Chapter 1 - Linear cryptanalysis.

James McLaughlin

1 Introduction.

Linear cryptanalysis was first introduced by Mitsuru Matsui in [12]. The cryptanalyst attempts to find a linear equation \( x_1 \oplus \ldots \oplus x_i = y_1 \oplus \ldots \oplus y_j \) in the input and output bits of some part of the cipher which holds true with probability sufficiently different to 0.5. “Sufficiently different” means that for a known-plaintext attack on a feasible number of known plaintexts, when the correct key (or part of it) is tried on all of these known plaintexts, the number of plaintexts for which the equation holds will deviate significantly from one half of the total.

Such an equation is known as a “linear approximation”.

Definition 1.1. For a given linear approximation to part of a cipher, let \( p \) be the probability that it holds. We refer to \( |p - 1/2| \) as the bias of the approximation.

Matsui refined his attack in a 1994 paper [13], in which he used it to attack the Data Encryption Standard. The refined attack could break the full 16-round DES with an estimated \( 2^{43} \) time complexity and an 85% probability of success. This time complexity was the best achieved by any attack on DES at the time, and has not subsequently been improved upon!

However, it also required \( 2^{43} \) known plaintexts; and the difficulty of obtaining these, as well as the resources required to store them, meant that a brute force attack was in practice more feasible. This remained the case despite later work by various authors suggesting that the attack could succeed with less resources:

- Nyberg [14], and Harpes, Kramer and Massey [9] carried out analyses which showed that some of Matsui’s assumptions had been overly pessimistic, and that the true complexity of the attack would be better than he had predicted (although new complexity estimates were not provided.)
- Junod [11] provided evidence that the time required by Matsui’s attack was in fact equivalent only to that required to perform \( 2^{41} \) DES encryptions.
- Biryukov et al [5], conducting research into the use of several linear approximations simultaneously (Matsui’s attack had used two) conjectured that the attack’s space requirements could be reduced to \( 2^{41} \) using a particular set of 108 approximations. (They do not seem to have successfully implemented this attack, however.) They also conjectured that, although new techniques would be needed, using approximately 10,000 approximations simultaneously would reduce the space requirements to \( 2^{36} \).

Any new cipher must be shown to be resilient to linear cryptanalysis before it can be considered for use. There are now block cipher design strategies, in particular Joan Daemen and Vincent Rijmen’s wide trail strategy [7] and Serge Vaudenay’s decorrelation theory [15, 16, 17, 18], which can be used to design block ciphers resilient against linear cryptanalysis, differential cryptanalysis ([3, 2, 4] - the subject of Chapter 2) and their variants. The first of these was in fact used to design the Advanced Encryption Standard [8].
2 The attack.

2.1 Overview.

In an attack on a cipher, linear cryptanalysis is typically used in one of two ways. Both of these require a large volume of known (plaintext, ciphertext) pairs:

1. Let $r$ denote the number of rounds in the cipher. A linear approximation is found for the first $(r - 1)$ rounds. The cryptanalyst looks at how the output bits of this linear equation are input into the final round, and partially deciphers it with her guessed values for the relevant key bits. Only for the correct guess at these bits - the “target partial subkey” (TPS) - should the approximation hold true as often as expected.

The reason why the cryptanalyst is able to partially decipher the final round using only some of the key is that for several block ciphers, a given input bit to the round function will not affect all of the output bits. (For instance, in DES, no round function input bit affects more than eight of the 32 output bits.) Furthermore, the complete set of input bits affecting these output bits is usually relatively small, the output bits being partitioned up and each partition assigned to a small subset of the input bits. The cryptanalyst therefore only needs to use the key bits in that subset of the input bits.

The number of key bits that should comprise the TPS is something the cryptanalyst will need to decide for herself. It must be small enough for an exhaustive search over all possible TPS values to be possible, but the cryptanalyst needs to consider how the remaining key bits will be obtained. Exhaustive search? More linear attacks with a different target partial subkey? This is something that will probably depend on various different factors.

This is known as the “1R” attack.

As well as the TPS, one more bit of key information can be obtained using this attack; although this will take the form of a linear equation relating some of the key bits. We will explain later how to identify these key bits, but basically whether they xor to 1 or 0 can be determined by whether the linear approximation held or did not hold for most of the plaintexts - for the rest of the attack this does not matter as long as they do one or the other with the expected bias.

