A Concrete Security Treatment of Symmetric Encryption *

M. BELLARE† A. DESAI† E. JOKIPII† P. ROGAWAY‡

Abstract

We study notions and schemes for symmetric (ie. private key) encryption in a concrete security framework.

We give four different notions of security against chosen plaintext attack and analyze the concrete complexity of reductions among them, providing both upper and lower bounds, and obtaining tight relations. In this way we classify notions (even though polynomially reducible to each other) as stronger or weaker in terms of concrete security.

Next we provide concrete security analyses of methods to encrypt using a block cipher, including the most popular encryption method, CBC. We establish tight bounds (meaning matching upper bounds and attacks) on the success of adversaries as a function of their resources.

1 Introduction

An encryption scheme enables Alice to send a message to Bob in such a way that an adversary Eve does not gain significant information about the message content. This is the classical problem of cryptography. It is usually considered in one of two settings. In the *symmetric* (private-key) one, encryption and decryption are performed under a key shared by the sender and receiver. In the *asymmetric* (public-key) setting the sender has some public information and the receiver holds some corresponding secret information.

In this paper we have two goals. The first is to study notions of symmetric encryption in the framework of concrete security. This means we will look at the concrete complexity of reductions between different notions. We want to prove both upper and lower bounds. In this way we can establish tight relations between the notions and can compare notions (even though polynomially reducible to each other) as stronger or weaker.

The second goal is to provide a concrete security analysis of some specific symmetric encryption schemes. One of the schemes we consider (CBC encryption) is in pervasive use, and yet has never received any formal analysis (concrete or otherwise) in the tradition of provable security. We want to remedy this. Once again the goal is to find tight bounds on the success probability of an adversary as a function of the resources she expends. This involves proving both an upper bound and a matching lower bound.

1.1 Background and Motivation

The pioneering work of Goldwasser and Micali [11] was the first to introduce formal notions of security for encryption. Specifically, they presented two notions of security for asymmetric encryption, "semantic security" and "polynomial security," and proved them equivalent with respect to polynomial-time reductions. Micali, Rackoff and Sloan [17] showed that (appropriate versions of) these notions were also equivalent to another notion, suggested by Yao [20]. A uniform complexity treatment of notions of asymmetric encryption is given by Goldreich [8]. Some adaptations of these notions to the symmetric setting are presented by Luby in [15, Chapters 11–12].

Goldwasser and Micali [11] also specified an asymmetric encryption scheme whose security (in the senses above) is polynomial-time reducible from quadratic residuosity. Subsequently many other schemes have emerged, based on various hard problems.

CONCRETE SECURITY. The viewpoint in all the works above is that two notions of security are equivalent if there is a polynomial-time reduction between them; and a scheme is declared provably secure if there is some polynomial-time reduction from a hard problem to it. These are certainly basic questions, but we believe that, once the answers are known, it is important to classify notions and schemes in a more precise way.

To make an analogy, caring only about polynomial-time reducibility in cryptography is a bit like caring only whether a computational problem is or is not in P. Yet we know there are a lot of interesting questions (including most of the field of algorithms, and much of complexity theory) centered around getting further information about problems already known to be in P. Such information helps

* Extended abstract. Full version available as [4].

† Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. E-Mail: {mihir, adesai, ej}@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/{mihir, adesai, ej}. Supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering.

‡ Dept. of Computer Science, Engineering II Bldg., University of California at Davis, Davis, CA 95616, USA. E-mail: rogaway@cs.ucdavis.edu. URL: http://wwwcsif.cs.ucdavis.edu/ rogaway. Supported in part by NSF CAREER Award CCR-9624560.

to better understand the problem and is also essential for practical applications.

Paying attention to the concrete complexity of polynomially-equivalent notions in cryptography has similar payoffs. In particular, when reductions are not security-preserving it means that one must use a larger security parameter to be safe, reducing efficiency. Thus, in the end, one pays for inefficient reductions in either assurance or running time.

Our approach for doing concrete security is that of [5, 6], wherein one parameterizes the resources involved and measures adversarial success by an explicit function on them. The approach is non-asymptotic and applicable to functions with a finite domain.

We will be concerned not only with proving security by exhibiting concrete bounds, but also with showing that these bounds are the best possible, which is done by exhibiting matching attacks. Again we follow works like [3, 5], who did this for certain message authentication schemes.

Though this paper is concerned with concrete security for symmetric encryption we believe that, in general, concrete security is one of the major emerging avenues for productive research in theoretical cryptography.

1.2 Notions of Security

We will consider four notions of security for symmetric encryption and examine the complexity of reductions between them. The first notion, which we call "real-or-random indistinguishability" is new, and the second, "left-or-right indistinguishability" is a variant of it. The next two notions, "find-then-guess security" and "semantic security" are adaptations of the notions of [11] to the symmetric setting.[1] Our goal, in all the notions, is to model chosen-plaintext attacks.

As indicated above, our approach to concrete security is via parameterization of the resources of the adversary A. We distinguish between A's running time, t (by convention, we include in this the space for A's program); the amount of ciphertext A sees, μ; and the number of queries, q, made by A to an encryption oracle. (To model a chosen-plaintext attack we must give the adversary the ability to see ciphertexts. In the public key setting she can create them herself given the public key, but in the symmetric key setting the encryption key is secret so we must modify the model and provide the adversary with an oracle for the encryption function. The presence of the encryption oracle is one reason it would be untrue to regard the notion of symmetric encryption as a special case of asymmetric encryption.) With an eye towards practical applications, it is important to treat all of these resources separately. (Previous works would neglect q and μ, since they are bounded by t. But as resources they are very different, because, typically, obtaining legitimate ciphertexts is more problematic than performing local computation.) We thus get a notion of $(t, q, \mu; \epsilon)$-security, meaning the success probability of an adversary is at most ϵ when its resources are as indicated. Of course how the success probability is measured varies across the four different notions.

[1] In [11] the term "polynomial security" is used for the notion analogous to what we call "find-then-guess security."

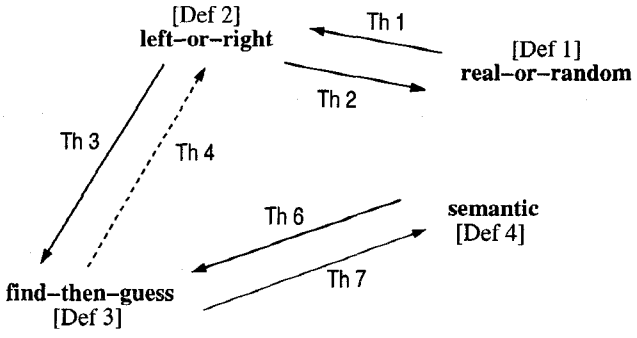

Figure 1: *Relating the notions. A solid line from notion **A** to notion **B** means that there is a security-preserving reduction from **A** to **B** A broken line indicates that the reduction is not security-preserving.*

1.3 Reductions Among the Notions

We show that real-or-random indistinguishability and left-or-right indistinguishability are equivalent, up to a small constant factor in the reduction. (That is, we have security-preserving reductions between them.) We also show a security-preserving reduction from these notions to find-then-guess security. However, the reduction from find-then-guess security to left-or-right (or real-or-random) indistinguishability is not security-preserving. However, we show that the reduction we give is tight; one cannot hope to do better.

