Last Revised: February 2, 2000

Chapter 4

Symmetric Encryption

There are two "versions" of this chapter; I haven't yet integrated the material that has separately evolved. Everything needs to be harmonized. But perhaps it is fine for now; you get two chances to understand things!

4.1 Syntax of a Symmetric Encryption Scheme

Before we even attempt to describe when a symmetric encryption scheme is "good," let us just write down what a symmetric encryption scheme *is*. You might suspect that this is just a pair consisting of an encryption algorithm and a decryption algorithm. If so, you've forgotten that the ability to generate a key is an important part of the process, too!

Definition 4.1.1 A (probabilistic) symmetric encryption scheme Π is a triple of algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic algorithm which takes no input and produces a string ("the key") $a \in \mathsf{Key}$. We write $a \leftarrow \mathcal{K}$ to indicate sampling from that space (running that algorithm).

- $\mathcal{E}$ is a probabilistic algorithm that takes a string $a \in \mathsf{Key}$ and a string $M \in \{0,1\}^*$ and produces a string in $C \in \mathsf{Ciphertext} \cup \{\perp\}$. If presented with a string $M \in \mathsf{Plaintext}$ the algorithm must produce a string $C \in \mathsf{Ciphertext}$; and if presented with a string $M \notin \mathsf{Plaintext}$ the algorithm must return the distinguished symbol $\perp$.

- $\mathcal{D}$ is a deterministic algorithm that takes a key $a \in \mathsf{Key}$ and a string $C \in \{0,1\}^*$. It produces either a string $M \in \mathsf{Plaintext}$, or else the distinguished symbol $\perp$.

We require that for any $a \in \mathsf{Key}$ and any $M \in \mathsf{Plaintext}$, if $C \stackrel{R}{\leftarrow} \mathcal{E}_a(M)$ then $\mathcal{D}_a(C) = M$. ▮

The sets Key, Plaintext and Ciphertext are part of the definition of $\mathcal{K}$, $\mathcal{E}$, and $\mathcal{D}$; they are intrinsic elements of Π. The set Plaintext may be finite or infinite, but we always demand that if $M \in \mathsf{Plaintext}$ then $M' \in \mathsf{Plaintext}$ for any M' of the same length as M.

It is important $\mathcal{D}$ should be defined on any string, not just legitimate ciphertexts. We'd hate that the adversary could make $\mathcal{D}$ explode by feeding it an evil string!

Sometimes our encryption schemes will be *stateful* instead of probabilistic. This means that $\mathcal{E}$ has some hidden variable, the "state," which is initialized in some specified manner and which is updated with each application of $\mathcal{E}$. For stateful encryption schemes, we permit that $\mathcal{E}$ outputs $\perp$ even when provided $a \in \mathsf{Key}$ and $M \in \mathsf{Plaintext}$. This might happen when, for example, the encryption scheme has encrypted as much as it is willing to encrypt. The condition saying that it must be possible to decrypt after encryption only applies to "proper" ciphertexts—if $M \in \mathsf{Plaintext}$, $a \in \mathsf{Key}$, and $C \leftarrow \mathcal{E}_a(M)$, then if $C \neq \perp$ we must have that $\mathcal{D}_a(C) = M$.

4.2 Model

Definitions are hard, sometimes to the point that it is better to think about doing a *model* even before one begins to think about a definition proper. The model describes the adversary's capabilities but is silent about what it means for the adversary to win. In the context of symmetric encryption, a model captures the type of attack you imagine the adversary capable of mounting. In the formal definition, the model may completely vanish, and so it is probably not very useful to be formal about the model.

Here are four natural models —attack scenarios— for symmetric encryption.

(1) In a **known-distribution attack** one assumes that there is a certain distribution, $\mathcal{M}$, on plaintexts. The adversary knows $\mathcal{M}$. The adversary sees samples of ciphertexts where the plaintexts have been drawn according to $\mathcal{M}$. The model amounts to giving the adversary a but-

ton, and each time she pushes it a random $M \leftarrow \mathcal{M}$ is selected and the adversary is handed C, a random ciphertext for M.

(2) In a **known-plaintext attack** there is, once again, some distribution $\mathcal{M}$ on plaintexts. But now the adversary sees (M, C) pairs, where the plaintexts are generated by sampling in $\mathcal{M}$. The model amounts to giving the adversary a button, and each time she pushes it a random $M \leftarrow \mathcal{M}$ is selected and the adversary is presented with (M, C), where C is a random ciphertext for M.

(3) In an (adaptive) **chosen-plaintext attack** there is *no* distribution on the inputs. Instead, the adversary is given *complete* control on what is encrypted. The adversary makes up messages $M_1, M_2, \ldots$ and she is provided an encryption for each. The attack is adaptive in the sense that M_i might depend on the earlier ciphertexts the adversary has learned. The model amounts to providing the adversary with an encryption oracle. When presented with a message M the oracle responds with $C \leftarrow \mathcal{E}_a(M)$.

(4) In an (adaptive) **chosen-ciphertext attack** we give the adversary the ability of a chosen-plaintext attack, and also the ability to learn the plaintexts for ciphertexts of her choice. It's like having (perhaps temporarily) access to encryption/decryption equipment (for some particular hidden key). Requests to encipher M_i and decipher C_j might be arbitrarily interleaved. The model amounts to providing the adversary with an encryption oracle and a decryption oracle: the first, on input M, responds with a $C \in \mathcal{E}_a(M)$; the second, on input C, responds with $M = \mathcal{D}_a(C)$.

In coming up with concrete attacks on encryption schemes it is certainly important to have the adversary use the least capabilities possible. But in coming up with schemes one wants to assume the adversary has considerable power. So we tend to favor the strong notions. In addition, assuming at least a chosen-plaintext attack gives rise to a robust theory, which has not been demonstrated with weaker notions.

In what follows, we will assume a chosen-plaintext attack.

4.3 Definitional Ideas

Now we've described *what* the adversary can do, but we've said absolutely nothing about what she is trying to accomplish. We've described the model but have not yet begun to describe the definition. How can we get at a definition?

Perhaps the first thing that comes to mind is to try and capture the following intuition: that no adversary can recover the key a from using the encryption scheme $\mathcal{E}_a(\cdot)$. But this notion is no good. For example, suppose the encryption of M under 100-bit key a is M itself. The key a is supposed to be 100 random bits. Now the adversary has only a tiny tiny chance to recover the key a (just 2^{-100}), no matter how much work the adversary does and no matter how much enciphered text the adversary obtains. But the scheme is clearly worthless.

A second idea might be this: formalize that the encryption of a string should look like random garbage. We would be able to formalize this notion quite easily, drawing on what we've learned from formalizing pseudorandom generators. But the notion would not be good: it would be *too strong.* Suppose you have a good encryption scheme— the encryption of every message looks random. Now modify the encryption scheme so that the encryption of each message happens to start with the bit "0". The scheme ought still be a good encryption —it is doing a fine job of hiding the plaintext— but it would no longer come close to meeting the definition. Looking random is basically *irrelevant.*

A third idea might be the following: that an adversary can't recover M from $\mathcal{E}_a(M)$. Such intuition can certainly be lifted to a definition, but the problem is that the resulting definition would end up being too weak. Let us see why.

In an encryption scheme in which all we know is that A can't recover M from $\mathcal{E}_a(M)$, there is always the danger that the adversary can still recover a lot of *partial information* about M. For example, maybe the adversary can compute some particular bit of M. And maybe that bit of M is all the adversary cares about! For example, if the adversary knows that the plaintext includes in a certain position either the ASCII encoding of "$500,000" or "$800,000," then knowing a certain bit of plaintext is all that the adversary requires to distinguish the one plaintext from the other.

Partial information can take many different forms. For example, maybe the adversary can't recover M, nor any particular bit of M, but still she can recover the XOR of all of its bits. Once again, this might be exactly the information the adversary is interested in.

We want our encryption schemes not just to hide the plaintext, but to leak *no* partial information about it. This idea —properly formalized— is what we will call *semantic security.* It is one of the notions which we will define.

In this phrase "properly formalized" we hide a lot. One immediate concern is that there seems to be some information about M which is *inherently* leaked by $\mathcal{E}_a(M)$. One such piece of information is the existence of some plaintext M. A second piece of information is length of M. And a third piece of information is an encryption of the message M under the (secret) key a.

Ultimately, we deal with the first two concerns by saying that it is not the function of *encryption* to hide a message's existence or its length. It is not that these goals are not "legitimate" or even desirable; it is simply that we don't consider them to be the domain of techniques which we call by "encryption." We deal with the third objection by saying that $C \leftarrow \mathcal{E}_a(M)$ is really *not* information about M— we regard information about M as being realized by some *deterministic* function f applied to M.

Goldwasser and Micali (1984) first captured the intuition above (in the domain of asymmetric encryption), defining and showing how to achieve semantic security (in an asymptotic sense, under a complexity-theoretic assumption). As they say it, an encryption scheme is semantically secure if "whatever is efficiently computable about the plaintext given the ciphertext is also efficiently computable in the absence of the ciphertext."

4.4 Three Definitions

We went through three candidate definitions. We will now restate these.

I'm not going to "talk through" any of these definitions, but you should definitely think about them and make sure that they all make good sense to you!

FIND-THEN-GUESS. Here's a slightly different idea. We capture the following: that an adversary can't come up with a pair of equal-length messages (M_0, M_1) such that if she gets a random encryption of one of them, she'll know which one it is.

Definition 4.4.1 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let A be an adversary. Then

$$\begin{aligned}\mathbf{Adv}_{\Pi}^{\mathrm{fg}}(A) \ \stackrel{\mathrm{def}}{=}\ & 2 \cdot \Pr[a \leftarrow \mathcal{K};\ (M_0, M_1, s) \leftarrow A^{\mathcal{E}_a(\cdot)}(\mathsf{find});\ b \leftarrow \{0,1\}; \\ & C \leftarrow \mathcal{E}_a(M_b) :\ A^{\mathcal{E}_a(\cdot)}(\mathsf{guess}, s, C) = b] - 1\ . \quad \blacksquare\end{aligned}$$

It is understood that in the experiment above the adversary must output M_0 and M_1 of equal length. Some convention is established so that every output encodes a pair of equal length strings (and the string s).

When we speak of complexity, let communication complexity, μ, include the length of M_0.

REAL-OR-RANDOM. Here's a different idea. It seems to stray a bit from the modeling of a chosen-plaintext attack, but it will turn out to be irrelevant. We capture the following: that an adversary can't tell if she has a "real" encryption oracle or a "bogus" encryption oracle. The latter, on query M, ignores M (except for its length), flips $|M|$ random coins, and encrypts them.

Definition 4.4.2 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let A be an adversary. Then

$$\mathbf{Adv}_{\Pi}^{\mathrm{rr}}(A) \stackrel{\mathrm{def}}{=} \Pr[a \leftarrow \mathcal{K} :\ A^{\mathcal{E}_a(\cdot)} = 1] - \Pr[a \leftarrow \mathcal{K} :\ A^{\mathcal{E}_a(\$^{|\cdot|})} = 1]\ . \quad \blacksquare$$

It wouldn't have mattered if we had encrypted $0^{|\cdot|}$ instead of $\$^{|\cdot|}$.

SEMANTIC SECURITY. Here is a version of semantic security. This is an unpublished definition which builds on our CRYPTO '98 paper on strong notions for asymmetric encryption.

Definition 4.4.3 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let A be an adversary. Then

$$\begin{aligned}\mathbf{Adv}_{\Pi}^{\mathrm{sem}}(A) \stackrel{\mathrm{def}}{=} \ & \Pr[a \leftarrow \mathcal{K};\ (\mathcal{M}, f, s) \leftarrow A^{\mathcal{E}_a(\cdot)}(\mathsf{select})];\ M \leftarrow \mathcal{M}; \\ & \quad C \leftarrow \mathcal{E}_a(M) :\ A^{\mathcal{E}_a(\cdot)}(\mathsf{predict}, C, s) = f(M)] - \\ & \Pr[a \leftarrow \mathcal{K};\ (\mathcal{M}, f, s) \leftarrow A^{\mathcal{E}_a(\cdot)}(\mathsf{select})];\ M, M' \leftarrow \mathcal{M}; \\ & \quad C \leftarrow \mathcal{E}_a(M') :\ A^{\mathcal{E}_a(\cdot)}(\mathsf{predict}, C, s) = f(M)]\ . \quad \blacksquare\end{aligned}$$

In the definition above, when the adversary outputs $\mathcal{M}$ that means to output the encoding of a message-space sampling algorithm. We require that the sampling algorithm only output strings of some particular length. (Why?) When the adversary outputs f that means that she outputs the description of a program that computes the function f. We don't distinguish $\mathcal{M}$ as a distribution from $\mathcal{M}$ as an algorithm, nor do we distinguish f as a function from f as a program that computes it.

When we speak of complexity, let time, t, be understood to include the time for computing f (once) and for sampling from M (once). Let communication complexity, μ, include not only the sum of the lengths of the oracle queries, but also the length of (one) sample from $\mathcal{M}$. These conventions make theorem statements simpler.

4.5 Real-or-Random ⇒ Find-then-Guess

We are now going to start showing some relationships. Here is one of the simplest.

