Last Revised: 2/8/96.

ECS 227: Modern Cryptography
Lectures 7, 8

Symmetric Encryption

Phillip Rogaway
{1, 6} *February 1996*

7 Symmetric Encryption

The "classic" problem of cryptography is *symmetric encryption*. The name "symmetric" stems from the trust model: both the sender, S, and the receiver, R, share the *same* piece of information, a secret key a. The adversary lacks this key. In these lectures we shall describe *what* the symmetric encryption goal means, and how to achieve it. It is the former which will take up most of our efforts.

7.1 The basic setup

A standard picture for symmetric encryption is show as Figure 1. Party S is trying to communicate with party R, but adversary A is listening in on the conversation. We indicate the key a which S and R share by writing it as a superscript to each party's name.

The encryption function is $\mathcal{E}$ and the decryption function is $\mathcal{D}$. When S wants to send a message x to R, she computes $y \leftarrow \mathcal{E}_a(x)$ and sends R the message y. Party R, on receipt of message y, computes $\mathcal{D}_a(y)$, which better be the string x. The string x is called the *plaintext* (or *cleartext*), while the string y is called the *ciphertext*.

As we shall see, a good encryption scheme $\mathcal{E}$ will need to be *probabilistic* or *stateful*. For now, ignore the latter. The former means that $\mathcal{E}$ depends on secret coins, r, which you can think of as a third argument to $\mathcal{E}$. A message x is

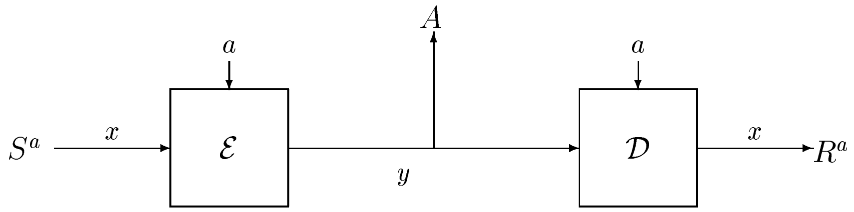

Figure 1: One view of symmetric encryption. Adversary A witnesses the encryptions of messages. The encryption function might be probabilistic, encrypting x differently each time $\mathcal{E}$ is applied.

encrypted by choosing a fresh sequence of random coin tosses, r, and computing $y = \mathcal{E}_a(x, r)$. When we write $y \leftarrow \mathcal{E}_a(x)$ and $\mathcal{E}$ is probabilistic we are indicating that y is a random sample from the induced probablity space $\mathcal{E}_a(x, \cdot)$. Since $\mathcal{D}$ is always deterministic, $\mathcal{D}$ will be obliged to recover x from $y = \mathcal{E}_a(x, r)$ for *any* sequence of coin tosses r.

7.2 Levels of adversarial control

The picture of Figure 1 would seem to indicate that adversary A is a completely passive eavesdropper. But in many real-life settings A has influence over what messages may be encrypted.

Likewise, the picture of Figure 1 would seem to indicate that A has no knowledge of what is being encrypted. But in many real-life settings A may have a great deal of information about the contents of some or all encrypted messages.

It is customary to distinguish the following levels of adversarial knowledge and control. As we go down the list, the adversary becomes more and more powerful.

- In a distributional attack one assumes that there is a certain distribution, $\mathcal{M}$, on plaintexts. The adversary sees samples of ciphertexts where the plaintexts have been drawn according to distribution $\mathcal{M}$.

- In a known-plaintext attack there is, once again, some distribution on plaintexts. But now the adversary sees $\langle$plaintext, ciphertext$\rangle$ pairs, where the plaintexts are generated by sampling in $\mathcal{M}$.

- In a 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 any messages $x_1, x_2, \ldots$ she likes, and she finds out

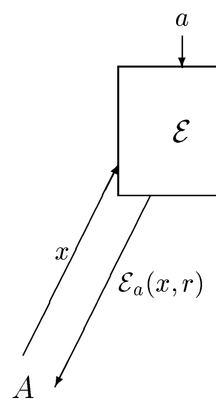

Figure 2: Another view of symmetric encryption. Adversary A is given an oracle which encrypts strings, modeling her potential ability to influence which strings are encrypted.

their encryptions $y_1, y_2, \ldots$. A chosen plaintext attack may be adaptive, meaning that x_i might depend on the earlier encrypted texts.

