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 **??** looks at authenticity, and Chapter **??** looks at providing both together. Chapters **??** and **??** 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 $M \leftarrow \mathcal{D}_K(C)$.

The scheme is said to provide *correct decryption* if for any key $K \in \mathsf{Keys}(\mathcal{SE})$ and any message $M \in \{0,1\}^*$

$$\Pr\left[\, C = \perp \text{ OR } \mathcal{D}_K(C) = M \;:\; C \xleftarrow{\$} \mathcal{E}_K(M) \,\right] \;=\; 1\,. \blacksquare$$

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 can then be transmitted to the receiver.

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

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

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

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

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

Many encryption schemes restrict the set of strings that they are willing to encrypt. (For example, perhaps the algorithm can only encrypt plaintexts of length a positive multiple of some block length n, and can only encrypt plaintexts of length up to some maximum length.) These kinds of restrictions are captured by having the

encryption algorithm return the special symbol $\perp$ when fed a message not meeting the required restriction. In a stateless scheme, there is typically a set of strings, called the *plaintext space*, such that

$$\Pr\left[\, C \neq \perp \,:\, C \xleftarrow{\$} \mathcal{E}_K(M) \,\right] \;=\; 1$$

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 is fed to it.

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

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

4.2 Some symmetric encryption schemes

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

4.2.1 The one-time-pad encryption scheme

We begin with the classical one-time-pad. For a security analysis of this scheme, see Chapter **??**.

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

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

Here $X[i \,..\, j]$ denotes the i-th through j-th bit of the binary string X. By $\langle ctr, C \rangle$ we mean a string that encodes the number ctr and the string C. The most natural encoding is to encode ctr using some fixed number of bits, at least $\lg k$, and to prepend this to C. Conventions are established so that every string Y is regarded as encoding some ctr, C for some ctr, C. The encryption algorithm XORs the message bits with key bits, starting with the key bit indicated by one plus the current counter value. The counter is then incremented by the length of the message. Key bits are not reused, and thus if not enough key bits are available to encrypt a message, the encryption algorithm returns $\perp$. Note that the ciphertext returned includes the value of the counter. This is to enable decryption. (Recall that the decryption algorithm, as per Definition 4.1, must be stateless and deterministic, so we do not want it to have to maintain a counter as well.) ▌

4.2.2 Some modes of operation

The following schemes rely either on a family of permutations (i.e., a 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: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ 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 $K \xleftarrow{\$} \mathcal{K}$ and returns it. The encryption and decryption algorithms are depicted in Fig. 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

```
algorithm E_K(M)
    if (|M| mod n ≠ 0 or |M| = 0) then return ⊥
    Break M into n-bit blocks M[1] ··· M[m]
    for i ← 1 to m do
        C[i] ← E_K(M[i])
    C ← C[1] ··· C[m]
    return C
```

```
algorithm D_K(C)
    if (|C| mod n ≠ 0 or |C| = 0) then return ⊥
    Break C into n-bit blocks C[1] ··· C[m]
    for i ← 1 to m do
        M[i] ← E_K^{-1}(C[i])
    M ← M[1] ··· M[m]
    return M
```

Figure 4.1: ECB mode.

did not make any random choices. (That does not mean it is not, technically, a randomized algorithm; it is simply a randomized algorithm that happened not to make any random choices.) ▌

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

Scheme 4.4 [CBC$ mode] Let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ 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 Fig. 4.2. The IV ("initialization vector") is $C[0]$, which is chosen at random by the encryption algorithm. This choice is made independently each time the algorithm is invoked. ▌

For the following schemes it is useful to introduce some notation. 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.

Scheme 4.5 [CBCC mode] Let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Operating it in CBC mode with counter IV yields a stateful symmetric encryption

algorithm $\mathcal{E}_K(M)$
  if ($|M|$ mod $n \neq 0$ 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**
    $C[i] \leftarrow E_K(C[i-1] \oplus M[i])$
  $C \leftarrow C[1] \cdots C[m]$
  return $\langle \text{IV}, C \rangle$

---

algorithm $\mathcal{D}_K(\langle \text{IV}, C \rangle)$
  if ($|C|$ mod $n \neq 0$ 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**
    $M[i] \leftarrow E_K^{-1}(C[i]) \oplus C[i-1])$
  $M \leftarrow M[1] \cdots M[m]$
  return M

Figure 4.2: CBC$ mode.

---

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 encryptor maintains a counter *ctr* which is initially zero. The encryption and decryption algorithms are depicted in Fig. 4.3. The IV ("initialization vector") is $C[0]$, which is set to the current value of the counter. The counter is then incremented each time a message is encrypted. The counter is a static variable, meaning that its value is preserved across invocations of the encryption algorithm. ▌

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

Scheme 4.6 [CTR$ mode] Let F: $\mathcal{K} \times \{0,1\}^\ell \to \{0,1\}^L$ be a family of functions. (Possibly a block cipher, but not necessarily.) Then CTR mode over F with a random starting point is a probabilistic, stateless symmetric encryption scheme, $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key-generation algorithm simply returns a random key for E. The encryption and decryption algorithms are depicted in Fig. 4.4. The starting

```
algorithm E_K(M)
    static ctr ← 0
    if (|M| mod n ≠ 0 or |M| = 0) then return ⊥
    Break M into n-bit blocks M[1] ··· M[m]
    if ctr ≥ 2^n then return ⊥
    C[0] ← IV ← [ctr]_n
    for i ← 1 to m do
        C[i] ← E_K(C[i − 1] ⊕ M[i])
    C ← C[1] ··· C[m]
    ctr ← ctr + 1
    return ⟨IV, C⟩
```

```
algorithm D_K(⟨IV, C⟩)
    if (|C| mod n ≠ 0 or |C| = 0) then return ⊥
    Break C into n-bit blocks C[1] ··· C[m]
    if IV + m > 2^n then return ⊥
    C[0] ← IV
    for i ← 1 to m do
        M[i] ← E_K^{-1}(C[i]) ⊕ C[i − 1])
    M ← M[1] ··· M[m]
    return M
```

Figure 4.3: CBCC mode.

point R is used to define a sequence of values on which F_K is applied to produce a "pseudo one-time pad" to which the plaintext is XORed. The starting point R chosen by the encryption algorithm is a random ℓ-bit string. To add an ℓ-bit string R to an integer i—when we write $F_K(R+i)$—convert the ℓ-bit string R into an integer in the range $[0 \,..\, 2^\ell - 1]$ in the usual way, add this number to i, take the result modulo 2^ℓ, and then convert this back into an ℓ-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 F: $\mathcal{K} \times \{0,1\}^\ell \to \{0,1\}^L$ be a family of functions. (Possibly a block cipher, but not necessarily.) Operating it in CTR mode with a counter starting point is a stateful symmetric encryption scheme, $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, which we call CTRC. The key-generation algorithm simply returns a random key for F. The encryptor maintains a counter *ctr* which is initially zero. The encryption and decryption algorithms are depicted in Fig. 4.5. Position index *ctr* is not allowed to wrap around: the encryption algorithm returns $\perp$ if this would happen. The

algorithm $\mathcal{E}_K(M)$
$m \leftarrow \lceil |M|/L \rceil$
$R \xleftarrow{\$} \{0,1\}^\ell$
$Pad \leftarrow F_K(R+1) \,\|\, F_K(R+2) \,\|\, \cdots \,\|\, F_K(R+m)$
$Pad \leftarrow$ the first $|M|$ bits of Pad
$C' \leftarrow M \oplus Pad$
$C \leftarrow R \,\|\, C'$
return C