We show a security-preserving reduction from semantic security to find-then-guess. In the other direction the complexity of our reduction depends on the time complexity of the "information function," f (representing the property of the plaintext semantic security is talking about) and "message-space sampling algorithm." The reduction is good if these complexities are low.

From the above results it is clear that when one wants to prove the security of some encryption scheme Π it is best to give a tight reduction from real-or-random indistinguishability or left-or-right indistinguishability, since that implies good reductions to the rest. A summary of the reductions is given in Figure 1.

Although concrete security has been considered before in the context of scheme analysis this is the first work that

considers it also for the purpose of relating different notions of security. That is, this is the first time *notions* are classified as weaker or stronger according to the complexity of the reductions between them.

Actually these results extend easily to the asymmetric setting. We focus on the symmetric mainly because that's the domain in which lie the schemes we want to analyze.

1.4 Security of Encryption Schemes

We analyze the security of some classic symmetric encryption schemes. Specifically, we look at two different modes of encryption with a block cipher (eg., DES): CBC (Cipher Block Chaining mode); and XOR (sometimes called counter mode). For the latter we look at both a probabilistic and a stateful version.

In these schemes the underlying primitive is a pseudorandom function (PRF) or pseudorandom permutation (PRP) family F in which a particular function F_a, specified by a key a, maps l-bits to L-bits for fixed l, L. (For permutations, $l = L$.) To encrypt a message the applications of F_a are iterated in some scheme-dependent way. We wish to see how the security of the encryption scheme depends on the assumed security of the PRF family. We define the concrete security of PRF and PRP families as in [6], via parameterization of the time t', number of oracle queries q', and maximum advantage ϵ' of the distinguisher. The question then is: assuming F is $(t', q'; \epsilon')$-secure as a PRF family, what are values of t, q, μ, ϵ such that the encryption scheme is $(t, q, \mu; \epsilon)$-secure? We seek upper and lower bounds. (The latter represent the best known attacks.)

For the stateful XOR scheme we show that the scheme is $(t, q, \mu; \epsilon)$ secure for $\epsilon = 2\epsilon'$, $\mu = q'l$, and t differing from t' by only an additive amount, meaning this scheme is about as good a scheme as one can possibly hope to get. For the probabilistic XOR scheme we show that the scheme is $(t, q, \mu; \epsilon)$ secure for $\epsilon = 2\epsilon' + \mu(q-1)/(L2^l)$ and $\mu = q'L$ and t as before. For CBC, the parameter values are $\epsilon = 2\epsilon' + (\mu^2 - \mu l)/(l^2 2^l)$ and $\mu = q'l$. In all cases, we show that these results are tight, up to a constant. We conclude that stateful XOR, based on a finite PRF, has the best security.

In all the above the security is in the sense of left-or-right indistinguishability. From what we said before this gives the other three notions with comparable bounds.

1.5 More history

We have already mentioned the most important related work, namely [11]. Here we provide some more detailed comparisons and histories and also discuss other work.

Since our results imply that the notions we consider are equivalent under polynomial time reductions, they can be viewed, at one level, as providing the analogue of [11] for the symmetric case.

In treating the asymmetric setting, [8] says that the symmetric case can be dealt with similarly. One ingredient missing in this view is that to model chosen-plaintext attack one must, in the symmetric setting, supply the adversary with some means to encrypt. We extend polynomial and semantic security by providing the adversary with an encryption oracle.

Stronger notions of asymmetric encryption than those of [11, 17] have also appeared, including [19, 7], but our concern here is restricted to preserving privacy under chosen-plaintext attack.

Luby [15] defines what is essentially find-then-guess security for symmetric encryption, and he mentions encryption using a pseudorandom function family whose output length is the number of bits you wish to encrypt. His treatment pays some attention to the efficiency of reductions (though he does not concern himself with concrete security to the same extent that we do).

Curiously, some early works had a more concrete treatment: in the asymmetric encryption arena, Alexi et. al. [1] were careful to specify the complexity of their reductions, a habit many later works unfortunately dropped.

The construction of a pseudorandom generator from a one-way function [13] provides a solution for symmetric encryption starting from a one-way function. In the current work existence is not the issue; we are interested in concrete security and the analysis of some particular schemes.

A concrete security analysis of the CBC MAC is provided in [6]. (The CBC MAC should not be confused with CBC encryption: The former is a message authentication code.) We build on their techniques, but those techniques don't directly solve the problems here. CBC mode encryption is standardized in [2, 14, 18].

1.6 This abstract

This is an extended abstract that contains only definitions, scheme descriptions and result statements. Proofs have been omitted due to page limits. A full version of our paper, containing all proofs, can be found as [4].

2 Notions of Encryption

For all complexity measures fix some probabilistic RAM model. We adopt the convention that "time" refers to the actual running time plus the size of the code (relative to some fixed programming language). Oracle queries are answered in unit time.

If $A(\cdot, \cdot, \ldots)$ is any probabilistic algorithm then $a \leftarrow A(x_1, x_2, \ldots)$ denotes the experiment of running A on inputs $x_1, x_2, \ldots$ and letting a be the outcome, the probability being over the coins of A. Similarly, if A is a set then $a \leftarrow A$ denotes the experiment of selecting a point uniformly from A and assigning a this value.

2.1 Syntax of Encryption Schemes

Let Coins be a synonym for $\{0,1\}^\infty$ (the set of infinite strings). Let $\mathsf{MsgSp} \subseteq \{0,1\}^*$ be a set, the *message space*, for which $x \in \mathsf{MsgSp}$ implies $x' \in \mathsf{MsgSp}$ for every x' of the same length as x. Let $\mathsf{KeySp} \subseteq \{0,1\}^*$ be set, denoting the *key space*. Let $\mathsf{CipherSp} = \{0,1\}^*$.

STATELESS ENCRYPTION. A (probabilistic, stateless, symmetric) encryption scheme, $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ is a three-tuple of algorithms:

$$\begin{array}{rcl} \mathcal{E} & : & \mathsf{KeySp} \times \mathsf{MsgSp} \times \mathsf{Coins} \to \mathsf{CipherSp} \\ \mathcal{D} & : & \mathsf{KeySp} \times \mathsf{CipherSp} \to \mathsf{MsgSp} \cup \{\perp\} \\ \mathcal{K} & : & \mathsf{Coins} \to \mathsf{KeySp} \end{array}$$

Algorithm $\mathcal{E}$ is called the *encryption* algorithm; $\mathcal{D}$ is the *decryption* algorithm; and $\mathcal{K}$ is the *key generator.* We require that for all $a \in \mathsf{KeySp}$, $x \in \mathsf{MsgSp}$, and $r \in \mathsf{Coins}$, $\mathcal{D}(a, \mathcal{E}(a,x,r)) = x$. We usually write the first argument to $\mathcal{E}$ and $\mathcal{D}$, the key, as a subscript. We call $\mathcal{E}_a(x,r)$ the encryption of *plaintext* x under key a and coins r, or more succinctly, the *ciphertext*. We call $\mathcal{D}_a(y)$ the decryption of ciphertext y under key a. Usually we omit mention of the argument to $\mathcal{K}$, thinking of $\mathcal{K}$ as a probabilistic algorithm, or else the induced probability space. Similarly, we often omit mention of the final argument to $\mathcal{E}$, thinking of $\mathcal{E}_a$ as a probabilistic algorithm, or thinking of $\mathcal{E}_a(x)$ as the induced probability space. We intend $\mathcal{D}_a(y) = \perp$ to be used in the case that y is not the encryption of any string x under key a.

