K(M)$
    If ($|M| \bmod n \neq 0$ or $|M| = 0$) then return $\perp$
    Break M into n-bit blocks $M[1] \cdots M[m]$
    $C[0] \leftarrow \text{IV} \xleftarrow{\$} \{0,1\}^n$
    For $i \leftarrow 1$ to m do
        $C[i] \leftarrow E_K(C[i-1] \oplus M[i])$
    EndFor
    $C \leftarrow C[1] \cdots C[m]$
    Return $\langle \text{IV}, C \rangle$

Algorithm $\mathcal{D}_K(\langle \text{IV}, C \rangle)$
    If ($|C| \bmod n \neq 0$ or $|M| = 0$) then return $\perp$
    Break C into n-bit blocks $C[1] \cdots C[m]$
    $C[0] \leftarrow \text{IV}$
    For $i \leftarrow 1$ to m do
        $M[i] \leftarrow E_K^{-1}(C[i]) \oplus C[i-1])$
    EndFor
    $M \leftarrow M[1] \cdots M[m]$
    Return M

Figure 4.2: CBC$ mode.

---

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 4.4 [CBC$ mode] Let E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. 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 block cipher, $K \xleftarrow{\$} \mathcal{K}$. The encryption and decryption algorithms are depicted in Figure 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. If $n \geq 1$ and $i \geq 0$ are integers then we let $[i]_n$ denote the n-bit string that is the binary representation of integer $i \bmod 2^n$. If we use a number $i \geq 0$ in 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.

Algorithm $\mathcal{E}_K(M)$
    static $ctr \leftarrow 0$
    If ($|M| \bmod n \neq 0$ or $|M| = 0$) then return $\perp$
    Break M into n-bit blocks $M[1]\cdots M[m]$
    If $ctr \geq 2^n$ then return $\perp$
    $C[0] \leftarrow \mathrm{IV} \leftarrow [ctr]_n$
    For $i \leftarrow 1$ to m do
        $C[i] \leftarrow E_K(C[i-1] \oplus M[i])$
    EndFor
    $C \leftarrow C[1]\cdots C[m]$
    $ctr \leftarrow ctr + 1$
    Return $\langle \mathrm{IV}, C\rangle$

Algorithm $\mathcal{D}_K(\langle \mathrm{IV}, C\rangle)$
    If ($|C| \bmod n \neq 0$ or $|C| = 0$) then return $\perp$
    Break C into n-bit blocks $C[1]\cdots C[m]$
    If $\mathrm{IV} + m > 2^n$ then return $\perp$
    $C[0] \leftarrow \mathrm{IV}$
    For $i \leftarrow 1$ to m do
        $M[i] \leftarrow E_K^{-1}(C[i]) \oplus C[i-1])$
    EndFor
    $M \leftarrow M[1]\cdots M[m]$
    Return M

Figure 4.3: CBCC mode.

Scheme 4.5 [CBCC mode] Let E: $\mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher. 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 block cipher, $K \stackrel{\$}{\leftarrow} \mathcal{K}$. The encryptor maintains a counter *ctr* which is initially zero. The encryption and decryption algorithms are depicted in Figure 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

Algorithm $\mathcal{E}_K(M)$
  $m \leftarrow \lceil |M|/n \rceil$
  $R \xleftarrow{\$} \{0,1\}^n$
  $Pad \leftarrow E_K(R+1) \,\|\, E_K(R+2) \,\|\, \cdots \,\|\, E_K(R+m)$
  $Pad \leftarrow$ the first $|M|$ bits of Pad
  $C' \leftarrow M \oplus Pad$
  $C \leftarrow R \,\|\, C'$
  Return C

Algorithm $\mathcal{D}_K(C)$
  If $|C| < n$ then return $\perp$
  Parse C into $R \,\|\, C'$ where $|R| = n$
  $m \leftarrow \lceil |C'|/n \rceil$
  $Pad \leftarrow E_K(R+1) \,\|\, E_K(R+2) \,\|\, \cdots \,\|\, E_K(R+m)$
  $Pad \leftarrow$ the first $|C'|$ bits of Pad
  $M \leftarrow C' \oplus Pad$
  Return M

Figure 4.4: CTR\$ mode using a block cipher E: $\mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$. This version of counter mode is probabilistic and stateless.

---

random and the other stateful, and, as we will see later, their security properties are different. For security analyses see Section 4.9 and Section 4.12.1.

Scheme 4.6 [CTR\$ mode] Let E: $\mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^\ell$ be be a block cipher. Then CTR mode over E 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 Figure 4.4. The starting point R is used to define a sequence of values on which E_K is applied to produce a "pseudo one-time pad" to which the plaintext is XORed. The starting point R chosen by the encryption algorithm is a random n-bit string. To add an n-bit string R to an integer i—when we write $E_K(R+i)$—convert the n-bit string R into an integer in the range $[0 \mathinner{..} 2^n - 1]$ 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 R is included in the ciphertext, to enable decryption. On encryption, the pad Pad is understood to be the empty string when $m = 0$. ▌

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

Scheme 4.7 [CTRC mode] Let E: $\mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher. Operating it in CTR mode with a counter starting point is a stateful symmetric

Algorithm $\mathcal{E}_K(M)$
  static $ctr \leftarrow 0$
  $m \leftarrow \lceil |M|/\ell \rceil$
  If $ctr + m - 1 \geq 2^n$ then return $\perp$
  $Pad \leftarrow E_K(ctr) \parallel E_K(ctr+1) \parallel \cdots \parallel E_K(ctr+m-1)$
  $Pad \leftarrow$ the first $|M|$ bits of Pad
  $C \leftarrow M \oplus Pad$
  $ctr \leftarrow ctr + m$
  Return $\langle ctr - m, C \rangle$

Algorithm $\mathcal{D}_K(\langle i, C \rangle)$
  $m \leftarrow \lceil |C|/\ell \rceil$
  $Pad \leftarrow E_K(i) \parallel E_K(i+1) \parallel \cdots \parallel E_K(i+m-1)$
  $Pad \leftarrow$ the first $C|$ bits of Pad
  $M \leftarrow Pad \oplus C$
  Return M

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

encryption scheme, $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, which we call CTRC. The key-generation algorithm simply returns a random key for E. The encryptor maintains a counter *ctr* which is initially zero. The encryption and decryption algorithms are depicted in Figure 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.

We comment that the "backwards direction" of E is never used and so CTRC and CTR\$ modes make sense even if E is not a block cipher but an arbitrary function family $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ instead. Indeed you could allow any function family $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^\ell$, not necessarily length-preserving, making the obvious changes to the schemes. In contrast, for modes CBCC and CBC\$ you do need to go backwards, though only for decryption. Thus even for these modes encryption we may consider encryption to be well-defined for any function family $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$.

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 $K \stackrel{\$}{\leftarrow} \mathcal{K}$. 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. 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. Yet, an absence of easy key recovery is not enough for the scheme to be secure; maybe the adversary can do something else.

One might want to say something like: 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.

One might also try to say that what we want is: given ciphertext C, the adversary can't easily recover the plaintext M. But actually, this isn't good enough. The 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.

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.

As an example, consider the ECB encryption scheme of Example 4.3. 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" block cipher, 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 I have just two messages, 0^n for *buy* and 1^n for *sell*. I keep sending data, but always one of these two. What happens? The adversary sees which plaintext blocks are the same. That is, it might see if the first two orders are the same, if they are equal to the third order, and so forth.

In a secure encryption scheme, it should not be possible to relate ciphertexts of different messages 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 B. 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

We have already discussed the issues in Section 4.3 above and will now distill a formal definition of security.

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, L and R. 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, $(L_1, R_1), \ldots, (L_q, R_q)$, where, in each pair, the two messages have the same length. We give to the adversary a sequence of ciphertexts $C_1, \ldots, C_q$. Either we consistently encrypt the L_i strings (the L stands for *left*) or we consistently encrypt the R_i strings (the R stands for *right*). In doing the encryptions, the algorithm uses fresh coins, or an updated state, each time. The adversary gets the sequence of ciphertexts and now it must guess if it has been the L_i strings or the R_i strings that we have been encrypting.

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 to which it can provide as input any pair (L, R) 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". To do this, first define the *left-or-right encryption oracle* $\mathcal{E}_K(\mathrm{LR}(\cdot, \cdot, b))$, as follows:

Oracle $\mathcal{E}_K(\mathrm{LR}(M_0, M_1, b))$ // $b \in \{0,1\}$ and $M_0, M_1 \in \{0,1\}^*$
  $C \leftarrow \mathcal{E}_K(M_b)$
  Return C

The oracle encrypts one of the messages, the choice of which being made according to the bit b. Now the two worlds are as follows:

World 0: The oracle provided to the adversary is $\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,0))$. So, whenever the adversary makes a query (M_0, M_1) to its oracle, the oracle computes $C \stackrel{\$}{\leftarrow} \mathcal{E}_K(M_0)$, and returns C as the answer.

World 1: The oracle provided to the adversary is $\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,1))$. So, whenever the adversary makes a query (M_0, M_1) to its oracle, the oracle computes $C \stackrel{\$}{\leftarrow} \mathcal{E}_K(M_1)$, and returns C as the answer.

We call the first world (or oracle) the "left" world (or oracle), and we call the second world (or oracle) the "right" world (or oracle). The problem for the adversary is, after talking to its oracle for some time, to tell which of the two oracles it was given. Before we pin this down, let us further clarify exactly how the oracles operate.

Think of an oracle as a subroutine to which A has access. Adversary A can make an oracle query (M_0, M_1) by calling the subroutine with arguments (M_0, M_1). In one step, the answer is then returned. Adversary A has no control on how the answer is computed, nor can A see the inner workings of the subroutine, which will typically depend on secret information that A is not provided. Adversary A has only an interface to the subroutine—the ability to call it as a black-box, and get back an answer.

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 $C \stackrel{\$}{\leftarrow} \mathcal{E}_K(M_b)$, 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 (either one) begins with a state value initialized to a value specified by the encryption scheme. For 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.

We clarify that 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 world 0, *all* message pairs sent to the oracle are answered by the oracle encrypting the left message in the pair, while in world 1, 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.