Proposition 4.5.1 Let $\Pi = (\mathcal{K}, \mathcal{E}, D)$ be a symmetric encryption scheme. Suppose there exists an adversary $A_{\rm fg}$ that $(t, q, \mu; \delta)$-breaks$^{\rm fg}$ Π. Then there exists an adversary $A_{\rm rr}$ that $(t', q', \mu'; \delta')$-breaks$^{\rm rr}$ Π, where

$$\begin{aligned} t' &= t + O(\mu), \\ q' &= q + 1, \\ \mu' &= \mu, \text{ and} \\ \delta' &= \delta/2 \,. \end{aligned}$$

Proof. We must construct the indicated adversary. First, ignore the bounds. One figures those out from looking at the efficiency of the constructed adversary.

How will the adversary $A_{\rm rr}$ work? Well, we can follow our noses!

Algorithm $A_{\rm rr}^g$:
  Run $A_{\rm fg}^h(\mathsf{find})$
    When $A_{\rm fg}^h$ asks its h-oracle some query x, answer using our own oracle, g.
  Eventually $A_{\rm fg}$ outputs some (M_0, M_1, s)
  Choose $b \leftarrow \{0, 1\}$.
  $C \leftarrow g(M_b)$.
  Run $A_{\rm fg}^h(\mathsf{guess}, C, s)$
    When $A_{\rm fg}^h$ asks its h-oracle some query x, answer using our own oracle, g.
  Eventually, $A_{\rm fg}$ outputs a bit, b'.
  If $b = b'$ then output 1 ("real") else output 0 ("random").

It is easy to check that $A_{\rm fg}$ runs in the claimed time bound and has the claimed bit complexity and query complexity. But we have to check that the advantage of $A_{\rm fg}$ is as indicated.

First think about what happens if we run $A_{\rm rr}^g$ where g is a realized by a "real" encryption oracle. Then $A_{\rm rr}$ perfectly simulates for $A_{\rm fg}$ the "native" environment which defines $A_{\rm fg}$'s success:

$$\begin{aligned} \Pr[a \leftarrow \mathcal{K} :\ A_{\rm rr}^{\mathcal{E}_a(\cdot)}] \;=\; & \Pr[a \leftarrow \mathcal{K};\ (M_0, M_1, s) \leftarrow A_{\rm fg}^{\mathcal{E}_a(\cdot)}(\mathsf{find});\ b \leftarrow \{0,1\}; \\ & C \leftarrow \mathcal{E}_a(M_b) :\ A_{\rm fg}^{\mathcal{E}_a(\cdot)}(\mathsf{guess}, C, s) = b] \end{aligned}$$

Now think about what happens if A_{rr}^{g} if g is realized by a "random-encrypting" encryption oracle. Then

$$\begin{aligned}
\Pr[a \leftarrow \mathcal{K}:\ A_{\mathrm{rr}}^{\mathcal{E}_a(\$^{|\cdot|})}] &= \Pr[a \leftarrow \mathcal{K};\ (M_0, M_1, s) \leftarrow A_{\mathrm{fg}}^{\mathcal{E}_a(\$^{|\cdot|})}(\mathsf{find}) \\
&\quad ;\ b \leftarrow \{0,1\};\ C \leftarrow \mathcal{E}_a(\$^{|M_b|}) : \\
&\quad A_{\mathrm{fg}}^{\mathcal{E}_a(\$^{|\cdot|})}(\mathsf{guess}, C, s) = b] \\
&= \frac{1}{2}\,.
\end{aligned}$$

Why is this 1/2? Because we've given A_{fg}, in her guess stage, absolutely no information about b. It doesn't matter how much time she spends or what strategy she uses, she can't predict a bit about which we've given her no information any better (or worse!) than 1/2.

But then

$$\begin{aligned}
\mathbf{Adv}_{A_{\mathrm{rr}}}^{\mathrm{rr}} &= \Pr[a \leftarrow \mathcal{K}:\ A_{\mathrm{rr}}^{\mathcal{E}_a(\cdot)}] - \Pr[a \leftarrow \mathcal{K}:\ A_{\mathrm{rr}}^{\mathcal{E}_a(\$^{|\cdot|})}] \\
&= \Pr[a \leftarrow \mathcal{K};\ (M_0, M_1, s) \leftarrow A_{\mathrm{fg}}^{\mathcal{E}_a(\cdot)}(\mathsf{find});\ b \leftarrow \{0,1\};\ C \leftarrow \mathcal{E}_a(M_b) : \\
&\quad A_{\mathrm{fg}}^{\mathcal{E}_a(\cdot)}(\mathsf{guess}, C, s) = b] - \frac{1}{2}
\end{aligned}$$

And so

$$\begin{aligned}
2 \cdot \mathbf{Adv}_{A_{\mathrm{rr}}}^{\mathrm{rr}} &= 2\Pr[a \leftarrow \mathcal{K};\ (M_0, M_1, s) \leftarrow A_{\mathrm{fg}}^{\mathcal{E}_a(\cdot)}(\mathsf{find});\ b \leftarrow \{0,1\};\ C \leftarrow \mathcal{E}_a(M_b) : \\
&\quad A_{\mathrm{fg}}^{\mathcal{E}_a(\cdot)}(\mathsf{guess}, C, s) = b] - 1 \\
&= \mathbf{Adv}_{A_{\mathrm{fg}}}^{\mathrm{fg}}
\end{aligned}$$

and so $\mathbf{Adv}_{A_{\mathrm{rr}}}^{\mathrm{rr}} = \frac{1}{2}\mathbf{Adv}_{A_{\mathrm{fg}}}^{\mathrm{fg}} \geq \delta/2$. ▮

4.6 Find-then-Guess ⇒ Real-or-Random

Theorem 4.6.1 Let $\Pi = (\mathcal{K}, \mathcal{E}, D)$ be a symmetric encryption scheme. Suppose there exists an adversary A_{rr} that $(t, q, \mu; \delta)$-breaks$^{\mathrm{rr}}$ Π. Then there exists an adversary A_{fg} that $(t', q', \mu'; \delta')$-breaks$^{\mathrm{fg}}$ Π, where

$$\begin{aligned}
t' &= t + O(\mu), \\
q' &= q,
\end{aligned}$$

$$\mu' = \mu, \text{ and}$$
$$\delta' = \delta/q\ .$$

Proof. This result seems harder to prove than the last one we did. We have an adversary A_{rr} that does a good job breaking Π in the real-or-random sense. We need to construct an adversary that does a good job breaking Π in the find-then-guess sense. This is a tall order—how is the adversary A_{fg} going to behave? The operating environment for a real-or-random adversary is very different from a find-then-guess adversary. The first has a single oracle, h, that we imagine to be a proper encryption oracle or else a bogus encryption oracle. The adversary experiments with it to try to figure out which. But a find-then-guess adversary always has a proper encryption oracle. It is trying to find a pair of messages that it can distinguish the encryptions of.

We will use a hybrid argument. Here is the adversary A_{fg}, parameterized by some number j.

Algorithm $A^g_{\mathrm{fg}}(\mathsf{find})$:
  Run A^h_{rr}
    When A^h_{rr} makes its i^{th} oracle query, x_i, answer as follows:
      If $i \le j$, then return $y_i \leftarrow g(\$^{|x_i|})$.
      If $i = j+1$, then we are done: save the current state in s and
        return $(M_0, M_1, s) \leftarrow (\$^{|x_i|}, x_i, s)$.

Algorithm $A^g_{\mathrm{fg}}(\mathsf{guess}, C, s)$
  Continue running A^h_{rr}, starting in state s.
    Oracle query x_{j+1} was just asked by A^h_{rr}. Answer it by C.
    When A^h_{rr} makes any additional query, x_i, answer with $y_i \leftarrow g(x_i)$.
  Eventually, A_{rr} outputs a bit, b'.
  Return b'.

That is, if A^h_{rr} thinks that h is a good encryption oracle (indicated by output of 1) then A_{fg} guesses that C is the encryption of $M_1 = x_j$; else A_{fg} guesses that C is the encryption of the random string M_0.

What is going on here? We know that if we run A_{rr} answering all of its questions with a real encryption oracle then we get a prediction from A_{rr} that is δ-different from what we get if we answer all of the questions with a random encryption oracle. Consider blending the one extreme into the other: answer the first $0, 1, 2, \ldots, j$ questions with a random encryption oracle, and the remaining with a real encryption oracle. See Figure 4.1.

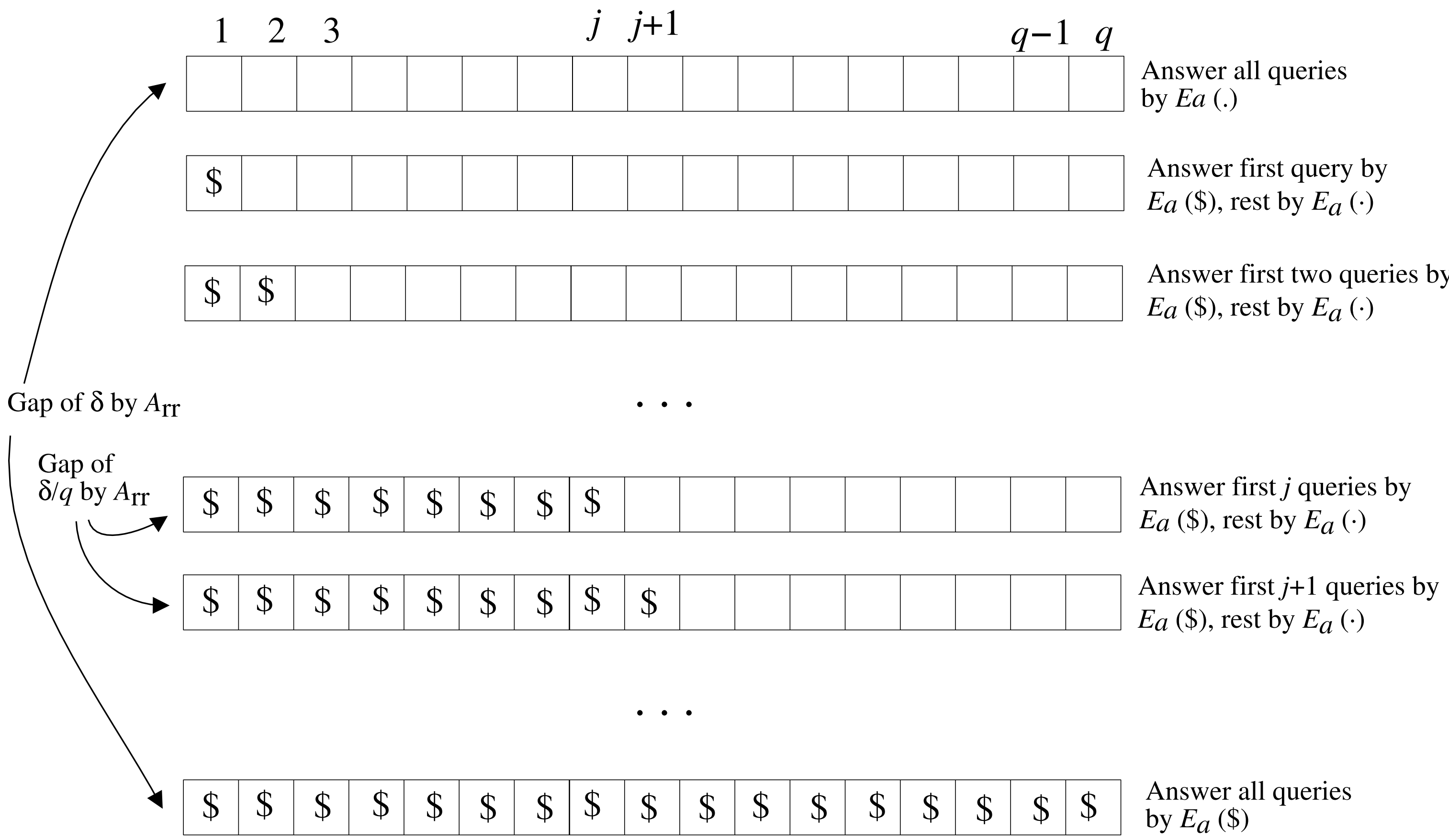

Figure 4.1: *Hybrid argument used for fg⇒rr.*

For $0 \leq i \leq q$, let P_i be the probability that A_{rr}^h outputs 1 if we answer the first i oracle queries with a $\mathcal{E}_a(\$^{|\cdot|})$-oracle, and the remaining queries with a $\mathcal{E}_a(\cdot)$-oracle. (In both cases, we initially choose a random $a \in \mathcal{K}$.) We know that

$$P_0 - P_q \geq \delta.$$

By the triangle inequality there exists a $0 \leq j < q$ such that

$$P_j - P_{j+1} \geq \delta/q.$$

Fix such a j and consider the adversary A_{fg} described earlier as specific to this particular j.

