An Improvement of Linear Cryptanalysis with Addition Operations with Applications to FEAL-8X

Eli Biham and Yaniv Carmeli(✉)

Computer Science Department, Technion - Israel Institute of Technology,
3200003 Haifa, Israel
{biham,yanivca}@cs.technion.ac.il
http://www.cs.technion.ac.il/∼{biham,yanivca}/

Abstract. FEAL is a Feistel cipher that uses addition operations. Since its introduction 26 years ago it played a key role in the development of many cryptanalytic techniques, including differential and linear cryptanalysis. For its 25th anniversary Mitsuru Matsui announced a challenge for an improved known plaintext attack on FEAL-8X. In this paper we describe our attack and introduce several improvements to linear cryptanalysis that allowed us to recover the key given 2^{14} known plaintexts in about 14 h of computation, and led us to win the challenge. An especially interesting improvement considers the approximation of addition-based S-boxes by partitioning into several sets in a way that amplifies the bias, and therefore allows for a reduction in the number of required known plaintexts as well as saving computation time. We also describe attacks that require only a few (even 2 or 3) known plaintexts that recover the key much faster than exhaustive search.

Keywords: FEAL · Linear cryptanalysis · Partitioning · Meet in the middle

1 Introduction

FEAL [13] was introduced in 1987 as a fast encryption algorithm which combines the simplicity of software-based operations with an improved security over prior designs. Over the years FEAL inspired the development of many cryptanalytic techniques, including differential and linear cryptanalysis [3,7,8]. The best known attacks on FEAL required (until recently) a few hundreds of chosen plaintexts [4] or 16 million known plaintexts [2,6].

In CRYPTO 2012 Mitsuru Matsui announced a year-long challenge [6] for developing improved attacks on FEAL-8X [9], and an award which will be given to the best attack capable of recovering the key of given sets of known plaintexts with various amounts of data. The attack recovering the key using the smallest number of known plaintexts would be declared the winner. In the course of this year we developed an improved attack capable of recovering the key of

A. Joux and A. Youssef (Eds.): SAC 2014, LNCS 8781, pp. 59–76, 2014.
DOI: 10.1007/978-3-319-13051-4_4

FEAL-8X, and three weeks before the deadline we submitted our solution for the challenge set with a million known plaintexts, and were the first to submit a correct solution. A few days later another group submitted a solution for a smaller set of 2^{15} known plaintexts. It took us another two weeks to finalize our program with all the additional tricks and to submit the solution for the set of 2^{14} known plaintexts, which became the winning solution. The secret key is `5681891EEC34CE1241ED0F52C9C23F65`.

In this paper we present the cryptanalytic attacks that we developed for this challenge, and the techniques that we used to improve linear cryptanalysis. We first describe a linear attack which uses a 6-round approximation and analyzes both the first and last rounds simultaneously, recovering 37 subkey bits in total. We then describe how running it a second time with a different approximation can reduce the number of required plaintexts and find 44 bit of the subkeys. We describe the rest of the steps needed in order to recover the remaining subkeys and show how the FEAL-8X key can be reconstructed from those subkeys. The above mentioned techniques can find the FEAL-8X key given 2^{15} known plaintexts in about 26 h on our computer.

We then present our main contribution – a new partitioning method that can amplify the bias of a linear approximation of addition. The data is partitioned into two sets such that in one of the sets the bias of the linear approximation is stronger than it is when all the messages are considered. Interestingly, we cannot tell in advance which of the two sets is the one with the increased bias, and therefore we try both of them. The amplified bias allows us to reduce the number of plaintexts needed for the attack while keeping the analysis time per plaintext the same. Due to the smaller number of required plaintexts the attack time when using this method even decreases. Incorporating this technique with our previous methods allowed us to find the key given 2^{14} known plaintexts in about 14 h. In the summary of this paper (Sect. 7) we discuss the differences between our technique and *Partitioning Cryptanalysis* [5].

In addition to the practical attacks we also discuss attacks that can find the key with fewer plaintexts faster than exhaustive search. We describe an attack that can recover the key given 2^{10} known plaintexts in time of 2^{62} FEAL-8X encryptions. In addition, we describe attacks in which given only 11–21 known or chosen plaintexts the FEAL-8X key can be recovered with complexity about 2^{80} and given 2 or 3 known plaintexts the FEAL-8X key can be recovered with complexity about 2^{96}. These attacks combine linear cryptanalysis and differential cryptanalysis with exhaustive search of many subkeys, as well as meet in the middle attacks. These attacks exploit the fact that the total size of the subkeys is not sufficiently larger than the size of the key.

The structure of the paper is as follows: In Sect. 2 we describe FEAL-8X, give two equivalent descriptions of the cipher, and define notations. In Sect. 3 we describe the linear attack that recovers the key given 2^{15} known plaintexts. In Sect. 4 we present the new partitioning method and how to recover the key given 2^{14} known plaintexts. In Sect. 5 we extend the methods from the previous sections, and describe an attack on 2^{10} known plaintexts faster than exhaustive search. Finally, in Sect. 6 we describe the attacks that require only a few known

ciphertexts or a few chosen plaintexts. In Appendix A we describe an efficient implementation of our attacks which is able to save a factor of about 2^6 in the attack time. Appendix B illustrates the linear approximations used in our attacks. In the full version of the paper we also show how to find the key of FEAL-8X given the subkeys that are found by our attacks.

2 The Cipher FEAL-8X

The block size of FEAL-8X is 64 bits and the key size is 128 bits. The key processing algorithm of FEAL-8X takes the 128-bit key and generates 16 subkeys, denoted by $K0$–Kf, each of length 16 bits.

FEAL-8X is an 8-round Feistel cipher. Before the first round the plaintext is mixed with a 64-bit whitening subkey ($K89ab$) which is followed by XORing the left half of the data into the right half. The inverse of this operation is performed after the last round, i.e., the left half of the data is XORed into the right half and the result is mixed with a 64-bit whitening key ($Kcdef$). In each round a function F is computed on the right half of the data and a 16-bit subkey (one of $K0$–$K7$), and the output is XORed into the left half. The two halves are then swapped.

