# The Security of DESX

Phillip Rogaway
Department of Computer Science
Engineering II Building
University of California
Davis, CA 95616 USA

DRAFT 2.0
July 5, 1996

With keys of just 56-bits, the susceptibility of DES to exhaustive key search has been a concern since the cipher was first made public. Now a simple extension of DES, called DESX, has been shown to be virtually immune to exhaustive key search. This note describes the DESX construction and the sense in which it is has been proven sound. The construction is due to Ron Rivest [4], while its soundness proof is due to Joe Kilian and myself [3].

### Strengthening DES

Despite impressive results on the differential and linear cryptanalysis of DES, the only practical attack described to date remains exhaustive key search. The cost of this attack keeps going down. In 1993 Wiener provided a careful estimate which showed that for $1 million one could build an engine which, given a single ⟨plaintext, ciphertext⟩ pair, could recover the key in about 3.5 expected hours [5]. It would seem that DES, when used in its most customary manner, has already become a rather marginal choice for meeting commercial data privacy requirements.

An attractive way to overcome DES's susceptibility to key search is to build a new block cipher out of DES, leaving alone DES's internal structure. One advantage of this approach is that it can preserve the assurance benefits which DES has gained over the years. Another advantage is that it allows one to gainfully employ existing DES hardware and software. Finally, making a new block cipher which is to be used in standard ways is more general and more conducive to analysis than coming up with entirely new DES-based modes of operation for specific higher-level tasks such as encryption.

The most well-known suggestion to strengthen DES is "triple DES," one version of this being defined by $\mathrm{EDE3}_{k1.k2.k3}(x) = \mathrm{DES}_{k3}(\mathrm{DES}^{-1}_{k2}(\mathrm{DES}_{k1}(x)))$. That is, the key for EDE3 is $56 \times 3 = 168$ bits, and one enciphers a 64-bit block by enciphering under one 56-bit subkey, deciphering under a second, then enciphering under a third. (The reason the second step is $\mathrm{DES}^{-1}_{k2}$ and not $\mathrm{DES}_{k2}$ is for DES-compatibility: set $K = k.k.k$ to make $\mathrm{EDE3}_K = \mathrm{DES}_k$. The reason for using DES three times instead of two is the existence of "meet-in-the-middle" attacks on double DES.)

The problem with triple DES is that it is much slower than DES itself—roughly a third the speed. When cipher block chaining EDE3, this slowdown will occur in hardware (even if one tries to compensate with additional hardware) as well as in software. In many situations, these performance penalties are unacceptable.

There is, then, a need for a *cheap* way to strengthen DES against key-search attack, injecting extra key bits without impacting the cipher's internal structure. Back in 1984, Ron Rivest came up with just such an extension of DES. It is called DESX.

### The DESX construction

DESX is defined by

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

That is, a DESX key $K = k.k1.k2$ consists of 56+64+64 = 184 bits comprising three different subkeys: a "DES key" $k$, a "pre-whitening key" $k1$, and a "post-whitening key" $k2$. To encrypt a message block we XOR it with $k1$, DES encrypt under $k$, then XOR with $k2$. Thus the work to DESX encrypt a message block is just two XORs more than the work to DES encrypt a message block.

The amazing thing about DESX is that these two XOR operations render the cipher much less susceptible to key-search attack. In the sequel we will quantify just how much the "DESX trick" really buys. Here we won't try to impart any intuition as to *why* DESX works, except to suggest that the pre- and post- whitening make it difficult for the attacker to single out even one valid $\langle x_i, \mathrm{DES}_k(x_i)\rangle$ pair when the attacker mounts a chosen-plaintext attack to get many $\langle P_j, \mathrm{DESX}_K(P_j)\rangle$ pairs.

### What is key search?

We want to emphasize that what we show in [3] is that there is no feasible *key-search* attack on DESX; we don't know that there's not a feasible attack of some other type. Thus it is essential to understand what exactly we mean by a key-search attack on DESX. Understanding this may take a careful reading or two.

Sometimes people think of a key-search attack as one in which the adversary systematically explores some universe of possible keys. But we mean something broader: we claim that the essence of key search is that the adversary carries out her mission in a way which doesn't exploit the internal structure of the underlying cipher— she uses the underlying cipher (e.g., DES) as a *black box.*

Now it's a very nebulous thing what we just said— what does it *mean* to use DES only as a black box? How can we make this idea formal? The first step to an answer, oddly enough, is to generalize the DESX construction.

For *any* block cipher $F$ we can define block cipher $FX$ by $FX_{k.k1.k2}(x) = k2 \oplus F_k(k1 \oplus x)$. Of course DESX = $FX$ when $F$ = DES.