Now consider the environment in which A_{rr} finds itself when g is a (good) encryption oracle. Either C is a random encryption of M_0 (which happens when $b = 0$) or else C is a random encryption of M_1 (which happens when $b = 1$). In the former case we have provided for $A_{\rm rr}$ an environment which is identical to that which defines P_{j+1}: the first $j+1$ oracles queries have been answered with a random encryption oracle, the rest with a proper encryption oracle. In the latter case we have provided for $A_{\rm rr}$ an environment which is identical to that which defines P_j. We have, therefore, that $A_{\rm fg}$'s advantage δ' is $P_j - P_{j+1} \geq \delta/q$.

The other resource bounds for $A_{\rm fg}$ are easy to check. ▮

4.7 The Bound Can Not Be Improved

The above proof gives up a factor of q in the security. It is interesting to ask if the loss essential, or if it is just an artifact of the proof.

We argue that the loss is essential; there is no proof that gives a better bound. Thus the real-or-random definition is in some quantitative sense *stronger* than the find-then-guess definition. Let me give the proof idea.

We'd like to construct an encryption scheme with a "gap"—an encryption scheme that is q times "more secure" in the find-then-guess sense than in the real-or-random sense. We'll make a scheme which is *terrible* in the real-or-random sense—some rr-adversary has advantage which is nearly 1—but which for which no adversary has advantage much more than $1/q$ when measured in the find-then-guess sense. As usual, q is the number of queries the adversaries ask, and it is a parameter of our example.

Suppose you have a good encryption scheme, $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Think of this scheme as being very strong in the rr-sense (and therefore the fg-sense, too): the rr-advantage is ≈ 0 for any adversary that spends a reasonable amount of time. We're going to modify Π to make an encryption scheme $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ which is disproportionately weaker in the rr-sense than in the fg-sense. For simplicity, let's make Π' stateful. At the beginning, $\mathcal{E}'$ chooses a random (secret) $j \in [1..q]$. Then we define $\mathcal{E}'_a(x)$ to return a random $0 \,\|\, \mathcal{E}_a(x)$ for all queries except the j^{th}, and to return $1 \,\|\, x$ for the j^{th} query.

Now the scheme Π' is terrible in the rr-sense: a simple adversary can get advantage which is very nearly 1. That adversary repeatedly asks for the encryptions of some particular message M. If the response $1M$ is ever returned, the adversary can be pretty sure that this was a real encryption oracle. If no such response ever comes back from the encryption oracle, then the adversary can be sure that it was a random encryption oracle that she was provided.

The scheme $|\Pi'|$ may not be great in the fg-sense, but it is not nearly so bad as in the rr-sense; no adversary will have a chance more than $1/q$ plus the advantage of an adversary of comparable resources in attacking the scheme Π (which we think of as 0). Why? Consider a find-then-guess adversary attacking Π'. It will be seeing q encryptions, all together: $q-1$ directly asked of its encryption oracle, and one encryption of either M_0 or M_1, as asked by the adversary at the end of its find-stage. Now if the M_b query is the j^{th} query, then we will give up—just assume that the adversary wins. But if it is any of the other $q-1$ queries in which we encrypt the query x as $1x$, we claim that we have not helped the adversary to accomplish her goal (identifying which M_b the challenge C correspond to). For the only new information we have given to the adversary is $1x$ in response to one of the queries x. The adversary could have computed this on her own, and it has nothing to do with whether C is the encryption of M_0 or of M_1.

The above is of course not a formal argument. See [BDJR97] for a proper theorem statement and proof.

4.8 Real-or-Random $\Rightarrow$ Semantic

This proof won't be a hybrid argument. The proof method is, well, just slight of hand!

Theorem 4.8.1 Let $\Pi = (\mathcal{K}, \mathcal{E}, D)$ be a symmetric encryption scheme. Suppose there exists an adversary $A_{\rm sem}$ that $(t, q, \mu; \delta)$-breaks$^{\rm sem}$ Π. Then there exists an adversary $A_{\rm rr}$ that $(t', q', \mu'; \delta')$-breaks$^{\rm rr}$ Π, where

$$\begin{aligned} t' &= t + O(\mu), \\ q' &= q + 1, \\ \mu' &= \mu, \text{ and} \\ \delta' &= \delta/2 \,. \end{aligned}$$

Proof. Let

$$\begin{aligned} p \;=\; & \Pr[a \leftarrow \mathcal{K};\; (\mathcal{M}, f, s) \leftarrow A_{\rm sem}^{\mathcal{E}_a(\cdot)}(\mathsf{select});\; M \leftarrow \mathcal{M};\; C \leftarrow \mathcal{E}_a(M) : \\ & \quad A_{\rm sem}^{\mathcal{E}_a(\cdot)}(\mathsf{predict}, C, s) = f(x)] \end{aligned}$$

and let

$$\begin{aligned} q \;=\; & \Pr[a \leftarrow \mathcal{K};\; (\mathcal{M}, f, s) \leftarrow A_{\rm sem}^{\mathcal{E}_a(\cdot)}(\mathsf{select});\; M, M' \leftarrow \mathcal{M};\; C \leftarrow \mathcal{E}_a(M') : \\ & \quad A_{\rm sem}^{\mathcal{E}_a(\cdot)}(\mathsf{predict}, C, s) = f(x)] \,. \end{aligned}$$

Now define

$$p_* \;=\; \Pr[a \leftarrow \mathcal{K};\; (\mathcal{M}, f, s) \leftarrow A_{\text{sem}}^{\mathcal{E}_a(\$^{|\cdot|})}(\mathsf{select});\; M \leftarrow \mathcal{M};\; C \leftarrow \mathcal{E}_a(\$^{|M|}) :\; A_{\text{sem}}^{\mathcal{E}_a(\$^{|\cdot|})}(\mathsf{predict}, C, s) = f(x)]$$

and

$$q_* \;=\; \Pr[a \leftarrow \mathcal{K};\; (\mathcal{M}, f, s) \leftarrow A_{\text{sem}}^{\mathcal{E}_a(\$^{|\cdot|})}(\mathsf{select});\; M, M' \leftarrow \mathcal{M};\; C \leftarrow \mathcal{E}_a(\$^{|M'|}) :\; A_{\text{sem}}^{\mathcal{E}_a(\$^{|\cdot|})}(\mathsf{predict}, C, s) = f(x)]$$

That is, the experiment which defines p_* is obtained from the experiment which defines p by replacing all of the "real" encryption oracles by "random" encryption oracles, and similarly for q and q_*. Now we know that

$$p - q \;\geq\; \delta.$$

We also observe that

$$p_* \;=\; q_*.$$

This is because $\mathcal{M}$ only outputs strings of some one particular length, so if $M, M' \leftarrow \mathcal{M}$ then a random string of length $|M|$ is the same thing as a random string of length M'. We thus have that

$$(p - p_*) + (q_* - q) \;\geq\; \delta$$

and so

$$p - p_* \geq \delta/2 \qquad \text{or} \qquad q_* - q \geq \delta/2\ .$$

Either way, we can construct an adversary A_{rr} with the success probability and resource bounds asserted. The adversary is the obvious one. Suppose that $p - p_* \geq \delta/2$. Then consider the Boolean experiment by:

$$[(\mathcal{M}, f, s) \leftarrow A_{\text{sem}}^{g}(\mathsf{select});\; M \leftarrow \mathcal{M};\; C \leftarrow g(M) :\; A_{\text{sem}}^{g}(\mathsf{predict}, C, s) = f(x)]\ .$$

If $g = \mathcal{E}_a(\cdot)$ for a random $a \in \mathcal{K}$ then the experiment exactly coincides with the one defining p, while if $g = \mathcal{E}_a(\$^{|\cdot|})$ for a random $a \in \mathcal{K}$ then the experiment exactly coincides with the one defining p_*. This immediately gives us an adversary A_{rr}^{g}. The other case is analogous. ▮

4.9 Semantic $\Rightarrow$ Find-then-Guess

The last reduction that we shall do today is interesting insofar as we use very little of our available power.

Theorem 4.9.1 Let $\Pi = (\mathcal{K}, \mathcal{E}, D)$ be a symmetric encryption scheme. Suppose there exists an adversary $A_{\rm fg}$ that $(t, q, \mu; \delta)$-breaks$^{\rm fg}$ Π. Then there exists an adversary $A_{\rm sem}$ that $(t', q', \mu'; \delta')$-breaks$^{\rm sem}$ Π, where

$$\begin{aligned} t' &= t + O(\mu), \\ q' &= q, \\ \mu' &= \mu, \text{ and} \\ \delta' &= \delta\,. \end{aligned}$$

Proof. We have an adversary attacking the fg-security of Π. We want an sm-adversary. What should this adversary use as its information function f? And what should it use as its message space $\mathcal{M}$? We make very simple choices. Here is the adversary's definition:

Algorithm $A^g_{\rm sem}(\textsf{select})$:
  Let $f = \mathrm{id}$ be the identity function
  Run $A^h_{\rm fg}(\textsf{find})$
    When $A_{\rm fg}$ makes an oracle query, x, answer it with $y \leftarrow g(x)$.
    Eventually, $A_{\rm fg}$ halts, outputting a triple (M_0, M_1, s), where $|M_0| = |M_1|$.
  Let $\mathcal{M}$ be the message space with two points: M_0 and M_1, each with probability .5.
  Return (f, M, s).

Algorithm $A^g_{\rm sem}(\textsf{predict}, C, s)$
  Run $A^h_{\rm fg}(\textsf{guess}, C, s)$
    When $A_{\rm fg}$ makes an oracle query, x, answer it with $y \leftarrow g(x)$.
  Eventually, $A_{\rm fg}$ outputs a bit, b.
  If $b = 0$ then output M_0 else output M_1.

With g an encryption oracle $\mathcal{E}_a(\cdot)$, for a random $a \in \mathcal{K}$, we provide $A_{\rm fg}$ an environment which exactly mimics the one which defines adversarial success for an fg-adversary. Adversary $A_{\rm fg}$ answering 0 corresponds to its prediction that C is the encryption of M_0, while outputting 1 corresponds to the prediction that C is the encryption of M_1. Thus by outputting M_0 and M_1 in these two cases we will be correctly predicting the identity function of the plaintext whenever Afg makes a correct prediction. The result follows. ∎

Version 2

A symmetric encryption scheme (also called a private-key encryption scheme) enables parties in possession of a shared secret key to achieve the goal of data privacy. This is the canonical goal of cryptography. In the Introductory chapter we saw how Shannon addressed it with one-time pads. Now we will see the modern approach.

4.10 Symmetric Encryption schemes

The first step is to say what constitutes a description of a symmetric encryption scheme. This step is totally divorced from security considerations. We only want to say what are the algorithms involved.

Definition 4.10.1 A *symmetric encryption scheme* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms, as follows:

- The *key generation* algorithm $\mathcal{K}$ is a randomized algorithm that returns a key K; we write $K \stackrel{R}{\leftarrow} \mathcal{K}$
- The *encryption* algorithm $\mathcal{E}$ is a randomized algorithm that takes the key K and a *plaintext* M to return a *ciphertext* C; we write $C \stackrel{R}{\leftarrow} \mathcal{E}_K(M)$
- The *decryption* algorithm $\mathcal{D}$ is a deterministic algorithm that takes the key K and a ciphertext C to return the corresponding plaintext M; we write $M \leftarrow \mathcal{D}_K(C)$.

Associated to the scheme is a *message space* MsgSp from which M is allowed to be drawn. We require that for any integer i, either MsgSp contains all strings of length i, or none of them. We require that $\mathcal{D}_K(\mathcal{E}_K(M)) = M$ for all $M \in$ MsgSp. ▮

The key generation algorithm, as the definition indicates, is randomized. This means that it will internally flip coins and use these coins to determine its output. It takes no inputs; it simply flips coins internally and uses these to select a key K. Typically, the key is just a random string of some length. This length is 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; it is an assumption we make. Once in possession of a shared key, the parties can encrypt data for transmission. To encrypt message M, the sender (or encryptor) runs the encryption algorithm, meaning executes the operation $C \stackrel{R}{\leftarrow} \mathcal{E}_K(M)$. The encryption algorithm too is randomized. This means that it too might internally choose some coins and use those to determine its response to the input M. Now C is transmitted to the receiver. The latter can recover the message via $M \leftarrow \mathcal{D}_K(C)$. The last part of the above definition tells us that this operation does indeed return the same message that was initially encrypted.

We optionally allow the encryption procedure to depend on a global variable such as a counter, which is updated upon each invocation of the encryption procedure. Thus, the encryptor maintains a counter that is initialized in some pre-specified way. The ciphertext returned by the encryption algorithm is computed as a function of the counter (as well as the key and message). Also each time the encryption algorithm is invoked, it updates the counter, as a side-effect of its computation returning the ciphertext. (The receiver does not need to maintain a counter, and in particular decryption does not require access to any global variable or call for any synchronization between parties.) We call such an encryption scheme *stateful*. When there is no such counter or global variable, the scheme is *stateless*. In stateful schemes the encryption algorithm typically does not flip coins internally. (It is still OK to call it a randomized algorithm. It just happens to not make use of its given random number generator!) In stateless schemes, randomization is essential to security, as we will see.

The message here is assumed to come from some space of messages MsgSp associated to the scheme. Ideally, any string can be encrypted, so that MsgSp $= \{0,1\}^*$. But we may be interested in more restrictive schemes using which one can, for example, only encrypt messages of length some multiple of a block length, in which case the message space is set accordingly.