For an encryption scheme to be useful, $\mathcal{E}$, $\mathcal{D}$, and $\mathcal{K}$ should be efficiently computable functions, but the notion of security makes no formal demands in this regard.

STATEFUL ENCRYPTION. We also consider *stateful* encryption schemes, in which the ciphertext is a function of some information, such as a counter, maintained by the encrypting party and updated with each encryption. Formally such a scheme has the same syntax as before except that

$$\mathcal{E} \ : \ \mathsf{KeySp} \times \mathsf{MsgSp} \times \mathsf{St} \times \mathsf{Coins} \to \mathsf{St} \times \mathsf{CipherSp} \ ,$$

were $\mathsf{St} \subseteq \{0,1\}^*$ is the set of possible *states*, containing a distinguished state, the empty string, ε, which we call the *initial state.* Let $\mathcal{E}^i$ $(i = 1, 2)$ denote the i-th component of $\mathcal{E}$. The *ciphertext* is now (the output of) $\mathcal{E}^2$, while $\mathcal{E}^1$ is an updated state, stored by the sender, and used as the third argument for the next application of the encryption function. Note that encryption becomes stateful but decryption does not.

2.2 Four Notions of Security

We now give four notions for security, each modeling chosen-plaintext attack. In each case, we allow the adversary access to an encryption oracle in some form; this is one feature distinguishing these definitions from previous ones. We will describe our definitions for stateless encryption schemes and later indicate how to modify them for stateful ones.

REAL-OR-RANDOM. The idea is that an adversary cannot distinguish the encryption of text from the encryption of an equal-length string of garbage. (By transitivity, the adversary cannot distinguish from each other the encryption of any two equal-length strings.) The formalization considers two different games. In Game 1 we start by choosing a random key $a \leftarrow \mathcal{K}$. Then the adversary is then given an oracle which, when asked a string $x \in \mathsf{MsgSp}$, responds with a (random) encryption of x under key a. In Game 2 we start by choosing a random key $a \leftarrow \mathcal{K}$. Then the adversary is given an oracle which, when asked a string $x \in \mathsf{MsgSp}$, responds with a (random) encryption (under key a) of a random string of length $|x|$. The encryption scheme is "good" if no "reasonable" adversary cannot obtain "significant" advantage in distinguishing Games 1 and 2.

Definition 1 [Real-or-random] *Encryption scheme $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ is said to be $(t, q, \mu; \epsilon)$-secure, in the real-or-random sense, if* $\mathsf{Adv}^{\mathrm{rr}}_A \stackrel{\mathrm{def}}{=}$

$$\Pr\left[a \leftarrow \mathcal{K} : \ A^{\mathcal{E}_a(\cdot)} = 1\right] - \Pr\left[a \leftarrow \mathcal{K} : \ A^{\mathcal{E}_a(\$^{|\cdot|})} = 1\right]$$

is $\leq \epsilon$, for any adversary A which runs in time at most t, makes at most q oracle queries, these totaling at most μ bits.

The notation $A^{\mathcal{E}_a(\cdot)}$ indicates A with an oracle which, in response to a query x, returns $y \leftarrow \mathcal{E}_a(x)$. (Meaning it picks a random string r and returns $\mathcal{E}_a(x,r)$. A new random string is chosen for each invocation of the oracle.) The notation $A^{\mathcal{E}_a(\$^{|\cdot|})}$ indicates A with an oracle which, in response to a query x, chooses $x' \leftarrow \{0,1\}^{|x|}$ and then returns $y \leftarrow \mathcal{E}_a(x')$.

LEFT-OR-RIGHT. We again consider two different games. In either game a query is a pair (x_1, x_2) of equal-length strings from MsgSp. In either game we start by choosing a random key $a \leftarrow \mathcal{K}$ and fixing this key for the duration of the game. In Game 1, an oracle receiving (x_1, x_2) responds with with a random sample from $\mathcal{E}_a(x_1)$. In Game 2 it responds with a random sample from $\mathcal{E}_a(x_2)$. Thus, Game 1 provides a "left" oracle and Game 2 provides a "right" oracle. We consider an encryption scheme to be "good" if "reasonable" adversary cannot obtain "significant" advantage in distinguishing Games 1 and 2.

Definition 2 [Left-or-Right] *Encryption scheme $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ is said to be $(t, q, \mu; \epsilon)$-secure, in the left-or-right sense, if for any adversary A which runs in time at most t and asks at most q queries, these totaling at most μ bits,*[2]

$$\mathsf{Adv}^{\mathrm{lr}}_A \ \stackrel{\mathrm{def}}{=} \ \Pr\left[a \leftarrow \mathcal{K} : \ A^{\mathcal{E}_a(\mathsf{left}(\cdot,\cdot))} = 1\right]$$

[2] We define the length of an oracle query (x_1, x_2) to be $|x_1| = |x_2|$.

$$- \Pr\left[a \leftarrow \mathcal{K} : \ A^{\mathcal{E}_a(\mathsf{right}(\cdot,\cdot))} = 1\right]$$

is $\leq \epsilon$.

The notation $A^{\mathcal{E}_a(\mathsf{left}(\cdot,\cdot))}$ indicates A with an oracle which, in response to query (x_1, x_2), returns $y \leftarrow \mathcal{E}_a(x_1)$. The notation $A^{\mathcal{E}_a(\mathsf{right}(\cdot,\cdot))}$ indicates A with an oracle which, in response to query (x_1, x_2), returns $y \leftarrow \mathcal{E}_a(x_2)$.

FIND-THEN-GUESS. This is an adaptation of the notion of polynomial security as given in [11, 17]. We imagine an adversary A that runs in two stages. During the adversary's find stage she endeavors to come up with a pair of equal-length messages, x_0 and x_1, whose encryptions she wants to try to tell apart. She also retains some state information s that she may want to preserve to help her later. In the adversary's guess stage she is given a random ciphertext y for one of the plaintexts x_0, x_1, together with the state information s. The adversary "wins" if she correctly identifies which plaintext goes with y. The encryption scheme is "good" if "reasonable" adversaries can't win significantly more than half the time.