- In a chosen-ciphertext attack we give the adversary all of the abilities of a chosen-plaintext attack, plus one more: now the adversary may choose ciphertexts $y_1, y_2, \ldots$ and learn, for each, its corresponding plaintext. This models an adversary who might (perhaps temporarily) have access to a some decryption device. As before, the ciphertexts might be chosen adaptively. Requests to encipher x_i and decipher y_j might be arbitrarily interleaved.

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.

In many applications of cryptography the system architecture makes chosen-ciphertext attacks impossible to carry out. On the other hand, real adversaries usually does have at least a little control of the messages which will be encrypted, and adversaries often have a substantial amount of *a. priori* information about plaintexts. (Adaptive) chosen-plaintext attack may be considered a worst-case scenario with respect to the these capabilities. Thus chosen-plaintext attack is the notion we prefer and, for the rest of these lectures, we restrict our attention to modeling this notion.

A fresh view of a chosen-plaintext attack is depicted in Figure 2. Adversary A chooses each message x which is to be encrypted, and each time she does this she gets back a ciphertext y for x. The ciphertext is chosen by sampling in $\mathcal{E}_a(x)$. We emphasize that that the oracle with which A speaks will, typically, give a different answer each time you ask it a given question.

We have now figured out a tiny piece of the definition— that it will involve an adversary A with access to an oracle, $E_a(\cdot)$. We will denote such an adversary by $A^{E_a(\cdot)}$.

7.3 Intuition for two different definitions

Now we've described *what* the adversary can do, but we've said nothing about what she is trying to accomplish. That is, we must define what it *means* for an adversary to succeed.

Perhaps the first thing that comes to mind is to try and capture the *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 we have a terrible encryption scheme —the identity function, let's say— the encryption of x under 100-bit key a is x itself. The key a is supposed to be 100 random bits. Now the adversary has only a tiny chance to recover the key a (just 2^{-100}), no matter how much work the adversary does, and not matter how much enciphered text the adversary obtains. But the scheme is clearly worthless.

A second idea would be to try to formalize that the encryption of a string looks 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.

More on track —and most people's intuition— would be to formalize the following: *an adversary can't recover x from $E_a(x)$.* Such intuition can certainly be lifted to a definition, but the definition that results will end up being *too weak.* Let us see why.

In an encryption scheme in which all we know is that A can't recover x from $\mathcal{E}_a(x)$, maybe the adversary can still recover some particular bit of x. And maybe that bit of x 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. Alternatively, maybe the adversary can't recover x, nor any particular bit of x, but still she can recover some interesting function of the plaintext, like 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.**

In this phrase "properly formalized" we hide a lot. One immediate concern is that there seems to be some information about x which is *inherently* leaked by $\mathcal{E}_a(x)$. One such piece of information is the existence of some plaintext x. A second piece of information is length of x. And a third piece of information is an encryption of the message x 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'; it is just that we don't consider this to be the domain of techniques which we call by "encryption." We deal with the third objection by saying that $\mathcal{E}_a(x, r)$ is not really information about x— information about x is realized by some *deterministic* function f applied to x.

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 cleartext given the cyphertext, is also efficiently computable without the cyphertext.

We shall formalize the sentence above.

Though strong and compelling, the notion of semantic security is a bit complex and hard to work with. Fortunately, there are other nice definitions for an encryption scheme to be secure. We now describe one of these.

We imagine an adversary with her oracle for $\mathcal{E}_a(\cdot)$, as in Figure 2. Suppose that $\mathcal{E}$ hides the plaintext so well that for any adversary A, A can not tell if what she is getting back from the oracle is really an encryption of x, or if, instead, it is the encryption of a random string of length $|x|$. That is, we are capturing the intuition:

> The encryption of x is indistinguishable from the encryption of an equal-length string of nonsense.

Let us rephrase the above. We think of two different games. In the first game, the oracle E given to the adversary behaves as follows: a random key a is selected at the beginning, and then oracle E responds to each question x by sampling from the distribution $E_a(x)$. This is a "real" encryption oracle. In the second game, the oracle E behaves as follows: a random key a is selected at the beginning, and then oracle E responds to each question x by choosing a random string \$ of length $|x|$, and then sampling from the distribution $E_a(\$)$. This is a "bogus" encryption oracle. We ask, for an encryption scheme to be secure, that adversary's should be unable to tell apart real and bogus encryption oracles.