Example 4.10.2 Let $F\colon \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l$ be a block cipher. Operating it in ECB mode yields a stateless symmetric encryption scheme, $\mathcal{ECB}^F = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The algorithms are defined as follows. We let $\mathcal{K}$ be the algorithm which picks a k-bit key K by flipping k coins and returning their outcome. The message space is the set of all strings whose length is a multiple of l bits. The encryption algorithm below takes as input such a plaintext M, and the decryption algorithm below takes as input a ciphertext y.

Algorithm $\mathcal{E}_K(M)$ Divide M into l bit blocks, $M = x_1 \ldots x_n$ For $i = 1, \ldots, n$ do $y_i \leftarrow F_K(x_i)$ Return $y_1 \ldots y_n$	Algorithm $\mathcal{D}_K(y)$ Divide y into l bit blocks, $y = y_1 \ldots y_n$ For $i = 1, \ldots, n$ do $x_i \leftarrow F_K^{-1}(y_i)$ Return $x_1 \ldots x_n$

Notice that here the encryption algorithm did not make any random choices. That does not mean we are not allowed to call it a randomized algorithm; it is simply a randomized algorithm that happened to choose to not make random choices. ▮

Example 4.10.3 Let F: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l$ be a block cipher. Operating it in CBC mode with random IV yields a stateless symmetric encryption scheme, $\mathcal{CBC}^F = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The algorithms are defined as follows. We let $\mathcal{K}$ be the algorithm which picks a k-bit key K by flipping k coins and returning their outcome. The message space is the set of all strings whose length is a multiple of l bits. The encryption algorithm below takes as input such a plaintext M, and the decryption algorithm below takes as input a ciphertext y.

Algorithm $\mathcal{E}_K(M)$ Divide M into l bit blocks, $M = x_1 \ldots x_n$ $y_0 \stackrel{R}{\leftarrow} \{0,1\}^l$ For $i = 1, \ldots, n$ do $y_i \leftarrow F_K(y_{i-1} \oplus x_i)$ Return $y_0 y_1 \ldots y_n$	Algorithm $\mathcal{D}_K(y)$ Divide y into l bit blocks, $y = y_0 y_1 \ldots y_n$ For $i = 1, \ldots, n$ do $x_i \leftarrow F_K^{-1}(y_i) \oplus y_{i-1}$ Return $x_1 \ldots x_n$

Here r is chosen at random by the encryption algorithm and used as the IV in CBC encryption of the message M. ▮

Example 4.10.4 Let F: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^L$ be a family of functions. (Not necessarily a family of permutations.) Operating it in CTR mode with starting point chosen at random anew for each message yields a stateless symmetric encryption scheme, $\mathcal{R}\text{-}\mathcal{CTR}^F = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The algorithms are defined as follows. We let $\mathcal{K}$ be the algorithm which picks a k-bit key K by flipping k coins and returning their outcome. The message space is the set of all strings whose length is a multiple of L bits. The encryption algorithm below takes as input such a plaintext M, and the decryption algorithm below takes as input a ciphertext y. Below we let $\langle j \rangle$ denote the binary representation of integer $j \in \{0, 1, \ldots, 2^l - 1\}$ as an l-bit string.

Algorithm $\mathcal{E}_K(M)$	Algorithm $\mathcal{D}_K(y)$
Divide M into L bit blocks, $M = x_1 \ldots x_n$	Let $\langle r \rangle$ be the first l bits of y
$r \stackrel{R}{\leftarrow} \{0, 1, \ldots, 2^l - 1\}$	Divide the rest of y into L bit blocks, $y_1 \ldots y_n$
For $i = 1, \ldots, n$ do $y_i \leftarrow F_K(\langle r + i \rangle) \oplus x_i$	For $i = 1, \ldots, n$ do $x_i \leftarrow F_K(\langle r + i \rangle) \oplus y_i$
Return $\langle r \rangle y_1 \ldots y_n$	Return $x_1 \ldots x_n$

The random value chosen by the encryption algorithm is an integer in the range $0, \ldots, 2^l - 1$. It is used to define a sequence of values on which F_K is applied to produce a "pseudo one-time pad" to which the data is XORed.

Example 4.10.5 Let F: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^L$ be a family of functions. (Not necessarily a family of permutations.) Operating it in CTR mode with counter yields a stateful symmetric encryption scheme, $\mathcal{C}\text{-}\mathcal{CTR}^F = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The algorithms are defined as follows. We let $\mathcal{K}$ be the algorithm which picks a k-bit key K by flipping k coins and returning their outcome. The message space is the set of all strings whose length is a multiple of L bits. The encryption algorithm below takes as input such a plaintext M, and the decryption algorithm below takes as input a ciphertext y. The state is a counter c. This is an integer whose value is initially 0. It is stored by the sender. Below we let $\langle j \rangle$ denote the binary representation of integer $j \in \{0, 1, \ldots, 2^l - 1\}$ as an l-bit string.

Algorithm $\mathcal{E}_K(M)$	Algorithm $\mathcal{D}_K(y)$
Divide M into L bit blocks, $M = x_1 \ldots x_n$	Let $\langle c \rangle$ be the first l bits of y
For $i = 1, \ldots, n$ do $y_i \leftarrow F_K(\langle c + i \rangle) \oplus x_i$	Divide the rest of y into L bit blocks, $y_1 \ldots y_n$
Return $\langle c \rangle y_1 \ldots y_n$	For $i = 1, \ldots, n$ do $x_i \leftarrow F_K(\langle c + i \rangle) \oplus y_i$
$c \leftarrow c + n$	Return $x_1 \ldots x_n$

For security, the counter should not be allowed to wrap around. Namely, the encryption algorithm should only be invoked on a message M if $c + n < 2^l$ where c is the current value of the counter. However, that restriction is not part of the scheme description. Remember that at present we are only describing the schemes, not discussing their security. ▮

Most of our treatment of symmetric encryption, including the development of a notion of security, holds regardless of whether the scheme is stateless or stateful; we just talk of a symmetric encryption scheme.

4.11 A notion of security

We will first try to build up some intuition about what properties an encryption scheme should have to call it "secure", and then distill a formal definition of security.

4.11.1 Issues in security

Recall the setting. We have fixed a particular symmetric encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Two parties share a key K for this scheme and the adversary does not a priori know K. 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 question is: what exactly does "glean" mean? What tasks, were the adversary to accomplish them, would make us declare the scheme insecure? And, correspondingly, what tasks, were the adversary unable to accomplish them, would make us declare the scheme secure?

It is much easier to think about *insecurity* than security, because we can certainly identify adversary actions that indubitably imply the scheme is insecure. For example, if the adversary can, from a few ciphertexts, derive the underlying key K, it can later decrypt anything it sees, so if the scheme allowed easy key recovery from a few ciphertexts it is definitely insecure. Yet, an absence of easy key recovery is certainly not enough for the scheme to be secure; maybe the adversary can do something else.

A first cut at a notion of security is to say that what we want is: given ciphertext C, the adversary can't easily recover the plaintext M. But actually, this isn't good enough. The reason is that the adversary might be able to figure out *partial information* about M. For example, even though she 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 l, 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 definitely 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 was 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. It is an important principle of our approach that the encryption scheme should yield security no matter what is the format of the data. That is, we don't want people to have to worry about how they format their data: it should be secure regardless.

In other words, as designers of security protocols, we cannot 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. And we want schemes that are secure in the strongest possible natural sense.

So 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. Of course, our setting will be "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.

As an example, consider the ECB encryption scheme of Example 4.10.2. Given the ciphertext, can an eavesdropping adversary figure out the message? Hard to see how, since it does not know K, and if F is a "good" block cipher, then it ought to have a hard time inverting F_K without knowledge of the underlying key. Nonetheless this is not a good scheme. Consider just the case $n = 1$ of a single block message. Suppose I have just two messages, 0^l for "buy" and 1^l for "sell." I keep sending data, but always one of these two. What happens? The adversary sees which are the same. That is, it might see that the first two are the same and equal to the third, etc.

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

This has a somewhat 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 is thought of as a code, a fixed mapping of plaintexts to ciphertexts. This is no longer true. A single plaintext will have many possible ciphertexts. Yet, it should be possible to decrypt! How can we set this up? We have already seen, above, many examples: the modes of operation.

4.11.2 A notion of security: Real-or-Random

The idea is that an encryption scheme is secure if an adversary (not in possession of the secret key) can't tell the encryption of a message from the encryption of a random string of the same length. Namely, consider the following experiment. Let M be a message, and I give it to the adversary. Now I give the adversary either $\mathcal{E}_K(M)$ or $\mathcal{E}_K(X)$ for a random X, the choice of which being made at random, and ask it to guess which of these two things I did. If it has a tough time telling, I want to say the scheme is secure.

We actually want to encrypt not one message, but a whole sequence of them, so this idea must be extended. We want that an adversary should not be able to tell whether it is looking at ciphertexts of a known sequence of messages, or ciphertexts of a sequence of random messages of the same lengths. Furthermore we want this to be true not only when the adversary knows the "test" messages in question, but when it actually chooses them. That is, we allow a *chosen plaintext attack*. An adversary can choose the sequence of messages on which the real-or-random test is executed, and then hope to win the test.

We can set this up via a "two worlds" type idea. In each world the adversary is provided with a particular box, and it has to tell which box it is talking to.

Let us fix a specific encryption scheme Π. (It could be either stateless or stateful). Let $\mathcal{K}, \mathcal{E}$ be the key generation and encryption algorithms of this scheme, respectively. (The decryption algorithm is not relevant to security.) We consider an adversary A. It is a program which has access to an oracle $\mathcal{O}$, to which it can provide as input any message M in the message space MsgSp of the encryption scheme. 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".

What does it mean that $\mathcal{O}$ is an oracle? Think of it as a subroutine to which A has access. A can make an oracle query M by writing M in some special, specified location in memory, and, in one step, the answer is returned. A has no control on how the answer is computed, nor can A even see the working of the oracle, which will typically depend on secret information that A is not given. A just has an interface to this subroutine; the ability to call it as a black-box, and get back an answer.

World 0: The oracle $\mathcal{O}$, given message M, ignores M except to determine its length. It then flips coins to get a string X of length $|M|$, and runs the encryption algorithm to compute $C \stackrel{R}{\leftarrow} \mathcal{E}_K(X)$. It returns C as the answer.

World 1: The oracle $\mathcal{O}$, given M, runs the encryption algorithm on M, namely $C \stackrel{R}{\leftarrow} \mathcal{E}_K(M)$. It returns C as the answer.

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

First assume the given symmetric encryption scheme Π is stateless. The oracle $\mathcal{O}$ in world 0 is probabilistic, for several reasons. First, it determines a string X at random, making it probabilistic right there. But most importantly, it calls the encryption algorithm. Recall that this algorithm is probabilistic. Above, when we say $C \stackrel{R}{\leftarrow} \mathcal{E}_K(X)$, it is implicit that $\mathcal{E}$ picks its own random coins implicitly and uses them to compute C. So that introduces a second level of probabilism in the oracle responses. Similarly, the oracle $\mathcal{O}$ in world 1 is probabilistic, because it also invokes the encryption function, and the latter might, each time it is called, make new random choices.

The choices of the encryption function are somewhat "under the rug" here, but should not be forgotten; they are central to the meaningfulness of the notion, as also the security of the schemes.

If the given symmetric encryption scheme Π was stateful, the oracles, in either world, become stateful too. (Think of a subroutine that maintains a global variable across calls to the subroutine.) Take world 0 first. The oracle $\mathcal{O}$ begins with a state value initialized according to the specification of the scheme. For example, for CTR mode with a counter, it is a counter c set to 0. Now, each time the oracle is invoked, it computes $\mathcal{E}_K(M)$ according to the specification of algorithm $\mathcal{E}$. This algorithm will, as a side-effect, update the counter, and upon the next invocation of the oracle, the new counter value will be used. Similarly in world 1.

We clarify that the choice of which world we are in is made once, a priori, and then the adversary executes. In world 0, *all* messages sent to the oracle are answered by the oracle encrypting a random string, and in world 1, all messages are answered by encrypting the given message. The choice of which does not flip-flop from message to message; it is made once and then remains the same for all messages.

The adversary queries makes some number of queries to its oracle, and then output a bit. This bit has some probability of equaling one. The probability is over the choice of the key K, any random choices made by the oracle, and any other random choices made by the adversary in its computation. We look at this probability in each of the two worlds as the basis for the definition.