Definition 3 [Find-then-Guess] *Encryption scheme* $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ *is said to be* $(t, q, \mu; \epsilon)$*-secure, in the find-then-guess sense, if* $\mathsf{Adv}^{\mathrm{fg}}_A \stackrel{\text{def}}{=}$

$$2 \cdot \Pr\Big[a \leftarrow \mathcal{K};\ (x_0, x_1, s) \leftarrow A^{\mathcal{E}_a(\cdot)}(\mathsf{find});\ b \leftarrow \{0,1\};\ y \leftarrow \mathcal{E}_a(x_b) :\ A^{\mathcal{E}_a(\cdot)}(\mathsf{guess}, y, s) = b\Big] - 1$$

is $\leq \epsilon$ *for any adversary* A *which runs in time at most* t *and asks at most* q *queries, these totaling at most* μ *bits.*

It is understood that, above, one demands $|x_0| = |x_1|$. The multiplication by 2 and subtraction by 1 are just scaling factors, to make a numeric value of 0 correspond to no advantage and a numeric value of 1 correspond to perfect advantage.

SEMANTIC. Goldwasser and Micali [11] explain semantic security by saying that whatever can be efficiently computed about the plaintext given the ciphertext can also be computed in the absence of the ciphertext. We adapt the formalizations of [11, 17] to the symmetric setting. Let $f : \mathsf{MsgSp} \to \{0,1\}^*$ be some function of the plaintext x. The function represents the information about x that the adversary is trying to figure out. Endow MsgSp with a probability distribution. More specifically, for any integer m, an m-distribution on the message space is a collection $\mathcal{M} = \{\mathcal{M}_\gamma\}_{\gamma \in \{0,1\}^{\leq m}}$ of probability distributions over MsgSp, indexed by strings $\gamma \in \{0,1\}^{\leq m}$. We assume each distribution is *valid*, meaning that for all γ, all strings in $\mathcal{M}_\gamma$ with non-zero probability have the same length, and this length is at most m. Let $p^*_{f,\mathcal{M}_\gamma} = \max_{y^*}\{\Pr[x \leftarrow \mathcal{M}_\gamma : f(x) = y^*]\}$. This is the probability of the most likely $f(\cdot)$-value.

Our adversary will run in two stages. During the adversary's select stage it endeavors to come up with an advantageous distribution $\mathcal{M}_\gamma$. In the adversary's predict stage it is given a random ciphertext y for a plaintext x chosen according to the distribution $\mathcal{M}_\gamma$ and it wants to guess $f(x)$. An encryption scheme is semantically secure for function f and distributions $\mathcal{M}$ if no reasonable adversary A can guess $f(x)$ with probability significantly better than $p^*_{f,\mathcal{M}_\gamma}$.

Previous formalizations required the condition to hold for all functions f. In our concrete treatment both the function f and the probability distribution $\mathcal{M}$ become parameters, so that we can measure how well particular properties of a plaintext are protected under particular distributions.

Definition 4 [Semantic] *Let* f: MsgSp $\to \{0,1\}^*$ *be a function and let* $\mathcal{M} = \{\mathcal{M}_\gamma\}_{\gamma \in \{0,1\}^{\leq m}}$ *be an* m*-distribution on* MsgSp. *Encryption scheme* $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ *is said to be* $(t, q, \mu; \epsilon)$*-secure in the semantic sense, for* f *over* $\mathcal{M}$*, if*

$$\mathsf{Adv}^{\mathrm{sm}}_A(f, \mathcal{M}) \stackrel{\text{def}}{=} \mathbf{E}\Big[a \leftarrow \mathcal{K};\ (\gamma, s) \leftarrow A^{\mathcal{E}_a(\cdot)}(\mathsf{select}) : \alpha(a, \gamma, s)\Big] \leq \epsilon\ ,\ \textit{where}$$

$$\alpha(a, \gamma, s) = \Pr\Big[x \leftarrow \mathcal{M}_\gamma;\ y \leftarrow \mathcal{E}_a(x) : A^{\mathcal{E}_a(\cdot)}(\mathsf{predict}, y, s) = f(x)\Big] - p^*_{f,\mathcal{M}_\gamma}\ ,$$

for any adversary A *that runs in time at most* t*, and makes at most* q *oracle queries, these totaling at most* μ *bits.*

MODIFYING THE DEFINITIONS FOR THE STATEFUL CASE. Definitions of security for stateful encryption schemes are obtained by modifying the above definitions in the natural way, adjusting how one answers oracle queries. For example, in Definition 1, $A^{\mathcal{E}_a(\cdot)}$ now means A with an oracle that maintains a state σ, initially ε. Upon receiving a query x it picks coins r and sets (σ', y) to be $\mathcal{E}_a(x, \sigma, r)$. It returns y as the answer to the oracle query and updates the state via $\sigma \leftarrow \sigma'$. Notice that the ciphertext (meaning y) is returned, but the updated state is not. (Thus we are abusing notation when we write $A^{\mathcal{E}_a(\cdot)}$; we ought to write $A^{\mathcal{E}^2_a(\cdot)}$.) Notice that the encryption oracles now have "memory": between invocations, the state is modified and retained. The notation $A^{\mathcal{E}_a(\$^{|\cdot|})}$ can be similarly re-interpreted, and the same approach applies to the other three definitions.

ASYMPTOTIC DEFINITIONS. Our definitions are easily extended to the standard asymptotic framework by simply saying that a scheme is secure, in a given sense, if the advantage of any polynomial time adversary is negligible, as a function of an underlying security parameter on which the scheme now depends. The above formulations just enable us to make more concrete statements.

3 Reductions Among the Notions

Here we look at the reductions among the different notions of security. We look at both upper bounds and lower bounds. The proofs of these results are in [4].

Because we are paying attention to concrete security bounds, we can use our results to decide how strong is a notion of security relative to other notions to which it is polynomially equivalent. This information is useful because it helps us identify the most desirable starting points for reductions. We implicitly use this information in Section 4 when we demonstrate the security of schemes via reductions from left-or-right indistinguishability.

In the theorems below, c is an absolute constant that depends only on details of the underlying model of computation. The first two theorems say that our first two notions, left-or-right indistinguishability and real-or-random indistinguishability, are of essentially equivalent strength.

Theorem 1 [Real-or-random implies left-or-right] *For some constant $c > 0$, if encryption scheme $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ is $(t_1, q_1, \mu_1; \epsilon_1)$-secure in the real-or-random sense then it is $(t_2, q_2, \mu_2; \epsilon_2)$-secure in the left-or-right sense, where $t_2 = t_1 - c \cdot \mu_2$ and $q_2 = q_1$ and $\mu_2 = \mu_1$ and $\epsilon_2 = 2\epsilon_1$.*

Theorem 2 [Left-or-right implies real-or-random] *For some constant $c > 0$, if encryption scheme $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ is $(t_2, q_2, \mu_2; \epsilon_2)$-secure in the left-or-right sense then it is $(t_1, q_1, \mu_1; \epsilon_1)$-secure in the real-or-random sense, where $t_1 = t_2 - c \cdot \mu_1$ and $q_1 = q_2$ and $\mu_1 = \mu_2$ and $\epsilon_1 = \epsilon_2$.*

Left-or-right indistinguishability and real-or-random indistinguishability constitute a *stronger* notion of security than the traditional find-then-guess notion. Intuitively, the adversary's job is harder with find-then-guess because it has to single out a single message pair on which to perform. This is illustrated by Theorems 3 and 4 and Proposition 5.