The function F takes four bytes as input, and starts by XORing the first and last bytes into the two middle bytes, and then XORs the subkey into the same bytes. It then applies four S-boxes in the order described in Fig. 1. Each S-box adds two bytes and an index (0 or 1) and rotates the output by two bits to the left. FEAL-8X and the F-function are outlined in Fig. 1.

2.1 An Equivalent Description of FEAL-8X

In order to simplify the analysis we prefer to eliminate the whitening keys. This is possible on one end of the cipher by extending the size of the subkeys to 32 bits in each round and by XORing the eliminated whitening key information into all the subkeys. We consider two equivalent descriptions of the cipher. In the first we eliminate the whitening at the beginning of the cipher, and in the second we eliminate the whitening at the end (this latter version is outlined in Fig. 2). The 32-bit subkeys of the equivalent description are called *actual subkeys*. We call the actual subkeys of the version with eliminated whitening key at beginning *encryption actual subkeys* and denote them by $EK0$–$EK7$, while we call the actual subkeys of the version with eliminated whitening key at the end *decryption actual subkeys*, and denote them by $DK0$–$DK7$. To simplify the description we define the function

$$mw(X, Y) = (Y_0, Y_0 \oplus Y_1 \oplus X_0, Y_2 \oplus Y_3 \oplus X_1, Y_3)$$

where X is a 16-bit value, Y is a 32-bit value, and $X_0, X_1, Y_0, Y_1, Y_2, Y_3$ are their individual bytes. Note that $mw(X, Y)$ is just the first part of the F-function before the S-boxes (see Fig. 1). The mapping between the subkeys of all three

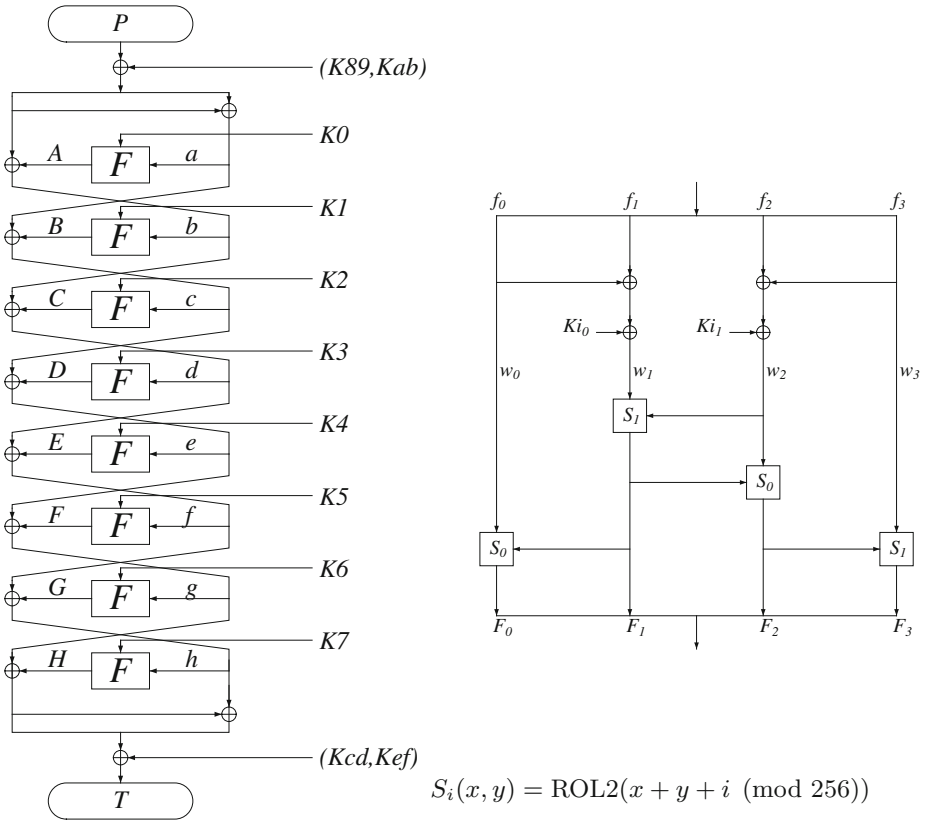

Fig. 1. The outline of FEAL-8 and of the F-function

descriptions of the cipher (the subkeys of FEAL and the two equivalent descriptions) is summarized in Table 1.

In our attacks when we analyze the last rounds of the cipher we assume the whitening at the end is zero, and therefore retrieve the bits of the decryption actual subkeys DK. Similarly, when we analyze the first rounds of the cipher we retrieve the bits of the encryption actual subkeys EK.

Note that since there is a linear relation between the subkeys of all three descriptions of FEAL it is possible to target actual subkeys of different descriptions in the same linear attack. For example, the attack presented in Sect. 3.2 targets both $EK0$ and $DK7$.

3 First Attack – Finding the Key Using 2^{15} Known Plaintexts

In this section we describe a linear attack that requires 2^{15} known plaintexts and finds the key in about 26 h on a server with an Intel(R) Xeon(R) X5650

Table 1. The Subkeys of FEAL-8X and the actual subkeys of the equivalent descriptions

Subkeys of FEAL-8X	Equivalent description without whitening at the beginning	Equivalent description without whitening at the end
$K89ab$	0	$(K89 \oplus Kcd \oplus Kef, Kab \oplus Kef)$
$K0$	$EK0 = mw(K0, K89 \oplus Kab)$	$DK0 = mw(K0, Kcd)$
$K1$	$EK1 = mw(K1, K89)$	$DK1 = mw(K1, Kcd \oplus Kef)$
$K2$	$EK2 = mw(K2, K89 \oplus Kab)$	$DK2 = mw(K2, Kcd)$
$K3$	$EK3 = mw(K3, K89)$	$DK3 = mw(K3, Kcd \oplus Kef)$
$K4$	$EK4 = mw(K4, K89 \oplus Kab)$	$DK4 = mw(K4, Kcd)$
$K5$	$EK5 = mw(K5, K89)$	$DK5 = mw(K5, Kcd \oplus Kef)$
$K6$	$EK6 = mw(K6, K89 \oplus Kab)$	$DK6 = mw(K6, Kcd)$
$K7$	$EK7 = mw(K7, K89)$	$DK7 = mw(K7, Kcd \oplus Kef)$
$Kcdef$	$(K89 \oplus Kab \oplus Kcd, Kab \oplus Kef)$	0

2.67 GHz processor with 12 cores. We first describe a 6-round linear approximation and then the basic attack which performs the analysis on both ends of the cipher simultaneously. We then describe how to use it with reduced number of plaintexts, and how to recover the rest of the actual subkeys and the full key.

