How to Protect DES Against Exhaustive Key Search (An Analysis of DESX)*

Joe Kilian† Phillip Rogaway‡

February 2, 2000

Abstract

The block cipher DESX is defined by $\mathrm{DESX}_{k.k1.k2}(x) = k2 \oplus \mathrm{DES}_k(k1 \oplus x)$, where $\oplus$ denotes bitwise exclusive-or. This construction was first suggested by Rivest as a computationally-cheap way to protect DES against exhaustive key-search attacks. This paper proves, in a formal model, that the DESX construction is sound. We show that, when F is an idealized block cipher, $FX_{k.k1.k2}(x) = k2 \oplus F_k(k1 \oplus x)$ is substantially more resistant to key search than is F. In fact, our analysis says that FX has an effective key length of at least $\kappa + n - 1 - \lg m$ bits, where κ is the key length of F, n is the block length, and m bounds the number of $\langle x,\ FX_K(x)\rangle$ pairs the adversary can obtain.

Key words: DESX, DES, Key search, Cryptanalysis, Export controls.

1 Introduction

With its 56-bit keys, the susceptibility of DES to exhaustive key search has been a concern and a complaint since the cipher was first made public; see, for example, [6]. The problem has escalated to the point that the Electronic Frontier Foundation has now built a DES cracking machine, at a cost of less than 250,000 USD, that can find the right key in about three days.

There have been many approaches suggested for reducing DES's vulnerability to exhaustive key search. One is to construct a DES-based block cipher which employs a longer key. Triple DES (typically in "EDE mode") is the best-known algorithm in this vein. It seems to be quite secure, but efficiency considerations make triple DES a rather painful way to solve the exhaustive key-search problem. Specifically, triple-DES encryption/decryption requires multiple DES encryptions/decryptions. This paper analyzes a much cheaper alternative.

Rivest [13] proposes an extension of DES, called DESX, defined by

$$\mathrm{DESX}_{k.k1.k2}(x) = k2 \oplus \mathrm{DES}_k(k1 \oplus x).$$

*An earlier version of this paper appears in [11].

† NEC Research Institute, 4 Independence Way, Princeton, NJ 08540, USA. E-mail: joe@research.nj.nec.com

‡ Department of Computer Science, University of California at Davis, Davis, CA 95616, USA. E-mail: rogaway@cs.ucdavis.edu

The key $K = k.k1.k2$ (here, . denotes concatenation) is now $56+64+64 = 184$ bits. Compatibility with DES is maintained by setting $k1 = k2 = 0^{64}$. Existing DES CBC hardware can be gainfully employed by first masking the plaintext, computing the DES CBC, and then masking the ciphertext. Most significantly, DESX has hardly any computational overhead over ordinary DES. Yet, somehow, DESX seems no longer susceptible to brute-force attacks of anything near 2^{56} time.

It is unintuitive that one should be able to substantially increase the difficulty of key search by something as simple as a couple of XORs. Yet working with the DESX definition for a while will convince the reader that undoing their effect is not so easy.

Does the "DESX trick" really work to improve the strength of DES against exhaustive key search? We give a strong positive result showing that it does.

1.1 Our model

Key-search strategies disregard the algebraic or cryptanalytic specifics of a cipher and instead treat it as a black-box transformation. Key-search strategies can be quite sophisticated; recent work by [16] is an example. We want a model generous enough to permit sophisticated key-search strategies, but restricted enough to permit *only* strategies that should be regarded as key search. We accomplish this as follows.

Let κ be the key length for a block cipher and let n be its block length. We model an *ideal* block cipher with these parameters as a *random* map $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ subject to the constraint that, for every key $k \in \{0,1\}^\kappa$, $F(k,\cdot)$ is a permutation on $\{0,1\}^n$. A key-search adversary A is an algorithm that is given the following two oracles:

- An F oracle that on input (k,x) returns $F(k,x)$ and
- An F^{-1} oracle that on input (k,y) returns $F^{-1}(k,y)$.

Here, $F^{-1}(k,y)$ denotes the unique point x such that $F(k,x) = y$.

A *generic key-search adversary* tries to perform some cryptanalytic task (to be specified) that depends on F. She may perform arbitrary computations, using unbounded amounts of time and space, but her only access to F is via the F/F^{-1} oracles. We analyze the adversary's rate of success in performing her cryptanalytic task as a function of the number of accesses she makes to the F/F^{-1} oracles.

To apply the above framework to DESX, we first generalize the DESX construction. Given any block cipher F we define $FX : \{0,1\}^{\kappa+2n} \times \{0,1\}^n \to \{0,1\}^n$ by

$$FX(k.k1.k2,\ x) = k2 \oplus F(k, k1 \oplus x).$$

For both F and FX we shall sometimes write their first argument (the key) as a subscript, $F_k(x)$ and $FX_K(x)$, where $K = k.k1.k2$. In this notation, F_k may be thought of as a permutation chosen from a family of (random) permutations that is indexed by k.

To investigate the strength of FX against key search we consider a generic key-search adversary A with oracles for F and F^{-1}, and determine how well A can play the following "FX–or–π" game.

A is given an "encryption oracle" E that has been randomly chosen in one of two ways (each with probability 0.5):

- A string $K \in \{0,1\}^{\kappa+2n}$ is chosen at random and $E(x) = FX_K(x)$, or
- A random permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ is selected and $E(x) = \pi(x)$.

A must guess which way E was chosen. The adversary "wins" the game if it guesses correctly with probability significantly greater than 0.5. The *FX* construction "works" if the resources needed to do a good job in winning the above game are substantially *greater* than the resources that suffice to break F.

As an example of a generic key-search attack, consider the weakened form of DESX, denoted DESW, in which k_1 is always set to $0^{|k_1|}$; that is,

$$\mathrm{DESW}_{k.k2}(x) = k2 \oplus \mathrm{DES}_k(x).$$

It is possible to mount a generic key-search attack DESW as follows. Given k and $\mathrm{DESW}_{k.k_2}(x)$ for an arbitrary x, one can compute $k_2 = \mathrm{DESW}_{k.k_2}(x) \oplus \mathrm{DES}_k(x)$. Thus, one can go through all possible keys k, compute the full key $k.k_2$, and test with high confidence whether $k.k_2$ is correct (given values of $\mathrm{DES}_{k.k_2}(y)$ for a couple of random y-values). Hence, DESW is no stronger than DES against generic key-search attacks. Similarly, if k_2 is always set to $0^{|k_2|}$, there is no significant improvement over DES, as long as two or three plaintext-ciphertext pairs are known. (There may be marginal benefits if only a single plaintext-ciphertext pair is known, or for ciphertext-only attacks, but these are comparatively small improvements.) It is the combination of the two XOR operations that give DESX its superior resistance to generic key-search attacks.

1.2 Our main result

We show that if generic key-search adversary A can make only a "reasonable" number to queries to her encryption oracle E, then A must ask an excessive number of F/F^{-1} queries in the *FX*-or-π game, and therefore A must run for an excessively long time. More specifically, we prove the following. Let m bound the number of $\langle x, FX_K(x)\rangle$ pairs that the adversary can obtain. (This number is usually under the control of the security architect, not the adversary.) Suppose the adversary makes at most t queries to her F/F^{-1} oracles. (This number is usually under the control of the adversary, not the security architect.) Then the adversary's advantage over random guessing (i.e., the difference between its success and failure probabilities) in winning the *FX*–or–π game is at most $mt \cdot 2^{-\kappa-n+1}$. In other words, the adversary's advantage is at most $t \cdot 2^{-\kappa-n+1+\lg m}$, so the effective key length of *FX*, with respect to key search, is at least $\kappa + n - 1 - \lg m$ bits.