We will call the above notion **security in the sense of indistinguishability.** The name comes from the fact that we are asking that the encryptions of (known) strings be indistinguishable, to the adversary, from encryptions of nonsense. This notion is an adaptation of what call Goldwasser and Micali called "polynomial security."

Over the next sections we will be lifting the above intuition to definitions.

7.4 Semantic security

Recall the intuition that we're trying to formalize: that whatever is efficiently computable about the plaintext given the cyphertext should also be efficiently computable without the cyphertext. Let us begin by focusing our attention on quantifying how well an adversary can compute "whatever" in the absence of any information about the specific plaintext or ciphertext.

Let $f : \{0,1\}^* \to \{0,1\}^*$ be an arbitrary function. It captures the "whatever" which the adversary may wish to compute about plaintexts. How well can the adversary compute $f(x)$ without seeing information related to x? Well certainly that depends not only on f, but also on the distribution of the messages. Here are a couple of examples to clarify this:

- Suppose $f(x)$ is the XOR of the bits of x, and $\mathcal{M} = U_8$ is the uniform encoding on 8-bit strings. Then the adversary can correctly guess $f(x)$ with probablity $1/2$. A guess of 0 or a guess of 1 achieves this rate of success. The adversary can't do better.
- Suppose $f(x)$ is the XOR of the bits of x, and $\mathcal{M}$ is the distribution on 8-bit strings which puts probability mass $1/5$ on the ASCII encodings of `a`, `e`, `i`, `o`, `u`. Then, without knowing anything about x, the adversary can still guess $f(x)$ correctly $3/5$ of the time— the adversary should guess 0, since exactly three of the five vowels have even parity.
- Suppose $f(x) = 01$ if every bit of x is 0, and $f(x) = 111$ in all other cases. Let $\mathcal{M}$ be the uniform distribution on 8-bit strings. Then, without knowing anything about x, the adversary can still guess $f(x)$ correctly $255/256$ of the time— the adversary should guess $f(x) = 111$, as this value is the most likely.

Generalizing a bit, given $\mathcal{M}$ and f, the adversary's best guess of $f(x)$ is any string y^* which maximizes $\Pr_{x \leftarrow M}[f(x) = y^*]$. When the adversary makes that guess, she'll be right with probablity $P^*_{\mathcal{M},f} \stackrel{\text{def}}{=} \Pr_{x \leftarrow \mathcal{M}}[f(x) = y^*]$.

We want to say that, in a "good" encryption scheme, the adversary can't guess $f(x)$ much better $P^*_{\mathcal{M},f}$. But there's a minor glitch: we've said that we don't object to the *length* of x being revealed by a good encryption scheme, and the adversary can certainly take this length into account to choose the best y^*_n for string length n she has seen. This was a non-issue in our examples because $\mathcal{M}$ gave non-zero probablity mass only to equal-length strings. In fact, let's raise this to a definition, saying that a distribution on messages is *valid* if it gives probablity mass only to strings of equal length. We will restrict our attention to valid distribution.

Once the following definition is understood, the reader can become convinced that other ways of dealing with the above issue are equivalent.

The reader should now be ready to understand the notion of semantic security:

Definition 7.4.1 *Let $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ be an encryption scheme, let $f : \{0,1\}^* \to \{0,1\}^*$ be a function, and let $\mathcal{M}$ be a valid distribution on plaintexts. Adversary A is said to (t, q, μ, ϵ)-break Π, in the sense of semantic security, with respect to f, $\mathcal{M}$, if*

$$\begin{aligned} \mathrm{Adv}_A^{\mathcal{M},f} &\stackrel{\mathrm{def}}{=} \Pr\left[a \leftarrow \mathcal{K};\ x \leftarrow \mathcal{M};\ y \leftarrow \mathcal{E}_a(x) :\ A^{\mathcal{E}_a(\cdot)}(y) = f(x)\right] - P^*_{\mathcal{M},f} \\ &\geq \epsilon \end{aligned}$$

and, in the experiment above, A runs in at most t steps and makes at most q queries, these totalling at most μ bits.

The above definition uses an as-yet-undefined definition of an *encryption scheme,* $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$. We will formally define this later. The first two components are the familiar encryption and decryption function. The last component, $\mathcal{K}$, is the *key-generation algorithm.* It samples from the space of keys. For example, if the encryption scheme is based on DES, $\mathcal{K}$ would typically be the (probabilistic) algorithm which chooses a random 56-bit key.

As a reminder, look how we've given A an oracle for $\mathcal{E}_a(\cdot)$. This was to model chosen-plaintext attack.