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 in distinguishing the cases $b = 0$ and $b = 1$ given access to the oracle, where reasonable reflects its resource usage. The technical notion is called left-or-right indistinguishability under chosen-plaintext attack, denoted IND-CPA.

Before presenting the technical definition we need to discuss a subtle point. Were we not careful, there would be certain queries that an adversary can make to its lr-encryption oracle which will definitely enable it to learn the value of the hidden bit b (meaning figure out in which world it is) but which we consider illegitimate. One is to query the oracle with messages M_0, M_1 of different lengths. We do not ask that encryption hide the length of the plaintext, and indeed common schemes reveal this because the length of the ciphertext depends on the length of the plaintext, so an adversary making such a query might easily win. Another, less obvious attack is for the adversary to make a query M_0, M_1 of equal-length messages such that $\mathcal{E}_K(M_0) \neq \perp$ and $\mathcal{E}_K(M_1) = \perp$. (If the scheme is stateless, this means M_0 is in the plaintext space and M_1 is not.) For some schemes, it is easy for the adversary to find such messages. However, the response of the lr-encryption oracle then gives away the bit b. We have chosen to deal with these issues by simply disallowing the adversary from making such queries. That is, let us say that an adversary is *illegitimate* if (for some coins it might be provided and for some sequence of oracle responses it might be given) it either makes an lr-encryption query consisting of two messages of different lengths or it makes an lr-encryption query M_0, M_1 for which $\mathcal{E}_K(M_0) = \perp$ or $\mathcal{E}_K(M_1) = \perp$. The adversary is legitimate if it is not illegitimate. We henceforth assume that any adversary for attacking IND-CPA indistinguishability is legitimate.

Assuming that any IND-CPA attacking adversary is legitimate is not a significant assumption, and, when convenient, we will always feel at liberty to make such restrictions on our universe of adversaries. When you make a restriction like this it is natural to ask: but what if the adversary is *not* legitimate. One answer is to say that the advantage of the adversary is, in such a case, defined as zero. A different answer is to say that the question is meaningless: when we define $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A)$ we are defining this value for adversaries that are of the type-correct, and asking about $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A)$ for an illegitimate adversary A doesn't make much sense than asking about $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A)$ where A is a real number, a set of strings or guinea pig.

Definition 4.8 Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. We define from $\mathcal{SE}$ the following two oracles:

Oracle Left	Oracle Right
Initialization	**Initialization**
$K \stackrel{\$}{\leftarrow} \mathcal{K}$	$K \stackrel{\$}{\leftarrow} \mathcal{K}$
On query (L, R)	**On query** (L, R)
if $\lvert L\rvert \neq \lvert R\rvert$ **then return** $\perp$	**if** $\lvert L\rvert \neq \lvert R\rvert$ **then return** $\perp$
$C \stackrel{\$}{\leftarrow} \mathcal{E}_K(L)$	$C \stackrel{\$}{\leftarrow} \mathcal{E}_K(R)$
Return C	Return C

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

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

▮

We discuss some important conventions for when we speak of the resource 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 experiment used to define some particular advantage notion). 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 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 $|M_0|$. (We can assume this equals $|M_1|$ since the adversary is assumed to be legitimate.) 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.

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. 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.

Let us move on to describe a somewhat different interpretation of left-or-right 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.

As usual we fix a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. We now consider the following game, or experiment.

Experiment $\mathbf{Exmt}^{\text{ind1-cpa}}_{\mathcal{SE}}(A)$
Pick a bit b at random
Let $K \xleftarrow{\$} \mathcal{K}$
$b' \xleftarrow{\$} A^{\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))}$
If $b = b'$ return 1 else return 0

Here, adversary A is run with an oracle for world b, where the bit b is chosen at random. A eventually outputs a bit b', its guess as to the value of b. The experiment returns 1 if A's guess is correct. Thus

$$\Pr\left[\mathbf{Exmt}^{\text{ind1-cpa}}_{\mathcal{SE}}(A) = 1\right]$$

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." So we re-scale the value and define

$$\mathbf{Adv}^{\text{ind1-cpa}}_{\mathcal{SE}}(A) \quad = \quad 2\Pr\left[\mathbf{Exmt}^{\text{ind1-cpa}}_{\mathcal{SE}}(A) = 1\right] - 1$$

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

Proposition 4.9 Let $\mathcal{SE}$ be a symmetric encryption scheme and let A be an adversary. Then

$$\mathbf{Adv}^{\text{ind1-cpa}}_{\mathcal{SE}}(A) \quad = \quad \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A)\ .$$

■

Proof of Proposition 4.9: We let $\Pr[\cdot]$ be the probability of event "$\cdot$" in the experiment $\mathbf{Exmt}^{\text{ind1-cpa}}_{\mathcal{SE}}(A)$, and refer below to quantities in this experiment. The claim of the Proposition follows by a straightforward calculation:

$$\begin{aligned}
&\Pr\left[\mathbf{Exmt}^{\text{ind1-cpa}}_{\mathcal{SE}}(A) = 1\right] \\
&\quad = \Pr[b = g] \\
&\quad = \Pr\left[b = b' \mid b = 1\right] \cdot \Pr[b = 1] + \Pr\left[b = b' \mid b = 0\right] \cdot \Pr[b = 0]
\end{aligned}$$

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

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 experiments of Definition 4.8. ▮

4.5 Example chosen-plaintext attacks

We illustrate the use of our CPA-IND 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 block cipher E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. The ECB symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ was described as Scheme 4.3. 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" block cipher. 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. This is what the following proposition says.

Proposition 4.10 Let E: $\mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher, and $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ the corresponding ECB symmetric encryption scheme as described in Scheme 4.3. Then there is an adversary A that runs in time $O(n)$ and asks a single query, this query having length $2n$, and for which

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

∎

The advantage of this adversary is 1 even though it uses hardly any resources. That is an indication that the scheme is insecure.

Proof of Proposition 4.10: We must construct the adversary A. Remember that A is given an lr-encryption oracle $\mathcal{E}_K(\text{LR}(\cdot,\cdot,b))$ 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 $A^{\mathcal{E}_K(\text{LR}(\cdot,\cdot,b))}$
  $M_1 \leftarrow 0^{2n}$; $M_0 \leftarrow 0^n \,\|\, 1^n$
  $C[1]C[2] \leftarrow \mathcal{E}_K(\text{LR}(M_0, M_1, b))$
  If $C[1] = C[2]$ then return 1 else return 0

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

$$\begin{aligned}
\Pr\left[\mathbf{Exmt}^{\text{ind-cpa-1}}_{\mathcal{SE}}(A) = 1\right] &= 1 \;\text{ and} \\
\Pr\left[\mathbf{Exmt}^{\text{ind-cpa-0}}_{\mathcal{SE}}(A) = 1\right] &= 0 \,.
\end{aligned}$$

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

Why are the two equations claimed above true? You have to return to the definitions of the quantities in question, and trace through the experiments defined there. In world 1, meaning $b = 1$, 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 world 0, meaning $b = 0$, 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. ∎

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 block cipher E is highly secure*. The weakness is not in the tool being used (here the block cipher) 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 block cipher 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 Deterministic, stateless schemes are always 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 4.11 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 4.11: We must describe the adversary A. Remember that A is given an lr-encryption oracle $f = \mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))$ that 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^f
  Let X, Y be distinct, m-bit strings in the plaintext space
  $C_1 \leftarrow f(X, Y)$
  $C_2 \leftarrow f(Y, Y)$
  If $C_1 = C_2$ then return 1 else return 0

Now, we claim that

$$\begin{aligned}\Pr\left[\mathbf{Exmt}_{\mathcal{SE}}^{\text{ind-cpa-1}}(A)=1\right] &= 1 \text{ and}\\ \Pr\left[\mathbf{Exmt}_{\mathcal{SE}}^{\text{ind-cpa-0}}(A)=1\right] &= 0\ .\end{aligned}$$

Hence $\mathbf{Adv}_{\mathcal{SE}}^{\text{ind-cpa}}(A) = 1 - 0 = 1$. 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.

Why are the two equations claimed above true? In world 1, meaning $b = 1$, 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 world 0, meaning $b = 0$, 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. ▮

4.5.3 Attack on CBC encryption with counter IV

Let us fix a block cipher $E\colon\ \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. We show that the counter-based version of the CBC encryption mode described in Scheme 4.5 is insecure. The reason is that the adversary can predict the counter value.

Proposition 4.12 Let $E\colon\ \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher, and $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ the corresponding CBCC symmetric encryption scheme as described in Scheme 4.5. Then there exists an adversary A such that

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

where A run in time for $O(n)$ and A asks just two queries, each of n bits. ▮

The advantage of this adversary is 1 even though it uses hardly any resources: just one query, and that query just one block long. That is clearly an indication that the scheme is insecure.

Proof of Proposition 4.12: Again, we must construct an adversary A that uses the specified resources and obtains the specified advantage. Remember the adversary A is given an lr-encryption oracle $f = \mathcal{E}_K(\text{LR}(\cdot,\cdot,b))$ that takes as 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 the hidden bit b. The goal of A is to determine the value of b. Our adversary works like this:

Adversary $A^{\mathcal{E}_K(\text{LR}(\cdot,\cdot,b))}$
  $L^1 \leftarrow 0^n$; $R^1 \leftarrow 0^n$
  $L^2 \leftarrow 0^n$; $R^2 \leftarrow 0^{n-1}1$
  $\langle \text{IV}^1, C^1\rangle \leftarrow f(L^1, R^1)$
  $\langle \text{IV}^2, C^2\rangle \leftarrow f(L^2, R^2)$
  If $C^1 = C^2$ then return 1 else return 0

We claim that

$$\begin{aligned}\Pr\left[\mathbf{Exmt}_{\mathcal{SE}}^{\text{ind-cpa-1}}(A)=1\right] &= 1 \text{ and} \\ \Pr\left[\mathbf{Exmt}_{\mathcal{SE}}^{\text{ind-cpa-0}}(A)=1\right] &= 0\,.\end{aligned}$$

Hence $\mathbf{Adv}_{\mathcal{SE}}^{\text{ind-cpa}}(A) = 1 - 0 = 1$, as desired. To justify the above equalities, first consider the case $b = 0$, meaning we are in world 0. In that case $\text{IV}^1 = 0$ and $\text{IV}^1 = 1$ and $C^1 = E_K(0)$ and $C^2 = E_K(1)$ and so $C^1 \neq C^2$ and the defined experiment returns 0. On the other hand, if $b = 1$, meaning we are in world 1, then $\text{IV}^1 = 0$ and $\text{IV}^1 = 1$ and $C^1 = E_K(0)$ and $C^2 = E_K(0)$, so the defined experiment returns 1. ▮

4.6 Notions equivalent to indistinguishability

One of the ways that we gain confidence that a definition is interesting and meaningful is to demonstrate that it is equivalent to alternative, different-looking formulations. In this section we'll look at two such alternatives. The first notion, indistinguishability from the encryption of random bits, captures the idea that an adversary shouldn't be able tell apart a "real" encryption oracle from an encryption oracle that, instead of encrypting its oracle query, just encrypts a string of random bits. Our second alternative notion, semantic security, captures the idea that a secure encryption scheme should hide all partial information about an unknown plaintext. This latter formulation may most directly match the intuition about what secure encryption ought to achieve.