---

algorithm $\mathcal{D}_K(C)$
if $|C| < \ell$ **then return** $\perp$
Parse C into $R \,\|\, C'$ where $|R| = \ell$
$m \leftarrow \lceil |C'|/L \rceil$
$Pad \leftarrow F_K(R+1) \,\|\, F_K(R+2) \,\|\, \cdots \,\|\, F_K(R+m)$
$Pad \leftarrow$ the first $|C'|$ bits of Pad
$M \leftarrow C' \oplus Pad$
return M

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

---

position index is included in the ciphertext in order to enable decryption. The encryption algorithm updates the position index upon each invocation, and begins with this updated value the next time it is invoked. ▌

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

4.3 Issues in privacy

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

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

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

---

algorithm $\mathcal{D}_K(\langle i, C \rangle)$
  $m \leftarrow \lceil |C|/L \rceil$
  $Pad \leftarrow F_K(i+1) \,\|\, F_K(i+2) \,\|\, \cdots \,\|\, F_K(i+m)$
  $Pad \leftarrow$ the first $|C|$ bits of Pad
  $M \leftarrow Pad \oplus C$
  return M

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

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

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

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

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

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

For a concrete example, say I am communicating to my broker a message which is a sequence of "buy" or "sell" decisions for a pre-specified sequence of stocks. That is, we have certain stocks, numbered 1 through m, and bit i of the message is 1 if I want to buy stock i and 0 otherwise. The message is sent encrypted. But if the first

bit leaks, the adversary knows whether I want to buy or sell stock 1, which may be something I don't want to reveal. If the sum of the bits leaks, the adversary knows how many stocks I am buying.

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

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

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

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

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

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

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

As an example, consider the ECB encryption scheme of Scheme 4.3. 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 a missile command center has just two messages, 1^n for *fire* and 0^n for *don't fire*. It keeps sending data, but always one of these two. What happens? When the first ciphertext C_1 goes by, the adversary may not know what is the plaintext. But then, let us say it sees a missile taking off. Now, it knows the message M_1 underlying C_1 was 1^n. But then it can easily decrypt all subsequent messages, for if it sees a ciphertext C, the message is 1^n if $C = C_1$ and 0^n if $C \neq C_1$.

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

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

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

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

4.4 Indistinguishability under chosen-plaintext attack

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

Oracle $\mathcal{E}_K(\mathrm{LR}(M_0, M_1, b))$ // $b \in \{0,1\}$ and $M_0, M_1 \in \{0,1\}^*$
if $|M_0| \neq |M_1|$ **then return** $\perp$
$C \xleftarrow{\$} \mathcal{E}_K(M_b)$
return C

Figure 4.6: Left-or-right (lor) encryption oracle used to define IND-CPA security of encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$.

4.4.1 Definition

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

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

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

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

Let us now formalize this. We fix some encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. It could be either stateless or stateful. We consider an adversary A. It is a program which has access to an oracle to which it can provide as input any pair of equal-length messages. The oracle will return a ciphertext. We will consider two possible ways in which this ciphertext is computed by the oracle, corresponding to two possible "worlds" in which the adversary "lives". To do this, first define the *left-or-right encryption oracle* (abbreviated lr-encryption oracle) $\mathcal{E}_K(\mathrm{LR}(\cdot, \cdot, b))$ as shown in Fig. 4.6. 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) with $|M_0| = |M_1|$, the oracle computes

$C \xleftarrow{\$} \mathcal{E}_K(M_0)$, 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) with $|M_0| = |M_1|$ to its oracle, the oracle computes $C \xleftarrow{\$} \mathcal{E}_K(M_1)$, and returns C as the answer.

We also call the first world (or oracle) the "left" world (or oracle), and 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 oracle operates.

Think of the 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 $C \xleftarrow{\$} \mathcal{E}_K(M_b)$, it is implicit that the oracle picks its own random coins and uses them to compute ciphertext C.

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

If the given symmetric encryption scheme $\mathcal{SE}$ is stateful, the oracles, in either world, become stateful, too. (Think of a subroutine that maintains a "static" variable across successive calls.) An oracle begins with a state value initialized to a value specified by the encryption scheme. For example, in CTRC mode, the state is an integer *ctr* that is initialized to 0. Now, each time the oracle is invoked, it computes $\mathcal{E}_K(M_b)$ according to the specification of algorithm $\mathcal{E}$. The algorithm may, as a side-effect, update the state, and upon the next invocation of the oracle, the new state value will be used.

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

Definition 4.8 Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, and let A be an algorithm that has access to an oracle. We consider the following experiments:

Experiment $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{SE}}(A)$	Experiment $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{SE}}(A)$
$K \xleftarrow{\$} \mathcal{K}$	$K \xleftarrow{\$} \mathcal{K}$
$d \xleftarrow{\$} A^{\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,1))}$	$d \xleftarrow{\$} A^{\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,0))}$
Return d	Return d

The oracle used above is specified in Fig. 4.6. The *IND-CPA advantage* of A is defined as

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

As the above indicates, the choice of which world we are in is made just once, at the beginning, before the adversary starts to interact with the oracle. In 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.

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

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

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

We discuss some important conventions regarding the resources of adversary A. The *running time* of an adversary A is the worst case execution time of A over all possible coins of A and all conceivable oracle return values (including return values that could never arise in the experiments used to define the advantage). Oracle queries are understood to return a value in unit time, but it takes the adversary one unit of time to read any bit that it chooses to read. By convention, the running time of A also includes the size of the code of the adversary A, in some fixed RAM model of computation. This convention for measuring time complexity is the same as used in other parts of these notes, for all kinds of adversaries.

Other resource conventions are specific to the IND-CPA notion. When the adversary asks its left-or-right encryption oracle a query (M_0, M_1) we say that length of this query is $\max(|M_0|, |M_1|)$. (This will equal $|M_0|$ for any reasonable adversary since an oracle query with messages of different lengths results in the adversary being returned $\perp$, so we can assume no reasonable adversary makes such a query.) The total length of queries is the sum of the length of each query. We can measure query lengths in bits or in blocks, with block having some understood number of bits n.

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

4.4.2 Alternative interpretation

Let us move on to describe a somewhat different interpretation of 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.