The above definition speaks about how well an adversary does her job. We can recast our definition to speak directly about the quality of the scheme Π.

Definition 7.4.2 *Encryption scheme $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ is (t, q, μ, ϵ)-secure, in the sense of semantic security, if for every $f : \{0,1\}^* \to \{0,1\}^*$ and every valid distribution on plaintexts $\mathcal{M}$, there is no adversary A which (t, q, μ, ϵ)-breaks Π, in the sense of semantic security, with respect to f, $\mathcal{M}$.*

You might "unwind" the definition (ie., express it without reference to the earlier definition), just to make sure you fully understand it.

7.5 Security in the sense of indistinguishability

Our other notion of security —security in the sense of indistinguishability— is much simpler to define.

Definition 7.5.1 *Let $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ be an encryption scheme. Adversary A is said to (t, q, μ, ϵ)-break Π, in the sense of indistinguishability, if*

$$\begin{aligned} \mathrm{Adv}_A^{E(\cdot),E(\$^{|\cdot|})} &\stackrel{\mathrm{def}}{=} \Pr\left[a \leftarrow \mathcal{K} :\ A^{\mathcal{E}_a(\cdot)} = 1\right] - \Pr\left[a \leftarrow \mathcal{K} :\ A^{\mathcal{E}_a(\$^{|\cdot|})} = 1\right] \\ &\geq \epsilon \end{aligned}$$

and, in the experiments above, A runs in at most t steps and makes at most q queries, these totalling at most μ bits.

In the notation $A^{\mathcal{E}_a(\$^{|\cdot|})}$, the indicated oracle is understood to behave as follows: in response to each query x, the oracle flips $|x|$ random bits and then returns an encryption of these bits. Thus the query x is ignored, apart from its length.

As before, we can recast the last definition to speak of the scheme, and not the adversary. The initial state is taken to be the empty string.

Definition 7.5.2 *Encryption scheme* $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ *is* (t, q, μ, ϵ)*-secure, in the sense of indistinguishability, if there is no adversary* A *which* (t, q, μ, ϵ)*-breaks* Π*, in the sense of indistinguishability.*

7.6 There is no secure encryption which is deterministic and stateless

Suppose encryption scheme Π is completely stateless and deterministic. Then Π can not possibly be a good encryption scheme.

To see that Π won't meet our notion of secure in the sense of indistinguishability (with reasonable bounds) just have adversary A repeatedly ask the same question. If A's oracle is an $\mathcal{E}_a(\cdot)$ oracle, then the oracle will keep returning the same answer. But if the oracle is an $\mathcal{E}_a(\$^{|\cdot|})$ oracle, then A will see different answers to the same query. Thus A will easily figure out which type of oracle she has.

The above isn't a problem—it's exactly as it should be. After all, if the adversary could witness a sequence of identical ciphertexts $y_1 = y_2 = y_3 = \cdots$, then the adversary will have learned something very interesting about the corresponding plaintexts: that they are all equal, too. A good encryption scheme should not be revealing such information about the sequence of plaintexts.

There is an alternative (and often better) way to ensure that $\mathcal{E}$ doesn't always return the same ciphertext in response to identical plaintexts: instead of making the function $\mathcal{E}$ probabilistic, we can make it *stateful.* This means that the process of performing an encryption not only yields enciphered text, but it also modifies the internal state of the encryption routine. If you call the encryption routine again, it will encrypt the same string differently because it will have started off in a different internal state.

A stateful encryption scheme off in some fixed *initial* state. The state may change in a way that depends on the messages which are being encrypted. Often the state is simply a counter.

We don't object that a stateful encryption scheme cease to operate (refuse to encrypt) at some point in time. For example, if the state is a counter, than there may be some upper bound on how large this counter is allowed to grow. Or the scheme may refuse to operate once the amount of enciphered text reaches some certain number of bytes.

Stateful encryption schemes can be easily handled programmatically, through "static" variables, or through information stored on a "key handle." In fact, it is usually easier to implement stateful encryption than probabilistic ones, since you don't have to worry about where to find random bits in your computer.

We emphasize that it is only the encryption scheme —the sender— which may be stateful. If the decryption scheme —the receiver— needs to know something of the state at the time a message was encrypted, than that state information needs to be communicated as part of the enciphered text.

7.7 An encryption scheme as a syntactic object

We've been using encryption schemes, $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$, without having formally defined one. Let's remedy that omission.