4.6.1 Real-or-zero security

Fix an encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Now consider two types of oracles: a *real* encryption-oracle and a *zero* encryption-oracle. The behavior of the oracles depends on the encryption scheme $\mathcal{SE}$. Both oracles begin by choosing a random key $K \xleftarrow{\$} \mathcal{K}$ from the keyspace. Afterwards, the Real encryption oracle responds to any query M by computing a ciphertext $C \xleftarrow{\$} \mathcal{E}_K(M)$ and returning C. In contrast, the Zero encryption-oracle responds to a query M by encrypting $0^{|M|}$: it computes and returns $C \xleftarrow{\$} \mathcal{E}_K(0^{|M|})$. The adversary's job is to identify if it is in possession of a Real encryption-oracle or a Zero encryption-oracle. The formal definition follows:

Definition 4.13 [Real-or-Zero security] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. We define the following two oracles:

Oracle Real	Oracle Zero
Initialization:	**Initialization:**
$K \xleftarrow{\$} \mathcal{K}$	$K \xleftarrow{\$} \mathcal{K}$
On query M**:**	**On query** M**:**
$C \xleftarrow{\$} \mathcal{E}_K(M)$	$C \xleftarrow{\$} \mathcal{E}_K(0^{\|M\|})$
return C	**return** C

The *RZ-CPA advantage* of A is defined as

$$\mathbf{Adv}^{\text{rz-cpa}}_{\mathcal{SE}}(A) \quad = \quad \Pr[A^{\text{Real}(\cdot)}{\Rightarrow}1] - \Pr[A^{\text{Zero}(\cdot)}{\Rightarrow}1]$$

▮

We now show that RZ-CPA security is equivalent to IND-CPA security. This entails that RZ-CPA security implies IND-CPA security, and IND-CPA security implies RZ-CPA.

Proposition 4.14 [IND-CPA $\Rightarrow$ RZ-CPA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let A be an adversary (for attacking the RZ-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 an adversary B (for attacking the IND-CPA security of $\mathcal{SE}$) such that

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

and where B runs in time $t' = t + O(\mu)$ and asks $q' = q$ queries, these queries totaling μ bits. ▮

Proof: The adversary B is constructed as follows:

Adversary B^g:
  Run A^f
    When A makes a query $f(M)$:
      $C \leftarrow g(0^{|M|},\ M)$
      Answer A's query by C
    When A halts, outputting a bit β
      Return β

So B runs in at most $t + O(\mu)$ time and asks at most q queries and these queries total at most μ bits. As for B's advantage, note that

$$\Pr[B^{\text{Right}(\cdot,\cdot)}{\Rightarrow}1] \quad = \quad \Pr[A^{\mathcal{E}_K(\cdot)}{\Rightarrow}1] \text{ and}$$

$$\Pr[B^{\text{Left}(\cdot,\cdot)}{\Rightarrow}1] \quad = \quad \Pr[A^{\mathcal{E}_K(0^{|\cdot|})}{\Rightarrow}1]$$

and so

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(B) &= \Pr[B^{\text{Right}(\cdot,\cdot)}\Rightarrow 1] - \Pr[B^{\text{Left}(\cdot,\cdot)}\Rightarrow 1] \\ &= \Pr[A^{\mathcal{E}_K(\cdot)}\Rightarrow 1] - \Pr[A^{\mathcal{E}_K(0^{|\cdot|})}\Rightarrow 1] \\ &= \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(B) = \mathbf{Adv}^{\text{rz-cpa}}_{\mathcal{SE}}(A)\end{aligned}$$

completing the proof. ▮

Proposition 4.15 [RZ-CPA ⇒ IND-CPA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let A be an adversary (for attacking the IND-CPA security of $\mathcal{SE}$) that runs in time at most t, asking at most q queries, these queries totaling at most μ bits. Then there exists an adversary B (for attacking the RZ-CPA security of $\mathcal{SE}$) such that

$$\mathbf{Adv}(B) \quad \geq \quad \mathbf{Adv}(A)$$

and where B runs in time $t' = t + O(\mu)$ and asks $q' = q$ queries, these queries totaling μ bits. ▮

The proof is by a *hybrid argument*. In such an argument one inserts one or more “intermediate behaviors” between the behaviors one is interested in. Here the hybrid is something we regard as being “between” the behavior of a left encryption-oracle and the behavior of a right encryption oracle.

Proof: Given the encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, define oracle O to behave in the following way:

<u>Oracle O</u>

Initialization:
$K \xleftarrow{\$} \mathcal{K}$

On query (L, R) **(where** $|L| = |R|$**):**
$C \xleftarrow{\$} \mathcal{E}_K(0^{|L|})$
return C

Now

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) &= \Pr[A^{\text{Right}(\cdot,\cdot)}\Rightarrow 1] - \Pr[A^{\text{Left}(\cdot,\cdot)}\Rightarrow 1] \\ &= \Big(\Pr[A^{\text{Right}(\cdot,\cdot)}\Rightarrow 1] - \Pr[A^{O(\cdot,\cdot)}\Rightarrow 1]\Big) + \\ &\quad\ \Big(\Pr[A^{O(\cdot,\cdot)}\Rightarrow 1] - \Pr[A^{\text{Left}(\cdot,\cdot)}\Rightarrow 1]\Big)\end{aligned}$$

and so either

$$\Pr[A^{\text{Right}(\cdot,\cdot)}\Rightarrow 1] - \Pr[A^{O(\cdot,\cdot)}\Rightarrow 1] \quad \geq \quad 0.5\ \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \tag{4.1}$$

or

$$\Pr[A^{O(\cdot,\cdot)}\Rightarrow 1] - \Pr[A^{\mathrm{Left}(\cdot,\cdot)}\Rightarrow 1] \quad \geq \quad 0.5\,\mathbf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathcal{SE}}(A)\ . \tag{4.2}$$

In the first case, we build the adversary B in one way; in the second case, will build the adversary B in a different way. Let us first assume Equation 4.1. In that case, adversary B works as follows:

Adversary B^g

Run A^f
    When A makes oracle query $f(L,R)$
        Compute $C \leftarrow f(R)$
        Answer the adversary with C
    When A halts with an output of β
        Return β

Note that B runs in time at most $t+O(\mu)$ and it asks q queries, these totaling μ bits. If B is presented a Real$(\cdot)$ oracle then A is provided an environment identical to its having been presented an oracle Right$(\cdot,\cdot)$. On the other hand, if B is presented a Zero$(\cdot)$ oracle then A is provided an environment identical to its having been presented an oracle $O(\cdot,\cdot)$. Thus

$$\begin{aligned}\mathbf{Adv}^{\mathrm{rz\text{-}cpa}}_{\mathcal{SE}}(B) \quad &= \quad \Pr[B^{\mathrm{Real}(\cdot,\cdot)}\Rightarrow 1] - \Pr[B^{\mathrm{Zero}(\cdot,\cdot)}\Rightarrow 1] \\ &= \quad \Pr[A^{\mathrm{Right}(\cdot,\cdot)}\Rightarrow 1] - \Pr[A^{O(\cdot,\cdot)}\Rightarrow 1] \\ &\geq \quad 0.5\,\mathbf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathcal{SE}}(A)\end{aligned}$$

which completes the first case.

If Equation (4.2) holds instead, then we will construct the adversary B as follows:

Adversary B^g

Run A^f
    When A makes oracle query $f(L,R)$
        Compute $C \leftarrow f(L)$
        Answer the adversary with C
    When A halts with an output of β
        Return $1-\beta$

Note that B runs in time at most $t+O(\mu)$ and it asks q queries, these totaling μ bits. If B is presented a Real$(\cdot)$ oracle then A is provided an environment identical to its having been presented an oracle Left$(\cdot,\cdot)$, and then we flip the answer that A would return. On the other hand, if B is presented a Zero$(\cdot)$ oracle then A is provided an environment identical to its having been presented an oracle $O(\cdot,\cdot)$, and then we flip the answer that A would return. Thus

$$\begin{aligned}\mathbf{Adv}^{\text{rz-cpa}}_{\mathcal{SE}}(B) &= \Pr[B^{\text{Real}(\cdot,\cdot)}{\Rightarrow}1] - \Pr[B^{\text{Zero}(\cdot,\cdot)}{\Rightarrow}1] \\ &= (1 - \Pr[A^{\text{Left}(\cdot,\cdot)}{\Rightarrow}1]) - (1 - \Pr[A^{O(\cdot,\cdot)}{\Rightarrow}1]) \\ &= \Pr[A^{O(\cdot,\cdot)}{\Rightarrow}1] - \Pr[A^{\text{Left}(\cdot,\cdot)}{\Rightarrow}1]) \\ &\geq 0.5\,\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A)\end{aligned}$$

which completes the second case and the proof. ▌

Note: need to add some technical condition about 0^i being in the domain?

