The Security of MacGuffin

David A. Wagner
April 19, 1995

A thesis presented to
PRINCETON UNIVERSITY
in partial fulfillment of the requirements for the degree of
BACHELOR OF ARTS
in the
DEPARTMENT OF MATHEMATICS

I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions for the purpose of scholarly research.

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The Security of MacGuffin

David A. Wagner
Princeton University
Princeton, NJ 08544
dawagner@princeton.edu

April 19, 1995

Abstract

This thesis examines MacGuffin, a recently proposed block cipher based on a new architecture similar to that of DES. The analysis begins with results from a variety of standard statistical tests. Next, a comparison of the resistance of MacGuffin and DES against several elementary attacks is presented. MacGuffin's strength against differential cryptanalysis is investigated, and a differential attack requiring $2^{51.5}$ chosen plaintexts is described. MacGuffin is shown to be weaker than DES against differential cryptanalysis. A weakness of the MacGuffin key schedule is examined. Finally, implications on the security of MacGuffin are discussed.

1 Introduction

The principles of block cipher design have been a focus of intense research in cryptography for the past two decades. The public unveiling of DES (the NBS Data Encryption Standard [1]) was a major landmark in the study of block ciphers, arousing interest in cryptography and cryptanalysis. By modifying a design element of DES and studying how this change affects security, researchers have gained insight into the exceptional strength of DES in particular and Feistel ciphers in general. In retrospect, we can see that DES is surprisingly well-designed [2], and research into the inner workings of DES continues to stimulate important advances.

A tremendous assortment of DES variants have been proposed over the years. Many have suggested using independent subkeys for each round [3]; some suggested using different or variable S-boxes. Schneier [4] includes Feal, GDES, IDEA, Khafre, Khufu, LOKI, Luby-Rackoff, Lucifer, Madryga, NewDES, REDOC, and RC2 among the list of important Feistel ciphers. Blowfish and RC5, two more recent encryption algorithms, are also structured very much like DES. Important lessons about the design of block ciphers have been learned from study of these DES variants.

In keeping with this long tradition, two researchers have proposed a new DES variant called MacGuffin [5]. DES is a balanced Feistel network: it splits the input up in half into two registers, operating on the left half every round. In contrast, a generalized unbalanced Feistel network (a GUFN) need not split the register evenly into two equal-sized parts. To illustrate the power of GUFNs, Blaze and Schneier have offered MacGuffin as an example of their approach, intending to incite debate and further research.

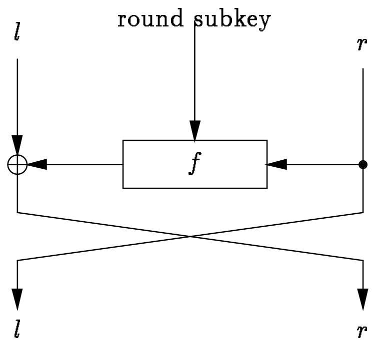
MacGuffin looks provocatively similar to DES, so a natural first step is to compare the security of MacGuffin and DES. All of the security in a Feistel network comes from a nonlinear function applied once per round. The DES round function operates on a 32 bit register and produces a 32 bit result, whereas MacGuffin takes 48 bits of input and yields 16 output bits. Because each round of MacGuffin affects just half as much of the block as DES, Blaze and Schneier suggest that two rounds of MacGuffin would equal approximately the strength of one DES round; thus, they conclude, MacGuffin with 32 rounds should be at least as secure as DES with 16 rounds [5]. This thesis is intended to examine the cryptographic strength of MacGuffin and to test that central hypothesis.

2 Description of MacGuffin and DES

This section provides only a cursory introduction to DES and MacGuffin. A more comprehensive treatment can be found in many texts—the original DES standard can be found in [1], while [4, 6, 7] provide more readable descriptions. MacGuffin is fully described and implemented in [5]. For a general introduction to cryptography and block ciphers, the reader is referred to [4, 8, 6, 7, 9].

2.1 DES

In any (balanced) Feistel network, the plaintext block is split up evenly into a left half and a right half. DES uses 64 bit blocks, so the left and right registers are 32 bits wide. A pictorial view of one DES round is shown in Figure 1.



$$\begin{array}{rcl} l & \leftarrow & l \oplus f(r) \\ (l, r) & \leftarrow & (r, l) \end{array}$$

Figure 1: The DES round structure

DES permutes each input plaintext block according to an initial permutation, iterates the round operation for 16 rounds, and then permutes the result by a final permutation to obtain the resulting ciphertext block. The final permutation consists of a swap of the left and right halves, to facilitate implementation of decryption, followed by the inverse of the initial permutation. As the initial and final permutations are unkeyed, they will have no cryptographic significance in the discussion to follow. For simplicity, the DES initial and final permutations will be ignored from here on.

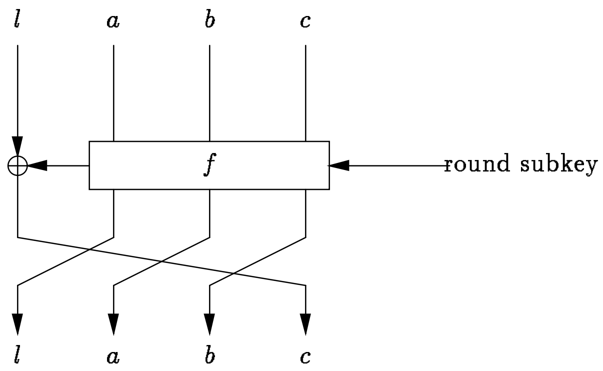
The DES f function expands a 32 bit input to 48 bits by duplicating some bits, xors this with the 48 bit round subkey, and then subjects the result to 8 substitution boxes (S-boxes). Each S-box is a non-linear function mapping 6 bit values to 4 bit outputs with carefully designed structure. Finally, the output from the S-boxes is gathered together and permuted to yield the output of f.

It's clear that much of the cryptographic strength in a Feistel network comes from careful choice of the f round function. The DES f function is cryptographically weak on its own, but it is believed to become quite secure when iterated for sufficiently many rounds. The properties of the DES f mapping have been well explored: for example, it is not bijective, though if two inputs produce the same output, there must be at least 3 S-boxes with differing inputs [10, 2].

The key scheduling algorithm specifies how to derive the round subkeys from the encryption key. DES uses a completely linear key schedule; in each round, 48 bits are selected from the 56 bit DES key. If the key schedule were removed and all round subkeys were the same, DES would be vulnerable to a particularly nasty attack [11]. In fact, these attacks have been carefully avoided in the DES key schedule.

2.2 MacGuffin

MacGuffin is an example of a generalized unbalanced Feistel cipher (a GUFN). MacGuffin splits the 64 bit plaintext block into four 16 bit registers l, a, b, and c. A representation of one MacGuffin round is shown in Figure 1.



$$
\begin{aligned}
l &\leftarrow l \oplus f(a,b,c) \\
(l,a,b,c) &\leftarrow (a,b,c,l)
\end{aligned}
$$

Figure 2: The MacGuffin round structure

MacGuffin uses twice as many rounds as DES, for a total of 32 iterations of the round function. Because each MacGuffin round affects just half as many bits as each DES round, it is suggested that twice as many rounds of MacGuffin achieve similar security to DES. One can see that two rounds of MacGuffin cannot be expressed as one round of a balanced Feistel cipher, because MacGuffin's f mapping is a function of more than just the right half of the input.

The MacGuffin f function permutes the 48 bit input, xors it with the round subkey, and applies 8 S-boxes ($S1$, $S2$, etc.) to the result. Each S-box is a boolean function mapping 6 bit strings to 2 bit strings, derived from the DES S-boxes by throwing away the middle 2 bits of their output. Finally, the outputs from all the S-boxes are gathered together and become the output of f. Notably, there is no expansion or duplication of bits before application of the S-boxes; thus, each bit of (a, b, c) affects only two bits of l. In contrast, each bit of the input to the DES f function affects four or eight bits of its output.

Unlike DES, MacGuffin has no initial or final permutation. The output is not rotated after the last round, so the decryption algorithm differs slightly from the encryption algorithm. Specifically, the direction of the rotations are reversed and the round subkeys are applied in reverse order for decryption.

The MacGuffin key schedule is quite complex. The key scheduling algorithm essentially hashes the key to obtain the round subkeys, which makes it harder for an attacker to derive the encryption key from partial or complete knowledge of the round subkeys. The key scheduling process is also intended to be computationally-intensive, which hopefully discourages exhaustive key search. A more complete description of the MacGuffin key schedule is given in Section 6. Most importantly, MacGuffin has a 128 bit key length to make the cost of exhaustive search prohibitive.

3 Statistical tests

One way to compare the strength of DES and MacGuffin is to examine reduced-round variants of the two block ciphers. Here are collected the results from applying a myriad of simple statistical tests on DES-r and MacGuffin-$2r$. (For convenience, DES-r will stand for the first r rounds of DES, with similar notation for MacGuffin.)

3.1 The avalanche property

In any good block cipher, toggling a bit in the plaintext input should change the ciphertext output unpredictably. The avalanche property holds if on average half the bits of the ciphertext change when one bit of the plaintext is changed [6, 12]. To test each cipher, 100 random keys and 500 plaintext blocks were chosen, for a total of $100 \cdot 500 \cdot (64 + 1) = 3250000$ encryptions per cipher tested.

Let the random variable X denote the average number of ciphertext bits toggled. For a good block cipher, the ciphertext bits should be indistinguishable from independent Bernoulli random variables. Thus, we should see the expectation $\mathbb{E}X = 32.0$; also X should have standard deviation

$$\frac{0.5\sqrt{64}}{\sqrt{100 \cdot 500 \cdot 64}} = 0.00224.$$

Three standard deviations is a convenient threshold for this test: the cipher is accepted if $|X - 32| \leq 0.00224 \cdot 3 = 0.0067$, which corresponds to a significance level of 0.3%. It turns out that 6 rounds of DES or 14 rounds of MacGuffin suffice to achieve the avalanche property. See Table 1 for a list of results.

Table 1: The avalanche property: average ciphertext bits toggled

r	DES-r	MacGuffin-$2r$
5	31.884	31.783
6	31.999	31.979
6.5	n/a	31.993
7	32.001	31.998

3.2 The strict avalanche property

The strict avalanche property is a generalization of the avalanche property: a block cipher is said to satisfy the strict avalanche property if changing one bit in the plaintext toggles each bit of the ciphertext with probability 0.5 [12]. To test for the strict avalanche property, 100 random keys and 500 random plaintext blocks were chosen. The random variable A_{ij} is defined to be the j-th bit of

$$E_k(P) \oplus E_k(P \oplus e_i)$$

for random k, P. Here e_i is the block with only the i-th bit set. Thus $100 \cdot 500 \cdot (64+1) = 3250000$ encryptions were required per cipher tested.

For a good cipher, A_{ij} is a random variable with mean $\mathbb{E}A_{ij} = 0.5$ and standard deviation $0.5/\sqrt{100 \cdot 500} = 0.00224$. Thus, there should be about $0.0026 \cdot 64^2 = 11$ matrix entries further than 3 standard deviations from the average. The strict avalanche property is achieved with 6 rounds of DES or 13 MacGuffin rounds. Table 2 contains a list of results.

Notice that the first avalanche test showed MacGuffin requires 14 rounds to achieve the avalanche property, while these measurements indicate that 13 rounds are enough to satisfy the strict avalanche property. There is no contradiction here: this is consistent with the imprecision and statistical error in the measurements.

Table 2: The strict avalanche test: non-random matrix entries

r	DES-r	MacGuffin-$2r$
5	365	644
6	10	28
6.5	n/a	9

3.3 Key-dependence avalanche

One can also formulate a variant of the avalanche criteria: in any good block cipher, toggling one key bit should change half the ciphertext bits on average. To test each cipher, 500 random keys and 100 plaintext blocks were chosen. Testing DES required a total of $500 \cdot 100 \cdot (56+1) = 2850000$ trial encryptions, and analyzing MacGuffin required $500 \cdot 100 \cdot (128+1) = 6450000$ encryptions.

If we let X be the average number of ciphertext bits changed when one key bit is changed as in Section 3.1, we see that again $\mathbb{E}X = 32.0$. With MacGuffin, X has standard deviation

$$\frac{0.5\sqrt{64}}{\sqrt{100 \cdot 500 \cdot 128}} = 0.00158.$$

For DES, X has standard deviation

$$\frac{0.5\sqrt{64}}{\sqrt{100 \cdot 500 \cdot 56}} = 0.00239.$$

This is not a double standard: DES has a shorter key, so fewer trials were needed, increasing the expected variance as shown above.