To understand the above formula, consider a block cipher F with 55-bit keys and a 64-bit block size.[1] Suppose key-search adversary A attacks *FX* and in the course of attack able to obtain up to $m = 2^{30}$ blocks of enciphered data. Suppose A runs in time at most T. Then A has advantage of at most $T \cdot 2^{-55-64+30+1} = T \cdot 2^{-88}$ to just guess whether the enciphered data really *was* produced by *FX*, and not a random permutation. A more detailed discussion of out main theorem is given in Section 4.

[1] Why we use 55 and not 56 is explained in the discussion in Section 4.

Because our main result indicates the infeasibility of key search even when we ignore the adversary's space requirement, this "omission" only strengthens what we are saying. Similarly, "good" adversaries may, necessarily, use an amount of time, T, which far exceeds their number of F/F^{-1} queries, t. So focusing on the query complexity makes our results all the more meaningful. Likewise, the weakness of the adversary's goal only strengthens the lower bound.

1.3 Related work

Even and Mansour [8] construct a block cipher $PX : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ from a random permutation $P : \{0,1\}^n \to \{0,1\}^n$ by $PX_{k1.k2}(x) = k2 \oplus P(k1 \oplus x)$. Clearly this is a special case of the FX construction, where $\kappa = 0$. While their motivation for looking at PX was quite different from our reasons to investigate FX, our model and methods are, in fact, quite similar. Our main result can be seen as a natural extension of their work.

The modeling of a block cipher by a family of random permutations has its roots in [15].

Ron Rivest invented DESX by May of 1984, but never described the scheme in any conference or journal paper [13]. DESX was implemented within products of RSA Data Security, Inc., and is described in the documentation for these products [14]. DESX has also been described at conferences organized by RSA DSI, including [18].

Encryption methods similar to DESX have been invented independently. Blaze [3] describes a DES mode of operation in which the ith block of plaintext, x_i, is encrypted using 112-bit key $k.k1$ by $E_{k.k1}(x_i) = s_i \oplus \mathrm{DES}_k(s_i \oplus x)$, where $s_1 s_2 \cdots$ is a stream of bits generated from $k1$ by, say, $s_i = \mathrm{DES}_{k1}^{(i)}(0^{64})$. Here $\mathrm{DES}^{(i)}$ denotes the i-th iterate of DES.

Many authors have suggested methods to increase the strength of DES by changing its internal structure. Biham and Biryukov [1] give ways to modify DES to use key-dependent S-boxes. Their suggestions improve the cipher's strength against differential, linear, and improved Davies' attacks, as well as exhaustive key search. Ciphers constructed using their ideas can exploit existing hardware exactly in those cases where the hardware allows the user to substitute his own S-boxes in place of the standard ones.

1.4 Discussion

Understanding our result. It may be hard to understand the ramifications of our main theorem, thinking it means more or less than it does. DES, of course, is not a family of random permutations, and we can *not* conclude from our theorem that there does not exist a reasonable machine M which breaks DESX in say, 2^{60} steps, given just a handful of $\langle$plaintext, ciphertext$\rangle$ pairs. What we can say is that such a machine would have to exploit structural properties of DES; it couldn't get away with treating DES as a black-box transformation. This contrasts with the sort of machines which have been suggested in the past for doing brute-force attack: they *do* treat the underlying cipher as a black-box transformation.

We note that while remarkable theoretical progress has been made on the linear and differential cryptanalysis of DES (see [2, 12]), thus far these attacks require an impractically large number of plaintext-ciphertext pairs. To date, the only published practical attacks against DES remain of the key-search variety. The DESX construction was not intended to improve the strength of DES against differential or linear attack, or any other attack which exploits structural properties of DES, and our theorem does not say anything about its resistance to these attacks.

On export controls tied to key length. Our results indicate how algorithmically trivial it can be to obtain extra bits of strength against exhaustive key-search attacks. The impact of these extra bits can be especially dramatic when the key length of the block cipher had been intentionally made short.

Consider a block cipher F with a 40-bit key and a 64-bit plaintext. (Some products using such block ciphers have been granted U.S. export approval.) With these parameters, our results guarantee an effective key length (with respect to exhaustive key search) of at least $40+64-1-\lg m = 103-\lg m$ bits. Under the reasonable assumption that $m < 2^{30}$, say, the 40-bit block cipher has been modified, with two XORs, to a new block cipher which needs at least 2^{73}-time for key exhaustive key search.

Allowing weak cryptography to be exported and strong cryptography not to be is a policy which can only make sense when it is impractical, for the given system, to replace the weak mechanism by a strong one. Our results indicate that this impracticality must cover algorithmic changes that are particularly trivial.

1.5 Outline of the paper

In Section 2 we define some basic notation and define what comprises a successful attack in our model. In Section 3 we state and prove our main theorem on the security of the DESX construction. Section 4 is a discussion. Section 5 demonstrates that the analysis underlying our main result is tight. In Section 6 we give some conclusions and open questions.

2 Preliminaries

Let $\mathcal{P}_n$ denote the space of all $(2^n)!$ permutations on n-bits.

We say that $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is a block cipher if for every $k \in \{0,1\}^\kappa$, $F(k,\cdot) \in \mathcal{P}_n$. We define F_k by $F_k(x) = F(k,x)$. Let $\mathcal{B}_{\kappa,n}$ denote the space of all block ciphers with parameters κ and n as above.

Given $F \in \mathcal{B}_{\kappa,n}$, we define the block cipher $F^{-1} \in \mathcal{B}_{\kappa,n}$ by $F^{-1}(k,y) = F_k^{-1}(y)$ for $k \in \{0,1\}^\kappa$. We interchangeably write $F_k^{-1}(y)$ and $F^{-1}(k,y)$.

Given $F \in \mathcal{B}_{\kappa,n}$, we define the block cipher $FX \in \mathcal{B}_{\kappa+2n,n}$ by $FX(K,x) = k2 \oplus F_k(k1 \oplus x)$, where $K = k.k1.k2$, $|k| = \kappa$ and $|k1| = |k2| = n$. We interchangeably write $FX_K(x)$ and $FX(K,x)$.

Given a partially defined function F from a subset of $\{0,1\}^m$ to a subset of $\{0,1\}^n$ we denote the domain and range of F by $\mathrm{Dom}(F)$ and $\mathrm{Range}(F)$, and define $\overline{\mathrm{Dom}}(F) = \{0,1\}^m - \mathrm{Dom}(F)$ and $\overline{\mathrm{Range}}(F) = \{0,1\}^n - \mathrm{Range}(F)$.

We denote by $x \stackrel{R}{\leftarrow} S$ the act of choosing x uniformly from S. We denote by $\Pr[A_1;\ A_2;\ \ldots : E]$ the probability of event E after performing actions $A_1, A_2, \ldots$.

Definition 2.1 *A generic key-search adversary is an algorithm A with access to three oracles, E, F and F^{-1}. Thus, A may make queries of the form $E(P)$, $F_k(x)$ or $F_k^{-1}(y)$. An (m,t) generic key-search adversary is a key-search adversary that makes m queries to the E oracle and a total of t queries to the F and F^{-1} oracles.*

For brevity, we will sometimes drop "generic" from our terminology. Note that A supplies the value of k as part of its queries to the F and F^{-1} oracles. We denote by $A^{E,\ F,\ F^{-1}}$ the adversary A interacting with oracles E, F and F^{-1}.