The first theorem says that a scheme with a certain security in the left-or-right sense has essentially the same security in the find-then-guess sense.

Theorem 3 [Left-or-right implies find-then-guess] *For some constant $c > 0$, if encryption scheme $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ is $(t_2, q_2, \mu_2; \epsilon_2)$-secure in left-or-right sense then it is $(t_3, q_3, \mu_3; \epsilon_3)$-secure in the find-then-guess sense, where $t_3 = t_2 - c \cdot \mu_3$ and $q_3 = q_2$ and $\mu_3 = \mu_2$ and $\epsilon_3 = \epsilon_2$.*

The next theorem says that if a scheme has a certain security in the find-then-guess sense, then it is secure in the left-or-right sense, but the security shown is quantitatively lower.

Theorem 4 [Find-then-guess implies left-or-right] *For some constant $c > 0$, if encryption scheme $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ is $(t_3, q_3, \mu_3; \epsilon_3)$-secure in the find-then-guess sense then it is $(t_2, q_2, \mu_2; \epsilon_2)$-secure in the left-or-right sense, where $t_2 = t_3 - c \cdot \mu_2$ and $q_2 = q_3$ and $\mu_2 = \mu_3$ and $\epsilon_2 = q_2\epsilon_3$.*

The following proposition says that the drop in security above is not due to any weakness in the reduction but is intrinsic— we present a scheme having a higher security in the find-then-guess sense than in the left-or-right sense, with the gap being the same as in the theorem above. Obviously we can make no such statement if there are no secure encryption schemes around at all, so the theorem assumes there exists a secure scheme, and then constructs a different scheme exhibiting the desired gap.

In the following think of ϵ' as very small (essentially zero). The constructed scheme Π' can be broken with probability $\epsilon_2 = 0.632$, using q queries, in the left-or-right sense, meaning it is completely insecure under this notion. However, the probability of breaking it (with comparable resources) in the find-then-guess sense is $\epsilon_3 \approx 1/q$. The probabilities obey the relation $q\epsilon_3 = \Theta(\epsilon_2)$, showing that Theorem 4 is essentially tight. Furthermore, if one allows the scheme to be stateful, one can make ϵ_2 exactly one, so that $q\epsilon_3 \approx \epsilon_2$.

Proposition 5 [Left-or-right is stronger than find-then-guess] *There is a constant $c > 0$ such that the following is true. Suppose there exists a stateless encryption scheme, over a message space containing $\{0,1\}$, that is $(t', q, \mu; \epsilon')$-secure in the find-then-guess sense. Then there exists a stateless encryption scheme Π' which is $(t_2, q, q; \epsilon_2)$-breakable in the left-or-right sense and $(t_3, q, \mu; \epsilon_3)$-secure in the find-then-guess sense, where $\epsilon_2 = 0.632$ and $\epsilon_3 = \epsilon' + 1/q$ and $t_2 = cq$ and $t_3 = t'$. Furthermore there exists a stateful encryption scheme Π'' which has the same features except that $\epsilon_2 = 1$.*

Semantic security is too complex to make it a good starting point for proving schemes secure. Still, as the next theorem indicates, it is nice that there is a strong reduction from semantic security to find-then-guess security. Notice that for this only requires semantic security to hold for a particular and simple function, the identity function, and a particular and simple distribution over the message space. This theorem is implicit in [11] for the asymmetric setting and their proof is easily adapted to the symmetric setting.

Theorem 6 [Semantic implies find-then-guess] *Let f be the identity function. For any pair $\gamma = (x_0, x_1)$ of equal length strings in* MsgSp *let $\mathcal{M}_\gamma$ be the distribution assigning probability $1/2$ to each of x_0 and x_1, and probability 0 to all other strings. Let $\mathcal{M}_\gamma$ be arbitrarily defined when γ does not have this form. Let $\mathcal{M} = \{\mathcal{M}_\gamma\}_{\gamma \in \{0,1\}^{\leq t_4}}$. Then for some constant $c > 0$, if Π is $(t_4, q_4, \mu_4; \epsilon_4)$-secure in the semantic sense for f over $\mathcal{M}$, then it is $(t_3, q_3, \mu_3; \epsilon_3)$-secure in the find-then-guess sense, where $t_3 = t_4 - c \cdot \mu_3$ and $q_3 = q_4$ and $\mu_3 = \mu_4$ and $\epsilon_3 = 2\epsilon_4$.*

Combining this with Theorem 4 yields a reduction from security in the semantic sense to security in the left-or-right

sense, but this reduction inherits the security loss of the reduction of Theorem 4. As before it turns out this loss is inherent: security in the left-or-right sense is a stronger notion. The example to see this is essentially the same as that in the proof of Proposition 5 but the setup becomes more complicated. We do not discuss it further here.

In the other direction, the time complexity of sampling the the message space and computing the function f come into the picture. Let $T_{\mathcal{M}}(\cdot)$ be a function taking $|\gamma|$ as input and returning a bound on the time to sample from $\mathcal{M}_\gamma$. Let $T_f(\cdot)$ denote the time to compute $f(x)$ given x, measured as a function of $|x|$. Both time functions are assumed monotone.

Theorem 7 [Find-then-guess implies semantic] *There is a constant $c > 0$ such that the following is true. Let f be a function that is computable in time $T_f(\cdot)$ and let $\mathcal{M}$ be a valid m-distribution over* MsgSp *sampleable in time $T_{\mathcal{M}}(\cdot)$. If $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ is $(t_3, q_3, \mu_3; \epsilon_3)$-secure in the find-then-guess sense then it is $(t_4, q_4, \mu_4; \epsilon_4)$-secure in the semantic sense for f over $\mathcal{M}$, where $t_4 = t_3 - 2T_{\mathcal{M}}(m) - T_f(m) - c \cdot \mu_4$ and $q_4 = q_3$ and $\mu_4 = \mu_3$ and $\epsilon_4 = 2\epsilon_3$.*

Notice that the larger the functions $T_{\mathcal{M}}(\cdot), T_f(\cdot)$, the less the semantic security for f over $\mathcal{M}$ as given by Theorem 7. Does this reflect a reality? That is, would we expect the adversary might have an easier time figuring out some complex property of the plaintext than figuring out simple properties of the plaintext? Perhaps. In any case, these theorems are most useful when the information function f is simple, like the XOR of all the bits.