Proposition 4.9 Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, and let A be an algorithm that has access to an oracle that takes input a pair of strings and returns a string. We consider the following experiment:

$$\begin{array}{l}\text{Experiment } \mathbf{Exp}^{\text{ind-cpa-cg}}_{\mathcal{SE}}(A)\\ \quad b \xleftarrow{\$} \{0,1\}\ ;\ K \xleftarrow{\$} \mathcal{K}\\ \quad b' \xleftarrow{\$} A^{\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))}\\ \quad \textbf{if}\ \ b = b'\ \textbf{then}\ \ \textbf{return}\ 1\ \textbf{else}\ \ \textbf{return}\ 0\end{array}$$

Then

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

In the above experiment, 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,

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

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

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

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

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

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

We began by expanding the quantity of interest via standard conditioning. The term of $1/2$ in the third line emerged because the choice of b is made at random. In the fourth line we noted that if we are asking whether $b = b'$ given that we know $b = 1$, it is the same as asking whether $b' = 1$ given $b = 1$, and analogously for $b = 0$. In the fifth line and sixth lines we just manipulated the probabilities and simplified. The next line is important; here we observed that the conditional probabilities in question are exactly the probabilities that A returns 1 in the experiments of Definition 4.8. ∎

4.4.3 Why is this a good definition?

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

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

In particular, here is one concern. In Section 4.3 we noted a number of security properties that are necessary but not sufficient for security. For example, it should be

computationally infeasible for an adversary to recover the key from a few plaintext-ciphertext pairs, or to recover a plaintext from a ciphertext.

A test of our definition is that it implies the necessary properties that we have discussed, and others. For example, a scheme that is secure in the IND-CPA sense of our definition should also be, automatically, secure against key-recovery or plaintext-recovery. Later, we will prove such things, and even stronger things. For now, let us continue to get a better sense of how to work with the definition by using it to show that certain schemes are insecure.

4.5 Example chosen-plaintext attacks

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

4.5.1 Attack on ECB

Let us fix a 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. We now construct such an adversary A. Remember that A is given a lr-encryption oracle $\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))$ 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 $A^{\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))}$
  $M_1 \leftarrow 0^{2n}$; $M_0 \leftarrow 0^n \,\|\, 1^n$
  $C[1]C[2] \leftarrow \mathcal{E}_K(\mathrm{LR}(M_0, M_1, b))$
  If $C[1] = C[2]$ **then return** 1 else return 0

Above, $X[i]$ denotes the i-th block of a string X, a block being a sequence of n bits. The adversary's single oracle query is the pair of messages M_0, M_1. Since each

of them is two blocks long, so is the ciphertext computed according to the ECB scheme. Now, we claim that

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

Why? 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.

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

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

There is an important feature of this attack that must be emphasized. Namely, ECB is an insecure encryption scheme *even if the underlying 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 Any deterministic, stateless schemes is insecure

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

Proposition 4.10 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 4.10: 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
  $C_1 \leftarrow \mathcal{E}_K(\mathrm{LR}(X, Y, b))$
  $C_2 \leftarrow \mathcal{E}_K(\mathrm{LR}(Y, Y, b))$
  If $C_1 = C_2$ **then return** 1 else return 0

Now, we claim that

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

Why? 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.

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

4.5.3 Attack on CBC encryption with counter IV

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

To justify our claim of insecurity, we present an adversary A. As usual it is given an lr-encryption oracle $\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))$ and wants to determine b. Our adversary works like this:

Adversary $A^{\mathcal{E}_K(\mathrm{LR}(\cdot,\cdot,b))}$
  $M_{0,1} \leftarrow 0^n$; $M_{1,1} \leftarrow 0^n$
  $M_{0,2} \leftarrow 0^n$; $M_{1,2} \leftarrow 0^{n-1}1$

$\langle \mathrm{IV}_1, C_1\rangle \xleftarrow{\$} \mathcal{E}_K(\mathrm{LR}(M_{0,1}, M_{1,1}, b))$
$\langle \mathrm{IV}_2, C_2\rangle \xleftarrow{\$} \mathcal{E}_K(\mathrm{LR}(M_{0,2}, M_{1,2}, b))$
If $C_1 = C_2$ **then return** 1 else return 0

We claim that

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

Why? First consider the case $b = 0$, meaning we are in world 0. In that case $\mathrm{IV}_1 = 0$ and $\mathrm{IV}_2 = 1$ 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 $\mathrm{IV}_1 = 0$ and $\mathrm{IV}_2 1 = 1$ and $C_1 = E_K(0)$ and $C_2 = E_K(0)$, so the defined experiment returns 1.

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

4.6 IND-CPA implies PR-CPA

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 4.11 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 $\mathbf{Exp}^{\text{pr-cpa}}_{\mathcal{SE}}(B)$
$K \xleftarrow{\$} \mathcal{K}$
$M' \xleftarrow{\$} \{0,1\}^m$
$C \xleftarrow{\$} \mathcal{E}_K(M')$
$M \xleftarrow{\$} B^{\mathcal{E}_K(\cdot)}(C)$
If $M = M'$ **then return** 1 else return 0

The *PR-CPA advantage* of B is defined as

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

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 in the PR-CPA sense.

Proposition 4.12 [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

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

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

Proof of Proposition 4.12: 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 $A^{\mathcal{E}_K(\text{LR}(\cdot,\cdot,b))}$
  $M_0 \xleftarrow{\$} \{0,1\}^m$; $M_1 \xleftarrow{\$} \{0,1\}^m$
  $C \leftarrow \mathcal{E}_K(\text{LR}(M_0, M_1, b))$
  Run adversary B on input C, replying to its oracle queries as follows
    When B makes an oracle query X to g do
      $Y \leftarrow \mathcal{E}_K(\text{LR}(X, X, b))$
      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

$$\Pr\left[\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{SE}}(A) = 1\right] \;\geq\; \mathbf{Adv}^{\text{pr-cpa}}_{\mathcal{SE}}(B) \tag{4.1}$$

$$\Pr\left[\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{SE}}(A) = 1\right] \;\leq\; 2^{-m}\ . \tag{4.2}$$

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

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

It remains to justify Equations (4.1) and (4.2).

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 (4.1) 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 (4.2) 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 4.12.

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.

4.7 Security of CTR modes

Recall that the CTR (counter) mode of operation of a family of functions comes in two variants: the randomized (stateless) version CTRC of Scheme 4.6, 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 $\mathsf{Func}(n)$ in the role of the underlying family of functions F—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 4.13 [Security of CTRC mode] Let $F\colon \mathcal{K} \times \{0,1\}^{\ell} \to \{0,1\}^{L}$ be a family of functions and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding CTRC symmetric encryption scheme as described in Scheme 4.7. 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 L-bit blocks. Then there exists an adversary B (attacking the PRF security of F) such that

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

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

Theorem 4.14 [Security of CTR$ mode] Let $F\colon \mathcal{K} \times \{0,1\}^{\ell} \to \{0,1\}^{L}$ 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 IND-CPA security of $\mathcal{SE}$) that runs in time at most t and asks at most q queries, these totaling at most σ L-bit blocks. Then there exists an adversary B (attacking the PRF security of F) such that