We now define what it means for a generic key-search adversary A to have an attack of a certain specified effectiveness. We begin by choosing a random block cipher F having κ-bit keys and n-bit blocks. This means that we select a random permutation $F_k \stackrel{R}{\leftarrow} \mathcal{P}_n$ for each κ-bit key k. Thus each F_k is chosen independently of each $F_{k'}$, for $k \neq k'$. Then we give A three oracles, E, F and F^{-1}. The F and F^{-1} oracles compute their respective functions. The encryptions oracle E, on input x, either computes $FX_K(x)$ for a random $(\kappa+2n)$-bit key K or computes $\pi(x)$, for a random permutation $\pi \stackrel{R}{\leftarrow} \mathcal{P}_n$. The adversary's job is to guess which type of encryption oracle she has. Our convention is that A outputs a 1 to guess that the encryption oracle is computing $FX_K(x)$. The adversary's *advantage* is her probability of guessing right, normalized to a $[-1,1]$ scale: -1 indicates a strategy that always guesses wrong; 1 indicates a strategy that always guesses correctly; guessing at random, or always guessing the same way, will give an advantage of 0.

Definition 2.2 *Let $\kappa, n \geq 0$ be integers, and let $\epsilon \geq 0$ be a real number. Generic key-search adversary A is said to ϵ-*break *the FX-scheme with parameters κ, n if*

$$
\begin{aligned}
\mathrm{Adv}_A \quad \stackrel{\text{def}}{=} \quad & \Pr\left[F \stackrel{R}{\leftarrow} \mathcal{B}_{\kappa,n};\ K \stackrel{R}{\leftarrow} \{0,1\}^{\kappa+2n} :\ A^{FX_K,\ F,\ F^{-1}} = 1\right] - \\
& \Pr\left[F \stackrel{R}{\leftarrow} \mathcal{B}_{\kappa,n};\ \pi \stackrel{R}{\leftarrow} \mathcal{P}_n :\ A^{\pi,\ F,\ F^{-1}} = 1\right] \\
\geq \quad & \epsilon\ .
\end{aligned}
$$

The above definition uses a very liberal notion of adversarial success. We are not demanding that, say, A recover K; nor do we ask A to decrypt a random $FX_K(x)$ or to produce a not-yet-asked $\langle x, FX_K(x)\rangle$ pair. Instead, we only ask A to make a good guess as to whether the $\langle$plaintext, ciphertext$\rangle$ pairs she has been receiving really *are* FX-encryptions, as opposed to random nonsense unrelated to F. The liberal notion of success is chosen to make our main result stronger: an adversary's inability to succeed becomes all the more meaningful.

3 Security of the DESX Construction

We now prove a bound on the security of FX against generic key-search attacks.

Theorem 3.1 *Let A be an (m,t) generic key-search adversary that ϵ-breaks the FX-scheme with parameters κ, n. Then $\epsilon \leq mt \cdot 2^{-\kappa-n+1}$.*

Proof: Before going into the detailed formal proof, we first give some intuition for why the proof works. Clearly, the *FX* construction is highly nonrandom if one makes all possible queries to the E, F and F^{-1} oracles. However, to defeat the adversary it suffices if the answers to its relatively few queries are random. For intuition, we erroneously think of F as a family of random functions (the formal analysis takes into account the fact that F_k is a permutation). We conceptually view F as undefined; as queries from the adversary come in we choose values of $F_k(x)$ at random. Note that queries to $E(x)$ implicitly make queries to F. If in computing $E(x)$ we make a "fresh" query to F (one that hasn't been made before), we generate a fresh answer that is random and independent of the entire history of the attack. This fresh randomness ensures that the resulting value of $E(x)$ will be random. However, randomness cannot be guaranteed when new queries depend on previously determined values. We show that if the adversary doesn't make many queries, then these bad events happen with low probability.

By a standard argument we may assume that A is deterministic (note that A may be computationally unbounded).[2] We may also assume that A always asks exactly m queries of her first oracle, which we shall call her E-oracle. (In the experiment that defines A's advantage, E was instantiated by either an FX_K-oracle or a π-oracle.) We may assume that A always asks exactly t queries (total) to her second and third oracles, which we shall call her F- and F^{-1}- oracles. We may further assume that A never repeats a query to an oracle. We may assume that if $F(k,x)$ returns an answer y, then there is no query (neither earlier nor later) of $F^{-1}(k,y)$. All of the above assumptions are without loss of generality in the sense that it is easy to construct a new adversary, A', that obeys the above constraints and has the same advantage as A.

We begin by considering two different games that adversary A might play. This amounts to specifying how to simulate a triple of oracles, $\langle E, F, F^{-1}\rangle$, for the benefit of A.

A FIRST GAME. The first game we consider, Game R (for "random"), will exactly correspond to the experiment which defines the second addend in the expression for the advantage:

$$P_R = \Pr\left[A^{\pi,\ F,\ F^{-1}} = 1\right].$$

The definition of Game R will be defined to contain several extra (and seemingly irrelevant) steps. These steps aren't needed in order to behave in a manner which is identical (as far as A sees) to the manner of behavior defining P_R; these steps are used, instead, to facilitate our analysis. To identify these "irrelevant" instructions we put them in italics. Game R is defined in Figure 1.

Let $\Pr_R[\cdot]$ denote the probability of the specified event with respect to Game R. From the definition of Game R we can see that:

Claim 3.1 $\Pr_R\left[A^{E,F,F^{-1}} = 1\right] = P_R$.

A SECOND GAME. Now we define a second game, Game X. It will exactly correspond to the experiment which defines the first term in the expression for the advantage:

$$P_X = \Pr\left[A^{FX_K,\ F,\ F^{-1}} = 1\right].$$

[2] Roughly, given unlimited computational capabilities, A can derandomize its strategy by exhaustively searching through its possible random choices, computing the effectiveness of the resulting attack, and then choosing the most efficacious choice.

Game R

Initially, let F and E be undefined. *Flag* ***bad*** *is initially unset. Randomly choose* $k^* \xleftarrow{R} \{0,1\}^\kappa$, $k_1^*, k_2^* \xleftarrow{R} \{0,1\}^n$. Then answer each query the adversary makes as follows:

$\boxed{E}$ On oracle query $E(P)$:

1. Choose $C \in \{0,1\}^n$ uniformly from $\overline{\text{Range}}(E)$.
2. *If* $F_{k^*}(P \oplus k_1^*)$ *is defined, then set* ***bad****.*
   If $F_{k^*}^{-1}(C \oplus k_2^*)$ *is defined, then set* ***bad****.*
3. Define $E(P) = C$ and return C.

$\boxed{F}$ On oracle query $F_k(x)$:

1. Choose $y \in \{0,1\}^n$ uniformly from $\overline{\text{Range}}(F_k)$.
2. *If* $k = k^*$ *and* $E(x \oplus k_1^*)$ *is defined then set* ***bad****.*
   If $k = k^*$ *and* $E^{-1}(y \oplus k_2^*)$ *is defined then set* ***bad****.*
3. Define $F_k(x) = y$ and return y.

$\boxed{F^{-1}}$ On oracle query $F_k^{-1}(y)$:

1. Choose $x \in \{0,1\}^n$ uniformly from $\overline{\text{Dom}}(F_k)$.
2. *If* $k = k^*$ *and* $E^{-1}(y \oplus k_2^*)$ *is defined then set* ***bad****.*
   If $k = k^*$ *and* $E(x \oplus k_1^*)$ *is defined then set* ***bad****.*
3. Define $F_k(x) = y$ and return x.