Again choosing three standard deviations as the threshold, we see that 6 rounds of DES or 5 rounds of MacGuffin suffice to achieve the key-based avalanche property; see Table 3 for a list of results. MacGuffin satisfies the key-dependent avalanche property with very few rounds because the key schedule generates the per-round subkeys by essentially hashing the 128 bit key. In contrast, DES uses a linear key schedule, which means more DES rounds are necessary to pass this test.

Table 3: Key-dependence: average ciphertext bits toggled

r	DES-r	MacGuffin-$2r$
1.5	n/a	24.001
2	11.517	31.996
2.5	n/a	32.000
4	30.869	31.999
5	31.944	32.001
6	32.000	32.000

3.4 Sequence complexity

One can use slightly more sophisticated statistical tests to measure whether $E_k(P) \oplus E_k(P \oplus e_i)$ appears to be randomly distributed, as should be the case for a good block cipher. Sequence complexity is one measure of the "randomness" of a bit string—it counts the number of new patterns appearing as one slides along the bits [13, 14]. For example, the sequence

$$0010111011001001 = 0/01/011/10110/0100/1$$

has sequence complexity 6.

Nearly all bit strings of length n have complexity at least $n/\log_2 n$, and so it is suggested in [13] and [14] to record how often the complexity of $E_k(P) \oplus E_k(P \oplus e_i)$ exceeds 10. The results of running this test for 100 random keys and 500 random plaintext blocks is shown in Table 4. In particular, DES-5 and MacGuffin-10 pass this test, but the variants with fewer rounds do not.

Table 4: Sequence complexity: non-random ciphertexts

r	DES-r	MacGuffin-$2r$
4	7.2%	8.4%
4.5	n/a	4.6%
5	4.1%	4.1%
16	4.0%	4.0%

3.5 Binary derivatives

Yet another test of randomness for bit strings counts the Hamming weight of repeated binary derivatives [15]. The binary derivative s' of a sequence s is defined by the equation

$$s'_i = s_i \oplus s_{i+1}$$

where s_i is the i-th bit of a sequence s. An example will explain it best:

$$\begin{aligned} s &= 0010111011001001 \\ s' &= 011100110101101 \\ s'' &= 10010101111011 \\ s''' &= 1011111000110 \\ \ldots \end{aligned}$$

Now let p_j denote the fraction of ones in the j-th binary derivative of s for $0 \le j \le k$, and set

$$\begin{aligned} r &= \max_j p_j - \min_j p_j \\ p &= \frac{1}{k+1} \sum_{j=0}^{k} p_j \end{aligned}$$

for some fixed value of k—[14] suggests using $k = 7$.

Note that r and p are random variables. According to [14], a block cipher should be accepted if $0.43 \le p \le 0.55$ and $r \le 0.33$ (with a 1% significance level). After using 100 random keys and 500 random plaintexts, the percentage of k, P that gave bad values of p and r is recorded in Table 5 and Table 6. It appears that 5 rounds of DES suffice to pass the binary derivative test, and 10 rounds are enough with MacGuffin.

One minor anomaly deserves mention. The threshold $r = 0.33$ was experimentally obtained for a 1% significance level in [14]; on the other hand, my examination of supposedly random data indicates that the $r = 0.33$ threshold is closer to a 2% significance level. Perhaps this imprecision arose because only 10000 random blocks were tested in [14], whereas the tests shown here analyzed at least 300 times as many blocks.

Table 5: Binary derivatives: bad values of p

r	DES-r	MacGuffin-2r
4	3.60%	3.96%
4.5	n/a	2.53%
5	1.96%	2.08%
16	1.95%	1.94%

3.6 Cycle Lengths

For any fixed key k, encryption by k induces a permutation on the set of plaintext blocks, so E_k may be considered as an element of the symmetric group on 2^{64} letters $\mathcal{S}_{2^{64}}$ with functional composition as the group operation. If MacGuffin is a good block cipher, each induced permutation should behave like a randomly chosen permutation. In particular, the sequence

$$a, E_k(a), E_k^2(a) = E_k(E_k(a)), E_k^3(a), E_k^4(a), \ldots$$

Table 6: Binary derivatives: bad values of r

r	DES-r	MacGuffin-$2r$
4	2.58%	1.93%
5	1.18%	1.21%
16	1.15%	1.16%

should have a period of approximately 2^{63} on average. Significantly shorter cycle lengths would indicate a weakness in MacGuffin.

Of course, the computing resources needed to iterate E_k 2^{63} times are rather difficult to acquire. For this reason, only a very limited version of the cycling test can be performed. I implemented an efficient parallel cycle detection algorithm [16] and iterated a random 128 bit key approximately 2^{33} times.

The test was repeated for the case where all round subkeys are set to 0. There does not appear to be any 128 bit encryption key which generates all zero subkeys under the MacGuffin key schedule; still, any short cycles here would be interesting. (For example, DES has very short cycles when all round subkeys are 0—every cycle is of length 1 or 2 [17].)

After 2^{33} encryptions, no cycles were found. It is important to point out that the logical implications of this result are very limited: one may conclude only that MacGuffin does not appear to suffer from certain cycling weaknesses. While lengthier computations might possibly discover short cycles in MacGuffin, completing a full test of cycle lengths is probably infeasible.

3.7 Small subgroup weaknesses

Identify the set of MacGuffin keys with the subset of $\mathcal{S}_{2^{64}}$ they induce, as before. We say MacGuffin is closed if the set of keys forms a subgroup of $\mathcal{S}_{2^{64}}$ under functional composition. Closed ciphers are subject to a meet-in-the-middle attack with complexity much smaller than exhaustive key search, so MacGuffin (for example) would fall with at most 2^{64} off-line trial encryptions if it were closed. More generally, if some subset of MacGuffin keys generates a small subgroup, this could indicate similar weaknesses.

Several papers have presented statistical evidence and conclusive proof that DES generates a large subgroup [18, 19, 20, 21]. Sadly, testing MacGuffin with any of these methods appears to be infeasible.

One statistical approach tests the cycling behavior of maps derived from MacGuffin encryption. Augment E_k to a map $\mathbb{Z}_2^{64n} \to \mathbb{Z}_2^{64n}$ according to

$$E_k(a_1, \ldots, a_n) = (E_k(a_1), \ldots, E_k(a_n)).$$

Fix $a, b \in \mathbb{Z}_2^{192}$ and search for k, k' so that $E_k(E'_k(a)) = b$. If MacGuffin is closed, there is only a 1 in 2^{64} chance that a useful key pair exists; conversely, if MacGuffin behaves like a random function, there will be approximately 2^{64} such key pairs on average. Thus, the test involves trying many random plaintext pairs a, b and searching for corresponding key pairs. If we find a key pair for each plaintext pair, then we can be assured that with high probability MacGuffin is not closed; if we fail to find any key pairs, then MacGuffin is probably closed. One can use cycling detection algorithms to find a useful key pair (if one exists) with about 2^{96} operations. Therefore computation of some cycle lengths allows one to form statistical conclusions on the likelihood that MacGuffin is closed. Unfortunately, the complexity of this test is far too large to be practical.

A closely related approach is based upon Lagrange's Theorem: the order of an element of a group divides the order of the group. Let G be the subgroup of $\mathcal{S}_{2^{64}}$ generated by the DES keys under functional composition, denote DES encryption under the all-zeros key by E_0, and define E_1 similarly. As it happens, the group element $E_0 \circ E_1$ often has an unusually short cycle length, i.e. an unusually small order in G [17]. Suppose we compute the cycle lengths l_j of the sequences formed by iterating $E_0 \circ E_1$ on random starting points a_j. Then we may conclude that $\mathrm{lcm}_j\, a_j$ divides $|G|$. In fact, this idea was successfully used to show that $|G| > 2^{57}$, so DES is certainly not closed [21]. Unfortunately, this method hinges upon finding a weak group element with order small enough to be computable; no such obvious candidates are available for testing MacGuffin.

Finally, one can sometimes establish by detailed algebraic arguments that a cipher is not closed. For example, Wernsdorf has shown that the DES round function generates the alternating group [20]. Regrettably, this approach requires intimate knowledge of group theory. I have not attempted to undertake such a study of the algebraic properties of MacGuffin.

3.8 Bit dependencies

It is possible to write each output bit of a block cipher as a boolean function of the plaintext and the key. In any good block cipher, each output bit will depend on all the inputs bits [6, 7, 22]. Let A_{ir} denote the number of plaintext bits which the i-th output bit of MacGuffin-r depends upon. In particular, MacGuffin-r passes this test if $A_{ir} = 64$ for all i. Table 7 contains a list of the A_{ir} matrix entries. By inspection, we see that 9 rounds of MacGuffin are necessary and sufficient to achieve full functional dependence for all the ciphertext bits. In comparison, 5 rounds of DES are required for complete functional dependence [6].

One can also count B_{jr}, the number of ciphertext bits which depend upon the j-th plaintext bit of MacGuffin-r. As before, for MacGuffin-r to pass this test, we demand that $B_{jr} = 64$ for all j. It is easy to see that $B_{jr} = 64$ for all j if and only if $A_{ir} = 64$ for all i, so this second bit dependency test is superfluous. Still, the B_{jr} matrix is a slightly different way of looking at the bit dependencies, so it is listed in Table 8 for completeness.

Theorem 1 *For any block cipher with r rounds, $\sum_i A_{ir} = \sum_j B_{jr}$.*

Proof: Let C_{ijr} be 1 if the i-th bit of the ciphertext depends on the j-th bit of the plaintext after r rounds, or 0 otherwise. Then

$$\sum_i A_{ir} = \sum_{i,j} C_{ijr} = \sum_j B_{jr}.$$

□

Corollary 2 *Consider any block cipher with r rounds and 64 bit blocks. Then $A_{ir} = 64$ for each i iff $B_{jr} = 64$ for each j.*

Proof: Follows immediately from the upper bounds $A_{ir} \leq 64$ and $B_{jr} \leq 64$. $\square$

Table 7: Ciphertext dependencies: A_{ir}

Bit		Round								
		1	2	3	4	5	6	7	8	9
l	0	1	1	1	7	18	42	60	64	64
l	1	1	1	1	7	18	41	60	64	64
l	2	1	1	1	7	18	40	60	64	64
l	3	1	1	1	7	18	40	60	64	64
l	4	1	1	1	7	17	36	57	63	64
l	5	1	1	1	7	16	35	57	63	64
l	6	1	1	1	7	16	32	51	63	64
l	7	1	1	1	7	16	32	51	63	64
l	8	1	1	1	7	19	40	55	64	64
l	9	1	1	1	7	18	40	55	64	64
l	10	1	1	1	7	17	35	56	64	64
l	11	1	1	1	7	17	34	56	64	64
l	12	1	1	1	7	18	35	59	64	64
l	13	1	1	1	7	19	35	59	64	64
l	14	1	1	1	7	19	44	61	64	64
l	15	1	1	1	7	19	45	61	64	64
a	0	1	1	7	18	42	60	64	64	64
a	1	1	1	7	18	41	60	64	64	64
a	2	1	1	7	18	40	60	64	64	64
a	3	1	1	7	18	40	60	64	64	64
a	4	1	1	7	17	36	57	63	64	64
a	5	1	1	7	16	35	57	63	64	64
a	6	1	1	7	16	32	51	63	64	64
a	7	1	1	7	16	32	51	63	64	64
a	8	1	1	7	19	40	55	64	64	64
a	9	1	1	7	18	40	55	64	64	64
a	10	1	1	7	17	35	56	64	64	64
a	11	1	1	7	17	34	56	64	64	64
a	12	1	1	7	18	35	59	64	64	64
a	13	1	1	7	19	35	59	64	64	64
a	14	1	1	7	19	44	61	64	64	64
a	15	1	1	7	19	45	61	64	64	64
b	0	1	7	18	42	60	64	64	64	64
b	1	1	7	18	41	60	64	64	64	64
b	2	1	7	18	40	60	64	64	64	64
b	3	1	7	18	40	60	64	64	64	64
b	4	1	7	17	36	57	63	64	64	64
b	5	1	7	16	35	57	63	64	64	64
b	6	1	7	16	32	51	63	64	64	64
b	7	1	7	16	32	51	63	64	64	64
b	8	1	7	19	40	55	64	64	64	64
b	9	1	7	18	40	55	64	64	64	64
b	10	1	7	17	35	56	64	64	64	64
b	11	1	7	17	34	56	64	64	64	64
b	12	1	7	18	35	59	64	64	64	64
b	13	1	7	19	35	59	64	64	64	64
b	14	1	7	19	44	61	64	64	64	64
b	15	1	7	19	45	61	64	64	64	64
c	0	7	18	42	60	64	64	64	64	64
c	1	7	18	41	60	64	64	64	64	64
c	2	7	18	40	60	64	64	64	64	64
c	3	7	18	40	60	64	64	64	64	64
c	4	7	17	36	57	63	64	64	64	64
c	5	7	16	35	57	63	64	64	64	64
c	6	7	16	32	51	63	64	64	64	64
c	7	7	16	32	51	63	64	64	64	64
c	8	7	19	40	55	64	64	64	64	64
c	9	7	18	40	55	64	64	64	64	64
c	10	7	17	35	56	64	64	64	64	64
c	11	7	17	34	56	64	64	64	64	64
c	12	7	18	35	59	64	64	64	64	64
c	13	7	19	35	59	64	64	64	64	64
c	14	7	19	44	61	64	64	64	64	64
c	15	7	19	45	61	64	64	64	64	64
avg		2.50	6.67	15.89	29.98	44.17	55.75	62.28	63.94	64.00