4.6.2 Semantic security

Definitions like that of IND-CPA or RZ-CPA can be criticized for failing to directly model what people "intend" when the speak of security in the face of a chosen-plaintext attack. Though such a thing is hard to pin down, the intent would seem to have more to do with the hope that encryption protects the privacy of plaintexts when you see ciphertexts and don't know what these plaintexts are. In particular, to call an encryption scheme secure one wants to believe that a ciphertext reveals nothing significant about the underlying plaintext. So far, we have given notions where the adversary's job has nothing to do with trying to say something meaningful about plaintexts whose ciphertexts are sees.

It takes work to raise intuition like that of the previous paragraph into a definition. In the case of encryption, Shafi Goldwasser and Silvio Micali were the first accomplish this. The intuition of theirs that we will try to capture in a definition is this: *that an encryption scheme is secure if that which can be determined about some plaintexts from their ciphertexts can just as easily be computed in the absence of those ciphertexts.* We want to raise this intuition to a definition, which we shall do in our own way.

To make for a very strong definition, we will let the adversary choose both the message spaces from which messages are drawn and the information-function about messages that the adversary wants to second guess. To make the definition as general as possible, we will let the adversary choose, in sequence, message spaces $\mathcal{M}_1, \ldots, \mathcal{M}_r$. From each message space $\mathcal{M}_i$ a message M_i will be chosen at random and encrypted to C_i using our encryption scheme. Then the adversary will try to compute something about the vector of plaintexts $(M_1, \ldots, M_r)$. The something that the adversary computes will be an arbitrary function F of the adversary's choice. We will compare how good a job A does at computing $F(M_1, \ldots, \mathcal{M}_r)$ with how good a job A does at a different task. That "different task" is computing $F(M_1, \ldots, M_r)$ where the adversary has no information about $M_1, \ldots, M_r$ except that these strings are drawn from the distributions the adversary has indicated. In carrying out all of the above, we give the adversary an encryption oracle, so that it can mount a chosen-plaintext attack. The definition now follows.

Definition 4.16 **[Semantic security]** Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and define the following oracle from it:

Oracle SampleThenEncrypt	Oracle SampleThenEncrypt$'$
Initialization:	**Initialization:**
$q \leftarrow 0$	$i \leftarrow 0$
$K \xleftarrow{\$} \mathcal{K}$	$K \xleftarrow{\$} \mathcal{K}$
On query $\mathcal{M}$:	**On query $\mathcal{M}$:**
$q \leftarrow q + 1$	$q \leftarrow q + 1$
$M_q \xleftarrow{\$} \mathcal{M}_q$	$M_q,\, M'_q \xleftarrow{\$} \mathcal{M}_q$
$C \xleftarrow{\$} \mathcal{E}_K(M_q)$	$C \xleftarrow{\$} \mathcal{E}_K(M'_q)$
return C	**return** C

The *SEM-CPA advantage* of A is defined as

$$\begin{aligned}\mathbf{Adv}^{\text{sem-cpa}}_{\mathcal{SE}}(A) \;&=\; \Pr[(F,Y) \xleftarrow{\$} A^{\text{SampleThenEncrypt}(\cdot)} \,:\, F(M_1,\ldots,M_q) = Y] - \\ &\;\; \Pr[(F,Y) \xleftarrow{\$} A^{\text{SampleThenEncrypt}'(\cdot)} \,:\, F(M_1,\ldots,M_q) = Y]\end{aligned}$$

∎

In the definition above, as the oracles are executed they define the variable q, for the total number of oracle queries, and the strings $M_1, \ldots, M_q$. These values are referred to in the subsequent experiment that defines SEM-CPA advantage. By $\mathcal{M}$ we denote a probabilistic algorithm, described relative to some standard encoding, that produces a string output. We insist that if M and M' are output with nonzero probability by $\mathcal{M}$ then $|M| = |M'|$. By F we denote a deterministic function (described by some standard encoding) that produces a string output. By Y we denote a string.

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 $F(M_1, \ldots, M_r)$ over any vector of strings. In speaking of the length of A's queries we include, for any message space $\mathcal{M}$ output by A, the length of a string M output with nonzero probability by $\mathcal{M}$.

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 get encrypted. We have let the adversary choose the partial information about the messages that it finds convenient. We have let the adversary to be fully adaptive. Note that the adversary can get the encryption of any desired message M simply by producing an algorithm $\mathcal{M}$ that samples that one desired message.

We now show that security in the sense of left-or-right indistinguishability implies semantic security.

Theorem 4.17 [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

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

and where B runs in time $2t + O(q + \mu)$ and asks at most q queries, these queries totaling μ bits. ▮

Proof: The adversary B^g is constructed as follows.

Adversary B^g

$i \leftarrow 0$
Run A^f
  When A makes an oracle query $\mathcal{M}$
    $i \leftarrow i + 1$
    Compute $M_i, M_i' \xleftarrow{\$} \mathcal{M}$
    Compute $C \leftarrow f(M_i, M_i')$
    Answer the adversary's query by C
  When A halts with an output (F, Y)
    if $F(M_1, \ldots, M_i) = Y$ **then return** 0 **else return** 1

We observe that the running time of B is at most $2t + O(\mu + q)$, where the multiplicative factor of two takes account of the fact that we are sampling two points instead of one from each message space $\mathcal{M}$ queried by the adversary A. In addition, adversary B asks at most q oracle queries and these queries total at most μ bits. To verify the advantage that is claimed of the adversary, note that if oracle g is a left encryption-oracle then we are providing A with a perfect simulation of the experiment associated to oracle SampleThenEncrypt, while if oracle g is a right encryption-oracle then we are providing A with a perfect simulation of the experiment associated to having a SampleThenEncrypt$'$ oracle. ▮

4.7 Indistinguishability implies security against plaintext recovery

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 these properties, in the sense that a scheme that is secure in the sense of our definition is also secure against key-recovery or plaintext-recovery.

The situation is analogous to what we saw in the case of PRFs. There we showed that a secure PRF is secure against key-recovery. In order to have some variation, this time we choose a different property, namely plaintext recovery. We formalize this, and then show if there was an adversary B capable of recovering the plaintext from a given ciphertext, then this would enable us to construct an adversary A that broke the scheme in the IND-CPA sense (meaning the adversary can identify which of the two worlds it is in). If the scheme is secure in the IND-CPA sense, that latter adversary could not exist, and hence neither could the former.

The idea of this argument illustrates one way to evidence that a definition is good—say the definition of left-or-right indistinguishability. Take some property that you feel a secure scheme should have, like infeasibility of key recovery from a few plaintext-ciphertext pairs, or infeasibility of predicting the XOR of the plaintext bits. Imagine there were an adversary B that was successful at this task. We should show that this would enable us to construct an adversary A that broke the scheme in the original sense (left-or-right indistinguishability). Thus the adversary B does not exist if the scheme is secure in the left-or-right sense. More precisely, we use the advantage function of the scheme to bound the probability that adversary B succeeds.

Let us now go through the plaintext recovery example in detail. The task facing the adversary will be to decrypt a ciphertext which was formed by encrypting a randomly chosen challenge message of some length m. In the process we want to give the adversary the ability to see plaintext-ciphertext pairs, which we capture by giving the adversary access to an encryption oracle. This encryption oracle is not the lr-encryption oracle we saw above: instead, it simply takes input a single message M and returns a ciphertext $C \xleftarrow{\$} \mathcal{E}_K(M)$ computed by encrypting M. To capture providing the adversary with a challenge ciphertext, we choose a random m-bit plaintext M, compute $C \xleftarrow{\$} \mathcal{E}_K(M)$, and give C to the adversary. The adversary wins if it can output the plaintext M corresponding to the ciphertext C.

For simplicity we assume the encryption scheme is stateless, and that $\{0,1\}^m$ is a subset of the plaintext space associated to the scheme. As usual, when either the encryption or the challenge oracle invoke the encryption function, it is implicit that they respect the randomized nature of the encryption function, meaning the latter tosses coins anew upon each invocation of the oracle.

Definition 4.18 Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a stateless symmetric encryption scheme whose plaintext space includes $\{0,1\}^m$ and let B be an algorithm that has access to an oracle. We consider the following experiment:

Experiment $\mathbf{Exmt}^{\text{pr-cpa}}_{\mathcal{SE}}(B)$

$K \xleftarrow{\$} \mathcal{K}$

$M' \xleftarrow{\$} \{0,1\}^m$

$C \xleftarrow{\$} \mathcal{E}_K(M')$

$M \xleftarrow{\$} B^{\mathcal{E}_K(\cdot)}(C)$

If $M = M'$ then return 1 else return 0

The *pr-advantage* of B is defined as

$$\mathbf{Adv}^{\text{pr-cpa}}_{\mathcal{SE}}(B) \quad = \quad \Pr\left[\mathbf{Exmt}^{\text{pr-cpa}}_{\mathcal{SE}}(B) = 1\right] .$$

▮

In the experiment above, B is executed with its oracle and challenge ciphertext C. The adversary B wins if it can correctly decrypt C, and in that case the experiment returns 1. In the process, the adversary can make encryption oracle queries as it pleases.

The following Proposition says that the probability that an adversary successfully recovers a plaintext from a challenge ciphertext cannot exceed the IND-CPA advantage of the scheme (with resource parameters those of the plaintext recovery adversary) plus the chance of simply guessing the plaintext. In other words, security in the IND-CPA sense implies security against plaintext recovery.

Proposition 4.19 [IND-CPA $\Rightarrow$ PR-CPA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a stateless symmetric encryption scheme whose plaintext space includes $\{0,1\}^m$. Suppose that B is a (plaintext-recovery) adversary that runs in time t and asks at most q queries, these queries totaling at most μ bits. Then there exists an adversary A such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \quad \geq \quad \mathbf{Adv}^{\text{pr-cpa}}_{\mathcal{SE}}(B) - \frac{1}{2^m} .$$

Furthermore, the running time of A is $t + O(\mu + m + c)$ where c bounds the length of the encryption of an m-bit string and A makes $q + 1$ oracle queries and these queries total at most $\mu + m$ bits. ▮

Recall that turning one adversary B into another type of adversary A is called a called a *reduction*. Don't do a reduction "backwards." When you transform B, an adversary for goal **B**, into A, an adversary for goal **A**, you are showing that *in*security in the **B**-sense implies insecurity in the **A**-sense, so security in the **A**-sense implies security in the **B**-sense.