Game X

Initially, let F and E be undefined. *Flag* ***bad*** *is initially unset.* Randomly choose $k^* \xleftarrow{R} \{0,1\}^\kappa$, $k_1^*, k_2^* \xleftarrow{R} \{0,1\}^n$. Then answer each query the adversary makes as follows:

$\boxed{E}$ On oracle query $E(P)$:

1. Choose $C \in \{0,1\}^n$ uniformly from $\overline{\text{Range}}(E)$.
2. If $F_{k^*}(P \oplus k_1^*)$ is defined, then $C \leftarrow F_{k^*}(P \oplus k_1^*) \oplus k_2^*$ *and set* ***bad****.*
   Else if $F_{k^*}^{-1}(C \oplus k_2^*)$ is defined, then *set* ***bad*** *and* goto Step 1.
3. Define $E(P) = C$ and return C.

$\boxed{F}$ On oracle query $F_k(x)$:

1. Choose $y \in \{0,1\}^n$ uniformly from $\overline{\text{Range}}(F_k)$.
2. If $k = k^*$ and $E(x \oplus k_1^*)$ is defined then $y \leftarrow E(x \oplus k_1^*) \oplus k_2^*$ *and set* ***bad****.*
   Else If $k = k^*$ and $E^{-1}(y \oplus k_2^*)$ is defined then *set* ***bad*** *and* goto Step 1.
3. Define $F_k(x) = y$ and return y.

$\boxed{F^{-1}}$ On oracle query $F_k^{-1}(y)$:

1. Choose $x \in \{0,1\}^n$ uniformly from $\overline{\text{Dom}}(F_k)$.
2. If $k = k^*$ and $E^{-1}(y \oplus k_2^*)$ is defined then $x \leftarrow E^{-1}(y \oplus k_2^*) \oplus k_1^*$ *and set* ***bad****.*
   Else if $k = k^*$ and $E(x \oplus k_1^*)$ is defined then *set* ***bad*** *and* goto Step 1.
3. Define $F_k(x) = y$ and return x.

Figure 1: Games R and X.

Once again, the definition of Game X will be defined to contain some "irrelevant" instructions, which, for clarity, are indicated in italics. Game X is defined in Figure 1.

The intuition behind Game X is as follows. We *try* to behave like Game R, choosing a random (not-yet-provided) answer for each $E(P)$, and a random (not-yet-provided for this k) answer for each $F_k(x)$, $F_k^{-1}(y)$. Usually this works fine for getting behavior which looks like the experiment defining P_X. But sometimes it doesn't work, because an "inconsistency" would be created between the FX-answers and the F/F^{-1}-answers. Game X is vigilant in checking if any such inconsistencies are being created. If it finds an inconsistency about to be created, it *changes* the value which it had "wanted" to answer in order to *force* consistency. Whenever Game X resorts to doing this it sets the flag ***bad***. In the analysis, we "give up" (regard the adversary as having won) any time this happens.

Let $\Pr_X[\cdot]$ denote the probability of the specified event with respect to Game X. The definition of Game X looks somewhat further afield from the experiment which defines P_X. Nonetheless, we claim the following:

Claim 3.2 $\Pr_X\left[A^{E,F,F^{-1}}=1\right]=P_X$.

The proof of this claim is in the appendix.

Bounding the advantage by $\Pr_R[\,\mathsf{BAD}\,]$. In either Game R or Game X, let BAD be the event that, at some point in time, the flag ***bad*** gets set. Games R and X have been defined so as to coincide up until event BAD. To see this, note that the corresponding oracles in these games are identical except for, in each case, Step 2. For each pair of oracles, Step 2 executes identical tests and based on the outcome of the test either does nothing in both cases or sets ***bad*** in both cases (and other actions, in which the oracles will differ in their behavior). Thus, any circumstance that causes Game R and Game X to execute different instructions will also cause both games to set ***bad***. The following two claims follow directly from this fact.

Claim 3.3 $\Pr_R[\,\mathsf{BAD}\,]=\Pr_X[\,\mathsf{BAD}\,]$.
Claim 3.4 $\Pr_R\left[A^{E,F,F^{-1}}=1|\overline{\mathsf{BAD}}\right]=\Pr_X\left[A^{E,F,F^{-1}}=1|\overline{\mathsf{BAD}}\right]$.

What we have shown so far allows us to bound the adversary's advantage by $\Pr_R[\,\mathsf{BAD}\,]$.

Claim 3.5 $\mathrm{Adv}_A\le\Pr_R[\,\mathsf{BAD}\,]$.

The argument is quite simple:

$$\begin{aligned}
\mathrm{Adv}_A &= P_X-P_R\\
&= \Pr_X\left[A^{E,F,F^{-1}}=1\right]-\Pr_R\left[A^{E,F,F^{-1}}=1\right] && \textit{(Claims 3.1, 3.2)}\\
&= \Pr_X\left[A=1|\overline{\mathsf{BAD}}\,\right]\Pr_X\left[\,\overline{\mathsf{BAD}}\,\right]+\Pr_X\left[A=1|\mathsf{BAD}\,\right]\Pr_X\left[\,\mathsf{BAD}\,\right]-\\
&\quad\ \Pr_R\left[A=1|\overline{\mathsf{BAD}}\,\right]\Pr_R\left[\,\overline{\mathsf{BAD}}\,\right]-\Pr_R\left[A=1|\mathsf{BAD}\,\right]\Pr_R\left[\,\mathsf{BAD}\,\right]\\
&= \Pr_R[\,\mathsf{BAD}\,](\Pr_X[A=1|\mathsf{BAD}\,]-\Pr_R[A=1|\mathsf{BAD}\,]) && \textit{(Claims 3.3, 3.4)}\\
&\le \Pr_R[\,\mathsf{BAD}\,]
\end{aligned}$$

Initially, let F and E be undefined. Answer each query the adversary makes as follows:

$\boxed{E}$ On oracle query $E(P)$:

1. Choose C uniformly from $\overline{\mathrm{Range}}(E)$.
2. Define $E(P) = C$ and return C.

$\boxed{F}$ On oracle query $F_k(x)$:

1. Choose y uniformly from $\overline{\mathrm{Range}}(F_k)$
2. Define $F_k(x) = y$ and return y.

$\boxed{F^{-1}}$ On oracle query $F_k^{-1}(y)$:

1. Choose x uniformly from $\overline{\mathrm{Dom}}(F_k)$.
2. Define $F_k(x) = y$ and return x.

After all the queries have been answered:
Flag ***bad*** *is initially unset.*
Randomly choose $k^* \stackrel{R}{\leftarrow} \{0,1\}^\kappa$, $k_1^*, k_2^* \stackrel{R}{\leftarrow} \{0,1\}^n$.
If $\exists\, x$ *such that* $F_{k^*}(x)$ *and* $E(x \oplus k_1^*)$ *are both defined then set* ***bad****.*
If $\exists\, y$ *such that* $F_{k^*}^{-1}(y)$ *and* $E^{-1}(y \oplus k_2^*)$ *are both defined then set* ***bad****.*

Figure 2: Game R'

A THIRD GAME. We have reduced our analysis to bounding $\Pr_R[\,\mathsf{BAD}\,]$. To bound $\Pr_R[\,\mathsf{BAD}\,]$, let us imagine playing Game R a little bit differently. Instead of choosing k^*, k_1^*, k_2^* at the beginning, we choose them at the end. Then we set ***bad*** to be *true* or *false* depending on whether or not the choice of k^*, k_1^*, k_2^* we've just made would have caused ***bad*** to be set to true in Game R (where the choice was made at the beginning). The new game, Game R', is described in Figure 2. From the definition of Game R' we see that:

Claim 3.6 $\Pr_R[\,\mathsf{BAD}\,] = \Pr_{R'}[\,\mathsf{BAD}\,]$.

COMPLETING THE PROOF. Now that we have sufficiently manipulated the games a simple calculation suffices to bound $\Pr_{R'}[\,\mathsf{BAD}\,]$, and, thereby, to bound Adv_A.