In earlier work [11, 17, 8] no restriction was made on the complexity of f; it was even allowed to be uncomputable. Clearly semantic security against such very complex functions does not follow from Theorem 7. However it is seems possible to do a different reduction by using the techniques of [8]. Here, the complexity of f would not enter (though the complexity of sampling $\mathcal{M}_\gamma$ would still matter). The dependencies on other parameters would be increased. Thus the theorem would be useful in talking about complex functions f, but less useful than Theorem 7 in talking about simple functions. We do not pursue this more at the moment because, as we have indicated above, other notions of security are more suitable targets than semantic security as targets for actual schemes to meet.

Putting things together, showing an encryption scheme left-or-right secure or real-or-random secure implies tight reductions to all other notions (modulo the technical restriction on the complexity of f and $\mathcal{M}$ for semantic security). Showing an encryption scheme find-then-guess secure or semantically secure does not. Thus, if the bounds are equal, it is better to demonstrate security with respect to one of the first two notions, since that immediately translates into equally-good bounds for the other notions.

4 Analysis of Some Schemes

Next we turn to analyzing schemes for symmetric encryption. All these schemes are based on finite pseudorandom functions, a concrete security version of the original notion of pseudorandom functions [10] introduced by [6]. We thus begin with some necessary definitions, following the latter paper. Proofs of results given in this section are in [4].

4.1 Finite PRFs and PRPs

A *function family* is a multiset F of functions where all of the functions in F have the same domain and range. Usually the domain is $\{0,1\}^l$ and the range is $\{0,1\}^L$ for some l, L called, respectively, the input length and the output length. We assume that each key a from some set K specifies a function F_a: $\{0,1\}^l \to \{0,1\}^L$ from F. Usually K is the set of all strings of some fixed length k. We write $f \leftarrow F$ to denote the operation of selecting a function at random from F according to the distribution given by picking a random $a \leftarrow K$ and assigning $f = F_a$.

For a function family F to be accessible to applications we usually want that, given a, one can easily compute F_a. But we make nor formal requirements in this regard, and indeed it is useful to think about "inaccessible" function families, as below.

We let $R_{l,L}$ be the function family consisting of all functions from the set of l-bit strings to the set of L-bit strings. (The key a can be viewed as the entire description of the function.) With l, L understood, we write R instead of $R_{l,L}$. Thus $f \leftarrow R$ is the operation of selecting a random function from l-bits to L-bits. Similarly, we let P_l be the function family consisting of all permutations on l-bit strings. With l understood we write P instead of P_l.

Let F, G be families of functions with the same input and output lengths. consider an oracle algorithm, known as a *distinguisher*, that attempts to distinguish between the case where its oracle h is chosen randomly from F and the case where h is chosen randomly from G. Let $\mathsf{Dist}_D(F,G) =$

$$\Pr\left[h \leftarrow F:\ D^{h(\cdot)} = 1\right] - \Pr\left[h \leftarrow G:\ D^{h(\cdot)} = 1\right] .$$

A pseudorandom function family has the property that the input-output behavior of F_a "looks random" to someone who does not know the randomly selected key a. There are two notions of "looking random" that are important. The first is looking like a random function, the second is looking like a random permutation. Accordingly, we define

$$\mathsf{Adv}^{\mathrm{rf}}_D(F) = \mathsf{Dist}_D(F,R) \text{ and } \mathsf{Adv}^{\mathrm{rp}}_D(F) = \mathsf{Dist}_D(F,P) .$$

Definition 5 [Concrete security of PRF/PRP families, [6]] Function family F is said to be a $(t, q; \epsilon)$-secure PRF (resp. PRP) family if for any distinguisher D who makes at most q oracle queries and runs in time at most t it is the case that $\mathsf{Adv}^{\mathrm{rf}}_D(F) \le \epsilon$ (resp. $\mathsf{Adv}^{\mathrm{rp}}_D(F) \le \epsilon$).

Notice that unlike Luby and Rackoff [16], we measure the quality of a PRP family by the distance to the family of random permutations, not random functions. This is motivated by the fact that PRPs, as we define them, are better models for block ciphers, like DES, than PRFs. (Of course, the distinction is only in the concrete security, but that is indeed our concern.) Nonetheless, the following relation between the two notions is often enough:

Proposition 8 [PRPs are PRFs] *Suppose F is a $(t, q; \epsilon)$-secure PRP family with input and output length l. Then F is a $(t, q; \epsilon')$-secure PRF family, where $\epsilon' = \epsilon - q^2 2^{-l-1}$.*

The estimated cryptanalytic strength of specific block ciphers gives us estimates for values of t, q, ϵ for which a particular block cipher, eg. DES, may be viewed as a $(t, q; \epsilon)$-secure PRP family. Using the above proposition gives us the bounds by which it can be viewed as a $(t, q; \epsilon)$-secure PRF.

4.2 The XOR Schemes

Fix a function family F with input length l, output length L, and key length k. We let a denote the key shared between the two parties who run the encryption scheme. It will be used to specify the function $f = F_a$. In fact, all the schemes depend only only on this function, in the sense that they can be implemented given an oracle for the function. We let $R = R_{l,L}$.

There are two version of the XOR scheme— one stateless (randomized) and the other stateful (counter based and deterministic).

SPECIFICATIONS. The scheme XOR\$(F) = ($\mathcal{E}$-XOR\$, $\mathcal{D}$-XOR\$, $\mathcal{K}$-XOR\$) works as follows. The key generation algorithm $\mathcal{K}$-XOR\$ just outputs a random k-bit key a for the underlying PRF family F, thereby specifying a function $f = F_a$ of l-bits to L-bits. The message x to be encrypted is regarded as a sequence of L-bit blocks (padding is done first, if necessary), $x = x_1 \cdots x_n$. We define $\mathcal{E}$-XOR\$$_a(x) = \mathcal{E}$-XOR\$$^{F_a}(x)$ and $\mathcal{D}$-XOR\$$_a(z) = \mathcal{D}$-XOR\$$^{F_a}(z)$, where:

function $\mathcal{E}$-XOR\$$^f(x)$
(1) $r \leftarrow \{0,1\}^l$
(2) **for** $i = 1, \ldots, n$ **do** $y_i = f(r + i) \oplus x_i$
(3) **return** $r \,\|\, y_1 y_2 \cdots y_n$

function $\mathcal{D}$-XOR\$$^f(z)$
(1) Parse z as $r \,\|\, y_1 \cdots y_n$
(2) **for** $i = 1, \ldots, n$ **do** $x_i = f(r + i) \oplus y_i$
(3) **return** $x = x_1 \cdots x_n$

We call r the *nonce*. Addition, above, is modulo 2^l, and the result is encoded as an l-bit string in the usual way.

This scheme also has a stateful variant, XORC = ($\mathcal{E}$-XORC, $\mathcal{D}$-XORC, $\mathcal{K}$-XORC). Here the role of r is played by a l-bit counter, denoted ctr, that is initially -1 and increases after each encryption by the number of encrypted blocks. Note only the sender maintains the counter and it is output as part of the ciphertext. A restriction placed on the scheme is that the total number of encrypted blocks be at most 2^l.