Table 8: Plaintext dependencies: B_{jr}

Bit		Round 1	2	3	4	5	6	7	8	9
l	0	1	3	9	23	38	52	62	64	64
l	1	1	3	7	17	30	44	56	62	64
l	2	1	3	9	23	38	52	62	64	64
l	3	1	3	9	23	38	52	62	64	64
l	4	1	3	9	19	34	48	58	64	64
l	5	1	3	9	21	36	50	60	64	64
l	6	1	3	9	21	36	50	60	64	64
l	7	1	3	9	19	34	48	58	64	64
l	8	1	3	9	21	36	50	60	64	64
l	9	1	3	9	21	36	50	60	64	64
l	10	1	3	9	19	34	48	58	64	64
l	11	1	3	9	19	34	48	58	64	64
l	12	1	3	9	21	36	50	60	64	64
l	13	1	3	7	15	30	44	56	64	64
l	14	1	3	9	21	36	50	60	64	64
l	15	1	3	7	17	30	44	56	62	64
a	0	3	7	17	33	47	58	64	64	64
a	1	3	7	19	35	49	60	64	64	64
a	2	3	7	17	33	47	58	64	64	64
a	3	3	7	17	31	45	56	62	64	64
a	4	3	7	17	33	47	58	64	64	64
a	5	3	6	14	30	44	56	64	64	64
a	6	3	6	16	30	44	56	62	64	64
a	7	3	6	16	30	44	56	62	64	64
a	8	3	7	17	33	47	58	64	64	64
a	9	3	6	14	28	42	54	62	64	64
a	10	3	7	19	35	49	60	64	64	64
a	11	3	7	19	35	49	60	64	64	64
a	12	3	6	14	26	40	52	60	64	64
a	13	3	7	19	35	49	60	64	64	64
a	14	3	7	15	29	43	54	62	64	64
a	15	3	7	15	29	43	54	62	64	64
b	0	3	9	21	35	49	59	62	64	64
b	1	3	9	18	34	48	58	64	64	64
b	2	3	9	20	34	48	58	62	64	64
b	3	3	9	22	38	52	62	64	64	64
b	4	3	7	17	31	45	57	62	64	64
b	5	3	7	14	28	42	54	62	64	64
b	6	3	7	14	30	44	56	64	64	64
b	7	3	9	18	34	48	58	64	64	64
b	8	3	7	16	30	44	56	62	64	64
b	9	3	9	18	34	48	58	64	64	64
b	10	3	9	21	37	51	61	64	64	64
b	11	3	9	16	32	46	56	64	64	64
b	12	3	7	16	30	44	56	62	64	64
b	13	3	9	22	38	52	62	64	64	64
b	14	3	9	20	36	50	60	64	64	64
b	15	3	9	21	37	51	61	64	64	64
c	0	3	9	23	38	52	62	64	64	64
c	1	3	7	17	30	44	56	62	64	64
c	2	3	9	23	38	52	62	64	64	64
c	3	3	9	23	38	52	62	64	64	64
c	4	3	9	19	34	48	58	64	64	64
c	5	3	9	21	36	50	60	64	64	64
c	6	3	9	21	36	50	60	64	64	64
c	7	3	9	19	34	48	58	64	64	64
c	8	3	9	21	36	50	60	64	64	64
c	9	3	9	21	36	50	60	64	64	64
c	10	3	9	19	34	48	58	64	64	64
c	11	3	9	19	34	48	58	64	64	64
c	12	3	9	21	36	50	60	64	64	64
c	13	3	7	15	30	44	56	64	64	64
c	14	3	9	21	36	50	60	64	64	64
c	15	3	7	17	30	44	56	62	64	64
avg		2.50	6.67	15.89	29.98	44.17	55.75	62.28	63.94	64.00

4 Elementary Attacks

Another way to compare the reduced-round variants of DES and MacGuffin is to examine several elementary attacks on these simplified ciphers. The full power of differential cryptanalysis will be reserved for later discussion.

4.1 Exhaustive search

Any cipher can be broken with a few known plaintext/ciphertext pairs by exhaustively trying all possible keys and checking whether the plaintexts encrypt as expected. MacGuffin has a 128 bit key, so the 2^{128} complexity of this naive approach is totally infeasible.

Complementation properties can speed up brute force key search [23]. With DES, there is a non-trivial identity

$$\overline{E_k(P)} = E_{\overline{k}}(\overline{P})$$

which holds for all plaintexts P and all keys k, where $\overline{x}$ stands for the bitwise complement of x. Suppose two known plaintext pairs P, C and $\overline{P}, C'$ are available. (Two such pairs can be found with 2 chosen plaintexts or about 2^{33} known plaintexts.) In this case, exhaustive key search can be optimized: for each k with a least significant bit of 0, compute $E_k(P)$, and check whether it is equal to C or C'. This lowers the complexity of DES exhaustive search by a factor of two: 2^{55} off-line trial encryptions suffice.

The DES complementation property arises from the linearity of the DES key scheduling algorithm. The MacGuffin key schedule is highly non-linear, which avoids this type of complementation property. In fact, MacGuffin appears to have no obvious complementation properties at all. Thus, brute force key search of MacGuffin probably requires 2^{128} trial encryptions—a tremendously large figure.

4.2 A meet-in-the-middle attack

Chaum and Evertse [24] show that DES-6 is vulnerable to a meet-in-the-middle attack requiring a few known plaintexts and 2^{54} trial encryptions. Variants with fewer rounds are even more susceptible to this attack. Their attack searches for a set S of data bits in a middle round and a set T of key bits so that any change in the T bits cannot affect the value of the S bits from either direction. From S, T one can break the cipher with about $2^{56-|T|} + 2^{|T|}$ trial encryptions and a few known plaintexts. This attack succeeds against DES mainly because of the linearity of its key schedule. In contrast, MacGuffin's complicated key schedule seems to foil the meet-in-the-middle attack.

Lemma 3 *Consider any non-empty subset T formed from bits of the 128 bit encryption key, and suppose there is a non-empty subset S of data bits in some intermediate round of MacGuffin-r which is useful in the attack described above. Then $r < 4$.*

Proof: First note that each bit of the encryption key affects each bit of every round subkey, so each bit of the output of the f function in each round depends on every bit of the encryption key. On the other hand, when $r \geq 4$ every data bit gets xor-ed with some round's f function output on its path from plaintext to ciphertext. Thus $r \geq 4$ implies $S = \emptyset$. □

Lemma 4 *Let T be a subset formed from bits of the round subkeys, and let S be a subset of data bits in some intermediate round of MacGuffin-r as above. If $r > 5$, then the meet-in-the-middle attack of Chaum and Evertse will be less efficient than exhaustive key search.*

Proof: The meet-in-the-middle attack will require about $2^{48r-|T|} + 2^{|T|}$ trial encryptions (unless $|S| < |T|$, in which case the attack might require even more trial encryptions). Now if $r \geq 6$ and $|T| < 128$ then

$$2^{48r-|T|} + 2^{|T|} > 2^{48 \cdot 6 - 128} > 2^{128},$$

proving the desired result. □

Theorem 5 *MacGuffin-r is immune from the attack of Chaum and Evertse for $r > 5$.*

Proof: Immediate from the preceding lemmas. □

4.3 A simple known-plaintext attack

Observe that knowledge of the plaintext bits entering, and the ciphertext bits leaving, a S-box allows one to derive some information about the key bits entering that S-box. A known-plaintext attack on MacGuffin-4 provides that information for each S-box in each round. On average, $1+\log_2 64/\log_2 4 = 4$ known plaintext/ciphertext pairs suffice to determine the 6 key bits entering a S-box. More precisely, with n known plaintexts the probability that this attack fails to uniquely derive the 6 key bits entering a S-box is at most $63/4^n$. Thus, with high probability $n = 7$ known plaintext/ciphertext pairs suffice to determine all round subkeys of MacGuffin-4, and this attack requires negligible amounts of computation.

This trivial attack also works nicely on MacGuffin-6. Suppose we wish to find the 6 key bits entering $S1$ in the final round of MacGuffin-6. We can guess the key bits entering $S2$, $S3$ in the first round, $S1$ in the second round, and $S1$ in the sixth round. This guess can be verified by checking that the xor of the outputs of $S1$ in the second and sixth rounds is as expected from the plaintext and ciphertext blocks. The correct guess will survive all checks, while only $1/4$ of the incorrect ones remain after checking each plaintext/ciphertext pair. Thus, on average $1 + 4 \cdot \log_2 64/\log_2 4 = 13$ known plaintexts suffice to determine the key bits entering those four S-boxes. The next step is to guess the key bits entering $S1$ in the first round and $S2$ in the second and sixth rounds; the same procedure will find the desired key bits by reusing the known plaintexts. By continuing this process, one can break MacGuffin-6 with 14 known plaintexts and approximately 2^{12} operations.

One can also carry out analogous attacks on DES. In comparison, breaking DES-2 requires about 3 known plaintexts and very little computation, while the attack on DES-3 needs 5 known plaintexts and has complexity approximately 2^9. The slight difference here between DES and MacGuffin arises mainly because more key bits enter into each MacGuffin round.

4.4 A ciphertext-only attack

Suppose the cryptanalyst only has access to ciphertexts with a guarantee that the plaintexts consist of ASCII characters with the high bit zero. Note that the DES initial permutation gathers all of the known bits in the plaintext into one byte. This special structure in the initial permutation allows the cryptanalyst to derive information about the inputs and outputs of some of the S-boxes in each round of DES. As we have seen, this means that an attacker can derive some key bits from each round subkey. The DES key schedule also plays an important role: because it repeats key bits to obtain the round subkeys, partial information about the subkeys can be used to derive much of the DES key. Finally, the remaining unknown key bits can be found by exhaustive search. In fact, this type of attack can break DES-3 with just 20 known ciphertexts and 2^{39} operations [25].

The complicated MacGuffin key schedule, along with the absence of an initial permutation, protects MacGuffin from this sort of attack. For example, even with just 1 round of MacGuffin and an endless supply of ciphertext blocks, an easy information theoretic argument shows that 36 of the 48 bits in the subkey must remain unknown. In MacGuffin-1, we know one bit of the output of $S4$ because $l_7 = 0$; by the known ciphertext assumption, all inputs to $S4$ are known, so the 6 subkey bits entering $S4$ can be computed with about 6 known ciphertexts. The same procedure can be used to calculate the 6 subkey bits entering $S8$.

On the other hand, the outputs of the other S-boxes are completely unknown, and so it's easy to see that the other 36 subkey bits cannot be computed without more information. Ciphertext-only attacks on MacGuffin variants with more than one round will also fail to recover most of the round subkey bits. Under the plausible assumption that the MacGuffin key schedule is more or less a one-way function (see Section 6), MacGuffin will be safe from a ciphertext-only attack.

4.5 A statistical known-plaintext attack

This attack (like the previous two) is described in [25] and has been adapted to MacGuffin. Let $S1_I$, $S1_I^*$ be two inputs to $S1$ with corresponding outputs $S1_O$, $S1_O^*$; also let $S1_P$, $S1_P^*$ be the plaintext bits entering $S1$ with corresponding key bits $S1_k$, $S1_k^*$. Thus we have $S1_I = S1_P \oplus S1_k$. Set $\Delta S1_I = S1_I \oplus S1_I^*$, etc. Note that $\Delta S1_I = \Delta S1_P \oplus \Delta S1_k$.