After having run the body of Game R', not having yet chosen k^*, k_1^*, k_2^*, let us simply count how many of the $2^{\kappa+2n}$ choices for (k^*, k_1^*, k_2^*) will result in ***bad*** getting set.

Fix any possible values for E and F which can arise in Game R'. Let $|E|$ denote the number of defined values $E(P)$, and let $|F|$ denote the number of defined values $F_k(x)$. Note that $|E| = m$ and $|F| = t$. Fix E and F. Call (k^*, k_1^*, k_2^*) *collision-inducing* (with respect to E and F) if there is some defined $y = F_k(x)$ and some defined $C = E(P)$ such that

$$k^* = k \ \text{ and } \ (P \oplus k_1^* = x \ \text{ or } \ C \oplus k_2^* = y).$$

Every choice of (k^*, k_1^*, k_2^*) which results in setting ***bad*** is collision-inducing, so it suffices to upper bound the number of collision-inducing (k^*, k_1^*, k_2^*).

Claim 3.7 *Fix E, F, where $|E| = m$ and $|F| = t$. There are at most $2mt \cdot 2^n$ collision-inducing $(k^*, k_1^*, k_2^*) \in \{0,1\}^\kappa \times \{0,1\}^n \times \{0,1\}^n$.*

The reason is as follows: for each defined $(P, E(P))$, $(k, x, F_k(x))$ there are at most $2 \cdot 2^n$ points (k^*, k_1^*, k_2^*) which induce a collision between these two points: they are the points $(k^*, k_1^*, k_2^*) \in \{k\} \times \{x \oplus P\} \times \{0,1\}^n\} \cup \{k\} \times \{0,1\}^n \times \{y \oplus C\}\}$. Now there are only mt pairs of such points, so the total number of collision-inducing (k^*, k_1^*, k_2^*) is as claimed.

Finally, in Game R' we choose a triple (k^*, k_1^*, k_2^*) at random, independent of E and F, so the chance that the selected triple is collision-inducing (for whatever E and F have been selected) is at most $2mt \cdot 2^n / 2^{\kappa+2n} = mt \cdot 2^{-\kappa-n+1}$. Pulling everything together, this probability bounds Adv_A, and we are done. $\diamondsuit$

4 Discussion

Health warnings. We emphasize that when F is a concrete block cipher, not a random one, its internal structure can interact with the FX-construction in such a way as to obviate the construction's benefits. As a trivial example, if F *already* has the structure that it XORs plaintext and ciphertext with key material, then doing it *again* is certainly of no utility.

Our model considers how much FX_K looks like a random permutation (when key K is random and unknown). It should be emphasized that some constructions which use block ciphers—particularly hash function constructions—assume something more of the underlying block cipher. The current results imply nothing about the suitability of FX in constructions which are *not* based on FX_K resembling a random permutation when K is random and unknown.

We also note that our analysis as stated only considers chosen-plaintext attacks and does not establish resistance to chosen-ciphertext attacks. However, it is straightforward to adapt our techniques to analyze chosen-ciphertext attacks, as was done in [8]. To do this, provide A an oracle for FX^{-1}, in addition to her other oracles. Now m will count the sum of the number of queries to the FX and FX^{-1} oracles. Theorem 3.1 will then continue to hold. The proof changes very little.

Structure in the block cipher F when F = DES. There is one structural property of DES which has been suggested to assist in brute-force attacks: the DES key-complementation property. This property comprises a significant sense in which DES is not behaving like a family of (independent) random permutations. To "factor out" the key-complementation property just think of DES as having a single key bit fixed. Then one can conclude that if this is the only structural property of DES to be exploited by a generic key-search attack, DESX will still limit the attack's advantage to $tm \cdot 2^{-55-64+1} = tm \cdot 2^{-118}$.

Setting $k1 = k2$. As mentioned in the introduction, the simpler constructions $FX^{\mathrm{pre}}_{k.k1}(x) = F_k(x \oplus k1)$ and $FX^{\mathrm{post}}_{k.k1}(x) = k1 \oplus F_k(x)$ don't significantly improve F's strength against generic key search attacks. But what about

$$FX'_{k.k1}(x) = k1 \oplus F_k(x \oplus k1)?$$

Is it OK to use the same key inside and out? In fact this does work, in the sense that Theorem 3.1 still goes through, the proof little changed. We analyzed the more "standard" general construction, with two keys, but the more restricted choice has the advantage of a smaller key-size, with no obvious loss of security.

NICER KEY LENGTHS. A minor inconvenience of DESX is its strange key size. In applications it would sometimes be preferable to extend the definition of DESX to use arbitrary-length keys, or else to use keys of some fixed but more convenient length. Standard key-separation techniques can be used.

We give one extension of DESX to arbitrary-length keys, as follows. Let $X_{1\ldots\ell}$ denotes the first ℓ bits of X, let SHA-1 be the map of the NIST Secure Hash Standard, and let C, $C1$ and $C2$ be fixed, distinct, equal-length strings. When $|K| \neq 184$, we can define $\mathrm{DESX}_K(x)$ to be equal to $\mathrm{DESX}_{K'}(x)$ where K' is defined as follows:

- If $|K| = 56$ then $K' = K.0^{64}, 0^{64}$.
- Otherwise, $K' = k.k1.k2$, where $\begin{cases} k &= \text{SHA-1}(C.K)_{1\ldots 56}, \\ k1 &= \text{SHA-1}(C1.K)_{1\ldots 64}, \text{ and} \\ k2 &= \text{SHA-1}(C2.K)_{1\ldots 64} \end{cases}$

Note that when $|K| = 56$, $\mathrm{DESX}_K(x) = \mathrm{DES}_K(x)$.

DIFFERENTIAL AND LINEAR CRYPTANALYSIS. OPERATIONS BESIDES XOR. We emphasize that the DESX construction was never intended to add strength against differential or linear cryptanalysis. The attacks of [2, 12] do not represent a threat against DES when the cipher is prudently employed (e.g., when a re-key is forced before an inordinate amount of text has been acted on); until these attacks are improved, it suffices that the DESX construction does not render differential or linear attack any *easier.*

Nonetheless, the proof of Theorem 3.1 goes through when $\oplus$ is replaced by a variety of other operations, and some of these alternatives may help to defeat attacks which were not addressed by our model, including differential and linear cryptanalysis. In particular, an attractive alternative to DESX may be the construction $\mathrm{DESP}_{k.k1.k2}(x) = k2 + \mathrm{DES}_k(k1 + x)$, where $LR + L'R' \stackrel{\mathrm{def}}{=} L \hat{+} L' \,.\, R \hat{+} R'$, where $|L| = |R| = |L'| = |R'| = 32$ and $\hat{+}$ denotes addition modulo 2^{32}. Burt Kaliski has suggested such alternatives, and analyzed their security with respect to differential and linear attacks [9].

5 Our Bound is Tight

We have shown that the adversary's advantage is at most $t \cdot 2^{-n-\kappa+1+\lg m}$. Turning this around, the adversary needs $\epsilon 2^{n+\kappa-1-\lg m}$ queries to the F/F^{-1} oracles to achieve an ϵ-advantage. We now show that for a wide range of m (comprising all m that would be considered in practice), an attacker can achieve an ϵ-advantage using very close to $2^{n+\kappa+4-\lg m}$ queries to the F/F^{-1} oracles (the exact bound is given in Corollary 5.2). This follows as a corollary of a more ambitious attack. This attack recovers a key $K' = k'.k1'.k_2'$ that is consistent with the encryptions under FX of m plaintexts chosen before F/F^{-1} oracles queries are made.