Definition 4.11.1 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, and let A be an adversary that has access to an oracle $\mathcal{O}$ and outputs a bit. Let

$$\begin{aligned}\mathbf{Succ}^{\text{se-ror}}_{\text{Real}}(\Pi, A) &= \Pr\left[K \stackrel{R}{\leftarrow} \mathcal{K} \,:\, A^{\mathcal{E}_K(\cdot)} = 1 \right] \\ \mathbf{Succ}^{\text{se-ror}}_{\text{Rand}}(\Pi, A) &= \Pr\left[K \stackrel{R}{\leftarrow} \mathcal{K} \,:\, A^{\mathcal{E}_K(\$^{|\cdot|})} = 1 \right] .\end{aligned}$$

We call the difference between them the "advantage" of A:

$$\mathbf{Adv}^{\text{se-ror}}(\Pi, A) = \mathbf{Succ}^{\text{se-ror}}_{\text{Real}}(\Pi, A) - \mathbf{Succ}^{\text{se-ror}}_{\text{Rand}}(\Pi, A) .$$

Finally for any t, q, μ we let

$$\mathbf{Adv}^{\text{se-ror}}(\Pi; t, q, \mu) = \max_A \{ \mathbf{Adv}^{\text{se-ror}}(\Pi, A) \}$$

where the maximum is over all A running in time t, making at most q oracle queries, these totaling at most μ bits. ▮

Above $\mathbf{Succ}^{\text{se-ror}}_{\text{Real}}(\Pi, A)$ is the probability the adversary outputs 1 in world 1. The fact that it is world 1 is captured by the choice made for the oracle $\mathcal{O}$. It is exactly the encryption function, taking a message M and returning $\mathcal{E}_K(M)$. Correspondingly $\mathbf{Succ}^{\text{se-ror}}_{\text{Rand}}(\Pi, A)$ is the probability the adversary outputs 1 in world 0. The fact that it is world 0 is captured by setting the oracle $\mathcal{O}$ to the function $\mathcal{E}_K(\$^{|\cdot|})$. This oracle, given an input M, returns $\mathcal{E}_K(\$^{|M|})$. That is, it flips $|M|$ many coins to determine a string X and returns $\mathcal{E}_K(X)$. It is understood that the adversary only makes oracle queries which are messages in the allowed space MsgSp of messages.

If $\mathbf{Adv}^{\text{se-ror}}(\Pi, A)$ is small, it means that A is outputting 1 just 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) then the adversary A is doing well, meaning our scheme Π is not secure.

For symmetric encryption scheme Π to be secure, the advantage of an adversary must be small, no matter what strategy the adversary tries. However, we expect that the advantage grows as the adversary invests more effort in the process. To capture this we define the insecurity function $\mathbf{Adv}^{\text{se-ror}}(\Pi; \cdot, \cdot, \cdot)$ as above. This is a function associated to any symmetric encryption scheme Π, which becomes fixed once we fix the scheme. The resources of the adversary we have chosen to use in the parameterization are three. First, its running time, which by convention, as usual, is on some fixed model of computation, and includes the size of the code describing the adversary as a program. Second, the number of oracle queries, or the number of messages the adversary asks of its oracle. These messages may have different lengths, and our third parameter is the sum of all these lengths, denoted μ.

Let us now try to exercise this definition a bit.

4.11.3 Exploring the notion: Implications

Sometimes, people are confused because in the above experiment, the messages are all in the clear. Isn't encryption about hiding the plaintext? Well as we said in our discussion above, encryption should certainly hide the plaintext. We claim that if a scheme is secure in the above sense, it has the property that it is computationally infeasible for an adversary to recover the plaintext from a ciphertext. This may not be apparent at first glance. The way we justify is to observe that 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 above sense, meaning figured out which of the two types of oracles it was given. But if the scheme is secure in our sense, that latter adversary could not exist. Hence, neither could the former.

The idea of this argument illustrates how we convince ourselves that the above definition is good, and captures all the properties we might want for security against chosen plaintext attack. Take some other property that you feel a secure scheme should have: infeasibility of key recovery from a few plaintext-ciphertext pairs; infeasibility of predicting the XOR of the plaintext bits; etc. Imagine there was an adversary B that was successful at this task. We claim this would enable us to construct an adversary A

that broke the scheme in the real-or-random sense, and hence B does not exist if the scheme is secure in the real-or-random sense. More precisely, we would use the insecurity function of the scheme to bound the probability that adversary B succeeds. Assuming the insecurity function is small at the specified parameter values, so is the chance that adversary B succeeds.

Let us go through one such example in detail. Let us suppose that there is an adversary that is capable of recovering a plaintext from a ciphertext. We even allow the adversary to first see some plaintext-ciphertext pairs, and then try to recover the plaintext from some new ciphertext. (Otherwise, the adversary's task is impossible for trivial reasons.) We will show that this adversary is unlikely to succeed as long as the number of pairs it was given, and the amount of time it invests, are not too large.

Proposition 4.11.2 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. Consider an adversary B that given a sequence $(M_1, C_1), \ldots, (M_m, C_m)$ of plaintext-ciphertext pairs, and a ciphertext C_{m+1}, returns a plaintext M. Define its success probability as

$$P_B = \mathbf{P}\Big[B((M_1, C_1), \ldots, (M_m, C_m), C) = M_{m+1} \; : \; K \stackrel{R}{\leftarrow} \mathcal{K} \, ; \, \texttt{for } i = 1, \ldots, m+1 \texttt{ do } M_i \stackrel{R}{\leftarrow} \{0,1\}^n \, ; \, C_i \stackrel{R}{\leftarrow} \mathcal{E}_K(M_i)\Big] .$$

Let t_B be the running time of B and let $t = t_B + O(nm)$. Then

$$P_B \leq 2^{-n} + \mathbf{Adv}^{\text{se-ror}}(\Pi; t, m+1, n(m+1)) . \quad \blacksquare$$

Here we pick a number m of messages of length n at random. B is provided the first m messages and their ciphertexts; but for the last message, it is provided only the ciphertext. Its job is to find M_{m+1}. We declare it successful if its output equals this message. The Proposition then says that the probability that B is successful is bounded above by the value of the insecurity function of the scheme at input parameters related to the attack parameters of B in a natural way, plus the negligible amount 2^{-n}. (Assume n is large, like 64 or more.) Think of the value of the insecurity function as being small. Then so too is B's success probability. In other words, it is hard to recover plaintext from a ciphertext, even if you already have some number of plaintext-ciphertext examples. But your ability grows as you get more examples, because the value of the insecurity function increases with m.

Let us now see why the above is true. It will illustrate a paradigm for using the definition.

Proof of Proposition 4.11.2: Let adversary B be given. We will define an adversary A that attacks the encryption scheme in the sense of real-or-random we have defined, using B as a subroutine, and show that

$$P_B \quad \leq \quad 2^{-n} + \mathbf{Adv}^{\text{se-ror}}(\Pi, A) \; . \tag{4.1}$$

Furthermore it will be true that A makes $m+1$ oracle queries, each consisting of an n-bit message, and runs for time $O(nm)$ over and above the time t_B taken for executing subroutine B. Hence $\mathbf{Adv}^{\text{se-ror}}(\Pi, A) \leq \mathbf{Adv}^{\text{se-ror}}(\Pi; t, m+1, n(m+1))$ and the claim of the proposition follows. So to complete the proof we only need to define A so that Equation (4.1) is true. Remember that A has access to an oracle $\mathcal{O}$. This oracle is one of two things, depending on which world A is in, and A does not know a priori which of the two kinds of oracles it has; indeed, it is trying to figure this out. B will help in this regard. Here is what A does:

Adversary $A^{\mathcal{O}}$
  For $i = 1, \ldots, m+1$ do
    $M_i \stackrel{R}{\leftarrow} \{0,1\}^n$; $C_i \stackrel{R}{\leftarrow} \mathcal{O}(M_i)$
  End For
  $M \leftarrow B((M_1, C_1), \ldots, (M_m, C_m), C)$
  If $M = M_{m+1}$ then return 1 else return 0

The idea is that A picks $m+1$ messages at random, each an n-bit string. It now calls its oracle $\mathcal{O}$ to obtain ciphertexts. What this returns depends on which world we are in. Remember that in world 1, $\mathcal{O}(M_i)$ returns $\mathcal{E}_K(M_i)$, an encryption of the given message, while in world 0, $\mathcal{O}(M_i)$ returns $\mathcal{E}_K(X_i)$, an encryption of a random string X_i of length n. It passes $(M_1, C_1), \ldots, (M_m, C_m)$ to B, along with the challenge ciphertext C_{m+1}, whose associated message M_{m+1} is *not* provided to B. Now, B will try to decrypt C_{m+1}, and return its guess M. A tests whether B was successful, namely whether $M = M_{m+1}$. If so, it bets that it (meaning A) was in world 1, meaning $\mathcal{O}$ was a real encryption oracle. Else A bets it is in world 0.

Why does this work? We need to compute the advantage of A in terms of P_B, the success probability of B. We claim that

$$\begin{aligned} \mathbf{Succ}^{\text{se-ror}}_{\text{Real}}(\Pi, A) &\;=\; P_B \\ \mathbf{Succ}^{\text{se-ror}}_{\text{Rand}}(\Pi, A) &\;\leq\; 2^{-n} \; . \end{aligned}$$

By definition of the advantage we then have

$$\mathbf{Adv}^{\text{se-ror}}(\Pi, A) \quad = \quad P_B - 2^{-n} \; .$$

Equation (4.1) follows. So now let us justify the above two equations.
To compute $\mathbf{Succ}^{\text{se-ror}}_{\text{Real}}(\Pi, A)$ we must consider what happens when $\mathcal{O}(\cdot) = \mathcal{E}_K(\cdot)$ is the encryption function. In that case, $C_i = \mathcal{E}_K(M_i)$. Now, look at the definition of P_B, and compare this to the code of A. You will see that A is simply executing the same experiment that P_B defines. Furthermore A returns one exactly when B returns M_{m+1}, so indeed $P_B = \mathbf{Succ}^{\text{se-ror}}_{\text{Real}}(\Pi, A)$.
To compute $\mathbf{Succ}^{\text{se-ror}}_{\text{Rand}}(\Pi, A)$ we must consider what happens when $\mathcal{O}(\cdot) = \mathcal{E}_K(\$^{|\cdot|})$ is function which given M picks a random X of the same length and returns $\mathcal{E}_K(X)$. Let us focus only on M_{m+1}. The ciphertext C_{m+1} is the result of $\mathcal{E}_K(X_{m+1})$ where X_{m+1} is randomly chosen by the oracle. This is the value that B is given. Now, the key observation is that B *is given absolutely no information about* M_{m+1}. Indeed, the information given to B is independent of M_{m+1}. So the chance that B can guess M_{m+1} is at most 2^{-n}. Note that this argument is true regardless of how much computational effort B invests; it is simply impossible to guess M_{m+1} with probability better than 2^{-n}.
It is worth going back over this proof to check out the various issues. For example, in our last argument above, we are running B on inputs $(M_1, C_1), \ldots, (M_m, C_m)$ where C_i is the encryption of some random X_i rather than an encryption of M_i. That is not what B "expects". But it does not matter. B is an algorithm; it will run on anything it is given. The only thing we care about is that its probability of guessing M_{m+1} in world 0 is the value we claimed, and this 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 real-or-random 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 4.11.2.

Of course one cannot exhaustively enumerate all desirable security properties. But you should be moving towards being convinced that our notion of real-or-random 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 being a little conservative. 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 real-or-random test, so that our formal notion too declares it insecure.

4.11.4 Alternative interpretation of advantage

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

As usual we fix a symmetric encryption scheme Π and let $\mathcal{K}, \mathcal{E}$ be its key generation and encryption algorithms respectively. We now consider the game, or experiment, of running the adversary in a random world and seeing whether or not it correctly guesses which world it is in.

Experiment $\text{CorGuessExp}^{\text{se-ror}}(\Pi, A)$
 Pick a bit b at random
 Let $K \stackrel{R}{\leftarrow} \mathcal{K}$
 Run A, replying to any query M to its oracle $\mathcal{O}$ as follows:
  If $b = 1$ then return $\mathcal{E}_K(M)$ to A
  Else pick $X \stackrel{R}{\leftarrow} \{0,1\}^{|M|}$ and return $\mathcal{E}_K(X)$ to A
 Until A stops making oracle queries
 Let g be the "guess" bit output by A
 If $b = g$ return 1 else return 0

Let $\mathbf{CorGuess}^{\text{se-ror}}(\Pi, A)$ denote the probability that the above experiment returns 1. This is the probability that A correctly guesses which world it is in. (The probability is over the initial choice of world as given by the bit b, the choice of K, the choices of X if $b = 0$, the random choices of $\mathcal{E}_K(\cdot)$ if any, and the coins of A if any.) The following proposition says that one-half of the advantage is exactly the excess above one-half of the chance that A correctly guesses which world it is in.