This attack on MacGuffin-5 is based on the observation that if $\Delta S1_O = 0$ then $\Delta S1_I = 0$ with probability about 1/16. Let $S1_O$ denote the output of $S1$ in the first round, let $S1_O^*$ stand for the output of $S1$ in the fifth round, and use similar notation for $S1_P, S1_P^*$, etc. Acquire a known plaintext, ciphertext pair (P, C), and note that we can compute the differences $\Delta S1_O$ and $\Delta S1_P$ from the value of $P \oplus C$. Select only encryptions where $\Delta S1_O = 0$. In 1/16 of these, we will have $\Delta S1_P = 0$, which gives us information on the key by $\Delta S1_k = \Delta S1_P$.

Count on $\Delta S1_k$: for each value of $\Delta S1_k$, count how many times it is suggested by a (P, C) pair. The correct value will occur in about 1/16 of the selected pairs, while any incorrect guess will only be suggested by a fraction $15/16 \cdot 1/63$ of them. This procedure can be repeated for each of the 8 S-boxes. For any one S-box, we need about four occurrences of the correct value of $\Delta S1_k$. With approximately $4 \cdot 16 \cdot 4 = 256$ known plaintexts, 48 bits (which are xors of round subkey bits) can be efficiently computed. Note that this attack really only requires the values of $P \oplus C$; the exact plaintext and ciphertext blocks are not necessary.

The attack above can be extended to find all the round subkeys of MacGuffin-5 with a few additional known plaintext/ciphertext pairs. Basically, we guess key values for the first round, deduce corresponding key values for the fifth round, unravel those rounds off, and then use a trivial attack on MacGuffin-3. Guess the value of the key bits $S2_k, S3_k$ entering $S2, S3$ in the first round; this determines the value of the key bits $S2_k^*, S3_k^*$ entering $S2, S3$ in the fifth round. Use about 4 known plaintexts to derive the 6 key bits entering $S1$ in the fourth round. Now check that the guessed values of $S2_k, S3_k$ and the deduced value of the key bits entering $S1$ in the fourth round are consistent with a few more known plaintexts. About 7 known plaintexts suffice to completely solve for the round subkeys of MacGuffin-5 after the first stage described above is finished. Note that this second stage requires complete knowledge of the pairs (P, C), while the first stage needs just the values $P \oplus C$.

In comparison, DES-3 falls to this attack with about 256 $P \oplus C$ xors [25]. The linearity of the DES key schedule renders a second stage unnecessary.

4.6 Another known-plaintext attack

Biham and Shamir also suggest another known-plaintext attack on DES-5. For each set of input and output bits to a S-box, consider the behavior of the xor of the output bits on all possible inputs. When the two output-bit xors are equal for a pair of inputs, count the probability that the two input-bit xors are equal. If this probability is high enough, a simple attack on DES or MacGuffin is feasible.

MacGuffin-7 can be attacked with this approach. In fact,

$$\Pr(\Delta S4_I[0,1,3,5] = 0 \mid \Delta S4_O[0,1] = 0) = .625,$$

where I use the notation that (for example) $x[0,1,3] = x_0 \oplus x_1 \oplus x_3$. Here $S4_I$ stands for the bits entering $S4$ in round 2, $S4_I^*$ is short for the bits entering $S4$ in round 6, and $\Delta S4_I = S4_I \oplus S4_I^*$ as before. Select only those known plaintexts (P, C) with $\Delta S4_O[0,1] = 0$. For the selected plaintexts, $\Delta S4_I[0,1,3,5] = 0$ with high probability. The values $S4_I[0,1,3,5]$ and $S4_I^*[0,1,3,5]$ in rounds 2 and 6 are known with high probability, up to xor with some subkey bits in rounds 2 and 6 (namely, $\Delta S4_k[0,1,3,5]$) and the output of some S-boxes in rounds 1 and 7 (namely, $S6$ from round 1 and $S7, S8$ from round 7). The inputs to $S6$ in round 1 and $S7, S8$ are known by the known plaintext assumption.

Thus, we count on the subkey bits entering $S6$ in round 1, $S7$ and $S8$ in round 7, and the xor $S4_k[0,1,3,5]$ of subkey bits in rounds 2 and 6. The right value for those 19 subkey bits will be suggested by about 62.5% of the selected known plaintexts, while incorrect values will be suggested by just 50% of them. About 250 known plaintexts should suffice to discover the right value of those 19 subkey bits with a very high success rate.

Of course, the attack may be continued to the other S-boxes. Every S-box has some correlation with probability at least .570, so the round 1 and round 7 subkey values should be computable with about 1000 known plaintexts. After this, the first and last rounds of MacGuffin-7 can be unraveled and the remaining 5 round cipher can be attacked with simpler methods discussed earlier.

In comparison, DES-5 falls to a similar statistical attack with at most 370 known plaintexts [25]. The attack on DES-5 can be extended to use just knowledge of $P \oplus C$—the exact values of (P, C) are not needed. Unfortunately, the MacGuffin-7 attack seems to require both plaintext and ciphertext blocks.

4.7 A chosen plaintext attack

The measurements of Section 3.2 show toggling the plaintext bit l_1 toggles the ciphertext bit c_1 with probability $.5 - .163$ in MacGuffin-9. We can use this fact to attack MacGuffin-13. Using the chosen plaintext assumption, we can obtain the encryption under MacGuffin-13 of pairs of plaintexts which differ only in bit l_1. Knowledge of the ciphertexts gives us knowledge of the plaintext bits entering $S1$ in the last round, and we know bit 1 of the output of $S1$ with probability $.5 + .163$ for any pair. For any guess at the 6 key bits entering $S1$ in the last round, we can check whether the output is as expected. Count on those key bits for 150 pairs. The correct key will be suggested by $(.5+.163)\cdot 150 \approx 100$ of those pairs. On the other hand, any one incorrect key value will be suggested by $.5 \cdot 150 = 75$ of those pairs on average. Moreover the probability that an incorrect key value will be suggested by more than 100 of the pairs is about 1/45000. Thus, with probability about .999, 300 chosen plaintexts suffice to find the 6 key bits entering $S1$ in the last round of MacGuffin-13.

We may continue this attack to find more key values. Suppose toggling a plaintext bit leaves a ciphertext bit in the c register unchanged after r rounds with probability $.5+\epsilon$. Then approximately $4/\epsilon^2$ chosen plaintext pairs suffice to find 6 key bits entering a S-box in the last round of MacGuffin-$r + 4$. This approach will succeed about 99% of the time.

This idea can also be improved by noting that toggling plaintext bit l_9 leaves ciphertext bits $c_{6,7}$ both unchanged after 9 rounds of MacGuffin encryption with probability $.25 + .354$. By counting on the 6 key bits entering $S4$ in the last round of MacGuffin-13, we can discover those key bits more efficiently than before. In general, suppose toggling one plaintext bit leaves two ciphertext bits in the c register unchanged after r rounds with probability $.25 + \epsilon$ (and we insist that those two ciphertext bits correspond to the two output bits of some S-box in round $r + 4$). Then about $3/\epsilon^2$ chosen plaintext pairs will enable the cryptanalyst to find 6 key bits entering a S-box in the last round of MacGuffin-$r + 4$ with success rate 99% or so.

After all the key bits in the last round of MacGuffin-$r + 4$ are found with this technique, one can peel the last round off all the ciphertexts and proceed to attack MacGuffin-$r+3$ by the same method. After several iterations, the problem will have been reduced to breaking MacGuffin-6; then the easy known-plaintext attack mentioned earlier can find the remaining key bits. The same approach can be used to break small-round variants of DES, too. See Table 9 for a summary of the complexity of the best such attacks.

Table 9: Chosen plaintexts needed for an avalanche-based attack

r	MacGuffin-r	DES-$\frac{r}{2}$
12	80	16
13	500	n/a
14	6800	1600
15	36000	n/a
16	83000	∞
17	330000	n/a

4.8 A correlation attack

Biham and Shamir briefly outline an unpublished statistical attack on DES-8 discovered by D. W. Davies [25]. This attack exploits a correlation between the outputs of adjacent S-boxes in the DES

f function. The correlation arises from the duplication of input bits between neighboring S-boxes by the DES E box. This attack on DES-8 derives 16 key bits with 2^{40} known plaintexts.

It is natural to wonder whether MacGuffin is vulnerable to this correlation attack. Note, though, that MacGuffin has no E box equivalent. Its round function never duplicates any input bits, and no two S-boxes share any common input bits. Thus from Biham and Shamir's brief description, MacGuffin appears to be safe from this exact attack. On the other hand, the lack of an E box is worrying, for DES is much weaker without an expansion permutation—DES sans E box falls to a differential attack requiring just 2^{28} chosen plaintexts [25].

One can formulate a variation on Davies' original idea to attack unbalanced Feistel ciphers like MacGuffin. In any one round, no two S-boxes share any common input bits—but there are many cases of S-box pairs in nearby rounds with duplicated input bits. For instance, bits 2 and 5 entering $S1$ in any round are the same as bits 1 and 3 entering $S3$ in the following round. Fix the xor of subkey bits $S1_k[2] \oplus S3^*_k[1]$ and $S1_k[5] \oplus S3^*_k[3]$ and suppose that the inputs to those S-boxes are otherwise randomly chosen. Then the joint distribution (conditioned on the fixed xors of subkey bits) of the two S-box outputs will be non-uniform. Furthermore, the conditional distribution will be recognizably different for each value of those two xors of subkey bits.

Here is a known-plaintext attack on MacGuffin-8 based on this observation. Let $S1_O$ denote the output of $S1$ in round r and $S3^*_O$ stand for the output of $S3$ in the next round. Put $X_r = (S1_O, S3^*_O)$ so that X_r is a random variable with a non-uniform distribution on $\mathbb{Z}_2^4$ conditioned on the value of some xors of subkey bits. Guess the round subkey bits entering $S3$ in round 1 and the subkey bits entering $S1$ in round 8. For each plaintext, ciphertext pair P, C we may calculate the value of X_4. Count on those 12 subkey bits in rounds 1 and 8 and record the distribution of X_4 for each. Incorrect guesses of the subkey bits are likely to lead to white noise when X_4 is calculated, so we can recognize the correct guess by the non-uniform distribution recorded for it. More post-processing will derive the xors of subkey bits $S1_k[2] \oplus S3^*_k[1]$ and $S1_k[5] \oplus S3^*_k[3]$, since the distribution will be different for each possible such xors.

There are several ways to implement the counting scheme. The simplest method uses an array of 2^{12+4} counters. In this scheme, array element number $2^4 g + x$ counts the frequency of $X_4 = x$ for each guess g at the 12 subkey bits in rounds 1 and 8. This method requires about $2^{12}n + 2^{12+4}$ computations to process to n known plaintexts.

If n is very large, a faster two-stage algorithm is available. In the pre-processing stage, set up an array of 2^{12+4} elements. Here counter number $2^{10}p + 2^4 c + o$ is incremented when the plaintext input to $S3$ in round 1 is p, the input to $S1$ in round 8 (from the ciphertext) is c, and o stands for the 4 bits from $P \oplus C$ which are xored with the outputs of $S1$ in round 4 and $S3$ in round 5. In the second stage, we proceed as above, filling in the values of the $g2^4 + x$ counters and finding subkey bits. The two-stage algorithm takes about $n + 2^{12+12+4} + 2^{12+4}$ steps to analyze n known plaintexts. Thus the two-pass method is preferred for $n > 2^{16}$.

To take advantage of this attack, we must be able to differentiate between non-uniform and uniform distributions efficiently. One can use the chi-squared test to compare each observed distribution to the ideal uniform distribution. Alternatively, we may use Bayes' Theorem (since the prior probabilities are known). In either case, the number of known plaintexts needed will be a function of the deviation of the conditional distribution of X_4 from uniform. Suppose that

$$\Pr(X_4 = i | \text{the xors of subkey bits for shared inputs}) = \frac{1}{16} + \epsilon_i,$$

and put $\delta = \sum_i \epsilon_i^2$. Then we will need about $n = O(\delta^{-1})$ known plaintexts.

