IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2018, No. 3, pp. 215–264. DOI:10.13154/tosc.v2018.i3.215-264

Conditional Linear Cryptanalysis – Cryptanalysis of DES with Less Than 2^{42} Complexity

Eli Biham and Stav Perle

Computer Science Department Technion – Israel Institute of Technology, Haifa, Israel
{biham,stavp}@cs.technion.ac.il

Abstract. In this paper we introduce a new extension of linear cryptanalysis that may reduce the complexity of attacks by conditioning linear approximations on other linear approximations. We show that the bias of some linear approximations may increase under such conditions, so that after discarding the known plaintexts that do not satisfy the conditions, the bias of the remaining known plaintexts increases. We show that this extension can lead to improvements of attacks, which may require fewer known plaintexts and time of analysis. We present several types of such conditions, including one that is especially useful for the analysis of Feistel ciphers. We exemplify the usage of such conditions for attacks by a careful application of our extension to Matsui's attack on the full 16-round DES, which succeeds to reduce the complexity of the best attack on DES to less than 2^{42}. We programmed a test implementation of our attack and verified our claimed results with a large number of runs. We also introduce a new type of approximations, to which we call scattered approximations, and discuss its applications.

Keywords: new cryptanalytic techniques · linear cryptanalysis · DES · conditional approximations · scattered approximations

1 Introduction

In this paper we present a new extension of linear cryptanalysis that uses conditions to discard some of the data, and increase the biases of the remaining data. This technique works best on Feistel ciphers [MvOV01, Definition 7.81]. We exemplify our technique by implementing it against DES, which is the most well known and most widely analyzed cipher of this kind. We also present other extensions and applications of our techniques.

Linear cryptanalysis was introduced by Mitsuru Matsui, who first applied the technique to FEAL [SM87, MY92], and later generalized it to an attack against DES [Nat77] using 2^{43} known plaintexts [Mat93, Mat94]. It statistically analyzes linear relations between plaintexts bits, ciphertexts bits and key bits, which in turn are used to predict the parity of a subset of key bits. Auxiliary analysis techniques can find additional key bits, and complete the attack to find the full key.

1.1 The Data Encryption Standard

The Data Encryption Standard (DES) [Nat77] was developed in the early 1970's by IBM, and adopted by the U.S. National Bureau of Standards as the standard encryption algorithm for commercial and sensitive but unclassified data in 1977.

DES consists of 16 rounds of permutations and arithmetic operations. The block size of DES is 64 bits, and the key size is 56 bits. The main part of the round function is the

Received: 2017-11-23, Revised: 2018-06-01, Accepted: 2018-08-01, Published: 2018-09-04

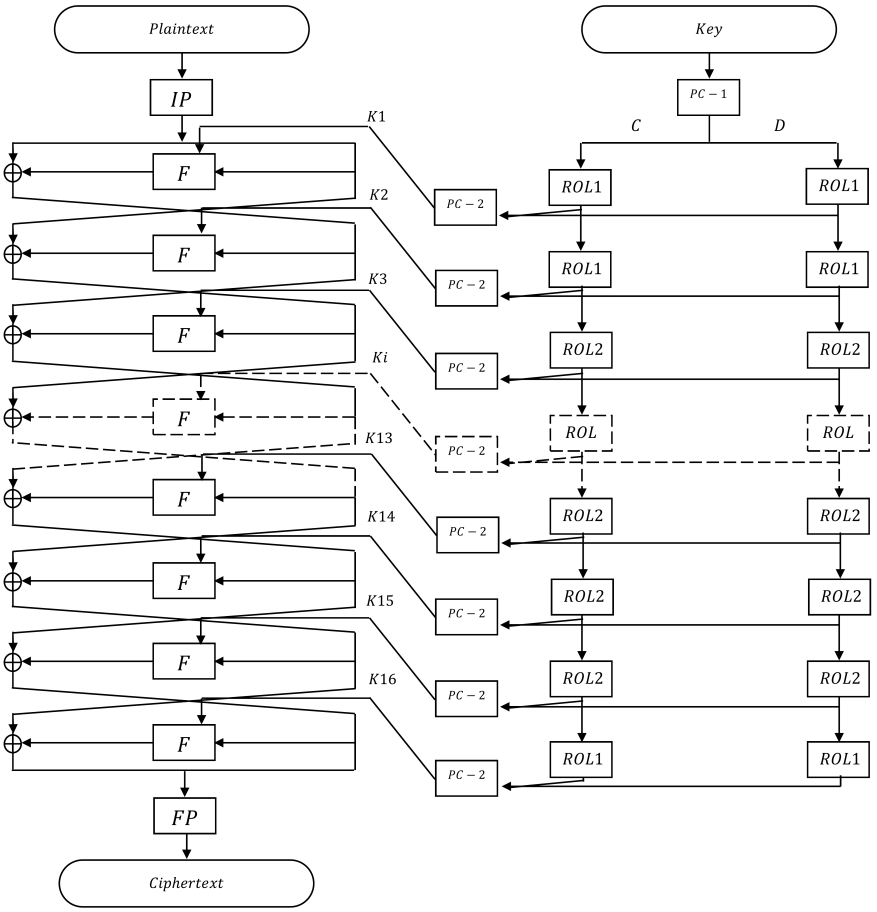

Figure 1: Outline of The Data Encryption Standard

F function, which mixes the right half of the data and a 48-bit round subkey. The F function expands the input to 48 bits, XORs them with the subkey, and transforms the result by eight 6-bit to 4-bit S boxes. Then, the order of the resulting 32 bits is permuted to become the 32 output bits of the F function. This output is then XORed with the left half of the data, and the two halves are swapped before the next round. Outlines of DES and its F function are given in Figure 1 and Figure 2. For more details on DES see [BS93] or [Nat77].

1.2 Linear Cryptanalysis

Linear cryptanalysis [Mat93, Mat94] uses statistical approximations that approximate parity of subsets of bits of the plaintext, ciphertext, and the subkeys. In an ideal cipher, any linear equation involving plaintext bits, ciphertext bits and key bits would hold with probability 1/2 when considering all the plaintexts in the plaintext space, and their corresponding ciphertexts.

Linear cryptanalysis takes advantage of the fact that some linear approximations have probabilities different than 1/2. Each approximation has a probability p to hold, which is the fraction of plaintexts whose encryption satisfies the approximation. The ability to distinguish whether an approximation holds highly depends on the distance of the probability from 1/2. We call this distance the *bias* of the approximation. The bias ε of linear approximation with probability p is $\varepsilon = p - 1/2$.

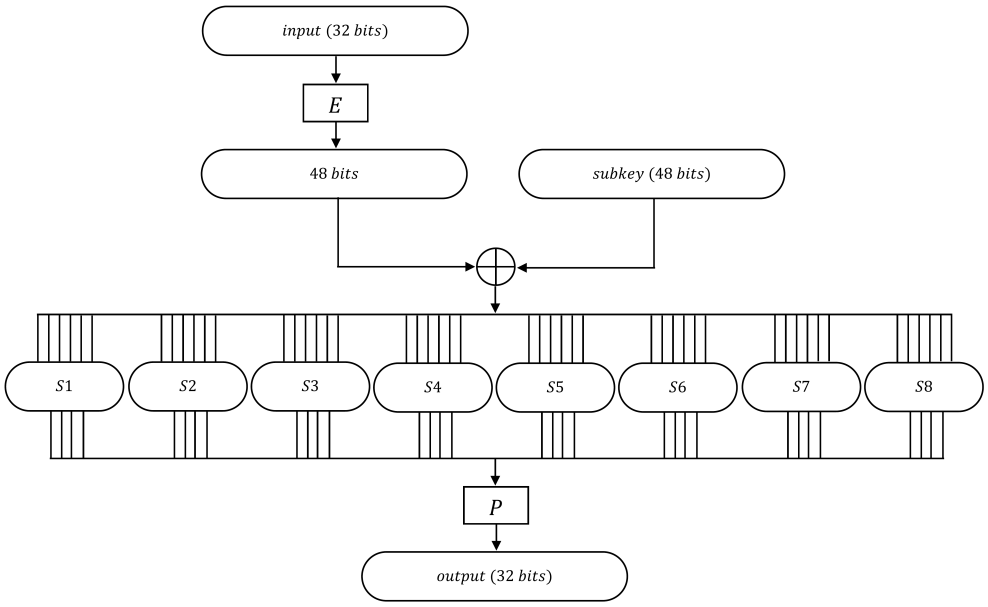

Figure 2: The F function

In DES, since the S boxes are the only non-linear component, the attack is based on statistical linear relations of the S boxes. For each S box, these relations are determined by choosing a subset of the input bits, a subset of the output bits, and by calculating the parity of these bits for each of the possible inputs of the S box. The probability of such an approximation is the fraction of inputs whose subset's parity is equal to 0.

1.3 Other Related Prior Work

Differential cryptanalysis [BS93] analyzes how differences evolve during encryption, and how differences of plaintext pairs evolve to differences of the resultant ciphertext pairs. Their study allows to assign probabilities to the possible keys and to locate the most probable key. This method usually works on many pairs of plaintexts with the same particular difference. Differential cryptanalysis was introduced by Eli Biham and Adi Shamir in the late 1980's. They published a number of attacks against various block ciphers and hash functions, including DES. The most well known attack of this kind can break the full 16-round DES in 2^{37} time complexity given 2^{47} chosen plaintexts. Although on the detailed level this method is quite different from linear cryptanalysis, on the structural level they are very similar [Bih94].

Differential and linear cryptanalysis are two major generic techniques for assessing the strength and vulnerabilities of block ciphers. These techniques have various extensions which can improve their success in various cases. Along with the Improved Davies' attack [BB94], they form all the known shortcut techniques against the Data Encryption Standard.

Other newer publications propose extensions to these shortcut techniques. Chosen plaintext linear cryptanalysis was proposed in [KM00]. The time complexity and the number of required known plaintexts for finding 12 key bits is 2^{42}. A series of papers extended linear cryptanalysis using many linear approximations, with various techniques. For example, multiple linear cryptanalysis [BCQ04] which considers statistically independent linear approximations, and multidimensional linear cryptanalysis [HCN09] which analyze the distribution of events spanned by dependent linear approximations.

A statistical extension of linear cryptanalysis was proposed in [Vau96] which showed how to improve the attack of Matsui by a small factor of $\approx 2^{0.07}$. An improved algorithm which is based on quadratic relations of S boxes was proposed in [SK98]. This algorithm is a combination of a non-linear approximation method and multiple approximation method. This improvement can reduce the number of required known plaintexts to $2^{42.55}$.

Finally, a recent paper [BV17] improves over linear cryptanalysis with multiple approximations using a large number of approximations with a more thorough analysis. Their actual attack against DES has complexity $2^{38.86}$ with $2^{42.78}$ known plaintexts or alternatively $2^{49.76}$ with 2^{41} known plaintexts. Unfortunately, their paper does not contain enough information to apply their actual attack and to directly compare it with our results, or even to merge their ideas with ours to gain an even better attack.

Several papers partitioned the data into several sets based on some function of the plaintext and potentially some guessed key bits, or even were able to potentially improve the probability of differential characteristics or the bias of linear approximations based on partition of the unknown keys to several sets (without knowing in advance to which set the unknown key belongs). In [BC14] Biham and Carmeli used a partition on data gained during the analysis. In terms of the particular rounds where the partition is computed, they took the plaintexts and partially encrypted them by two rounds under guessed key bits in order to get the bit by which they partitioned. Ben Aroya and Biham [BB93] showed that the probability of differential characteristics may depend on unknown key bits and used this fact to improve attacks. Biham and Shamir [BS93, Section 4.4.5, Table 4.8] described a similar case.

In addition to cryptanalytic extensions, some papers suggested algorithmic improvements that make the analysis more efficient. The most important one of this kind (which is highly useful for our results) is [CSQ07], which shows how to optimize the counting scheme and key recovery in linear attacks using FFT.

1.4 Our Results

In this paper, we develop an extension of linear cryptanalysis that conditions linear approximations on other linear approximations. Under such conditions, the bias of some linear approximations may increase or decrease. We call such a bias a *conditional bias*.

We show that our new extension can reduce the time complexity of the attack against the Data Encryption Standard further below any previously known attack. We use conditions to discard some of the data (or skip computing based on some data depending on partial encryption or decryption during our computation), so the bias of the remaining data increases or decreases. Conditions can be defined by any observable information available: plaintext bits, ciphertext bits, partially encrypted or decrypted data, and any formulae on them. Examples for conditions include 1) the XOR of the outputs of the F function in all odd rounds (or all even rounds), which are computable from the plaintexts and ciphertexts, 2) inputs of the F function in the first and last rounds, or 3) validity of other linear approximations. Conditions may be by a single parity bit of the observable information, or by several, including by a distribution of data consisting of several bits, or selections of several cases from such a distribution.

Using this extension we succeed to improve the bias of many linear approximations, including of 13-round, 14-round and 16-round linear approximations. For applying the attack against the full 16-round DES we use two conditional linear approximations and an improved algorithm of analysis. This results with the best current attack against DES, with complexity $2^{41.9}$ known plaintexts and time of analysis. See Table 1 and Table 2 for comparison of our results with previously published attacks. Table 1 considers the standard model where the time of analysis is smaller or equal to the time of generating the known plaintexts. Table 2 assumes a model with a larger time of analysis, which we fixed at 2^{50}, and optimizes the number of known plaintexts to even lower numbers. This latter

Table 1: Complexity of Attacks Against DES: Time of Analysis $\leq$ # Known Plaintexts

Attack	Complexity Data	Complexity Time	Success Prob.	Attack Scenario
Differential Cryptanalysis [BS93]	$2^{47.00}$	$2^{37.00}$	58%	Chosen Plaintext
Linear Cryptanalysis [Mat94]	$2^{43.00}$	$2^{43.00}$	85%	Known Plaintext
Linear Cryptanalysis [Mat94, Jun01]	$2^{43.00}$	$2^{39.00}$	50%	Known Plaintext
Improved Davies' Attack [BB94]	$2^{50.00}$	$2^{50.00}$	51%	Known Plaintext
Statistical Cryptanalysis [Vau96]	$2^{42.93}$	$2^{42.86}$	?*1	Known Plaintext
Quadratic Relations [SK98]	$2^{42.55}$	$2^{42.55}$	84%*2	Known Plaintext
Multiple Linear Cryptanalysis [BV17]	$2^{42.78}$	$2^{38.86}$	85%	Known Plaintext
Conditional Linear Cryptanalysis (this paper)[*3]	$\mathbf{2^{43.00}}$	$\mathbf{2^{38.00}}$	**100%**	Known Plaintext
	$\mathbf{2^{42.00}}$	$\mathbf{2^{42.00}}$	**90%**[*4]	Known Plaintext
	$\mathbf{2^{41.90}}$	$\mathbf{2^{41.90}}$	**85%**	Known Plaintext

*1 The success rate is not mentioned in the paper.
*2 Interpolated from Table 2 of [SK98].
*3 Each success probability is based on several thousands of runs of an implementation of our attack (based on 8-round reductions).
*4 Also verified experimentally on the full DES.

Table 2: Complexity of Attacks Against DES: Time of Analysis $\leq 2^{50}$

Attack	Complexity Data	Complexity Time	Success Prob.	Attack Scenario
Linear Cryptanalysis [Mat94]	$2^{38.00}$	$2^{50.00}$	10%	Known Plaintext
Statistical Cryptanalysis [Vau96]	$2^{42.00}$	$2^{48.00}$	?*1	Known Plaintext
CP Linear Cryptanalysis [KM00]	$2^{42.00}$	$2^{44.00}$	86%*2	Chosen Plaintext
Multiple Linear Cryptanalysis [BCQ04]	$2^{41.00}$*3	$2^{43.00}$	?*1	Known Plaintext
Multiple Linear Cryptanalysis [BV17]	$2^{41.00}$	$2^{49.76}$	85%	Known Plaintext
Conditional Linear Cryptanalysis (this paper)[*4]	$\mathbf{2^{42.00}}$	$\mathbf{2^{50.00}}$	**100%**[*5]	Known Plaintext
	$\mathbf{2^{41.00}}$	$\mathbf{2^{50.00}}$	**92%**	Known Plaintext
	$\mathbf{2^{40.00}}$	$\mathbf{2^{50.00}}$	**64%**	Known Plaintext
	$\mathbf{2^{39.00}}$	$\mathbf{2^{50.00}}$	**36%**[*5]	Known Plaintext
	$\mathbf{2^{38.00}}$	$\mathbf{2^{50.00}}$	**17%**	Known Plaintext
	$\mathbf{2^{37.00}}$	$\mathbf{2^{50.00}}$	**12%**[*6]	Known Plaintext

*1 The success rate is not mentioned in the paper.
*2 2^{42} time complexity to find 12 key bits as in [KM00], and then finding the rest of the key in 2^{44} time complexity.
*3 Conjectured in [BCQ04].
*4 Each success probability is based on several thousands of runs of an implementation of our attack (based on 8-round reductions).
*5 Also verified experimentally on the full DES.
*6 The time complexity per found key is about 2^{53}.

model is addressing the fact that the known plaintext generation phase on the victim's machine is practically more difficult than a computation of the same complexity on the attacker's own machines. It is therefore, that our attack is still attractive for finding DES keys even with as low as 2^{37} known plaintexts. We verified our results experimentally with our test programs.

We use the FFT method [CSQ07] in our attack to speed-up computations, and thus for being able to increase the number of guessed key bits during the attack. We extend the FFT method, and duplicate plaintext and ciphertext bits that are XORed with two different key bits during encryption. In addition, our extension allows us to guess key bits in the second round, in addition to the first and last rounds, although these bits are non-linear in the plaintext and the ciphertext. We also find it necessary to introduce new "virtual" key bits to the cipher to allow us to represent the condition of the approximation with the FFT representation.

We also discuss the applicability of the conditioning technique to differential cryptanalysis, and how conditional approximations can be constructed. In an appendix we introduce a new kind of approximations that contain intermediate bits in the parity subsets, to which we call scattered approximations, and a particularly interesting scattered conditional approximations in which a short approximation is conditioned on a longer containing approximation. We discuss the application of scattered approximations for constructing new approximations and for attacks.

1.5 Structure of the Paper

This paper is organized as follows: Section 2 introduces the notion of conditional linear approximations and conditional biases, introduces how conditional linear attacks are structured, and introduces the notations used throughout this paper. Section 3 presents examples of conditional linear approximations of round-reduced DES, and some simple attacks. Section 4 describes our full attack on 16-round DES as well as our computer implementation, practical experiments, and success rates. Section 5 discusses strategies for finding good conditional linear approximations. Section 6 discusses the applications of our ideas to differential cryptanalysis. Section 7 discuss the relation between conditional linear cryptanalysis, linear cryptanalysis with multiple approximations, and multidimensional linear cryptanalysis. Finally, Section 8 summarizes this paper. Appendix A presents scattered approximations and discuss their applications. The proof of the conditional biases piling up lemma is given in Appendix B.

2 Introduction to Conditional Linear Cryptanalysis of Feistel Ciphers

In this section we introduce conditional linear cryptanalysis, and the related notions of conditional linear approximations and conditional biases. We also describe the most useful linear approximations, used throughout this paper to exemplify our methods. But note that we do not describe here the most general kind of conditional linear approximations, and indeed a different kind of conditional linear approximations is described in Section A.1. In this section we show that the bias of certain linear approximations may grow (or reduce) depending on some XOR bits whose values we know for certain. It is, therefore, advantageous to take these extended linear approximations into consideration when designing a linear attack.