The key generation algorithm $\mathcal{K}$-XORC is the same as before, meaning just outputs a random key a for the PRF family. With the same formatting conventions as above, we define $\mathcal{E}$-XORC$_a(x, ctr) = \mathcal{E}$-XORC$^{F_a}(x, ctr)$ and $\mathcal{D}$-XORC$_a(z) = \mathcal{D}$-XORC$^{F_a}(z)$, where:

function $\mathcal{E}$-XORC$^f(x, ctr)$
(1) **for** $i = 1, \ldots, n$ **do** $y_i = f(ctr + i) \oplus x_i$
(2) $ctr \leftarrow ctr + n$
(3) **return** $(ctr,\ ctr \,\|\, y_1 y_2 \cdots y_n)$

function $\mathcal{D}$-XOR\$$^f(z)$
(1) Parse z as $ctr \,\|\, y_1 \cdots y_n$
(2) **for** $i = 1, \ldots, n$ **do** $x_i = f(ctr + i) \oplus y_i$
(3) **return** $x = x_1 \cdots x_n$

FEATURES OF THE SCHEMES. Notice that decryption does not require the ability to invert $f = F_a$. Thus F_a need not be a permutation.

The XOR schemes have some computational advantages over the more common modes of operation. Namely, the F_a computations on different blocks can be done in parallel since the computation on a block is independent of the other blocks. This parallelizability advantage can be realized through either hardware or software support. Decryption does not have to be done in order if each block is tagged with its index. These schemes also support off-line processing, in the sense that the F_a computations can be done during idle times before the messages they are to be used with become available.

SECURITY OF XOR\$. We first derive a lower bound on the success of an adversary trying to break the XOR\$(F) scheme in the left-or-right sense. In the common cryptographic terminology, this means, simply, that we are providing an attack. The attack we specify is on the "ideal" scheme, namely the one where the underlying function f is truly random.

Proposition 9 [Lower bound on security of XOR\$ in random function model] *There is an adversary E for XOR\$($R_{l,L}$), in the left-or-right sense, who makes up to q queries, totaling at most μ bits, ($\mu q/L \leq 2^l$) and achieves* $\mathbf{Adv}^{\mathrm{lr}}_E \geq 0.316 \cdot [\mu \cdot (q-1)]/[L \cdot 2^l]$.

This is a "birthday" attack. It may be easier to gauge if we let $\bar{n} = \mu/(Lq)$ be the average number of blocks per query, so that $\mu = Lq \cdot \bar{n}$. Then we see that $\mathbf{Adv}^{\mathrm{lr}}_E = \Omega(q^2/2^l) \cdot \bar{n}$, a typical birthday behavior exhibiting a quadratic dependence on the number of queries.

Since we prove a lower bound in the random function model, we do not discuss the time used by E. However it is

clear from the strategy that the total time used by E would be just a little overhead besides the time for the oracle calls. This is true for all lower bounds and we won't mention it again.

Proposition 9 indicates that even when the underlying block cipher F is very good (it can't get better than truly random) the XOR scheme leaks some information as more and more data is encrypted. Next, we show that the above is essentially the best attack: one can't get a better advantage, up to a constant factor. The crucial point below is that the bound holds for *any* adversary.

Lemma 10 [Upper bound on security of XOR$ in random function model] *Let E be any adversary attacking $\mathrm{XOR\$}(R_{l,L})$ in the left-or-right sense, making at most q queries, totaling at most μ bits. Then $\mathrm{Adv}^{\mathrm{lr}}_E \le \delta_{\mathrm{XOR\$}} = [\mu(q-1)]/[L \cdot 2^l]$.*

Of course, an indication of security in the ideal model is not an indication of security when we use a block cipher. The "real-world" case however is easily derived from the above:

Theorem 11 [Security of XOR$ using a pseudorandom function] *There is a constant $c > 0$ such that the following is true. Suppose F is a $(t', q'; \epsilon')$-secure PRF family with input length l and output length L. Then for any q the $\mathrm{XOR\$}(F)$ scheme is $(t, q, \mu; \epsilon)$-secure in the left-or-right sense, for $\mu = q'L$ and $t = t' - c \cdot \frac{\mu}{L}(l + L)$ and $\epsilon = 2\epsilon' + \delta_{\mathrm{XOR\$}}$, where $\delta_{\mathrm{XOR\$}} \stackrel{\mathrm{def}}{=} [\mu(q-1)]/[L \cdot 2^l]$.*

SECURITY OF XORC. The stateful version of the scheme has better security. The adversary has *no* advantage in the ideal case:

Lemma 12 [Upper bound on security of XORC in random function model] *Let E be any adversary attacking $\mathrm{XORC}(R_{l,L})$ in the left-or-right sense, making at most q queries, totaling at most $\mu < L2^l$ bits. Then $\mathrm{Adv}^{\mathrm{lr}}_E = 0$.*

This translates into the following "real-world" security:

Theorem 13 [Security of XORC using a pseudorandom function] *There is a constant $c > 0$ such that the following is true. Suppose F is a $(t', q'; \epsilon')$-secure PRF family with input length l and output length L. Then for any q the $\mathrm{XORC}(F)$ scheme is $(t, q, \mu; \epsilon)$-secure in the left-or-right sense, for $\mu = \min(q'L, L2^l)$ and $t = t' - c \cdot (\mu/L)(l + L)$ and $\epsilon = 2\epsilon'$.*

4.3 The CBC Scheme

For the CBC scheme we require that $l = L$ (the input and output lengths of F are the same) and that each F_a be a permutation such that given a we can compute not only F_a but also F_a^{-1}. As far as security goes, however, we still view F a pseudorandom function family. Having stated the results for this case we will discuss what happens when F is a PRP family.

SPECIFICATION. The scheme $\mathrm{CBC\$}(F) = (\mathcal{E}\text{-CBC\$}, \mathcal{D}\text{-CBC\$}, \mathcal{K}\text{-CBC\$})$ has the same key generation algorithm as the previous schemes, meaning the key for encryption is the key a specifying $f = F_a$. The message x to be encrypted is regarded as a sequence of l bit blocks, $x = x_1 \ldots x_n$. We define $\mathcal{E}\text{-CBC\$}_a(x) = \mathcal{E}\text{-CBC\$}^{F_a}(x)$ and $\mathcal{D}\text{-CBC\$}_a(z) = \mathcal{D}\text{-CBC\$}^{F_a}(z)$, where:

function $\mathcal{E}\text{-CBC\$}^f(x)$
(1) $y_0 \leftarrow \{0,1\}^l$
(2) **for** $i = 1, \ldots, n$ **do** $y_i = f(y_{i-1} \oplus x_i)$
(3) **return** $y_0 \,\|\, y_1 y_2 \cdots y_n$

function $\mathcal{D}\text{-CBC\$}^f(z)$
(1) Parse z as $y_0 \,\|\, y_1 \cdots y_n$
(2) **for** $i = 1, \ldots, n$ **do** $x_i = f^{-1}(y_i) \oplus y_{i-1}$
(3) **return** $x = x_1 \ldots x_n$

The value y_0 is called *initial vector*, or *nonce*. See discussion below for the counter variant.