Proposition 4.11.3 Let Π be a symmetric encryption scheme and let A be an adversary attacking it in the real-or-random sense. Then

$$\mathbf{CorGuess}^{\text{se-ror}}(\Pi, A) \;=\; \frac{1}{2} + \frac{\mathbf{Adv}^{\text{se-ror}}(\Pi, A)}{2}\;.$$

Proof: We let $\mathbf{P}[\cdot]$ be the probability of event "$\cdot$" in the experiment $\mathrm{CorGuessExp}^{\text{se-ror}}(\Pi, A)$, and refer below to quantities in this experiment. The claim of the Proposition follows by a straightforward calculation:

$$\begin{aligned}
\mathbf{CorGuess}^{\text{se-ror}}(\Pi, A) &= \mathbf{P}[b = g] \\
&= \Pr[b = 1 \mid b = g] \cdot \mathbf{P}[b = 1] + \Pr[b = 0 \mid b = g] \cdot \mathbf{P}[b = 0] \\
&= \Pr[b = 1 \mid b = g] \cdot \frac{1}{2} + \Pr[b = 0 \mid b = g] \cdot \frac{1}{2} \\
&= \Pr[b = 1 \mid g = 1] \cdot \frac{1}{2} + \Pr[b = 0 \mid g = 0] \cdot \frac{1}{2} \\
&= \Pr[b = 1 \mid g = 1] \cdot \frac{1}{2} + (1 - \Pr[b = 0 \mid g = 1]) \cdot \frac{1}{2} \\
&= \frac{1}{2} + \frac{1}{2} \cdot (\Pr[b = 1 \mid g = 1] - \Pr[b = 0 \mid g = 1]) \\
&= \frac{1}{2} + \frac{1}{2} \cdot (\mathbf{Succ}^{\text{se-ror}}_{\text{Real}}(\Pi, A) - \mathbf{Succ}^{\text{se-ror}}_{\text{Rand}}(\Pi, A)) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\text{se-ror}}(\Pi, A)\;.
\end{aligned}$$

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 $b = g$ given that we know $b = 1$, it is the same as asking whether $g = 1$ 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 success probabilities in the real and random games respectively. That meant we had recovered the advantage, as desired. ▮

When we have such a strong and stringent notion of security, the next question is how to achieve it. We look to the modes of operation of a block cipher for the answer.

4.12 Security of the modes of operation

Some are secure, some are not. Of those that are secure, some are more secure than others. We begin by showing that ECB is insecure. We then turn to CTR. Although not as popular as CBC it is secure and easier to analyze than CBC. Finally we look at CBC.

4.12.1 Electronic Codebook Mode

Let us fix a block cipher F: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l$ be a block cipher. The ECB symmetric encryption scheme $\mathcal{ECB}^F = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is that of Example 4.10.2. Suppose an adversary sees a ciphertext $C = \mathcal{E}_K(x)$ corresponding to some unknown plaintext text x, encrypted under the key K also unknown to the adversary. Can the adversary recover x? Not easily, if F is a "good" block cipher. For example if F is RC6, 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 x and x' 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 the definition captures these weaknesses and also finds the scheme insecure. It does. To show this, we want to present an adversary that breaks the scheme in the sense of real-or-random. It must quickly and with high probability be able to tell which world it is in. The result, and the adversary that witnesses it, follow.

Proposition 4.12.1 Let F: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l$ be a block cipher, and $\mathcal{ECB}^F = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ the corresponding ECB symmetric encryption scheme as described in Example 4.10.2. Then there is an adversary A, running in $t = O(l)$ time, making 1 query to its oracle $\mathcal{O}$, this query being of length $2l$, and having

$$\mathbf{Adv}^{\text{se-ror}}(\mathcal{ECB}^F, A) \quad = \quad 1 - 2^{-l} \ .$$

It follows that

$$\mathbf{Adv}^{\text{se-ror}}(\mathcal{ECB}^F; t, 1, 2l) \quad \geq \quad 1 - 2^{-l} \ . \quad \blacksquare$$

The advantage of this adversary is almost 1 (for typical values of l like $l \geq 64$) even though it uses hardly any resources: just one query, and not a long one at that. That is clearly an indication that the scheme is insecure.

Proof of Proposition 4.12.1: Insecurity is justified by presenting an adversary that breaks the scheme in the model of symmetric encryption that we have presented. Remember such an adversary A is given an oracle $\mathcal{O}$, such that $\mathcal{O}$ is either a real encryption oracle, or returns the encryption of a random text of the same length as its argument. The goal of A is to tell which type of oracle it has. If A can do this with high probability using few queries and little time, the scheme is broken. And, for ECB, we can indeed present an adversary that succeeds. It works like this:

Adversary $A^{\mathcal{O}}$
  Pick some l-bit string w
  Let $x_1 = x_2 = w$ and let $x = x_1 x_2$
  Call $\mathcal{O}(x)$ to get back ciphertext $y_1 y_2$.
  If $y_1 = y_2$ output 1 else output 0.

How w is chosen is not important; A could simply set it to some fixed l-bit string, like the string of all zeros. Now, we claim that

$$\begin{aligned} \mathbf{Succ}^{\text{se-ror}}_{\text{Real}}(\mathcal{ECB}^F, A) &= 1 \\ \mathbf{Succ}^{\text{se-ror}}_{\text{Rand}}(\mathcal{ECB}^F, A) &= 2^{-l} . \end{aligned}$$

Hence $\mathbf{Adv}^{\text{se-ror}}(\mathcal{ECB}^F, A) = 1 - 2^{-l}$ is almost one, for typical value of l like $l = 64$. And A achieved this advantage by making just one oracle query, consisting of $2l$ bits. So $\mathbf{Adv}^{\text{se-ror}}(\mathcal{ECB}^F; t, 1, 2l) \geq 1 - 2^{-l}$.
Why are the two equations claimed above true? You have to return to the definitions of the quantities in question, and trace through the experiments defined there. In the first case, the oracle returns $\mathsf{E\text{-}ECB}^{F_K}(x_1x_2) = F_K(x_1)F_K(x_2)$, and since $x_1 = x_2$ we have $y_1 = y_2$. (We are using here some notation from the section on modes of operation in the chapter on block ciphers.) In the second case, the oracle picks at random a $2l$-bit string r and returns $\mathsf{E\text{-}ECB}^{F_K}(r) = F_K(r_1)F_K(r_2)$ where $r = r_1r_2$. Since F_K is a permutation, y_1 will equal y_2 only when $r_1 = r_2$. What is the probability of the latter event? We are picking a $2l$ bit string at random and asking what is the probability that its two halves are the same. This probability is 2^{-l}.
∎

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 F is highly secure.* The weakness is not in the tool being used, but in the manner we are using it. It is the ECB mechanism that is at fault. Even a good tool is useless if you don't use it well.

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 produces a good meal. If you don't use good ingredients? Well, that is your problem.

4.12.2 CTR mode

Let us fix a block cipher F: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^L$ be a family of functions. The CTR symmetric encryption scheme comes in two variants: the randomized (stateless) one of Example 4.10.4 and the counter-based (stateful) one of Example 4.10.5. Both are secure, but, interestingly, the counter version is more secure than the randomized version. We will first state the main theorems about the schemes, discuss them, and then prove them.

Security theorems and discussion

For the counter version we have:

Theorem 4.12.2 Suppose F: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^L$ is a PRF, and let $\mathcal{C}\text{-}\mathcal{CTR}^F = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding counter-based CTR symmetric encryption scheme as described in Example 4.10.5. Then for any t, q, μ with $\mu < L2^l$ we have

$$\mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^F; t, q, \mu) \quad \leq \quad 2 \cdot \mathbf{Adv}_F^{\text{prf}}(t', q') ,$$

where $t' = t + O(\mu)$ and $q' = \mu/L$. ▮

And for the randomized version:

Theorem 4.12.3 Suppose F: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^L$ is a PRF, and let $\mathcal{R}\text{-}\mathcal{CTR}^F = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the corresponding randomized CTR symmetric encryption scheme as described in Example 4.10.4. Then for any t, q, μ with $\mu < L2^l$ we have

$$\mathbf{Adv}^{\text{se-ror}}(\mathcal{R}\text{-}\mathcal{CTR}^F; t, q, \mu) \quad \leq \quad 2 \cdot \mathbf{Adv}_F^{\text{prf}}(t', q') \; + \; \frac{\mu(q-1)}{L2^l} ,$$

where $t' = t + O(\mu)$ and $q' = \mu/L$. ▮

This kind of result is what this whole approach and course are 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 cipher treated as a PRF) is secure. They 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 design flaws of ECB mode made 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 anyway. We are thus getting a conviction that *all attacks fail* even though we do not even know exactly how these attacks 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 actually have the ability to get 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 4.12.4 Let us suppose that F is RC6. So the key size is $k = 128$ and the block size is $l = L = 128$. Suppose I want to encrypt $q = 2^{40}$ messages, each $128 * 2^3$ bits long, so that I am encrypting a total of $\mu = 2^{50}$ bits of data. Can I do this securely using counter-mode CTR? What is the chance that an adversary figures out something about my data? Well, if the adversary has $t = 2^{60}$ computing cycles, then by definition its chance is not more than $\mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^F; t, q, \mu)$. That has nothing to do with the theorem: it is just our definitions, which say that this is the maximum probability of being able to break the encryption scheme in these given resources. So the question of whether the scheme is secure for my chosen parameters boils down to asking what is the value of $\mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^F; t, q, \mu)$. A priori, we have no idea. But now, we appeal to Theorem 4.12.2, which says that this chance is at most the value $2 \cdot \mathbf{Adv}_F^{\text{prf}}(t', q')$, where t', q' are given in the theorem. Lets figure them out. First $t' = t + O(\mu)$ is about 2^{60} since μ is so much less than t and we will assume the hidden constant in the big-oh is small. Second, $q' = \mu/L = 2^{50}/128 = 2^{43}$. So the question is, what is the value of $\mathbf{Adv}_F^{\text{prf}}(t', q')$ with these values of t', q'?

Thus, what the theorem has done is reduce the question of whether my data is securely encrypted to the question of whether RC6 is a good block cipher. If the latter is true, $\mathbf{Adv}_F^{\text{prf}}(t', q')$ is small, and I am now guaranteed that my data is private.

Let us explore the application further by seeing what one might say about RC6 in this context. We are asking how secure RC6 is as a PRF. A reasonable conjecture is that for any given values of $\tilde{t}$ and $\tilde{q}$ we have

$$\mathbf{Adv}_F^{\text{prf}}(\tilde{t}, \tilde{q}) \quad \leq \quad \frac{\tilde{q}^2}{2^l} + \frac{\tilde{t}}{2^k} \; .$$

Why? Recall that there is a birthday attack on any block cipher viewed as a PRF, which we have analyzed in the chapter on PRFs. The above assumption is saying that this birthday attack, or an exhaustive key search attack, are probably the best possible. We are not sure this is true, but it is a reasonable assumption today.

Now plug in $\tilde{t} = t' = 2^{60}$ and $\tilde{q} = q' = 2^{43}$ and take into account what we computed above. We get

$$\begin{aligned} \mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^F; t, q, \mu) \quad & \leq \quad 2 \cdot \mathbf{Adv}_F^{\text{prf}}(t', q') \\ & \leq \quad \frac{2(q')^2}{2^l} + \frac{2t'}{2^k} \\ & \approx \quad \frac{2^{43*2+1}}{2^{128}} + \frac{2^{61}}{2^{128}} \\ & \approx \quad \frac{1}{2^{41}} \; . \end{aligned}$$

That is, the chance the adversary gets any information about our encrypted data is about 2^{-41}, even though we allow this adversary computing time up to 2^{60}, and are encrypting 2^{50} bits of data. This is a very small chance, and we can certainly live with it. It is in this sense that we say the scheme is secure. ▮

Example 4.12.5 You are encouraged to work out another example along the following lines. Don't assume F is RC6, but rather assume it is an even better PRF. It still has $k = l = L = 128$, but assume it is not a permutation, so that there are no birthday attacks; specifically, assume

$$\mathbf{Adv}_F^{\mathrm{prf}}(\tilde{t}, \tilde{q}) \;\approx\; \frac{\tilde{q}}{2^l} + \frac{\tilde{t}}{2^k} \,.$$

Now, consider both the counter-based CTR scheme and the randomized one. In the theorems, the difference is the $\mu(q-1)/L2^l$ term. Try to see what kind of difference this makes. For each scheme, consider how high you can push q, μ, t and still have some security left. For which scheme can you push them higher? Which scheme is thus "more secure"?

These examples illustrate how to use the theorems to figure out how much security you will get from the CTR encryption scheme in some application.

Security analysis of the counter-based CTR scheme

We will now see how Theorem 4.12.2 is proved. 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.

Let us first introduce a piece of notation. For any function $f\colon \{0,1\}^l \to \{0,1\}^L$ we define an algorithm $\mathsf{E\text{-}CTR}^f$ that processes an input $x_1 \ldots x_n$ using f as an oracle. This means that the algorithm only has black-box access to f; there is an interface at which it can provide an l-bit value w, and the oracle returns $f(w)$ in one step. The algorithm implements counter-based CTR mode encryption using f to process the data. It uses and modifies the counter, which again is a global variable.

Algorithm $\mathsf{E\text{-}CTR}^f(x_1 \ldots x_n)$
  For $i = 1, \ldots, n$ do $y_i \leftarrow f(\langle c + i\rangle) \oplus x_i$
  Return $\langle c\rangle y_1 \ldots y_n$
  $c \leftarrow c + n$

Now, go back to the description of the encryption scheme $\mathcal{C}\text{-}\mathcal{CTR}^F$ in Example 4.10.5, and observe that the encryption algorithm defined there can be written in terms of the above algorithm, namely:

$$\mathcal{E}_K(x_1 \ldots x_n) \quad = \quad \mathsf{E\text{-}CTR}^{F_K}(x_1 \ldots x_n) \ .$$

This is simply saying that to encrypt, you do not need *direct* access to the key K. It suffices that you have a subroutine to compute the function F_K. That subroutine, from your point of view, is simply a box that takes an l-bit input and returns an L-bit output. This is a seemingly trivial observation that is actually crucial. The fact that the encryption algorithm makes only oracle access to F_K means that we can substitute some other function f in place of F_K, and see what happens to the encryption procedure. The latter will do something; it is an algorithm and if we run it, it will execute. And it is by plugging in something different for f that we will be able to measure the security of the algorithm.