Theorem 5.1 *Let m be even, $m < 2^n$ and $\epsilon < \frac{1}{2}$. Let block cipher F be uniformly distributed over $\mathcal{B}_{\kappa,n}$ and let key K be uniformly distributed over $\{0,1\}^{\kappa+2n}$. Then there exists an adversary $A(m,\epsilon)$ that initially makes m distinct queries $t_1,\ldots,t_m$ (the test set) to an oracle computing FX_K. Adversary A then makes*

$$\left(2^{n+\kappa+1-\lg m} + 2^n + 2^\kappa\right)\left(\epsilon+\epsilon^2\right)$$

expected queries to the F/F^{-1} oracles. With probability at least ϵ it returns a K' such that $FX'_K(t_i) = FX_K(t_i)$ for $1 \le i \le m$. The probability is taken over the choice of F, K and A's coin tosses.

It follows that our analysis is essentially tight, given our measure on the attacker's resources, which roughly corresponds to time. We note that in practice it is also important to consider the memory requirements of an attack. Conceivably, there exists stronger attacks than require the same amount of time but much less memory. However, if the time requirements are sufficiently high, the memory issue becomes moot. However, it is an interesting open question whether imposing a reasonable space bound can allow us to improve our time bound.

For reasonable values of m, the task performed by $A(m,\epsilon)$ is at least as strong as simply distinguishing FX from a purely random permutation. To see this, consider any family of permutations $\{FX_K\}$ on $\{0,1\}^n$, where $|K| = \kappa + 2n$. We say that π is *plausible* if for some K, $\pi(x_i) = FX_K(x_i)$ for $1 \le i \le m$. If π is chosen at random, then by a simple counting argument the probability that it is plausible is at most

$$\rho(\kappa,n,m) \stackrel{\text{def}}{=} \frac{2^{\kappa+2n}}{2^n(2^n-1)\cdots(2^n-m+1)} .$$

For example, if $\kappa \le n$, $n > 20$ and $m > 6$ then $\rho < 10^{-24}$. So an attacker who outputs a 1 iff she finds a consistent K has an advantage of $\epsilon - \rho(\kappa,n,m)$, which is essentially ϵ.

A minor technical point is that our lower bound considered attackers with worst-case instead of expected case bounds. However, we can convert the expectation into a worst case bound by observing that if an expected value is at most Q then with probability $\frac{1}{2}$ it is at most $2Q$. Hence, for $\epsilon < \frac{1}{2}$ we can set the attacker A in Theorem 5.1 to find a consistent K with probability 2ϵ, and time out if A takes more than twice its expected number of F/F^{-1} queries. The resulting attack uses at most

$$\begin{aligned} t &\le \left(2^{n+\kappa+2-\lg m} + 2^{n+1} + 2^{\kappa+1}\right)\left(2\epsilon+4\epsilon^2\right) \\ &\le \left(2^{n+\kappa+4-\lg m} + 2^{n+3} + 2^{\kappa+3}\right)\epsilon \end{aligned}$$

worst-case queries to the F/F^{-1} oracles.

Finally, since the advantage is $\epsilon - \rho(\kappa,n,m)$ we can set ϵ to be $\rho(\kappa,n,m)$ bigger than the desired advantage, giving the following corollary.

Corollary 5.2 *Let m be even, $m < 2^n$ and $\epsilon < \frac{1}{2} - \rho(\kappa, n, m)$. Let block cipher F be uniformly distributed over $\mathcal{B}_{\kappa,n}$, key K be uniformly distributed over $\{0,1\}^{\kappa+2n}$ and permutation π be a uniformly distributed over $\mathcal{P}_n$. There exists an attacker $A(m, \epsilon)$ that makes m queries to oracle E (computing either FX_K or π) and makes*

$$\left(2^{n+\kappa+4-\lg m} + 2^{n+3} + 2^{\kappa+3}\right)(\epsilon + \rho(\kappa, n, m))$$

queries to the F/F^{-1} oracles. A solves the FX-or-π game with advantage at least ϵ.

The rest of this section is devoted to the proof of Theorem 5.1.

To motivate our attack, we can view the *FX* block cipher as choosing a random key k and then applying the Even-Mansour construction to the function F_k. We can therefore trivially adapt Daemen's chosen-plaintext attack [5] on the Even-Mansour construction [8]. Unfortunately, we don't know the value of k, so we instead try all possible ones. For completeness, we describe the attack and calculate the amount of work required to have probability ϵ of recovering the key.

5.1 Preliminaries

Assume that m is even, $m \leq 2^n$, and $\epsilon < \frac{1}{2}$. Fix a constant $C \in \{0,1\}^n - \{0^n\}$. For any function G, define $G^\Delta(x) = G(x \oplus C) \oplus G(x)$. Given an oracle for G one can compute G^Δ by making two calls. Let the secret key $K = k.k1.k2$. Let E by a synonym for FX. By our definitions and simple algebra we have

$$E_K^\Delta(x) = F_k^\Delta(x \oplus k1) = F_k^\Delta(x \oplus C \oplus k1).$$

5.2 The key-search attack

The attacker A works as follows. A uses oracles computing FX_K (for the correct $K = k.k1.k2$) and F. Attacker A takes as parameters m, the maximum number of queries it is allowed to make to the FX_K oracle and ϵ a required lower bound on its probability of producing a key K' that gives consistent results on the m queries it made to FX_K.

$A(m, \epsilon)$

1. Choose $x_1, \ldots, x_{m/2} \in \{0,1\}^n$ arbitrarily so that

$$\text{TEST} = x_1, \ldots, x_{m/2}, x_1 \oplus C, \ldots, x_{m/2} \oplus C$$

   has m distinct elements.

2. Using the FX_K oracle, compute $FX_K(t)$ for $t \in \text{TEST}$, and then compute

$$FX_K^\Delta(x_1), \ldots, FX_K^\Delta(x_{m/2}).$$

3. For i from 1 to $\ell = \left\lceil \frac{-2^n \ln(1-\epsilon)}{m} \right\rceil$ do
   Choose $r \xleftarrow{R} \{0,1\}^n$

```
For all k' ∈ {0,1}^κ, 1 ≤ j ≤ m/2 do
    If F^Δ_{k'}(r) = FX^Δ_K(x_j)
      /* Hope that k' = k and r is either x_j ⊕ k1 or x_j ⊕ C ⊕ k1 */
      For k1' ∈ {x_j ⊕ r, x_j ⊕ C ⊕ r}
         k2' = F_{k'}(x_1 ⊕ k1') ⊕ FX_K(x_1); K'=k'.k1'.k2'
         If FX_{K'}(t) = FX_K(t) for t ∈ TEST
            Return K'
```

5.3 Analysis of the attack

To analyze this attack we first bound the oracle-query complexity of testing each r. We then compute how many r's are needed in order to succeed with probability ϵ.

We say that r is *good* if it is equal to $x_j \oplus k1$ or $x_j \oplus C \oplus k1$ for some j. If r is good then as soon as the attacker tries $k' = k$ (remember she tries them all) she will obtain the correct values for $k1'$ and then $k2'$ (though she may try some incorrect values as well).

We now bound the expected cost of trying each r. For each value of r (good or bad), the attacker must go through, in the worst case, all 2^κ values for k'. For each value of k', it makes 2 calls to the F oracle in order to compute $F^{\Delta}_{k'}(r)$, giving a base cost of $2^{\kappa+1}$ calls to the F oracle. Given a promising (j, k'), where $F^{\Delta}_{k'}(r) = FX^{\Delta}_K(x_j)$, the attacker generates 2 guesses $k1'$, and for each $k1'$ she makes an additional call to the F oracle to compute $k2'$. Testing $k', k1'$ and $k2'$ requires no further oracle calls.