This attack can be generalized. There are many sets of S-boxes which give rise to correlations in their outputs. In general, we can find larger biases by considering the joint distribution of larger sets of S-boxes. This has advantages and disadvantages: more memory and more processing is required, but δ^{-1} above can often be decreased, and the constant factor in $n = O(\delta^{-1})$ decreases as more S-boxes are used. Moreover, we can take advantage of any correlation four times, once for each of the l, a, b, c registers.

This correlation attack would not be very noteworthy if it were only applicable to MacGuffin-8. Fortunately, we can attack MacGuffin-$4j$ if we can find significant biases in the distribution of $X_4 \oplus X_8 \oplus \ldots \oplus X_{4j-4}$. This often is indeed non-uniform, though the magnitude of the bias decreases exponentially in j. After sufficiently many rounds, the correlation will die out, defeating the correlation attack.

I have mounted a search for sets of S-boxes with shared input bits which give rise to detectable correlations. Table 10 records the "best" attacks found (though more tradeoffs between time, space, discovered subkey bits, and known plaintexts are probably possible).

Recall that DES-8 fell to a correlation attack requiring 2^{40} known plaintexts and a similar amount of computation. We can see that MacGuffin-16 is significantly more vulnerable to correlations between S-boxes than DES-8. At the same time, the full 32-round MacGuffin does appear to be safe from these correlation attacks.

The figures in Table 10 are based solely on theory, small simulations, and projections. Therefore they should be considered approximate predictions, not tried-and-true observations. Testing the predictions would require some experimental cryptanalysis, and would be an interesting area for further study.

Table 10: A correlation attack on MacGuffin

Rounds	Known plaintexts	Time	Memory	S-boxes	Subkey bits
8	2^9	2^{21}	2^{16}	$S1; S7$	48+8
	2^6	2^{32}	2^{32}	$S1, S6; S4, S8$	96+24
12	2^{20}	2^{28}	2^{17}	$S1; S7$	48+16
	2^{17}	2^{41}	2^{32}	$S1, S6; S4, S8$	96+48
16	2^{32}	2^{32}	2^{17}	$S1; S7$	48+24
	2^{28}	2^{56}	2^{33}	$S1, S6; S4, S8$	96+72
20	2^{43}	2^{43}	2^{17}	$S1; S7$	48+32
	2^{38}	2^{56}	2^{33}	$S1, S6; S4, S8$	96+96
24	2^{54}	2^{54}	2^{17}	$S1; S7$	48+40
	2^{50}	2^{56}	2^{33}	$S1, S6; S4, S8$	96+120

5 Differential Cryptanalysis

5.1 Introduction

As before, let $S1_I$, $S1_I^*$ be two inputs to $S1$ with corresponding outputs $S1_O$, $S1_O^*$; also let $S1_P$, $S1_P^*$ be the plaintext bits entering $S1$ with corresponding key bits $S1_k$. Note that $S1_I = S1_P \oplus S1_k$. Set $\Delta S1_I = S1_I \oplus S1_I^*$, etc. In this discussion, we will always have $S1_k = S1_k^*$. All S-box and f function input and output values will be expressed in hexadecimal notation.

If there exist $S1_I, S1_I^*$ with $\Delta S1_I = X$ and $\Delta S1_O = Y$, we say that $X \to Y$ by $S1$. Furthermore, we say that $X \to Y$ by $S1$ with probability

$$\Pr(X \to Y \text{ by } S1) = \frac{|\{(S1_P, S1_P^*, S1_k) : \Delta S1_P = X, \Delta S1_O = Y\}|}{|\{(S1_P, S1_P^*, S1_k) : \Delta S1_P = X\}|}.$$

We can also make similar statements about the round function: $X \to Y$ by f with probability

$$\Pr(X \to Y \text{ by } f) = \Pr(f(x) \oplus f(x^*) = Y | x \oplus x^* = X)$$

where the probability is taken over all possible input pairs (x, x^*) and over all possible subkey values.

Lemma 6 *Suppose $X \to Y$ by the MacGuffin f function with probability p, and let (x, x^*) be any fixed input pair with $x \oplus x^* = X$. Then every such (x, x^*) causes the f function output xor to be Y by the same fraction p of the subkey values.*

Proof: Biham and Shamir [25] prove this result for DES in Lemma 3.1. A simple modification of their proof shows that the same result holds for MacGuffin: the idea is that $f(x)$ depends only upon $x \oplus k$ and has no direct x dependence (if we let k denote the round subkey). □

Corollary 7 *The probability of $X \to Y$ by the MacGuffin f function is just the product of the probabilities of the xors at each S-box.*

Each such X, Y for the MacGuffin f function gives us a one round characteristic. Suppose that $X^{(0)} \to X^{(1)}$ and $X^{(1)} \to X^{(2)}$ by the f function; then we can concatenate these two one round characteristics to obtain a two round characteristic with plaintext xor $X^{(0)}$ and ciphertext xor $X^{(2)}$. This process can be continued to obtain a multiple round characteristic. Characteristics which can be concatenated with themselves are called iterative characteristics.

Biham and Shamir describe how to break a r round Feistel cipher given a $r-1$ round characteristic with a large enough probability. Since finding all $r-1$ round characteristics is impractical for even fairly small values of r, instead we search for a relatively short iterative characteristic which can be repeated to obtain a long characteristic. For more information on differential cryptanalysis see e.g. [25, 26, 27, 28, 29, 30].

5.2 MacGuffin characteristics

I wrote several small programs to search for short iterative characteristics using mathematically plausible heuristics. The best characteristics found are listed in Table 11. These characteristics may not be optimal, but they suffice to provide an upper bound on the security of MacGuffin.

Most importantly, MacGuffin admits a 4 round iterative characteristic with probability 1/148.9. This characteristic is shown in full detail in Figure 3. In comparison, the best DES iterative characteristic has 2 rounds and probability 1/234 [25]. Thus, differential cryptanalysis penetrates MacGuffin-$2r$ more successfully than DES-r.

Table 11: Iterative characteristics for MacGuffin

Rounds	Probability	ΔP
1	1/53.2	$(0A00, 0A00, 0A00, 0A00)$
2	1/148.9	$(2000, 0041, 2000, 0041)$
3	(none found)	
4	1/148.9	$(2000, 0001, 0000, 0040)$
5	(none found)	
6	(none found)	
7	(none found)	
8	1/49187.7	$(0002, 0400, 0000, 0000)$

5.3 The experiments

MacGuffin with a reduced number of rounds can be broken efficiently by a $4R$ attack with repetitions of the iterative characteristic covering the earlier rounds. To test the behavior of this attack in practice, I implemented it for MacGuffin-8, MacGuffin-12, and MacGuffin-16. The resulting observations should help estimate the complexity of differential cryptanalysis of the full 32 round version.

In general, 10–20 right pairs were needed to have a significant success rate. This rather large figure is explained by the need to repeat the attack for every round subkey. With fewer right pairs, the attack failed to find a unique value for all the round subkeys, but it often narrowed them down to a handful of possibilities. The observed graceful degradation is reassuring (to the attacker!).

Because nearly all wrong pairs are discarded, the clique method is preferred here. Also, the use of structures greatly reduced the number of chosen plaintexts required. These two enhancements were the most sophisticated features of the attack that was actually carried out.

My experimental implementation of the differential attack on reduced-round versions of MacGuffin is undoubtedly sub-optimal. For example, bypassing the first couple of rounds would improve the attack. After the first pass, we can easily derive partial information about the next-to-last round subkey: taking advantage of this knowledge would probably increase the efficiency of the subsequent passes greatly. A search for better non-iterative characteristics for the reduced-round variants would probably improve efficiency even more dramatically. The point, though, was to test the essential features of a differential attack, and this goal was achieved successfully without any fancy optimizations.

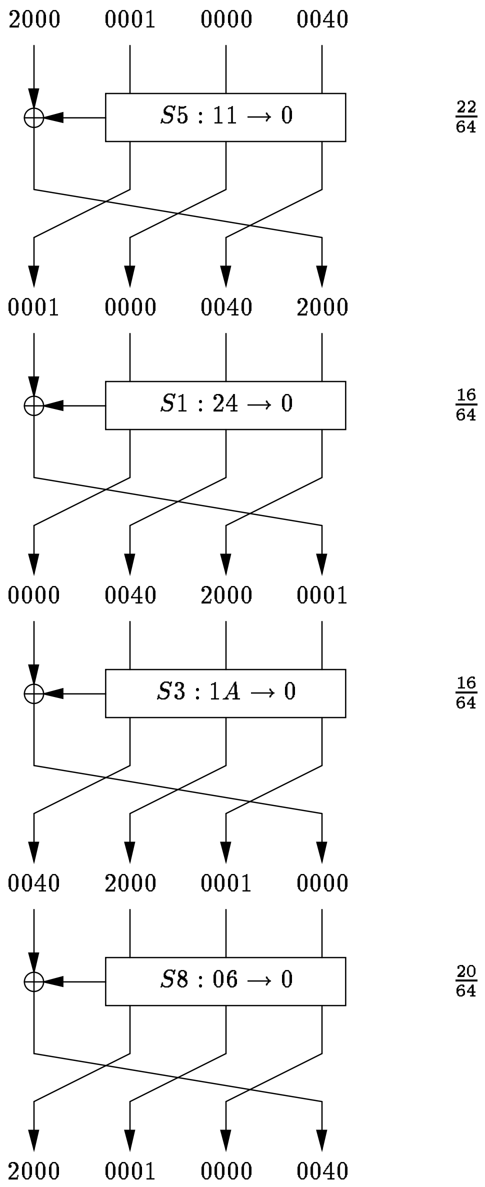


Figure 3: An iterative characteristic for MacGuffin

5.4 Attacks on all 32 rounds

The simplest differential attack on MacGuffin-32 is a $2R$ attack which uses the 4 round iterative characteristic to cover the first 30 rounds. We get a 30 round characteristic based on Figure 3 with probability

$$\frac{20}{64} \cdot \frac{1}{148.9^7} \cdot \frac{22}{64} \approx 2^{-53.6}.$$

Each right pair is recognized by the fact that the ciphertext xor must have zeros in 58 fixed bit positions. By using 2^{58} chosen plaintext pairs, we expect to find 20 right pairs and one wrong pair. Because of the excellent recognition of right pairs, we can use the clique method and avoid the memory requirements of the normal counting method. Experiments with MacGuffin-16 indicate that we recover at least 12 bits of the last round subkey with high probability.

Repeating the attack with rotated versions of the 30 round characteristic will fill in the remaining subkey bits from the last round. These other characteristics will have approximately the same probability; by using the partial knowledge of the last round subkey, we can achieve nearly perfect recognition of right pairs. Then the last round subkey can be peeled off and the attack can be repeated on MacGuffin-31, continuing until all round subkeys are known. By using structures and three extra rotated versions of the 30 round characteristic, we get an algorithm for breaking MacGuffin with about 2^{59} chosen plaintexts. Unfortunately, all 2^{59} ciphertexts must be stored for use in each pass, so the low memory requirement of the clique method is no longer an advantage.

The attack can be optimized if sufficient memory is available. Bypass the first three rounds with the techniques of Biham and Shamir [28], use a $3R$ attack, and cover the middle 26 rounds with repetitions of the 4 round iterative characteristic. The characteristic has probability

$$\frac{20}{64} \cdot \frac{1}{148.9^6} \cdot \frac{22}{64} \approx 2^{-46.5}.$$

Count on 54 round subkey bits. Since a right pair must have zeros in 50 bits of the ciphertext xor, we expect $S/N \approx 12$. (Note that the S/N ratio degrades to an unusable value if we try to use a $4R$ attack.) The attacker will need approximately 10–15 right pairs for this first stage, so $2^{50.5}$ chosen pairs should suffice to find much of the last round subkey.

The cryptanalyst may again repeat the attack with rotated versions of the iterative characteristic to find the rest of the final round subkey. As before, the partial information obtained above can be used to greatly improve the S/N ratio for this stage. Finally the last round subkey may be peeled off and the attack repeated until all round subkeys are known. By combining the four characteristics into one structure, the attack will need $2^{51.5}$ chosen plaintexts. Since the attack uses 2^{54} counters, and the ciphertext blocks must be saved in for use in each pass, the cryptanalyst will need approximately $2^{54} + 8 \cdot 2^{51.5} \approx 2^{55}$ bytes of storage.

Biham and Shamir have noted that any chosen plaintext differential attack can be carried out under the weaker known plaintext assumption, at some loss of efficiency [25]. Suppose the attack needs m chosen plaintexts, but instead $2^{32}\sqrt{\pi m/2}$ known plaintexts are given. Then with high probability the required plaintext xors can be found among the available data and the differential attack can be carried out.