3.1 The Linear Approximations

In [1,11] eight 7-round linear approximations with a bias of about 2^{-9} were presented. The attack we present in this paper uses two 6-round approximations with bias of about 2^{-6}, which we got by truncating two of the 7-round approximations of [1,11] by one round. These approximations are outlined in Figs. 4 and 5 in Appendix B.

3.2 The Basic Attack

The attack we present targets both the encryption actual subkey of the first round ($EK0$), and the decryption actual subkey of the last round ($DK7$). The six-round linear approximation covers the six middle rounds of the cipher (rounds 1–6), while the first and last rounds are used for analysis. We found that when using Approximation 1 there are only 37 bits of $EK0$ and $DK7$ that affect the parity of the bits in the approximation: 22 bits in the last actual subkey ($DK7$, given by the mask `03 FF FF 0F`), and 15 bits of the first actual subkey ($EK0$, given by the mask `00 7F 7F 00` and the parity of the two bits `00 80 80 00`). The remaining 27 bits of $EK0$ and $DK7$ have no impact on the parity of the bits in the linear approximation of the six middle rounds. It is therefore that this basic attack finds the 37 bits of the two actual subkeys.

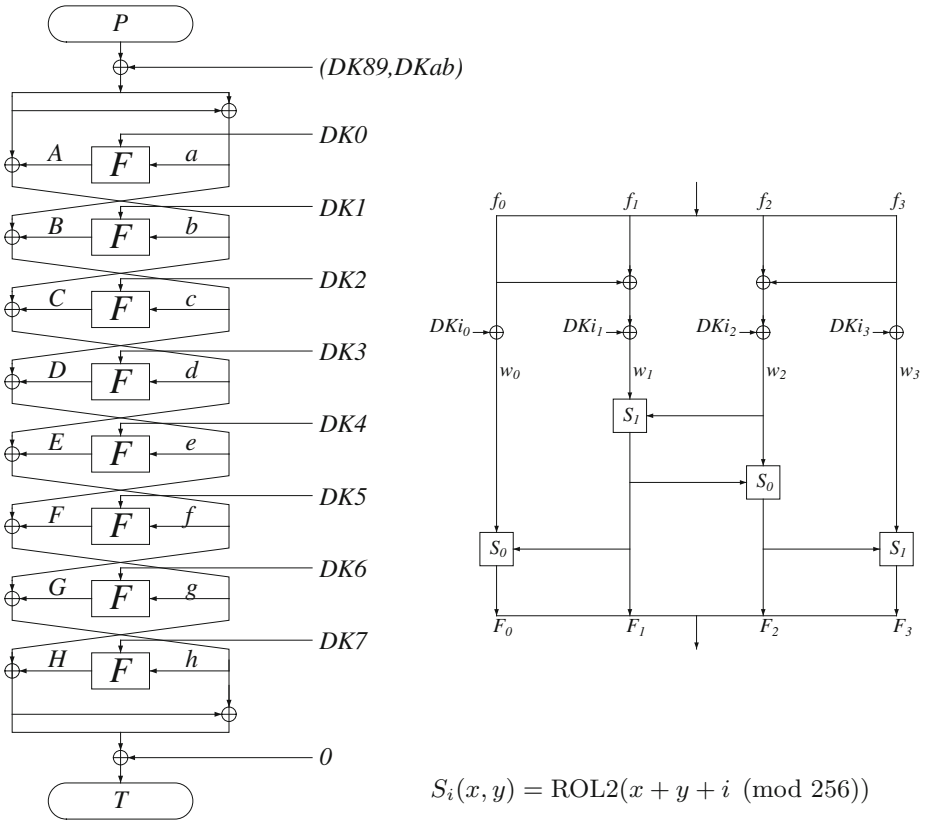

Fig. 2. Equivalent description of FEAL-8X without whitening at the end

The attack is as follows:

1. For each of the 2^{15} candidates for the 16 bits of $EK0$:
   (a) For each of the 2^{22} candidates for the 22 bits of $DK7$:
      i. For each known plaintext P, C:
         A. Decrypt C by one round using $DK7$.
         B. Encrypt one round of P using $EK0$.
         C. Compute the parity of the approximated bits.
      ii. Count the number of messages for which the linear approximation holds and compute the bias.
2. The correct key is expected to be the one with the highest bias.

We also observed that not all 37 bits of the subkeys have the same impact on the bias. While some bits completely throw off the observed bias if guessed incorrectly, others have only a minor impact. We can take advantage of this observation to reduce the running time of the attack by excluding a few such bits with a minor impact on the bias, and to search for them only when the rest

of the bits are already known. For example, instead of guessing 15 bits of $EK0$ with a bias of about 2^{-6}, we may guess only 13 bits (the 12 bits whose mask is 00 6E 7F 00, and the parity of the two bits 00 80 80 00) with a slightly lower expected bias of $2^{-6.5}$, and save a factor of 4 in computation time.

Clearly, the more data we have at our disposal the more accurate the results are (since it is easier to detect the linear bias). If the available data is a lot larger than required in order to detect the bias then we have more freedom to exclude such minor-impact bits (as the measurement of the bias is only slightly inaccurate). As the number of known plaintexts decreases, the identification of the correct key becomes harder (as the bias is harder to detect), and in this case we usually cannot afford to reduce the bias in return for speeding up the attack.