2. Matsui’s second paper defined the “2R” attack; which he used in his attack on DES. The cryptanalyst finds a linear approximation for all of the cipher’s rounds except the first and the last. As before, she uses candidate TPS values to decipher the final round output bits affected by the linear approximation’s output bits. However, the TPS bits used in this no longer comprise the whole TPS - for another subset of key bits, she encrypts some of the bits of each known plaintext to obtain the linear approximation’s input bits. She then checks to see whether the linear approximation holds.

As before, the correct TPS is expected to be the only one such that the approximation holds with the expected bias, and one more bit of key information can be obtained depending on whether it held or did not hold for most of the trials.

We stop to note that differential cryptanalysis, which is in many ways extremely similar to linear cryptanalysis, seems to be more flexible in terms of which rounds can and can’t be covered by their equivalent to the linear approximation. For instance, [4] attacked DES using a differential attack in which two rounds at the end were not covered, and in which the first round was intended to cause the input to the remaining rounds to have the properties required. Perhaps it’s just that these variations didn’t prove to be as effective against DES, and that they’re harder to apply to non-Feistel ciphers.
2.2 Deriving the approximation.

Let’s assume that we’re attempting to cryptanalyse a Feistel cipher like DES, or an SPN. Typically the only parts of these ciphers which cannot be expressed as linear Boolean formulae are the S-boxes.

Therefore, for each S-box and every possible subset of the input and output bits to it, we calculate using all possible inputs whether the xor of that subset of the input bits is equal to the xor of that subset of the output bits. From this, we calculate the probability bias of every possible linear approximation to the S-box.

We assign to each S-box a linear approximation table, in which we record these biases. For example, this is the linear approximation table for the example S-box in [10]:

<table>
<thead>
<tr>
<th>Input Sum</th>
<th>Output Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>0</td>
<td>+8 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 $-2$ $-2$ 0 0 $-2$ $+6$ $+2$ +2 0 0 $+2$ $+2$ 0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 $-2$ $-2$ 0 0 $-2$ $-2$ 0 0 $+2$ +2 0 0 $-6$ $+2$</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 0 0 0 0 0 0 0 $+2$ $-6$ $-2$ $-2$ $+2$ $+2$ $-2$ $-2$</td>
</tr>
<tr>
<td>4</td>
<td>0 $+2$ 0 $-2$ $-2$ $-4$ $-2$ 0 0 $-2$ 0 $+2$ $+2$ $-4$ $+2$ 0</td>
</tr>
<tr>
<td>5</td>
<td>0 $-2$ $-2$ 0 $-2$ 0 $+4$ $+2$ $-2$ 0 $-4$ $+2$ 0 $-2$ $-2$ 0</td>
</tr>
<tr>
<td>6</td>
<td>0 $+2$ $-2$ $+4$ $+2$ 0 0 $+2$ 0 $-2$ $+2$ $+2$ $-4$ $-2$ 0 0 $-2$</td>
</tr>
<tr>
<td>7</td>
<td>0 $-2$ 0 $+2$ $+2$ $-4$ $+2$ 0 $-2$ 0 $+2$ 0 $+4$ $+2$ $+2$ 0</td>
</tr>
<tr>
<td>8</td>
<td>0 0 0 0 0 0 0 0 $-2$ $+2$ $+2$ $-2$ $+2$ $-2$ $-2$ $-2$</td>
</tr>
<tr>
<td>9</td>
<td>0 0 $-2$ $-2$ 0 0 $-2$ $-2$ $-4$ 0 $-2$ $+2$ 0 $+4$ $+2$ $-2$</td>
</tr>
<tr>
<td>A</td>
<td>0 $+4$ $-2$ $+2$ $-4$ 0 $+2$ $-2$ $+2$ 0 0 $+2$ $+2$ 0 0 $+2$</td>
</tr>
<tr>
<td>B</td>
<td>0 $+4$ 0 $-4$ $+4$ 0 $+4$ 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>C</td>
<td>0 $-2$ $+4$ $-2$ $-2$ 0 $+2$ 0 $+2$ $+4$ 0 $+2$ 0 $-2$</td>
</tr>
<tr>
<td>D</td>
<td>0 $+2$ $+2$ 0 $-2$ $+4$ 0 $+2$ $-4$ $-2$ $+2$ 0 $+2$ 0 0 $+2$</td>
</tr>
<tr>
<td>E</td>
<td>0 $+2$ $+2$ 0 $-2$ $-4$ 0 $+2$ $-2$ 0 0 $-2$ $-4$ $+2$ $-2$ 0</td>
</tr>
<tr>
<td>F</td>
<td>0 $-2$ $-4$ $-2$ $-2$ 0 $+2$ 0 $0$ $-2$ $+4$ $-2$ $-2$ 0 $+2$ 0</td>
</tr>
</tbody>
</table>