In comparison, the best differential attack on DES with 16 rounds of independent subkeys requires approximately 2^{60} chosen plaintexts and very little memory [25]. Vanilla DES with dependent round subkeys is a bit weaker: there is a differential attack requiring 2^{47} chosen plaintexts and very little memory [28]. However, this special attack succeeds only by taking advantage of the linear DES key schedule. MacGuffin has a much more complicated key schedule, so the most valid comparison is with the independent subkey variant of DES. In short, MacGuffin is weaker than DES against a differential attack.

5.5 An attack based on a key schedule weakness

A novel version of the previous attacks allows an adversary to find the original 128 bit MacGuffin key without repeated passes over the collected data. Reverse a standard differential attack to obtain the first round subkey bits by using some chosen ciphertext pairs. After the first round subkey is known, a search method described in Section 6 can derive the full 128 bit key with 2^{80} off-line trial encryptions. These computations can be carried out in parallel efficiently. As there is no need to repeat the differential attack for each round, fewer pairs will be needed on average than when recovering all round subkeys.

This attack is most practical when off-line trial encryptions are very cheap. When this assumption holds, it is best to use the $2R$ algorithm described earlier. Since the $2R$ attack uses the clique method, very little storage is required, and all ciphertexts can be processed in real time and discarded. Then both stages can be parallelized efficiently and the memory consumption will be very low.

If 2^{22} related-key chosen plaintexts are also available, then the full 128 bit key can be recovered with 2^{58} off-line trial encryptions (see Section 6). In this case the total complexity of the attack will be reduced to approximately 2^{59} operations with negligible storage.

5.6 Enhanced characteristics

If partial information about the first round subkey is available, we can often improve the probability of the first round of a characteristic [25]. Table 12 lists the possible input values of $S8$ for $06 \to 0$. We can see that $\frac{7}{10}$ of the inputs have bit 3 set. Suppose we know that the corresponding bit of the first round subkey (bit 20) is α. Then we may choose only plaintexts with $b_4 = \alpha \oplus 1$ so that the input to $S8$ in the first round always has bit 3 set. The probability and S/N ratio of the iterative characteristic are then 1.4 times as good.

The $2R$ attack described earlier can be improved with this technique. We need only $\frac{5}{7} \cdot 2^{58}$ chosen plaintexts with this modification if bit 20 of the first round subkey is known. Similar improvements can be had with the other rotated versions of the iterative characteristic if other bits of the first round subkey are known. On the other hand, this method cannot be used to improve differential attacks which bypass the first three or more rounds.

Because the DES E box duplicates input bits among neighboring S-boxes, the best iterative characteristic for DES propagates with a probability which is dependent on each round subkey [25]. Though MacGuffin has no E box, the 4 round iterative characteristic for MacGuffin also has a slight key dependence. This happens because the active S-boxes in nearby rounds share several input bits.

For example, consider the inputs to $S8$ in one round and to $S5$ in the next round. We can see that $S8_P[4] = S5^*_P[2]$. Furthermore, $\frac{12}{20}$ of the the possible inputs for $S8 : 06 \to 0$ have bit 4 of the input cleared (see Table 12). At the same time, $\frac{12}{22}$ of the inputs satisfying $S5 : 11 \to 0$ have bit

2 set. If $S8_k[4] \oplus S5^*_k[2] = 1$, the probability and S/N ratio of the iterative characteristic will be improved by a factor of

$$\frac{12 \cdot 12 + 8 \cdot 10}{2 \cdot 32^2} / \frac{20 \cdot 22}{64^2} \approx 1.02.$$

If the xor of the subkey bits is zero, the probability and S/N ratios will be .98 times the average values.

Table 12: Possible inputs for $S8 : 06 \rightarrow 0$

x						$x \oplus 06$						$S8(x)$	
0	1	2	3	4	5	0	1	2	3	4	5	0	1
1	0	1	0	0	0	1	1	0	0	0	0	1	1
0	0	1	1	0	0	0	1	0	1	0	0	1	1
1	0	1	1	0	0	1	1	0	1	0	0	1	0
0	1	1	0	1	0	0	0	0	0	1	0	0	1
0	1	1	1	1	0	0	0	0	1	1	0	1	0
1	1	1	1	1	0	1	0	0	1	1	0	0	0
0	1	1	0	0	1	0	0	0	0	0	1	1	0
0	0	1	1	0	1	0	1	0	1	0	1	0	1
1	0	1	1	0	1	1	1	0	1	0	1	0	1
0	0	1	1	1	1	0	1	0	1	1	1	1	0

Another key dependence is derived from the observation that $S5_P[0,4] = S3^*_P[1,4]$. Here $S5_P$ is the input to $S5$ in some round and $S3^*_P$ denotes the input to $S3$ two rounds later. If $S5_k[0,4] \oplus S3^*_k[1,4] = 1$, the probability and S/N ratio of the iterative characteristic will be improved by a factor of

$$\frac{12 \cdot 12 + 10 \cdot 4}{2 \cdot 32^2} / \frac{22 \cdot 16}{64^2} \approx 1.05.$$

All combinations of input bits were exhaustively searched to find the strongest key dependencies of the iterative characteristic. I found that there are no other simple key dependencies based on the xor of S-box input bits being duplicated from round to round.

There are some more complex key dependencies, but their effect is in general less pronounced. For example, note that

$$S3_P[2] \oplus S3_k[2] = S5^*_O[0] \oplus S3^\dagger_P[2] \oplus S3^\dagger_k[2],$$

where $S3_P$ stands for the input to $S3$ in some round r, $S5^*_O$ denotes the output from $S5$ in round $r-2$, and $S3^\dagger_P$ represents the input to $S3$ at round $r-4$. We have $S5^*_O[0] = 0$ with probability $\frac{6}{11}$ and $S3 : 1A \rightarrow 0$ with probability $\frac{5}{8} \cdot \frac{16}{64}$ when $S3_P[2] = 0$. Thus if $S3_k[2] \oplus S3^\dagger_k[2] = 0$ the characteristic's probability will be improved by a factor of about

$$(\frac{1}{2}p + \frac{1}{2}q) / \frac{16 \cdot 22 \cdot 16}{64^3} \approx 1.05$$

where p and q are defined as

$$\begin{aligned} p &= \frac{10 \cdot 12 \cdot 10}{32 \cdot 64 \cdot 32} + \frac{10 \cdot 10 \cdot 6}{32 \cdot 64 \cdot 32} \\ q &= \frac{6 \cdot 12 \cdot 6}{32 \cdot 64 \cdot 32} + \frac{6 \cdot 10 \cdot 12}{32 \cdot 64 \cdot 32}. \end{aligned}$$

The other complex key dependencies have a very small effect.

This dependence on the round subkeys is very weak. In the best possible situation, every four rounds of the iterative characteristic could be improved by a factor of about $1.02 \cdot 1.05 \cdot 1.05$. This leads to a total improvement in the probability and the S/N ratio by at most a factor of two even if all the round subkeys are favorable. Since this event occurs with probability 2^{-18}, the key dependence is essentially negligible. The possible enhancement is so poor that we may essentially ignore all key dependence in differential attacks which bypass the first few rounds.

5.7 Other design choices

Blaze and Schneier do not report taking any special precautions during the design process to protect against differential cryptanalysis. It is interesting to perturb the original design and compare design choices.

One possibility is to investigate different choices of the input permutation used to direct f inputs to the inputs of the S-boxes. Blaze and Schneier follow these design criteria:

P1 There is no expansion permutation or duplication of inputs.

P2 The six inputs to each S-box come from different bits in each of the a, b, c registers.

P3 For each S-box there is a "mate" box with which it shares no inputs, so that the a, b, c registers may be masked, or-ed together, and used to index a 2^{16} element lookup table containing the output from both mates (without needing shifts).

P4 Each S-box receives two of its six bits from each of the a, b, c registers.

P5 No S-box output goes to a bit position that is used as a direct input to itself in the next four rounds.

P6 *(not specified explicitly)* No S-box in any round sends both its output bits to any S-box in any other round.

Criterion P1 was chosen so that the S-boxes of DES could be used directly; it has dubious cryptographic merit (for example, Biham and Shamir found that removal of the E box weakens DES by about 9 orders of magnitude [25]). Principles P2–P3 were chosen for performance reasons. Criterion P4 is a logical new design rule for GUFNs which ensures that each output bit affects S-boxes in the next three rounds. Criterion P5 is adopted directly from DES (e.g. R2 of [31]). Blaze and Schneier do not explicitly state rule P6 in [5], but their choice of input permutation is in agreement with it. Its origin can clearly be seen in DES: see R1 of [31] or P-2 of [2]. Notice that

Blaze and Schneier make no distinction between the "end" and "middle" bits of the input to an S-box, in sharp contrast to DES [31, 2].

Another obvious design element to modify is the choice of S-boxes. Since the MacGuffin S-boxes were based on the DES ones, and much documentation is available on design rules for creating DES-like S-boxes, it is convenient to generate new DES-like S-boxes and truncate them. I followed S-1, S-2, S-3, S-4, and S-7 from Coppersmith's account of the DES design process [2]; the other criteria were relevant only to the carefully balanced machinery in DES and appear to be of limited value to MacGuffin. Probably with more care one could generate S-boxes specifically tailored towards use in MacGuffin; on the other hand, possibly the use of such small S-boxes is fatally flawed.

Many new input permutations and new S-boxes were generated according to the previously discussed criteria; then they were tested for resistance to differential cryptanalysis when used to perturb the original MacGuffin specification. No attempt was made to evaluate the effectiveness of non-differential attacks on the new variations, so there is no guarantee that they'd actually improve MacGuffin; but haphazardly replacing some component with a new one that looks better is in general poor practice. The results of these experiments are reported in Table 13, with p denoting the best 4 round iterative characteristic found for each choice of input permutation. (No useful iterative characteristics with more than 4 rounds were found.) See also Section 5.9 for further information on the choice of S-boxes. The apparent mediocrity of MacGuffin's design elements is disturbing.

Table 13: Other design choices for MacGuffin

Category	P1–P5	P1–P6	S-boxes
Best p	1/655.4	1/546.1	1/336.1
Median p	1/256.0	1/240.1	1/132.4
Worst p	1/60.7	1/93.1	1/47.0
% with $p < 1/148.9$	75%	90%	50%
% with $p < 1/243$	58%	55%	33%

5.8 Markov ciphers

Lai, Massey, and Murphy [30] propose to treat the propagation of differential characteristics as a Markov chain. The concept of a Markov cipher is defined in their paper, but essentially it is a cipher in which differential characteristics propagate with the memoryless property found in Markov chains. See [30] for the precise definitions. The following results are immediate (though they are mainly of theoretical interest).

Lemma 8 *MacGuffin is a Markov cipher under the group* $(\mathbb{Z}_2^{64}, \oplus)$.

Proof: This is just Lemma 6 in disguise. □

Theorem 9 *Under the hypothesis of stochastic equivalence and the assumption that the MacGuffin round subkeys are uniformly and independently distributed, MacGuffin is secure against differential cryptanalysis after sufficiently many rounds.*

Proof: See Theorem 3 of [30]. □

More practically, a design principle was suggested:

> The transition probability matrix of a Markov cipher should not be symmetric.

The transition probability matrix Π is defined according to

$$\Pi_{ij} = \Pr(\Delta C = a_j | \Delta P = a_i)$$

where $a_0, a_1, \ldots$ is an enumeration of all non-trivial block differences; also, $\Delta P, \Delta C$ stand for the plaintext difference and the resulting ciphertext difference after one round. Of course any symmetry of Π leads immediately to a two-round iterative characteristic, so high probability symmetries must be avoided. It is possible to show that MacGuffin passes the design criterion.