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

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

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

If we want to entrust our data to some encryption mechanism, we want to know that this encryption mechanism really provides privacy. If it is ill-designed, it may not. We saw this happen with ECB. Even if we used a secure 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 4.15 Let us suppose that F is the block cipher AES, so that $\ell = L = 128$. Suppose I want to encrypt $q = 2^{30}$ messages, each being one kilobyte (2^{13} bits) long. I am thus encrypting a total of 2^{43} bits, which is to say $\sigma = 2^{36}$ blocks. (This is about one terabyte). Can I do this securely using CTR\$? Let A be an adversary attacking the privacy of my encryption. Theorem 4.14 says that there exists a B satisfying the stated conditions. How large can $\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{AES}}(B)$ be? It makes $q = 2^{36}$ queries, and it is consistent with our state of knowledge of the security of AES to assume that such an adversary cannot do better than mount a birthday attack, meaning its advantage is no more than $q^2/2^{128}$. Then, the theorem tells us that

$$\mathbf{Adv}^{\text{rnd-cpa}}_{\mathcal{SE}}(A) \;\le\; \frac{\sigma^2}{2^{128}} + \frac{0.5\,\sigma^2}{2^{128}} \;=\; \frac{1.5\,2^{72}}{2^{128}} \;\le\; \frac{1}{2^{55}}\,.$$

This is a very small number indeed, saying that our encryption is secure, at least under the assumption that the best attack on the PRF security of AES is a birthday attack. Note however that if we encrypt 2^{64} blocks of data, 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.

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

4.7.1 Proof of Theorem 4.13

Yes, but it is not there now, and this creates a gap. As long as it is

algorithm $\mathcal{E}_g(M)$
  static $ctr \leftarrow 0$
  $m \leftarrow \lceil |M|/L \rceil$
  If $ctr + m \geq 2^\ell$ **then return** $\perp$
  $Pad \leftarrow g(ctr+1) \,\|\, g(ctr+2) \,\|\, \cdots \,\|\, g(ctr+m)$
  $Pad \leftarrow$ the first $|M|$ bits of Pad
  $C \leftarrow M \oplus Pad$
  $ctr \leftarrow ctr + m$
  return $\langle ctr - m, C \rangle$

algorithm $\mathcal{D}_g(\langle i, C \rangle)$
  $m \leftarrow \lceil |C|/L \rceil$
  $Pad \leftarrow g(i+1) \,\|\, g(i+2) \,\|\, \cdots \,\|\, g(i+m)$
  $Pad \leftarrow$ the first $|C|$ bits of Pad
  $M \leftarrow Pad \oplus C$
  return M

Figure 4.7: Version $\mathcal{SE}[G] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ of the CTRC scheme parameterized by a family of functions G.

The paradigm used is quite general in many of its aspects, and we will use it again, not only for encryption schemes, but for other kinds of schemes that are based on pseudorandom functions.

An important observation regarding the CTR scheme is that the encryption and decryption operations do not need direct access to the key K, but only access to a subroutine, or oracle, that implements the function F_K. This is important because one can consider what happens when F_K is replaced by some other function. To consider such replacements, we reformulate the scheme. We introduce a scheme that takes as a parameter any given family of functions G having domain $\{0,1\}^\ell$ and range $\{0,1\}^L$. As we will see later the cases of interest are $G = F$ and $G = \mathsf{Func}(\ell,L)$. Let us first however describe this parameterized scheme. In the rest of this proof, $\mathcal{SE}[G] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ denotes the symmetric encryption scheme defined as follows. The key generation algorithm simply returns a random instance of G, meaning that it picks a function $g \xleftarrow{\$} G$ from family G at random, and views g as the key. The encryption and decryption algorithms are shown in Fig. 4.7. (The scheme is stateful, with the encryptor maintaining a counter that is initially zero). As the description indicates, the scheme is exactly CTRC, except that function g is used in place of F_K. This seemingly cosmetic change of viewpoint is quite useful, as we will see.

We observe that the scheme in which we are interested, and which the theorem is about, is simply $\mathcal{SE}[F]$ where F is our given family of functions as per the theorem. Now, the proof breaks into two parts. The first step removes F from the picture,

and looks instead at an "idealized" version of the scheme. Namely we consider the scheme $\mathcal{SE}[\mathsf{Func}(\ell,L)]$. Here, a random function g of ℓ-bits to L-bits is being used where the original scheme would use F_K. We then assess an adversary's chance of breaking this idealized scheme. We argue that this chance is actually zero. This is the main lemma in the analysis.

This step is definitely a thought experiment. No real implementation can use a random function in place of F_K because even storing such a function takes 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 can show that this idealized scheme is secure, and that means that the mode itself is good.

It then remains to see how this "lifts" to a real world, in which we have no ideal random functions, but rather want to assess the security of the scheme $\mathcal{SE}[F]$ that uses the given family F. Here we exploit the notion of pseudorandomness to say that the chance of an adversary breaking the $\mathcal{SE}[F]$ can differ from its chance of breaking the ideal-world scheme $\mathcal{SE}[\mathsf{Func}(\ell,L)]$ by an amount not exceeding the probability of breaking the pseudorandomness of F using comparable resources.

Lemma 4.16 [Security of CTRC using a random function] Let A be any IND-CPA adversary attacking $\mathcal{SE}[\mathsf{Func}(\ell,L)]$, where the scheme is depicted in Fig. 4.7. Then

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}[\mathsf{Func}(\ell,L)]}(A) \;=\; 0 \;.$$

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 σ L-bit blocks. The lemma does not care about the values of t, q, or σ. (Recall, however, that after encrypting a total of 2^ℓ blocks, the encryption mechanism will "shut up" and be of no use.) It says the adversary has zero advantage, meaning no chance at all of breaking the scheme. 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.

Of course, this lemma refers to the idealized scheme, namely the one where the function g being used by the encryption algorithm is random. But remember that ECB was insecure even in this setting. (The attacks we provided for ECB work even if the underlying cipher E is $\mathsf{Perm}(n)$, the family of all permutations on n-bit strings.) So the statement is not content-free; it is saying something quite meaningful and important about the CTR mode. It is not true of all modes.

We postpone the proof of the lemma. Instead we will first see how to use it to conclude the proof of the theorem. The argument here is quite simple and generic.

The lemma tells us that the CTRC encryption scheme is (very!) secure when g is a random function. But we are interested in the case where g is is an instance of our given family F. So our worry is that the actual scheme $\mathcal{SE}[F]$ is insecure even though the idealized scheme $\mathcal{SE}[\mathsf{Func}(\ell,L)]$ is secure. In other words, we worry that

there might be an adversary having large IND-CPA advantage in attacking $\mathcal{SE}[F]$, even though we know that its advantage in attacking $\mathcal{SE}[\mathsf{Func}(\ell,L)]$ is zero. But we claim that this is not possible if F is a secure PRF. Intuitively, the existence of such an adversary indicates that F is not approximating $\mathsf{Func}(\ell,L)$ since there is some detectable event, namely the success probability of some adversary in a certain experiment, that happens with high probability when F is used and with low probability when $\mathsf{Func}(\ell,L)$ is used. To concretize this intuition, let A be a IND-CPA adversary attacking $\mathcal{SE}[F]$. We associate to A an adversary B that is given oracle access to a function $g\colon \{0,1\}^\ell \to \{0,1\}^L$ and is trying to determine which world it is in, where in world 0 g is a random instance of $\mathsf{Func}(\ell,L)$ and in world 1 g is a random instance of F. We suggest the following strategy to the adversary. It runs A, and replies to A's oracle queries in such a way that A is attacking $\mathcal{SE}[\mathsf{Func}(\ell,L)]$ in B's world 0, and A is attacking $\mathcal{SE}[F]$ in B's world 1. The reason it is possible for B to do this is that it can execute the encryption algorithm $\mathcal{E}_g(\cdot)$ of Fig. 4.7, which simply requires access to the function g. If the adversary A wins, meaning it correctly identifies the encryption oracle, B bets that g is an instance of F; otherwise, B bets that g is an instance of $\mathsf{Func}(\ell,L)$.

