On Multiple Linear Approximations

Alex Biryukov*, Christophe De Cannière**, and Michael Quisquater**

Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10,
B–3001 Leuven-Heverlee, Belgium
{abiryuko, cdecanni, mquisqua}@esat.kuleuven.ac.be

Abstract. In this paper we study the long standing problem of information extraction from multiple linear approximations. We develop a formal statistical framework for block cipher attacks based on this technique and derive explicit and compact gain formulas for generalized versions of Matsui's Algorithm 1 and Algorithm 2. The theoretical framework allows both approaches to be treated in a unified way, and predicts significantly improved attack complexities compared to current linear attacks using a single approximation. In order to substantiate the theoretical claims, we benchmarked the attacks against reduced-round versions of DES and observed a clear reduction of the data and time complexities, in almost perfect correspondence with the predictions. The complexities are reduced by several orders of magnitude for Algorithm 1, and the significant improvement in the case of Algorithm 2 suggests that this approach may outperform the currently best attacks on the full DES algorithm.
Keywords: Linear cryptanalysis, multiple linear approximations, stochastic systems of linear equations, maximum likelihood decoding, key-ranking, DES, AES.

1 Introduction

Linear cryptanalysis [9] is one of the most powerful attacks against modern cryptosystems. In 1994, Kaliski and Robshaw [6] proposed the idea of generalizing this attack using multiple linear approximations (the previous approach considered only the best linear approximation). However, their technique was limited to cases where all approximations derive the same parity bit of the key. Unfortunately, this approach imposes a very strong restriction on the approximations, and the additional information gained by the few surviving approximations is often negligible.

In this paper we start by developing a theoretical framework for dealing with multiple linear approximations. We first generalize Matsui's Algorithm 1 based on this framework, and then reuse these results to generalize Matsui's Algorithm 2. Our approach allows to derive compact expressions for the performance of the attacks in terms of the biases of the approximations and the amount of data available to the attacker. The contribution of these theoretical expressions is twofold. Not only do they clearly demonstrate that the use of multiple approximations can significantly improve classical linear attacks, they also shed a new light on the relations between Algorithm 1 and Algorithm 2.

The main purpose of this paper is to provide a new generally applicable cryptanalytical tool. In order to illustrate the potential of this new approach, we implemented two attacks against reduced-round versions of DES, using this cipher as a well established benchmark for linear cryptanalysis. The experimental results, discussed in the second part of this paper, are in exact correspondence with our theoretical predictions and show that the latter are well justified.

This paper is organized as follows: Sect. 2 describes a very general maximum likelihood framework, which we will use in the rest of the paper; in Sect. 3 this framework is applied to derive and analyze an optimal attack algorithm based on multiple linear approximations. In the last part of this section, we provide a more detailed theoretical analysis of the assumptions made in order to derive the performance expressions. Sect. 4 presents experimental results on DES as

* F.W.O. Researcher, sponsored by the Fund for Scientific Research – Flanders (Belgium).
** F.W.O. Research Assistant, sponsored by the Fund for Scientific Research – Flanders (Belgium).

an example. Finally, Sect. 5 discusses possible implications of our techniques to modern block ciphers such as the AES. A more detailed discussion of the practical aspects of the attacks, the relation with coding theory, an overview of previous work, and a search algorithm for the m best approximations can be found in the appendices.

2 General Framework

In this section we discuss the main principles of statistical cryptanalysis and set up a generalized framework for analyzing block ciphers based on maximum likelihood. This framework can be seen as an adaptation or extension of earlier frameworks for statistical attacks proposed by Murphy *et al.* [13], Vaudenay [16], Junod [5] and Selçuk [15].

2.1 Attack Model

We consider a block cipher E_k which maps a plaintext $P \in \mathcal{P}$ to a ciphertext $C = E_k(P) \in \mathcal{C}$. The mapping is invertible and depends on a secret key $k \in \mathcal{K}$. We now assume that an adversary is given N different plaintext–ciphertext pairs (P_i, C_i) encrypted with a particular secret key k^* (a known plaintext scenario), and his task is to recover the key from this data. A general statistical approach — also followed by Matsui's original linear cryptanalysis — consists in performing the following three steps:

Distillation phase. In a typical statistical attack, only a fraction of the information contained in the N plaintext–ciphertext pairs is exploited. A first step therefore consists in extracting the relevant parts of the data, and discarding all information which is not used by the attack. In our framework, the distillation operation is denoted by a function $\psi : \mathcal{P} \times \mathcal{C} \to \mathcal{X}$ which is applied to each plaintext–ciphertext pair. The result is a vector $\mathbf{x} = (x_1, \ldots, x_N)$ with $x_i = \psi(P_i, C_i)$, which contains all relevant information. If $|\mathcal{X}| \ll N$, which is usually the case, we can further reduce the data by counting the occurrence of each element of $\mathcal{X}$ and only storing a vector of counters $\mathbf{t} = (t_0, \ldots, t_{|\mathcal{X}|-1})$. In this paper we will not restrict ourselves to a single function ψ, but consider m separate functions ψ_j, each of which maps the text pairs into different sets $\mathcal{X}_j$ and generates a separate vector of counters $\mathbf{t}_j$.

Analysis phase. This phase is the core of the attack and consists in generating a list of key candidates from the information extracted in the previous step. Usually, candidates can only be determined up to a set of equivalent keys, i.e., typically, a majority of the key bits is transparent to the attack. In general, the attack defines a function $\sigma : \mathcal{K} \to \mathcal{Z}$ which maps each key k onto an equivalent key class $z = \sigma(k)$. The purpose of the analysis phase is to determine which of these classes are the most likely to contain the true key k^* given the particular values of the counters $\mathbf{t}_j$.

Search phase. In the last stage of the attack, the attacker exhaustively tries all keys in the classes suggested by the previous step, until the correct key is found. Note that the analysis and the searching phase may be intermixed: the attacker might first generate a short list of candidates, try them out, and then dynamically extend the list as long as none of the candidates turns out to be correct.

2.2 Attack Complexities

When evaluating the performance of the general attack described above, we need to consider both the data complexity and the computational complexity. The data complexity is directly determined by N, the number of plaintext–ciphertext pairs required by the attack. The computational complexity depends on the total number of operations performed in the three phases of the attack. In order to compare different types of attacks, we define a measure called the *gain* of the attack:

Definition 1 (Gain). *If an attack is used to recover an n-bit key and is expected to return the correct key after having checked on the average M candidates, then the* gain *of the attack, expressed in bits, is defined as:*

$$\gamma = -\log_2 \frac{2 \cdot M - 1}{2^n} \tag{1}$$

Let us illustrate this with an example where an attacker wants to recover an n-bit key. If he does an exhaustive search, the number of trials before hitting the correct key can be anywhere from 1 to 2^n. The average number M is $(2^n + 1)/2$, and the gain according to the definition is 0. On the other hand, if the attack immediately derives the correct candidate, M equals 1 and the gain is $\gamma = n$. There is an important caveat, however. Let us consider two attacks which both require a single plaintext–ciphertext pair. The first deterministically recovers one bit of the key, while the second recovers the complete key, but with a probability of 1/2. In this second attack, if the key is wrong and only one plaintext–ciphertext pair is available, the attacker is forced to perform an exhaustive search. According to the definition, both attacks have a gain of 1 bit in this case. Of course, by repeating the second attack for different pairs, the gain can be made arbitrary close to n bits, while this is not the case for the first attack.

2.3 Maximum Likelihood Approach

The design of a statistical attack consists of two important parts. First, we need to decide on how to process the N plaintext–ciphertext pairs in the distillation phase. We want the counters $\mathbf{t}_j$ to be constructed in such a way that they concentrate as much information as possible about a specific part of the secret key in a minimal amount of data. Once this decision has been made, we can proceed to the next stage and try to design an algorithm which efficiently transforms this information into a list of key candidates. In this section, we discuss a general technique to optimize this second step. Notice that throughout this paper, we will denote random variables by capital letters.