3.3 Matching Subkeys from the Backward and Forward Directions

As noted above, the basic attack does not suffice to find the correct key using 2^{15} known plaintexts. In this section we apply the basic attack twice: once in the forward direction, and once in the backward direction.

We first generate a list $L1$ of the N (for some parameter N) keys which exhibit the highest bias according to Approximation 1 in the forward direction, as described in Sect. 3.2. Recall that for each such key we get 15 bits of the first encryption actual subkey $EK0$, and 22 bits of the last decryption actual subkey $DK7$.

We now run the attack again in the backward direction, i.e., we use the reverse of Approximation 1. In this run we guess 22 bits of $EK0$ and 15 bits of $DK7$. We generate a second list $L2$ of the N keys that exhibit the highest biases.

There is an overlap of 15 bits between the bits we guess in $EK0$ in both runs, and similarly, an overlap of 15 bits in $DK7$. Seven bits of $EK0$ are available only in $L2$, and seven bits of $DK7$ are available only in $L1$. The correct value of these 30 overlapping bits is expected to be in both lists. In such a case, we can easily find the correct value of $30 + 7 + 7 = 44$ bits of the actual subkeys as the (usually single) value that has a match in those 30 bits in both lists.

As we noted earlier, some of the bits of the key only have a minor impact on the measured bias if they are guessed incorrectly. If we cannot find a match between an entry in $L1$ and an entry in $L2$, we can try looking for entries that have a low Hamming distance in the overlapping bits, and between these prefer entries that differ in bits that are known to have a minor impact on the bias.

This is the most time-consuming part of our attack. When we ran it[1] on the server mentioned above it found the 44 bits of the actual subkeys within 24 h using 2^{15} known plaintexts (12 h for each call to the basic attack). The correct key bits were among the top $N = 3200$ keys in each list.

3.4 Retrieving the Rest of the Subkeys

In the previous section we found 44 bits of the actual subkeys. In this section we briefly describe additional steps for finding the rest of the bits of $EK0$ and $DK7$,

[1] With the implementation improvement described in Appendix A.

as well as the rest of the actual subkeys. The steps are described in the order in which they are performed, as each step assumes knowledge of the subkey bits that are retrieved in the preceding steps.

Finding 8 additional bits of *EK*0 and *DK*7. This step is similar to the attack presented in Sect. 3.2, but uses Approximation 2 instead of Approximation 1. Since the linear approximation is different, there are also different bits of the subkeys of Rounds 0 and 7 that affect the parity. The bits of $EK0$ that affect the parity are given by the mask `3F FF FF 00` and the bits of $DK7$ are given by the mask `0F FF FF 03`. Most of those bits are already known, except for eight bits. The correct values of these remaining eight bits can be identified by standard linear cryptanalysis techniques, similarly to the attack of Sect. 3.2. After this step is performed we know 26 bits in each of $EK0$, $DK7$, a total of 52 actual subkey bits.

Finding 4 additional bits of *DK*7 and 15 bits of *DK*6. At this point there are still 6 bits missing in the subkey $DK7$, which are difficult to retrieve by analyzing Round 7. We therefore move on to analyze Round 6 by using a shorter linear Approximation. We use the first five rounds of Approximation 1 with a bias of 2^{-3}, and use it to cover rounds 1–5. In order to compute the parity of the approximated bits in Round 5 we need to guess the values of four more bits of $DK7$[2], and 15 bits of $DK6$. After this step is performed we know a total of 30 bits of $DK7$ (given by the mask `7F FF FF 7F`) and 15 bits of $DK6$ (given by the mask `00 7F 7F 00` and in addition the parity of bits 00 80 80 00).

Finding 7 additional bits of *DK*6. This step is similar to the previous step, but this time we use a 5-round approximation obtained from the last five rounds of Approximation 1, which covers Rounds 1–5. There are 22 bits in $DK6$ that affect the parity of this linear approximation. We already found 15 of them in the previous step, and we should now search for the remaining seven.

Finding 4 additional bits of *DK*6. We use a 5-round approximation comprised of the last five rounds of Approximation 2, which covers Rounds 1–5 We can obtain four more bits of $DK6$, and get a total of 26 bits of $DK6$.

Finding the rest of the subkeys *DK*1–*DK*7. In a similar way, we can attack the rest of the rounds until we have all the actual subkeys $DK1$–$DK7$. Note that as we progress in the attack, analyzing each additional round becomes easier for two main reasons: First, we use shorter approximations with higher biases, which significantly decrease the chances of errors. Second, since the actual subkeys $DK0$, $DK2$, $DK4$ and $DK6$ have 16 bits in common (and similarly for $DK1$, $DK3$, $DK5$ and $DK7$) there are only 16 bits to retrieve in each of those actual subkeys once $DK6$ and $DK7$ are fully known.

Finding *EK*0–*EK*6. Once we finish recovering the decryption actual subkeys, we can repeat the entire process in the reverse direction in order to find the

[2] The value of the two remaining bits of $DK7$ can only be determined when we analyze round 3. Until then those bits have only a linear effect on the parity of the approximation, and therefore cannot be discovered by methods of linear cryptanalysis.

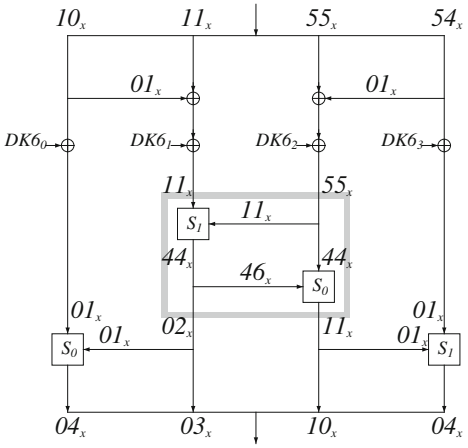

Fig. 3. The approximation of the seventh round

encryption actual subkeys $EK0$–$EK6$.[3] These actual subkeys depend on the whitening key of the plaintext, and are needed in order to retrieve the FEAL-8X key.