We stress the key point that makes this argument work. It is that the encryption function of the CTRC scheme invokes the function F_K purely as an oracle. If it had, instead, made some direct use of the key K, the paradigm above would not work. The full proof follows.

Proof of Theorem 4.13: Let A be any IND-CPA adversary attacking $\mathcal{SE} = (\mathcal{K},\mathcal{E},\mathcal{D})$. Assume A makes q oracle queries totaling μ bits, and has time-complexity t. There there is an adversary B such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;\;\leq\;\; 2\cdot \mathbf{Adv}^{\text{prf}}_{F}(B)\;. \tag{4.3}$$

Furthermore, B will make σ oracle queries and have time-complexity that of A plus $O(q+(\ell+L)\sigma)$. Now, the statement of Theorem 4.13 follows.

Remember that B takes an oracle $g\colon \{0,1\}^\ell \to \{0,1\}^L$. This oracle is either drawn at random from F or from $\mathsf{Func}(\ell,L)$ and B does not know which. To find out, B will use A, running it as a subroutine. But remember that A too gets an oracle, namely an lr-encryption oracle. From A's point of view, this oracle is simply a subroutine: A can write, at some location, a pair of messages, and is returned a response by some entity it calls its oracle. When B runs A as a subroutine, it is B that will "simulate" the lr-encryption oracle for A, meaning B will provide the responses to any oracle queries that A makes. Here is the description of B:

Adversary B^g
  $b \xleftarrow{\$} \{0,1\}$
  Run adversary A, replying to its oracle queries as follows
    When A makes an oracle query (M_0,M_1) do
      $C \xleftarrow{\$} \mathcal{E}_g(M_b)$

Return C to A as the answer
Until A stops and outputs a bit b'
If $b' = b$ then return 1 else return 0

Here $\mathcal{E}_g(\cdot)$ denotes the encryption function of the generalized CTRC scheme that we defined in Fig. 4.7. The crucial fact we are exploiting here is that this function can be implemented given an oracle for g. Adversary B itself picks the challenge bit b representing the choice of worlds for A, and then sees whether or not A succeeds in guessing the value of this bit. If it does, it bets that g is an instance of F, and otherwise it bets that g is an instance of $\mathsf{Func}(\ell,L)$. For the analysis, we claim that

$$\Pr\left[\mathbf{Exp}_F^{\text{prf-1}}(B) = 1\right] = \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}[F]}(A) \tag{4.4}$$

$$\Pr\left[\mathbf{Exp}_F^{\text{prf-0}}(B) = 1\right] = \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}[\mathsf{Func}(\ell,L)]}(A)\ . \tag{4.5}$$

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

$$\begin{aligned}
\mathbf{Adv}^{\text{prf}}_F(B) &= \Pr\left[\mathbf{Exp}_F^{\text{prf-1}}(B) = 1\right] - \Pr\left[\mathbf{Exp}_F^{\text{prf-0}}(B) = 1\right] \\
&= \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}[F]}(A) - \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}[\mathsf{Func}(\ell,L)]}(A) \qquad (4.6) \\
&= \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}[F]}(A)\ .
\end{aligned}$$

The last inequality was obtained by applying Lemma 4.16, which told us that the term $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}[\mathsf{Func}(\ell,L)]}(A)$ was simply zero. Re-arranging terms gives us Equation (4.3). Now let us check the resource usage. Each computation $\mathcal{E}_g(M_b)$ requires $|M_b|/L$ applications of g, and hence the total number of queries made by B to its oracle g is σ. The time-complexity of B equals that of A plus the overhead for answering the oracle queries. It remains to justify Equations (4.4) and (4.5).

Adversary B returns 1 when $b = b'$, meaning that IND-CPA adversary A correctly identified the world b in which it was placed, or, in the language of Section 4.4.2, made the "correct guess." The role played by B's world is simply to alter the encryption scheme for which this is true. When B is in world 1, the encryption scheme, from the point of view of A, is $\mathcal{SE}[F]$, and when B is in world 0, the encryption scheme, from the point of view of A, is $\mathcal{SE}[\mathsf{Func}(\ell,L)]$. Thus, using the notation from Section 4.4.2, we have

$$\begin{aligned}
\Pr\left[\mathbf{Exp}_F^{\text{prf-1}}(B) = 1\right] &= \Pr\left[\mathbf{Exp}^{\text{ind-cpa-cg}}_{\mathcal{SE}[F]}(A) = 1\right] \\
\Pr\left[\mathbf{Exp}_F^{\text{prf-0}}(B) = 1\right] &= \Pr\left[\mathbf{Exp}^{\text{ind-cpa-cg}}_{\mathcal{SE}[\mathsf{Func}(\ell,L)]}(A) = 1\right]\ .
\end{aligned}$$

To obtain Equations (4.4) and (4.5) we can now apply Proposition 4.9. ▌

For someone unused to PRF-based proofs of security the above may seem complex, but the underlying idea is actually very simple, and will be seen over and over again. It is simply that one can view the experiment of the IND-CPA adversary attacking the encryption scheme as information about the underlying function g being used, and if the adversary has more success in the case that g is an instance of F than that g is an instance of $\mathsf{Func}(\ell,L)$, then we have a distinguishing test between F and $\mathsf{Func}(\ell,L)$. Let us now prove the lemma about the security of the idealized CTRC scheme.

Proof of Lemma 4.16: The intuition is simple. When g is a random function, its value on successive counter values yields a one-time pad, a truly random and unpredictable sequence of bits. As long as the number of data bits encrypted does not exceed $L2^\ell$, we invoke g only on distinct values in the entire encryption process. And if an encryption would result in more queries than this, the algorithm simply shuts up, so we can ignore this. The outputs of g are thus random. Since the data is XORed to this sequence, the adversary gets no information whatsoever about it.

Now, we must make sure that this intuition carries through in our setting. Our lemma statement makes reference to our notions of security, so we must use the setup in Section 4.4. The adversary A has access to an lr-encryption oracle. Since the scheme we are considering is $\mathcal{SE}[\mathsf{Func}(\ell,L)]$, the oracle is $\mathcal{E}_g(\mathrm{LR}(\cdot,\cdot,b))$, where the function $\mathcal{E}_g$ was defined in Fig. 4.7, and g is a random instance of $\mathsf{Func}(\ell,L)$, meaning a random function.