Now if you've come up with a way to break DESX and it doesn't use anything structural about DES (that is, it uses DES as a "black box"), then your method ought to break $FX$ for any $F$ with a 56-bit key and a 64-bit block size. In other words, we can recast the problem *analyze how well* DESX *resists key-search attack* to the problem *analyze how well FX resists key-search attack, for some other function F.* But if we just switch from looking at $FX$ with $F$ = DES to looking at $FX$ with $F$ being some other particular function, then we haven't gained anything: whatever function $F$ we choose, it also might have structure the adversary could exploit.

To overcome this difficulty we will demand that the adversary attack $FX$ for a *random* block cipher $F$. An adversary can't exploit structure in a random block cipher because there is no structure to exploit! The *thesis* underlying our analysis is that for any particular function $F$, we can measure how well a *key-search* adversary can attack $FX$ by measuring how well an *arbitrary* adversary can attack the $FX$-construction for a *random* block cipher $F$. In particular, it is our thesis that one can measure how well a key-search adversary can attack DESX by measuring how well an arbitrary adversary can attack the $FX$-construction, when $F$ is a random block cipher with a key length and block length to match DES.

### The formal model

Not using the structure of DES is one thing, but even a key-search adversary on DESX deserves the ability to compute DES or $\mathrm{DES}^{-1}$ at points of her choosing. Analogously, if we're going to ask an adversary to attack $FX$ it's only fair to give her the capability to compute $F$ and $F^{-1}$. So we will provide our adversary with "black boxes" to compute these. The $F$ black box, on input $(k, x)$, computes $F_k(x)$. The $F^{-1}$ black box, on input $(k, y)$, computes $F_k^{-1}(y)$.

Now we're ready to make quantitative the security of the $FX$ construction: we will define the *advantage* an adversary $A$ obtains in attacking it.

Fix parameters $\kappa$ (the key length) and $n$ (the block

length). Choose a random block cipher $F$ which uses a $\kappa$-bit key and an $n$-bit block length. This means to choose a random permutation for each $\kappa$-bit key $k$. We start off by giving adversary $A$ black boxes for $F$ and $F^{-1}$. This affords $A$ the ability to compute the underlying block cipher without letting her exploit its structure. Next we need to present $A$ with a "test" to see how well she can attack $FX$.

Here is that test. We present $A$ with one of two types of encryption oracles. Which type $A$ gets is chosen at random.

- A **good** encryption oracle is one which chooses a random $(\kappa + 2n)$-bit $FX$-key, $K$, and then answers each $n$-bit question $P$ by $FX_K(P)$.
- A **bogus** encryption oracle is one which chooses a random permutation $\pi$ on the space of $n$-bit blocks and then answers each $n$-bit question $P$ by $\pi(P)$.

To win the above "$FX_K$-or-$\pi$" game, $A$ is supposed to correctly identify if she was given a **good** encryption oracle or a **bogus** encryption oracle. The *advantage* of $A$ is defined as the probability that $A$ answers that she has a **good** encryption oracle given that we give her a **good** encryption oracle, minus the probability that $A$ answers that she has a **good** encryption oracle given that we give her a **bogus** encryption oracle. An advantage of 1 indicates the the adversary is doing a great job of distinguishing $FX_K$ from a random permutation $\pi$ unrelated to $F$; she always answers correctly. An advantage of 0 indicates that $A$ is doing a worthless job of making this distinction; she could just as well answer by flipping a coin. An advantage of $-1$ indicates that $A$ is doing a terrible job of guessing; in this case, she'd do well to just swap when she says "**good**" and when she says "**bogus**."

We have given the adversary what would seem to be a very easy game— much easier, say, then asking her to try to recover the key $K$ when presented with a bunch of $FX_K$-encrypted points. Indeed if there were an adversary $A$ who could recover $K$ from $FX_K$-encrypted points, then there would be another adversary $A'$ who, with resources comparable to $A$'s, could get an advantage of $\approx 1$ in the $FX_K$-or-$\pi$ game. In general, we use the $FX_K$-or$\pi$ game because adopting this liberal notion of adversarial success makes our results stronger: if no adversary can distinguish $FX_K$ from a random permutation, then no adversary can break $FX$ in any useful way.

### Like having 118−lg $m$ bits

Having defined the $FX_K$-or-$\pi$ game and the advantage of an adversary achieves in playing it, it becomes possible to analyze the maximal advantage that *any* adversary can obtain, assuming that the adversary expends only some specified computational effort. We won't describe any details of this analysis; we'll just state the final answer. See [3] for the proof. That proof and its setup builds, in turn, on [1].