Finding The Key Itself. Given $DK1$–$DK7$ and $EK0$–$EK6$ we find the FEAL-8X key within a fraction of a second. The details are omitted here due to space constraint, but the algorithm is described in the full version of the paper.

4 The Partitioning Technique – Finding the Key Using 2^{14} Known Plaintexts

In this section we describe a technique that can reduce the number of known plaintexts by a factor of 3.1 compared to the algorithm of Sect. 3.2. In this technique we partition the data into several sets, such that the bias of the approximation in some of them is higher than when measured across all the data, with a ratio that overcomes the smaller number of messages in those sets. Therefore, fewer messages are required in order to detect the amplified bias.

4.1 A Simplified Example

We apply this technique to Round 6 of the cipher in the inner loop of the algorithm, after the output of the last F-function is already (partially) computed. It is therefore that most bits of the inputs to the S-boxes of Round 6 are known up to an XOR with $DK6$.

[3] We note that instead of searching for $EK0$–$EK6$, we can continue the analysis in the decryption direction and retrieve the actual subkey $DK0$ and the whitening key. Once all the decryption actual subkeys $DK0$–$DK7$ and the whitening key are known, the encryption actual subkeys $EK0$–$EK7$ can easily be computed (see Table 1).

At Round 6 we approximate the first S-box by 11 11 → 44 (see Fig. 3). The input mask 11 11 is approximated to the output mask 44 through the addition operations in the S-boxes (and the rotation), and therefore the quality of the approximation is determined by the carry bits from lower bits into the approximated bits.

We are interested in improving our control on the carry bits, which in turn will improve our approximations.

For that we identified that some of the bits in the inputs to this S-box (denoted by w_1 and w_2 in Fig. 2) in this round are known to us up to an XOR with the actual subkey $DK6$ (as mentioned above).

The approximation 11 11 → 44 approximates two bits through the addition operations. One of them involves the addition of the least significant bits of the inputs (mask 01 01 or $w_{1,0} + w_{2,0} = F_{1,2}$, where w_i are the input bytes to the S-boxes and F_i is the output, as denoted in Fig. 2, and $w_{i,j}$ is bit j of w_i). The approximation of this bit has probability 1, as there cannot be a carry into the LSB. The other approximates Bit 4 of both inputs (mask 10 10), the carry to which involves Bit 3 of both inputs ($w_{1,3}$ and $w_{2,3}$, identified by the mask 08 08). If we would know in advance that the unknown values of these two bits are both 0 then it is certain that there cannot be any carry into Bit 4, which would ensure that this approximation will also have probability 1 (bias +0.5). Similarly, when both bits are 1, a carry from this bit to the next one is guaranteed, and therefore we would also be able to make the approximation with probability 1 (knowing that the carry always flips the approximated output, thus the bias is −0.5). In the other cases (where the bits $w_{1,3}$ and $w_{2,3}$ are either 0,1 or 1,0), we have no idea what the carry is, but we expect that it would occur in about half of the inputs, which would cause the bias to be much closer to zero. We refer to the four possible cases by the values of $w_{1,3}$, $w_{2,3}$ as cases 00, 11, 01, and 10, respectively. The bias of the S-box (on all inputs) is close to 0.25, and therefore the bias of the entire Approximation 1 is 0.25α, for some α that depends on the other parts of the approximation.

If we could choose only plaintexts of cases 00 and 11 and run the attack only on these plaintexts, we would need fewer messages due to the larger bias. Unfortunately, the values of $w_{1,3}$ and $w_{2,3}$ are only known up to a XOR with two missing bits of $DK6$ (see Fig. 2):

$$w_{1,3} = f_{0,3} \oplus f_{1,3} \oplus DK6_{1,3}, \qquad w_{2,3} = f_{2,3} \oplus f_{3,3} \oplus DK6_{2,3},$$

and therefore they clearly cannot be chosen or known directly. Nevertheless, the corresponding bits $f_{0,3}$, $f_{1,3}$, $f_{2,3}$ and $f_{3,3}$ in the input of the F-function are all known as a result of the partial guess of the actual subkey $DK7$. We observe that we can still partition all the data into the same four sets according to $f_{0,3} \oplus f_{1,3}$ and $f_{2,3} \oplus f_{3,3}$, instead of $w_{1,3}$ and $w_{2,3}$, but we do not know which of the four sets have the amplified biases.

Though we cannot identify the two sets with an amplified bias, we can run this inner part of the attack four times, once on each of the sets. We expect the following results: In each set we would have about a quarter of the known

plaintexts but in two of them we would have a bias twice as large as we had originally (meaning $\pm 0.5\alpha$).[4] Therefore the number of required plaintexts in these sets is about 4 times ($0.5^2/0.25^2$) smaller than would have been needed without applying this technique.

A more careful analysis shows that we can merge the two sets with bias ± 0.5 (with an appropriate sign coefficient) and partition the plaintexts only to two sets. This merges the sets of cases 00 and 11 into one set, and the sets of cases 01 and 10 into another set according to the parity of the two bits $f_{0,3} \oplus f_{1,3}$ and $f_{2,3} \oplus f_{3,3}$. Denote the number of known plaintexts required for the original attack by m. As discussed above, the amplified bias can be detected with $m/4$ plaintexts. Since each of the two unified sets has about half of the plaintexts, we deduce that $m/2$ known plaintexts suffice for the partitioning technique.

4.2 The Attack

The attack follows the lines of the above example, but considers that the details of the approximation of the S-boxes are more complicated than described so far. While for a single S-box and appropriate independence assumptions the technique would work as described, in practice there is a correlation between the approximation of the two middle S-boxes of F. We give the combination of both middle S-boxes the name *T-box* (marked by a rectangle in Fig. 3). The joint approximation of the two S-boxes in the T-box cannot be described as a combination of two independent approximations since the input bits to the second S-box are all either inputs of the first S-box or its output. Therefore, a closer examination of the joint distribution is in order.