Lemma 10 *If the symmetry $\Pi_{ij} = \Pi_{ji}$ holds for $i \neq j$, then $\Pi_{ij} = \Pi_{ji} < 0.01$.*

Proof: Put

$$a_i = (\Delta l, \Delta a, \Delta b, \Delta c)$$

so that

$$a_j = (\Delta a, \Delta b, \Delta c, \Delta l \oplus \Delta y).$$

Since $a_j \to a_i$, we must have $\Delta b = \Delta l$, $\Delta c = \Delta a$, and furthermore $\Delta y = 0$. Finally, the condition $i \neq j$ implies $\Delta l \neq \Delta a$. Thus for the symmetry to occur we require

$$\Pr(\Delta l, \Delta a, \Delta l \to 0000) = \Pr(\Delta a, \Delta l, \Delta a \to 0000).$$

An exhaustive search of all such $(\Delta l, \Delta a)$ with at most 6 S-boxes active in one of the input xors finds

$$\Pr(1400, 0010, 1400 \to 0000) = \Pr(0010, 1400, 0010 \to 0000) = \frac{1}{105.0} < 0.01$$

as the best such symmetry. At the same time, any other symmetry in the transition matrix must have at least 7 S-boxes active in an input xors to the f function. Now note that $X \to 0$ by f with probability at most

$$\frac{30 \cdot 36 \cdot 28 \cdot 32^4}{64^7} < 0.01$$

if X has at least 7 active S-boxes. Thus, any symmetry in the transition matrix must have probability less than 0.01. □

Theorem 11 *MacGuffin-11 is secure against a differential cryptanalytic attack which uses only symmetries of the transition matrix.*

Proof: Observe that $0.01^{10} < 2^{-64}$. □

5.9 Related research

Shortly before this work was completed, I was alerted that a related paper by Rijmen and Preneel on the security of MacGuffin would appear soon; through the kindness of Vincent Rijmen, I was able to examine a copy before publication. A comparison of our results showed that the first version of their paper contained an inadvertent misreading of the MacGuffin specification [32]. After this initial false start, Rijmen and Preneel independently found the MacGuffin iterative characteristic depicted in Figure 3 and showed that MacGuffin is no stronger than DES [33]. They further demonstrated that MacGuffin is also susceptible to linear cryptanalysis (but MacGuffin is apparently more resistant to linear cryptanalysis than DES). Their work is especially impressive, for they cryptanalyzed MacGuffin at the same workshop where it was first formally presented.

Recall that the MacGuffin f function is formed from S-boxes based on the DES S-boxes: the middle two bits of the DES S-boxes are discarded. Rijmen and Preneel examined other subsets of output bits from the DES S-boxes, and found that the current choice is sub-optimal. This is a troublesome reflection on the comprehensiveness of the MacGuffin design process.

6 The MacGuffin Key Schedule

One major criticism of DES is that it has a linear key schedule. Many newer block ciphers use a more sophisticated key scheduling algorithm with the goal of achieving a certain amount of one-wayness. MacGuffin follows this approach; the key schedule (paraphrased from [5]) is shown in Figure 4.

KeySched(K)
1. for $r \leftarrow 0$ to 31
2.   for $i \leftarrow 0$ to 3
3. $subkey[r][i] \leftarrow 0$
4. UpdateSubkey($subkey$, $K_{0\ldots63}$)
5. UpdateSubkey($subkey$, $K_{64\ldots127}$)

UpdateSubkey($subkey$, $block$)
1. $(l, a, b, c) \leftarrow block$
2. for $r \leftarrow 0$ to 31
3. $(l, a, b, c) \leftarrow$ MacGuffinEncrypt($subkey$, (l, a, b, c))
4. $subkey[r][0] \leftarrow subkey[r][0] \oplus l$
5. $subkey[r][1] \leftarrow subkey[r][1] \oplus a$
6. $subkey[r][2] \leftarrow subkey[r][2] \oplus b$

Figure 4: The MacGuffin key scheduling algorithm

6.1 Weak keys

For a variety of reasons, DES has several keys which are especially dangerous. First, these weak keys generate only a few different subkey values. This makes them vulnerable, or more vulnerable than usual, to certain attacks based on related keys [11, 34]. It also raises doubts about the security of those keys on general grounds.

Secondly, the encryption and decryption operations are identical for some of the weak keys. With DES, decryption is just encryption with the order of the round subkeys reversed; therefore encryption is self-reversing for those keys which cause all round subkeys to be equal. The other weak keys pair up as (k, k') where

$$E_k(P) = E_{k'}(P) \quad \forall P.$$

This is clearly a problem.

Finally, the weak keys lead to unusually short cycles [17, 35]. Let E_0 denote encryption by the all-zero DES key (which causes all round subkeys to be zero as well), and let E_1 similarly stand for DES encryption by the all-ones key. These two weak keys satisfy relations such as

$$E_0(E_0(P)) = E_1(E_1(P)) = E_0(\overline{E_1(\overline{P})}) = P \quad \forall P.$$

There are also about 2^{32} fixed points of E_0 or E_1, which is far more than would be expected with a random function. (A fixed point of E_0 is a plaintext P for which $E_0(P) = P$.) These two facts, as it happens, imply that the sequence

$$P, (E_0 \circ E_1)(P), (E_0 \circ E_1)^2(P), (E_0 \circ E_1)^3(P), \ldots$$

has many cycles of length at most 2^{32} [17]. These have exceptionally short cycle lengths—one would expect a cycle length of 2^{63} on average if $E_0 \circ E_1$ behaved like a random function.

The DES weak keys arise from a linear key scheduling algorithm. Other Feistel ciphers have also suffered from the problem of weak keys, so it is natural to wonder whether MacGuffin has any weak keys. The complexity of MacGuffin's key schedule eliminates all the obvious candidates, but there still might be subtle interactions.

Consider for the moment what would happen if all (or most) round subkey values were somehow equal in MacGuffin. For concreteness, let E_0 stand for MacGuffin encryption with all round subkeys set to all zeros, and define E_1 similarly. Note that E_0 fails to be an involution for two reasons. First, DES left out the final swap of (L, R) just before the final permutation to ensure that the decryption process is just encryption with the round keys reversed—but MacGuffin explicitly includes the final rotation after the last round, spoiling the similarity between encryption and decryption. Secondly, in any GUFN the decryption process must involve rotations which go in the opposite direction as for encryption; for a balanced Feistel cipher with just 2 registers (L, R) a forward rotation is the same as a backwards rotation. (In the language of group theory, any 2-cycle is its own inverse, whereas a 4-cycle never is.) The same reasoning shows that E_1 is not an involution either. So, as chance would have it, MacGuffin does not suffer from this particular weakness.

The short cycles when iterating $E_0 \circ E_1$ in DES arose from a complementation property and the fact that E_0, E_1 are involutions [17]. Since both those properties are avoided in MacGuffin, one might guess that those types of short cycles aren't present either. I implemented an efficient parallel cycle detection algorithm [16] and applied it to the sequence

$$P, (E_0 \circ E_1)(P), (E_0 \circ E_1)^2(P), \ldots$$

After 10 billion iterations, no cycles were found. The experiment was repeated, this time iterating E_0 approximately 24 billion times; again no cycles were found. This is a partial indication that these types of short cycles have been avoided even when repeated round subkeys are artificially forced on MacGuffin.

Because of the attacks on ciphers with repeated round subkey values [11, 34], it is still interesting to search for weak keys. There are no obvious weak keys, and exhaustively searching over all 128 bit MacGuffin keys is infeasible, so the search for weak keys requires some cleverness. Here is one way which a weak key might occur.

Lemma 12 *Suppose $P = (0, 0, 0, c)$ is a fixed point of E_0, and assume that K is the 128 bit MacGuffin key satisfying $K_{0\ldots63} = K_{64\ldots127} = P$. Then K causes all round subkeys to be* 0 *under the MacGuffin key schedule.*

Proof: It suffices to prove that the round subkeys are all 0 after one call to UpdateSubkey(*subkey*, P) if they start that way. This in turn can be proved by showing that $l = a = b = 0$ after a call to

MacGuffin(*subkey*, (l, a, b, c)) if they start out that way and if the round subkeys are all 0. But this follows immediately from the fact that $(0, 0, 0, c)$ is a fixed point of E_0. □

Lemma 13 *Suppose that subkey*$[r] = (x, y, z)$ *for all* $r \leq 32$ *and that* $P = (l, a, b, c)$ *is a fixed point for those subkeys. Let subkey*$'$ *be the value of the subkey array after a call to UpdateSubkey(subkey, P), and suppose that the result of encrypting P by r rounds is the same whether subkey or subkey*$'$ *are used, for every* $r \leq 32$. *Then*

$$\mathit{subkey}'[r] = (x', y', z') = (x \oplus l, y \oplus a, z \oplus b) \qquad \forall r \leq 32.$$

Proof: Note that P is a fixed point of MacGuffin encryption where the first r rounds use *subkey*$'$ and the last $32-r$ rounds use *subkey*, for every r. The result follows from the definition of UpdateSubkey. □

Theorem 14 *Let K be a 128 bit MacGuffin key, and suppose* $K_{0\ldots63}$ *satisfies the hypotheses of Lemma 13 with* $P = K_{0\ldots63}$ *and* $x = y = z = 0$, *producing* $(x', y', z') = (\alpha, \beta, \delta)$. *Further suppose* $K_{64\ldots127}$ *satisfies the hypotheses of Lemma 13 with* $P = K_{64\ldots127}$ *and* $(x, y, z) = (\alpha, \beta, \delta)$. *Then K generates a weak key by the MacGuffin key schedule.*

Proof: Immediate upon inspection of the key schedule. □

Observe that Lemma 12 is a special case of Lemma 13 with

$$(x, y, z) = (l, a, b) = (x', y', z') = (0, 0, 0).$$

In fact, the second lemma can be further generalized to the case where P is a fixed point of MacGuffin-2^a for $0 \leq a \leq 10$. So it seems that the problem of weak keys is at least partially tied to the existence of fixed points and very short cycles.

The first step is to look for fixed points falling under the case $x = y = z = 0$. It is impossible to exhaustively check all possible plaintext blocks, so I resorted to heuristics instead:

- All fixed points of MacGuffin-4 were found.
- All fixed points of the form $P = (l, a, b, c) = (0, 0, 0, c)$ were found.
- All fixed points of the form $P = (l, a, b, c) = (l, a, l, a)$ were found.
- About a billion random values of P were tried.

The results are shown in Table 14. None of these lead to weak keys, and the exhaustive search shows that there are no fixed points satisfying the conditions of Lemma 12.

There could still be other fixed points. Much of the search space is not covered by the heuristics—even when we restrict our attention to the case $x = y = z = 0$. No search was undertaken for fixed points when $(x, y, z) \neq (0, 0, 0)$, because knowledge of such a fixed point appears to be useless until we find a fixed point with $x = y = z = 0$; see Theorem 14. Finally, no search was undertaken for fixed points of MacGuffin-2^a where $6 \leq a \leq 10$, though some may very well exist. With more computational resources it would be possible to extend the search, but ruling out the existence of weak keys seems to require more subtle insights.

Table 14: Some fixed points of E_0

l	a	b	c
6677	3CD7	6677	3CD7
093E	093E	093E	093E
CEC2	CEC2	CEC2	CEC2
3CD7	6677	3CD7	6677

6.2 Inverting the key schedule

Ideally the MacGuffin key scheduling algorithm should behave like a random function. Failing that, it should be a cryptographically secure collision-resistant one-way function. Finding a collision in the key schedule (two 128 bit keys which generate the same round subkeys) should be infeasible or impossible. Inverting the key schedule (determining the 128 bit key given some of the round subkeys) should require approximately 2^{128} operations. In fact, the last goal is not met, and the MacGuffin key schedule does *not* behave like a random function.

Inverting the key schedule is simplest when the first round subkey is known. If the first round subkey is (x, y, z) the algorithm shown in Figure 5 will derive the 128 bit encryption key. Steps 8–10 require about two trial encryptions per loop iteration, and the loop in step 7 runs through 2^{16} iterations. Steps 2–6 run much faster than the loop in Step 7. Since the loop in Step 1 runs through 2^{64} iterations, this algorithm will discover the true 128 bit encryption key with about 2^{81} encryptions in the worst case and about 2^{80} encryptions on average. This is many orders of magnitude faster than exhaustive search.