The proof breaks into two parts. The first step removes the PRF F from the picture, and looks instead at an "idealized" version of the scheme. Namely we consider CTR mode encryption in which we use a random function f in place of F_K. We then assess an adversary's chance of breaking this idealized scheme. For the counter-based CTR scheme, we argue that this chance is actually zero. For the randomized scheme, we argue that while it is not necessarily zero, we an upper bound it quite well. These are the main lemmas 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 real block ciphers like RC6. Here we exploit the notion of pseudorandomness to say that the chance of an adversary breaking the real-world scheme (meaning the one using RC6) can differ from its chance of breaking the ideal-world scheme (the one using a random function) by an amount not exceeding the probability of breaking the pseudorandomness of the given block cipher. So if the block cipher is a good PRF, the adversary's chance of breaking the real-world (ie. block cipher based) CTR encryption scheme is small.

Recall that $R^{l,L}$ is the family of all functions of l-bits to L-bits, so that a random function is simply a randomly chosen member of $R^{l,L}$. The scheme considered below is exactly that in which a random function is used in place of F_K.

Lemma 4.12.6 Let $\mathcal{C}\text{-}\mathcal{CTR}^{R^{l,L}} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the counter-based CTR symmetric encryption scheme corresponding to the family $R^{l,L}$ of all functions of l-bits to L-bits. Let A be any adversary attacking this scheme in the real-or-random setting such that the total number μ of message bits in A's oracle queries is at most $L2^l$. Then

$$\mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^{R^{l,L}}, A) \;=\; 0\,. \quad \blacksquare$$

The lemma makes quite a strong statement. It says that the scheme is totally secure: an adversary has absolutely no chance of breaking it. The only slight caveat is that the total amount of data you can encrypt must be at most $l2^l$ bits. But this is hardly a restriction at all when $l = 128$, or even when $l = 64$. The conclusion holds regardless of how much computing time the adversary invests, and regardless of what strategy it uses. The adversary simply can't win.

Of course, this is in a world where the function f being used by the encryption algorithm is random. But remember that ECB was insecure even in that setting. 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.

Proof of Lemma 4.12.6: The idea underlying this is very simple. When f is a random function, its value on successive counter values yields a one-time pad, a truly random and unpredictable sequence of bits. The key point is that as long as the number of data bits encrypted does not exceed $L2^l$, we invoke f only on distinct values in the entire encryption process. The outputs of f 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 4.11.2. The adversary A has access to an oracle $\mathcal{O}$ that is instantiated in one of the two ways described there, corresponding to being either in world 0 or in world 1. Remember that in both worlds, the oracle encrypts under a randomly chosen key K, using the scheme under consideration. (The difference is in what message gets encrypted.) In the scheme $\mathcal{C}\text{-}\mathcal{CTR}^{R^{l,L}} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ we are considering now, the key K indexes into $R^{l,L}$, since the latter plays the role of the family F. That is, we do CTR encryption using $R_K^{l,L}$ as the function. For succinctness, call this f; it is simply a random function. So now we can see more clearly what are the two possible oracles. In world 1, the oracle $\mathcal{O}$, given a message $M = x_1 \ldots x_n$, returns $\mathsf{E\text{-}CTR}^f(x_1 \ldots x_n)$. In world 0, the oracle $\mathcal{O}$, given $M = x_1 \ldots x_n$, chooses $X = w_1 \ldots w_n$ at random and returns $\mathsf{E\text{-}CTR}^f(w_1 \ldots w_n)$. We want to figure out the adversary's advantage.

The adversary makes some number q of oracle queries. Let M_i be the i-th query; n_i the number of blocks in it; and $M_{i,j}$ the value of the j-th block. Let C_i be the response returned by the oracle to M_i. It consists of $n_i + 1$ blocks, the first block being the value of the counter at the start of the encryption of M_i; the other blocks are denoted $C_{i,1} \ldots C_{i,n_i}$. Pictorially:

$$
\begin{array}{lclcccl}
M_1 & = & M_{1,1}M_{1,2}\ldots M_{1,n_1} & \Longrightarrow & C_1 & = & \langle 0\rangle C_{1,1}\ldots C_{1,n_1} \\
M_2 & = & M_{2,1}M_{2,2}\ldots M_{2,n_2} & \Longrightarrow & C_2 & = & \langle n_1\rangle C_{2,1}\ldots C_{2,n_2} \\
\vdots & & & \vdots & & & \vdots \\
M_q & = & M_{q,1}M_{q,2}\ldots M_{q,n_q} & \Longrightarrow & C_q & = & \langle n_1+\cdots+n_{q-1}\rangle C_{q,1}\ldots C_{q,n_q} .
\end{array}
$$

What kind of distribution do the output received by A have? We claim that the $n_1 + \cdots + n_q$ values $C_{i,j}$ ($i = 1, \ldots, q$ and $j = 1, \ldots, n_i$) are randomly and independently distributed, not only of each other, but of the messages $M_1, \ldots, M_q$, 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 f on a counter. We observe that according to the scheme

$$
C_{i,j} \quad = \quad f(\langle n_1 + \cdots + n_{i-1} + j\rangle) \;\oplus\; \begin{cases} M_{i,j} & \text{if we are in world 1} \\ X_{i,j} & \text{if we are in world 0} \end{cases} ,
$$

where $X_i = X_{i,1} \ldots X_{i,n_i}$ is the message chosen randomly by $\mathcal{O}$ in world 0 for $i = 1, \ldots, q$. Now, we can finally see that the idea we started with is really the heart of it. The values on which f is being applied above are all distinct. So the outputs of f are all random and independent. It matters not, then, what we XOR these outputs with, whether a known quantity or a random one; 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 saying one must be the same in both worlds,

$$
\mathbf{Succ}^{\text{se-ror}}_{\text{Real}}(\mathcal{C}\text{-}\mathcal{CTR}^{R^{l,L}}, A) \quad = \quad \mathbf{Succ}^{\text{se-ror}}_{\text{Rand}}(\mathcal{C}\text{-}\mathcal{CTR}^{R^{l,L}}, A) .
$$

Hence A's advantage is zero. ▌

Now for part 2; lifting this to the real-world scheme that uses a given PRF F, in practice usually a block cipher; this corresponds to proving Theorem 4.12.2 using Lemma 4.12.6.

We have seen that the encryption scheme is (very!) secure when f is a random function. But that's not the scheme we use; we set $f = F_K$. So our worry is that the actual scheme is bad even though the ideal one is good. Imagine that were true. We claim that this contradicts the pseudorandomness of F. To see this, imagine that A is an adversary that has high probability of breaking $\mathcal{C}\text{-}\mathcal{CTR}^F$. We claim that using A we can design a way of winning the game that determines the pseudorandomness of F. Recall that in that game, a distinguisher D got an oracle f for a function that was either randomly chosen from F or was truly random, and had to say which. We suggest the following strategy to the distinguisher. Use the oracle f to implement the algorithm $\mathsf{E\text{-}CTR}^f$, so that you can produce encryptions under the CTR scheme using function f. See whether the adversary can break this scheme, meaning win its real-or-random game. If so, f must have been from F, because were f truly random, we know the adversary had zero chance of winning the game.

The full proof follows. It does not use the above contradiction argument, but the difference is superficial. The key point is that it defines the distinguisher we have just suggested.

Proof of Theorem 4.12.2: Let A be any adversary playing the real-or-random game with symmetric encryption scheme $\mathcal{C}\text{-}\mathcal{CTR}^F = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Assume A makes q oracle queries totaling μ bits, and runs for time t. We will design a distinguisher D such that

$$\mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^F, A) \quad \leq \quad 2 \cdot \mathbf{Adv}^{\text{prf}}_F(D) \ . \tag{4.2}$$

Furthermore, D will make $q' = \mu/L$ oracle queries and run for time $t' = t + O(\mu)$. Now, the statement of Theorem 4.12.2 follows as usual, by taking maximums. So the main thing is to provide the distinguisher for which Equation (4.2) is true. This distinguisher uses the strategy we discussed above, and in particular uses A as a subroutine.

Remember that D takes an oracle $f\colon \{0,1\}^l \to \{0,1\}^L$. This oracle is either drawn at random from F or from $R^{l,L}$ and D does not know a priori which. To find out, D will use A. But remember that A too gets an oracle, $\mathcal{O}$, according to the real-or-random game. From A's point of view, this oracle is simply a subroutine: A can write, at some location, a message, and is returned a response by some entity it calls $\mathcal{O}$. In the real-or-random game,

we gave it $\mathcal{O}$, but here, when D runs A as a subroutine, it is D that will "simulate" $\mathcal{O}$ for A, providing the responses to any oracle queries that A makes. It will do this in such a way that A feels no difference; A is being put in a "virtual reality" in which everything feels like it should, just like all those people in the movie *Matrix*, in case you happened to see it.
D is going to try to determine whether or not A can break the encryption scheme that uses f as the underlying function. What does break mean? That A correctly decides whether it had a real or a random oracle. So D will first pick which oracle to simulate for A; this choice is made by the bit b in the following code. Then D will see whether A guesses b right. If so, D declares that A has won, and bets that f came from F. Here, finally, is the actual code for D.

Distinguisher D^f
    Pick at random $b \in \{0,1\}$
    Run A
        When A makes an oracle query M, reply to it as follows:
            If $b = 1$ then return $\textsf{E-CTR}^f(M)$ to A
            Else pick a random string X of length $|M|$ and return $\textsf{E-CTR}^f(X)$ to A
    Until A stops making oracle queries and outputs its guess $c \in \{0,1\}$
    If $c = b$ then return 1 else return 0

A crucial aspect of D is the way it uses its oracle f. It uses it to compute the values of the encryptions to return to A. Namely, to execute $\textsf{E-CTR}^f(M)$, distinguisher D must invoke its own oracle f. This is the only place that f is used; without it, D cannot simulate the encryption and answer A's oracle queries. We use here the trivial-seeming fact about the encryption scheme that was mentioned above, namely that it is possible to compute encryptions given an oracle for the function.
We now analyze this distinguisher. We start with:

$$\begin{aligned}
\mathbf{Succ}^{\mathrm{prf}}_F(D) &= \Pr\left[\, f \stackrel{R}{\leftarrow} F \,:\, D^f = 1 \,\right] \\
&= \Pr\left[\, f \stackrel{R}{\leftarrow} F \,:\, b = c \,\right] \\
&= \Pr\left[\, f \stackrel{R}{\leftarrow} F \,:\, A \text{ is correct} \,\right] \\
&= \mathbf{CorGuess}^{\text{se-ror}}(\Pi, A) \\
&= \frac{1}{2} + \frac{\mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^F, A)}{2} \,.
\end{aligned}$$

The first equality is by definition of $\mathbf{Succ}_F^{\mathrm{prf}}(D)$ as per the chapter on PRFs. The second equality is true by the code of D, which returns 1 if and only if $b = c$. The third equality just notes that $b = c$ means that A is correct in its decision as to which oracle $\mathcal{O}$ it was given. The last equality is by Proposition 4.11.3.
Let $\delta = 0$. (This a rather trivial definition whose value will be clearer in the next subsubsection.) Then proceeding as above

$$\begin{aligned}
\mathbf{Succ}_{R^{l,L}}^{\mathrm{prf}}(D) &= \Pr\left[\, f \stackrel{R}{\leftarrow} R^{l,L} \;:\; D^f = 1 \,\right] \\
&= \Pr\left[\, f \stackrel{R}{\leftarrow} R^{l,L} \;:\; b = c \,\right] \\
&= \Pr\left[\, f \stackrel{R}{\leftarrow} R^{l,L} \;:\; A \text{ is correct} \,\right] \\
&= \frac{1}{2} + \frac{\mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^{R^{l,L}}, A)}{2} \\
&= \frac{1}{2} + \frac{\delta}{2} \,.
\end{aligned}$$

The last step is crucial: it used Lemma 4.12.6.
Now we can subtract to get

$$\begin{aligned}
\mathbf{Adv}_F^{\mathrm{prf}}(D) &= \mathbf{Succ}_F^{\mathrm{prf}}(D) - \mathbf{Succ}_{R^{l,L}}^{\mathrm{prf}}(D) \\
&= \left[\frac{1}{2} + \frac{\mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^{F}, A)}{2}\right] - \left[\frac{1}{2} + \frac{\delta}{2}\right] \\
&= \frac{\mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^{F}, A) - \delta}{2} \,.
\end{aligned}$$

Re-arranging terms gives us

$$2 \cdot \mathbf{Adv}_F^{\mathrm{prf}}(D) + \delta \;=\; \mathbf{Adv}^{\text{se-ror}}(\mathcal{C}\text{-}\mathcal{CTR}^{F}, A)$$

which implies Equation (4.2). (Remember that $\delta = 0$). ▌

Security analysis of the randomized CTR scheme

The security analysis of the randomized version of CTR mode will be able to re-use a lot of what we did for the counter-based version. The paradigm remains the same; we first look at the scheme when f is a random function, and then use the pseudorandomness of the given family F to deduce the theorem.