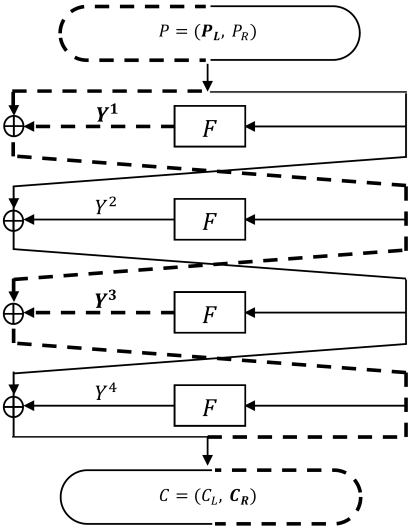

Figure 3: The XOR of the Odd Rounds in DES Reduced to Four Rounds, And its Equality to $P_L \oplus C_R$

2.1 The Most Useful Conditional Linear Approximations

We observe that in Feistel ciphers [MvOV01, Definition 7.81] we can compute various linear combinations of internal bits directly from the plaintext and ciphertext by applying the XOR operation on subsets of bits of the plaintext and the ciphertext. Notice that such a subset may be viewed as another linear approximation. Our main observation is that the bias of a linear approximation may be dependent on another linear approximation (e.g., computed by these XOR bits). In the case of Feistel ciphers we found a specific type of conditions that is highly based on their specific structure. In particular, we refer to bits that are the XOR of the outputs of the F function of the odd rounds (or similarly of the even rounds). Figure 3 outlines the XOR of the odd rounds in DES reduced to four rounds, and its equality to the XOR of the plaintext and ciphertext halves, i.e., $P_L \oplus C_R = Y^1 \oplus Y^3$. It is important to emphasize that this information on the XOR values is not probabilistic, unlike the situation in general linear approximations, i.e., these XOR values are certainly correct. We can consider any one of the XOR bits in $P_L \oplus C_R$ as a linear approximation. Such XOR bits, when viewed as linear approximations, typically have bias 0, or close to 0, but they are very useful as conditions to other approximations.

We start with an example of such a linear dependency with a single active S box. Consider the S box $S5$ of DES. The best linear approximation of $S5$: $10_x \rightarrow F_x$ (where 10_x and F_x are the masks of the subsets of bits in the input and output of $S5$) is outlined in Figure 4. We denote this linear approximation by λ_1. It approximates the second bit of input to the XOR of the four output bits. The bias of this linear approximation is $-20/64 = 12/64 - 1/2$, as in 12 cases of the 64 possible inputs the results are the same, while in 52 they are different.

We observe the following important and unexpected observation: only one of the 12 "same" cases satisfies LSB=1, while the other 11 satisfy LSB=0. Consider only encryptions in which the least significant output bit of the S box is 1. In this case we use only about half of the data, i.e., the data in which the LSB is 1. In this half of the data the bias grows from $-20/64$ to $-15/32 = -30/64$ (this bias represents a single case with parity 0, and 31 with parity 1), while the bias of the rest of the data (the discarded data whose LSB is 0) reduces to $-5/32 = -10/64$.

We explain our observation using Figure 5 that presents a scan from Adi Shamir's

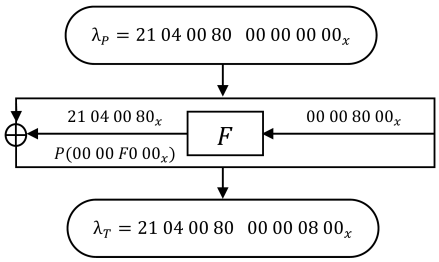

Figure 4: The Best Single-Round Linear Approximation of DES with Bias $-20/64$ (λ_1). The subsets of bits participating in the parities are denoted as bit masks.

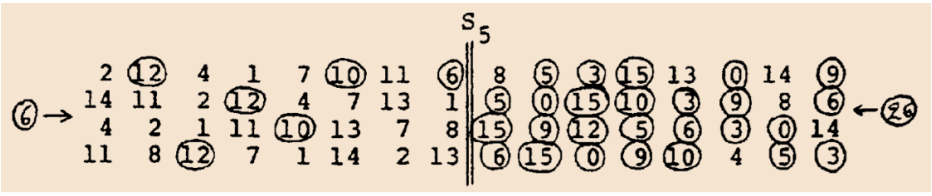

Figure 5: The 12 vs. 52 in $S5$ — Scan from Adi Shamir's CRYPTO'85 Paper

CRYPTO'85 paper [Sha85] that presented the correlation between the second input bit and the parity of the output bits for the first time. On the left side of the vertical line appear all the outputs whose second bit of input is 0, and on the right side of the line all the outputs whose second bit of input is 1. Shamir circled the values with an even parity of the four output bits. Therefore, in the left side he circled the outputs which satisfy our conditional parity approximation, and in the right side he circled the outputs which do not satisfy our conditional parity approximation. Notice that there are 32 outputs with LSB 1. As we can see, in the left side all the odd numbers (outputs with LSB 1) are not circled, and in the right side all the odd numbers, except one, are circled. This means that in case the LSB is 1, only one out of the 32 cases satisfies the parity approximation. Table 3 shows the effect of this condition on the LSB of the output of $S5$.

When applying this observation to linear approximations with more than one round, we cannot condition on an output bit of $S5$ in a single round, because we cannot know such a value. Instead, we condition the approximation on the XOR of the LSB of the output of $S5$ in all the odd rounds (or similarly even rounds). We can gain increased biases even in such a case. A detailed example will be given. As we show later, the bias may grow by a factor of about two (or even more) in some cases. Table 4 presents the biases and conditional biases of various conditional linear approximations that we discuss in this paper, along with the factor of improvement.

Table 3: Condition on the LSB of the Four Output Bits of $S5$

Condition	λ_1 holds	λ_1 fails	Bias
none (not conditioned)	12	52	$-\frac{20}{64}$
LSB=0	11	21	$-\frac{5}{32} = -\frac{10}{64}$
LSB=1	1	31	$-\frac{15}{32} = -\frac{30}{64}$

Table 4: Comparisons Between the Bias of Linear Approximations and Conditional Linear Approximations

Number of Rounds	Approximation	Bias	Conditional Bias	Factor
1	λ_1 (Figure 4)	$-2^{-1.67}$	$-2^{-1.09}$	1.5
4	λ_2 (Figure 7)	$2^{-7.35}$	$2^{-6.35}$	2
8	λ_3 (Figure 8)	$2^{-13.71}$	$2^{-12.71}$	2
13	λ_4 (Figure 9)	$2^{-20.06}$	$2^{-19.06}$	2
14	λ_5 (Figure 10)	$-2^{-20.75}$	$-2^{-20.16}$	1.54
14	λ_6 (Figure 10)	$-2^{-21.48}$	$-2^{-20.26}$	2.28
2	λ_7 (Figure 11)	0	$2^{-5.50}$	∞

2.2 A General Description of the Attack

Suppose for example that a regular linear attack that uses some linear approximation with bias ε requires about n known plaintexts, and suppose that a conditional linear approximation in which this linear approximation is conditioned on another has a higher bias ε^*. Lets also assume that $\varepsilon^* = 2\varepsilon$. We claim that our new version of the attack requires $n/2$ known plaintexts. The attack starts by discarding about half of the data that fails to satisfy the condition, leaving only the encryptions that satisfy the condition. We then apply the original attack on the remainder of the data.

This smaller number of known plaintexts suffices, as given $n/2$ known plaintexts, the attack discards about half of them, and remains with about $n/4$ plaintexts. Since attacks with twice the bias require about a quarter of the known plaintexts, these $n/4$ plaintexts suffice to find the key using the original attack algorithm. In general, when $\varepsilon^* > \sqrt{2}\varepsilon$, the gain is expected to be by a factor of about $(\varepsilon^*/\varepsilon)^2/2$.

2.3 Notations

Throughout this paper we omit the initial permutation IP and the final permutation IP^{-1}, and use the following notations:

P_L	is the 32 leftmost bits of the plaintext.
P_R	is the 32 rightmost bits of the plaintext.
C_L	is the 32 leftmost bits of the ciphertext.
C_R	is the 32 rightmost bits of the ciphertext.
Y^i	is the 32 output bits of the F function in the ith round ($1 \le i \le 16$).
X_j	is the jth bit of X, for any variable X ($1 \le j \le \lvert X \rvert$, starting from the left of X).

Notice that under this notations:

$$P_L \oplus C_R = \bigoplus_{k\ is\ odd} Y^k$$

$$P_R \oplus C_L = \bigoplus_{k\ is\ even} Y^k. \tag{1}$$

Given an r-round linear approximation λ, we can treat λ as r concatenated one-round approximations:

λ^i	is the approximation of round i.
I^i	$\in \{0,1\}$. $I^i = 0$ indicates that round i satisfies the parity approximation of λ^i.
$\lambda^{i,j}$	is the $(j-i+1)$-round approximation derived from λ from the ith round to the jth round.
λ^{odd}	is an extended approximation composed of all the odd rounds of all λ^i with odd i's.
λ^{even}	is an extended approximation composed of all the even rounds of all λ^i with even i's.
$\frac{1}{2}+\varepsilon(\lambda^{odd})$	is the probability of λ^{odd}, i.e., the probability that $\bigoplus_{k\,is\,odd} I^k = 0$.
$\varepsilon(\lambda^{odd})$	is the bias of λ^{odd}.
$\frac{1}{2}+\varepsilon(\lambda^{even})$	is the probability of λ^{even}, i.e., the probability that $\bigoplus_{k\,is\,even} I^k = 0$.
$\varepsilon(\lambda^{even})$	is the bias of λ^{even}.

Given an iterative linear approximation λ:

$\lambda^{\otimes r}$	is the iteration of λ to r rounds.

Notice that although λ^{odd} and λ^{even} are approximations of non-consecutive rounds (i.e., only odd or only even rounds), still the computations of their biases follow the regular computation of biases in linear cryptanalysis, e.g., piling up lemma. It is therefore that "concatenation" of λ^{odd} and λ^{even} results with the original λ, whose bias satisfies the usual formula $\varepsilon(\lambda) = 2\varepsilon(\lambda^{odd})\varepsilon(\lambda^{even})$.

As mentioned earlier, we expect to be able to improve the bias $\varepsilon(\lambda^{odd})$ of λ^{odd} (or the bias $\varepsilon(\lambda^{even})$ of λ^{even}) by conditioning on other linear approximations, in which case we also improve the bias of the full λ under the same conditions.

For an r-round linear approximation which has at most one active S box in each round:[1]

y^{λ^i}	is the four output bits of the active S box in the i-th round.
$\delta^{\lambda,odd}$	$\stackrel{\Delta}{=} \bigoplus_{k\,is\,odd} y^{\lambda^k}$.
$\delta^{\lambda,even}$	$\stackrel{\Delta}{=} \bigoplus_{k\,is\,even} y^{\lambda^k}$.

We observe that in a linear approximation in which all the odd rounds have the same active S box, the value of $\delta^{\lambda,odd}$ is known given the plaintext and the ciphertext.

For any linear approximation λ in which every odd round has the same active S box, and any $V \in \{0,1\}^4$:

$\frac{1}{2}+\varepsilon(\lambda^{odd}\|V)$	is the conditional probability that $\bigoplus_{k\,is\,odd} I^k = 0$, given $\delta^{\lambda,odd} = V$.
$\varepsilon(\lambda^{odd}\|V)$	is the conditional bias of λ^{odd}, given $\delta^{\lambda,odd} = V$.

We stress that we use the bit numbering of DES as in the standard [Nat77].[2]

2.4 Conditional Biases Piling Up Lemma

Matsui showed that by concatenating a linear approximation with bias ε_1 with a linear approximation with bias ε_2, the bias of the concatenated approximation is $\varepsilon = 2\varepsilon_1\varepsilon_2$. A similar, but more complex piling up lemma is available also in our case. Let X and Y be binary random variables.

[1] For simplicity of the discussion and notations, throughout this paper we assume that we have at most one active S box in each round. This assumption is correct for the most useful linear approximations. Whenever we discuss approximations with two or more active S boxes in a round, it will clearly be deduced from the context.

[2] Unlike some other papers, e.g., [Mat94] that invented their own numbering system.

$\varepsilon(X)$	is the bias of X, i.e.: $\varepsilon(X) = Pr(X = 0) - \frac{1}{2}$,
$\varepsilon(X\|Y = t)$	is the conditional bias of X under the event that $Y = t$, i.e.: $\varepsilon(X\|Y = t) = Pr(X = 0\|Y = t) - \frac{1}{2}$.

Let X_1, X_2, Y_1, Y_2 be binary random variables such that X_1 and Y_1 are independent of X_2 and Y_2. Denote:

$$\begin{aligned} p_1 &= Pr(Y_1 = 0) \\ p_2 &= Pr(Y_2 = 0) \\ \alpha_1 &= \varepsilon(X_1|Y_1 = 0) \\ \alpha_2 &= \varepsilon(X_2|Y_2 = 0) \\ \beta_1 &= \varepsilon(X_1|Y_1 = 1) \\ \beta_2 &= \varepsilon(X_2|Y_2 = 1) \end{aligned}$$

Lemma 1.

$$\varepsilon(X_1 \oplus X_2|Y_1 \oplus Y_2 = 0) = \frac{2p_1p_2\alpha_1\alpha_2 + 2(1-p_1)(1-p_2)\beta_1\beta_2}{p_1p_2 + (1-p_1)(1-p_2)}.$$

Lemma 2.

$$\varepsilon(X_1 \oplus X_2|Y_1 \oplus Y_2 = 1) = \frac{2(1-p_1)p_2\beta_1\alpha_2 + 2p_1(1-p_2)\alpha_1\beta_2}{(1-p_1)p_2 + p_1(1-p_2)}.$$

Conclusion 1. *If both* $\alpha_1 = \alpha_2 = 0$ *or both* $\beta_1 = \beta_2 = 0$ *then*

$$\varepsilon(X_1 \oplus X_2|Y_1 \oplus Y_2 = 1) = 0.$$

The proofs are given in Appendix B.

2.5 Computing Conditional Biases in DES

We can calculate conditional biases of conditional approximation, assuming that the DES subkeys are independent, and that this bias is a good approximation for the bias of a random DES key processed through the DES key schedule.

For any linear approximation λ in which every odd round has the same active S box, the calculation is as follows: Given an r-round linear approximation λ, and $V \in \{0,1\}^4$, we can calculate the value of $\varepsilon(\lambda^{odd}|V)$ recursively. For $r \leq 2$ the calculation is trivial. For $r > 2$, we can treat λ as two concatenated approximations, $\lambda^{1,2}$ and $\lambda^{3,r}$, such that $\lambda^{1,2}$ consists of the first two rounds of λ, and $\lambda^{3,r}$ consists of the last $r-2$ rounds of λ. Given $V \in \{0,1\}^4$ we can calculate the value of $\varepsilon(\lambda^{odd}|V)$ in terms of $\lambda^{1,2}$ and $\lambda^{3,r}$. Notice that for any $V \in \{0,1\}^4$, there exist $2^4 = 16$ pairs of vectors $X, Y \in \{0,1\}^4$, such that $X \oplus Y = V$. Therefore,

$$\varepsilon(\lambda^{odd}|V) = \frac{1}{16} \sum_{X|X \in \{0,1\}^4} 2\varepsilon(\lambda^{1,2,odd}|X)\varepsilon(\lambda^{3,r,odd}|Y). \tag{2}$$

3 Examples of Conditional Linear Approximations and Simple Attacks

In this section we present examples for conditional linear approximations, and discuss some of the attacks based on them.

Table 5: The Biases of Single-Round Conditional Linear Approximation

$\delta^{\lambda_1,odd}$	$\varepsilon(\lambda_1\|\delta^{\lambda_1,odd})$	$\delta^{\lambda_1,odd}$	$\varepsilon(\lambda_1\|\delta^{\lambda_1,odd})$
0000	–0.5	1000	0
0001	–0.5	1001	–0.5
0010	–0.5	1010	0
0011	–0.5	1011	–0.5
0100	–0.25	1100	0.25
0101	–0.5	1101	–0.25
0110	–0.25	1110	0
0111	–0.5	1111	–0.5

3.1 A Single-Round Conditional Linear Approximation

Consider the best one-round linear approximation λ_1 from Figure 4. It has one active S box, $S5$: $10_x \rightarrow F_x$. Every S box has four output bits, therefore we can condition $\varepsilon(\lambda_1)$ on each of the 16 possible values of the four output bits of $S5$. In Table 5 we show the bias of λ_1 conditioned on each of these 16 possible values. We can filter the data according to any subset of the 16 possible values of the output of $S5$ in order to increase the bias. Notice that conditions can be made on more than one bit, or several multi-bit values, e.g., discarding all data with outputs in the set {1000, 1010, 1110} or {0100, 0110, 1000, 1010, 1100, 1101, 1110}.

3.1.1 Algorithm for Simple Attack

Linear cryptanalysis of DES reduced to three rounds is based on a single-round linear approximation. We set λ_1 in the second round, and the analysis guesses key bits related to the first and third rounds. Given a condition on the output of $S5$ in λ_1, our attack starts by discarding the data that fails to satisfy the condition, leaving only the data that do satisfy the condition. We then apply the original attack on the remaining data.

We apply the following algorithm:

1. Given n known plaintexts and their corresponding ciphertexts $C_i = DES_K(P_i)$, where K is the unknown secret key.
2. For each plaintext P_i and the corresponding ciphertext C_i:
   (a) Compute the value v of the four output bits of $S5$ in the second round using P_i and C_i.
   (b) If v fails to satisfy the condition, discard P_i and C_i.
3. Apply the original attack on the remaining data.

Notice that in Step 2a, the output of the F function in Round 2 is $P_{iR} \oplus C_{iR}$, and v consists from bits of this value.

3.1.2 Attack Based on Several Multi-Bit Conditions

In Table 5 we can see that 9 out of the 16 conditional biases are $-1/2$. Therefore, we can condition λ_1 on these 9 values in order to filter the data. Due to this filtering, we discard about 7/16 of the data in which the output of $S5$ is in {0100, 0110, 1000, 1010, 1100, 1101, 1110}. Then, the bias of the remaining data increases from $-20/64$ to $-1/2$ by a factor of about 1.6. Therefore, the resultant attack requires a total of about $\frac{16}{9} \cdot n/1.6^2 \approx 0.695n$ known plaintexts, where n is the number of known plaintexts required by the original attack. The detailed attack follows the steps of the algorithm of Section 3.1.1.

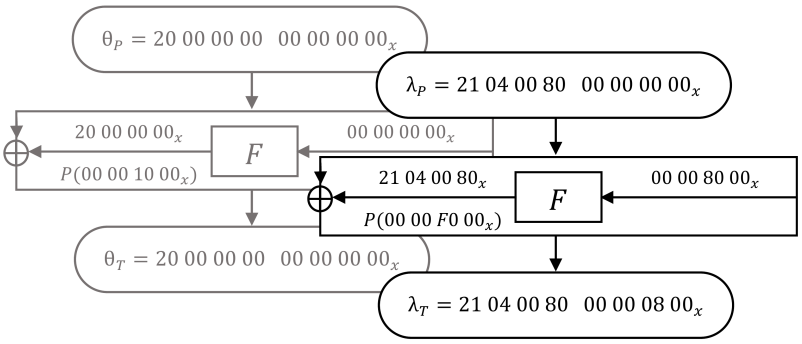

Figure 6: λ_1 Conditioned on the LSB of the Four Output Bits of $S5$ $(\lambda_1|\theta_1)$

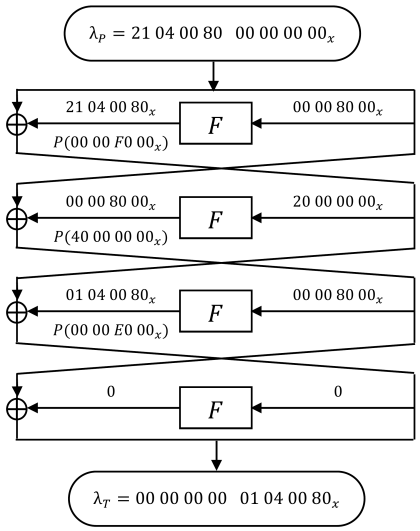

Figure 7: Four Rounds of Matsui's Linear Approximation (λ_2)

3.1.3 Attack Based on Single-Bit Conditions

In half of the data (in which the LSB is 1) the bias grows from $-20/64$ to $-15/32 = -30/64$, and in the other half (in which the LSB is 0) the bias reduces to $-5/32 = -10/64$. Therefore, we can condition λ_1 on the LSB in order to increase the bias. This condition can be presented as a linear approximation $S5$: $0_x \rightarrow 1_x$ (denoted by θ_1) with bias 0. Figure 6 presents λ_1 conditioned on the LSB of the output of $S5$, where the conditioned approximation λ_1 is in black on top of the condition θ_1 that is marked in gray. The corresponding attack starts by discarding about half of the data, and using only the data in which the LSB is 1. In this half of the data the bias grows from $-20/64$ to $-15/32 = -30/64$. The detailed attack follows the steps of the algorithm of Section 3.1.1.

