Cryptanalysis of the Mercy Block Cipher

Scott R. Fluhrer
sfluhrer@cisco.com

Cisco Systems, Inc.
170 West Tasman Drive, San Jose, CA 95134

Abstract. The Mercy block cipher is examined, and a differential attack is shown. Using this differential, a method of distinguishing the cipher from a random permutation is given.

1 Introduction

Mercy is a block cipher that was presented at Fast Software Encryption 2000 [1], and was designed by Paul Crowley. It has a block size of 4096 bits, 128 bits of spice[1], and a variable sized key. It was specifically designed for bulk disk encryption, where each disk block would be encrypted separately, and with the sector index used as the spice. An explicit design goal was that any procedure for distinguishing Mercy encryption using a 128 bit key from a sequence of 2^{128} random permutations should show no more bias toward correctness than a key guessing attack with the same work factor.

We show that this design goal is not achieved, by using a differential to distinguish the cipher from randomness with 2^{27} chosen plaintexts. In addition, we describe how to rederive some of the hidden whitening bits. Further, we analyze why Mercy was vulnerable to this attack.

This paper is structured as follows. In Section 2, the Mercy block cipher is described. Section 3 describes the differential, and section 4 shows how this differential can be used to attack the cipher. Section 5 summarizes our conclusions.

2 Description of Mercy

Mercy is a Feistel network with 6 rounds, and partial pre- and post-whitening. The key is used to derive a substitution table[2] $T : \{0,1\}^8 \rightarrow \{0,1\}^{32}$. The T function is used to define a function $M : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$, as shown on Figure

[1] Spice is a parameter to the cipher, designed to be known to the attacker. It is intended to allow the user to modify the actual permutation implemented by the block cipher, possibly on a per-block basis, without forcing a full rekey

[2] It actually derives affine transformations that is used along with a fixed table $N : \{0,1\}^8 \rightarrow \{0,1\}^8$ to expand the T substitution table, but this is irrelevant to this attack.

1[3]. The important point for this attack is that, as long as the most significant 8 bits have zero differential, M preserves any differential in the remaining bits with probability 1, shifting that differential over 8 bits.

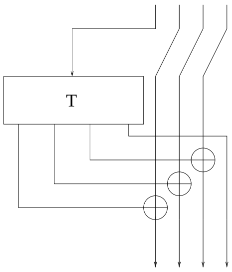

Fig. 1. Operation M, with each line standing for 8 bits.

The M function is used, along with an exclusive-or and two additions modulo 2^{32}, to define the function $Q : \{0,1\}^{128} \times \{0,1\}^{32} \rightarrow \{0,1\}^{128} \times \{0,1\}^{32}$, as shown on Figure 2.

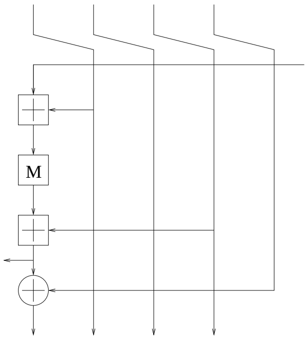

Fig. 2. Operation Q, with each line standing for 32 bits.

The Q function is used to define the function $F : \{0,1\}^{2048} \times \{0,1\}^{128} \rightarrow \{0,1\}^{2048}$, as shown on Figure 3.

The F function is then used as the round function of a six round Feistel network, along with partial pre- and post-whitening, as shown in Figure 4.

There are no attacks on Mercy previous to this attack described in the open literature.

[3] Note that the format of this diagram differs from the diagram in [1] in that the least significant octet on the right.

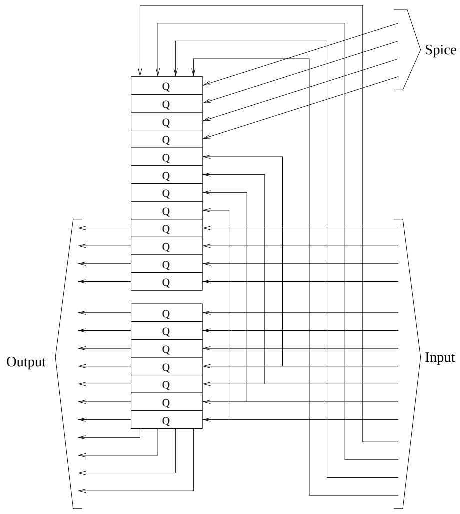

Fig. 3. Operation F, with each line standing for 32 bits.