The proof is simplified by first re-defining the algorithm $\mathsf{E\text{-}CTR}^f$ we introduced above. For any function $f\colon \{0,1\}^l \to \{0,1\}^L$, the new version of algorithm $\mathsf{E\text{-}CTR}^f$ processes an input $x_1 \ldots x_n$, using f as an oracle, as follows:

Algorithm $\mathsf{E\text{-}CTR}^f(x_1 \ldots x_n)$
  $r \stackrel{R}{\leftarrow} \{0, 1, \ldots, 2^l - 1\}$
  For $i = 1, \ldots, n$ do $y_i \leftarrow f(\langle r + i \rangle) \oplus x_i$
  Return $\langle r \rangle y_1 \ldots y_n$

Namely it picks a random integer in the range $0, \ldots, 2^l - 1$ and uses it as the starting point to create the f-based one-time pad with which the data is XORed. Now, go back to the description of the encryption scheme $\mathcal{R}\text{-}\mathcal{CTR}^F$ in Example 4.10.4, and observe that the encryption algorithm defined there can be written in terms of the above algorithm, namely:

$$\mathcal{E}_K(x_1 \ldots x_n) \quad = \quad \mathsf{E\text{-}CTR}^{F_K}(x_1 \ldots x_n) \ .$$

We will make use of this below. Here is the main lemma.

Lemma 4.12.7 Let $\mathcal{R}\text{-}\mathcal{CTR}^{R^{l,L}} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the randomized CTR symmetric encryption scheme corresponding to the family $R^{l,L}$ of all functions of l-bits to L-bits. Let A be any adversary attacking this scheme in the real-or-random setting using at most q oracle queries, these totaling at most $\mu < L2^l$ bits. Then

$$\mathbf{Adv}^{\text{se-ror}}(\mathcal{R}\text{-}\mathcal{CTR}^{R^{l,L}}, A) \quad \leq \quad \frac{\mu(q-1)}{L2^l} \ . \quad \blacksquare$$

The proof of Theorem 4.12.3 given this lemma is easy at this point because it is almost identical to the above proof of Theorem 4.12.2. So let us finish that first, and then go on to prove Lemma 4.12.7.

Proof of Theorem 4.12.3: Let A be any adversary playing the real-or-random game with symmetric encryption scheme $\mathcal{R}\text{-}\mathcal{CTR}^F = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and proceed just like in the proof of Theorem 4.12.2. We want to define distinguisher D so that

$$\mathbf{Adv}^{\text{se-ror}}(\mathcal{R}\text{-}\mathcal{CTR}^F, A) \quad \leq \quad 2 \cdot \mathbf{Adv}^{\text{prf}}_F(D) \ + \ \frac{\mu(q-1)}{L2^l} \ . \qquad (4.3)$$

The code for D is unchanged; but note that the underlying algorithm $\mathsf{E\text{-}CTR}^f(\cdot)$ that D runs has just been re-defined above to conform to the randomized version of the scheme, so that D is working with the randomized version of the scheme. For the analysis, the only change is to set $\delta = \mu(q-1)/L2^l$, the quantity from Lemma 4.12.7, rather than $\delta = 0$ as before. We get the same final equation

$$2 \cdot \mathbf{Adv}_F^{\mathrm{prf}}(D) + \delta \quad = \quad \mathbf{Adv}^{\mathrm{se\text{-}ror}}(\mathcal{R}\text{-}\mathcal{CTR}^F, A)$$

and thus derive Equation (4.3) as desired. ▮

The above illustrates how general and generic was the "simulation" argument of the proof of Theorem 4.12.2. Indeed, it adapts easily not only to the randomized version of the scheme, but to the use of pseudorandom functions in many other schemes, even for different tasks like message authentication. The key point that makes it work is that the scheme itself invokes f as an oracle.

Before we prove Lemma 4.12.7, 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 4.12.8 Let n, q, l be positive integers, and let $n_1, \ldots, n_q < 2^l$ also be positive integers. Suppose we pick q integers $r_1, \ldots, r_q$ from the range $\{0, 1, \ldots, 2^l - 1\}$, uniformly and independently at random. We consider the following $n_1 + \cdots + n_q$ numbers:

$$\begin{array}{cccc} r_1+1, & r_1+2, & \cdots, & r_1+n_1 \\ r_2+1, & r_2+2, & \cdots, & r_2+n_2 \\ \vdots & & & \vdots \\ r_q+1, & r_q+2, & \cdots, & r_q+n_q \ , \end{array}$$

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, and let Col denote the event that a collision occurs. Then

$$\mathbf{P}\left[\mathsf{Col}\right] \quad \leq \quad \frac{(q-1)(n_1 + \cdots + n_q)}{2^l} \ .$$

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

Think of having 2^l bins, numbered $0, 1, \ldots, 2^l - 1$. 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 $r_1+1, \ldots, r_1+n_1$. 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 $r_2+1, \ldots, r_2+n_2$. 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

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

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

$$\Pr[\mathsf{NoCol}_{i-1} \mid \mathsf{Col}_i] \le \frac{(i-1)n_i + n_{i-1} + \cdots + n_1}{2^l} \, . \tag{4.4}$$

Let us first see why this proves the lemma and then return to justify it. From the above and Equation (4.4) we have

$$\mathbf{P}[\mathsf{Col}] \le \sum_{i=2}^{q} \Pr[\mathsf{NoCol}_{i-1} \mid \mathsf{Col}_i]$$

$$
\begin{aligned}
&\leq \sum_{i=2}^{q} \frac{(i-1)n_i + n_{i-1} + \cdots + n_1}{2^l} \\
&= \frac{(q-1)(n_1 + \cdots + n_q)}{2^l}\,.
\end{aligned}
$$

How did we do the last sum? The term n_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 (4.4). 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 $r_2+1, \ldots, r_2+n_2$ defined by the second ball is equal to one of the numbers $r_1+1, \ldots, r_1+n_1$ already in the table? View r_1 as fixed. Observe that a collision occurs if and only if $r_1 - n_2 + 1 \leq r_2 \leq r_1 + n_1 - 1$. So there are $(r_1 + n_1 - 1) - (r_1 - n_2 + 1) + 1 = n_1 + n_2 - 1$ choices of r_2 that could yield a collision. This means that $\Pr[\mathsf{NoCol}_1 \mid \mathsf{Col}_2] \leq (n_2 + n_1 - 1)/2^l$.

We need to extend this argument as we throw in more balls. So now suppose $i-1$ balls have been thrown in, where $2 \leq i \leq q$, 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

$$
\begin{aligned}
\Pr[\mathsf{NoCol}_{i-1} \mid \mathsf{Col}_i] &\leq \frac{(n_i + n_1 - 1) + (n_i + n_2 - 1) + \cdots + (n_i + n_{i-1} - 1)}{2^l} \\
&= \frac{(i-1)n_i + n_{i-1} + \cdots + n_1 - (i-1)}{2^l}\,,
\end{aligned}
$$

and Equation (4.4) follows by just dropping the negative term in the above.
∎

Let us now extend the proof of Lemma 4.12.6 to prove Lemma 4.12.7.

Proof of Lemma 4.12.7: Recall that the idea of the proof of Lemma 4.12.6 was that when f is a random function, its value on successive counter values yields a one-time pad. This holds whenever f is applied on some set of distinct values. In the counter case, the inputs to f 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 be the i-th query; n_i the number of blocks in it; and $M_{i,j}$ the value of the j-th block. Let C_i be the response returned by the oracle to M_i. It consists of $n_i + 1$ blocks, the first block being the random integer $r_i \in \{0, 1, \ldots, 2^l - 1\}$ chosen by $\mathsf{E\text{-}CTR}^f(M_i)$; the other blocks are denoted $C_{i,1} \ldots C_{i,n_i}$. Pictorially:

$$
\begin{array}{rcl}
M_1 & = & M_{1,1}M_{1,2}\ldots M_{1,n_1} & \Longrightarrow & C_1 & = & \langle r_1\rangle C_{1,1}\ldots C_{1,n_1} \\
M_2 & = & M_{2,1}M_{2,2}\ldots M_{2,n_2} & \Longrightarrow & C_2 & = & \langle r_2\rangle C_{2,1}\ldots C_{2,n_2} \\
\vdots & & & \vdots & & & \vdots \\
M_q & = & M_{q,1}M_{q,2}\ldots M_{q,n_q} & \Longrightarrow & C_q & = & \langle r_q\rangle C_{q,1}\ldots C_{q,n_q} \ .
\end{array}
$$

Let NoCol be the event that the following $n_1 + \cdots + n_q$ values are all distinct:

$$
\begin{array}{cccc}
r_1+1, & r_1+2, & \cdots, & r_1+n_1 \\
r_2+1, & r_2+2, & \cdots, & r_2+n_2 \\
\vdots & & & \vdots \\
r_q+1, & r_q+2, & \cdots, & r_q+n_q
\end{array}
$$

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:

$$
\begin{array}{rcl}
\mathbf{P}_0\left[\cdot\right] & = & \text{The probability of event “$\cdot$” in world 0} \\
\mathbf{P}_0\left[\cdot\right] & = & \text{The probability of event “$\cdot$” in world 1} \ .
\end{array}
$$

So by definition the advantage of A is

$$
\mathbf{Adv}^{\text{se-ror}}(\mathcal{R}\text{-}\mathcal{CTR}^{R^{l,L}}, A) \quad = \quad \mathbf{P}_1\left[A=1\right] - \mathbf{P}_0\left[A=1\right] .
$$

The oracle $\mathcal{O}$ is different in the two cases, being set according to the game being played. There is nothing deep going on here; we are just changing notation to make it more succinct.

Our goal is to upper bound $\mathbf{Adv}^{\text{se-ror}}(\mathcal{R}\text{-}\mathcal{CTR}^{R^{l,L}}, A)$. 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: $\mathbf{P}_1[\mathsf{Col}] = \mathbf{P}_0[\mathsf{Col}]$. □

The second claim says that A has zero advantage in winning the real-or-random game in the case that no collisions occur in the table. Namely, its probability of outputting one is identical in these two world under the assumption that no collisions have occurred in the values in the table.

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

We can say nothing about the advantage of A if a collision does occur in the table. That is, 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: $\mathbf{P}_0[\mathsf{Col}] \leq \dfrac{\mu(q-1)}{L2^l}$. □

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:

$$\begin{aligned}
&\mathbf{Adv}^{\text{se-ror}}(\mathcal{R}\text{-}\mathcal{CTR}^{R^{l,L}}, A)\\
&= \mathbf{P}_1[A=1] - \mathbf{P}_0[A=1]\\
&= \Pr_1[\mathsf{Col} \mid A=1]\cdot \mathbf{P}_1[\mathsf{Col}] + \Pr_1[\mathsf{NoCol} \mid A=1]\cdot \mathbf{P}_1[\mathsf{NoCol}]\\
&\qquad - \Pr_0[\mathsf{Col} \mid A=1]\cdot \mathbf{P}_0[\mathsf{Col}] - \Pr_0[\mathsf{NoCol} \mid A=1]\cdot \mathbf{P}_0[\mathsf{NoCol}]
\end{aligned}$$

Using Claim 1 and Claim 2, the above equals

$$\begin{aligned}
&= \left(\Pr_1[\mathsf{Col} \mid A=1] - \Pr_0[\mathsf{Col} \mid A=1]\right)\cdot \mathbf{P}_0[\mathsf{Col}]\\
&\leq \mathbf{P}_0[\mathsf{Col}]\,.
\end{aligned}$$

In the last step we simply 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 $\mathsf{E}\text{-}\mathsf{CTR}^f$. 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 and the pad is generated. □

Proof of Claim 2: Given the event NoCol, we have that, in either game, the function f is evaluated at a new point each time it is invoked. (Here we use the assumption that $\mu < L2^l$, since otherwise there may be wraparound in even a single query.) 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 4.12.6 applies. Namely we observe that according to the scheme

$$C_{i,j} \quad = \quad f(\langle r_i + j \rangle) \;\oplus \left\{ \begin{array}{ll} M_{i,j} & \text{if we are in world 1} \\ X_{i,j} & \text{if we are in world 0} \end{array} \right. ,$$

where $X_i = X_{i,1} \ldots X_{i,n_i}$ is the message chosen randomly by $\mathcal{O}$ in world 0 for $i = 1, \ldots, q$. 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. $\Box$

Proof of Claim 3: This follows from Lemma 4.12.8. We simply note that $n_1 + \cdots + n_q = \mu/L$. $\Box$ ▮

4.13 Problems

Problem 4.1 Analyze the exhaustive key search attack in the real-or-random setting.

Acknowledgments

Thanks to the following for corrections to earlier drafts: Andre Barroso, Alexandra Boldyreva, Alex Gantman, Chanathip Namprempre.

Bibliography

[1] Mihir Bellare, Anand Desai, Eron Jokipii and Phillip Rogaway, "A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation," FOCS '97. Available from `http://www.cs.ucdavis.edu/~rogaway/papers/sym-enc-abstract.html`