3.2 A Four-Round Conditional Linear Approximation

Consider four consecutive rounds of Matsui's linear approximation as outlined in Figure 7, and denote this four-round linear approximation by λ_2. This linear approximation has three active S boxes: $S5$ in the first and third rounds, and $S1$ in the second round. We notice that in λ_2 every odd round has the same active S box, and therefore we can calculate the value of $\delta^{\lambda_2,odd}$ directly from the plaintext and ciphertext. We use this XOR values to filter the data. In Table 6 we show $\varepsilon(\lambda_2^{odd})$ for each of the 16 possible values of $\delta^{\lambda_2,odd}$

Table 6: Conditioning $\varepsilon({\lambda_2}^{odd})$ on Each of the 16 Values of $\delta^{\lambda_2,odd}$

$\delta^{\lambda_2,odd}$	$\varepsilon(\lambda_2^{odd}\|\delta^{\lambda_2,odd})$	$\delta^{\lambda_2,odd}$	$\varepsilon(\lambda_2^{odd}\|\delta^{\lambda_2,odd})$
0000	–0.14	1000	–0.23
0001	0	1001	0
0010	–0.14	1010	–0.23
0011	0	1011	0
0100	–0.15	1100	–0.25
0101	0	1101	0
0110	–0.15	1110	–0.25
0111	0	1111	0

(calculated by Equation 2). We see in the table that 8 of the 16 entries have zero biases, and that the other 8 are distributed in the range -0.14 to -0.25. The condition to choose between these two cases is by the LSB of the output bits of $S5$, and the average bias when the condition holds is -0.1925. Alternatively, it is possible to consider only the 4 values with the highest biases. In this case the condition is by two bits (the LSB and the MSB of the output), and the average bias is -0.24. Though the bias increases in this latter case compared to the former one, in order to get this bias we need to discard more data, which leaves us with half of the data that remains in the former case.

3.2.1 Attack on Four Rounds

For applying Matsui's Algorithm 1 with our observation, we discard about half of the known plaintexts, and use only the plaintexts with $\delta_4^{\lambda_2,odd} = 0$. Using only these plaintexts, the bias $\varepsilon(\lambda_2^{odd})$ is increased by a factor of about two. We request the same type of data as Matsui would for such a 4-round attack, and discard encryptions which do not follow our relation. We then apply the original algorithm of Matsui on the remaining data. Due to our filtering, the bias ε increases by a factor of about two. As the number of required data is quadratic in ε^{-1}, it reduces the number of known plaintexts by a factor of about four. On the other hand, since we discard about half of the data, we need twice this reduced number of known plaintexts. I.e., the resultant attack requires a total of about half the known plaintexts than the original attack, discards about half of them, and uses the remaining quarter to find the key.

3.3 An Eight-Round Conditional Linear Approximation

Consider eight consecutive rounds of Matsui's iterative linear approximation as outlined in Figure 8. We denote this linear approximation by λ_3, where $\lambda_3^{1,4}$ equals to λ_2 (the four-round approximation described in Section 3.2). The odd rounds of $\lambda_3^{5,8}$ are the same odd rounds as of $\lambda_3^{1,4}$ (in a different order), therefore Table 6 also shows the calculation of $\varepsilon({\lambda_3^{5,8}}^{odd})$ for each of the 16 possible values of $\delta^{\lambda_3^{5,8},odd}$. In about half of the cases, $\delta_4^{\lambda_3^{1,4},odd} = 0$, and in the other half $\delta_4^{\lambda_3^{1,4},odd} = 1$. Similarly, in about half of the cases $\delta_4^{\lambda_3^{5,8},odd} = 0$, and in the other half $\delta_4^{\lambda_3^{5,8},odd} = 1$. Thus, we have four cases for the possible values of $\delta_4^{\lambda_3^{1,4},odd}$ and $\delta_4^{\lambda_3^{5,8},odd}$, as outlined in Table 7. We can learn from the table that even in this case using only the half of the data in which the full 8-round condition $\delta_4^{\lambda_3,odd} = 0$ holds (i.e., the main diagonal in the table) the bias $\varepsilon(\lambda_3^{odd}|\delta_4^{\lambda_3,odd} = 0)$ is increased by a factor of about two.

Alternatively, can also calculate $\varepsilon(\lambda_3^{odd}|\delta_4^{\lambda_3,odd} = 1) = \varepsilon(\lambda_3^{1,4odd} \oplus \lambda_3^{5,8odd}|\delta_4^{\lambda_3^{1,4},odd} \oplus \delta_4^{\lambda_3^{5,8},odd} = 1)$ by substituting the "shorter" biases and probabilities in the conditional biases

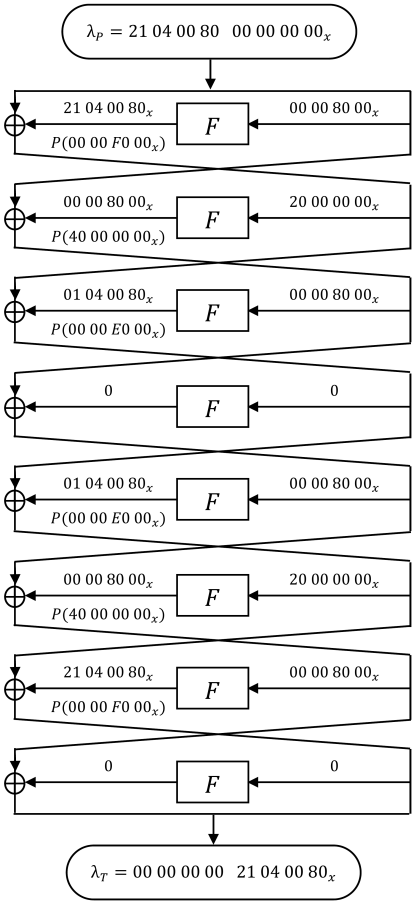

Figure 8: Matsui's Eight-Round Iterative Linear Approximation (λ_3)

Table 7: Four Cases for the Possible Values of $\delta_4^{\lambda_3^{1,4},odd}$ and $\delta_4^{\lambda_3^{5,8},odd}$

	$\delta_4^{\lambda_3^{1,4},odd} = 0$	$\delta_4^{\lambda_3^{1,4},odd} = 1$
$\delta_4^{\lambda_3^{5,8},odd} = 0$	$\delta_4^{\lambda_3,odd} = 0$ $\varepsilon(\lambda_3) > 0$	$\delta_4^{\lambda_3,odd} = 1$ $\varepsilon(\lambda_3) = 0$
$\delta_4^{\lambda_3^{5,8},odd} = 1$	$\delta_4^{\lambda_3,odd} = 1$ $\varepsilon(\lambda_3) = 0$	$\delta_4^{\lambda_3,odd} = 0$ $\varepsilon(\lambda_3) = 0$

Table 8: The Bias of λ_4 Conditioned on $\delta_4^{\lambda_4,even}$

Condition	$\varepsilon(\lambda_4)$
none	$2^{-20.06}$
$\delta_4^{\lambda_4,even} = 0$	$2^{-19.06}$
$\delta_4^{\lambda_4,even} = 1$	0

Table 9: Conditioning $\varepsilon({\lambda_5}^{odd})$ on Each of the 16 Values of $\delta^{\lambda_5,odd}$

$\delta^{\lambda_5,odd}$	$\varepsilon(\lambda_5^{odd}\|\delta^{\lambda_5,odd})$	$\delta^{\lambda_5,odd}$	$\varepsilon(\lambda_5^{odd}\|\delta^{\lambda_5,odd})$
0000	0.0011	1000	0.0011
0001	0.0034	1001	0.0034
0010	0.0011	1010	0.0011
0011	0.0034	1011	0.0034
0100	0.0011	1100	0.0011
0101	0.0034	1101	0.0034
0110	0.0011	1110	0.0011
0111	0.0034	1111	0.0034

piling up lemma. As $\beta_1 = \varepsilon({\lambda_3^{1,4}}^{odd}|\delta_4^{\lambda_3^{1,4},odd} = 1) = \beta_2 = \varepsilon({\lambda_3^{5,8}}^{odd}|\delta_4^{\lambda_3^{5,8},odd} = 1) = 0$ then by Conclusion 1 we get $\varepsilon(\lambda_3^{odd}|\delta_4^{\lambda_3,odd} = 1) = 0$.

3.4 A 13-Round Conditional Linear Approximation

Consider 13 consecutive rounds of Matsui's iterative linear approximation as outlined in Figure 9, and denote this 13-round linear approximation by λ_4. Table 8 shows the bias of λ_4 conditioned on $\delta_4^{\lambda_4,even}$. Using only the data in which $\delta_4^{\lambda_4,even} = 0$ the bias is increased by a factor of about two, while the bias of the data in which $\delta_4^{\lambda_4,even} = 1$ reduces to 0. A second approximation with the same condition and biases is based on λ_4 while flipping its order of rounds from the last to the first. We call it flipped λ_4. We later use both of these 13-round conditional linear approximations for our main attack on 16-round DES.

3.5 A 14-Round Conditional Linear Approximations

We present here two 14-round linear approximations and show their transformation into conditional linear approximations. The first presented linear approximation, denoted by λ_5, is the best 14-round linear approximation, same as Matsui uses in his attacks on the full 16-round DES. The second linear approximation, denoted by λ_6, differs from λ_5 in the first round. The bias of λ_6 is more than twice worse than the bias of λ_5. However, as we will soon see, it is attractive for our attacks because it has a higher increase in the bias when conditioning it, so it's conditional bias is very close to the conditional bias of λ_5. Both linear approximations are shown together in Figure 10 due to their similarity. The first round of λ_5 is demonstrated on the upper left, and the first round of λ_6 is demonstrated on the upper right. The rest of the rounds are the same in both linear approximations.

In Table 9 we calculate $\varepsilon(\lambda_5^{odd}|\delta^{\lambda_5,odd})$ conditioned on each of the 16 possible values of $\delta^{\lambda_5,odd}$, and in Table 10 we calculate a similar table for $\varepsilon(\lambda_5|\delta^{\lambda_5,odd})$, which is the bias of the full λ_5 conditioned on each of the 16 possible values of $\delta^{\lambda_5,odd}$.

Similarly, in Table 11 we calculate $\varepsilon(\lambda_6^{odd}|\delta^{\lambda_6,odd})$ conditioned on each of the 16 possible values of $\delta^{\lambda_6,odd}$, and in Table 12 we calculate a similar table for $\varepsilon(\lambda_6|\delta^{\lambda_6,odd})$,

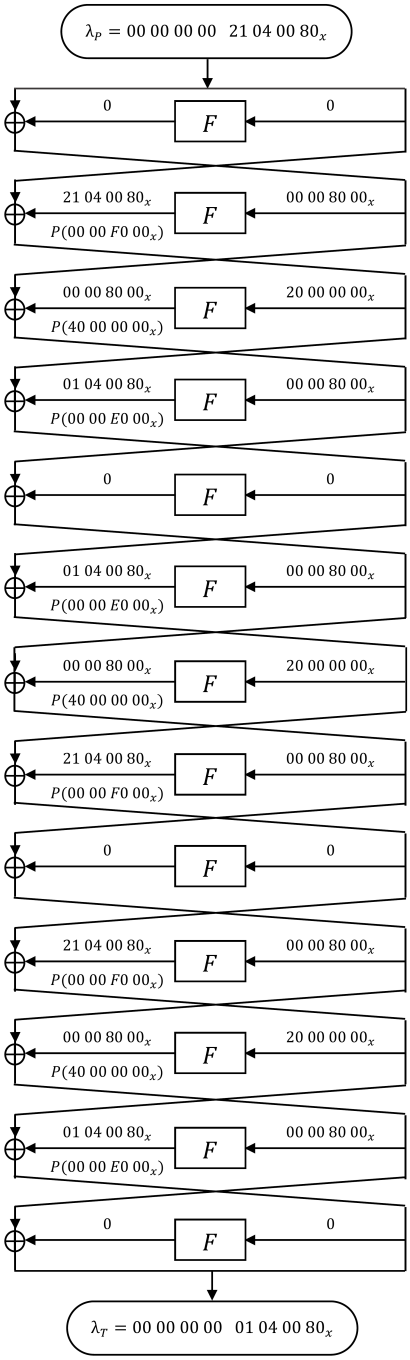

Figure 9: 13 Rounds of Matsui's Linear Approximation (λ_4)

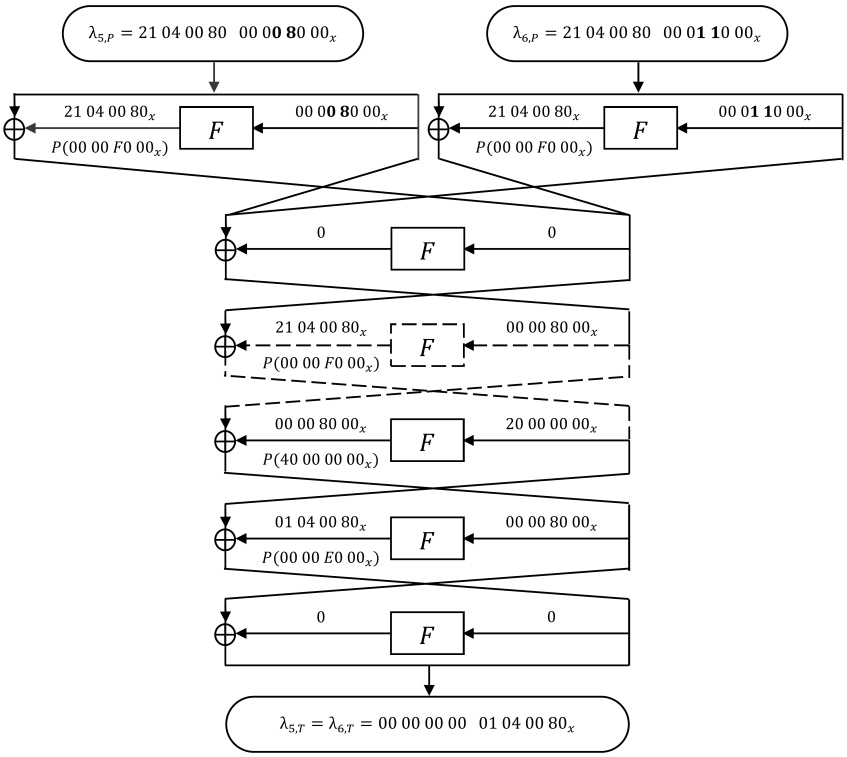

Figure 10: λ_5 and λ_6. The Figure shows 6 rounds of 14 where the broken lines represent an inclusion of additional 8 rounds of the iterative approximation λ_3 of Figure 8. The first round of λ_5 is demonstrated on the upper left, and the first round of λ_6 is demonstrated on the upper right. The rest of the rounds are the same in both.

Table 10: Conditioning $\varepsilon(\lambda_5)$ on Each of the 16 Possible Values of $\delta^{\lambda_5,odd}$

$\delta^{\lambda_5,odd}$	$\varepsilon(\lambda_5\|\delta^{\lambda_5,odd})$	$\delta^{\lambda_5,odd}$	$\varepsilon(\lambda_5\|\delta^{\lambda_5,odd})$
0000	$-2^{-21.77}$	1000	$-2^{-21.71}$
0001	$-2^{-20.16}$	1001	$-2^{-20.16}$
0010	$-2^{-21.77}$	1010	$-2^{-21.71}$
0011	$-2^{-20.16}$	1011	$-2^{-20.16}$
0100	$-2^{-21.77}$	1100	$-2^{-21.77}$
0101	$-2^{-20.16}$	1101	$-2^{-20.16}$
0110	$-2^{-21.77}$	1110	$-2^{-21.71}$
0111	$-2^{-20.16}$	1111	$-2^{-20.16}$

Table 11: Conditioning $\varepsilon({\lambda_6}^{odd})$ on Each of the 16 Values of $\delta^{\lambda_6,odd}$

$\delta^{\lambda_6,odd}$	$\varepsilon(\lambda_6^{odd}\|\delta^{\lambda_6,odd})$	$\delta^{\lambda_6,odd}$	$\varepsilon(\lambda_6^{odd}\|\delta^{\lambda_6,odd})$
0000	0.0032	1000	0.0032
0001	–0.0004	1001	–0.0004
0010	0.0032	1010	0.0032
0011	–0.0004	1011	–0.0004
0100	0.0032	1100	0.0032
0101	–0.0004	1101	–0.0004
0110	0.0032	1110	0.0032
0111	–0.0004	1111	–0.0004

Table 12: Conditioning $\varepsilon(\lambda_6)$ on Each of the 16 Possible Values of $\delta^{\lambda_6,odd}$

$\delta^{\lambda_6,odd}$	$\varepsilon(\lambda_6\|\delta^{\lambda_6,odd})$	$\delta^{\lambda_6,odd}$	$\varepsilon(\lambda_6\|\delta^{\lambda_6,odd})$
0000	$-2^{-20.26}$	0100	$-2^{-20.26}$
1000	$2^{-23.13}$	1100	$2^{-23.00}$
0001	$-2^{-20.26}$	0101	$-2^{-20.26}$
1001	$2^{-23.13}$	1101	$2^{-23.00}$
0010	$-2^{-20.26}$	0110	$-2^{-20.26}$
1010	$2^{-23.13}$	1110	$2^{-23.00}$
0011	$-2^{-20.26}$	0111	$-2^{-20.26}$
1011	$2^{-23.13}$	1111	$2^{-23.00}$

the full λ_6 conditioned on each of the 16 possible values of $\delta^{\lambda_6,odd}$.

We can learn from the tables that in case we use only encryptions in which $\delta_4^{\lambda_6,odd} = 0$, the absolute bias $|\varepsilon(\lambda_6^{odd})|$ of λ_6 grows from 0.0014 to 0.0032 by a factor of about 2.28. Similarly, in case we use only encryptions in which $\delta_4^{\lambda_5,odd} = 1$, the absolute bias $|\varepsilon(\lambda_5^{odd})|$ of λ_5 grows from 0.0022 to 0.0034 by a factor of about 1.54.

Table 13 summarizes the biases of λ_5 and λ_6, as well as their conditional biases conditioned on each possible value of $\delta_4^{\lambda_5,odd}$.