In order to minimize the amount of trials in the search phase, we want the candidate classes which have the largest probability of being correct to be tried first. If we consider the correct key class as a random variable Z and denote the complete set of counters extracted from the observed data by $\mathbf{t}$, then the ideal output of the analysis phase would consist of a list of classes $\{z\}$, sorted according to the conditional probability

$$\Pr\left[Z = z \mid \mathbf{t}\right] .$$

Taking the Bayesian approach, we express this probability as follows:

$$\Pr\left[Z = z \mid \mathbf{t}\right] = \frac{\Pr\left[\mathbf{T} = \mathbf{t} \mid z\right] \cdot \Pr\left[Z = z\right]}{\Pr\left[\mathbf{T} = \mathbf{t}\right]} . \tag{2}$$

The factor $\Pr\left[Z = z\right]$ denotes the a priori probability that the class z contains the correct key k^*, and is equal to the constant $1/|\mathcal{Z}|$, with $|\mathcal{Z}|$ the total number of classes, provided that the key was chosen at random. The denominator is determined by the probability that the specific set of counters $\mathbf{t}$ is observed, taken over all possible keys and plaintexts. The only expression in (2) that depends on z, and thus affects the sorting, is the factor $\Pr\left[\mathbf{T} = \mathbf{t} \mid z\right]$, compactly written as $P_z(\mathbf{t})$. This quantity denotes the probability, taken over all possible plaintexts, that a key from a given class z produces a set of counters $\mathbf{t}$. When viewed as a function of z for a fixed set $\mathbf{t}$, the expression $\Pr\left[\mathbf{T} = \mathbf{t} \mid z\right]$ is also called the *likelihood* of z given $\mathbf{t}$, and denoted by $L_{\mathbf{t}}(z)$, i.e.,

$$L_{\mathbf{t}}(z) = P_z(\mathbf{t}) = \Pr\left[\mathbf{T} = \mathbf{t} \mid z\right] .$$

This likelihood and the actual probability $\Pr\left[Z = z \mid \mathbf{t}\right]$ have distinct values, but they are proportional for a fixed $\mathbf{t}$, as follows from (2). Typically, the likelihood expression is simplified by applying a logarithmic transformation. The result is denoted by

$$\mathcal{L}_{\mathbf{t}}(z) = \log L_{\mathbf{t}}(z)$$

and called the *log-likelihood*. Note that this transformation does not affect the sorting, since the logarithm is a monotonously increasing function.

Assuming that we can construct an efficient algorithm that accurately estimates the likelihood of the key classes and returns a list sorted accordingly, we are now ready to derive a general expression for the gain of the attack.

Let us assume that the plaintexts are encrypted with an n-bit secret key k^*, contained in the equivalence class z^*, and let $\mathcal{Z}^* = \mathcal{Z} \setminus \{z^*\}$ be the set of classes *different* from z^*. The expected number of classes checked during the searching phase before the correct key is found, is given by the expression

$$1 + \sum_{z \in \mathcal{Z}^*} \Pr\left[\mathcal{L}_{\mathbf{T}}(z) \geq \mathcal{L}_{\mathbf{T}}(z^*) \mid z^*\right],$$

where the random variable $\mathbf{T}$ represents the set of counters generated by a key from the class z^*, given N random plaintexts. Note that this number includes the correct key class, but since this class will be treated differently later on, we do not include it in the sum. In order to compute the probabilities in this expression, we define the sets $\mathcal{T}_z = \{\mathbf{t} \mid \mathcal{L}_{\mathbf{t}}(z) \geq \mathcal{L}_{\mathbf{t}}(z^*)\}$. Using this notation, we can write

$$\Pr\left[\mathcal{L}_{\mathbf{T}}(z) \geq \mathcal{L}_{\mathbf{T}}(z^*) \mid z^*\right] = \sum_{\mathbf{t} \in \mathcal{T}_z} P_{z^*}(\mathbf{t}).$$

Knowing that each class z contains $2^n/|\mathcal{Z}|$ different keys, we can now derive the expected number of trials M^*, given a secret key k^*. Note that the number of keys that need to be checked in the correct equivalence class z^* is only $(2^n/|\mathcal{Z}| + 1)/2$ on the average, yielding

$$M^* = \frac{2^n}{|\mathcal{Z}|} \cdot \left[\frac{1}{2} + \sum_{z \in \mathcal{Z}^*} \sum_{\mathbf{t} \in \mathcal{T}_z} P_{z^*}(\mathbf{t})\right] + \frac{1}{2}. \tag{3}$$

This expression needs to be averaged over all possible secret keys k^* in order to find the expected value M, but in many cases[1] we will find that M^* does not depend on the actual value of k^*, such that $M = M^*$. Finally, the gain of the attack is computed by substituting this value of M into (1).

3 Application of the Framework to Multiple Approximations

In this section, we apply the ideas discussed above to construct a general framework for analyzing block ciphers using multiple linear approximations.

The starting point in linear cryptanalysis is the existence of unbalanced linear expressions involving plaintext bits, ciphertext bits, and key bits. In this paper we assume that we can use m such expressions (we will show how to find them in App. E):

$$\Pr\left[P[\chi_P^j] \oplus C[\chi_C^j] \oplus K[\chi_K^j] = 0\right] = \frac{1}{2} + \epsilon_j, \quad j = 1, \ldots, m, \tag{4}$$

with (P, C) a random plaintext–ciphertext pair encrypted with a random key K. The notation $X[\chi]$ stands for $X_{l_1} \oplus X_{l_2} \oplus \ldots \oplus X_{l_a}$, where $X_{l_1}, \ldots, X_{l_a}$ represent particular bits of X. The deviation ϵ_j is called the *bias* of the linear expression.

We now use the framework of Section 2.1 to design an attack which exploits the information contained in (4). The first phase of the cryptanalysis consists in extracting the relevant parts from the N plaintext–ciphertext pairs. The linear expressions in (4) immediately suggest the following functions ψ_j:

$$x_{i,j} = \psi_j(P_i, C_i) = P_i[\chi_P^j] \oplus C_i[\chi_C^j], \quad i = 1, \ldots, N,$$

[1] In some cases the variance of the gain over different keys would be very significant. In these cases it might be worth to exploit this phenomenon in a weak-key attack scenario, like in the case of the IDEA cipher.

with $x_{i,j} \in \mathcal{X}_j = \{0, 1\}$. These values are then used to construct m counter vectors $\mathbf{t_j} = (t_j, N - t_j)$, where t_j and $N - t_j$ reflect the number of plaintext–ciphertext pairs for which $x_{i,j}$ equals 0 and 1 respectively.[2]

In the second step of the framework, a list of candidate key classes needs to be generated. We represent the equivalent key classes induced by the m linear expressions in (4) by an m-bit word $z = (z_1, \ldots, z_m)$ with $z_j = k[\chi_K^j]$. Note that m might possibly be much larger than n, the length of the key k. In this case, only a subspace of all possible m-bit words corresponds to a valid key class. The exact number of classes $|\mathcal{Z}|$ depends on the number of *independent* linear approximations (i.e., the rank of the corresponding linear system).

3.1 Computing the Likelihoods of the Key Classes

We will for now assume that the linear expressions in (4) are statistically independent for different plaintext–ciphertext pairs and for different values of j (in the next section we will discuss this important point in more details). This allows us to apply the maximum likelihood approach described earlier in a very straightforward way. In order to simplify notations, we define the probabilities p_j and q_j, and the *imbalances*[3] c_j of the linear expressions as

$$p_j = 1 - q_j = \frac{1 + c_j}{2} = \frac{1}{2} + \epsilon_j \,.$$

We start by deriving a convenient expression for the probability $P_z(\mathbf{t})$. To simplify the calculation, we first give a derivation for the special key class $z' = (0, \ldots, 0)$. Assuming independence of different approximations and of different (P_i, C_i) pairs, the probability that this key generates the counters t_j is given by the product

$$P_{z'}(\mathbf{t}) = \prod_{j=1}^{m} \binom{N}{t_j} \cdot p_j^{t_j} \cdot q_j^{N - t_j} \,. \tag{5}$$

In practice, p_j and q_j will be very close to 1/2, and N very large. Taking this into account, we approximate the m-dimensional binomial distribution above by an m-dimensional Gaussian distribution:

$$P_{z'}(\mathbf{t}) \approx \prod_{j=1}^{m} \frac{e^{-\frac{(t_j - p_j \cdot N)^2}{N/2}}}{\sqrt{\pi \cdot N/2}} = \prod_{j=1}^{m} \frac{e^{-\frac{N}{2}(\hat{c}_j - c_j)^2}}{\sqrt{\pi \cdot N/2}} = \frac{e^{-\frac{N}{2}\sum (\hat{c}_j - c_j)^2}}{\left(\sqrt{\pi \cdot N/2}\right)^m} \,.$$

The variable $\hat{c}_j$ is called the *estimated imbalance* and is derived from the counters t_j according to the formula:

$$N \cdot \frac{1 + \hat{c}_j}{2} = t_j \,.$$

For any key class z, we can repeat the reasoning above, yielding the following general expression:

$$P_z(\mathbf{t}) \approx \frac{e^{-\frac{N}{2}\sum (\hat{c}_j - (-1)^{z_j} \cdot c_j)^2}}{\left(\sqrt{\pi \cdot N/2}\right)^m} \tag{6}$$

This formula has a useful geometrical interpretation: if we take a key from a fixed key class z^* and construct an m-dimensional vector $\hat{\mathbf{c}} = (\hat{c}_1, \ldots, \hat{c}_m)$ by encrypting N random plaintexts, then $\hat{\mathbf{c}}$ will be distributed around the vector $\mathbf{c_{z^*}} = ((-1)^{z_1^*} c_1, \ldots, (-1)^{z_m^*} c_m)$ according to a Gaussian distribution with a diagonal variance-covariance matrix $1/\sqrt{N} \cdot I_m$, where I_m is an $m \times m$ identity matrix. This is illustrated in Fig. 1. From (6) we can now directly compute the log-likelihood:

[2] The vectors $\mathbf{t_j}$ are only constructed to be consistent with the framework described earlier. In practice of course, the attacker will only calculate t_j (this is a minimal sufficient statistic).