We note that for any r, when $k' \neq k$ the distribution on $F^{\Delta}_{k'}(r)$ is random even conditioned on the answers to all of the FX oracle queries. Thus, the expected number of j such that (k', j) is promising is at most $m/2^{n+1}$. When $k' = k$, then in the worst case, $m/2$ promising values of (k', j) are tested. Therefore, the expected extra number of oracle queries needed to evaluate promising candidates is at most $m + m2^\kappa/2^n$ for each value of r selected. Thus, for each random r selected, a total of at most

$$2^{\kappa+1} + m + m2^{\kappa-n}$$

expected queries are required.

It remains to bound the number of r's that must be tried in order to select a good r with probability at least ϵ. There are exactly m good r out of 2^n possibilities. Thus, the probability that ℓ randomly selected values for r will fail to be good is at most $(1 - m/2^n)^\ell$. We thus need to select ℓ so that $(1 - m/2^n)^\ell \leq 1 - \epsilon$. Using the identity $(1 + a)^b \leq e^{ab}$, it suffices to achieve

$$e^{-m\ell/2^n} \leq 1 - \epsilon,$$

or equivalently,

$$\ell \geq \frac{-2^n \ln(1-\epsilon)}{m}.$$

For $0 < \epsilon < \frac{1}{2}$, $-\ln(1-\epsilon) < \epsilon + \epsilon^2$, so it suffices that

$$\ell \geq \frac{2^n(\epsilon + \epsilon^2)}{m}.$$

Summarizing the above, there is an attack which finds a consistent key $K' = k'.k1'.k2'$ with probability ϵ using m queries to the FX_K oracle, and at most expected

$$\left(2^{n+\kappa+1-\lg m} + 2^n + 2^\kappa\right)\left(\epsilon + \epsilon^2\right)$$

queries to the F/F^{-1} oracles. The theorem follows. $\diamondsuit$

6 Open Problems and Conclusions

Analysis of other multiple encryption schemes. The model we have used to upper bound the worth of key search applies to many other block-cipher based constructions. For example, it would be interesting to apply this model to bound the maximal advantage an adversary can get for triple DES with three distinct keys, or triple DES with the first and third keys equal, or the method of [4]. It would be interesting to demonstrate that some construction has a better effective key length than DESX (e.g., $k + n - 1$ bits).

Use it! Work within some standards bodies continues to specify encryption based on DES in its most customary mode of operation. We recommend DESX (or one of its variants, as in Section 4). DESX is efficient, DES-compatible, patent-unencumbered, and resists generic key-search attacks. In virtually every way, DESX would seem to be a better DES than DES.

Acknowledgments

Mihir Bellare was closely involved in the early stages of our investigation. Burt Kaliski, Ron Rivest, and anonymous referees provided useful comments and information.

References

[1] E. Biham and A. Biryukov, "How to strengthen DES using existing hardware." *Advances in Cryptology— ASIACRYPT '94.* Springer-Verlag (1994).

[2] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag (1993).

[3] M. Blaze, "A cryptographic file system for UNIX." *1st ACM Conference on Computer and Communications Security,* 9–16 (November 1993).

[4] D. Coppersmith, D. Johnson and M. Matyas, "A proposed mode for triple-DES encryption." IBM Journal of Research and Development, vol. 40, no. 2, 253–261 (1996).

[5] J. Daemen, "Limitations of the Even-Mansour construction" (abstract of a rump-session talk). *Advances in Cryptology— ASIACRYPT '91.* Lecture Notes in Computer Science, vol. 739, 495–498, Springer-Verlag (1992).

[6] W. Diffie and M. Hellman, "Exhaustive cryptanalysis of the NBS Data Encryption Standard." *Computer*, vol. 10, no. 6, 74–84 (June 1977).

[7] Electronic Frontier Foundation, *Cracking DES: Secrets of Encryption Research, Wiretap Politics, & Chip Design*. O'Reilly and Associates (1998).

[8] S. Even and Y. Mansour, "A construction of a cipher from a single pseudorandom permutation." *Journal of Cryptology*, vol. 10, no. 3, 151–162 (Summer 1997). Earlier version in *Advances in Cryptology— ASIACRYPT '91.* Lecture Notes in Computer Science, vol. 739, 210–224, Springer-Verlag (1992).

[9] B. Kaliski, *personal communication* (April 1996).

[10] B. Kaliski and M. Robshaw, "Multiple encryption: weighing security and performance," *Dr. Dobb's Journal*, 123–127 (January 1996).

[11] J. Kilian and P. Rogaway, "How to protect DES against exhaustive key search." *Advances in Cryptology— CRYPTO '96.* Lecture Notes in Computer Science, vol. 1109, 252–267, Springer-Verlag (1996). Earlier version of this paper.

[12] M. Matsui, "The first experimental cryptanalysis of the data encryption standard." *Advances in Cryptology— CRYPTO '94.* Lecture Notes in Computer Science, vol. 839, 1–11, Springer-Verlag (1994).

[13] R. Rivest, *personal communication* (1995, 1996).

[14] RSA Data Security, Inc.. Product documentation, "Mailsafe Note #3."

[15] C. Shannon, "Communication theory of secrecy systems." Bell Systems Technical Journal, 28(4), 656–715 (1949).

[16] P. van Oorschot and M. Wiener, "Parallel collision search with cryptanalytic applications." *Journal of Cryptology*, vol. 12, no. 1, 1–28 (1999) Earlier version in *2nd ACM Conference on Computer and Communications Security,* 210–218 (1994).

[17] M. Wiener, "Efficient DES key search." Technical Report TR-244, School of Computer Science, Carleton University (May 1994). Reprinted in *Practical Cryptography for Data Internetworks,* W. Stallings, editor, IEEE Computer Society Press, 31–79 (1996).

[18] Y. Yin, The 1995 RSA Laboratories Seminar Series, "Future directions for block ciphers." Seminar proceedings (page 23) for a talk given in Redwood Shores, California (August 1995).

A Proof of Claim 3.2

We first define a new game, denoted Game X', which matches more directly the definition of the experiment defining P_X. Game X' is defined in Figure 3.

First, note that no adversary can distinguish between playing Game X' and playing with oracles $\langle FX_K, F, F^{-1}\rangle$ drawn according to the experiment defining P_X. Indeed the only difference

Initially, let F be undefined. Randomly choose $k^* \xleftarrow{R} \{0,1\}^\kappa$, $k_1^*, k_2^* \xleftarrow{R} \{0,1\}^n$. Then answer each query the adversary makes as follows:

$\boxed{E}$ On oracle query $E(P)$:

1. If $F_{k^*}(P \oplus k_1^*)$ is defined, return $F_{k^*}(P \oplus k_1^*) \oplus k_2^*$.
2. Otherwise, choose y uniformly from $\overline{\mathrm{Range}}(F_{k^*})$, define $F_{k^*}(P \oplus k_1^*) = y$ and return $y \oplus k_2^*$.

$\boxed{F}$ On oracle query $F_k(x)$:

1. If $F_k(x)$ is defined, return $F_k(x)$.
2. Else, choose $y \in \{0,1\}^n$ uniformly from $\overline{\mathrm{Range}}(F_k)$, define $F_k(x) = y$ and return y.

$\boxed{F^{-1}}$ On oracle query $F_k^{-1}(y)$:

1. If $F_k^{-1}(y)$ is defined, return $F_k^{-1}(y)$.
2. Else, choose $x \in \{0,1\}^n$ uniformly from $\overline{\mathrm{Dom}}(F_k)$, define $F_k(x) = y$ and return x.

Figure 3: Game X'

between these scenarios is that Game X' generates values for E and F by "lazy evaluation," whereas the experiment defining P_X would generate these values all at the beginning. Thus $\Pr_{X'}\left[A^{E,F,F^{-1}} = 1\right] = P_X$.