3.5.1 Attack Using a Single Conditional Linear Approximation

In order to attack the full 16-round DES, Matsui uses λ_5, which is the best 14-round approximation with probability $\frac{1}{2} - 2^{-20.75}$. We use it's corresponding conditional linear approximation with probability $\frac{1}{2} - 2^{-20.16}$. Our attack is very similar to Matsui's original attack on this cipher. In order to apply Matsui's Algorithm 2 with our conditional linear approximation, we discard about half of the known plaintexts, and use only the plaintexts with $\delta_4^{\lambda_5,odd} = 1$. Using only these plaintexts the bias $\varepsilon(\lambda_5^{odd})$ is increased by a factor of about $\frac{-2^{-20.16}}{-2^{-20.75}} \approx 1.505$. As the number of required data is quadratic in ε^{-1}, it reduces

Table 13: The Bias of λ_5 and λ_6 Conditioned on $\delta_4^{\lambda_5,odd}$

Condition	$\varepsilon(\lambda_5)$	$\varepsilon(\lambda_6)$
none	$-2^{-20.75}$	$-2^{-21.48}$
$\delta_4^{\lambda_5,odd} = 0$	$-2^{-21.74}$	$-\mathbf{2^{-20.26}}$
$\delta_4^{\lambda_5,odd} = 1$	$-\mathbf{2^{-20.16}}$	$2^{-23.06}$

the number of known plaintexts by a factor of about $1.505^2 \approx 2.266$. On the other hand, since we discard about half of the data, we need two times this reduced number of known plaintexts. Therefore, the resultant attack requires a total of about $2n/2.266 = 0.88n$ known plaintexts, where n is the number of known plaintexts required by the original attack. We conclude that even using only this single conditional linear approximation, we already save 12% over Matsui's attack.

To be more precise, in this attack we follow all the details of Matsui's attack. For example, Matsui uses his best approximation along with it's flipped version, and for each he considers the highest 100 values. Our improved attack follows his steps using our smaller amount of filtered data, instead of using all the data.

3.5.2 Attack Using Both Conditional Linear Approximations

Using both conditional linear approximations (along with both flipped versions) we can save even more. One of them, λ_5, suggests 12 key bits, combining 6 key bits related to $S5$ in the last round and 6 key bits related to $S1$ in the first round. The other linear approximation, λ_6, suggests the same 6 key bits in the last round, but 12 key bits in the first round, which related to $S3$ and $S4$. In principle, we can recover 13 bits of the key of the full 16-round DES using λ_5 by setting the 14-round linear approximation from Round 2 to Round 15, if we have sufficient amount of known plaintexts. Similarly, in principle we can recover 19 bits of the key of the full 16-rounds DES using λ_6 by setting the 14-round linear approximation from Round 2 to Round 15 (6 of these 19 bits are shared with λ_5). The flipped versions suggest a similar number of additional key bits, which add a total of 14 key bits over the ones suggested by the original non-flipped approximations. Due to the additional conditional linear approximations, this attack requires fewer known plaintexts and a shorter time of analysis than the previous one. We skip the full details of this attack, as we will later elaborate on a more efficient attack on the same cipher.

3.5.3 Analysis

Notice that if we only had $0.88 \cdot 2^{43}$ known plaintexts, as in Section 3.5.1, the right value of the 12 corresponding key bits should have been included in the best 100 values.[3] But as we are interested in fewer known plaintexts, it is not ensured that the right value of the key bits is included in the top 100 values. Therefore, a more sophisticated analysis that involves all four linear approximations together is required. Notice that the key bits that corresponds to $S1$, $S3$ and $S4$ are all part of the C register of the key schedule [Nat77] and include together only 26 bits, so together they contain duplication that can be checked for redundancy. Along with the 12 bits corresponding to $S5$ of register D, we get a total of 38 key bits. In our algorithm we test all these 2^{38} cases for validity against the $26 + 12$ known bits. This step can be performed efficiently, but notice that even inefficiently it is efficient enough for our purposes. Each of these cases is checked against the known duplications of these 10 bits. We take the ones with the highest product of the biases derived from the four conditional linear approximations, and test each by exhaustively searching for the remaining $14 = 56 - 38 - 4$ key bits (where the 4 stands for the 4 parity bits received from the four linear approximations). Assuming that the correct 42-bit guess is withing the 2^{28} highest in the list, the complexity of this step has less then 2^{42}. Given 2^{42} known plaintexts it is still expected that the correct value is well within the above range (as was also shown in our experiments), so that the complexity of analysis is much less than 2^{42}.

[3] Matsui limited his attack to the 100 best values, as higher numbers would increase his analysis complexity over 2^{43}. Clearly, if he had used more values, and could somehow reduce their time of analysis, his probability of success would have been increased.

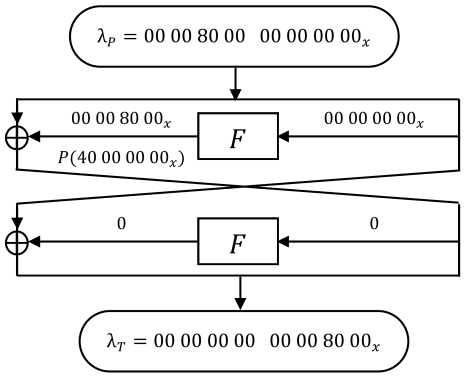

Figure 11: Two-Round Iterative Linear Approximation (λ_7)

3.6 A Zero Bias Approximation Conditioned to Non-Zero Bias

In [Bih94] Biham shows that there exist single-round linear approximations with an empty input set of the F function, but a non empty output set. These approximations are then extended to become two-round iterative linear approximations, where the other round contains the trivial one-round linear approximation with bias 1/2. In these approximations there are (at least) two active S boxes in the active round, where the parity bit of the approximation involves bits from the outputs of both S boxes.

In the case of conditional approximations we can present similar two-round linear approximations with two active S boxes in the active round, but in which the parity involves bits from the output of only one of the active S boxes. In the example below we will actually show that only one active bit in the output of the F function is possible.

There is no prior example of any kind of approximations in which there is no active bit in the output of an active S box. In fact, such approximations (featuring only input bits without output bits in their approximated subsets) must have bias 0. However, in our case, we have the extra power to condition on the output bits of both S boxes, and we use it in a way that increases the bias of the active S box due to the condition. As such, the condition on the output bits substitutes the active bits in the output.

Such an approximation, to which we call λ_7, is outlined in Figure 11. It is a two-round iterative approximation, with a single mask bit at the output of $S1$ in the first round, and no masked bit anywhere else in F. Clearly, its bias is 0.

We observe that when it is conditioned on the parity of the bits masked 9_x in the output of $S1$ and the bits masked E_x in the output of $S2$ (a parity of a total of five bits), the bias changes dramatically. Moreover, this conditional bias depends on the parity of the four key bits which are mixed with the shared input bits of $S1$ and $S2$. This condition can be presented as a linear approximation (denoted by θ_2), as outlined in Figure 12. For example, when λ_7 is conditioned on the parity of the five bits in the output of $S1$ and $S2$ being 0, and on the parity of the four key bits being 0, the bias becomes $0.022 \approx 2^{-5.50}$, and when it is conditioned on the parity of the five bits in the output of $S1$ and $S2$ being 1, and on the parity of the four key bits being 0, the bias becomes $-0.025 \approx -2^{-5.32}$.

When iterated, the conditional biases (as computed by the conditional piling up lemma) remain non zero. They are reported in Table 14. Later in this paper the 8-round iteration of λ_7 ($\lambda_7^{\otimes 8}$) is named λ_8. Notice that when looking at the masks of the parities of the inputs and outputs in the active round of λ_7, it might look like $S2$ is not an active S box because it has no active bits in its output, and there are no active bits at all in the input of the F function. However, by carefully analyzing the situation we see that two of its input bits are active. The reason that the input of the F function has no active input bits is because the two active input bits of $S1$ share the same bits in the input of

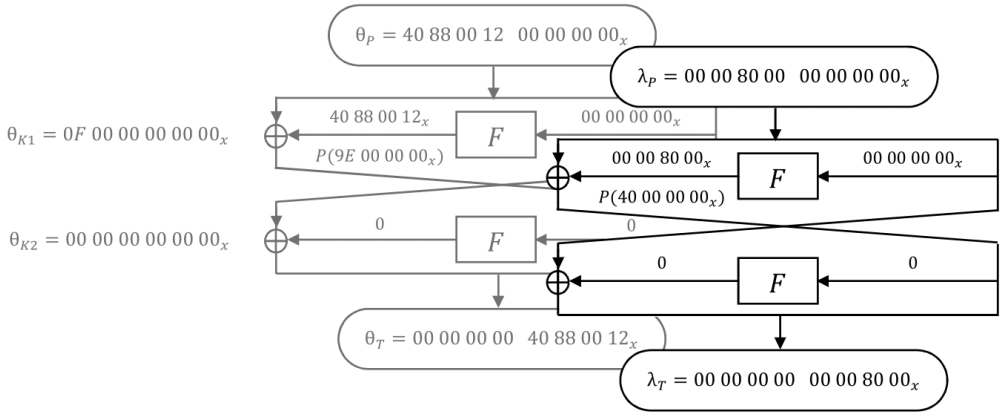

Figure 12: λ_7 Conditioned on θ_2 $(\lambda_7|\theta_2)$

Table 14: The Conditional Biases of Iterated λ_7

Number of Rounds (r)		$\varepsilon(\lambda_7^{\otimes r})$	$\varepsilon(\lambda_7^{\otimes r}\|\theta_2^{\otimes r}=0)$	$\varepsilon(\lambda_7^{\otimes r}\|\theta_2^{\otimes r}=1)$
2	λ_7	0	$2^{-5.50}$	$-2^{-5.32}$
4		0	$2^{-9.83}$	$-2^{-9.82}$
8	λ_8	0	$2^{-18.66}$	$-2^{-18.66}$
16		0	$2^{-36.23}$	$-2^{-36.23}$

the F function with those of $S2$, so the active bits of $S1$ and of $S2$ cancel each others' activity in the input of the F function. As we already explained, on the output side of $S2$ there are no active bits because the input is unbalanced due to the condition, which makes it possible to approximate input bits even without active output bits. Notice that both $\varepsilon(\lambda_7|\theta_2=0)$ and $\varepsilon(\lambda_7|\theta_2=1)$ have non-zero biases, so in actual applications we may prefer to use both partitions of the data (with the condition being 0 and being 1), without discarding any data.

4 Our Best Conditional Linear Cryptanalysis of the Full 16-Round DES

This section applies our extension on the full 16-round DES. The attack in this section is based on two 13-round linear approximations: λ_4 from Section 3.4 and Figure 9, along with its flipped version. Each of λ_4 and the flipped λ_4 provide candidates for 27 key bits and one additional parity bit. Table 15 lists the 27 key bits provided by λ_4 and the 27 key bits provided by the flipped λ_4. The underlined key bits in the table mark the 10 common bits provided by both.

The attack assumes that λ_4 is set in Rounds 3–15, and the analysis guesses key bits related to Rounds 1, 2 and 16. Figure 13 outlines this setting.[4] In addition, the attack applies an inverted version of this setting for use with the flipped version of λ_4, where the flipped λ_4 is set in Rounds 2–14, and the analysis guesses key bits related to Rounds 1, 15 and 16. This latter setting applies the same search algorithm as the former one, while exchanging the use of the plaintexts and the ciphertext in the computation. In the following description, we only describe the algorithm of the former setting with λ_4. The

[4]The 16 rounds are represented by 8, where the broken lines represent an inclusion of additional 8 rounds of the iterative approximation λ_3 of Figure 8.

Table 15: The Key Bits Involved in the Two Conditional Linear Approximations Used in the Attack (the underlined bits are common to both approximations)

Linear Approx.	Subkey	S box	Key Bits
λ_4	$K1$	$S1$	10, 51, <u>34</u>, 60, <u>49</u>, 17
	$K1$	$S3$	<u>3</u>, <u>35</u>, <u>26</u>, <u>25</u>, <u>44</u>, 58
	$K1$	$S4$	<u>59</u>, <u>1</u>, 36, 27, <u>18</u>, 41
	$K2$	$S5$	14, 20, 29
	$K16$	$S5$	30, 5, 47, 62, 45, 12
	Parity		[55, 33, 23, 54, 42, 30, 61, 43, 29]
Flipped λ_4	$K1$	$S5$	22, 28, 39, 54, 37, 4
	$K15$	$S5$	38, 13, 53
	$K16$	$S1$	<u>18</u>, <u>59</u>, 42, <u>3</u>, 57, <u>25</u>
	$K16$	$S3$	11, 43, <u>34</u>, 33, 52, <u>1</u>
	$K16$	$S4$	2, 9, <u>44</u>, <u>35</u>, <u>26</u>, <u>49</u>
	Parity		[4, 49, 39, 7, 50, 46, 14, 59, 45]

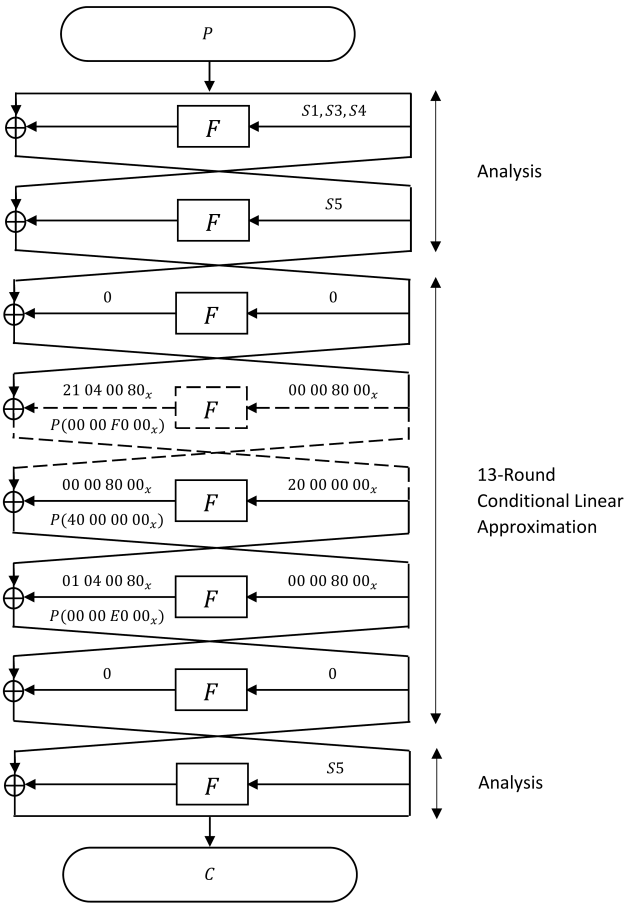

Figure 13: Outline of the 16-Round Attack (Using λ_4)

Table 16: Conditioning $\varepsilon(\lambda_1)$ on Each of the 8 Values of the 3 Selected Input Bits of $S5$

	000	001	010	011	100	101	110	111
λ_1 holds	0	4	4	0	2	0	0	2
λ_1 fails	8	4	4	8	6	8	8	6
Bias	–0.5	0	0	–0.5	–0.25	–0.5	–0.5	–0.25

reader can easily deduce the latter algorithm from the former by exchanging the roles of the plaintext and the ciphertext and the order of subkeys in the same algorithm.

The attack algorithm using λ_4 applies analysis on the second round based on a special linear approximation of $S5$: $10_x \to F_x$ (which we called earlier λ_1) conditioned on the LSB of the output and on 3 input bits. In addition, key bits of the first round and last round are guessed in the standard way. Our complexity is gained by a combination of the improved bias of the conditional linear approximation, and the best known algorithms for analysis, including using FFT [CSQ07].

4.1 Our Special Conditional Linear Approximation of Round 2

In this attack we perform a special analysis of the S box $S5$ in Round 2. The full analysis that requires the full knowledge of it's six input bits and guessing six key bits for it as well as six bits for each of six S boxes in the first round is too inefficient for our attack. Instead, we consider only three input bits, and keep the other three inconclusive. This setting allows us to guess fewer key bits in Round 2, and moreover, guess key bits of fewer S boxes in Round 1, leading to guess of $3 \cdot 6 + 3 = 21$ bits instead of $6 \cdot 6 + 6 = 42$ bits. On the other hand, the parity of the output of $S5$ cannot be directly computed by this setting.

We observe that we can approximate the parity of the four output bits given the values of these selected three input bits of $S5$, where in some cases the parity may be completely known, and in some it may only be statically correlated. The three selected input bits are the first input bit, the second input bit and the fifth input bit of $S5$, whose values depend on the outputs of $S1$, $S3$ and $S4$ in the first round. Table 16 shows the number of times λ_1 holds (or fails) for each value of these three input bits. Notice that the computation of these values is similar to the computation of linear approximation tables, counting the number of times the parity is 0 given the exact value of these three bits over the set of all the eight possibilities of the other three input bits. It is therefore that the notion of biases is relevant also in this case, and it can be computed for each of these eight input values. These biases are also given in Table 16.

Together with our condition on the LSB of the output of $S5$, we get the biases as listed in Table 17. Notice that the columns of Table 17 correspond to the columns of Table 16, where the content is divided to the case the where LSB is 0, and to the case where the LSB is 1. In each column of Table 16 we have a total of 8 counts, and thus in each column of Table 17 we have a total of 8 counts distributed over the two possible values of the condition. In most cases 4 are counted for each value (either 0 or 1) of the condition, while in others it may be distributed as $5 + 3$ or $6 + 2$. The conditional biases are then computed over the 4, or over the 5, or 3 (etc.) cases of the corresponding entry. Notice that 14 out of the 16 conditional biases are ± 0.5.

We use these biases by summing biases up in our attack. In cases where the biases are only ± 0.5 this summing is equivalent to counting suggesting plaintexts, as is usually performed by Algorithm 2, where Algorithm 2 increments the counter for values corresponding to bias $+0.5$, and does not increments for bias -0.5. In our case we also sum biases which are between -0.5 and $+0.5$, thus extending Algorithm 2 with a more general counting scheme, using floating point counters.

Table 17: Conditioning $\varepsilon(\lambda_1)$ on Each of the 16 Values of the LSB of the Output and the Three Selected Input Bits of $S5$

	0000	0001	0010	0011	0100	0101	0110	0111
λ_1 holds	0	4	3	0	2	0	0	2
λ_1 fails	5	0	0	4	2	4	6	0
Bias	–0.5	0.5	0.5	–0.5	0	–0.5	–0.5	0.5

	1000	1001	1010	1011	1100	1101	1110	1111
λ_1 holds	0	0	1	0	0	0	0	0
λ_1 fails	3	4	4	4	4	4	2	6
Bias	–0.5	–0.5	–0.3	–0.5	–0.5	–0.5	–0.5	–0.5

Table 18: Four Cases for the Possible Values of $\delta_4^{\lambda_1,odd}$ and $\delta_4^{\lambda_4,even}$

	$\delta_4^{\lambda_1,odd}=0$	$\delta_4^{\lambda_1,odd}=1$
$\delta_4^{\lambda_4,even}=0$	condition bit $=0$ $\varepsilon(\lambda_4)>0$	condition bit $=1$ $\varepsilon(\lambda_4)>0$
$\delta_4^{\lambda_4,even}=1$	condition bit $=1$ $\varepsilon(\lambda_4)=0$	condition bit $=0$ $\varepsilon(\lambda_4)=0$

4.2 The Condition Bit

By guessing the 6 key bits related to $S5$ in Round 16, and partially decrypting $S5$, we get an estimated value for the LSB of the output of $S5$ in Round 16. From this estimated value and from the plaintext, we can calculate the estimated value of the XOR of the LSB of the output of $S5$ of all the even rounds between Rounds 2–15. We call this value the *condition bit*.