InvertKeySchedule(x,y,z)
1. for each possible value of $K_{0\ldots63}$
2. $temp[r][i] \leftarrow 0 \quad \forall r, i$
3.   UpdateSubkey($temp$, $K_{0\ldots63}$)
4. $l \leftarrow x \oplus temp[0][0]$
5. $a \leftarrow y \oplus temp[0][1]$
6. $b \leftarrow z \oplus temp[0][2]$
7.   for each possible 16 bit value c
8. $K_{64\ldots127} \leftarrow$ MacGuffinDecrypt($temp$, (l, a, b, c))
9.   check the guessed value K on a few known plaintexts
10.   if it passes them all, return K as the true key

Figure 5: Inversion given the first round subkey

An extension of this method can be used when the second round subkey is known. Let

$$H(l, a, b, c) = l \oplus f(l \oplus a, a \oplus b, b \oplus c),$$

and define the map $H_{a,b,c} : l \mapsto H(l, a, b, c)$ for fixed a, b, c. Note that the expected cardinality of the inverse image $H^{-1}_{a,b,c}(y)$ is exactly 1, so $H_{a,b,c}$ will usually have just one inverse (or at most just a few inverses) for any given image y. Now the problem of determining the 128 bit encryption key from a known second round subkey can can be described as the problem of inverting $H_{a,b,c}$. Clearly $H_{a,b,c}$ can be inverted by just trying all 2^{16} possibilities for l, which leads to an algorithm with roughly 2^{96} complexity for inverting the MacGuffin key schedule given the second round subkey.

By using the special structure of the f function, we can in fact do better. First guess the 6 bits $l_{0...5}$, so that the inputs and outputs to $S1$ and $S2$ are known. Discard the guesses which are inconsistent with a, b, c, y, which leaves about 2^2 possibilities. Guess the bit l_6, so that the inputs and outputs to $S3$ are known, and again discard impossible guesses. Continue guessing bits from the sequence

$$l_{0...5}, l_6, l_{7,12,14}, l_{11,13,15}, l_9, l_{8,10}$$

discarding inconsistent guesses at each stage. This approach requires about

$$2^6 + 2^2(2 + 2^{-1}(2^3 + 2(2^3 + 2(2 + 2^{-1}(2^2))))) \approx 2^7$$

table lookups, so the map $H_{a,b,c}$ can be inverted at least as quickly as a trial encryption can be performed. A careful counting shows that this method enables an attacker to derive the 128 bit encryption key from knowledge of the round two subkey with expected complexity of about 2^{81} encryptions.

When a later round subkey is known, a similar analysis can be carried through by a fairly straightforward application of these two ideas. The complexity of the best such inversion attacks on the MacGuffin key schedule is presented in Table 15. These results show that the MacGuffin key schedule falls short of achieving ideal one-wayness.

Table 15: Average-case complexity of deducing K given a round subkey

Round	Encryptions
1	2^{80}
2	2^{81}
3	2^{97}
4	1.5×2^{113}
5–32	2^{127}

It may be possible to fix this deficiency of the MacGuffin key scheduling algorithm with a slight modification. Append the following line to the KeySched procedure in Figure 4:

6. UpdateSubkey(*subkey*, $K_{0...63}$)

This change foils the attack described earlier. More research is needed to decide whether the modified version is a completely secure one-way key schedule.

6.3 Inverting the key schedule with related keys

Several authors have introduced the idea of related key cryptanalysis [36, 34]. Suppose we are allowed 2^t ($t < 64$) related key chosen plaintext queries, and we wish to speed up inversion of the key schedule. Choose one plaintext P and obtain the encryption of P under the related keys

$$\mathcal{K} = \{k_{\text{orig}} \oplus \chi : \chi_{t\ldots127} = 0\}.$$

Replace the loop in line 1 of InvertKeySchedule (see Figure 5) by one which only examines keys satisfying $K_{0\ldots t-1} = 0$. Next change line 9 so that the result of the trial encryption $E_K(P)$ is compared against the set of ciphertexts $\{E_k(P) : k \in \mathcal{K}\}$. Upon finding a match $E_K(P) = E_k(P)$ for $k = k_{\text{orig}} \oplus \chi$ with χ, K known, we may guess $K = k$ and thereby guess $k_{\text{orig}} = K \oplus \chi$. By testing this against a few known plaintexts, we can filter out all incorrect guesses with very high probability.

One can see that the modified InvertKeySchedule algorithm still correctly inverts the key schedule given the first round subkey. Observe that there must be some $K \in \mathcal{K}$ so that $K_{0\ldots t-1} = 0$, so the modified InvertKeySchedule will terminate successfully when that value of $K_{0\ldots63}$ is chosen in line 1. The described change can also be applied in a straightforward way to the more general algorithms described earlier for inverting the key schedule given the r-th round subkey when $r > 1$. Thus the complexity of the inversion attacks (see Table 15) is decreased by a factor of 2^t when 2^t related key chosen plaintext queries are possible.

7 Conclusion

MacGuffin was tested for weaknesses with a variety of standard statistical tests. These results were also used to compare MacGuffin to DES, a cipher which is believed to be strong. No weaknesses were found, and MacGuffin behaved about as well as DES in these simple tests.

The cryptographic literature documents a great variety of elementary attacks on DES, and most apply naturally to MacGuffin. By comparing the efficiency of these attacks on DES and MacGuffin, we can obtain partial information on the security of MacGuffin. MacGuffin was at least as resistant as DES to the simple attacks; but two more sophisticated statistical attacks penetrated reduced-round variants of MacGuffin successfully enough to be worrisome.

The powerful techniques of differential cryptanalysis were brought to bear upon MacGuffin. MacGuffin was found to be weaker than DES against differential cryptanalysis. One attack breaks MacGuffin with about 1/350-th the complexity of the best differential attack against DES with independent round subkeys. A differential attack was implemented on a workstation, cryptanalyzing MacGuffin-16 with great success.

Several aspects of the MacGuffin key schedule were examined for weaknesses. No weak keys were found. However, the key scheduling algorithm can be inverted in certain circumstances much faster than is possible with a one-way function. The complexity of inversion is reduced significantly when related key chosen plaintexts are available. A modification was suggested to fix this defect.

MacGuffin has several problematic weaknesses. It is more vulnerable to differential cryptanalysis than DES. It is not sufficiently resistant against linear cryptanalysis [33]. The key schedule is not perfect. The MacGuffin parameters were apparently chosen arbitrarily: the choice of S-boxes is not the best possible, and the input permutation can be greatly improved. In short, long-term use of MacGuffin is not recommended.

On the other hand, MacGuffin is just one particular incarnation of the more general GUFN structure. DES was carefully crafted so all components would mesh together to provide the maximum security possible [2]; it appears that MacGuffin was at a disadvantage with a less circumspect design. It is natural to wonder whether a more careful design based on the GUFN scheme might reap substantial benefits. This is worthy of further study.

8 Acknowledgements

I would like to extend my gratitude to Steve Bellovin for proposing this research topic and offering many helpful suggestions; his kindness is appreciated. Thanks also to Vincent Rijmen for useful discussions.

References

[1] NBS. Data encryption standard, January 1977. Federal Information Processing Standards Publication 46.

[2] D. Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. *IBM J. Res. Develop.*, 38(3):243–250, May 1994.

[3] Thomas A. Berson. Long key variants of DES. In *Advances in Cryptology: CRYPTO '82*, pages 311–313, New York, 1983. Plenum Press.

[4] Bruce Schneier. *Applied Cryptography.* John Wiley and Sons, New York, 1994.

[5] Matt Blaze and Bruce Schneier. The MacGuffin block cipher algorithm. In Bart Preneel, editor, *Fast Software Encryption: Proceedings of K. U. Leuven Workshop on Algorithms.* Springer-Verlag, 1995. (to appear).

[6] Alan G. Konheim. *Cryptography: A Primer.* John Wiley and Sons, New York, 1981.

[7] Carl H. Meyer and Stephen M. Matyas. *Cryptography: A New Dimension in Computer Data Security.* John Wiley and Sons, New York, 1982.

[8] Gustavus J. Simmons, editor. *Contemporary Cryptography: The Science of Information Integrity.* Institute of Electrical and Electronics Engineers, 1992.

[9] Dorothy Denning, editor. *Cryptography and Data Security.* Addison-Wesley, Reading, MA, 1982.

[10] Yvo Desmedt, Jean-Jacques Quisquater, and Marc Davio. Dependence of output on input in DES: Small avalanche characteristics. In *Advances in Cryptology: CRYPTO '84*, pages 359–376. Springer-Verlag, 1985.

[11] Edna K. Grossman and Bryant Tuckerman. Analysis of a weakened Feistel-like cipher. In *1978 International Conference on Communications*, pages 46.3.1–46.3.5. Alger Press Limited, 1978.

[12] A. F. Webster and S. E. Tavares. On the design of S-boxes. In *Advances in Cryptology: CRYPTO '85*, pages 522–534. Springer-Verlag, 1986.

[13] A. K. Leung and S. E. Tavares. Sequence complexity as a test for cryptographic systems. In *Advances in Cryptology: CRYPTO '84*, pages 458–474. Springer-Verlag, 1985.

[14] Helen Gustafson, Ed Dawson, and Bill Caelli. Comparison of block ciphers. In *Advances in Cryptology: AUSCRYPT '90*, pages 208–220. Springer-Verlag, 1990.

[15] John M. Carroll and Lynda E. Robbins. Using binary derivatives to test an enhancement of DES. *Cryptologia*, XII(4):193–208, October 1988.

[16] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with application to hash functions and discrete logarithms. In *2nd ACM Conference on Computer and Communications Security*, 1994.

[17] Don Coppersmith. The real reason for Rivest's phenomenon. In *Advances in Cryptology: CRYPTO '85*, pages 535–536. Springer-Verlag, 1986.

[18] B. S. Kaliski Jr., R. L. Rivest, and A. T. Sherman. Is DES a pure cipher? (results of more cycling experiments on DES). In *Advances in Cryptology: CRYPTO '85*, pages 212–226. Springer-Verlag, 1984.

[19] Hikaru Morita, Kazuo Ohta, and Shoji Miyaguchi. A switching closure test to analyze cryptosystems. In *Advances in Cryptology: CRYPTO '91*, pages 183–193. Springer-Verlag, 1992.

[20] Ralph Wernsdorf. The one-round alternating functions of the DES generate the alternating group. In *Advances in Cryptology: EUROCRYPT '92*, pages 99–112. Springer-Verlag, 1993.

[21] Keith W. Campbell and Michael J. Wiener. DES is not a group. In *Advances in Cryptology: CRYPTO '92*, pages 512–520. Springer-Verlag, 1992.

[22] Walter Fumy. On the f-function of Feal. In *Advances in Cryptology: CRYPTO '87*, pages 434–437. Springer-Verlag, 1988.

[23] M. E. Hellman, R. C. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer. Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard. Technical Report SEL 76-042, Department of Electrical Engineering, Stanford University, 1976.

[24] David Chaum and Jan-Hendrik Evertse. Cryptanalysis of DES with a reduced number of rounds. In *Advances in Cryptology: CRYPTO '85*, pages 192–211. Springer-Verlag, 1986.

[25] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, New York, 1993.

[26] Eli Biham and Adi Shamir. Differential cryptanalysis of Feal and N-Hash. In *Advances in Cryptology: EUROCRYPT '91*, pages 1–16. Springer-Verlag, 1991.

[27] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Advances in Cryptology: CRYPTO '90*, pages 2–21. Springer-Verlag, 1991.

[28] Eli Biham and Adi Shamir. Differential cryptanalysis of the full 16-round DES. In *Advances in Cryptology: CRYPTO '92.* Springer-Verlag, 1993.

[29] Eli Biham and Adi Shamir. Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI, and Lucifer. In *Advances in Cryptology: EUROCRYPT '91*, pages 156–171. Springer-Verlag, 1992.

[30] Xiejia Lai, James L. Massey, and Sean Murphy. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology: EUROCRYPT '91*, pages 17–38. Springer-Verlag, 1991.

[31] Lawrence Brown and Jennifer Seberry. On the design of permutation P in DES type cryptosystems. In *Advances in Cryptology: EUROCRYPT '89*, pages 696–705. Springer-Verlag, 1990.

[32] Vincent Rijmen and Bart Preneel. Cryptanalysis of McGuffin. In Bart Preneel, editor, *Fast Software Encryption: Proceedings of K. U. Leuven Workshop on Algorithms.* Springer-Verlag, 1995. (to appear).

[33] Vincent Rijmen and Bart Preneel. Cryptanalysis of McGuffin, 1995. (a corrected version of [32]).

[34] Eli Biham. New types of cryptanalytic attacks using related keys. In *Advances in Cryptology: EUROCRYPT '93*, pages 398–409. Springer-Verlag, 1994.

[35] J. H. Moore and G. J. Simmons. Cycle structure of the DES with weak and semi-weak keys. In *Advances in Cryptology: CRYPTO '86*, pages 3–32. Springer-Verlag, 1987.

[36] Robert Winternitz and Martin Hellman. Chosen-key attacks on a block cipher. *Cryptologia*, XI(1):16–20, January 1987.