We want to show that $\Pr_X\left[A^{E,F,F^{-1}} = 1\right] = \Pr_{X'}\left[A^{E,F,F^{-1}} = 1\right]$: no adversary A can distinguish whether she is playing Game X or X'. We emphasize that A's ability to distinguish between Games X and X' is based strictly on the input/output behavior of the oracles; the adversary can not see, for example, whether or not the flag ***bad*** has been set.

We will show something even stronger than that Games X and X' look identical to any adversary. Observe that both Game X and Game X' begin with random choices for k^*, k_1^* and k_2^*. We show that, for any particular values of k^*, k_1^* and k_2^*, Game X with these initial values of k^*, k_1^* and k_2^* is identical, to the adversary, to Game X' with these same initial values of k^*, k_1^* and k_2^*. So, for the remainder of the proof, we consider k^*, k_1^* and k_2^* to have fixed, arbitrary values.

A basic difference between Games X and X' is that Game X separately defines both E and F_{k^*} while Game X' only defines F_{k^*} and computes $E(P)$, in response to a query P, by $F_{k^*}(P \oplus k_1^*) \oplus k_2^*$. The essence of our argument is that Game X can *also* be viewed as answering its $E(P)$ queries by referring to F_{k^*}. But, strictly speaking, it's not really F_{k^*} which can be consulted. We get around this as follows.

Given partial functions E and F_{k^*}, these functions having arisen in Game X, define the partial function $\widehat{F}_{k^*}$ by

$$\widehat{F}_{k^*}(x) = \begin{cases} F_{k^*}(x) & \text{if } F_{k^*}(x) \text{ is defined,} \\ E(x \oplus k_1^*) \oplus k_2^* & \text{if } E(x \oplus k_1^*) \text{ is defined, and} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Thus, in executing Game X, defining a value for E or F_{k^*} can implicitly define a new value for $\widehat{F}_{k^*}$.

At face value, the above definition might be inconsistent—this could happen if both $F_{k^*}(x)$ and $E(x \oplus k_1^*)$ are defined for some x, and with "clashing" values (ie., values which do not differ by k_2^*). Before we proceed, we observe that this can never happen:

Claim A.1 *Let E and F_{k^*} be partial functions which may arise in in Game X. Then the function $\widehat{F}_{k^*}$, as described above, is well-defined.*

The proof is by induction on the number of "Define" steps (Steps E-3, F-3, or F^{-1}-3) in the definition of Game X, where points of $\widehat{F}_{k_*}$ become defined as Game X executes. The basis (when E and F^{-1} are completely undefined) is trivial. So suppose that, in step E-3, we set $E(P) = C$. Is it possible that this definition of $E(P)$ will cause $\widehat{F}_{k_*}$ to become ill-defined? The only potential conflict is between the new $E(P)$ value and a value already selected for $F_{k^*}(P \oplus k_1^*)$. So if $F_{k^*}(P \oplus k_1^*)$ was not yet defined, there is no new conflict created in Step E-3. If, on the other hand, $F_{k^*}(P \oplus k_1^*)$ was already defined, then its value, by virtue of Step E-2, is $E(P) \oplus k_2^*$. This choice results in $\widehat{F}_{k^*}$ remaining well-defined. The analysis for the cases corresponding to Steps F-3 and F^{-1}-3 is exactly analogous, and is omitted. $\diamondsuit$

The function $\widehat{F}_{k^*}$, as defined for Game X, also makes sense for Game X', where $\widehat{F}_{k^*}(x) = F_{k^*}(x)$. Our strategy, then, is to explain the effect of each E, F_{k^*}, and $F_{k^*}^{-1}$ query strictly in terms of $\widehat{F}_{k^*}$. We then observe that Game X' responds to its oracle queries in an absolutely identical way. This suffices to show the games equivalent.

Case 1. We first analyze the behavior of Game X on oracle query $E(P)$. To begin, note that Game X never defines the value of $E(P)$ unless it has received P as a query. So since A never repeats queries (see the assumptions just following the theorem statement) $E(P)$ must be undefined at the time of query P. Consequently, at the time of query P, $\widehat{F}_{k^*}(P \oplus k_1^*)$ will be defined iff $F_{k^*}(P \oplus k_1^*)$ is defined, and $\widehat{F}_{k^*}(P \oplus k_1^*) = F(P \oplus k_1^*)$. *Case 1a.* When $\widehat{F}_{k^*}(P \oplus k_1^*)$ is defined, then Game X returns the value of $C = \widehat{F}_{k^*}(P \oplus k_1^*) \oplus k_2^*$. In this case, setting $E(P) = C$ leaves $\widehat{F}_{k^*}$ unchanged. *Case 1b.* When $\widehat{F}_{k^*}(P \oplus k_1^*)$ is undefined, then C is repeatedly chosen uniformly from $\overline{\text{Range}}(E)$ until $F_{k^*}^{-1}(C \oplus k_2^*)$ is undefined. By the definition of $\widehat{F}_{k^*}$ it follows that $y = C \oplus k_2^*$ is uniformly distributed over $\overline{\text{Range}}(\widehat{F}_{k^*})$. In this case, setting $E(P) = C$ sets $\widehat{F}_{k^*}(P \oplus k_1^*) = y$.

Now compare the above with Game X' on query $E(P)$. When $F_{k^*}(P \oplus k_1^*)$ is defined, then $C = F_{k^*}(P \oplus k_1^*) \oplus k_2^*$ is returned and no function values are set. When $F_{k^*}(P \oplus k_1^*)$ is undefined, y is chosen uniformly from $\overline{\text{Range}}(F_{k^*})$, $F_{k^*}(P \oplus k_1^*)$ is set to y (and implicitly $\widehat{F}_{k^*}(P \oplus k_1^*)$ is set to y), and $C = y \oplus k_2^*$ is returned. Thus, the behavior of Game X' on query $E(P)$ is identical to the behavior of Game X on query $E(P)$.

Case 2. We will be somewhat briefer with our analyses of the F and F^{-1} oracles, which are similar to the analysis above. *Case 2a.* On oracle query $F_k(x)$, when $k \neq k^*$ then the behavior of Game X is clearly identical to Game X'. *Case 2b.* When $k = k^*$ then $F_{k^*}(x)$ is defined iff a query of the form $E(x \oplus k_1^*)$ has been made. This holds iff $\widehat{F}_{k^*}(x)$ is defined (since $F_{k^*}(x)$ would not have been queried before). By a straightforward argument the value y returned from the query $F(x)$ will then be $y = E(x \oplus k_1^*) \oplus k_2^* = \widehat{F}_{k^*}(x)$ in both games. *Case 2c.* When $\widehat{F}_{k^*}(x)$ is undefined, then in both games y is uniformly chosen from $\overline{\text{Range}}(\widehat{F}_{k^*})$ and $\widehat{F}_{k^*}(x)$ is defined to be y. Thus, in all cases, Game X behaves identically to Game X'.

Case 3. Finally, on oracle query $F_k^{-1}(y)$, the case $k \neq k^*$ is again trivial. When $k = k^*$, then $\widehat{F}_{k^*}^{-1}(y)$ will be defined iff $E^{-1}(y \oplus k_2^*)$ is defined, in which case $x = E^{-1}(y \oplus k_2^*) \oplus k_1^* = \widehat{F}_{k^*}^{-1}(y)$ in both games. When $\widehat{F}_{k^*}^{-1}(y)$ is undefined, then in both games x is chosen uniformly from $\overline{\mathrm{Dom}}(\widehat{F}_{k^*})$ and $\widehat{F}_{k^*}(x)$ is defined to be y. Again, Game X behaves identically to Game X'.