The adversary makes some number q of oracle queries. Let $(M_{i,0}, M_{i,1})$ be the i-th query, and let m_i be the number of blocks in $M_{i,0}$. (We can assume this is the same as the number of blocks in $M_{i,1}$, since otherwise the lr-encryption oracle returns $\perp$). Let $M_{i,c}[j]$ be the value of the j-th ℓ-bit block of $M_{i,b}$ for $b \in \{0,1\}$. Let C'_i be the response returned by the oracle to query $(M_{i,0}, M_{i,1})$. It consists of a value that encodes the counter value, together with m_i blocks of ℓ bits each, $C_i[1] \ldots C_i[m_i]$. Pictorially:

$$\begin{array}{rcl}
M_{1,b} & = & M_{1,b}[1]M_{1,b}[1]\ldots M_{1,b}[m_1] \\
C_1 & = & \langle 0, C_1[1]\cdots C_1[m_1]\rangle \\
M_{2,b} & = & M_{2,b}[1]M_{2,b}[2]\ldots M_{2,b}[m_2] \\
C_2 & = & \langle m_1, C_2[1]\cdots C_2[m_2]\rangle \\
\vdots & & \vdots \\
M_{q,b} & = & M_{q,b}[1]M_{q,b}[2]\ldots M_{q,b}[m_q] \\
C_q & = & \langle m_1+\cdots+m_{q-1}, C_q[1]\cdots C_q[m_q]\rangle
\end{array}$$

What kind of distribution do the outputs received by A have? We claim that the $m_1 + \cdots + m_q$ values $C_i[j]$ ($i = 1,\ldots,q$ and $j = 1,\ldots,m_i$) are randomly and independently distributed, not only of each other, but of the queried messages and the bit b, and moreover this is true in both worlds. Why? Here is where we use a crucial property of the CTR mode, namely that it XORs data with the value of g

algorithm $\mathcal{E}_g(M)$
$m \leftarrow \lceil |M|/L \rceil$
$R \xleftarrow{\$} \{0,1\}^\ell$
$Pad \leftarrow g(R+1) \,\|\, g(R+2) \,\|\, \cdots \,\|\, g(R+m)$
$Pad \leftarrow$ the first $|M|$ bits of Pad
$C' \leftarrow M \oplus Pad$
$C \leftarrow R \,\|\, C'$
return C

---

algorithm $\mathcal{D}_g(C)$
if $|C| < \ell$ **then return** $\perp$
Parse C into $R \,\|\, C'$ where $|R| = \ell$
$m \leftarrow \lceil |C'|/L \rceil$
$Pad \leftarrow g(R+1) \,\|\, g(R+2) \,\|\, \cdots \,\|\, g(R+m)$
$Pad \leftarrow$ the first $|C'|$ bits of Pad
$M \leftarrow C' \oplus Pad$
return M

Figure 4.8: Version $\mathcal{SE}[G] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ of the CTR$ scheme parameterized by a family of functions G.

---

on a counter. We observe that according to the scheme

$$C_i[j] \;=\; g([m_1 + \cdots + m_{i-1} + j]_l) \;\oplus \begin{cases} M_{i,1}[j] & \text{if we are in world 1} \\ M_{i,0}[j] & \text{if we are in world 0.} \end{cases}$$

Now, we can finally see that the idea we started with is really the heart of it. The values on which g is being applied above are all distinct. So the outputs of g are all random and independent. It matters not, then, what we XOR these outputs with; what comes back is just random.

This tells us that any given output sequence from the oracle is equally likely in both worlds. Since the adversary determines its output bit based on this output sequence, its probability of returning 1 must be the same in both worlds,

$$\Pr\left[\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{SE}[\mathsf{Func}(\ell,L)]}(A) = 1\right] \;=\; \Pr\left[\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{SE}[\mathsf{Func}(\ell,L)]}(A) = 1\right] .$$

Hence A's IND-CPA advantage is zero. ▮

4.7.2 Proof of Theorem 4.14

The proof of Theorem 4.14 re-uses a lot of what we did for the proof of Theorem 4.13 above. We first look at the scheme when g is a random function, and then use the

pseudorandomness of the given family F to deduce the theorem. As before we associate to a family of functions G having domain $\{0,1\}^\ell$ and range $\{0,1\}^L$ a parameterized version of the CTR\$ scheme, $\mathcal{SE}[G] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key generation algorithm simply returns a random instance of G, meaning picks a function $g \xleftarrow{\$} G$ from family G at random, and views g as the key, and the encryption and decryption algorithms are shown in Fig. 4.8. Here is the main lemma.

Lemma 4.17 [Security of CTR\$ using a random function] Let A be any IND-CPA adversary attacking $\mathcal{SE}[\mathsf{Func}(\ell,L)]$, where the scheme is depicted in Fig. 4.8. Then

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}[\mathsf{Func}(\ell,L)]}(A) \;\leq\; \frac{0.5\,\sigma^2}{2^\ell}\;,$$

assuming A asks a number of queries whose total length is at most σ L-bit blocks. ∎

The proof of Theorem 4.14 given this lemma is easy at this point because it is almost identical to the above proof of Theorem 4.13, and it is the subject of Problem 4.3. We go on to prove Lemma 4.17.

Before we prove Lemma 4.17, we will analyze a certain probabilistic game. The problem we isolate here is purely probabilistic; it has nothing to do with encryption or even cryptography.

Lemma 4.18 Let ℓ, q be positive integers, and let $m_1, \ldots, m_q < 2^\ell$ also be positive integers. Suppose we pick q integers $r_1, \ldots, r_q$ from $[0..2^\ell - 1]$ uniformly and independently at random. We consider the following $m_1 + \cdots + m_q$ numbers:

$$\begin{array}{llll} r_1+1, & r_1+2, & \cdots, & r_1+m_1 \\ r_2+1, & r_2+2, & \cdots, & r_2+m_2 \\ \vdots & & & \vdots \\ r_q+1, & r_q+2, & \cdots, & r_q+m_q\;, \end{array}$$

where the addition is performed modulo 2^ℓ. We say that a *collision* occurs if some two (or more) numbers in the above table are equal. Then

$$\Pr[\mathsf{Col}] \;\leq\; \frac{(q-1)(m_1+\cdots+m_q)}{2^\ell}\;, \tag{4.7}$$

where Col denotes the event that a collision occurs.

Proof of Lemma 4.18: As with many of the probabilistic settings that arise in this area, this is a question about some kind of "balls thrown in bins" setting, related to the birthday problem studied in Appendix **??**. Indeed a reader may find it helpful to study that appendix first.