Let the single-round approximation λ_1 be set in the second round, and let the 13-round linear approximation λ_4 be set in Rounds 3–15. Then $\delta_4^{\lambda_1,odd}$ is the LSB of the output of $S5$ in the second round, and $\delta_4^{\lambda_4,even}$ is the XOR of the LSB of the outputs of $S5$ of all the even rounds between Rounds 3–15. Under this notations, the condition bit is equal to $\delta_4^{\lambda_1,odd} \oplus \delta_4^{\lambda_4,even}$. We have four cases for the possible values of $\delta_4^{\lambda_1,odd}$ and $\delta_4^{\lambda_4,even}$, as outlined in Table 18. We can learn from the table that only in the case that the value of $\delta_4^{\lambda_1,odd}$ is equal to the value of the condition bit, the bias of λ_4 is greater than 0. Therefore our attack takes into consideration only this case, and forces $\delta_4^{\lambda_1,odd}$ in Round 2 to be equal to the condition bit. This technique does not discard half of the data, and instead use the condition information to calculate the value of $\delta_4^{\lambda_1,odd}$ in Round 2 accordingly. This is equivalent to discarding data as otherwise we would have to try the two possible cases of $\delta_4^{\lambda_1,odd}$ in Round 2, each would inherit a value for $\delta_4^{\lambda_4,even}$ in Rounds 3–15, and only then we would discard half of these cases, leading to the same remaining values as in our method.

4.3 A Simplified Version of the Attack

We start by showing a pedagogical (but highly inefficient) simplified version of our attack, and later we show how to optimize it into an efficient equivalent version that has the exact

same outputs. We apply a variant of Algorithm 2 as follows:

1. Given n known plaintexts and their corresponding ciphertexts $C_i = DES_K(P_i)$, where K is the unknown secret key.

2. Initialize two arrays of floating point counters by zero:

   (a) 2^{27} floating point counters $A[k]$ $(0 \leq k < 2^{27})$ that correspond to the 2^{27} values of the 27 key bits deduced from λ_4 in Step 3.

   (b) 2^{27} floating point counters $B[l]$ $(0 \leq l < 2^{27})$ that correspond to the 2^{27} values of the 27 key bits deduced from the flipped λ_4 in Step 4.

   The exact corresponding bits are listed in Table 15. These counters sum the biases of Round 2, rather than the usual version where they count the number of suggesting plaintexts, because in some cases we use fractional counters.

3. For each plaintext P_i and the corresponding ciphertext C_i:

   (a) Try all the 2^{27} possible key candidates k (of which 18 bits correspond to the first round, 6 bits to the last round, and 3 bits to the second round, as listed in Table 15). For each value of k perform:

      i. Partially decrypt $S5$ in the last round by it's 6 corresponding key bits.

      ii. Partially encrypt $S1$, $S3$ and $S4$ in the first round by their 18 corresponding key bits.

      iii. Compute the value of the 3 input bits entering to $S5$ in the second round using the outputs of Step 3(a)ii.

      iv. Compute the value of the condition bit c, using P_i, C_i and the output of Step 3(a)i.

      v. According to the value of the 3 input bits and the condition bit, fetch the conditional bias of the second round from Table 17, and call it b. The sign of b represents the parity of the 3 input bits and the output bit of Round 2.

      vi. Calculate the partial parity $p = P_{iR}[3, 8, 14, 25] \oplus P_{iL}[17] \oplus F_1(P_{iR}, K1)[17] \oplus C_{iL}[8, 14, 25] \oplus F_{16}(C_{iR}, K16)[8, 14, 25]$ from P_i and C_i and from the values computed in Step 3(a)i and Step 3(a)ii.

      vii. Add the conditional bias to the counter $A[k]$ by $A[k] += (-1)^p \cdot b$.

4. Repeat Step 3 to compute the counters of $B[\,]$ using the flipped version of λ_4, while exchanging the roles of P_i and C_i (and reverse order of the subkeys).

5. Use the counters $A[\,]$ and $B[\,]$ to find the secret unknown key K:

   (a) Extract the parity bit for each of the 2^{27} cases deduced from λ_4 by the signs of $A[k]$. Positive values correspond to parity 0, and negative to parity 1.

   (b) Similarly extract the parity bit for each of the 2^{27} cases deduced from the flipped λ_4 by the signs of $B[l]$.

   (c) Join $A[\,]$ and $B[\,]$ together by the value of the 10 common bits into an array of $2^{44} = 2^{27} \cdot 2^{27} \cdot 2^{-10}$ entries, each corresponds to a single value of the $44 = 27 + 27 - 10$ key bits. The value in each entry is the absolute value of the product of the values in $A[\,]$ and $B[\,]$ that correspond to this entry.

   (d) Sort the 2^{44} values in descending order, while keeping the 46 key bits (the 44-bit index and the 2 parity bits) along with the sorted values.

(e) For each of the 2^{44} cases in the sorted list try all the 2^{10} options of the $10 = 56 - 44 - 2$ remaining unknown key bits, therefore computing 56-bit trial keys. For each trial key, perform a trial encryption. If it succeeds, try a second plaintext, and if it also succeeds, announce that the found key is the unknown secret key K, and stop.

6. (optional) If all the $2^{44} \cdot 2^{10}$ trial encryptions failed, perform an exhaustive search of the $3 \cdot 2^{54}$ remaining untested keys.

The sums of biases in $A[\,]$ and $B[\,]$ correspond to the counters usually used in Algorithm 2. The highest the absolute value of the bias, the more probable that the corresponding key is correct. Since we use two linear approximations, we multiply their biases, and start trying the values with the highest product of both biases. Each such value is computed with all the 2^{10} options of the 10 extra bits and trial encryptions are performed on the resultant full keys. Also, since our (optimized) analysis is relatively cheap, we can afford trying more keys than earlier attacks did, and use them to improve our success rate.

4.4 Optimization of Steps 2–4

We optimize Steps 2–4 separately from optimizing Step 5. In principle we could apply any one of these two optimization independently, as in between Step 4 and Step 5 of the optimized algorithm we get exactly the same values of $A[\,]$ and $B[\,]$ as in the original algorithm. We use the FFT method [CSQ07] in our attack, which allows us to increase the number of guessed key bits during the attack without falling into a huge complexity of analysis.

In addition, we extend the FFT method to guess key bits in the second round in addition to the first and last rounds, although these bits are non-linear in the plaintext and the ciphertext, as well as to address condition bits via "virtual" key bits. To handle the fact that some of the plaintext bits are XORed with two different key bits during the encryption, we duplicate these plaintext bits. Therefore, we consider 27 effective text bits of λ_4, and 27 effective text bits of the flipped λ_4. In order to condition the approximation on the condition bit, we used a virtual key bit increasing the number of effective key bits to 28. We also show that we can compact two computations related to two linear approximations in one FFT computation as long as the circulant matrix is the same, by placing one in the real part and the other in the imaginary part.

The optimized algorithm creates a new preprocessing Step 0', and replaces Steps 2–4 by new Steps 2'–4':

0'. Preprocessing Step: Compute the first row of the circulant matrix:

(a) Let F be a vector of size 2^{28} complex values initialized by zeroes.

(b) For all 28-bit key values v:

i. Partially encrypt and decrypt P_0 and C_0, the zero plaintext/ciphertext pair (where both are considered zero) by the key bits v.

ii. Fetch the conditional bias b of the second round from Table 17, according to the value of the 3 input bits and the condition bit (see the original algorithm for details). Notice that the condition bit is first XORed with the 28th "virtual" key bit.

iii. Calculate the partial parity $p = F_1(P_{0R}, K1)[17] \oplus F_{16}(C_{0R}, K16)[8, 14, 25]$.

iv. Set $F[v] = (-1)^p \cdot b$.

(c) Compute $F_{FFT}[\,] = FFT(F[\,])$.

This preprocessing step takes less than 10 seconds to compute on a modern single core. Due to this speed, the implementor of a practical attack may wish to compute it as part of Step 4'.

2'. Initialize an array $C[\,]$ of 2^{28} complex counters by zero:

   (a) The real part corresponds to the values of the 27 effective text bits of λ_4 plus the condition bit of λ_4.

   (b) The imaginary part corresponds to the values of the 27 effective text bits of λ_4 plus the condition bit of the flipped λ_4.

3'. For each plaintext P_i and the corresponding ciphertext C_i:

   (a) Compute u_1, a 28-bit value consisting of the value of the 27 effective text bits of λ_4 plus the parity of the bits related to the condition bit of λ_4.

   (b) Compute u_2, a 28-bit value consisting of the value of the 27 effective text bits of the flipped λ_4 plus the parity of the bits related to the condition bit of the flipped λ_4.

   (c) Increment $C[u_1]$ by 1, and $C[u_2]$ by $\sqrt{-1}$ (i.e., increment the imaginary part by 1).

4'. Apply a 28-dimensional FFT method:

   (a) Compute $C_{FFT}[\,] = FFT(C[\,])$.

   (b) Compute $G_{FFT}[\,] = C_{FFT}[\,] \circ F_{FFT}[\,]$ (where $\circ$ is the Hadamard product, i.e., entrywise product).

   (c) Compute $G[\,] = FFT^{-1}(G_{FFT}[\,])/2^{28}$.

   (d) Split $G[\,]$ into $A[\,]$ and $B[\,]$, where $A[\,]$ consists of the real parts of the first 2^{27} elements of $G[\,]$, and $B[\,]$ consists of the imaginary parts of the same elements of $G[\,]$ (i.e., $G[j] = A[j] + \sqrt{-1}B[j]$ $(0 \leq j < 2^{27})$.

Let us address some issues regarding the implementation of the FFT technique:

1. The FFT technique requires that the number of analyzed key bits is equal to the number of bits of plaintext and ciphertext (or parities thereof) that affect the computation. One of them is the number of rows and the other the number of columns in the (square) circulant matrix. Moreover, each such key bit is directly related to a plaintext or ciphertext bit or a parity bit of some subset of them. Therefore, whenever the same plaintext bit is duplicated and then XORed with a different key bit, the plaintext bit must also be duplicated, though it is clear that it's two copies are always equal. On the other hand, when a parity of some subset of plaintext and ciphertext bits does not have a corresponding key bit, such a bit must virtually be created. This latter case occurs with the condition bit of our attack, and thus we create a virtual key bit, virtually XORed to the data during computation of the condition bit, whose real value is known to be 0, but for sake of computing the circulant matrix we must consider it to be 1 as well.

   At the end of the attack, when we extract candidate keys based on the corresponding values in the $A[\,]$ and $B[\,]$ arrays, there is no further need for the virtual key bit, and we fix it back to zero.

2. Also note that the FFT technique as presented in [CSQ07] assumes that all key bits are related to the first and last rounds. We on the other hand also consider such bits in the second round, though the inputs related to them there are not linear in the plaintext bits.

3. As the FFT array related to λ_4 and it's counterpart related to the flipped λ_4 are the same, we compute only one of them. The counters of the corresponding plaintexts or ciphertexts that we count in the $C[\,]$ array are different for λ_4 and the flipped λ_4. But as the counters are real (not complex), we can keep the two sets of counters in a single array. The real part corresponds to the counters related to λ_4 and the imaginary part corresponds to the counters related to the flipped λ_4. The output of Step 4' is therefore holding the two sets of biases. The real part corresponds to the biases related to λ_4, and the imaginary part to those related to the flipped λ_4. Notice that as the output of the FFT method requires normalization, we divide all entries of the output by 2^{28} in Step 4'c.

4. The FFT technique as presented in Step 0' and 4' requires six complex arrays of size 2^{28}. In practice it is possible to use only three such arrays by merging $F[\,]$, $C[\,]$ and $G[\,]$ into the same memory space, and merging C_{FFT} and G_{FFT} into the same memory space. As a complex variable is represented as two 64-bit reals, the total amount of memory required is $3 \cdot 2^{28} \cdot 2 \cdot 8 = 12$GB, which is commonly available in today's computers.

5. The resulting counters in $A[\,]$ and $B[\,]$ are exactly the same as after Step 4 of the simplified algorithm.

4.5 Optimization of Step 5

Replace Step 5 by the following Step 5':

5'. Use the counters $A[\,]$ and $B[\,]$ to find the secret unknown key K:

   (a) Extract the parity bit for each of the 2^{27} cases deduced from λ_4 by the signs of $A[k]$. Positive values correspond to parity 0, and negative to parity 1.

   (b) Similarly, extract the parity bit for each of the 2^{27} cases deduced from the flipped λ_4 by the signs of $B[l]$.

   (c) Split $A[\,]$ into 2^{10} arrays $A_m[\,]$ of size 2^{17} whose indices $A_m[s]$ ($0 \le m < 2^{10}$, $0 \le s < 2^{17}$) are partitioned between the common and non-common bits: m is the value of the 10 common bits, s is the value of the 17 non-common bits, and $A_m[s]$ is a structure containing the absolute value of the counter corresponding to these 27 bits, the parity bit, and the 17 non-common bits.

   (d) Similarly, split $B[\,]$ into 2^{10} arrays $B_m[\,]$ of size 2^{17} whose indices $B_m[t]$ ($0 \le m < 2^{10}$, $0 \le t < 2^{17}$) are partitioned between the common and non-common bits: m is the value of the 10 common bits, t is the value of the 17 non-common bits, and $B_m[t]$ is a structure containing the absolute value of the counter corresponding to these 27 bits, the parity bit, and the 17 non-common bits.

   (e) Sort the 2^{17} entries in each of the arrays $A_m[\,]$ and $B_m[\,]$ in descending order of the absolute counter.

   (f) Initialize an empty max heap (see [CLRS01, Chapter 6]) of size 2^{27}, whose nodes (m, s, t, v) will contain a value v and three indices: m, s and t. The max heap pop operation will extract the node with the highest value v, and output the indices (m, s, t) to the caller.

   (g) Create and insert to the max heap all the 2^{27} nodes whose $t = 0$, i.e., of the form $(m, s, 0, A_m[s] \cdot B_m[0])$. Notice that $t = 0$ in all of them.

   (h) Repeat as long as the heap is not empty and the key is not found:

      i. Pop the maximal node from the heap: (m, s, t).

ii. Try all the 2^{10} options of the $10 = 56 - 44 - 2$ remaining unknown key bits (not included in m, $A_m[s]$ and $B_m[t]$), therefore computing 56-bit trial keys. For each trial key, perform a trial encryption. If it succeeds, try a second plaintext, and if it also succeeds, announce that the found key is the unknown secret key K, and stop.

iii. If $t < 2^{17} - 1$, create a new node $(m, s, t + 1, A_m[s] \cdot B_m[t + 1])$ and insert to the heap.

This step joins the arrays $A[\,]$ and $B[\,]$ by their 10 common bits: we first divide each of them to 2^{10} buckets by these 10 common bits, and then prepare the cartesian product of each of the buckets in $A[\,]$ by the corresponding bucket in $B[\,]$. Each such cartesian product is of size $2^{17} \cdot 2^{17}$, but at every moment in time only the yet unprocessed entry with the highest value of every column is kept in memory.

Since the columns are sorted by decreasing order, the next highest value in the column is the one selected to replace a processed entry. And since the highest unprocessed value in each column is in the heap, the highest value that is not yes processed is at the top of the heap. Therefore, the entries are processed by a decreasing order of the absolute value of the product of their biases in $A[\,]$ and $B[\,]$.

4.6 Complexity of the Attack

The complexity of the optimized attack is as follows:

Step 0' partially encrypts four S boxes in the first and last rounds, and accesses Table 17 for $S5$ in Round 2. Together Step 0' performs $5 \cdot 2^{28}$ table lookups, which is equivalent to about $5 \cdot 2^{28}/128 \approx 2^{23}$ DES operations. Then Step 0' computes a 28-dimensional FFT, whose time is about $28 \cdot 2^{28}$ operations. The total time of Step 0' on our computer is less than 10 seconds on a single core.

Step 2' initializes an array of 2^{28} complex numbers by zeros, which takes about one second. Step 3' performs two counter increments for each of the given known plaintexts. For 2^{42} known plaintexts this means about 2^{43} increments which is faster than $2^{43}/128 = 2^{36}$ DES operations. Step 4' performs two FFT computations, along with 2^{28} multiplications, which take together less than 20 seconds on a single core.

Step 5' is more complicated to analyze, as it's time depends on the success of the attack. Steps 5'a–5'd are fixed-time steps taking about one second of computation. The sorting in Step 5'e performs 2^{11} sort of 2^{17} values each, thus costs about $2^{11} \cdot 17 \cdot 2^{17} \approx 2^{32}$ operations, which are equivalent to less than $2^{32}/128 = 2^{25}$ DES operations. Step 5'f is instantaneous. Step 5'g performs 2^{27} multiplications and 2^{27} insert operations to the heap.

Step 5'h is the key search itself. It find the unknown key if the parity bits were predicted correctly. It's time is exactly the number of keys tried till the unknown key is found. Notice that the overhead of the heap pop and insert operations is negligible, as one pop and one insert operations are performed for every 2^{10} DES operations. Assuming that the pop and insert operations take together about 100 CPU clock cycles, and a single DES trial encryption takes about 500 cycles, the overhead of the heap operations is about $100/(500 \cdot 1024) \approx 0.0002 = 0.02\%$.

We expect that in most cases Step 6 will not be reached. However, in cases where Step 5' does not find the key, the additional complexity of Step 6 is about $3 \cdot 2^{53}$ on average. We see that if Step 5' finds the unknown key, the complexity of Steps 0'–4' is small compared to the computation of Step 5', and Step 6 is not performed.

4.7 The Computer Experiment

We created a special version of the attack that implements all the above steps, except for the replacement of Steps 5' and 6 by a faster version that counts the number of keys

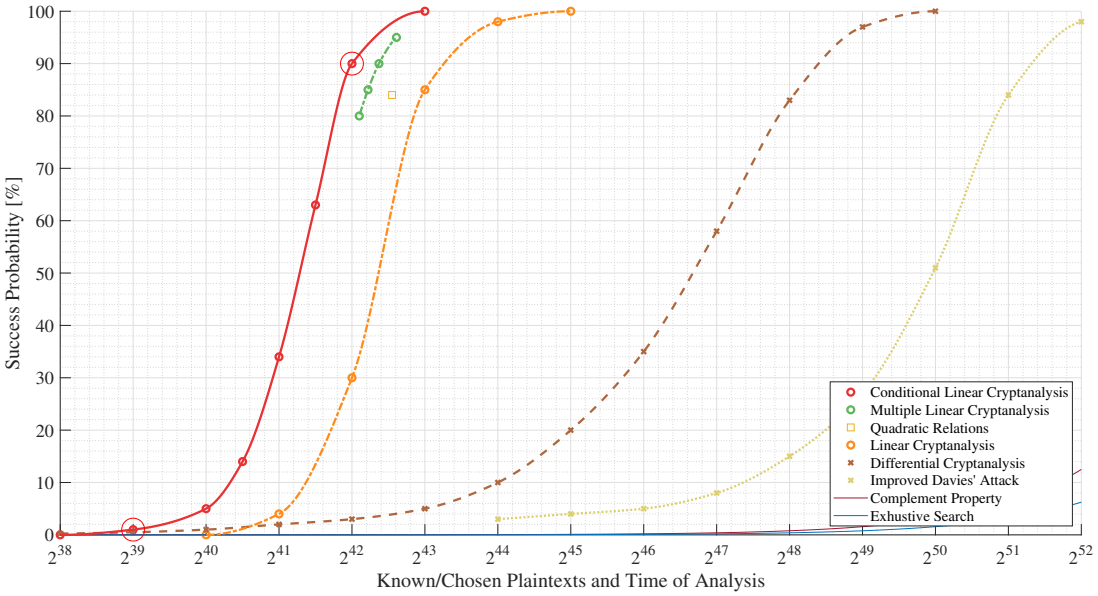

Figure 14: Comparison of the Success Probability by Complexity of the Attack (The marked points were experimentally verified on 79 runs of our attack on the full 16-round DES).

tried before the correct unknown key is found (instead of the slower search for the key). The complexity of the altered steps is about 2^{28}, and the whole program takes less than one minute on a single core, using 12 GB of RAM. Based on Matsui technique for basing the complexity claims for 16 rounds on the results on 8-round DES we ran our program on a slightly modified 8-round version of DES. Matsui needed $1.49 \cdot 2^{17} \approx$195,000 known plaintexts for his 8-round DES, from which he deduced the need for 2^{43} plaintexts on 16-round DES. We use a similar number of known plaintexts for our 2^{43} results, and a relative adjusted fraction for the other results. To avoid any difference from the case of 16-round DES, we use a slightly modified version of 8-round DES whose key schedule is partly modified so that the subkeys $K7$ and $K8$ are replaced by $K15$ and $K16$.[5]

The complexity of our attack as found by our test programs is given in Table 1 and Figure 14. Both also compare our results to previously published results.[6] We also present Table 2 and Figure 15 in which we let the computation time be increased to 2^{50} in order to reduce the number of known plaintexts. In this figure we also compare our results to two previously published attacks that quoted complexities for this case.