The next step is to link these linear approximations together to form an overall approximation of the entire cipher except for the last (and maybe the first) round. Finding the best such approximation, or even a “good enough” approximation is not a trivial task. Nor is it obvious how best to do this, and we intend at some point in the future to find out if nature-inspired search techniques can be applied here. One currently-employed tactic is to find a good approximation for a small number of rounds such that the subset of input bits is the same as the subset of output bits. This allows the approximation to be iterated several times to approximate the full set of $r - 1$ or $r - 2$ rounds.

2.2.1 Building up the full approximation.

We’ll discuss joining up the individual approximations later, but let’s look at how we build up the set thereof. Consider an input bit that forms part of an $(r - 1)$-round linear approximation to the cipher. We’ll have to incorporate a linear approximation to the first round S-box it enters, and that input approximation can only involve the input bits from our overall approximation that actually enter it.

For the output bits involved in that linear approximation, we trace each one through the cipher until it enters another S-box. We’ll need to incorporate a linear approximation to that S-box as well. And, once again, the input bits of this approximation must be the ones we traced to that S-box... Eventually, a set of branching paths are traced through the cipher passing through various S-boxes along the way until they get to the approximation’s output bits.
A linear approximation to the first three rounds of the Heys SPN.

(Note that joining linear approximations together is more complicated in the case of a Feistel cipher. Whenever a path forks, you only follow the bits down one of the forks, and must choose which. In addition, you might have built up some of the approximation by starting from the last-but-one round and working backwards, or by working outwards from one of the rounds in the middle... This makes such diagrams for Feistel ciphers like DES much harder to follow. We may appear to “skip” rounds that did not need to be approximated, and round input bits resulting from working backwards can seem to appear from thin air if you’re trying to trace the approximation from the start of the cipher.)

When choosing a linear approximation for each S-box, there are two desirable properties which, unfortunately, may conflict with each other. Both of them result from the same fact; that the bias of the overall approximation is determined by multiplying together the biases of the individual approximations.

**Lemma 2.1.** For each value \(1 \leq i \leq n\), let \(X_i\) be a random variable, independent of \(X_j\) for all \(j \neq i\), such that:

\[
P(X_i = 0) = p_i \\
P(X_i = 1) = (1 - p_i)
\]

Then \(P(X_1 \oplus X_2 \oplus ...X_n = 0)\) is:

\[
\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} (p_i - 1/2).
\]
We use the Piling-Up Lemma in estimating the overall bias. Each $X_i$ corresponds to a linear approximation, taking the value 1 if it holds and 0 if it does not. This looks at first sight like an exact calculation; however the various $X_i$ are not in fact independent of each other. Despite this, the values given by the Piling-up Lemma have been observed to work well in practice, and do in fact slightly underestimate the probability $p$ that the approximation holds (as shown by arguments in [14, 9], and as evidenced by Junod’s experiments in [11]).

By examining the Piling-up Lemma, we see that it is desirable for each individual approximation to have as high a bias as possible, and for there to be as few individual approximations as possible. In general, though, the highest-bias approximation for a given S-box may not be the one that has the fewest output bits, so choosing it may push up the number of S-boxes we will have to approximate in the next round. This was certainly the case for DES, where the design team were particularly careful to avoid good linear approximations with only one output bit. [6] Similarly, the S-boxes of the cipher Serpent [1] were designed so that any linear approximation with one input and one output bit could not have a bias exceeding $1/8$ in magnitude; whereas the upper bound for absolute biases of other linear approximations was $1/4$.

### 2.3 Launching the attack.

Having found a good approximation, we next do whatever is necessary to obtain our known plaintext/ciphertext pairs. There is no simple equation relating the number of pairs $N$ to the success rate. However, in [12], Matsui gives approximate success rates for various values $N = c/|p - 1/2|^2$ ($p$ is the probability that the approximation holds and $c$ is some constant). Since a 99.9% success rate corresponds to $c = 16$, it would appear that to secure a cipher against linear cryptanalysis there must be no approximation with a probability $p$ such that $c/|p - 1/2|^2$ (plaintext, ciphertext) pairs can be feasibly stored except for extraordinarily low values of $c$.