Think of having 2^ℓ bins, numbered $0, 1, \ldots, 2^\ell - 1$. We have q balls, numbered $1, \ldots, q$. For each ball we choose a random bin which we call r_i. We choose the bins one by one, so that we first choose r_1, then r_2, and so on. When we have thrown in the first ball, we have defined the first row of the above table, namely the values $r_1 + 1, \ldots, r_1 + m_1$. Then we pick the assignment r_2 of the bin for the second ball. This defines the second row of the table, namely the values $r_2 + 1, \ldots, r_2 + m_2$. A collision occurs if any value in the second row equals some value in the first row. We continue, up to the q-th ball, each time defining a row of the table, and are finally interested in the probability that a collision occurred somewhere in the process. To upper bound this, we want to write this probability in such a way that we can do the analysis step by step, meaning view it in terms of having thrown, and fixed, some number of balls, and seeing whether there is a collision when we throw in one more ball. To this end let Col_i denote the event that there is a collision somewhere in the first i rows of the table, for $i = 1, \ldots, q$. Let NoCol_i denote the event that there is no collision in the first i rows of the table, for $i = 1, \ldots, q$. Then by conditioning we have

$$\begin{aligned}
\Pr[\mathsf{Col}] &= \Pr[\mathsf{Col}_q] \\
&= \Pr[\mathsf{Col}_{q-1}] + \Pr[\mathsf{Col}_q \mid \mathsf{NoCol}_{q-1}] \cdot \Pr[\mathsf{NoCol}_{q-1}] \\
&\le \Pr[\mathsf{Col}_{q-1}] + \Pr[\mathsf{Col}_q \mid \mathsf{NoCol}_{q-1}] \\
&\le \ \vdots \\
&\le \Pr[\mathsf{Col}_1] + \sum_{i=2}^{q} \Pr[\mathsf{Col}_i \mid \mathsf{NoCol}_{i-1}] \\
&= \sum_{i=2}^{q} \Pr[\mathsf{Col}_i \mid \mathsf{NoCol}_{i-1}] \ .
\end{aligned}$$

Thus we need to upper bound the chance of a collision upon throwing the i-th ball, given that there was no collision created by the first $i - 1$ balls. Then we can sum up the quantities obtained and obtain our bound.

We claim that for any $i = 2, \ldots, q$ we have

$$\Pr[\mathsf{Col}_i \mid \mathsf{NoCol}_{i-1}] \ \le \ \frac{(i-1)m_i + m_{i-1} + \cdots + m_1}{2^\ell} \ . \tag{4.8}$$

Let us first see why this proves the lemma and then return to justify it. From the above and Equation (4.8) we have

$$\begin{aligned}
\Pr[\mathsf{Col}] &\le \sum_{i=2}^{q} \Pr[\mathsf{Col}_i \mid \mathsf{NoCol}_{i-1}] \\
&\le \sum_{i=2}^{q} \frac{(i-1)m_i + m_{i-1} + \cdots + m_1}{2^\ell}
\end{aligned}$$

$$= \frac{(q-1)(m_1 + \cdots + m_q)}{2^\ell} .$$

How did we do the last sum? The term m_i occurs with weight $i-1$ in the i-th term of the sum, and then with weight 1 in the j-th term of the sum for $j = i+1, \ldots, q$. So its total weight is $(i-1) + (q-i) = q-1$.

It remains to prove Equation (4.8). To get some intuition about it, begin with the cases $i = 1, 2$. When we throw in the first ball, the chance of a collision is zero, since there is no previous row with which to collide, so that is simple. When we throw in the second, what is the chance of a collision? The question is, what is the probability that one of the numbers $r_2 + 1, \ldots, r_2 + m_2$ defined by the second ball is equal to one of the numbers $r_1 + 1, \ldots, r_1 + m_1$ already in the table? View r_1 as fixed. Observe that a collision occurs if and only if $r_1 - m_2 + 1 \le r_2 \le r_1 + m_1 - 1$. So there are $(r_1 + m_1 - 1) - (r_1 - m_2 + 1) + 1 = m_1 + m_2 - 1$ choices of r_2 that could yield a collision. This means that $\Pr[\mathsf{Col}_2 \mid \mathsf{NoCol}_1] \le (m_2 + m_1 - 1)/2^\ell$.

We need to extend this argument as we throw in more balls. So now suppose $i-1$ balls have been thrown in, where $2 \le i \le q$, and suppose there is no collision in the first $i-1$ rows of the table. We throw in the i-th ball, and want to know what is the probability that a collision occurs. We are viewing the first $i-1$ rows of the table as fixed, so the question is just what is the probability that one of the numbers defined by r_i equals one of the numbers in the first $i-1$ rows of the table. A little thought shows that the worst case (meaning the case where the probability is the largest) is when the existing $i-1$ rows are well spread-out. We can upper bound the collision probability by reasoning just as above, except that there are $i-1$ different intervals to worry about rather than just one. The i-th row can intersect with the first row, or the second row, or the third, and so on, up to the $(i-1)$-th row. So we get

$$\begin{aligned} \Pr[\mathsf{Col}_i \mid \mathsf{NoCol}_{i-1}] &\le \frac{(m_i + m_1 - 1) + (m_i + m_2 - 1) + \cdots + (m_i + m_{i-1} - 1)}{2^\ell} \\ &= \frac{(i-1)m_i + m_{i-1} + \cdots + m_1 - (i-1)}{2^\ell} , \end{aligned}$$

and Equation (4.8) follows by just dropping the negative term in the above. ∎

Let us now extend the proof of Lemma 4.16 to prove Lemma 4.17.

Proof of Lemma 4.17: Recall that the idea of the proof of Lemma 4.16 was that when g is a random function, its value on successive counter values yields a one-time pad. This holds whenever g is applied on some set of distinct values. In the counter case, the inputs to g are always distinct. In the randomized case they may not be distinct. The approach is to consider the event that they are distinct, and say that in that case the adversary has no advantage; and on the other hand, while it may have a large advantage in the other case, that case does not happen often. We now flush all this out in more detail.

The adversary makes some number q of oracle queries. Let $(M_{i,0}, M_{i,1})$ be the i-th query, and let m_i be the number of blocks in $M_{i,0}$.(We can assume this is the same as the number of blocks in $M_{i,1}$, since otherwise the lr-encryption oracle returns $\perp$). Let $M_{i,b}[j]$ be the value of the j-th L-bit block of $M_{i,b}$ for $b \in \{0,1\}$. Let C'_i be the response returned by the oracle to query $(M_{i,0}, M_{i,1})$. It consists of the encoding of a number $r_i \in [0..2^\ell - 1]$ and a m_i-block message $C_i = C_i[1] \cdots C_i[m_i]$. Pictorially:

$$\begin{array}{rcl} M_{1,b} & = & M_{1,b}[1]M_{1,b}[1] \ldots M_{1,b}[m_1] \\ C_1 & = & \langle r_1, C_1[1] \cdots C_1[m_1] \rangle \\ \\ M_{2,b} & = & M_{2,b}[1]M_{2,b}[2] \cdots M_{2,b}[m_2] \\ C_2 & = & \langle r_2, C_2[1] \ldots C_2[m_2] \rangle \\ \\ \vdots & & \vdots \\ \\ M_{q,b} & = & M_{q,b}[1]M_{q,b}[2] \cdots M_{q,b}[m_q] \\ C_q & = & \langle r_q, C_q[1] \ldots C_q[m_q] \rangle \end{array}$$

Let NoCol be the event that the following $m_1 + \cdots + m_q$ values are all distinct:

$$\begin{array}{cccc} r_1 + 1, & r_1 + 2, & \cdots, & r_1 + m_1 \\ r_2 + 1, & r_2 + 2, & \cdots, & r_2 + m_2 \\ \vdots & & & \vdots \\ r_q + 1, & r_q + 2, & \cdots, & r_q + m_q \end{array}$$

Let Col be the complement of the event NoCol, meaning the event that the above table contains at least two values that are the same. It is useful for the analysis to introduce the following shorthand:

$$\begin{array}{rcl} \Pr_0[\cdot] & = & \text{The probability of event ``}\cdot\text{'' in world 0} \\ \Pr_0[\cdot] & = & \text{The probability of event ``}\cdot\text{'' in world 1 .} \end{array}$$

We will use the following three claims, which are proved later. The first claim says that the probability of a collision in the above table does not depend on which world we are in.