Figure 16 presents several curves of the success probability of conditional linear cryptanalysis by time of analysis, where each curve is computed for a different amount of known plaintexts. For comparison, Figure 17 presents a similar graph for standard linear cryptanalysis.

[5]The common bits of $K7$ and $K8$ with $K1$ and $K2$ are different than the common bits of $K15$ and $K16$ with $K1$ and $K2$. Moreover, even the number of the common bits is different. Unlike in Matsui's case, this difference matters to our attack, as in our case some of these bits are the bits common to the analysis using λ_4 and using the flipped λ_4.

[6]It is interesting to note that at low amounts of data differential cryptanalysis is the best attack. As can be seen in Figure 14 it surpasses the success rate of linear cryptanalysis below 2^{41}, and surpasses the success rate of conditional linear cryptanalysis below 2^{39}.

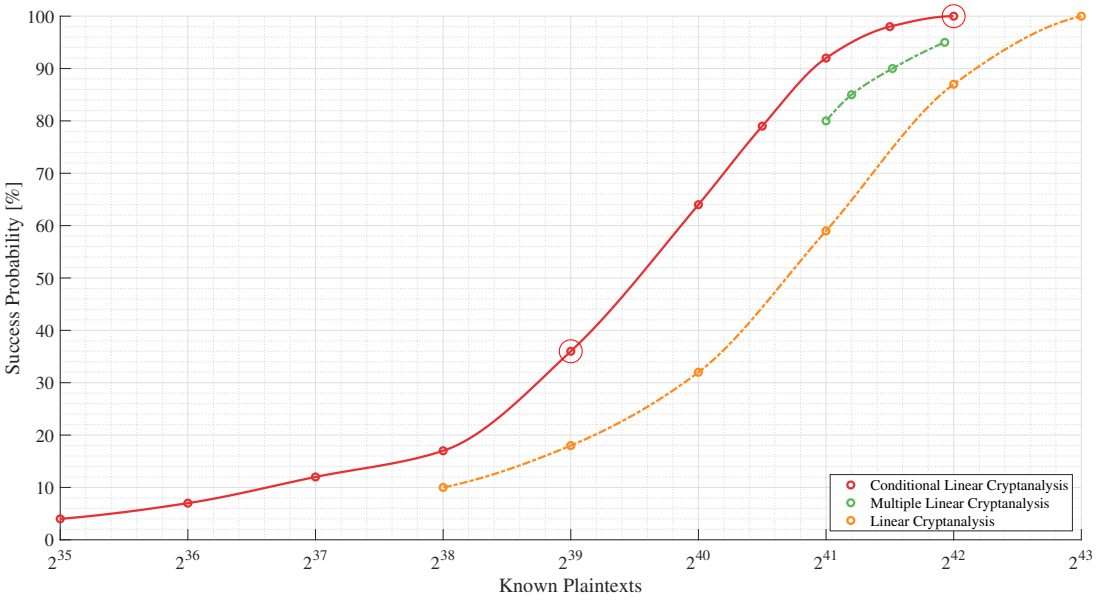

Figure 15: Comparison of the Success Probability by Number of Plaintexts for 2^{50} Time of Analysis (The marked points were experimentally verified on 79 runs of our attack on the full 16-round DES).

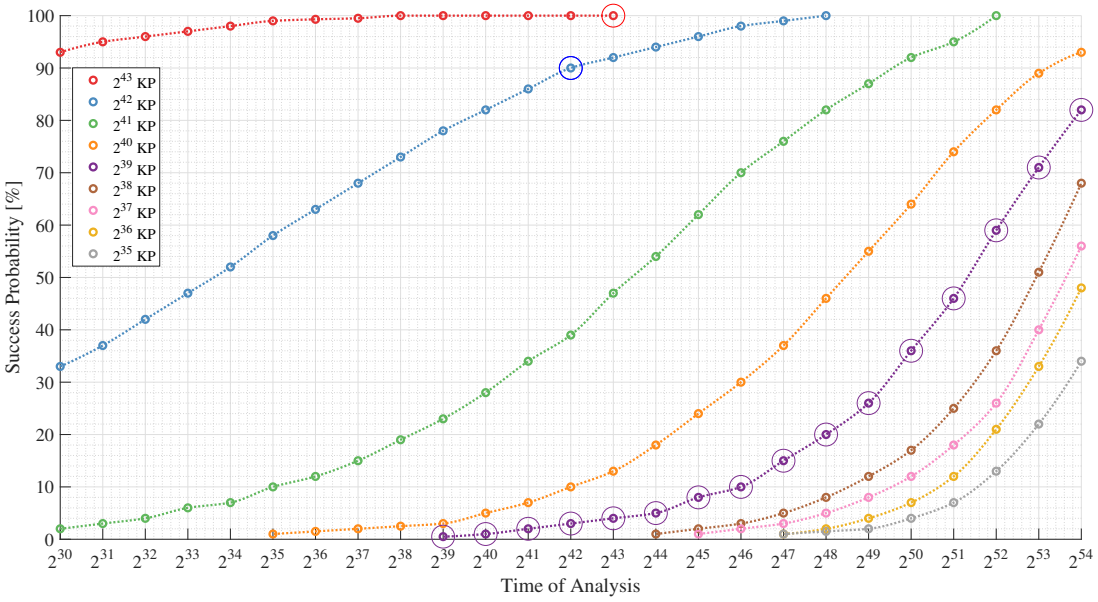

Figure 16: Success Probability of Conditional Linear Cryptanalysis by Known Plaintexts and Time of Analysis (The marked points were experimentally verified on 79 runs of our attack on the full 16-round DES).

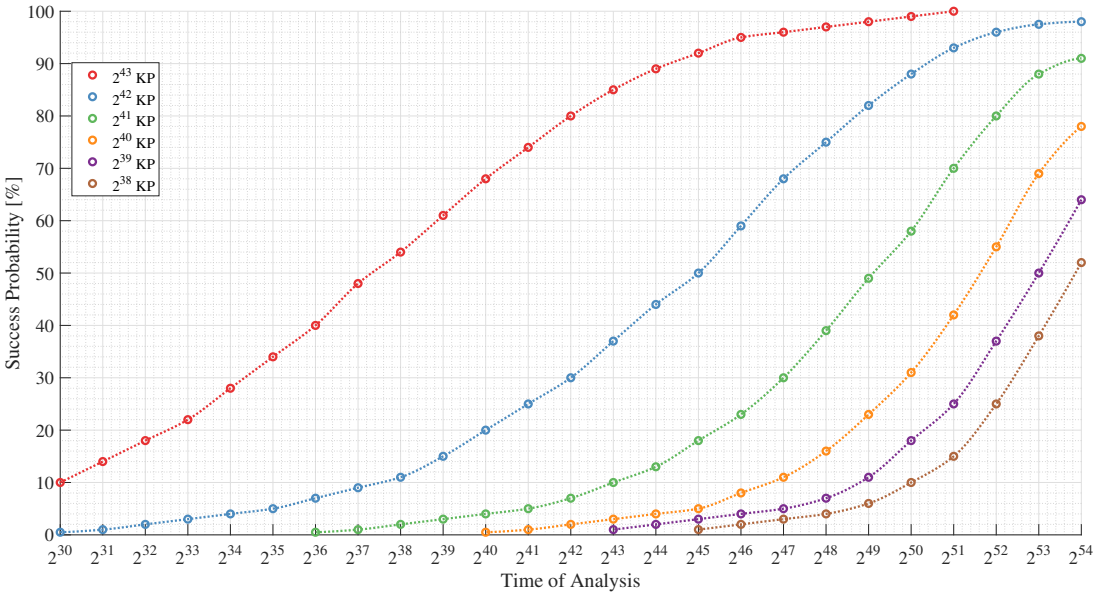

Figure 17: Success Probability of Linear Cryptanalysis by Known Plaintexts and Time of Analysis

4.7.1 Direct Implementation on the Full DES

We have adapted our implementation to work on the full DES directly, but due to longer computation times we were running only several dozens of experiments. The data collection phase takes a few hours when running in parallel on six computers in our lab with a total of 48 threads. Then, the phase that finds the actual time of key recovery takes about a minute on a single thread. The application of the full key recovery step to find the key would take a few more hours on the labs' computers. We had run 79 instances of this attack on our lab computers. The actual results fit to the prediction of Table 1 and Figures 14, 15 and 16.

5 Search for Conditional Approximations

There are several strategies to find the best conditional linear approximations. We searched for the best of the "intuitive" types, and types similar to the best regular ones. Most approximations shown in this paper were found manually based on heuristics, either similar to known (non-conditional) linear approximation or based on good conditional properties. For some kinds of conditional approximations (e.g., those used in Section 3 and in Section 4) it is possible to search automatically by a search program similar to Matsui's search algorithm for best approximations with adoptions to conditional cases. The algorithm is more complex, as it requires to consider two approximations at the same time. On the other hand, the computation time of this algorithm is highly dependent on the approximations used as it's starting point. In our case, known non-conditional approximations may serve as a good starting point for the search, which will save a lot of computation time.

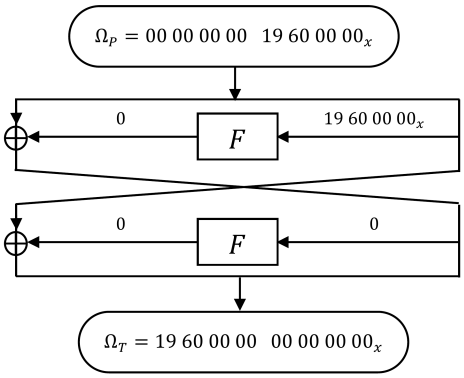

Figure 18: The Iterative Differential Characteristic of DES (Ω_1)

6 Application to Differential Cryptanalysis

The idea of conditioning approximation on other approximations can be extended to differential cryptanalysis by conditioning characteristics (or differentials) on approximations. The improvement in complexity in this case is relatively small (only second-order saving), because (unlike in the case of linear cryptanalysis) the increase in probability is linearly correlated to the decrease in the number of required data. So the same order of data may be required before discarding the ones that fail the conditions. But some second-order saving may still be possible.

Notice also that conditioning characteristic on their corresponding differentials is a standard (unnamed) practice, that is the base for example for $0R$-attacks (which expect for the internal differences with a very high probability after filtering out wrong external differences).

In addition, conditioning characteristics on key bits (or parities of key bits) was also shown in the past (e.g., in [BB93]), but in that case the usage is not for filtering out wrong external differences but for making a better prediction of the probability of the characteristic.

6.1 Saving Data Using Conditions

Another interesting application of our conditioning technique to differential cryptanalysis may allow us to discard some of the wrong pairs in advance before making any analysis of the differences, or even when we receive the first member of the pair while the other member is not yet known. In this latter case, the attacker can save the request for encrypting the second member of the pair, if he knows for sure (or having a good statistical evidence) that the pair cannot be (or is unlikely to be) a right pair.

For example, we concentrate on the two-round iterative characteristic (Ω_1) of DES, which is outlined in Figure 18. Table 19 shows the probabilities of Ω_1 conditioned on each of the 16 values of $\delta^{odd}\{S2\}$, where $\delta^{odd}\{S2\}$ in this case is the output of $S2$ in the first round (which can be directly computed from the plaintext and the ciphertext). In case of a two-round reduced DES, when we condition this characteristic on the 16 values of the output of $S2$, the right output difference comes with only four possible outputs: 1_x, 7_x, A_x or E_x. Therefore, it suffices to check only one member of the pair in order to discard some of the wrong pairs.

When extending this observation to longer instances of the iterative characteristic the number of possible values may grow and the relative probability of the possible values may increase or decrease. In the case of a 16-round characteristic (that iterates Ω_1 eight times) the number of the possible values grow to 8, and the number of impossible values reduces

Table 19: Conditioning the Probability of Ω_1 on Each of the 16 Possible Values of $\delta^{odd}\{S2\}$

$\delta^{odd}\{S2\}$	Prob	$\delta^{odd}\{S2\}$	Prob
0000	0	1000	0
0001	$2^{-5.87}$	1001	0
0010	0	1010	$2^{-5.87}$
0011	0	1011	0
0100	0	1100	0
0101	0	1101	0
0110	0	1110	$2^{-5.87}$
0111	$2^{-5.87}$	1111	0

Table 20: Conditioning the Probability of the 16-Round Iterative Characteristic on Each of the 16 Possible Values of $\delta^{odd}\{S2\}$

$\delta^{odd}\{S2\}$	Prob	$\delta^{odd}\{S2\}$	Prob
0000	$2^{-61.97}$	1000	0
0001	0	1001	$2^{-61.97}$
0010	$2^{-61.97}$	1010	0
0011	0	1011	$2^{-61.97}$
0100	$2^{-61.97}$	1100	0
0101	0	1101	$2^{-61.97}$
0110	$2^{-61.97}$	1110	0
0111	0	1111	$2^{-61.97}$

to 8. Table 20 shows the probabilities of this 16-round iterative characteristic conditioned on each of the 16 values of $\delta^{odd}\{S2\}$, where $\delta^{odd}\{S2\}$ is the XOR of the outputs of $S2$ in the odd rounds (which can be directly computed from the plaintext and the ciphertext). Note that a similar conditioning by the four bits related to $S1$ (or $S3$), does not provide impossible cases, but instead affects the probabilities of the 16 cases by an order of up to $\pm 10\%$.

From Table 20 it is also easy to see that the eight possible values are those whose LSB and MSB of $\delta^{odd}\{S2\}$ are equal. So the test for these cases is simply checking whether the XOR of the two corresponding plaintext bits is equal to the XOR of the two corresponding ciphertext bits.

Such information can be used to reduce the number of chosen plaintexts in several ways. The first and the simplest is that given an evidence, deduced from one member of the pair, that the pair cannot be a right pair, the attacker will not request the second member of the pair, and continue directly to request the next pair. This test can also be incorporated in the data collection phase, whenever possible. In such a case it saves transmission of both members of the pair to the attacker. In some other cases, the evidence may only be probabilistic, meaning the probability that a pair is expected to be a right pair may increase or decrease by the new evidence. Clearly, in such a case it may be more advantageous to start the analysis with the pairs with the higher probabilities if the order of analysis may affect the complexity, or to analyze only pairs with high probabilities (thus requiring fewer analyzed pairs), if the number of analyzed pairs affect the complexity.

Table 21: The Distribution of the Four Possible Events, in Case θ and λ Are Independent

	θ holds	θ fails
λ holds	$\frac{1}{2} \cdot (\frac{1}{2} + \varepsilon)$	$\frac{1}{2} \cdot (\frac{1}{2} + \varepsilon)$
λ fails	$\frac{1}{2} \cdot (\frac{1}{2} - \varepsilon)$	$\frac{1}{2} \cdot (\frac{1}{2} - \varepsilon)$

Table 22: An Example of a Dependent Distribution

	θ holds	θ fails
λ holds	$\frac{1}{2} \cdot (\frac{1}{2} + 2\varepsilon)$	$\frac{1}{2} \cdot \frac{1}{2}$
λ fails	$\frac{1}{2} \cdot (\frac{1}{2} - 2\varepsilon)$	$\frac{1}{2} \cdot \frac{1}{2}$

7 Relation to Linear Cryptanalysis with Multiple Approximations and to Multidimensional Linear Cryptanalysis

Prior studies of linear cryptanalysis with multiple approximations [BCQ04] considers statistically independent linear approximations. Multidimensional linear cryptanalysis [HCN09], on the other hand, can also handle the case of dependent linear approximations by considering the full distribution of 2^k events spanned by k linear approximations. Our extension considers highly dependent approximations, and discards many of the 2^k events by discarding their data, leading to a better ability of distinguishing between the remaining events. We can reduce the number of degrees of freedom and make the statistics more conclusive.

We demonstrate the difference between the approaches by the following example: consider two linear approximations, θ with probability $1/2$, and λ with probability $1/2 + \varepsilon$. According to Bayes' theorem,

$$Pr(\lambda, \theta) = Pr(\theta) \cdot Pr(\lambda|\theta).$$

In case θ and λ are independent, $Pr(\lambda|\theta) = Pr(\lambda)$. Table 21 shows the distribution of the $2^2 = 4$ possible events in case that θ and λ are independent.

We show that in some cases, the bias of λ may increase by a factor of about 2, when we restrict ourselves to cases when θ holds. Table 22 shows an example distribution of the same $2^2 = 4$ possible events in case where θ and λ are dependent. Notice that the bias of λ when θ holds is twice the original bias of λ. On the other hand, the events where θ fails show only zero bias, and is useless for linear cryptanalysis. Our extension discards these useless events by disregarding the data where θ fails. We keep the data with the bias that is twice the original bias of λ. In other examples we can even show even a larger increase in the conditional bias.

8 Summary

In this paper we showed that linear approximations are highly affected by conditioning them on other approximations. The simplest case is conditioning an approximation on a containing, longer, approximation, but it is also possible to condition on other, same length approximations, and even on approximations with bias 0. Furthermore, such conditional approximations can be combined in new ways to create unexpected conditional approximations. We showed how to use such conditional approximations for attacks, and that such conditions are especially effective in case of Feistel ciphers. We implemented our attack on the most widely studied Feistel cipher the Data Encryption Standard (DES) which lead to the best current attack against DES with complexity about $2^{41.9}$.

Acknowledgements

The authors would like to thank Kaisa Nyberg for various comments on early drafts, and in particular for constructing the conditional biases piling up lemma in the form given in this paper. The authors would also like to thank the anonymous reviewers for various challenging comments that greatly improved the content, scope, and presentation of this paper. This research was partially supported by the Technion Hiroshi Fujiwara cyber security research center and the Israel national cyber directorate.

References

[BB93] Ishai Ben-Aroya and Eli Biham. Differtial Cryptanalysis of Lucifer. *Advances in Cryptology — CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22–26, 1993, Proceedings*, pages 187–199, 1993.

[BB94] Eli Biham and Alex Biryukov. An Improvement of Davies' Attack on DES. *Advances in Cryptology — EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9–12, 1994, Proceedings*, pages 461–467, 1994.

[BC14] Eli Biham and Yaniv Carmeli. An Improvement of Linear Cryptanalysis with Addition Operations with Applications to FEAL-8X. *Selected Areas in Cryptography – SAC 2014 – 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, pages 59–76, 2014.

[BCQ04] Alex Biryukov, Christophe De Cannière, and Michaël Quisquater. On Multiple Linear Approximations. *Advances in Cryptology — CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15–19, 2004, Proceedings*, pages 1–22, 2004.

[Bih94] Eli Biham. On Matsui's Linear Cryptanalysis. *Advances in Cryptology — EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9–12, 1994, Proceedings*, pages 341–355, 1994.

[BS93] Eli Biham and Adi Shamir. Differential Cryptanalysis of the Data Encryption Standard. 1993. Available at http://www.cs.technion.ac.il/~biham/.