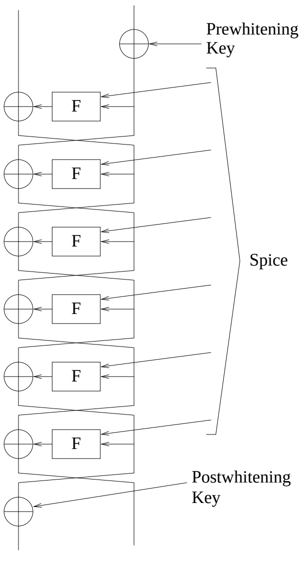

Fig. 4. Mercy, with each line standing for 2048 bits, except for the lines carrying spice, which are 128 bits.

3 Description of the Differential

The first thing to note about this structure is in the F function (Figure 3). If we assume that the last 8 32 bit words of the input halfblock and the spice have zero differential, then the effect of an input differential[4] in a particular input word can be characterized by a differential in the corresponding output word, and a differential in the 128 descending bits output by that Q box. Normally, a differential in the 128 descending bits would avalanche throughout the remainder of the block. However, what is possible is if several consecutive input words have nonzero differentials, then the descending lines immediately afterwards may have zero differential. If the rest of the input block also has zero differential, that means that the sole effect of that input differential would be an output differential in the corresponding output words.

The second thing to note about Mercy is that, within the Q function, only 8 bits are presented to a key-dependent sbox. All other operations within the Q function are either linear or near linear. Hence, if we arrange things so that the input to the key-dependent sbox always has zero differential, we have a realistic prospect of having a high probability differential.

Figure 5 has the simplest example of such a differential (which is not a differential used in the attack – the differentials actually used are approximately twice as long). A specific difference comes in from the input half-block, flows through the Q boxes as indicated, outputs a specific differential to the output half-block, and leaves the descending bits to the next Q box with a zero differential. The differentials that come in from the input halfblock have been carefully chosen to cancel out the effects of the differentials within the Q box itself before they can avalanche. This is how these types of differentials work.

Also note that this is not a probability 1 differential: in three of the additions (marked by *), we need to assume that the carry between bits[5] 23 and 24 has zero differential, and thus the addition treats the differential exactly the same as an exclusive-or would. Assuming that the actual values of the bits are effectively random (which we will assume), this holds with probability approximately[6] 0.5, and so this differential holds with probability 2^{-3}. The unmarked additions with differential inputs work only with differentials in bit 31. Because the addition ignores the carry from bit 31, addition always treats that differential exactly the same as exclusive-or would.

We also note that the exact place within the halfblocks where the differential occurs doesn't matter. As long as the last 8 32-bit words have zero differential (required because those words are fed into the beginning of the Q stack), the attacker can place the differentials where convenient.

Of course, we need to do more than have a differential for a single round. We need to set up differentials so that it leaves the cipher in a state where the next

[4] All differentials considered within this paper are bitwise differentials.

[5] This paper will use the convention that the least significant bit in a word is labeled as bit 0.

[6] Because the carry is biased slightly towards zero, the probability is slightly higher.

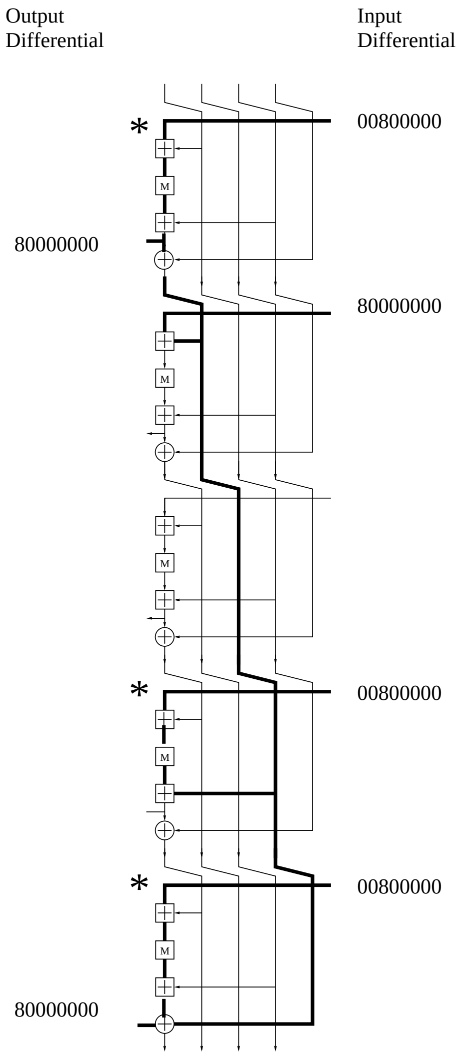

Fig. 5. An example differential within 5 Q boxes contained within the F function. Boldened lines contain a nonzero differential, and the input and output lines are labeled with the exact differential values.