[3] Also known in the literature as "correlations".

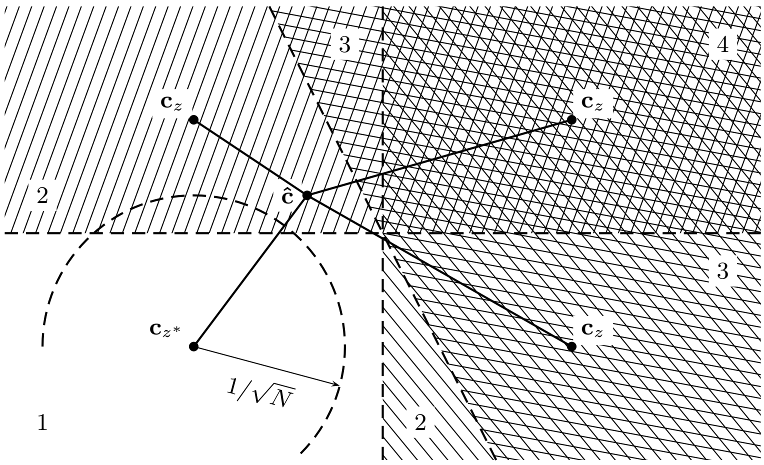

Fig. 1. Geometrical interpretation for $m = 2$. The correct key class z^* has the second largest likelihood in this example. The numbers in the picture represent the number of trials M^*.

$$\mathcal{L}_{\mathbf{t}}(z) = \log L_{\mathbf{t}}(z) = \log P_z(\mathbf{t}) \approx C - \frac{N}{2} \sum_{j=1}^{m} (\hat{c}_j - (-1)^{z_j} \cdot c_j)^2 \,. \tag{7}$$

The constant C depends on m and N only, and is irrelevant to the attack. From this formula we immediately derive the following property.

Lemma 1. *The relative likelihood of a key class z is completely determined by the Euclidean distance $|\hat{\mathbf{c}} - \mathbf{c}_{\mathbf{z}}|$, where $\hat{\mathbf{c}}$ is an m-dimensional vector containing the estimated imbalances derived from the known texts, and $\mathbf{c}_{\mathbf{z}} = ((-1)^{z_1} c_1, \ldots, (-1)^{z_m} c_m)$.*

The lemma implies that $\mathcal{L}_{\mathbf{T}}(z) > \mathcal{L}_{\mathbf{T}}(z^*)$ if and only if $|\hat{\mathbf{c}} - \mathbf{c}_{\mathbf{z}}| < |\hat{\mathbf{c}} - \mathbf{c}_{\mathbf{z}^*}|$. This is a common result in coding theory.

3.2 Estimating the Gain of the Attack

Based on the geometrical interpretation given above, and using the results from Section 2.3, we can now easily derive the gain of the attack.

Theorem 1. *Given m approximations and N independent pairs (P_i, C_i), an adversary can mount a linear attack with a gain equal to:*

$$\gamma = -\log_2 \left[2 \cdot \frac{1}{|\mathcal{Z}|} \sum_{z \in \mathcal{Z}^*} \Phi\left(-\sqrt{N} \cdot \frac{|\mathbf{c}_z - \mathbf{c}_{z^*}|}{2} \right) + \frac{1}{|\mathcal{Z}|} \right] , \tag{8}$$

where $\Phi(\cdot)$ is the cumulative normal distribution function, $\mathbf{c}_{\mathbf{z}} = ((-1)^{z_1} c_1, \ldots, (-1)^{z_m} c_m)$, and $|\mathcal{Z}|$ is the number of key classes induced by the approximations.

Proof. The probability that the likelihood of a key class z exceeds the likelihood of the correct key class z^* is given by the probability that the vector $\hat{\mathbf{c}}$ falls into the half plane $\mathcal{T}_c = \{\mathbf{c} \mid |\hat{\mathbf{c}} - \mathbf{c}_{\mathbf{z}}| \le |\hat{\mathbf{c}} - \mathbf{c}_{\mathbf{z}^*}|\}$. Considering the fact that $\hat{\mathbf{c}}$ describes a Gaussian distribution around $\mathbf{c}_{z^*}$ with a variance-covariance matrix $1/\sqrt{N} \cdot I_m$, we need to integrate this Gaussian over the half plane $\mathcal{T}_c$ and due to zero covariances, we immediately find:

$$\Pr\left[\mathcal{L}_{\mathbf{T}}(z) \ge \mathcal{L}_{\mathbf{T}}(z^*) \mid z^*\right] = \Phi\left(-\sqrt{N} \cdot \frac{|\mathbf{c}_z - \mathbf{c}_{z^*}|}{2} \right) .$$

By summing these probabilities as in (3) we find the expected number of trials:

$$M^* = \frac{2^n}{|\mathcal{Z}|} \cdot \left[\frac{1}{2} + \sum_{z \in \mathcal{Z}^*} \Phi\left(-\sqrt{N} \cdot \frac{|\mathbf{c}_z - \mathbf{c}_{z^*}|}{2}\right)\right] + \frac{1}{2}\,. \tag{9}$$

The gain is obtained by substituting this expression for M^* in equation (1).

□

The formula derived in the previous theorem can easily be evaluated as long as $|\mathcal{Z}|$ is not too large. In order to estimate the gain in the other cases as well, we need to make a few approximations.

Corollary 1. *If $|\mathcal{Z}|$ is sufficiently large, the gain derived in Theorem 1 can accurately be approximated by*

$$\gamma \approx -\log_2 \left[2 \cdot \frac{|\mathcal{Z}| - 1}{|\mathcal{Z}|} \cdot \Phi\left(-\sqrt{\frac{N \cdot \bar{c}^2}{2}}\right) + \frac{1}{|\mathcal{Z}|}\right] \triangleq f(N \cdot \bar{c}^2, |\mathcal{Z}|)\,, \tag{10}$$

where $\bar{c}^2 = \sum_{j=1}^{m} c_j^2$.

Proof. The corollary is proved in App. A.

An interesting conclusion that can be drawn from the corollary above is that the gain of the attack is mainly determined by the product $N \cdot \bar{c}^2$. As a result, if we manage to increase $\bar{c}^2$ by using more linear characteristics, then the required number of known plaintext–ciphertext pairs N can be decreased with the same factor, without affecting the gain. Since the quantity $\bar{c}^2$ plays a very important role in the attacks, we give it a name and define it explicitly.

Definition 2. *The* capacity $\bar{c}^2$ *of a system of m approximations is defined as:*

$$\bar{c}^2 = \sum_{j=1}^{m} c_j^2 = 4 \cdot \sum_{j=1}^{m} \epsilon_j^2\,.$$

3.3 Extension: Multiple Approximations and Matsui's Algorithm 2

The approach taken in the previous section can be seen as an extension of Matsui's Algorithm 1. Just as in Algorithm 1, the adversary analyses parity bits of the known plaintext–ciphertext pairs and then tries to determine parity bits of internal round keys. An alternative approach, which is called Algorithm 2 and yields much more efficient attacks in practice, consists in guessing parts of the round keys in the first and the last round, and determining the probability that the guess was correct by exploiting linear characteristics over the remaining rounds. In this section we will show that the results derived above can still be applied in this situation, provided that we modify some definitions.

Let us denote by $\mathcal{Z}_O$ the set of possible guesses for the targeted subkeys of the outer rounds (round 1 and round r). For each guess z_O and for all N plaintext–ciphertext pairs, the adversary does a partial encryption and decryption at the top and bottom of the block cipher, and recovers the parity bits of the intermediate data blocks involved in m different $(r-2)$-round linear characteristics. Using this data, he constructs $m' = |\mathcal{Z}_O| \cdot m$ counters t_j, which can be transformed into a m'-dimensional vector $\hat{\mathbf{c}}$ containing the estimated imbalances.

As explained in the previous section, the m linear characteristics involve m parity bits of the key, and thus induce a set of equivalent key classes, which we will here denote by $\mathcal{Z}_I$ (I from *inner*). Although not strictly necessary, we will for simplicity assume that the sets $\mathcal{Z}_O$ and $\mathcal{Z}_I$ are independent, such that each guess $z_O \in \mathcal{Z}_O$ can be combined with any class $z_I \in \mathcal{Z}_I$, thereby determining a subclass of keys $z = (z_O, z_I) \in \mathcal{Z}$ with $|\mathcal{Z}| = |\mathcal{Z}_O| \cdot |\mathcal{Z}_I|$.