We computed the joint approximation of the S-boxes (the T-box) with the approximation `11 55` $\rightarrow$ `02 11` and observed that the partition to two sets (by the value of $f_{0,3} \oplus f_{1,3} \oplus f_{2,3} \oplus f_{3,3}$) has the following effect: In the cases 01 and 10 the bias is increased by a factor of about 2.49 compared to the original bias, while the absolute value of the bias in the other cases (00 and 11) is halved. It is therefore that the number of known plaintexts needed by the attack is reduced by a factor of about $2.49^2/2 \approx 3.1$.

We also note that there are other possible partitions (by other control bits) that yield an increased bias in one or more of the sets, that can be used for alternative implementations of this technique.

We applied this improvement to the attack of Sect. 3 and successfully reduced the number of required known plaintexts from 2^{15} to 2^{14}. Applying this technique did not add a noticeable overhead to the running time of the attack. In fact, the time it took to recover the 44 bits of the actual subkeys using 2^{14} plaintexts was 12 h – which is about half the time that was required using 2^{15} plaintexts (without using this technique). The rest of the attack took about two more hours, and the key was found after 14 h of computation. The key that was found for the challenge with 2^{14} plaintexts is `5681891EEC34CE1241ED0F52C9C23F65`.

[4] For the purpose of this simplified example we assume that the linear approximation of this S-box is independent of the rest of Approximation 1. We will see later that this is not the case.

5 Attacking FEAL-8X Using 2^{10} Known Plaintexts with Complexity 2^{62}

The methods we described in the previous sections can be used to break FEAL-8X with even fewer known plaintexts in time which is still faster than exhaustive search. In particular, the key can be found given 2^{10} known plaintexts in time of about 2^{62} FEAL encryptions.

To justify the above claim, we describe an attack on seven rounds of FEAL, which is based on the attack of Sect. 3.2, and then extend it to 8 rounds by exhaustively searching for the subkey of the last round.

The attack on seven rounds of FEAL uses the first five rounds of Approximation 1, with a bias of 2^{-3}. Similarly to the attack of Sect. 3.2, the approximation covers the five middle rounds and the analysis is performed on the first and last rounds. In each of the first and last rounds there are 15 bits that we need to guess in order to compute the parity of the linear approximation, and therefore the attack requires encrypting/decrypting an equivalent of $2^{15} \cdot 2^{15} \cdot 2^{10} \cdot 2 = 2^{41}$ rounds of FEAL.

In order to extend the attack to eight rounds, we also guess 30 bits of the actual subkey $DK7$ of the last round (recall that two of the 32 bits have no effect on the parity of the linear approximation). For each candidate for these 30 bits of $DK7$ we decrypt the last round of all the inputs, and then apply the above attack to the remaining seven rounds. The attack on seven rounds is performed 2^{30} times, and therefore the total time complexity is equivalent to computing $2^{30} \cdot 2^{41} = 2^{71}$ rounds of FEAL (or 2^{68} encryptions of the full cipher), which is much faster than exhaustively searching for the 128-bit key.

When applying the optimization improvements described in Appendix A we get an even lower complexity of about 2^{62} FEAL encryptions.[5]

6 Attacks with a Few Known or Chosen Plaintexts

In this section we describe several attacks that require only a few (even 2 or 3) known or chosen plaintexts, which are based on linear cryptanalysis or differential cryptanalysis combined with exhaustive search of most subkeys, as well as meet in the middle attacks.

6.1 Differential and Linear Exhaustive Search Attacks

During the work on this paper we noticed that the actual subkeys of FEAL-8X are mixed very slowly through the encryption function. In particular, we observed that only 112 bits of the actual subkeys are needed in order to decrypt a ciphertext by 5 rounds and compute the data after the third round of the cipher from the ciphertext. In addition, we recalled that there are four independent

[5] Recall that the key size of FEAL-8X is 128 bit.

3-round linear approximations with probability 1 (creating a total of 15 non-trivial approximations) and two independent 3-round differential characteristics with probability 1 (creating a total of 3 characteristics). These approximations and characteristics can be found in [2,4].

In the case of the linear approximations with probability 1, each allows us to test one parity bit of the data after the third round and to compare to a parity bit of the plaintext. Therefore, a total of 4 bits can be tested on each plaintext (except for the first known plaintext to whose parities we compare). Given 5 known plaintexts the attack would be:

1. For each value of the set of subkeys $DK3$, $DK4$, $DK5$, $DK6$, $DK7$ (in total these 160 bits only contain 112 independent bits).
   (a) For each plaintext-ciphertext pair (P, C) decrypt the ciphertext by 5 rounds to D_3 and compute the parity of each approximation $P\lambda_P^3 \oplus D_3\lambda_T^3$, where $\lambda_P^3 \rightarrow \lambda_T^3$ is the mask of the linear approximation in use.
   (b) Discard any guess for which the five results (each of 4 bits, one for each approximation) are not the same.
   (c) Note that at this point only about 2^{96} of the guesses of the subkeys remain.
   (d) For each value of the subkey $DK2$ (16 more bits)
      i. Note that at this point we have about 2^{112} guesses of the subkeys.
      ii. We will now use four 2-round approximations $\lambda_P^2 \rightarrow \lambda_T^2$ which are based on the last two rounds of the prior ones.
      iii. For each plaintext-ciphertext pair (P, C) decrypt the ciphertext by 6 rounds to D_2 and compute the parity of each approximation $P\lambda_P^2 \oplus D_2\lambda_T^2$.
      iv. Discard any guess for which the five results are not the same.
      v. Note that at this point we are left again with only about 2^{96} guesses of the subkeys.
      vi. For each value of the subkey $DK1$ (16 more bits)
         A. Note that at this point we have about 2^{112} guesses of the subkeys.
         B. For each plaintext-ciphertext pair (P, C) decrypt the ciphertext by 7 rounds to D_1 and compute the XOR of both halves of the whitening key $DK89 \oplus DKab$ (32 bits in total).
         C. Discard any guess for which the five results are not the same.
         D. Note that at this point we expect that only the correct values of all the above guesses remain.
         E. Complete the rest of the subkeys by guessing $DK0$ and comparing the resulting $DK89$ in 2^{16} time.
         F. Recover the original key. The algorithm is described in the full version of the paper (note that given all the decryption actual subkeys and the whitening key it is easy to compute the encryption actual subkeys needed by that algorithm).