[BV17] Andrey Bogdanov and Philip S. Vejre. Linear Cryptanalysis of DES with Asymmetries. *Advances in Cryptology — ASIACRYPT 2017 — 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3–7, 2017, Proceedings, Part I*, pages 187–216, 2017.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. 2001.

[CSQ07] Baudoin Collard, François-Xavier Standaert, and Jean-Jacques Quisquater. Improving the Time Complexity of Matsui's Linear Cryptanalysis. *Information Security and Cryptology — ICISC 2007, 10th International Conference, Seoul, Korea, November 29-30, 2007, Proceedings*, pages 77–88, 2007.

[HCN09] Miia Hermelin, Joo Yeon Cho, and Kaisa Nyberg. Multidimensional Extension of Matsui's Algorithm 2. *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22–25, 2009, Revised Selected Papers*, pages 209–227, 2009.

[Jun01] Pascal Junod. On the Complexity of Matsui's Attack. *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers*, pages 199–211, 2001.

[KM00] Lars R. Knudsen and John Erik Mathiassen. A Chosen-Plaintext Linear Attack on DES. *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10–12, 2000, Proceedings*, pages 262–272, 2000.

[Mat93] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. *Advances in Cryptology — EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23–27, 1993, Proceedings*, pages 386–397, 1993.

[Mat94] Mitsuru Matsui. The First Experimental Cryptanalysis of the Data Encryption Standard. *Advances in Cryptology — CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21–25, 1994, Proceedings*, pages 1–11, 1994.

[MvOV01] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of Applied Cryptography. 2001.

[MY92] Mitsuru Matsui and Atsuhiro Yamagishi. A New Method for Known Plaintext Attack of FEAL Cipher. *Advances in Cryptology — EUROCRYPT '92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, May 24–28, 1992, Proceedings*, pages 81–91, 1992.

[Nat77] National Bureau of Standards. "Data Encryption Standard". *US. Department of Commerce, Federal Information Processing Standards Publication, FIPS 46*, 1977.

[Sha85] Adi Shamir. On the Security of DES. *Advances in Cryptology — CRYPTO '85, Santa Barbara, California, USA, August 18–22, 1985, Proceedings*, pages 280–281, 1985.

[SK98] Takeshi Shimoyama and Toshinobu Kaneko. Quadratic Relation of S-box and Its Application to the Linear Attack of Full Round DES. *Advances in Cryptology — CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23–27, 1998, Proceedingss*, pages 200–211, 1998.

[SM87] Akihiro Shimizu and Shoji Miyaguchi. Fast Data Encipherment Algorithm FEAL. *Advances in Cryptology — EUROCRYPT '87, Workshop on the Theory and Application of of Cryptographic Techniques, Amsterdam, The Netherlands, April 13–15, 1987, Proceedings*, pages 267–278, 1987.

[Vau96] Serge Vaudenay. An Experiment on DES Statistical Cryptanalysis. *CCS '96, Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, March 14–16, 1996*, pages 139–147, 1996.

A Scattered Linear Approximations

Traditionally, linear cryptanalysis assumes that linear approximations are built using a sequential process, in which there is a plaintext mask that is then approximated by a mask of the intermediate data after the first round through a linear approximation of the first round. The parity of the masked bits after the first round is expected to be equal to the parity of the masked bits of the plaintexts with the probability of the one-round

approximation. Then, the latter mask is approximated by a similar procedure by a mask of the intermediate data after the second round through a linear approximation of the second round. The probability of the two-round construction is computed by the piling up lemma from the probabilities of the two one-round approximations, where instead of using the probabilities directly, it is convenient to use biases, which are the probabilities minus half. Then the process repeats itself one more round at a time, till the full approximation is created.[7]

We propose here a different kind of linear approximations, which does not follow these steps, and cannot be outlined in the standard way we draw linear approximations. In particular, they have no corresponding plaintext and ciphertext masks that predict the parities of the corresponding plaintext and ciphertexts subsets of bits to be same (with the probability of the approximation).

We call this new approximations **scattered linear approximations** (and also scattered conditional linear approximations). Rather than predicting the parity of three subsets of bits (from the plaintext, ciphertext and subkeys) they predict the parity of subsets of four subsets, adding a new subset of intermediate data bits. For example, such an scattered approximation may predict the parity of the subset consisting of the 13th bit of the input of the F function in Round 4, the 17th bit in the output of F in Round 5, and the 17th bit in the output of F in Round 9 along with a few bits of the plaintext, ciphertext, and subkeys. Such predictions are impossible under the current formulation of linear cryptanalysis, and may even seem unuseful for attacking the cipher with the currently known attack algorithms, but clearly such predictions give us information that is available without them. Their importance will be decided by the ability to find such scattered approximations, whether their biases are large enough, and whether we can construct attack algorithms that use them.

A simple particular example of such scattered approximations follows from the conditions that we used earlier in the paper, i.e., the XOR of the output of the F function in all even rounds equals the right half of the plaintext XOR the left half of the ciphertext ($\bigoplus_{k\,is\,even} Y^k = P_R \oplus C_L$, see Equation 1 in Section 2.3). This equation predicts 32 parities of subsets of intermediate bits, where each of them predicts the parity of a subset of eight intermediate bits, one from each output of F in the even rounds. Clearly, the probability of this scattered approximation is 1 (bias 1/2). Though this approximation has so high probability, there is no way to write or draw it in the standard formalism of linear cryptanalysis, and there are no plaintext and ciphertext subsets whose bits are predicted to have equal parities.

In the next subsections we present examples of scattered approximations, how they can be concatenated, their applications. Since this paper is focused on conditional linear cryptanalysis, we mainly focus here on scattered approximations in the framework of conditional linear cryptanalysis, i.e., construct and use scattered conditional approximations.

A.1 A Short Approximation Conditioned by a Longer One

In this section we show that the bias of a short linear approximation can be improved by conditioning it on the parity of a longer linear approximation. For example, we show that the bias of the first eight rounds of Matsui's linear approximation conditioned on the parity of full 16 rounds of that approximation is about twice higher than the original bias of the same eight rounds.

A simple case of a linear approximation conditioned on a longer containing one, with same bias of the contained approximation and of the rest of the containing approximation, is described here-forth.[8] Consider an r-round containing linear approximation λ, which is

[7] There are shortcuts to this process that allow to concatenate longer approximations, but they are equivalent in structure to concatenating a single round at a time.

[8] The same phenomenon can be applied to cases with different biases, but the factor reduces below two.

Table 23: The Four Possible Cases of $\lambda^{1,m}$ and $\lambda^{m+1,r}$

	$\lambda^{1,m}$ holds $p = (\frac{1}{2} + \varepsilon)$	$\lambda^{1,m}$ fails $p = (\frac{1}{2} - \varepsilon)$
$\lambda^{m+1,r}$ holds $p = (\frac{1}{2} + \varepsilon)$	λ holds $p = (\frac{1}{2} + \varepsilon)^2$	λ fails $\rightarrow$ data discarded $p = (\frac{1}{2} + \varepsilon)(\frac{1}{2} - \varepsilon)$
$\lambda^{m+1,r}$ fails $p = (\frac{1}{2} - \varepsilon)$	λ fails $\rightarrow$ data discarded $p = (\frac{1}{2} - \varepsilon)(\frac{1}{2} + \varepsilon)$	λ holds $p = (\frac{1}{2} - \varepsilon)^2$

a concatenation of two linear approximations $\lambda^{1,m}$ and $\lambda^{m+1,r}$ (with m rounds and $r - m$ rounds, respectively), where the bias of $\lambda^{1,m}$ equals to the bias of $\lambda^{m+1,r}$. We denote this bias by ε. The condition that discards data in this case is based on the satisfaction of the (long) approximation λ. All the plaintexts/ciphertext pairs for which λ holds remain, while all others are discarded.

We observe that the bias of $\lambda^{1,m}$ in the remaining data is about twice the original biases. The simplest way to show it is by considering the four cases outlined in Table 23. In this table the primary diagonal corresponds to the remaining data, while the secondary diagonal corresponds to the discarded data. As can be seen there, the two discarded cases have exactly the same probability, where in one $\lambda^{1,m}$ holds, and in the other $\lambda^{1,m}$ fails. Therefore, the total bias of $\lambda^{1,m}$ in the discarded data is exactly 0. Since the original bias is the weighted average of the remaining and discarded cases, the conditional bias is higher than the original bias. In cases where ε is very small, the fraction of discarded data is almost exactly half, and thus the conditional bias becomes about twice larger than the original. Accurate calculations of the biases are presented at the end of this appendix.

Notice that the shorter approximation $\lambda^{1,m}$ has the standard form of a linear approximation, but also notice that once it is conditioned on the longer approximation λ, the analyzed cipher becomes longer than the number of rounds of $\lambda^{1,m}$, so $\lambda^{1,m}$ no longer spreads on all the rounds of the cipher. Moreover, the former mask at the ciphertext side becomes a mask on an intermediate data. It thus becomes a simple case of an scattered approximation.

A conditional piling up lemma is available also in this case. Let X_1, Y_1, Y_2 be binary random variables such that X_1 and Y_1 are independent of Y_2. Denote:

$$\begin{aligned} p_1 &= Pr(Y_1 = 0) \\ p_2 &= Pr(Y_2 = 0) \\ \alpha_1 &= \varepsilon(X_1|Y_1 = 0) \\ \beta_1 &= \varepsilon(X_1|Y_1 = 1) \end{aligned}$$

Lemma 3.

$$\varepsilon(X_1|Y_1 \oplus Y_2 = 0) = \frac{p_1 p_2 \alpha_1 + (1 - p_1)(1 - p_2)\beta_1}{p_1 p_2 + (1 - p_1)(1 - p_2)}.$$

The proofs is given in Appendix B.

A.1.1 Three-Round Examples

Consider a three-round cipher, and let λ_9 be the best three-round linear approximation, outlined in Figure 19. We can treat λ_9 as two concatenated approximations: $\lambda_9^{1,2}$ and λ_9^3, where $\lambda_9^{1,2}$ consists of the first two rounds of λ_9, and λ_9^3 consists of the last round of λ_9. For simplicity of discussion, assume (without loss of generality) that the key bit involved in $\lambda_9^{1,2}$ and the one involved in λ_9^3 are equal in value. Thus, the bias of $\lambda_9^{1,2}$ is equal to the bias of λ_9^3, and the biases equal $\varepsilon = -0.31$. The bias of the full λ_9 is $2\varepsilon^2 = 0.19$. Using

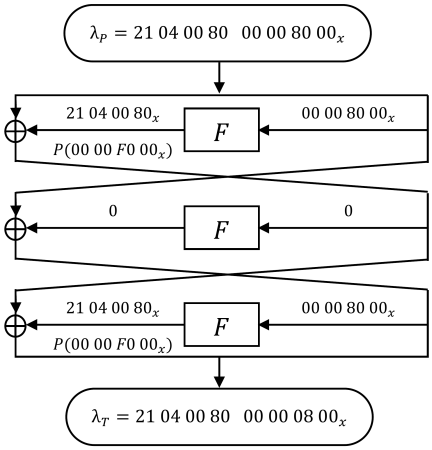

Figure 19: The Best Three-Round Linear Approximation (λ_9)

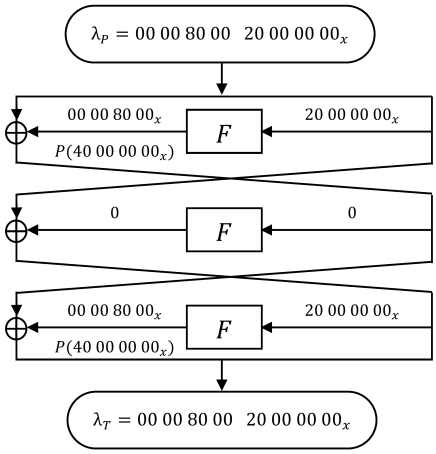

Figure 20: Another Three-Round Linear Approximation (λ_{10})

the formulae of Appendix A.5, we can compute the bias of $\lambda_9^{1,2}$ given the fact that the data satisfies the approximation λ_9 to be $\varepsilon(\lambda_9^{1,2}|\lambda_9 = 0) = \varepsilon^* = -0.45$. Since in this case the bias ε is relatively large, the factor of increase of the conditional bias, $\varepsilon^*/\varepsilon$ is only $\varepsilon^*/\varepsilon = 0.45/0.31 \approx 1.45$ (rather than 2).

Another example is λ_{10}, a three-round linear approximation outlined in Figure 20. The bias of $\lambda_{10}^{1,2}$ is equal to the bias of λ_{10}^{3}, both biases equal $\varepsilon = -\frac{2}{64} = -0.03125$. The bias of the full λ_{10} is $2\varepsilon^2 = 2(\frac{2}{64})^2 = 0.00195$. The bias of $\lambda_{10}^{1,2}$ given the fact that the data satisfies the approximation λ_{10} is $\varepsilon^* = \varepsilon(\lambda_{10}^{1,2}|\lambda_{10} = 0) = (\frac{1}{2} - \frac{2}{64})^2/\frac{1}{2} + 2(\frac{2}{64})^2 - \frac{1}{2} = -0.06225$. And $\varepsilon^*/\varepsilon = -0.06225/-0.03125 \approx 2$.

A.1.2 A 16-Round Example

Consider Matsui's eight-round iterative linear approximation as outlined in Figure 8, and denote its iteration to 16 rounds ($\lambda_3^{\otimes 16}$) by λ_{11}. The biases of these approximations are $2^{-13.7}$ and $2 \cdot (2^{-13.7})^2 = 2^{-26.4}$, respectively. Assuming that the cipher is the full DES, it is easy to divide the data into two sets based on the parity of λ_{11}, where in one set λ_{11} holds on all data, and in the other fails on all data.

In this case, the conditional bias of the eight-round approximation $\lambda_{11}^{1,8}$ when condition

on the longer 16-round approximation λ_{11} is almost doubled to

$$\varepsilon(\lambda_{11}^{1,8}|\lambda_{11}=0)=\frac{(\frac{1}{2}+2^{-13.7})^2}{\frac{1}{2}+2\cdot(2^{-13.7})^2}-\frac{1}{2}\approx 2^{-12.7}=2\cdot\varepsilon(\lambda_{11}^{1,8})$$

in one of the sets, and becomes

$$\varepsilon(\lambda_{11}^{1,8}|\lambda_{11}=1)=\frac{(\frac{1}{2}+2^{-13.7})\cdot(\frac{1}{2}-2^{-13.7})}{\frac{1}{2}-2\cdot(2^{-13.7})^2}-\frac{1}{2}=0$$

in the other.

A.2 Non-consecutive Active Rounds

Consider λ_7 from Section 3.6, and consider λ_8, which is its iteration to 8 rounds ($\lambda_7^{\otimes 8}$) conditioned as in Section 3.6. The biases of λ_7 and λ_8 are 0 but as mentioned in Section 3.6 the conditional biases are non zero. We start with a 8-round approximation, and condition it on $\theta_2^{\otimes 16}$.

We can calculate $\varepsilon(\lambda_8|\theta_2^{\otimes 16}=0)$ by applying the conditional biases piling up lemma:

$$\varepsilon(\lambda_8|\theta_2^{\otimes 16}=0)=\varepsilon(\lambda_8|\theta_2^{\otimes 8}\oplus\theta_2^{\otimes 8}=0).$$

In this case

$$\begin{aligned}p_1&=Pr(\theta_2^{\otimes 8}=0)=0.5000076\\ p_2&=Pr(\theta_2^{\otimes 8}=0)=0.5000076\\ \alpha_1&=2^{-18.6}\\ \beta_1&=-2^{-18.6}\end{aligned}$$

and then substitute them in the equation of Lemma 3

$$\begin{aligned}\varepsilon(\lambda_8|\theta_2^{\otimes 16}=0)&=\varepsilon(\lambda_8|\theta_2^{\otimes 8}\oplus\theta_2^{\otimes 8}=0)=\\ &=\frac{p_1p_2\alpha_1+(1-p_1)(1-p_2)\beta_1}{p_1p_2+(1-p_1)(1-p_2)}=\\ &=\frac{(0.5000076)^2\cdot 2^{-18.6}+(1-0.5000076)^2\cdot -2^{-18.6}}{(0.5000076)^2+(1-0.5000076)^2}=2^{-34.66}.\end{aligned}$$

This improved bias holds even in the more general scattered case where the shorter approximation is selected to have any (not necessarily consecutive) sets of rounds that fulfill the other criteria, e.g., rounds 3, 7, 11 and 15 (whose index is 3 modulo 4), instead of 1, 3, 5 and 7. We get a scattered conditional linear approximation with four active rounds distanced from each other by four rounds. Call it λ_{12}. It is outlined in Figure 21. It is a four-round iterative scattered conditional linear approximation. The intermediate bit that participates in the parity is marked in bold.

A.3 Conversion to Input-Bit Based Scattered Approximation

Due to the properties of a Feistel cipher the bit in the output of Round 3 of λ_{12} (marked in bold in Figure 21) equals the XOR of the input bits in the same location in Round 2 and Round 4. Similarly, the bit in Round 7 is the XOR of the corresponding input bits in rounds 6 and 8, and so on. Therefore, this parity bit of the output of the four rounds equals to the parity of the XOR of the corresponding input bits in all the even rounds. The resulting approximation is again a four-round iterative scattered conditional linear approximation. Call it λ_{13}. It is outlined in Figure 22. The intermediate approximated

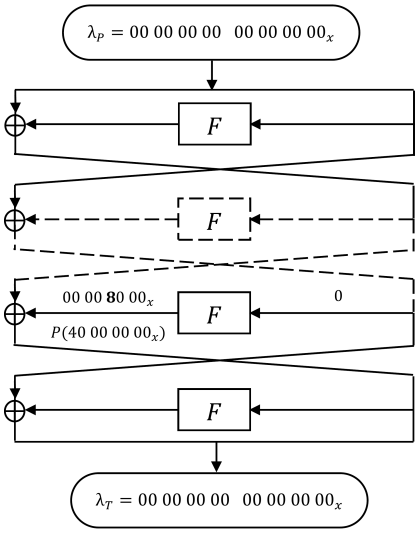

Figure 21: The Scattered Conditional Approximation λ_{12} (4 Rounds of 16)

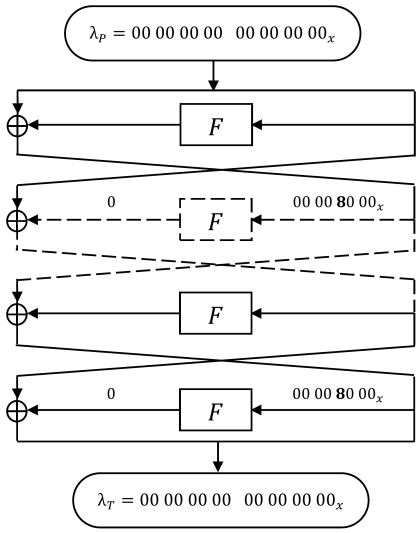

Figure 22: The Scattered Conditional Approximation λ_{13} (4 Rounds of 16)

bits are marked in bold. We emphasize that the bits of the parity approximation of λ_{13} correspond only to bits of the inputs of the F functions without any non-empty output mask. However since it is a scattered approximation it is not easy to use it directly for an attack.

A.4 Concatenating Scattered Approximations

It remains to discuss the concatenation of scattered approximations. In traditional approximations the approximation building process can be viewed as sequential, by setting a one-round approximation in the first round, and then concatenating to it another one-round approximation (that must satisfy the required concatenation relations), and so on till the full approximation is finished.

In scattered approximations the situation is not so structured. The trivial case is that any traditional approximation can be set anywhere through the cipher, and the result becomes a scattered approximation that approximates the parity of the intermediate bits where the original plaintext and ciphertext subsets are located.