At this point, the situation is very similar to the one described in the previous section, the main difference being a higher dimension m'. The only remaining question is how to construct the m'-dimensional vectors $\mathbf{c_z}$ for each key class $z = (z_O, z_I)$. To solve this problem, we will need to make some assumptions. Remember that the coordinates of $\mathbf{c_z}$ are determined by the expected imbalances of the corresponding linear expressions, given that the data is encrypted with a key from class z. For the m counters that are constructed after guessing the correct subkey z_O, the expected imbalances are determined by z_I and equal to $(-1)^{z_{I,1}}c_1, \ldots, (-1)^{z_{I,m}}c_m$. For each of the $m' - m$ other counters, however, we will assume that the wrong guesses result in independent random-looking parity bits, showing no imbalance at all.[4] Accordingly, the vector $\mathbf{c_z}$ has the following form:

$$\mathbf{c_z} = (0, \ldots, 0, (-1)^{z_{I,1}}c_1, \ldots, (-1)^{z_{I,m}}c_m, 0, \ldots, 0)$$

With the modified definitions of $\mathcal{Z}$ and c_z given above, both Theorem 1 and Corollary 1 still hold (the proofs are in App A). Notice however that the gain of the Algorithm-2-style linear attack will be significantly larger because it depends on the capacity of linear characteristics over $r-2$ rounds instead of r rounds.

3.4 Influence of Dependencies

When deriving (5) in the Section 3, we assumed statistical independence. This assumption is not always fulfilled, however. In this section we discuss different potential sources of dependencies and estimate how they might influence the cryptanalysis.

Dependent plaintext–ciphertext pairs. A first assumption made by equation (5) concerns the dependency of the parity bits $x_{i,j}$ with $1 \leq i \leq N$, computed with a single linear approximation for different plaintext–ciphertext pairs. The equation assumes that the probability that the approximation holds for a single pair equals $p_j = 1/2 + \epsilon_j$, regardless of what is observed for other pairs. This is a very reasonable assumption if the N plaintexts are chosen randomly, but even if they are picked in a systematic way, we can still safely assume that the corresponding ciphertexts are sufficiently unrelated as to prevent statistical dependencies.

Dependent text mask. The next source of dependencies is more fundamental and is related to dependent text masks. Suppose for example that we want to use three linear approximations with plaintext–ciphertext masks (χ_P^1, χ_C^1), (χ_P^2, χ_C^2), (χ_P^3, χ_C^3), and that $\chi_P^1 \oplus \chi_P^2 \oplus \chi_P^3 = \chi_C^1 \oplus \chi_C^2 \oplus \chi_C^3 = 0$. It is immediately clear that the parity bits computed for these three approximations cannot possibly be independent: for all (P_i, C_i) pairs, the bit computed for the 3rd approximation $x_{i,3}$ is equal to $x_{i,1} \oplus x_{i,2}$.

Even in such cases, however, we believe that the results derived in the previous section are still quite reasonable. In order to show this, we consider the probability that a single random plaintext encrypted with an equivalent key z yields a vector[5] of parity bits $\mathbf{x} = (x_1, \ldots, x_m)$. Let us denote by χ_T^j the concatenation of both text masks χ_P^j and χ_C^j. Without loss of generality, we can assume that the m masks χ_T^j are linearly independent for $1 \leq j \leq l$ and linearly dependent (but different) for $l < j \leq m$. This implies that $\mathbf{x}$ is restricted to a l-dimensional subspace $\mathcal{R}$. We will only consider the key class $z' = (0, \ldots, 0)$ in order to simplify the equations. The probability we want to evaluate is:

$$P_{z'}(\mathbf{x}) = \Pr\left[X_j = x_j \text{ for } 1 \leq j \leq m \mid z'\right]$$

These (unknown) probabilities determine the (known) imbalances c_j of the linear approximations through the following expression:

$$c_j = \sum_{\mathbf{x} \in \mathcal{R}} P_{z'}(\mathbf{x}) \cdot (-1)^{x_j} .$$

[4] Note that for some ciphers, other assumptions may be more appropriate. The reasoning in this section can be applied to these cases just as well, yielding different but very similar results.

[5] Note a small abuse of notation here: the definition of $\mathbf{x}$ differs from the one used in Section 2.1.

We now make the (in many cases reasonable) assumption that all $2^l - m$ masks χ_T, which depend linearly on the masks χ_T^j, but which differ from the ones considered by the attack, have negligible imbalances. In this case, the equation above can be reversed (note the similarity with the Walsh-Hadamard transform), and we find that:

$$P_{z'}(\mathbf{x}) = \frac{1}{2^l} \sum_{j=1}^{m} c_j \cdot (-1)^{x_j} .$$

Assuming that $m \cdot c_j \ll 1$ we can make the following approximation:

$$P_{z'}(\mathbf{x}) \approx \frac{2^m}{2^l} \prod_{j=1}^{m} \frac{1 + c_j \cdot (-1)^{x_j}}{2} .$$

Apart from an irrelevant constant factor $2^m/2^l$, this is exactly what we need: it implies that, even with dependent masks, we can still multiply probabilities as we did in order to derive (5). This is an important conclusion, because it indicates that the capacity of the approximations continues to grow, even when m exceeds twice the block size, in which case the masks are necessarily linearly dependent.

Dependent trails. A third type of dependencies might be caused by merging linear trails. When analyzing the best linear approximations for DES, for example, we notice that most of the good linear approximations follow a very limited number of trail through the inner rounds of the cipher, which might result in dependencies. Although this effect did not appear to have any influence on our experiments (with up to 100 different approximations), we cannot exclude at this point that they will affect attacks using much more approximations.

Dependent key masks. We finally note that we did not make any assumption about the dependency of key masks in the previous sections. This implies that all results derived above remain valid for dependent key masks.

4 Experimental Results

In Section 3 we derived an optimal approach for cryptanalyzing block ciphers using multiple linear approximations. In this section, we implement practical attack algorithms based on this approach and evaluate their performance when applied to DES, the standard benchmark for linear cryptanalysis. Our experiments show that the attack complexities are in perfect correspondence with the theoretical results derived in the previous sections.

4.1 Attack Algorithm MK 1

Table 1 summarizes the attack algorithm presented in Section 2 (we call this algorithm *Attack Algorithm MK 1*). In order to verify the theoretical results, we applied the attack algorithm to 8 rounds of DES. We picked 86 linear approximations with a total capacity $\bar{c}^2 = 2^{-15.6}$ (see Definition 2). In order to speed up the simulation, the approximations were picked to contain 10 linearly independent key masks, such that $|\mathcal{Z}| = 1024$. Fig. 2 shows the simulated gain for Algorithm MK 1 using these 86 approximations, and compares it to the gain of Matsui's Algorithm 1, which uses the best one only ($\bar{c}^2 = 2^{-19.4}$). We clearly see a significant improvement. While Matsui's algorithm requires about 2^{21} pairs to attain a gain close to 1 bit, only 2^{16} pairs suffice for Algorithm MK 1. The theoretical curves shown in the figure were plotted by computing the gain using the exact expression for M^* derived in Theorem 1 and using the approximation from Corollary 1. Both fit nicely with the experimental results.

Note, that the attack presented in this section is a non-optimized proof of concept demonstration, even higher gains would be possible with dedicated attacks. For a more detailed discussion of the technical aspects playing a role in the implementation of Algorithm MK 1, we refer to App. B.

Table 1. Attack Algorithm MK 1 and its complexity.

Distillation phase. Obtain N plaintext–ciphertext pairs (p_i, c_i). For $1 \leq j \leq m$, count the number t_j of pairs satisfying $p_i[\chi_P^j] \oplus c_i[\chi_C^j] = 0$ and compute the estimated imbalance $\hat{c}_j = 2 \cdot t_j - N$.

Analysis phase. For each equivalent key class $z \in \mathcal{Z}$, determine the distance

$$|\hat{\mathbf{c}} - \mathbf{c_z}|^2 = \sum_{j=1}^{m} (\hat{c}_j - (-1)^{z_j} \cdot c_j)^2$$

and use these values to construct a sorted list, starting with the class with the smallest distance.

Search phase. Run through the sorted list and exhaustively try all n-bit keys contained in the equivalence classes until the correct key is found.

	Data compl.	Time compl.	Memory compl.
Distillation:	$O(1/\bar{c}^2)$	$O(m/\bar{c}^2)$	$O(m)$
Analysis:	-	$O(m \cdot \lvert\mathcal{Z}\rvert)$	$O(\lvert\mathcal{Z}\rvert)$
Search:	-	$O(2^{n-\gamma})$	$O(\lvert\mathcal{Z}\rvert)$

4.2 Attack Algorithm MK 2

In this subsection, we discuss the experimental results for the generalization of Matsui's Algorithm 2 using multiple linear approximations (called *Attack Algorithm MK 2*). We simulated the attack algorithm on 8 rounds of DES and compared the results to the gain of the corresponding Algorithm 2 attack described in Matsui's paper [10].