If our TPS has $t$ bits, we’ll also need enough memory to hold $2^t$ integer values, one for each possible value of the TPS. A TPS’s corresponding allocation here is known as its counter. At the start of the attack, we initialise all these counters to zero.

Once we have obtained our pairs, for each pair and for each TPS guess we decrypt the relevant parts of the ciphertext (and encrypt the relevant parts of the plaintext), and check to see if our linear approximation holds for the thus-exposed bits. If it does, we add 1 to its counter.

We expect that the correct guess will be the only one for which the calculated probability bias is seen to occur. For the others, the bias observed should usually be extremely small. However, in practice other values will also produce significant biases, although not as significant as the correct one.

There are various possible reasons for this, stemming from the fact that the linear approximations of the various S-boxes are not after all entirely independent of each other.

What, then, should we do about these other significant biases? If we have been able to acquire a large number of known plaintexts, the bias exhibited by the correct TPS may noticeably exceed them, and so we may just choose to ignore them. Perhaps, though, we can’t be sure that the greatest bias observed corresponds to the correct guess. If so, we may want to assume that the largest bias corresponds to the correct guess at first, but if attempts to acquire the remaining key bits don’t result in any successful decryptions, switch to the value with the next most likely bias.

One question that has not hitherto been asked is whether these biases, which seem similar to local optima, might in some way be useful in applying nature-inspired search techniques like hill-climbing to the attack. It doesn’t look likely that hill-climbing can be directly employed, but
it may be possible, by analysing how these biases occurred, to obtain some useful information. We hope to be able to investigate this further at some point in the future.

2.3.1 The other bit of key information.

As we’ve stated, one more bit of key information can be obtained depending on whether the linear approximation held, or failed to hold, with the bias predicted. However, we haven’t really explained this.

Let us consider a simplified approximation, in which the linear approximation is traced through two 4 × 4-bit S-boxes. Like DES, we’ll assume that the data bits are xored with key bits before they enter the S-boxes. Let the first S-box be denoted $S_1$, and assume that the only plaintext bit $P_1$ entering it can be traced to input bit $x_2$ of $S_1$. Now, before that bit actually does enter the S-box, it’s xored with some key bit - call it $k_1$.

If our linear approximation involves only one output bit, say $y_4$, then our linear approximation is $x_2 = y_4$ and has probability $p_1$. Since $x_2 = P_1 \oplus k_1, P_1 \oplus k_1 = y_4$ with probability $p_1$.

Let’s trace this output bit to the second S-box, $S_2$. Let’s assume it enters the S-box as input bit $x_3$. Again, it would be xored with a key bit - call it $k_2$ - before entering it. Let the output bits this time be $y_2$ and $y_4$. We have the linear approximation $x_3 = y_2 \oplus y_4$, holding with probability $p_2$.

We now need to link the linear approximations together. We’ll slightly change our notation so that the input and output bits have a superscripted number identifying their corresponding S-box:

1. $x_2^1 = y_4^1$, which we translate to $P_1 \oplus k_1 = y_4^1$.
2. $x_2^2 = y_2^2 \oplus y_4^2$, which we translate to $y_4^1 \oplus k_2 = y_2^2 \oplus y_4^2$. Rearranging this gives us $y_4^1 = k_2 \oplus y_2^2 \oplus y_4^2$.

By substituting for $y_4^1$, we are now able to link the two approximations together and obtain $P_1 \oplus k_1 = k_2 \oplus y_2^2 \oplus y_4^2$. For our simplified example, we’ll assume that the cipher only has three rounds, and so we’re only approximating these two. The overall approximation has probability $p = p_1 \times p_2$.

Let’s rearrange that equation. With probability $p$, $P_1 \oplus y_2^2 \oplus y_4^2 = k_1 \oplus k_2$. Hence, if for the TPS value we accept as correct $P_1 \oplus y_2^2 \oplus y_4^2 = 0$ for $\approx p \times N$ pairs, we assume that the xor of those two key bits is zero - and vice versa.

In practice, however, the arguments in [14] that showed that the main section of the attack would succeed with higher probability than Matsui predicted, also show that this part of the attack can be significantly weakened for the same reason. The arguments on which this is based are something of a “double-edged sword” - this part could also be strengthened under certain circumstances; unfortunately it is shown that this part’s performance and chances of success are much harder to predict than initially seems to be the case.

References