Moreover, any two or more traditional approximations can be set anywhere (covering distinct rounds), and approximate the parity of the intermediate bit of all these approximations.[9]

In the more interesting case of concatenation, two scattered approximations that have no common active S boxes are concatenated together by combining each of their rounds with the corresponding round of the other, where the active S boxes take preference over inactive ones. The result is a scattered approximation which approximates the XOR of the parity formulae of both original scattered approximations, i.e., each intermediate (or plaintext, ciphertext or key) bit that belonged only to one of the two approximated parities remains in the approximated subset, while any bit that does not belong to any, or that belongs to both is not in the approximated subset.

In all these cases, the bias computation of the concatenated approximation follows the regular piling up lemma (or the conditional piling up lemma in case of conditional approximations).[10]

A.4.1 A Simple Example

Consider the scattered approximation where λ_1 is set in all the even rounds. Call it λ_{14} as outlined in Figure 23 (with approximated parity bits marked in bold). Since λ_1 is a non conditional approximation, so is λ_{14}. The bias of λ_{14} is $(-20/64)^8 \cdot 2^7 = 2^{-6.4}$. The XOR of all the active bits in the output of its F functions can be linearly computed from the plaintexts and ciphertexts. The active bits in the inputs of F are therefore the intermediate bits that cause this approximation to be scattered, while the scattered approximation approximates the parity of these eight input bits, four plaintext bits and four ciphertext bits.

A.4.2 Concatenating Two Scattered Approximations into a Non-Scattered Approximation

Consider two different scattered approximations that can be concatenated (i.e., have no common active S boxes). If all the intermediate bits in their predicted parity subsets are equal, then the concatenated approximation becomes non-scattered. The external subsets of bits are XORed (as subsets), resulting with a new subset that contains only the external bits that occurred only in one of the two concatenated approximations.

[9] We ignore the case of common active S boxes as in that case the joint bias may need different analysis, and as the predicted bits may not be predicted any more in the combined case.

[10] As long as the conditions are concatenatable to a single one, as required by our presented form the piling up lemma.

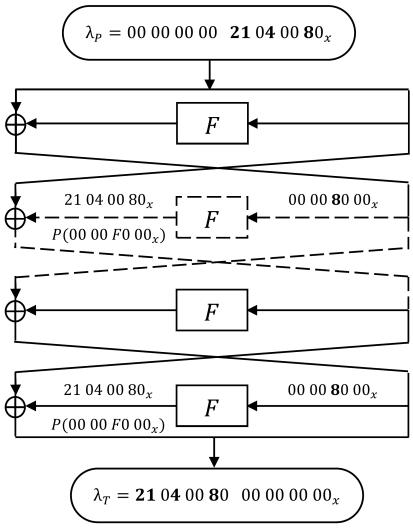

Figure 23: The Scattered approximation λ_{14} (4 Rounds of 16)

As a side note we refer the reader to a fact (that he had probably already observed) that λ_{13} and λ_{14} follow these requirements, having the same intermediate subset, but different external subsets. Therefore, they can be concatenated to a non-scattered approximation, which is outlined in Figure 24. The bias of this approximation is $2 \cdot \varepsilon(\lambda_{13}) \cdot \varepsilon(\lambda_{14}) = 2 \cdot 2^{-6.4} \cdot 2^{-34.66} = 2^{-40.08}$.

A.5 An Accurate Calculation of the Conditional Biases

Consider an r-round containing linear approximation λ, which is a concatenation of two linear approximations $\lambda^{1,m}$ and $\lambda^{m+1,r}$ (with m rounds and $r-m$ rounds, respectively), where the probability of $\lambda^{1,m}$ equals to the probability of $\lambda^{m+1,r}$. We denote this probability by $\frac{1}{2}+\varepsilon$.

A.5.1 The Exact Calculation

Notice that

- The probability that λ satisfies the parity approximation is $(\frac{1}{2}+\varepsilon)^2+(\frac{1}{2}-\varepsilon)^2 = \frac{1}{2}+2\varepsilon^2$.
- The probability that both $\lambda^{1,m}$ and $\lambda^{m+1,r}$ satisfy the parity approximations is $(\frac{1}{2}+\varepsilon)^2 = \frac{1}{4}+\varepsilon+\varepsilon^2$.

Given that λ satisfies the parity approximation, the probability that $\lambda^{1,m}$ (or $\lambda^{m+1,r}$) satisfies the parity approximation can be improved. Let $\varepsilon^* = \varepsilon(\lambda^{1,m}|\lambda=0)$. Therefore,

$$\frac{1}{2}+\varepsilon^* = \frac{\frac{1}{4}+\varepsilon+\varepsilon^2}{\frac{1}{2}+2\varepsilon^2} = \frac{\frac{1}{4}+\varepsilon^2}{\frac{1}{2}+2\varepsilon^2} + \frac{\varepsilon}{\frac{1}{2}+2\varepsilon^2} = \frac{1}{2}+\frac{\varepsilon}{\frac{1}{2}+2\varepsilon^2}.$$

The denominator in the above equation is the fraction of un-discarded plaintexts, i.e., those plaintexts in which both sub-approximations either both follow the approximation or both not follow the approximation. The nominator consists of only the plaintexts that follow the approximation in both sub-approximations, i.e., all cases that the first sub-approximation follows the approximation in the un-discarded data. Then,

$$\varepsilon^* = \frac{\varepsilon}{\frac{1}{2}+2\varepsilon^2},$$

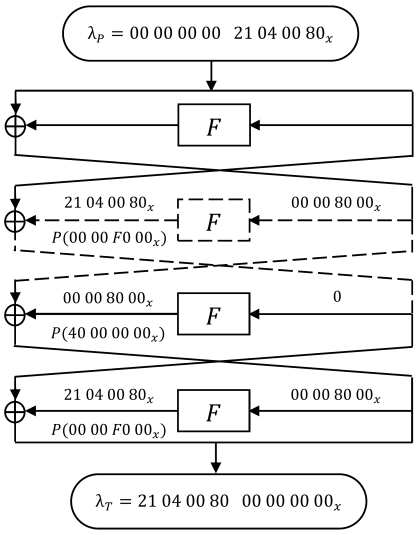

Figure 24: Four-Round Iterative Conditional Linear Approximation (the Concatenation of λ_{13} and λ_{14})

from which it is easy to deduce the fraction $\varepsilon^*/\varepsilon$

$$\frac{\varepsilon^*}{\varepsilon} = \frac{1}{\frac{1}{2} + 2\varepsilon^2}.$$

In most cases $\varepsilon \ll 1$, from which we can deduce that

$$\varepsilon^* \approx 2\varepsilon.$$

A.5.2 An Alternate Exact Calculation

We can alternatively calculate $\varepsilon^* = \varepsilon(\lambda^{1,m}|\lambda = 0)$ by applying the conditional biases piling up lemma:

$$\varepsilon^* = \varepsilon(\lambda^{1,m}|\lambda = 0) = \varepsilon(\lambda^{1,m}|\lambda^{1,m} \oplus \lambda^{m+1,r} = 0).$$

In this case we let

$$\begin{aligned} p_1 &= Pr(\lambda^{1,m} = 0) = \tfrac{1}{2} + \varepsilon \\ p_2 &= Pr(\lambda^{m+1,r} = 0) = \tfrac{1}{2} + \varepsilon \\ \alpha_1 &= \varepsilon(\lambda^{1,m}|\lambda^{1,m} = 0) = \tfrac{1}{2} \\ \beta_1 &= \varepsilon(\lambda^{1,m}|\lambda^{1,m} = 1) = -\tfrac{1}{2} \end{aligned}$$

and then substitute them in the equation of Lemma 3

$$\begin{aligned} \varepsilon^* &= \varepsilon(\lambda^{1,m}|\lambda^{1,m} \oplus \lambda^{m+1,r}) = \\ &= \frac{p_1p_2\alpha_1 + (1-p_1)(1-p_2)\beta_1}{p_1p_2 + (1-p_1)(1-p_2)} = \frac{(\frac{1}{2}+\varepsilon)\cdot(\frac{1}{2}+\varepsilon)\cdot\frac{1}{2} + (\frac{1}{2}-\varepsilon)\cdot(\frac{1}{2}-\varepsilon)\cdot-\frac{1}{2}}{(\frac{1}{2}+\varepsilon)\cdot(\frac{1}{2}+\varepsilon) + (\frac{1}{2}-\varepsilon)\cdot(\frac{1}{2}-\varepsilon)} = \\ &= \frac{\frac{1}{2}((\frac{1}{2}+\varepsilon)^2 - (\frac{1}{2}-\varepsilon)^2)}{(\frac{1}{2}+\varepsilon)^2 + (\frac{1}{2}-\varepsilon)^2} = \\ &= \frac{\varepsilon}{\frac{1}{2} + 2\varepsilon^2}. \end{aligned}$$

A.5.3 The Bias of the Discarded Data

The complementing case $\varepsilon(\lambda^{1,m}|\lambda=1)$ also deserve to be accurately calculated. Notice that in this case

- The probability that λ does not satisfy the parity approximation is $2\cdot(\frac{1}{2}+\varepsilon)\cdot(\frac{1}{2}-\varepsilon) = \frac{1}{2}-2\varepsilon^2$.
- The probability that $\lambda^{1,m}$ satisfies the parity approximation but $\lambda^{m+1,r}$ does not satisfy the parity approximation is $(\frac{1}{2}+\varepsilon)\cdot(\frac{1}{2}-\varepsilon) = \frac{1}{4}-\varepsilon^2$.

Given that λ does not satisfy the parity approximation, the probability that $\lambda^{1,m}$ (or $\lambda^{m+1,r}$) satisfies the parity approximation is

$$Pr(\lambda^{1,m}|\lambda=1) = (\frac{\frac{1}{4}-\varepsilon^2}{\frac{1}{2}-2\varepsilon^2}) = \frac{1}{2}.$$

Thus the bias is exactly zero:

$$\varepsilon(\lambda^{1,m}|\lambda=1) = Pr(\lambda^{1,m}|\lambda=1) - \frac{1}{2} = 0.$$

B Conditional Biases Piling Up Lemma

Let X and Y be binary random variables.

$\varepsilon(X)$	is the bias of X, i.e.: $\varepsilon(X) = Pr(X=0) - \frac{1}{2}$,
$\varepsilon(X\|Y=t)$	is the conditional bias of X under the event that $Y=t$, i.e.: $\varepsilon(X\|Y=t) = Pr(X=0\|Y=t) - \frac{1}{2}$.

Let X_1, X_2, Y_1, Y_2 be binary random variables such that X_1 and Y_1 are independent of X_2 and Y_2. Denote:

$$\begin{aligned}
p_1 &= Pr(Y_1=0)\\
p_2 &= Pr(Y_2=0)\\
\alpha_1 &= \varepsilon(X_1|Y_1=0)\\
\alpha_2 &= \varepsilon(X_2|Y_2=0)\\
\beta_1 &= \varepsilon(X_1|Y_1=1)\\
\beta_2 &= \varepsilon(X_2|Y_2=1)
\end{aligned}$$

Lemma 1.

$$\varepsilon(X_1\oplus X_2|Y_1\oplus Y_2=0) = \frac{2p_1p_2\alpha_1\alpha_2 + 2(1-p_1)(1-p_2)\beta_1\beta_2}{p_1p_2+(1-p_1)(1-p_2)}.$$

Proof.

$$
\begin{aligned}
Pr(Y_1 \oplus Y_2 = 0) &= p_1p_2 + (1-p_1)(1-p_2) \\
Pr(X_1 \oplus X_2 = 0, Y_1 \oplus Y_2 = 0) &= Pr(X_1 = X_2 = 0, Y_1 = Y_2 = 0) \\
&+ Pr(X_1 = X_2 = 1, Y_1 = Y_2 = 0) \\
&+ Pr(X_1 = X_2 = 0, Y_1 = Y_2 = 1) \\
&+ Pr(X_1 = X_2 = 1, Y_1 = Y_2 = 1) = \\
&= Pr(X_1 = 0, Y_1 = 0)Pr(X_2 = 0, Y_2 = 0) \\
&+ Pr(X_1 = 1, Y_1 = 0)Pr(X_2 = 1, Y_2 = 0) \\
&+ Pr(X_1 = 0, Y_1 = 1)Pr(X_2 = 0, Y_2 = 1) \\
&+ Pr(X_1 = 1, Y_1 = 1)Pr(X_2 = 1, Y_2 = 1) = \\
&= p_1(\frac{1}{2} + \alpha_1)p_2(\frac{1}{2} + \alpha_2) + p_1(\frac{1}{2} - \alpha_1)p_2(\frac{1}{2} - \alpha_2) \\
&+ (1-p_1)(\frac{1}{2} + \beta_1)(1-p_2)(\frac{1}{2} + \beta_2) \\
&+ (1-p_1)(\frac{1}{2} - \beta_1)(1-p_2)(\frac{1}{2} - \beta_2) = \\
&= p_1p_2((\frac{1}{2} + \alpha_1)(\frac{1}{2} + \alpha_2) + (\frac{1}{2} - \alpha_1)(\frac{1}{2} - \alpha_2)) \\
&+ (1-p_1)(1-p_2)((\frac{1}{2} + \beta_1)(\frac{1}{2} + \beta_2) \\
&+ (\frac{1}{2} - \beta_1)(\frac{1}{2} - \beta_2)) = \\
&= p_1p_2(\frac{1}{2} + 2\alpha_1\alpha_2) + (1-p_1)(1-p_2)(\frac{1}{2} + 2\beta_1\beta_2)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
Pr(X_1 \oplus X_2 | Y_1 \oplus Y_2 = 0) &= \frac{Pr(X_1 \oplus X_2 = 0, Y_1 \oplus Y_2 = 0)}{Pr(Y_1 \oplus Y_2 = 0)} = \\
&= \frac{p_1p_2(\frac{1}{2} + 2\alpha_1\alpha_2) + (1-p_1)(1-p_2)(\frac{1}{2} + 2\beta_1\beta_2)}{p_1p_2 + (1-p_1)(1-p_2)} = \\
&= \frac{\frac{1}{2}(p_1p_2 + (1-p_1)(1-p2))}{p_1p_2 + (1-p_1)(1-p2)} \\
&+ \frac{2p_1p_2\alpha_1\alpha_2 + 2(1-p_1)(1-p_2)\beta_1\beta_2}{p_1p_2 + (1-p_1)(1-p_2)} = \\
&= \frac{1}{2} + \frac{2p_1p_2\alpha_1\alpha_2 + 2(1-p_1)(1-p_2)\beta_1\beta_2}{p_1p_2 + (1-p_1)(1-p_2)}
\end{aligned}
$$

□

Notice that Matsui's (non-conditional) piling up lemma is a special case when Y_1, Y_2 are constants (for example $Y_1 = Y_2 = 0$ and $P_1 = P_2 = 1$):

$$
\varepsilon(X_1 \oplus X_2) = \varepsilon(X_1 \oplus X_2 | Y_1 \oplus Y_2 = 0) = \frac{2 \cdot 1 \cdot 1 \cdot \alpha_1 \cdot \alpha_2 + 2 \cdot 0 \cdot 0 \cdot \beta_1 \cdot \beta_2}{1 \cdot 1 + 0 \cdot 0} = 2\alpha_1\alpha_2
$$

Lemma 2.

$$
\varepsilon(X_1 \oplus X_2 | Y_1 \oplus Y_2 = 1) = \frac{2(1-p_1)p_2\beta_1\alpha_2 + 2p_1(1-p_2)\alpha_1\beta_2}{(1-p_1)p_2 + p_1(1-p_2)}.
$$

Proof.

Let's write p_1, α_1 and β_1 in the following form:

$$\begin{aligned} p_1 &= Pr(Y_1 = 0) = Pr(Y_1 \oplus 1 = 1) \\ \alpha_1 &= \varepsilon(X_1|Y_1 = 0) = \varepsilon(X_1|Y_1 \oplus 1 = 1) \\ \beta_1 &= \varepsilon(X_1|Y_1 = 1) = \varepsilon(X_1|Y_1 \oplus 1 = 0) \end{aligned}$$

Then $\varepsilon(X_1 \oplus X_2|Y_1 \oplus Y_2 = 1)$ can be written as:

$$\varepsilon(X_1 \oplus X_2|Y_1 \oplus Y_2 = 1) = \varepsilon(X_1 \oplus X_2|((Y_1 \oplus 1) \oplus Y_2 = 0)$$

By substituting $Y_1 \oplus 1$ as Y_1 in Lemma 1 (and therefore also swapping between p and $1 - p$) we get:

$$\varepsilon(X_1 \oplus X_2|((Y_1 \oplus 1) \oplus Y_2 = 0) = \frac{2(1-p_1)p_2\beta_1\alpha_2 + 2p_1(1-p_2)\alpha_1\beta_2}{(1-p_1)p_2 + p_1(1-p_2)}$$

□

Lemma 3.

$$\varepsilon(X_1|Y_1 \oplus Y_2 = 0) = \frac{p_1p_2\alpha_1 + (1-p_1)(1-p_2)\beta_1}{p_1p_2 + (1-p_1)(1-p_2)}.$$

Proof.

$$\begin{aligned} Pr(Y_1 \oplus Y_2 = 0) &= p_1p_2 + (1-p_1)(1-p_2) \\ Pr(X_1 = 0, Y_1 \oplus Y_2 = 0) &= Pr(X_1 = 0, Y_1 = Y_2 = 0) \\ &+ Pr(X_1 = 0, Y_1 = Y_2 = 1) \\ &= Pr(X_1 = 0, Y_1 = 0)Pr(Y_2 = 0) \\ &+ Pr(X_1 = 0, Y_1 = 1)Pr(Y_2 = 1) \\ &= p_1(\frac{1}{2} + \alpha_1)p_2 + (1-p_1)(\frac{1}{2} + \beta_1)(1-p_2) \\ &= p_1p_2(\frac{1}{2} + \alpha_1) + (1-p_1)(1-p_2)(\frac{1}{2} + \beta_1) \end{aligned}$$

Therefore,

$$\begin{aligned} Pr(X_1|Y_1 \oplus Y_2 = 0) &= \frac{Pr(X_1 = 0, Y_1 \oplus Y_2 = 0)}{Pr(Y_1 \oplus Y_2 = 0)} = \\ &= \frac{p_1p_2(\frac{1}{2} + \alpha_1) + (1-p_1)(1-p_2)(\frac{1}{2} + \beta_1)}{p_1p_2 + (1-p_1)(1-p_2)} = \\ &= \frac{\frac{1}{2}(p_1p_2 + (1-p_1)(1-p2))}{p_1p_2 + (1-p_1)(1-p2)} \\ &+ \frac{p_1p_2\alpha_1 + (1-p_1)(1-p_2)\beta_1}{p_1p_2 + (1-p_1)(1-p_2)} = \\ &= \frac{1}{2} + \frac{p_1p_2\alpha_1 + (1-p_1)(1-p_2)\beta_1}{p_1p_2 + (1-p_1)(1-p_2)} \end{aligned}$$

□

Notice that this is a special case of Lemma 1 when $X_2 = 0$ (in which case $\alpha_2 = \beta_2 = 0.5$):

$$\varepsilon(X_1|Y_1 \oplus Y_2 = 0) = \varepsilon(X_1 \oplus 0|Y_1 \oplus Y_2 = 0) = \frac{2 \cdot p_1 \cdot p_2 \cdot \alpha_1 \cdot 0.5 + 2 \cdot (1-p_1) \cdot (1-p_2) \cdot \beta_1 \cdot 0.5}{p_1 p_2 + (1-p_1)(1-p_2)} =$$
$$= \frac{p_1 p_2 \alpha_1 + (1-p_1)(1-p_2)\beta_1}{p_1 p_2 + (1-p_1)(1-p_2)}$$