Our attack uses eight 6-round linear approximations with a total capacity $\bar{c}^2 = 2^{-11.9}$. In order to compute the parity bits of these equations, eight 6-bit subkeys need to be guessed in the first and the last rounds (how this is done in practice is explained in App. B). Figure 3 compares the gain of the attack to Matsui's Algorithm 2, which uses the two best approximations ($\bar{c}^2 = 2^{-13.2}$). For the same amount of data, the multiple linear attack clearly achieves a much higher gain. This reduces the complexity of the search phase by multiple orders of magnitude. On the other hand, for the same gain, the adversary can reduce the amount of data by at least a factor 2. For example, for a gain of 12 bits, the data complexity is reduced from $2^{17.8}$ to $2^{16.6}$. This is in a close correspondence with the ratio between the capacities. Note that both simulations were carried out under the assumption of independent subkeys (this was also the case for the simulations presented in [10]). Without this assumption, the gain will closely follow the graphs on the figure, but stop increasing as soon as the gain equals the number of independent key bits involved in the attack.

As in the previous subsection our goal was not to provide the best attack on 8-round DES, but to show that Algorithm-2 style attacks do gain from the use of multiple linear approximations, with a data reduction proportional to the increase in the joint capacity. We refer to App. B for the technical aspects of the implementation of Algorithm MK 2.

4.3 Capacity – DES Case Study

In Section 3 we argued that the minimal amount of data needed to obtain a certain gain compared to exhaustive search is determined by the capacity $\bar{c}^2$ of the linear approximations. In order to get a first estimate of the potential improvement of using multiple approximations, we calculated the total capacity of the best m linear approximations of DES for $1 \leq m \leq 2^{16}$. The capacities were computed using an adapted version of Matsui's algorithm (see Appendix E). The results, plotted for different number of rounds, are shown in Figures 4 and 5, both for approximations restricted to a single S-box per round and for the general case. Note that the single best approximation is not visible on these figures due to the scale of the graphs.

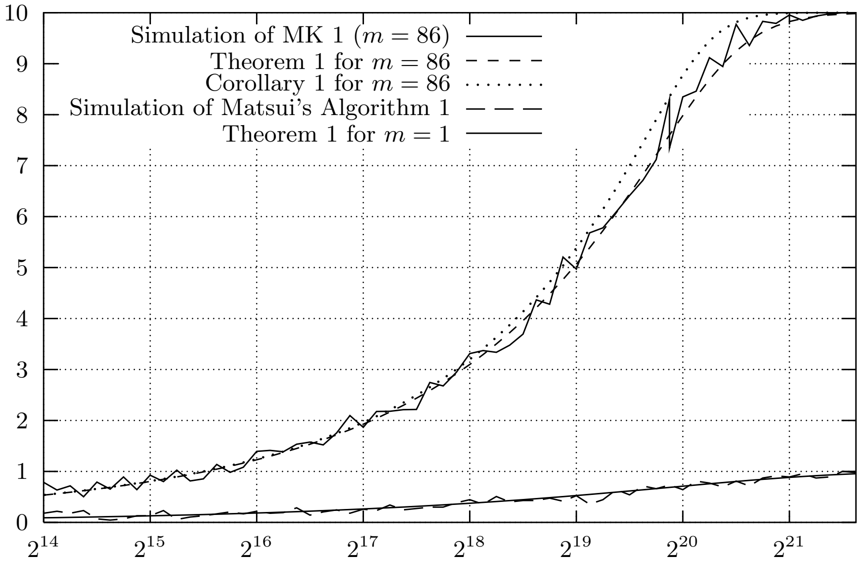

Fig. 2. Gain (in bits) as a function of data (known plaintext) for 8-round DES.

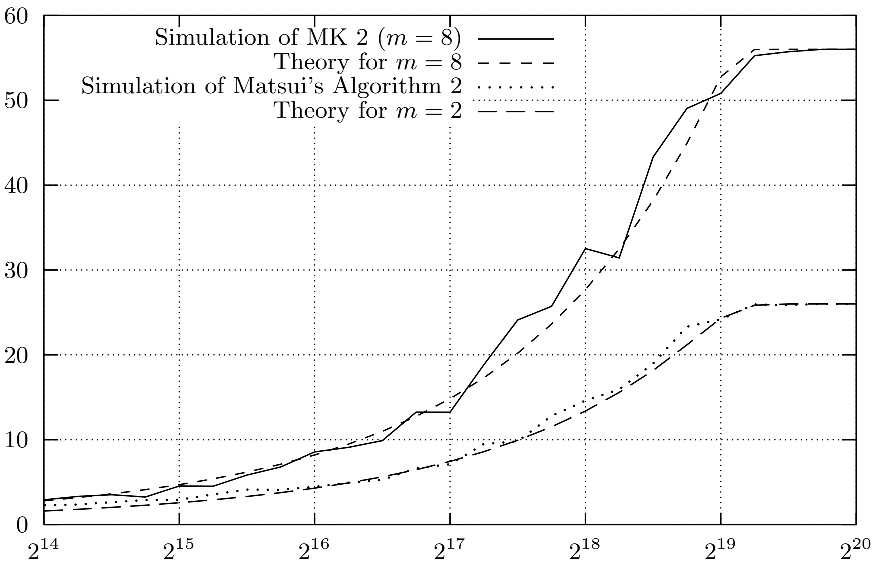

Fig. 3. Gain (in bits) as a function of data (known plaintext) for 8-round DES.

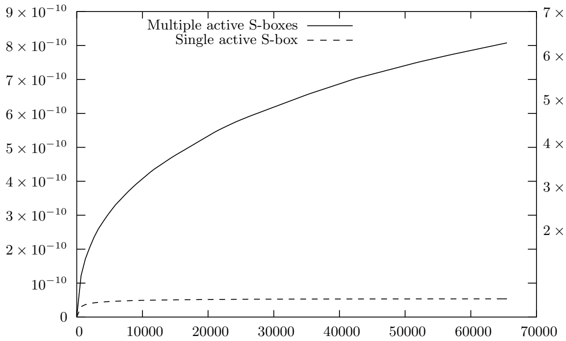

Fig. 4. Capacity (14 rounds).

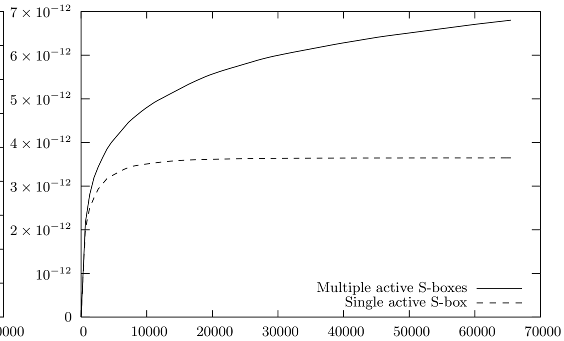

Fig. 5. Capacity (16 rounds).

In [6], Kaliski and Robshaw showed that the first 10 006 approximations with a single active S-box per round have a joint capacity of $4.92 \cdot 10^{-11}$ for 14 rounds of DES.[6] Fig. 4 shows that this capacity can be increased to $4 \cdot 10^{-10}$ when multiple S-boxes are allowed. Comparing this to the capacity of Matsui's best approximation ($\bar{c}^2 = 1.29 \cdot 10^{-12}$), the factor 38 gained by Kaliski and Robshaw is increased to 304 in our case. Practical techniques to turn this increased capacity into an effective reduction of the data complexity are presented in this paper, but exploiting the full gain of 10 000 unrestricted approximations will require additional techniques. In theory, however, it would be possible to reduce the data complexity form 2^{43} (in Matsui's case, using two approximations) to about 2^{36} (using 10 000 approximations).

In order to provide a more conservative (and probably rather realistic) estimation of the implications of our new attacks on full DES, we searched for 14-round approximations which only require three 6-bit subkeys to be guessed simultaneously in the first and the last rounds. The capacity of the 108 best approximations satisfying this restriction is $9.83 \cdot 10^{-12}$. This suggests that an MK 2 attack exploiting these 108 approximations might reduce the data complexity with a factor 4 compared to Matsui's Algorithm 2 (i.e., 2^{41} instead of 2^{43}). This is comparable to the Knudsen-Mathiassen reduction [7], but would preserve the advantage of being a known-plaintext attack rather than chosen-plaintext.

Using very high numbers of approximations is somewhat easier in practice for MK 1 because we do not have to impose restrictions on the plaintext and ciphertext masks (see App. B). Analyzing the capacity for the 10 000 best 16-round approximations, we now find a capacity of $5 \cdot 10^{-12}$. If we restrict the complexity of the search phase to an average of 2^{43} trials (i.e., a gain of 12 bits), we expect that the attack will require 2^{41} known plaintexts. As expected, this theoretical number is larger than for the MK 2 attack using the same amount of approximations.