The complexity of this attack is 2^{112}, taking into consideration that the various decryptions need not be computed several times (once by 5 rounds, then by 6,

then by 7), but that the intermediate values can be cached to save computation time. A careful implementation would require an average computation of only two rounds in each guess for each of the three guessing loops. Thus the total complexity is about $3 \cdot 2 \cdot 2^{112}$ round computations $= 0.75 \cdot 2^{112}$ encryption of FEAL-8X.

A similar attack that uses the 3-round differential characteristics with probability 1 requires only three chosen plaintexts (whose plaintexts differ by the two plaintext differences of the two characteristics). Since each differential characteristic predicts 64 bits of the intermediate difference, we have a much better elimination of wrong guesses, and thus we need only three chosen plaintexts. The complexity of the attack is also 2^{112}.

6.2 Meet in the Middle Attacks

The attack that requires the least number of known plaintexts is a meet in the middle attack. We observe that the number of (independent) bits of the actual subkeys that are required to partially encrypt (or decrypt) four rounds of the cipher is 96. Therefore, a meet in the middle attack using two (or three) known plaintexts computes 2^{96} 4-round partial encryptions of two blocks plus 2^{96} 4-round partial decryptions of two blocks. This attack also requires 2^{96} memory words of size 128 bits (or even 96 bits). The list of about 2^{64} (or 2^{96}) colliding values should then be checked by auxiliary techniques, and be completed to a full key with the same known plaintexts.

An improvement of this attack may reduce the complexity to 2^{80}, by encrypting or decrypting only three rounds from each end, using 11 known plaintexts. This improvement considers that the F-function in the fourth round can be approximated by the four independent linear approximations with probability 1 (each one is represented by a single parity bit in the output of encryption and a single parity bit in the output of decryption). The fifth round can be approximated similarly. This way, each known plaintext contributes 8 bits to the colliding values (except for the first, whose 8 parity bits are XORed into the parity bits of all the other ones), and thus in order to collide on 80 bits, we need 11 known plaintexts. Each of the 2^{80} colliding values can then be checked by auxiliary techniques, and be completed to the full key.

We also note that these meet in the middles attacks can be transformed to memoryless meet in the middle attacks by standard techniques [10,12]. The simplest implementation of the former encrypts/decrypts three blocks at a time, each encrypted or decrypted by four rounds, resulting in a collision on 192 intermediate data bits, which ensures that the real value of the subkeys are easily identified in time 2^{96}. The simplest implementation of the latter encrypts/decrypts 21 blocks at a time, each encrypted or decrypted by three rounds, resulting in a collision on 160 intermediate data bits, which ensures that the real value of the subkeys be easily identified in time $21 \cdot (3/8 + 3/8) \cdot 2^{80} \approx 2^{84}$.

7 Summary

We presented the techniques which allowed us to break FEAL-8X with only 2^{14} known plaintexts and recover the secret key. This is an improvement of the best known-plaintext attacks prior to this paper. Our attack is based on a few improvements and optimizations to linear cryptanalysis, the most important of which is the new partitioning technique which allowed us to reduce the amount of known plaintexts needed for the attack.

In addition to the practical attacks on FEAL-8X we also presented a few attacks which are based on linear and differential cryptanalysis in combination with meet-in-the-middle techniques. Those attacks can find the secret key given only a few messages in time which is faster than exhaustive search.

We wish to discuss the similarities and differences between our partitioning technique and partitioning cryptanalysis [5]. They both partition the data into several sets based on functions that take the plaintexts or ciphertexts and guessed key bits, where each set of the input-partition is related to some linear approximation and expected biases. In that sense, our technique is a variant of partitioning cryptanalysis. However, in partitioning cryptanalysis the expected biases are known in advance for each input block of the partition, and thus the attacker can select the best block and choose all the *chosen plaintexts* to be in that block. In our case we succeed (in the particular case of the addition operation) to take one step further and divide to partitions such that we do not know which set should have which bias. The identification of the sets is part of the attack, and it is therefore that our technique is a *known plaintext attack*. But perhaps the most significant improvement of our technique stems directly from the motivation that is the basis of our partition – we use the partition in order to discard (or rather ignore) messages that do not contribute to the linear bias. By doing so the bias in the remaining set is higher, which allows us to reduce the number of messages needed for the attack. We also note that our technique may in some cases be applied both on the plaintext side and on the ciphertext side simultaneously, and gain the extra factor in cases that partitioning cryptanalysis may not.

Acknowledgements. The authors would like to thank Mitsuru Matsui for initiating the FEAL 25 Years challenge. We would also like to thank Orr Dunkelman for his insightful comments and helpful suggestions.

A Efficient Implementation

We describe an optimization to the implementation of the attack of Sect. 3.2 which saves a factor of about 2^6 in the computation time of the attack. This optimization can also be applied to other attacks presented in this paper that are based on the attack of Sect. 3.2.

Recall that in the attack of Sect. 3.2 we iterate over 2^{15} possible values for (16 bits of) the encryption actual subkey of the first round ($EK0$), and 2^{22} possible values for (22 bits of) the decryption actual subkey of the last round ($DK7$). For each of the 2^{37} combinations, two rounds of FEAL are encrypted/decrypted for each known plaintext. We denote the number of known plaintexts by m.