SECURITY OF CBC$. Birthday attacks remain possible even when the underlying block cipher is ideal:

Proposition 14 [Lower bound on security of CBC$ in random function model] *There is an adversary E for $\mathrm{CBC\$}(R_{l,l})$, in the left-or-right sense, who makes up to q queries, totaling at most μ bits, ($\mu \le l \cdot 2^{\frac{l}{2}}$) and achieves*

$$\mathrm{Adv}^{\mathrm{lr}}_E \ge 0.316 \cdot \left(1 - 2 \cdot 2^{-l/2}\right) \cdot \left(\frac{\mu^2}{l^2} - \frac{\mu}{l}\right) \cdot \frac{1}{2^l} .$$

However, these are the best possible attacks up to a constant factor:

Lemma 15 [Upper bound on security of CBC$ in random function model] *Let E be any adversary attacking $\mathrm{CBC\$}(R_{l,l})$ in the left-or-right sense, making at most q queries, totaling at most μ bits. Then*

$$\mathrm{Adv}^{\mathrm{lr}}_E \le \delta_{\mathrm{CBC\$}} \stackrel{\mathrm{def}}{=} \left([\mu^2/l^2] - [\mu/l]\right) \cdot 2^{-l} .$$

The "real-world" security follows:

Theorem 16 [Security of CBC$ using a pseudorandom function] *There is a constant $c > 0$ such that the following is true. Suppose F is a $(t', q'; \epsilon')$-secure PRF family with input length l and output length L. Then for any q the $\mathrm{CBC\$}(F)$ scheme is $(t, q, \mu; \epsilon)$-secure in the left-or-right sense, for $\mu = q'l$ and $t = t' - c\mu$ and $\epsilon = 2\epsilon' + \delta_{\mathrm{CBC\$}}$, where $\delta_{\mathrm{CBC\$}} \stackrel{\mathrm{def}}{=} \left([\mu^2/l^2] - [\mu/l]\right) \cdot 2^{-l}$.*

CBC should really be analyzed assuming F is a PRP family, not a PRF family, because the scheme must indeed be used with permutations, not functions. For the upper bound, it doesn't really make a difference, because we can apply Proposition 8 to Lemma 15 to make the translation. For the lower bound, however, this will not help. Thus at this point, it is conceivable that if F is a PRP family, CBC encryption is much *more* secure than the upper bound indicates. Yet in fact this is not true. The following says the same lower bound holds for permutations.

Proposition 17 [Lower bound on security of CBC$ in random permutation model] *There is an adversary E for CBC$($P_{l,l}$), in the left-or-right sense, who makes queries totaling at most μ bits, ($\mu \leq l \cdot 2^{\frac{l}{2}}$) and achieves*

$$\mathrm{Adv}^{\mathrm{lr}}_{E} \;\geq\; 0.316 \cdot \left([\mu^2/l^2] - [\mu/l]\right) \cdot 2^{-l} \;.$$

Note that Proposition 14 held for any q. In contrast, in Proposition 17, given μ, we allow the adversary to choose a convenient q. (Which turns out to be $q = \mu/l$.) In this sense Proposition 17 is weaker. We believe it should be possible to improve Proposition 17 but have not done the analysis at this time.

CBC WITH COUNTERS. It is tempting to make a counter variant of CBC and hope that the security is increased (or at least preserved). Indeed it is suggested in various books that the initialization vector may be a counter. But this does not work; knowing the next value of the counter, the adversary can choose a message query that forces a collision in the inputs to f, thus breaking the scheme (under any of the definitions).

To make a proper counter version of CBC$, one can let the initialization vector be $y_0 = f(ctr)$ and increment *ctr* by one following every encryption. The scheme is capable of encrypting at most 2^l messages. An analog to Theorem 16 is then possible. The result is easiest (following as a corollary to Theorem 16) if the key used to determine y_0 is separate from the key used for the rest of the CBC encryption.

Acknowledgments

We thank Ran Canetti, who gave some helpful comments on an earlier draft, and Jim Gray, who suggested the variant of Definition 1 which appears here.

References

[1] W. ALEXI, B. CHOR, O. GOLDREICH, C. SCHNORR, "RSA and Rabin functions: Certain parts are as hard as the whole," *SIAM Journal on Computing* Vol. 17, No. 2, 1988, pp. 194–209.

[2] ANSI X3.106, "American National Standard for Information Systems – Data Encryption Algorithm – Modes of Operation," American National Standards Institute, 1983.

[3] M. BELLARE, R. CANETTI AND H. KRAWCZYK "Psuedorandom functions revisited: The cascade construction and its concrete security," FOCS 96.

[4] M. BELLARE, A. DESAI, E. JOKIPII AND P. ROGAWAY, "A concrete security treatment of symmetric encryption," full version of this paper, available at `http://www-cse.ucsd.edu/users/mihir`.

[5] M. BELLARE, R. GUÉRIN AND P. ROGAWAY, "XOR MACs: New methods for message authentication using finite pseudorandom functions," Crypto 95.

[6] M. BELLARE, J. KILIAN AND P. ROGAWAY, "The security of cipher block chaining," Crypto 94.

[7] D. DOLEV, C. DWORK AND M. NAOR, "Non-malleable cryptography," STOC 91.

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

[9] O. GOLDREICH, R. IMPAGLIAZZO, L. LEVIN, R. VENKATESAN AND D. ZUCKERMAN, "Security preserving amplification of hardness," FOCS 90.

[10] O. GOLDREICH, S. GOLDWASSER AND S. MICALI, "How to construct random functions," *Journal of the ACM*, Vol. 33, No. 4, 1986, pp. 210–217.

[11] S. GOLDWASSER AND S. MICALI, "Probabilistic encryption," *J. of Computer and System Sciences*, Vol. 28, April 1984, pp. 270–299.

[12] A. HERZBERG AND M. LUBY, "Public randomness in cryptography," Crypto 92.

[13] J. HÅSTAD, R. IMPAGLIAZZO, L. LEVIN AND M. LUBY, "Construction of a pseudo-random generator from any one-way function," *ICSI Technical Report*, No. 91-068, submitted to SICOMP.

[14] ISO 8372, "Information processing – Modes of operation for a 64-bit block cipher algorithm," International Organization for Standardization, Geneva, Switzerland, 1987.

[15] M. LUBY, *Pseudorandomness and Cryptographic Applications*, Princeton University Press, 1996.

[16] M. LUBY AND C. RACKOFF, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM J. Computation*, Vol. 17, No. 2, April 1988.

[17] S. MICALI, C. RACKOFF AND R. SLOAN, "The notion of security for probabilistic cryptosystems," *SIAM J. of Computing*, April 1988.

[18] National Bureau of Standards, NBS FIPS PUB 81, "DES modes of operation," U.S Department of Commerce, 1980.

[19] M. NAOR AND M. YUNG, "Public-key cryptosystems provably secure against chosen ciphertext attacks," STOC 90.

[20] A. C. YAO, "Theory and applications of trapdoor functions," FOCS 82.