5 Consequences for Rijndael-like and Other Ciphers

Recently many ciphers with "provable security" against linear attack have been designed (for ex. AES [1]). In order to attain such provable security, the designer usually estimates the maximal S-box bias (which for specially constructed 8-bit S-boxes is 2^{-3}) and then tries to give a lower bound on the number of active S-boxes for the worst possible "trails". The designer then chooses the number of rounds for which the square of the total approximation bias is smaller than the block size plus some security margin. Since for Rijndael any 4-round trail will have at least 25 active S-boxes (one of the best trails of active S-boxes is for ex. 1–4–16–4) the best approximation bias is bounded by 2^{-75}. The bias profile for ciphers like Rijndael is artificially flattened by the designers, but this happens at the expense of many approximations having the same bias. That means that a multiple linear approximation strategy would be more powerful against such a

[6] Note that Kaliski and Robshaw calculated the sum of squared biases: $\sum \epsilon_j^2 = \bar{c}^2/4$.

cipher than against a cipher like DES, where the biases of the two best approximations are much higher than the others.

Considering the attacks described in this paper, we expect that one may need to add a few rounds when defining bounds of provable security against linear cryptanalysis, based only on best approximations. Still, since AES has a large security margin against linear cryptanalysis we do not believe that linear attacks enhanced with multiple linear approximations will pose a practical threat to the security of the AES.

6 Conclusions

In this paper, we have studied the problem of generalizing linear cryptanalytic attacks given m multiple linear approximations, which has been stated in 1994 by Kaliski and Robshaw [6]. In order to solve the problem, we have developed a statistical framework based on maximum likelihood decoding. This approach is optimal in the sense that it utilizes all the information that is present in the multiple linear approximations. We have derived explicit and compact gain formulas for the generalized linear attacks and have shown that for a constant gain, the data-complexity N of the attack is proportional to the inverse joint capacity $\bar{c}^2$ of the multiple linear approximations: $N \propto 1/\bar{c}^2$. The gain formulas hold for the generalized versions of both algorithms proposed by Matsui (Algorithm 1 and Algorithm 2).

In the second half of the paper we have proposed several practical methods which deliver the theoretical gains derived in the first part of the paper. We have proposed a key-recovery algorithm MK 1 which has a time complexity $O(m/\bar{c}^2 + m \cdot |\mathcal{Z}|)$ and a data complexity $O(1/\bar{c}^2)$, where $|\mathcal{Z}|$ is the number of solutions of the system of m equations defined by the linear approximations. We have also designed an algorithm MK 2 which is a direct generalization of Matsui's Algorithm 2, as described in [10]. The performances of both algorithms are very close to our theoretical estimations and confirm that the data-complexity of the attack decreases proportionally to the increase in the joint capacity of multiple approximations. We have used 8-round DES as a standard benchmark in our experiments and in all cases our attacks perform significantly better than those given by Matsui. However our goal in this paper was not to produce the most optimal attack on DES, but to construct a new cryptanalytic tool applicable to a variety of ciphers. Nevertheless we do expect that ideas expressed in this paper will lead to an improvement in the state-of-the-art attacks on DES.

References

[1] J. Daemen and V. Rijmen, *The Design of Rijndael: AES — The Advanced Encryption Standard.* Springer-Verlag, 2002.

[2] Y. Desmedt, ed., *Advances in Cryptology – CRYPTO'94*, vol. 839 of *Lecture Notes in Computer Science*, Springer-Verlag, 1994.

[3] J. D. Golic, V. Bagini, and G. Morgari, "Linear cryptanalysis of bluetooth stream cipher," in *Advances in Cryptology – EUROCRYPT 2002* (L. R. Knudsen, ed.), vol. 2332 of *Lecture Notes in Computer Science*, pp. 238–255, Springer-Verlag, 2002.

[4] P. Junod, "On the complexity of Matsui's attack," in *Selected Areas in Cryptography, SAC 2001* (S. Vaudenay and A. M. Youssef, eds.), vol. 2259 of *Lecture Notes in Computer Science*, pp. 199–211, Springer-Verlag, 2001.

[5] P. Junod, "On the optimality of linear, differential, and sequential distinguishers," in *Advances in Cryptology – EUROCRYPT 2003* (E. Biham, ed.), Lecture Notes in Computer Science, pp. 17–32, Springer-Verlag, 2003.

[6] B. S. Kaliski and M. J. Robshaw, "Linear cryptanalysis using multiple approximations," in Desmedt [2], pp. 26–39.

[7] L. R. Knudsen and J. E. Mathiassen, "A chosen-plaintext linear attack on DES," in *Fast Software Encryption, FSE 2000* (B. Schneier, ed.), vol. 1978 of *Lecture Notes in Computer Science*, pp. 262–272, Springer-Verlag, 2001.

[8] L. R. Knudsen and M. J. B. Robshaw, "Non-linear approximations in linear cryptanalysis," in *Proceedings of Eurocrypt'96* (U. Maurer, ed.), no. 1070 in Lecture Notes in Computer Science, pp. 224–236, Springer-Verlag, 1996.

[9] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Advances in Cryptology – EUROCRYPT'93* (T. Helleseth, ed.), vol. 765 of *Lecture Notes in Computer Science*, pp. 386–397, Springer-Verlag, 1993.

[10] M. Matsui, "The first experimental cryptanalysis of the Data Encryption Standard," in Desmedt [2], pp. 1–11.

[11] M. Matsui, "Linear cryptanalysis method for DES cipher (I)." (extended paper), unpublished, 1994.

[12] M. Matsui, "On correlation between the order of S-boxes and the strength of DES," in Santis [14], pp. 366–375.

[13] S. Murphy, F. Piper, M. Walker, and P. Wild, "Likelihood estimation for block cipher keys," Technical report, Information Security Group, Royal Holloway, University of London, 1995.

[14] A. D. Santis, ed., *Advances in Cryptology – EUROCRYPT'94*, vol. 950 of *Lecture Notes in Computer Science*, Springer-Verlag, 1995.

[15] A. A. Selçuk, "On probability of success in linear and differential cryptanalysis," in *Proceedings of SCN'02* (S. Cimato, C. Galdi, and G. Persiano, eds.), vol. 2576 of *Lecture Notes in Computer Science*, Springer-Verlag, 2002. Also available at `https://www.cerias.purdue.edu/papers/archive/2002-02.ps`.

[16] S. Vaudenay, "On the weak keys of blowfish," in *Fast Software Encryption, FSE'96* (D. Gollmann, ed.), vol. 1039 of *Lecture Notes in Computer Science*, pp. 27–32, Springer-Verlag, 1996.

A Proofs

A.1 Proof of Corollary 1

Corollary 1. *If $|\mathcal{Z}|$ is sufficiently large, the gain derived in Theorem 1 can accurately be approximated by*

$$\gamma \approx -\log_2 \left[2 \cdot \frac{|\mathcal{Z}|-1}{|\mathcal{Z}|} \cdot \Phi\left(-\sqrt{\frac{N \cdot \bar{c}^2}{2}} \right) + \frac{1}{|\mathcal{Z}|} \right] , \tag{11}$$

where $\bar{c}^2 = \sum_{j=1}^{m} c_j^2$ is called the total capacity *of the m linear characteristics.*

Proof. In order to show how (11) is derived from (8), we just need to construct an approximation for the expression

$$\frac{1}{|\mathcal{Z}^*|} \sum_{z \in \mathcal{Z}^*} \Phi\left(-\sqrt{N} \cdot \frac{|\mathbf{c}_z - \mathbf{c}_{z^*}|}{2} \right) = \frac{1}{|\mathcal{Z}^*|} \sum_{z \in \mathcal{Z}^*} \Phi\left(-\sqrt{N/4 \cdot |\mathbf{c}_z - \mathbf{c}_{z^*}|^2} \right) . \tag{12}$$

We first define the function $f(x) = \Phi(\sqrt{N/4 \cdot x})$. Denoting the average value of a set of variables by $E[\cdot] = \widehat{\cdot}$, we can reduce (12) to the compact expression $E[f(x)]$, with $x = |\mathbf{c}_z - \mathbf{c}_{z^*}|^2$. By expanding $f(x)$ into a Taylor series around the average value $\widehat{x}$, we find

$$E[f(x)] = f(\widehat{x}) + 0 + f''(\widehat{x}) \cdot E[(x - \widehat{x})^2] + \dots .$$

Provided that the higher order moments of x are sufficiently small, we can use the approximation $E[f(x)] \approx f(\widehat{x})$. Exploiting the fact that the jth coordinate of each vector $\mathbf{c}_z$ is either c_j or $-c_j$, we can easily calculate the average value $\widehat{x}$:

$$\widehat{x} = \frac{1}{|\mathcal{Z}^*|} \sum_{z \in \mathcal{Z}^*} |\mathbf{c}_z - \mathbf{c}_{z^*}|^2 = 2 \cdot \frac{|\mathcal{Z}|}{|\mathcal{Z}^*|} \sum_{j=1}^{m} c_j^2 .$$

When $|\mathcal{Z}|$ is sufficiently large (say $|\mathcal{Z}| > 2^8$), the right hand part can be approximated by $2 \cdot \sum_{j=1}^{m} c_j^2 = 2 \cdot \bar{c}^2$ (remember that $\mathcal{Z}^* = \mathcal{Z} \setminus \{z^*\}$, and thus $|\mathcal{Z}^*| = |\mathcal{Z}| - 1$). Substituting this into the relation $E[f(x)] \approx f(\widehat{x})$, we find

$$\frac{1}{|\mathcal{Z}^*|} \sum_{z \in \mathcal{Z}^*} \Phi\left(-\sqrt{N} \cdot \frac{|\mathbf{c}_z - \mathbf{c}_{z^*}|}{2} \right) \approx \Phi\left(-\sqrt{\frac{N \cdot \bar{c}^2}{2}} \right) .$$

By applying this approximation to the gain formula derived in Theorem 1, we directly obtain expression (11).