We observe that given a known plaintext-ciphertext pair P, C, the parity of the approximated bits can be written as $b_P \oplus b_C$, where b_P is a bit that depends only on the plaintext and the actual subkey of the first round, and b_C is a bit that depends only on the ciphertext and the actual subkey of the last round. Therefore, we can change the attack as follows:

1. For each of the 2^{15} candidates for the 16 bits of $EK0$:
   (a) Compute a vector B_P of length D bits, where $(B_P)_i = b_{P_i}$. Save all the vectors in a table.
2. For each of the 2^{22} candidates for the 22 bits of $DK7$:
   (a) Compute a vector B_C of length m bits, where $(B_C)_i = b_{C_i}$.
   (b) For each of the 2^{15} candidates for the 16 bits of $EK0$, get the vector B_P from the table, and compute the number of plaintexts for which the parity of the Approximations is 1 by $H(B_P \oplus B_C)$, where H is the Hamming weight function.
   (c) Compute the bias for approximation.
3. The correct key is expected to be the one with the highest bias.

Assuming a processor with a word size of 64 bits, this optimization lets us compute the parity of 64 plaintexts at the same time, and therefore saves a factor of about 2^6 in the attack.

We note that this optimized implementation also works with the partitioning technique described in Sect. 4. In Step 2a of the algorithm above, in addition to generating the vector B_C we generate a third vector W. The i-th bit of W determines to which set of the partition the i-th plaintext belongs. We can compute the number of plaintexts with a parity of 1 in the bits of the approximation in each of the sets as $H((B_P \oplus B_C)\&W)$ and $H((B_P \oplus B_C)\&\overline{W})$, where & is the bitwise-and operator, and $\overline{W}$ denotes the binary complement of W.

B The Linear Approximations Used in Our Attacks

The appendix lists the two linear approximations from [1,11] which we use in our attacks. Approximation 1 is presented in Fig. 4 and Approximation 2 is in Fig. 5.

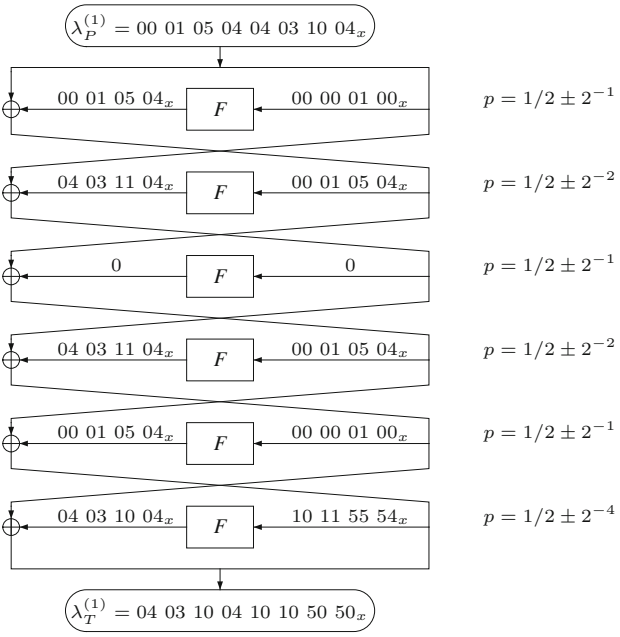

Fig. 4. Approximation 1 – A six round approximation with bias 2^{-6}

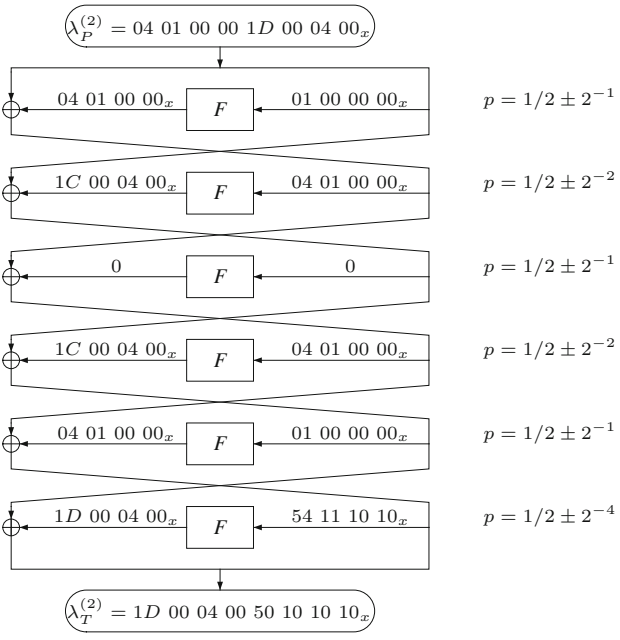

Fig. 5. Approximation 2 – A six round approximation with a bias 2^{-6}

References

1. Aoki, K., Ohta, K., Moriai, S., Matsui, M.: Linear cryptanalysis of FEAL. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E81–A**(1), 88–97 (1998)
2. Biham, E.: On matsui's linear cryptanalysis. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 341–355. Springer, Heidelberg (1995)
3. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cyptosystems. J. Cryptol. **4**(1), 3–72 (1991)
4. Biham, E., Shamir, A.: Differential cryptanalysis of feal and N-Hash. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 1–16. Springer, Heidelberg (1991)
5. Harpes, C., Massey, J.L.: Partitioning cryptanalysis. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 13–27. Springer, Heidelberg (1997)
6. Matsui, M.: Celebrating the 25th year of FEAL - A new prize problem, rump session of CRYPTO'12. http://crypto.2012.rump.cr.yp.to/19997d5a295baee62c05ba73534745ef.pdf
7. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
8. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 81–91. Springer, Heidelberg (1993)
9. Miyaguchi, S.: News on FEAL Cipher, talk at the rump session at CRYPTO'90 (1990)
10. Morita, H., Ohta, K., Miyaguchi, S.: A switching closure test to analyze cryptosystems. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 183–193. Springer, Heidelberg (1992)
11. Ohta, K., Aoki, K.: Linear cryptanalysis of fast data encipherment algorithm. Technical Report of IEICE (1994)
12. Quisquater, J.-J., Delescaille, J.-P.: How easy is collision search. New results and applications to DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 408–413. Springer, Heidelberg (1990)
13. Shimizu, A., Miyaguchi, S.: Fast data encipherment algorithm FEAL. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 267–278. Springer, Heidelberg (1988)