round differential is also bounded. We need this to occur for at least 4 rounds, for reasons that will be discussed below. It turns out that we can actually manage 5 rounds.

We will use the abbreviated format for differentials where a single character corresponds to an entire octet, and a 0 stands for a zero differential, and an X stands for a differential in the seventh bit of the octet. Hence, a differential of 00008000 (in hexadecimal) in a word would be abbreviated as 00X0. We will also list differentials only in the area that the attacker selects for the differential to occur. All other regions of the halfblocks always have zero differential.

Then, let us consider the following differential in the corresponding sections in the plaintext half-blocks, and where the spice has zero differential:

Left: X0X0 0000 X000 0000 0000 0000 00X0 0000 X0X0
Right: 0X00 X000 0000 0X00 0000 X000 0000 0X00 0X00

This differential goes through the pre-whitening without effect. Then, the right halfblock is presented to the F function. This differential is similar to the example differential given above, except that the single bit difference cycles through the 4 internal lines twice. This differential holds with probability 2^{-4} and produces this differential on the F output:

X000 0000 0000 0000 0000 0000 0000 0000 X000

Exclusive-oring that into the the left half-block, and swapping half-blocks gives as output of round 1 the following differential:

Left: 0X00 X000 0000 0X00 0000 X000 0000 0X00 0X00
Right: 00X0 0000 X000 0000 0000 0000 00X0 0000 00X0

The right halfblock is presented to the F function, and is somewhat more complex. This differential holds with probability 2^{-11} and produces this differential on the F output:

0X00 X000 0000 0X00 0000 X000 0000 0X00 0X00

It turns out that this differential is the same as the differential in the left half-block, and so, after exclusive-oring and swapping half-blocks, we get the following differential after two rounds:

Left: 00X0 0000 X000 0000 0000 0000 00X0 0000 00X0
Right: 0000 0000 0000 0000 0000 0000 0000 0000 0000

This is the trivial one-round differential, and so with probability 1, this produces a zero differential on the F output, and after swapping half-blocks, we get the following differential after round 3:

Left: 0000 0000 0000 0000 0000 0000 0000 0000 0000
Right: 00X0 0000 X000 0000 0000 0000 00X0 0000 00X0

The right halfblock differential is exactly the same as in round 2, and hence it, with probability 2^{-11} produces the same differential on the F output, and after the exclusive-or, and swapping half-blocks, we get the differential after round 4:

Left: 00X0 0000 X000 0000 0000 0000 00X0 0000 00X0
Right: 0X00 X000 0000 0X00 0000 X000 0000 0X00 0X00

As we will show below, a four round differential is sufficient for use as a distinguisher, and we have demonstrated one at total probability $2^{-(4+11+0+11)} = 2^{-26}$. However, if we do need another round, we can continue. The right half-block differential is identical to the differential in round 1, and so with probability 2^{-4} produces the same differential on the F output, and after the exclusive-or, and swapping half-blocks, we get the differential after round 5:

Left: 0X00 X000 0000 0X00 0000 X000 0000 0X00 0X00
Right: X0X0 0000 X000 0000 0000 0000 00X0 0000 X0X0

This 5 round differential holds with a probability $2^{-(4+11+0+11+4)} = 2^{-30}$

Note that the characteristic formed from this differential are not formed by an iterated characteristic. Instead it is a specific concatenation of single round characteristics, where each characteristic leaves the cipher in exactly the state the next round characteristic requires.