Proof of Proposition 4.19: As per Definition 4.8, adversary A will be provided an lr-encryption oracle and will try to determine in which world it resides. To do so, it will run adversary B as a subroutine. We provide the description followed by an explanation and analysis.

Adversary $A^{f(\cdot,\cdot)}$
$M_0 \xleftarrow{\$} \{0,1\}^m$; $M_1 \xleftarrow{\$} \{0,1\}^m$
$C \leftarrow f(M_0, M_1)$
Run adversary $B^g(C)$, replying to its oracle queries as follows
When B makes an oracle query X to g do
$Y \leftarrow f(X, X)$
Return Y to B as the answer
When B halts and outputs a plaintext M
If $M = M_1$ then return 1 else return 0

Here A is running B and itself providing answers to B's oracle queries. To make the challenge ciphertext C for B, adversary A chooses random messages M_0 and M_1 and uses its lr-oracle to get the encryption C of one of them. When B makes an encryption oracle query X, adversary A needs to return $\mathcal{E}_K(X)$. It does this by invoking its lr-encryption oracle, setting both messages in the pair to X, so that regardless of the value of the bit b, the ciphertext returned is an encryption of X, just as B wants. When B outputs a plaintext M, adversary A tests whether $M = M_1$ and if so bets that it is in world 1. Otherwise, it bets that it is in world 0. Now we claim that

$$\Pr\left[\mathbf{Exmt}^{\text{ind-cpa-1}}_{\mathcal{SE}}(A) = 1\right] \;\geq\; \mathbf{Adv}^{\text{pr-cpa}}_{\mathcal{SE}}(B) \tag{4.3}$$

$$\Pr\left[\mathbf{Exmt}^{\text{ind-cpa-0}}_{\mathcal{SE}}(A) = 1\right] \;\leq\; 2^{-m} \,. \tag{4.4}$$

We will justify these claims shortly, but first let us use them to conclude. Subtracting, as per Definition 4.8, we get

$$\begin{aligned} \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) &= \Pr\left[\mathbf{Exmt}^{\text{ind-cpa-1}}_{\mathcal{SE}}(A) = 1\right] - \Pr\left[\mathbf{Exmt}^{\text{ind-cpa-0}}_{\mathcal{SE}}(A) = 1\right] \\ &\geq \mathbf{Adv}^{\text{pr-cpa}}_{\mathcal{SE}}(B) - 2^{-m} \,. \end{aligned}$$

It remains to justify Equations (4.3) and (4.4).

Adversary B will return the $M = \mathcal{D}_K(C)$ with probability at least $\mathbf{Adv}^{\text{pr-cpa}}_{\mathcal{SE}}(B)$. In world 1, ciphertext C is an encryption of M_1, so this means that $M = M_1$ with probability at least $\mathbf{Adv}^{\text{pr-cpa}}_{\mathcal{SE}}(B)$, and thus Equation (4.3) is true. Now assume A is in world 0. In that case, adversary A will return 1 only if B returns $M = M_1$. But B is given no information about M_1, since C is an encryption of M_0 and M_1 is chosen randomly and independently of M_0. It is simply impossible for B to output M_1 with probability greater than 2^{-m}. Thus Equation (4.4) is true. ▮

Similar arguments can be made to show that other desired security properties of a symmetric encryption scheme follow from this definition. For example, is it possible that some adversary B, given some plaintext-ciphertext pairs and then a challenge ciphertext C, can compute the XOR of the bits of $M = \mathcal{D}_K(C)$? Or the sum of these bits? Or the last bit of M? Its probability of doing any of these cannot be more than marginally above $1/2$ because were it so, we could design an adversary A that won the left-or-right game using resources comparable to those used by B. We leave as an exercise the formulation and working out of other such examples along the lines of Proposition 4.19.

Of course one cannot exhaustively enumerate all desirable security properties. But you should be moving towards being convinced that our notion of left-or-right security covers all the natural desirable properties of security under chosen plaintext attack. Indeed, we err, if anything, on the conservative side. There are some attacks that might in real life be viewed as hardly damaging, yet our definition declares the scheme insecure if it succumbs to one of these. That is all right; there is no harm in making our definition a little demanding. What is more important is that if there is any attack that in real life would be viewed as damaging, then the scheme will fail the left-or-right test, so that our formal notion too declares it insecure.

4.8 Indistinguishability from random bits implies indistinguishability

In this section we give what might seem to be an inappropriately *strong* definition of security: we consider demanding that ciphertexts look like sequences of random bits. We show that this notion of security implies left-or-right indistinguishability. On the other hand, we show that left-or-right indistinguishability does not imply indistinguishability from random bits.

Why do we say that indistinguishability from random bits might seem *too* strong for a notion of encryption-scheme privacy? Because our intuition is that, fundamentally, it simply *does not matter* if ciphertexts look random or not. If a ciphertext hides everything about the underlying plaintext, but happens not to look random, is that not good enough?

Said differently, suppose you started with an encryptions scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and then you modified it to make an encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ by arranging that every ciphertext has tacked on to its end 100 zero-bits. Decryption would ignore these final 100 bits. Now even if ciphertexts happened to look random in the original scheme, most certainly they would not look random in the modified scheme. But what you have done seems irrelevant to preserving the user's privacy. If you were protecting the privacy of your plaintexts before than you're still protecting the privacy, just as well, when you expand ciphertexts by appending a fixed pattern of bits. Of course the modified encryption scheme is less efficient than the original one—it is "wasting" 100 bits on every ciphertext—but that is not our present concern.

Despite the argument above, we *like* the notion of indistinguishability from random bits. There are several reasons for our liking it. First, we are going to see that indistinguishability from random bits implies left-or-right indistinguishability, and when your goal is to show that some particular scheme is secure there is no downside in selecting an unnecessarily strong notion of security. While there is no downside in such an application, there is a potential benefit: an encryption scheme whose output has been shown to be indistinguishable from random bits can be used in some contexts where the goal was not *not* message privacy. In particular, such an encryption scheme can be used as a pseudorandom generator. Next, when we go to prove security for practical encryption schemes like CTRC, CTR$, and CBC$, we find that not only do they achieve this stronger notion of security but that it is just as easy—indeed it is a bit easier—to directly demonstrate the stronger security property. Finally, we find indistinguishability from random bits to be the very simplest and easiest-to-understand notion of encryption-scheme security that has been offered.

Our definition will again speak of placing an adversary in one of two world, denoted world 0 and world 1. In both cases the adversary is given an oracle f. In world 1 the oracle works by encrypting whatever message M the adversary asks. The encryption is done using a key K that is randomly sampled from the underlying space of keys at the beginning of the experiment.. In world 0 the oracle works by returning a bunch of random bits. The number of random bits that are returned is the number of bits that the adversary would see were the message actually encrypted. The message M that the adversary asks of the oracle is used *only* to determine the length of the ciphertext output. The formal definition now follows.

Definition 4.20 Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, let $b \in \{0,1\}$, and let A be an algorithm that has access to an oracle. We consider the following experiments:

Experiment $\mathbf{Exmt}^{\text{rnd-cpa-1}}_{\mathcal{SE}}(A)$	Experiment $\mathbf{Exmt}^{\text{rnd-cpa-0}}_{\mathcal{SE}}(A)$
$K \stackrel{\$}{\leftarrow} \mathcal{K}$	$K \stackrel{\$}{\leftarrow} \mathcal{K}$
$b' \stackrel{\$}{\leftarrow} A^{\mathcal{E}_K(\cdot)}$	Run A^g
Return b'	When A asks an oracle query $g(X)$
	$C \stackrel{\$}{\leftarrow} \mathcal{E}_K(X)$
	If $C = \perp$ then answer $\perp$
	$R \stackrel{\$}{\leftarrow} \{0,1\}^{\lvert C \rvert}$
	Answer the query with R
	When A halts with bit b
	Return b'

The *RND-CPA advantage* of A is defined as

$$\mathbf{Adv}^{\text{rnd-cpa}}_{\mathcal{SE}}(A) \quad = \quad \Pr\left[\mathbf{Exmt}^{\text{ind-cpa-1}}_{\mathcal{SE}}(A) = 1\right] - \Pr\left[\mathbf{Exmt}^{\text{ind-cpa-0}}_{\mathcal{SE}}(A) = 1\right]$$

▮

For concision we will write as $\$(\cdot)$ the oracle whose behavior is described by the experiment $\mathbf{Exmt}^{\text{rnd-cpa-0}}_{\mathcal{SE}}(A)$. The expression RND-CPA advantage is then simplified to $\Pr[A^{\mathcal{E}_K(\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot)} \Rightarrow 1]$.

We now show that security in the RND-sense implies security in the IND-sense. As usual, we think in terms of the converse: insecurity in the IND-sense implies insecurity in the RND-sense. In other words, given an adversary A that does well at distinguishing a left encryption oracle from a right encryption oracle we will need to construct an adversary B that does well at distinguishing an encryption oracle from a source of random bits. The result is as follows.

Proposition 4.21 [RND-CPA $\Rightarrow$ IND-CPA] Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Let A be a (left-or-right distinguishing) adversary that runs in time at most t and asks at most q queries, these queries totaling at most μ bits. Then there exists a (real-or-random distinguishing) adversary B such that

$$\mathbf{Adv}^{\text{rnd-cpa}}_{\mathcal{SE}}(B) \quad \geq \quad \frac{1}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A)$$

where the running time of B is $t + O(\mu)$ and B makes at most q oracle queries and these queries total at most $\mu + m$ bits. ▮

Proof of Proposition 4.21: Adversary B^g is constructed from adversary A^f as follows.