A (probabilistic, stateless, symmetric) encryption scheme, Π, is a three-tuple of string-valued functions, $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$, where

$$\begin{aligned} \mathcal{E} &: \text{Keys} \times \mathcal{M} \times \{0,1\}^\infty \to \{0,1\}^* \\ \mathcal{D} &: \text{Keys} \times \{0,1\}^* \to \mathcal{M} \cup \{\star\} \\ \mathcal{K} &: \{0,1\}^\infty \to \text{Keys} \end{aligned}$$

and Keys is the target of $\mathcal{K}$. We demand that, for every $k \in \text{Keys}$, $x \in \mathcal{M}$, and $r \in \{0,1\}^\infty$, $\mathcal{D}(k, \mathcal{E}(k, x, r)) = x$. We further demand that, if $\mathcal{M}$ contains some string x, then $\mathcal{M}$ contains every string of length $|x|$.

The set $\mathcal{M}$ is called the *message space*. The set Keys is called the *key space*.

The argument $\{0,1\}^\infty$ provided to $\mathcal{E}$ and $\mathcal{K}$ represent infinite sequences of random coins. Often we omit writing this argument. In that case, the function is regarded as a probabilistic map, or else a probability space. The first argument to $\mathcal{E}$ and $\mathcal{D}$ are usually written as a subscripts.

The return value of "$\star$" for $\mathcal{D}$ is meant to be used to indicate "not a valid ciphertext." One could demand that, for every $a \in \text{Keys}$, $\mathcal{D}_a(y) = \star$ whenever $y \neq \mathcal{E}_a(x, r)$ for any $x \in \mathcal{M}$, $r \in \{0,1\}^\infty$. In developing a theory for symmetric encryption, it does not seem to be important to make this demand. Remember that encryption is only meant to preserve privacy; it has nothing to do with guaranteeing the authenticity of transmitted messages.

For an encryption scheme to be useful, $\mathcal{E}$, $\mathcal{D}$, and $\mathcal{K}$ should be efficiently-computable functions. Usually it is the efficiency of $\mathcal{E}$ and $\mathcal{D}$ which matter more, since keys are chosen less frequently than messages are encrypted or decrypted. Nonetheless, $\mathcal{K}$ is often trivial.

In the last section we discussed how it is sometimes useful for Π to be *stateful*. In this case, $\mathcal{E}$ becomes a map

$$\mathcal{E} : \text{Keys} \times \mathcal{M} \times \text{State} \times \{0,1\}^\infty \to \{0,1\}^* \times \text{State}$$

where the new argument, $\text{State} \subseteq \{0,1\}^*$, holds the "current state." The set State includes the empty string, ε. The return value is (y, s'), for ciphertext y and *revised state* s'. A stateful encryption scheme may return the distinguished message ε as an indication of unwillingness to encrypt the requested message.

In the case that Π is stateful we must modify the definition of the behavior of an oracle $\mathcal{E}_a(\cdot)$. The oracle still has only one argument, but now the oracle maintains a *current state*, s, which is initially ε. When an oracle in state s and receives a query x it samples $(y, s') \leftarrow \mathcal{E}_a(x, s)$, updates the current state to s', and returns y.

7.8 A secure encryption scheme based on a PRG

We give a stateful encryption scheme based on a PRG $g : \{0,1\}^\ell \to \{0,1\}^L$. Let us call the scheme $\Pi[g]$. It is defined as follows:

> We let Keys $= \{0,1\}^\ell$, with the uniform distribution.
>
> The state is a number s, where $0 \le s < L$. The initial state is 0. The encryption of x under key a and current state s is $\langle s,\ g(a)[s..t] \oplus x \rangle$, where $t = s + |x| - 1$ and the bits of $g(a)$ are numbered starting from 0, assuming that $s + |x| < L$. If, instead, $s + |x| \ge L$, then $\Pi[g]$ refuses to encrypt x, returning a distinguished message ε as an indication of this.
>
> Decryption proceeds in the obvious way.

We have left as a homework for you to prove the following theorem.

Theorem 7.8.1 *Let A be an adversary that (t, q, μ, ϵ)-breaks the encryption scheme $\Pi[g]$, in the sense of indistinguishability. Then there exists an adversary A' that (t', μ', ϵ')-break the PRG g, where $t' = t + O(\mu)$, $\mu' = \mu$, and $\epsilon' = e$.*

The hidden constant inside the big-O is just some absolute constant which depends only on details of the model of computation.