□

A.2 Gain Formulas for the Algorithm-2-style Attack

With the modified definitions of $\mathcal{Z}$ and c_z given in Section 3.3, Theorem 1 can immediately be applied. This results in the following corollary.

Corollary 2. *Given m approximations and N independent pairs (P_i, C_i), an adversary can mount an Algorithm-2-style linear attack with a gain equal to:*

$$\gamma = -\log_2 \left[2 \cdot \frac{1}{|\mathcal{Z}|} \sum_{z \in \mathcal{Z}^*} \Phi\left(-\sqrt{N} \cdot \frac{|\mathbf{c}_z - \mathbf{c}_{z^*}|}{2} \right) + \frac{1}{|\mathcal{Z}|} \right] . \tag{13}$$

The formula above involves a summation over all elements of $\mathcal{Z}^*$. Motivated by the fact that $|\mathcal{Z}^*| = |\mathcal{Z}_O| \cdot |\mathcal{Z}_I| - 1$ is typically very large, we now derive a more convenient approximated expression similar to Corollary 1. In order to do this, we split the sum into two parts. The first part considers only keys $z \in \mathcal{Z}_1^* = \mathcal{Z}_1 \setminus \{z^*\}$ where $\mathcal{Z}_1 = \{z \mid z_O = z_O^*\}$; the second part sums over all remaining keys $z \in \mathcal{Z}_2 = \{z \mid z_O \neq z_O^*\}$. In this second case, we have that $|\mathbf{c}_z - \mathbf{c}_{z^*}|^2 = 2 \cdot \sum_{j=1}^m c_j^2 = 2 \cdot \bar{c}^2$ for all $z \in \mathcal{Z}_2$, such that

$$\sum_{z \in \mathcal{Z}_2} \Phi\left(-\sqrt{N} \cdot \frac{|\mathbf{c}_z - \mathbf{c}_{z^*}|}{2}\right) = |\mathcal{Z}_2| \cdot \Phi\left(-\sqrt{\frac{N \cdot \bar{c}^2}{2}}\right) .$$

For the first part of the sum, we apply the approximation used to derive Corollary 1 and obtain a very similar expression:

$$\sum_{z \in \mathcal{Z}_1^*} \Phi\left(-\sqrt{N} \cdot \frac{|\mathbf{c}_z - \mathbf{c}_{z^*}|}{2}\right) \approx |\mathcal{Z}_1^*| \cdot \Phi\left(-\sqrt{\frac{N \cdot \bar{c}^2}{2}}\right) .$$

Combining both result we find the counterpart of Corollary 1 for an Algorithm-2-style linear attack.

Corollary 3. *If $|\mathcal{Z}|$ is sufficiently large, the gain derived in Theorem 2 can accurately be approximated by*

$$\gamma \approx -\log_2 \left[2 \cdot \frac{|\mathcal{Z}| - 1}{|\mathcal{Z}|} \cdot \Phi\left(-\sqrt{\frac{N \cdot \bar{c}^2}{2}}\right) + \frac{1}{|\mathcal{Z}|}\right] , \qquad (14)$$

where $\bar{c}^2 = \sum_{j=1}^m c_j^2$ is the total capacity of the m linear characteristics.

Notice that although Corollaries 1 and 3 contain identical formulas, the gain of the Algorithm-2-style linear attack will be significantly larger because it depends on the capacity of linear characteristics over $r - 2$ rounds instead of r rounds.

B Discussion – Practical Aspects

When attempting to calculate the optimal estimators derived in Section 3, the attacker might be confronted with some practical limitations, which are often cipher-dependent. In this section we discuss possible problems and propose ways to deal with them.

B.1 Attack Algorithm MK 1

When estimating the potential gain in Section 3, we did not impose any restrictions on the number of approximations m. However, while it does reduce the complexity of the search phase (since it increases the gain), having an excessively high number m increases both the time and the space complexity of the distillation and the analysis phase. At some point the latter will dominate, cancelling out any improvement made in the search phase.

Analyzing the complexities in Table 1, we can make a few observations. We first note that the time complexity of the distillation phase should be compared to the time needed to encrypt $N \propto 1/\bar{c}^2$ plaintext–ciphertext pairs. Given that a single counting operation is much faster than an encryption, we expect the complexity of the distillation to remain negligible compared to the encryption time as long as m is only a few orders of magnitude (say $m < 100$).

The second observation is that the number of different key classes $|\mathcal{Z}|$ clearly plays an important role, both for the time and the memory complexities of the algorithm. In a practical situation, the memory is expected to be the strongest limitation. Different approaches can be taken to deal with this problem:

Straightforward, but inefficient approach. Since the number of different key classes $|\mathcal{Z}|$ is bounded by 2^m, the most straightforward solution is to limit the number of approximations. A realistic upper bound would be $m < 32$. The obvious drawback of this approach is that it will not allow us to attain very high capacities.

Exploiting dependent key masks. A better approach is to impose a bound on the number l of *linearly independent* key masks χ_K^j. This way, we limit the memory requirements to $|\mathcal{Z}| = 2^l$, but still allow a large number of approximations (for ex. a few thousands). This approach restricts the choice of approximations, however, and thus reduces the maximum attainable capacity. This is the approach taken in Section 4.1. Note also that the attack described in [6] can be seen as a special case of this approach, with $l = 1$.

Merging separate lists. A third strategy consists in constructing separate lists and merging them dynamically. Suppose for simplicity that the m key masks χ_K^j considered in the attack are all independent. In this case, we can apply the analysis phase twice, each time using $m/2$ approximations. This will result in two sorted lists of intermediate key classes, both containing $2^{m/2}$ classes. We can then dynamically compute a sorted sequence of final key classes constructed by taking the product of both lists. The ranking of the sequence is determined by the likelihood of these final classes, which is just the sum of the likelihoods of the elements in the separate lists. This approach slightly increases[7] the time complexity of the analysis phase, but will considerably reduce the memory requirements. Note that this approach can be generalized in order to allow some dependencies in the key masks.

B.2 Attack Algorithm MK 2

We now briefly discuss some practical aspects of the Algorithm-2-style multiple linear attack, called Attack Algorithm MK 2. As discussed earlier, the ideas of the attack are very similar to Attack Algorithm MK 1, but there are a number of additional issues. In the following paragraphs, we denote the number of rounds of the cipher by r.

Choice of characteristics. In order to limit the amount of guesses in rounds 1 and r, only parts of the subkeys in these rounds will be guessed. This restricts the set of useful $r-2$-round characteristics to those that only depend on bits which can be derived from the plaintext, the ciphertext, and the partial subkeys. This obviously reduces the maximum attainable capacity.

Efficiency of the distillation phase. During the distillation phase, all N plaintexts need to be analyzed for all $|\mathcal{Z}_O|$ guesses z_O. Since $|\mathcal{Z}_O|$ is rather large in practice, this could be very computational intensive. For example, a naive implementation would require $O(N \cdot |\mathcal{Z}_O|)$ steps and even Matsui's counting trick would use $O(N + |\mathcal{Z}_O|^2)$ steps. However, the distillation can be performed in $O(N + |\mathcal{Z}_O|)$ steps by gradually guessing parts of z_O and re-processing the counters.

Merging Separate lists. The idea of working with separate lists can be applied here just as for MK 1.

Computing distances. In order to compare the likelihoods of different keys, we need to evaluate the distance $|\hat{\mathbf{c}} - \mathbf{c}_\mathbf{z}|^2$ for all classes $z \in \mathcal{Z}$. The vectors $\hat{\mathbf{c}}$ and $\mathbf{c}_\mathbf{z}$ are both $|\mathcal{Z}_O| \cdot m$-dimensional. When calculating this distance as a sum of squares, most terms do not depend on z, however. This allows the distance to be computed very efficiently, by summing only m terms.

B.3 Attack Algorithm MD 1 (distinguishing/key-recovery)

The main limitation of Algorithm MK 1 and MK 2 is the bound on the number of key classes $|\mathcal{Z}|$. In this section, we show that this limitation disappears if our sole purpose is to distinguish

[7] In cases where the gain of the attack is several bits, this approach will actually decrease the complexity, since we expect that only a fraction of the final sequence will need to be computed.

an encryption algorithm E_k from a random permutation R. As usual, the distinguisher can be extended into a key-recovery attack by adding rounds at the top and at the bottom.