Adversary B^g
    $b \xleftarrow{\$} \{0,1\}$
    Run adversary A^f
        When A^f makes an oracle query (M_0, M_1),
            answer the oracle query with $g(M_b)$
        When A^f halts, outputting a bit b'
            If $b = b'$ then return 1 else return 0

We observe that B runs in time $t + O(\mu)$, that B asks at most q queries, and that these queries total at most μ bits. To analyze the advantage of B in the RND-sense, suppose first that B's oracle g is an encryption oracle $\mathcal{E}_K(\cdot)$ for a random key $K \xleftarrow{\$} \mathcal{K}$. Then half the time B is providing A a perfect simulation of a left oracle, and half the time B is providing A a perfect simulation of a right oracle. Adversary B outputs 1 exactly when A correctly identifies which of the two kinds of oracles it has. By the definition and result of Section **??**, we thus have that

$$\begin{aligned}
\Pr[B^{\mathcal{E}_K(\cdot)} \Rightarrow 1] &= \Pr[\mathbf{Exmt}^{\text{ind1-cpa}}_{\mathcal{SE}}(A) = 1] && (4.5)\\
&= \frac{1}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) + \frac{1}{2} && (4.6)
\end{aligned}$$

On the other hand, suppose first that B's oracle g is realized as a $\$(\cdot)$-oracle. Because queries to f must have the same length, there is, information-theoretically, no correlation between oracle-responses provided to A and the contents of A's queries.

In particular, there is no correlation between oracle-responses provided to A and the bit b, and so

$$\Pr[B^{\$(\cdot)} \Rightarrow 1] \quad = \quad \frac{1}{2}$$

By the definition of RND-CPA advantage and the last two equations, we have that

$$\begin{aligned}
\mathbf{Adv}^{\text{rnd-cpa}}_{\mathcal{SE}}(B) &= \Pr[B^{\mathcal{E}_K(\cdot)} \Rightarrow 1] - \Pr[B^{\$(\cdot)} \Rightarrow 1] \\
&= \frac{1}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) + \frac{1}{2} - \frac{1}{2}
\end{aligned}$$

and so

$$\begin{aligned}
\mathbf{Adv}^{\text{rnd-cpa}}_{\mathcal{SE}}(B) &= \Pr[B^{\mathcal{E}_K(\cdot)} \Rightarrow 1] - \Pr[B^{\$(\cdot)} \Rightarrow 1] \\
&= \frac{1}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) + \frac{1}{2} - \frac{1}{2} \\
&= \frac{1}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A)
\end{aligned}$$

which completes the proposition. ▮

We emphasize that IND-CPA security does *not* imply RND-CPA security. The example given at the beginning of this section proves that. Namely, starting with *any* IND-CPA secure encryption scheme we showed how to modify it so that it will be just as IND-CPA secure, but completely RND-CPA insecure. This was done by tacking on a number of zero bits to the end of each encryption, so as to ensure that ciphertexts do *not* look random. It is easy to see that the modification does not adversely impact IND-CPA security.

4.9 Security of CTR modes

This section torn apart, in process of being replaced

Recall that the CTR (counter) mode of symmetric encryption comes in two variants: the randomized (stateless) version CTRC of Scheme 4.6, and the counter-based (stateful) mechanism CTR\$ of Scheme 4.7. 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 the random function family $\mathsf{Rand}(n)$ in the role of the underlying block cipher E—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 4.22 [Security of CTRC mode] Let E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding CTRC symmetric encryption scheme as described in Scheme 4.7. 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 where

$$\mathbf{Adv}_E^{\mathrm{prf}}(B) \quad \geq \quad \mathbf{Adv}_{\mathcal{SE}}^{\mathrm{ind\text{-}cpa}}(A)$$

(and so $\mathbf{Adv}_E^{\mathrm{prp}}(B) \geq \mathbf{Adv}_{\mathcal{SE}}^{\mathrm{ind\text{-}cpa}}(A) - 0.5\,\sigma^2/2^n$) and such that B runs in time at most $t' = t + O(q + n\sigma)$ and asks at most $q' = \sigma$ oracle queries. ▌

And for the randomized version:

Theorem 4.23 [Security of CTR\$ mode] Let E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding CTR\$ symmetric encryption scheme as described in Scheme 4.6. Let A be an adversary (for attacking the RND-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 where

$$\mathbf{Adv}_F^{\mathrm{prf}}(B) \quad \geq \quad \mathbf{Adv}_{\mathcal{SE}}^{\mathrm{rnd\text{-}cpa}}(A) - \frac{0.5\,\sigma^2}{2^n}$$

(and so $\mathbf{Adv}_E^{\mathrm{prp}}(B) \geq \mathbf{Adv}_{\mathcal{SE}}^{\mathrm{rnd\text{-}cpa}}(A) - \sigma^2/2^n$) and such that B runs in time at most $t' = t + O(q + n\sigma)$ and asks at most $q' = \sigma$ oracle queries. ▌

This kind of result is what this whole 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 block cipher, 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 block cipher, you are *assured* that nobody will break your encryption scheme. One cannot ask for more, since if one does not use a good block cipher, 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 4.24 Let us suppose that E is AES, so the the blocksize is $n = 128$. Suppose I want to encrypt $q = 2^{30}$ messages, each being one kilobyte (2^{13} bits) long. I am thus encrypting a total of one terabyte (2^{43} bits), which is to say $\sigma = 2^{36}$ blocks. Can I do this securely using counter-mode encryption, say CTR\$? Well, Theorem 4.23 says that if there is an adversary A that breaks CTR\$ with advantage δ then there is adversary B that breaks AES—breaks it in the sense of distinguishing a randomly select $\mathrm{AES}_K(\cdot)$ from a random permutation—that gets advantage $\delta - (2^{36})^2/2^{128} = \delta - 2^{-56} \approx \delta$ and where B uses comparable resources to that used by A. In other words, adversary B is doing essentially as well in attacking AES as adversary A was doing at attacking CTR\$[AES]. So if there exists a practical attack on CTR\$[AES] (breaking it in the sense of IND-CPA) then there exists a practical attack on AES (breaking it in the PRP-sense). Since we don't think there is any practical attack on AES (to break it in the PRP-sense) we must also believe that there is no practical attack on CBC\$[AES] when you one terabyte of data. Note that as the amount of data that is encrypted goes up, the value corresponding to 2^{-56} increases, until finally it becomes a large enough value to have to concern yourself with. By the time that you have encrypted 2^{64} blocks worth of data, all provable security has been lost. ▮

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.

We point out that our security theorems say that CTRC is IND-CPA secure and CTR\$ is, instead, RND-CPA security. Recall that the latter notion is stronger than IND-CPA security. As we have defined things, we can't possibly hope to prove RND-CPA security for CTRC mode, because the counter that is included with each ciphertext will *not* look random. If we had defined things a little differently, so that the IV was not regarded as part of the ciphertext, then we could indeed have had that the ciphertext was random-looking for CTRC.

4.9.1 Proof of Theorem 4.22

The paradigm we are going to follow is one we will use again and again. In order to analyze CTRC mode with a function family E, we will begin by focusing on the security of the construction with the random function family $\mathsf{Rand}(n)$. That is, we look at CTRC[$\mathsf{Rand}(n)$] and only when that is done concern ourselves with CTRC[E].

Focusing on CTRC[$\mathsf{Rand}(n)$] is definitely a thought experiment, insofar as no real implementation can use a random function in place of a real block cipher E_K, for even storing such a function would take an exorbitant amount of memory. But this analysis of the idealized scheme enables us to focus on any possible weaknesses of the CTR mode itself, as opposed to weaknesses arising from properties of the underlying block cipher. We will then can show that this idealized scheme is secure, and then we will show how that means that the original mode is good.

Lemma 4.25 Let A be an adversary (for attacking the IND-CPA security of $\mathcal{SE}[\mathsf{Rand}(n,\ell)]$). Then

$$\mathbf{Adv}^{\text{ind-cpa}}_{\text{CTRC}[\mathsf{Rand}(n)]}(A) \quad = \quad 0 \; .$$

∎

The lemma considers an arbitrary adversary. Let us say this adversary has time-complexity t, makes q queries to its lr-encryption oracle, these totaling μ bits. The lemma does not care about the values of t, q, an μ. (Recall, however, that CBCC "shuts up" after encrypting 2^n blocks, so that is effectively the limit in which the mode is useful.) The theorem says the adversary has *zero* advantage, meaning that the adversary is completely clueless if it is speaking to a left encryption oracle or a right encryption oracle. The fact that no restriction is made on t indicates that the result is information-theoretic: it holds regardless of how much computing time the adversary invests. Passing to the information-theoretic setting is almost always done in carrying out our analyses of block-cipher constructions.

Proof of Lemma 4.25: We must show that, regardless of A's behavior, $\Pr[A^{\text{Left}((,\cdot),\cdot)}{\Rightarrow}1] = \Pr[A^{\text{Right}((,\cdot),\cdot)}{\Rightarrow}1]$ where $\text{Left}((,\cdot),\cdot)$ is the oracle that, on query (L,R), CTRC-encrypts L using a random function $\rho \stackrel{\$}{\leftarrow} \mathsf{Rand}(n)$ and where $\text{Right}((,\cdot),\cdot)$ is the oracle that, on query (L,R), CTRC-encrypts R using a random function $\rho \stackrel{\$}{\leftarrow} \mathsf{Rand}(n)$.

To be finished later: either oracle, on input of an m-bit query, returns m random bits or returns $\perp$, the latter only when the number total query length makes that necessary. ∎

4.9.2 Proof of Theorem 4.23

Give a game playing argument

4.10 Security of CBC with a random IV

In this section we show that CBC encryption using a random IV, the algorithm we called CBC$[$E$], is secure as long as E is a secure block cipher. By "secure" we

```
Initialization:
    for X ∈ {0,1}^n do ρ(X) ← undef
    bad ← false

In response to an oracle query M_1 ··· M_m:
    Y_0 ←$ {0,1}^n
    for i ← 1 to m do
        X_i ← Y_{i−1} ⊕ M_i
        Y_i ←$ {0,1}^n
        if X_i ∈ Domain(ρ) then bad ← true , Y_i ← ρ(X_i)
        ρ(X_i) ← Y_i
    return Y_0 Y_1 ··· Y_m
```

Figure 4.6: Games used in the analysis of CBC\$. Game C is the game as written, while game R omits the shaded statement. The former provides an accurate simulation of a CBC encryption oracle, while the latter provides an accurate simulation of random-bit source $\$(\cdot)$.

---

mean the strongest notion we have talked about—indistinguishability from random bits. Remember that this notion of security implies all the other notions of security we have talked about, like left-or-right indistinguishability and semantic security.