4 Using this Differential

The most obvious use of this differential is as a distinguisher. The four round differential works as a distinguisher because the F function does not have complete diffusion. A 32 bit word at offset S of the F output is a function of the last 8 words and the words at offset 0 through S of the right halfblock, the spice and the key. With the above four round differential, at round five, all these inputs have zero differential for all offset S prior to the actual differential, as well as words 0 through S in the left halfblock. Therefor, the corresponding words in the left halfblock will still have zero differential after round 5, and since those words are never modified afterwards (other than xoring the postwhitening key) those words are output with zero differential. If we chose to put the input differential near the end of the halfblocks, the occurrence of the four round differential will cause the majority of one of the ciphertext halfblocks to have zero differential. This area is sufficiently large[7] that the probability of a zero differential of that size by coincidence is negligible.

We also note that this attack can be extended for variants of Mercy with more rounds. For example, the 5 round differential can be used in a boomerang configuration [2] to distinguish a 10 round Mercy variant from a random permutation with approximately 2^{122} chosen plaintexts and ciphertexts. In addition, there is a more complex differential over 6 rounds at probability 2^{-114}, which can be used to distinguish an 8 round Mercy variant from a random permutation

[7] Possibly as large as 1504 bits

using only chosen plaintexts. These show that, rather than adding additional rounds, modifying the F function would be recommended.

At first glance, this differential would appear to be sensitive to the precise organization of operation Q. To study this, a computerized search was done over a considerable number of variants of operation Q. Each variant studied had 1-4 operations where one of the internal lines were added or exclusive-ored into the leftmost data line, and where the the various operations were done in various orders. It turns out that all variants studied permitted differentials similar to the one presented in section 3. In addition, it turns out that the actual operation Q that Mercy used was optimal for variants with three operations in terms of differential probability, except for some variants that applied operation M directly to the data from the right halfside, and for some variants which used three additions, both of which possibly weaken the cipher for other reasons. When variants with an additional add or exclusive-or operation were considered, the optimal variants turned out to permit a four round differential with probability 2^{-52}, and one such variant shown on Figure 6 (and the other three are slight modifications of this). It is apparent that even this modified operation Q is still insufficient to prevent exploitable differentials.

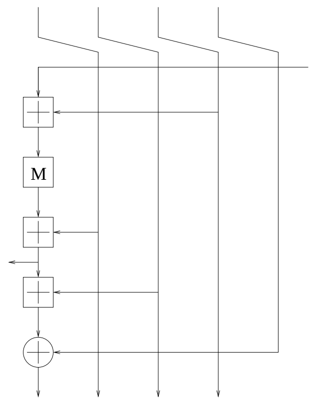

Fig. 6. Modified operation Q that yields a four round differential resistance of 2^{-52}.

5 Conclusions

We have shown a differential with a probability 2^{-26} that is detectable after 6 rounds. We have shown how to use this as a distinguisher which violates the Mercy design criteria.

This differential works because of two specific properties of the cipher. The first property is that, in spite of having a large number of sbox lookups during a block encryption, the actual number of sbox inputs are relatively small compared to the number of bits within the block itself. During a single block encryption, a total of 3264 bits are presented to the sbox. This would appear to be a large number, except there is a total of 4096 bits within the block, or in other words, there is less than a single sbox lookup per input octet. This attack took advantage of that relative paucity of sbox lookups by managing to control the sbox inputs while having nonzero differentials elsewhere in the cipher.

The other property that this attack used was the fact that, if we examine the effect of a differential within a sequential part of the right half-block, we find the effects can be described by the differential in the descending lines in the Q array immediately afterwards and in the corresponding segment in the left half-block. If we managed to cancel out the differential immediately afterwards within the Q array, the only remaining effect was confined to an area that would, in the next round, modify the first area (and the descending outputs of that round's Q array). Hence, by repeatedly confining the differential, we were able to have a differential that was zero in most of the cipher, while two limited areas interacted with each other. If one were to modify the relationship between F inputs and outputs so that the effect of a differential could not be easily confined to two small areas, this should prevent such limited differentials. It would not, in fact, prevent large scale differentials, however, large scale differentials would have large numbers of additions with active differentials, and because each such addition[8] would stop the differential with nontrivial probability, a large number of them would make the differential hold with extremely low probability.

It will require more research to determine whether this attack can derive information on the internal d tables (or equivalently, the T function), or how to derive a Mercy variant that is immune to versions of this attack.

References

1. Crowley, P., "Mercy: a fast large block cipher for disk sector encryption", Proceedings of the Fast Software Encryption 2000, Springer-Verlag.
2. Wagner, D., "The boomerang attack", Proceedings of the Fast Software Encryption 1999, Springer-Verlag.

[8] Other than additions with differentials only in the MSBit.