Claim 1: $\Pr_1[\mathsf{Col}] = \Pr_0[\mathsf{Col}]$. □

The second claim says that A has zero advantage in winning the left-or-right game in the case that no collisions occur in the table. Namely, its probability of outputting

one is identical in these two worlds under the assumption that no collisions have occurred in the values in the table.

Claim 2: $\Pr_0[A=1 \mid \mathsf{NoCol}] = \Pr_1[A=1 \mid \mathsf{NoCol}]$. □

We can say nothing about the advantage of A if a collision does occur in the table. It might be big. However, it will suffice to know that the probability of a collision is small. Since we already know that this probability is the same in both worlds (Claim 1) we bound it just in world 0:

Claim 3: $\Pr_0[\mathsf{Col}] \leq \dfrac{\sigma^2}{2^\ell}$. □

Let us see how these put together complete the proof of the lemma, and then go back and prove them.

Proof of Lemma given Claims: It is a simple conditioning argument:

$$\begin{aligned}
&\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}[\mathsf{Func}(\ell,L)]}(A) \\
&\quad = \Pr_1[A=1] - \Pr_0[A=1] \\
&\quad = \Pr_1[A=1 \mid \mathsf{Col}] \cdot \Pr_1[\mathsf{Col}] + \Pr_1[A=1 \mid \mathsf{NoCol}] \cdot \Pr_1[\mathsf{NoCol}] \\
&\qquad\qquad - \Pr_0[A=1 \mid \mathsf{Col}] \cdot \Pr_0[\mathsf{Col}] - \Pr_0[A=1 \mid \mathsf{NoCol}] \cdot \Pr_0[\mathsf{NoCol}] \\
&\quad = (\Pr_1[A=1 \mid \mathsf{Col}] - \Pr_0[A=1 \mid \mathsf{Col}]) \cdot \Pr_0[\mathsf{Col}] \\
&\quad \leq \Pr_0[\mathsf{Col}] \ .
\end{aligned}$$

The second-last step used Claims 1 and 2. In the last step we simply upper bounded the parenthesized expression by 1. Now apply Claim 3, and we are done. □

It remains to prove the three claims.

Proof of Claim 1: The event NoCol depends only on the random values $r_1, \ldots, r_q$ chosen by the encryption algorithm $\mathcal{E}_g(\cdot)$. These choices, however, are made in exactly the same way in both worlds. The difference in the two worlds is what message is encrypted, not how the random values are chosen. □

Proof of Claim 2: Given the event NoCol, we have that, in either game, the function g is evaluated at a new point each time it is invoked. Thus the output is randomly and uniformly distributed over $\{0,1\}^L$, independently of anything else. That means the reasoning from the counter-based scheme as given in Lemma 4.16 applies. Namely, we observe that according to the scheme

$$C_i[j] \quad = \quad g(r_i + j) \ \oplus \begin{cases} M_{i,1}[j] & \text{if we are in world 1} \\ M_{i,0}[j] & \text{if we are in world 0.} \end{cases}$$

Thus each cipher block is a message block XORed with a random value. A consequence of this is that each cipher block has a distribution that is independent of any previous cipher blocks and of the messages. □

algorithm $\mathcal{E}_g(M)$
  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**
    $C[i] \leftarrow g(C[i-1] \oplus M[i])$
  $C \leftarrow C[1] \cdots C[m]$
  return $\langle \text{IV}, C \rangle$

---

algorithm $\mathcal{D}_g(\langle \text{IV}, C \rangle)$
  return $\perp$

Figure 4.9: Version $\mathcal{SE}[G] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ of the CBC\$ scheme parameterized by a family of functions G.

Proof of Claim 3: This follows from Lemma 4.18. We simply note that $m_1 + \cdots + m_q = \sigma$. □

This concludes the proof. ▌

4.8 Security of CBC with a random IV

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

Theorem 4.19 [Security of CBC\$ 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 CBC\$ symmetric encryption scheme as described in Scheme 4.6. Let A be an adversary (for attacking the IND-CPA security of $\mathcal{SE}$) that runs in time at most t and asks at most q queries, these totaling at most σ n-bit blocks. Then there exists an adversary B (attacking the PRF security of E) such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;\leq\; \mathbf{Adv}^{\text{prf}}_{E}(B) + \frac{\sigma^2}{2^{n+1}} \;.$$

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

To prove this theorem, we proceed as before to introduce a scheme that takes as a parameter any given family of functions G having domain and range $\{0,1\}^n$. The cases of interest are $G = E$ and $G = \mathsf{Func}(n,n)$. The algorithms of the scheme are depicted in Fig. 4.9. Note that the decryption algorithm simply returns $\perp$, so

that this scheme does not have the correct decryption property. But one can still discuss its security, and it is important for us to do so. Now, the main result is the information-theoretic one in which the underlying function family is $\mathsf{Func}(n,n)$.

Lemma 4.20 [Security of CBC$ using a random function] Let A be any IND-CPA adversary attacking $\mathcal{SE}[\mathsf{Func}(n,n)]$, where the scheme is depicted in Fig. 4.9. Then

$$\mathbf{Adv}^{\text{ind-cpa}}_{\text{CBC\$}[\mathsf{Func}(n,n)]}(A) \;\leq\; \frac{\sigma^2}{2^{n+1}}\,,$$

assuming A asks a number of queries whose total length is at most σ n-bit blocks. ▮

Given this lemma, the proof of Theorem 4.19 follows in the usual way, so our main task is to prove the lemma. This is postponed for now.

4.9 Historical notes

The pioneering work on the theory of encryption is that of Goldwasser and Micali [3], with refinements by [4, 2]. This body of work is however in the asymmetric (i.e., public key) setting, and uses the asymptotic framework of polynomial-time adversaries and negligible success probabilities. The treatment of symmetric encryption we are using is from [1]. In particular Definition 4.1 and the concrete security framework are from [1]. The analysis of the CTR and CBC mode encryption schemes, as given in Theorems 4.13, 4.14 and 4.19 is also from [1]. The approach taken to the analysis of CBC mode is however new.

4.10 Problems

Problem 4.1 Formalize a notion of security against key-recovery for symmetric encryption schemes, and prove an analog of Proposition 4.12. ▮

Problem 4.2 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 4.3 Using the proof of Theorem 4.13 as a template, prove Theorem 4.14 assuming Lemma 4.17. ▮

Problem 4.4 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? ▮

Bibliography

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

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

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

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

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

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