We begin with the information-theoretic result in which the the underlying function family to CBC\$ is the set of random functions. The result is then as follows.

Theorem 4.26 [Security of CBC\$ using a random function] Let $n \geq 1$ be a number and let A be an adversary (for attacking CBC\$-encryption in the IND\$-CPA sense) that queries at most σ blocks. Then

$$\mathbf{Adv}^{\text{rnd-cpa}}_{\text{CBC\$}[\mathsf{Rand}(n)]}(A) \quad \leq \quad \frac{\sigma^2}{2^{n+1}}$$

∎

In other words, if the adversary is limited to querying σ blocks than its ability to distinguish between CBC-mode encryptions (where CBC uses a random IV and uses a random n-bit to n-bit function as though it were the underlying block cipher) and a bunch of random bits is limited to $\sigma^2/2^n$. Thus if $\sigma \ll 2^{n/2}$ the adversary just can't do a good job at this task. The proof is by another game-playing argument.

Proof: Consider the two games specified in Figure 4.6. Recall that Domain(ρ) denotes the set of all $X \in \{0,1\}^n$ such that $\rho(X) \neq$ `undef`. This set grows as the game executes and more and more queries are answered by the adversary.

The first game, which we call game C, is the pseudocode exactly as written. The game is easily seen to behave as a CBC-encryption oracle would if that oracle used a random function $\rho \in \mathsf{Rand}(n)$ and a random IV. Since the adversary's view is identical in those two cases, we know that

$$\Pr[\rho \xleftarrow{\$} \mathsf{Rand}(n) : A^{\mathrm{CBC\$}_\rho(\cdot)} \Rightarrow 1] \;=\; \Pr[A^{\text{Game C}} \Rightarrow 1] \tag{4.7}$$

The second game, which we call game R, is the identical pseudocode except for *omitting* the highlighted statement. This game, in response to any m-block query $M_1 \cdots M_m$, responds with $m+1$ random blocks (meaning a random string of length $(m+1)n$ bits). Thus

$$\Pr[A^{\$(\cdot)} \Rightarrow 1] \;=\; \Pr[A^{\text{Game R}} \Rightarrow 1] \tag{4.8}$$

where the behavior of the $\$(\cdot)$ oracle was given in the experiment used in the definition of IND\$-CPA advantage. Thus we have that

$$\begin{aligned}\mathbf{Adv}^{\text{rnd-cpa}}_{\mathrm{CBC\$}[\mathsf{Rand}(n)]}(A) &= \Pr[A^{\mathrm{CBC\$}_\rho(\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot)} \Rightarrow 1] \\ &= \Pr[A^{\text{Game C}} \Rightarrow 1] - \Pr[A^{\text{Game R}} \Rightarrow 1]\end{aligned}$$

Now we have set up games R and C so as to be identical apart from statements that immediately follow $bad \leftarrow \mathsf{true}$. In such a case the game-playing paradigm allows us to conclude that

$$\Pr[A^{\text{Game C}} \Rightarrow 1] - \Pr[A^{\text{Game R}} \Rightarrow 1] \;\leq\; \Pr[A^{\text{Game R}} \text{ sets } bad] \tag{4.9}$$

Our analysis, then has been reduced to bounding the probability that *bad* get set to true in game R. Keep clear that, because we are in game R, the highlighted statement of Figure 4.6 does not exist; go ahead and mentally (or physically!) cross it out.

To analyze the probability that *bad* gets set to true in game R we use the sum bound: we add up the probability that *bad* gets set to true during each of the at most σ times that line 14 is executed. Note that the first time that line 14 is executed Domain(ρ) will have not points in it; the next time that line 14 is executed Domain(ρ) will have one point in it; and, in general, the i^{th} time that line 14 gets executed Domain(ρ) will have at most $i-1$ points in it. We *claim* that if Domain(ρ) has i points in it than the probability that X_i will fall in Domain(ρ) will be at most $i/2^n$. We will prove this in a moment. For now, let us verify that, with this claim, we are done. For the sum bound will then tell us that the probability that *bad* ever gets set is at most $0/2^n + 1/2^n + \cdots + (\sigma-1)/2^n = (1+2+\cdots+(\sigma-1))/2^n = \sigma(\sigma-1)/2^{n+1} \leq \sigma^2/2^n$.

Now to see that the claim is true we first note that X_i is a random n-bit string and we are testing if it is present in a set Domain(ρ) of size i, so the probability that X_i will be in this set is exactly $i/2^n$ *if X_i is independent of* Domain(ρ). But *is* X_i independent of Domain(ρ) when we do this test at line 14? (That is to say, we are looking at the random variables X_i and Domain(ρ) at some particular, unnamed point in time.) Since line 15 adds points X_i to the Domain(ρ) it might at first *look* like these random variables are not going to be independent. But realize that the sequence of steps is like this (mentally unroll the loop, if that is helpful): (1) first a uniformly distributed Y_{i-1} value is selected (at line 10 or at line 13); (2) then X_i is defined by xoring Y_{i-1} with something that the adversary provided, M_i, creating a uniformly distributed X_i value; (3) now we test if X_i is in the set Domain(ρ); and, finally, (4) we grow Domain(ρ) by X_i. The point is that Domain(ρ) is grown by X_i only *after* the test, and Y_{i-1} does not influence Domain(ρ) prior to then. At the time of the test, the randomly chosen Y_{i-1} has not yet had a chance to influence Domain(ρ). This completes the proof. ▮

The theorem above addresses the maximal adversarial advantage when CBC\$ is done over a random n-bit to n-bit function. What if we use a random n-bit to n-bit permutation, instead? Applying the Switching Lemma (that is, Lemma 3.17) to the result above lets us bound this new advantage.

Corollary 4.27 [Security of CBC\$ using a random permutation] Let $n \geq 1$ be a number and let A be an adversary (for attacking CBC\$-encryption in the RND-CPA sense) that queries at most σ blocks. Then

$$\mathbf{Adv}^{\text{rnd-cpa}}_{\text{CBC\$}[\mathsf{Perm}(n)]}(A) \quad \leq \quad \frac{\sigma^2}{2^n}$$

▮

Proof: We have that

$$\begin{aligned}
\mathbf{Adv}^{\text{rnd-cpa}}_{\text{CBC\$}[\mathsf{Perm}(n)]}(A) &= \Pr[\pi \xleftarrow{\$} \mathsf{Perm}(n) :\ A^{\text{CBC}_\pi(\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot)} \Rightarrow 1] \\
&= \Big(\Pr[\pi \xleftarrow{\$} \mathsf{Perm}(n) :\ A^{\text{CBC}_\pi(\cdot)} \Rightarrow 1] - \Pr[\rho \xleftarrow{\$} \mathsf{Rand}(n) :\ A^{\text{CBC}_\rho(\cdot)} \Rightarrow 1]\Big) + \\
&\qquad \Big(\Pr[\rho \xleftarrow{\$} \mathsf{Rand}(n) :\ A^{\text{CBC}_\rho(\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot)} \Rightarrow 1]\Big) \\
&\leq \sigma^2/2^{n+1} + \Pr[\rho \xleftarrow{\$} \mathsf{Rand}(n) :\ A^{\text{CBC}_\rho(\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot)} \Rightarrow 1] \\
&\leq \sigma^2/2^{n+1} + \mathbf{Adv}^{\text{rnd-cpa}}_{\text{CBC\$}[\mathsf{Rand}(n)]}(A) \\
&\leq \sigma^2/2^{n+1} + \sigma^2/2^{n+1} \\
&= \sigma^2/2^n
\end{aligned}$$

The first line is the definition of RND-CPA security. The forth line is by the Switching Lemma: think of building from A an adversary B that runs A and answers oracle queries by carrying out the CBC\$ computation on its own, but using its oracle as the underlying block cipher. The second to last line is Theorem 4.28. ▮

After you gain some experience with this kind of argument you will begin to just "know" that you can trade off the $\mathsf{Perm}(n)$ with $\mathsf{Rand}(n)$ if you correspondingly increase the bound by $\sigma^2/2^{n+1}$.

Finally, we can look at what happens when we use a "real" block cipher within CBC mode.

Corollary 4.28 [Security of CBC$ using a block cipher] Let $n \geq 1$ and let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let A be an adversary (for attacking CBC$-encryption in the RND-CPA sense) that runs in time at most t and that queries at most σ blocks. Then there exists an adversary B (for attacking block cipher E in the PRP-CPA sense) where

$$\mathbf{Adv}_E^{\mathrm{prp}}(B) \quad \geq \quad \mathbf{Adv}_{\mathrm{CBC\$}[E]}^{\text{rnd-cpa}}(A) - \frac{\sigma^2}{2^n}$$

and where B makes at most σ queries and runs in time at most $t + O(n\sigma)$. ▮

Proof: Adversary B^g works by running adversary A^f, answering A's oracle queries by simulating a CBC$ oracle with CBC computed over the function g that is B's oracle. To describe this, let $\mathrm{CBC}_\rho^{C_0}(M_1 \cdots M_m)$ be the string $C_0 C_1 \cdots C_m$ where $C_i = \rho(M_{i-1} \oplus C_i)$. Let $\mathrm{CBC}_\rho^{\$}(\cdot)$ denote an oracle that, on input M, returns $\mathrm{CBC}_\rho^{\mathrm{IV}}$ for a random n-bit string IV. Adversary B is then the following:

Algorithm B^g:
  Run A^f
    When A makes an oracle call $f(M_1 \cdots M_m)$:
      Let IV $\stackrel{\$}{\leftarrow} \{0,1\}^n$
      Computer $\mathrm{CBC}_\rho^{\mathrm{IV}}(M_1 \cdots M_m)$ and return this to A
    When A halts, outputting a bit β
      Return β

Note that the running time of B is $t + O(n\sigma)$, as required. To see its advantage, first note that

$$\Pr[B^{E_K} \Rightarrow 1] \quad = \quad \Pr[A^{\mathrm{CBC}_{E_K}^{\$}(\cdot)} \Rightarrow 1]$$

and

$$\Pr[B^{\pi} \Rightarrow 1] \quad = \quad \Pr[A^{\mathrm{CBC}_{\pi}^{\$}(\cdot)} \Rightarrow 1]$$

Thus

$$\begin{aligned}
\mathbf{Adv}_E^{\mathrm{prp}}(B) &= \Pr[B^{E_K} \Rightarrow 1] - \Pr[B^{\pi} \Rightarrow 1] \\
&= \Pr[A^{\mathrm{CBC}_{E_K}^{\$}(\cdot)} \Rightarrow 1] - \Pr[A^{\mathrm{CBC}_{\pi}^{\$}(\cdot)} \Rightarrow 1] \\
&= \left(\Pr[A^{\mathrm{CBC}_{E_K}^{\$}(\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot)} \Rightarrow 1]\right) - \left(\Pr[A^{\mathrm{CBC}_{\pi}^{\$}(\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot)} \Rightarrow 1]\right) \\
&= \mathbf{Adv}_{\mathrm{CBC\$}[E]}^{\text{rnd-cpa}}(A) - \mathbf{Adv}_{\mathrm{CBC\$}[\mathsf{Perm}(n)]}^{\text{rnd-cpa}}(A) \\
&\geq \mathbf{Adv}_{\mathrm{CBC\$}[E]}^{\text{rnd-cpa}}(A) - \frac{\sigma^2}{2^n}
\end{aligned}$$

concluding the proof. ∎

We have so far shown that the RND-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.