Consider the maximal advantage that any adversary can achieve in the $FX_K$-or-$\pi$ game when the adversary asks a total of $t$ questions of her $F/F^{-1}$ black boxes and a total of $m$ questions of her $FX_K$-or-$\pi$ encryption oracle. This value, $MaxAdv$, depends on $m$ and $t$, as well as on $F$'s block length $n$ and key length $\kappa$. The main result of [3] says that $MaxAdv(m, t, n, \kappa) \leq mt \cdot 2^{-n-\kappa+1}$.

Since each query to an $F/F^{-1}$ black box takes at least one unit of time, we see that an adversary's which runs in time $T$ can get advantage which is at most $T \cdot 2^{-n-\kappa+1+\lg m}$. So another way to state the result of [3] is that the effective key length of the $FX$-construction, with respect to key search, is at least $n + \kappa - 1 - \lg m$ bits.

Let's apply the above result. Assume that you use DESX on a given key to encrypt at most $2^{30}$ blocks. Use the thesis stated earlier. Then a key-search adversary attacking DESX which runs in time $T$ could gain an advantage of at most $T \cdot 2^{-56-64+1+30} = T \cdot 2^{-89}$. For example, if $T < 2^{70}$ then the adversary's advantage is less than $2^{-19}$.

Now the truth is that there is one structural property of DES which attacks sometimes do exploit: it is the DES key-complementation property: $\mathrm{DES}_k(x) = \overline{\mathrm{DES}_{\overline{k}}(\overline{x})}$. We can "factor out" this property (that is, allow the adversary to exploit this particular structural property) by fixing some particular bit of the DES key. This reduces the DES key length by 1. Thus, taking account of the DES key-complementation property and then apply-

ing our thesis, we get that the effective key length of DESX, with respect to key search, is at least $55 + 64 - 1 - \lg m = 118 - \lg m$ bits.

### Concluding remarks

DESX should be used just as one would use DES. For example, instead of applying cipher block chaining (CBC) to DES, one would apply it to DESX. DESX interacts particularly nicely with CBC, as DESX-CBC encryption amounts to masking the plaintext with key material, then DES-CBC encrypting, then masking what results with more key material. In particular, hardware which performs DES-CBC encryption can be gainfully employed to perform the non-trivial part of DESX-CBC encryption.

A minor inconvenience of DESX is its strange key size. In applications it might 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. As an example, in RSA DSI's BSAFE 3.0 implementation of DESX, the underlying key *key* may be 128 bits, in which case the DES key and the pre-whitening key are determined directly from *key*, while the post-whitening key is a complex function of all 16 bytes of *key*.

DESX was intended to *improve* DES's strength against key search, and *preserve* its strength with respect to other possible attacks. But as Kaliski and Robshaw indicate, DESX actually does add security against differential and linear cryptanalysis [2], increasing the required number of known or chosen plaintexts to be in excess of $2^{60}$. Further added strength against these attacks would seem to be achieved by replacing the XOR operations of DESX by addition, as in DES-PEP$_{k.k1.k2}(x) = k1 +$ DES$_k(k2+x)$, where $L.R+L'.R' = (L\hat{+}L')\ .\ (R\hat{+}R')$, $|L| = |R| = |L'| = |R'| = 32$, and $\hat{+}$ denotes addition modulo $2^{32}$. The $\kappa + n - 1 - \lg m$ bound of [3] also applies to variants such as this one.

As we've explained, our results don't say that it's infeasible to build a machine which would break DESX in a reasonable amount of time. But they do imply that such a machine would have to employ some radically new idea: it couldn't be a machine implementing a key-search attack, in the general sense which we've described.

DESX would seem, in virtually every sense, to be a "better DES than DES." It is simple, DES-compatible, efficiently implementable in hardware, essentially the same speed as single DES in software, can profitably use existing DES hardware, and it has been proven, in a strong sense, to add much strength against exhaustive key search. Emerging standards which specify DES or triple DES would do well to consider DESX.

## References


[1] S. Even and Y. Mansour, "A construction of a cipher from a single pseudorandom permutation." Asiacrypt '91.

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

[3] J. Kilian and P. Rogaway, "How to protect DES against exhaustive key search." Crypto '96, and http://wwwcsif.cs.ucdavis.edu/~rogaway.

[4] R. Rivest, *personal communication.*

[5] M. Wiener, "Efficient DES key search." Manuscript of August 20, 1993, and Technical Report TR-244, School of Computer Science, Carleton University, May 1994.