If we observe N plaintext–ciphertext pairs and assume for simplicity that the a priori probability that they were constructed using the encryption algorithm is 1/2, we can construct a distinguishing attack using the maximum likelihood approach in a similar way as in Section 3. Assuming that all secret keys k are equally probable, one can easily derive the likelihood that the encryption algorithm was used, given the values of the counters $\mathbf{t}$:

$$L_E(\mathbf{t}) \approx \frac{1}{2^m} \prod_{j=1}^{m} \binom{N}{t_j} \cdot \left(p_j^{t_j} \cdot q_j^{N-t_j} + q_j^{t_j} \cdot p_j^{N-t_j} \right) .$$

This expression is correct if all text masks and key masks are independent, but is still expected to be a good approximation, if this assumption does not hold (for the reasons discussed in Section 3.4). A similar likelihood can be calculated for the random permutation:

$$L_R(\mathbf{t}) = \prod_{j=1}^{m} \binom{N}{t_j} \cdot \left(\frac{1}{2} \right)^N .$$

Contrary to what was found for Algorithm MK 1, both likelihoods can be computed in time proportional to m, i.e., independent of $|\mathcal{Z}|$. The complete distinguishing algorithm, called *Attack Algorithm MD 1* consists of two steps:

Distillation phase. Obtain N plaintext–ciphertext pairs (P_i, C_i). For $1 \le j \le m$, count the number t_j of pairs satisfying $P_i[\chi_P^j] \oplus C_i[\chi_C^j] = 0$.

Analysis phase. Compute $L_E(\mathbf{t})$ and $L_R(\mathbf{t})$. If $L_E(\mathbf{t}) > L_R(\mathbf{t})$, decide that the plaintexts were encrypted with the algorithm E_k (using some unknown key k).

The analysis of this algorithm is a matter of further research.

C Links of Multiple Approximation Attack to Coding Theory

The problem of extracting information from many linear approximations may be viewed as a decoding problem in which k information bits about the secret key are encoded through a set of m linear equations into an m-bit codeword[8]. Given a single known plaintext-ciphertext pair the attacker sees an extremely 'noisy' version of this m-bit codeword. However, the attacker is allowed to request additional transmissions (i.e., more known plaintext-ciphertext pairs) in order to improve the quality of the channel. The cryptanalysis problem thus seems to be related to an NP-complete problem of decoding a random linear code. However, worst case hardness does not ensure average case hardness, and there are several important distinctions with the general case; the parity check matrix is produced by approximations derived from the cipher and thus contains some structure and is often sparse. Recovery of even a single information bit or distinguishing a system from random might be sufficient for the attack. The attacker can adaptively request more data, based on partial decoding results. The attacker has always access to a perfect (though expensive) information source by checking key candidates against (P_i, C_i) pairs and eliminating all false key-candidates. Such additional channel is not available in a pure coding-theoretic scenario.

D Previous Work: History of Linear Cryptanalysis

In this section we cover the history of linear cryptanalysis and its various refinements. In [9] linear cryptanalysis of DES was described. The paper suggests two algorithms: the first one is

[8] A similar framework is used for linear cryptanalysis of stream ciphers in [3].

the so called Algorithm 1, which covers the full cipher with a single approximation and recovers a single parity bit of the key. The second algorithm (called Algorithm 2) covers $(r-1)$ rounds of a cipher by a single approximation and guesses the 6-bit key which enter the active S-boxes of the last round. By repeating this approach twice (using the encryption-decryption symmetry of Feistel ciphers) the attacker is able to recover 14-bits of the key with 2^{47} known plaintexts and with a success rate of 95%. The remaining 42-bits would be recovered by exhaustive key search. In [11] it is suggested to use two 14-round approximations and to guess the keys both at the input and at the output (the so called Algorithm 2-B). As a result the attacker guesses 13-bits of the key (12 bits entering the S-boxes and one parity bit of the approximation) for each of the two approximations in parallel. In total he gains 26 bits of the key (due to the DES key schedule the bits are disjoint) using 2^{45} known plaintexts and with 98.8% success probability. The remaining 30 bits are found by fast exhaustive search. One notices that the effort spent in the two phases of this attack is very unbalanced. By using the idea of key-ranking [10] the attacker introduces a "time of analysis/data" tradeoff to the attack. The idea is to require only 2^{43} known plaintexts, but to allow the correct key to be within the first 2^{13} from a merged list of 2^{26} keys. The complexity of the attack is thus $2^{13} \cdot 2^{30} = 2^{43}$, and the success rate is 85%. In this paper, when we refer to "Algorithm 2", we are referring to this last attack.

In [12] Matsui proposed a search algorithm for the best linear approximation. The algorithm is a branch and bound algorithm with an underestimating heuristic. It works by induction on the number of rounds. It is very sensitive to the initial approximation which is used to cut the tree branches: for a good choice it runs several seconds for full 16-round DES and for arbitrary number of active S-boxes.

Junod [4] shows that Matsui's ranking estimate is pessimistic (as was predicted by Matsui and others) and that the analysis phase has a complexity of at most 2^{41} if one uses an optimal ranking criteria. However this observation does not help to reduce the data complexity much, since the approximations are very sensitive to the amount of data and degrade rapidly if less data is available.

Several generalizations of the linear cryptanalysis method have been proposed in the last 10 years: Kaliski-Robshaw [6] suggested to use many linear approximations instead of one, but did not provide a method for doing so, except for the case when all the approximations cover the same parity bit of the key. The idea of using non-linear approximations has been suggested by Knudsen-Robshaw [8]. Knudsen-Mathiassen [7] suggest to convert linear cryptanalysis into a chosen plaintext attack, which would gain the first round of approximation for free. The gain is small, since Matsui's attack gains the first round rather efficiently as well.

E Search for the m-best Linear Approximations

In this section we describe an algorithm which searches for the m-best linear approximations (allowing several active S-boxes per-round). The algorithm is a very simple adaptation of Matsui's algorithm [12]. The main idea is to keep a queue of m-best approximations for n-rounds: $Q_n = (q_n^1, \ldots, q_n^m)$, sorted in order of decreasing bias. Here $q_i^n = (\text{pattern}_n^i, B_n^i)$ stores both the linear mask pattern for all the intermediate rounds pattern_n^i, and the bias of the approximation B_n^i. The algorithm will use the worst approximation in the current list $B_n^m = \min_i(B_n^i)$ for pruning the tree (instead of the best approximation which is used in Matsui's pruning).

The algorithm may be rewritten to run much faster by using the following observations: given a list of m best approximations for $i-1$ rounds, queue Q_{i-1}, we can try to find a rough approximation of the list for i rounds, queue Q_i, by extending approximations from Q_{i-1} by one round in all the possible ways. Performance of algorithm is very sensitive to the quality of the initial approximation $\overline{B_n^m}$ and since approximations found this way are usually very good we avoid checking many unnecessary branches. It is actually possible to rewrite an algorithm completely in a way that will use only the bound on the worst-approximation, and this is how the capacity graphs in this paper were computed.

Algorithm 1 Search for Multiple Linear Approximations (SMA)

Procedure Round-1
BEGIN the program
while There are candidates for ΓY_1 and ΓX_1 **do**
    $p_1 \Leftarrow (\Gamma Y_1, \Gamma X_1)$.
    if $[p_1, B_{n-1}] \geq \overline{B_n^m}$ **then**
        Call **Procedure Round-2.**
    end if
end while
END the program
Procedure Round-2:
while There are candidates for ΓY_2 and ΓX_2 **do**
    $p_2 \Leftarrow (\Gamma Y_2, \Gamma X_2)$.
    if $[p_1, p_2, B_{n-2}] \geq \overline{B_n^m}$ **then**
        Call **Procedure Round-3**.
    end if
end while
Procedure Round-i ($3 \leq i \leq n-1$):
while There are candidates for ΓX_i **do**
    $\Gamma Y_i \Leftarrow \Gamma Y_{i-2} \oplus \Gamma X_{i-1}$.
    $p_i \Leftarrow (\Gamma Y_i, \Gamma X_i)$.
    if $[p_1, p_2, \ldots, p_i, B_{n-i}] \geq \overline{B_n^m}$ **then**
        Call **Procedure Round-(i+1)**.
    end if
end while
Procedure Round-n:
while There are candidates for ΓX_n **do**
    $\Gamma Y_n \Leftarrow \Gamma Y_{n-2} \oplus \Gamma X_{n-1}$.
    $p_n \Leftarrow (\Gamma Y_n, \Gamma X_n)$.
    if $[p_1, p_2, \ldots, p_n] \geq \overline{B_n^m}$ **then**
        Insert $[p_1, p_2, \ldots, p_n]$ into Q_n.
        $\overline{B_n^m} \Leftarrow \min_j (B_n^j)$
    end if
end while