Since we are giving an attack, it makes our result stronger if we go back to our original, *weaker* definition of security, IND-CPA. That is, when we are proving security we like to choose a strong definition—one that is as hard to achieve as possible. But when we are giving an attack we want to break the scheme as convincingly as possible, so we prefer a weak definition of security. The result is then as follows.

Proposition 4.29 Let $n \geq 1$, let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a function family, and let $\sigma \in [0 \mathinner{..} \sqrt{2}\, 2^{n/2} - 1]$. 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, $C_0C_1 \ldots C_\sigma$. If there is an $i, I \in [0 \mathinner{..} \sigma]$ 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 block cipher. 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 $C_0C_1\cdots C_\sigma$ uniform. So the probability in question is

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

Now the birthday bound (Appendix A, Theorem A.1) 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

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

completing the proof. ▮

4.11 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:

World 0: The adversary is provided the oracle $\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,0))$ as well as the oracle $\mathcal{D}_K(\cdot)$.

World 1: The adversary is provided the oracle $\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,1))$ as well as the oracle $\mathcal{D}_K(\cdot)$.

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 4.12 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 4.30 Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, let $b \in \{0, 1\}$, and let A be an algorithm that has access to two oracles and returns a bit. We consider the following experiment:

Experiment $\mathbf{Exmt}_{\mathcal{SE}}^{\text{ind-cca-}b}(A)$
  $K \xleftarrow{\$} \mathcal{K}$
  $b \leftarrow A^{\mathcal{E}_K(\text{LR}(\cdot,\cdot,b)),\, \mathcal{D}_K(\cdot)}$
  If A queried $\mathcal{D}_K(\cdot)$ on a ciphertext previously returned by $\mathcal{E}_K(\text{LR}(\cdot,\cdot,b))$
    then return 0
    else return b

The *ind-cca advantage* of A is defined as

$$\mathbf{Adv}_{\mathcal{SE}}^{\text{ind-cca}}(A) \;=\; \Pr\left[\mathbf{Exmt}_{\mathcal{SE}}^{\text{ind-cca-1}}(A) = 1\right] - \Pr\left[\mathbf{Exmt}_{\mathcal{SE}}^{\text{ind-cca-0}}(A) = 1\right] .$$

For any $t, q_e, \mu_e, q_d, \mu_d$ we define the *ind-cca advantage* of $\mathcal{SE}$ via

$$\mathbf{Adv}_{\mathcal{SE}}^{\text{ind-cca}}(t, q_e, \mu_e, q_d, \mu_d) \;=\; \max_A\{\mathbf{Adv}_{\mathcal{SE}}^{\text{ind-cca}}(A)\}$$

where the maximum is over all A having time-complexity t, making to the lr-encryption oracle at most q_e queries the sum of whose lengths is at most μ_e bits, and making to the decryption oracle at most q_d queries the sum of whose lengths is at most μ_d bits. ∎

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, and the time-complexity is the execution time of the entire experiment plus the size of the code of the adversary.

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.

4.12 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.

4.12.1 Attacks on the CTR schemes

Let $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^\ell$ 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 4.6. 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 ℓ-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 4.31 Let $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^\ell$ 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 4.6. Then

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

for $t = O(n + \ell)$ 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 4.31: We will present an adversary algorithm A, having time-complexity t, making 1 query to its lr-encryption oracle, this query being of length ℓ, making 1 query to its decryption oracle, this query being of length $n+\ell$, and having

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

The Proposition follows.

Remember that the lr-encryption oracle $\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))$ 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 $A^{\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))\,,\,\mathcal{D}_K(\cdot)}$
  $M_0 \leftarrow 0^\ell$; $M_1 \leftarrow 1^\ell$
  $\langle r, C\rangle \leftarrow \mathcal{E}_K(\mathrm{LR}(M_0, M_1, b))$
  $C' \leftarrow C \oplus 1^\ell$
  $M \leftarrow \mathcal{D}_K(\langle r, C'\rangle)$
  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

$$\Pr\left[\mathbf{Exmt}^{\text{ind-cca-1}}_{\mathcal{SE}}(A)=1\right] \;=\; 1$$
$$\Pr\left[\mathbf{Exmt}^{\text{ind-cca-0}}_{\mathcal{SE}}(A)=1\right] \;=\; 0\,.$$

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 ℓ bits, and just one decryption oracle query, whose length is $n+\ell$ bits (assuming an encoding of $\langle r, X\rangle$ as $n + |X|$-bits). So $\mathbf{Adv}^{\text{pr-cpa}}_{\mathcal{SE}}(t, 1, \ell, 1, n+\ell) = 1$.

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

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

Now notice that

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

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

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

Now notice that

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

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 4.7) is similar, and is left to the reader.

4.12.2 Attack on CBC$

Let E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the associated CBC$ symmetric encryption scheme as described in Scheme 4.4. 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 4.4.) 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 4.32 Let E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding CBC\$ encryption scheme as described in Scheme 4.4. 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 4.32: 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 $\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))$ 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 $A^{\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))\,,\,\mathcal{D}_K(\cdot)}$
    $M_0 \leftarrow 0^n$; $M_1 \leftarrow 1^n$
    $\langle \mathrm{IV}, C[1] \rangle \leftarrow \mathcal{E}_K(\mathrm{LR}(M_0, M_1, b))$
    $\mathrm{IV}' \leftarrow \mathrm{IV} \oplus 1^n$
    $M \leftarrow \mathcal{D}_K(\langle \mathrm{IV}', C[1] \rangle)$
    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

$$\Pr\left[\mathbf{Exmt}^{\text{ind-cca-1}}_{\mathcal{SE}}(A) = 1\right] \;=\; 1$$
$$\Pr\left[\mathbf{Exmt}^{\text{ind-cca-0}}_{\mathcal{SE}}(A) = 1\right] \;=\; 0 \,.$$

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

$$\begin{aligned} M &= \mathcal{D}_K(\langle \mathrm{IV}', C[1] \rangle) \\ &= E_K^{-1}(C[1]) \oplus \mathrm{IV}' \\ &= E_K^{-1}(E_K(\mathrm{IV} \oplus 0^n)) \oplus \mathrm{IV}' \\ &= (\mathrm{IV} \oplus 0^n) \oplus \mathrm{IV}'[0] \\ &= (\mathrm{IV} \oplus 0^n) \oplus (\mathrm{IV} \oplus 1^n) \\ &= 1^n \\ &= M_1 \ . \end{aligned}$$

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

4.13 Historical notes

The pioneering work on the theory of encryption is that of Goldwasser and Micali [18], with refinements by [28, 13]. 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 [3]. In particular Definition 4.1 and the concrete security

```
Algorithm E_K^(m)(M)
    Break M into n-bit blocks M[1], ..., M[m]
    For i ← 1 to m do
        C[i] ← E_K(M[i])
    EndFor
    C ← C[1] ··· C[m]
    Return C
```

```
Algorithm D_K^(m)(C)
    Break C into n-bit blocks C[1] ··· C[m]
    For i ← 1 to m do
        M[i] ← D_K(C[i])
        If M[i] = ⊥ then return ⊥
    EndFor
    M ← M[1] ··· M[m]
    Return M
```

Figure 4.7: Encryption scheme for Problem 4.2.

framework are from [3]. The analysis of the CTR mode encryption schemes, as given in Theorems 4.22 and 4.23, is also from [3]. The approach taken to the analysis of CBC mode is new.

4.14 Problems

Problem 4.1 Formalize a notion of security against key-recovery for symmetric encryption schemes, and prove an analog of Proposition 4.19.

Problem 4.2 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 length n. In order to be able to encrypt longer messages, says ones of mn bits for $m \geq 1$, 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 Figure 4.7.

(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 4.3 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 4.4 Our definition of RND-CPA security 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 $C \stackrel{\$}{\leftarrow} \mathcal{E}_K(M)$ but it might also get answered with random bits. Consider providing the adversary an additional, "reference" oracle that always encrypts the queried string. Define the corresponding advantage notion in the natural way: for an encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, let

$$\mathbf{Adv}^{\text{rnd2}-\text{cpa}}_{\mathcal{SE}}(A) \quad = \quad \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K} :\ A^{\mathcal{E}_K(\cdot),\, \mathcal{E}_K(\cdot)} \Rightarrow 1] - \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K} :\ A^{\mathcal{E}_K(\cdot),\, \$(\cdot)} \Rightarrow 1]$$

State and prove a theorem that shows that this notion of security is equivalent to our original RND-CPA notion.

Problem 4.5 *Write a problem on padding.*

Problem 4.6 *Write a problem on OFB mode.*

Problem 4.7 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 be simple, should "look like" CBC mode as much as possible, and it should coincide with CBC mode when the message being encrypted is a multiple of the blocklength. How would you prove your algorithm secure?

Problem 4.8 *Write a problem on whichkey-revealing.*