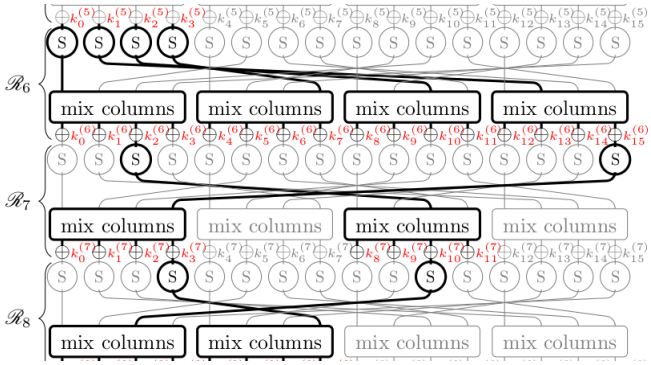
Graz University of Technology
IAIK
Institute for Applied Information Processing and Communications
Inffeldgasse 16a
A–8010 GRAZ

IAIK
TU Graz

Bachelor Thesis



Automated Security Proofs for Symmetric Ciphers

Version 1.0

Franz Scherr

Supervisor: Maria Eichlseder

Automated Security Proofs for Symmetric Ciphers

Franz Scherr

September 26, 2015

Abstract

We describe a general approach to computationally prove security bounds for ciphers against differential cryptanalysis. In particular the problem is reduced to a linear integer optimization problem that finds upper bounds for the probability of the most probable differential trail. This requires the modelling of the encryption algorithm in terms of linear equalities. Subsequently the model is evaluated by mixed-integer linear program solvers to find the minimum of active S-boxes. In succession, further approaches to extend the level of detail included in the models are introduced. Especially the Advanced Enryption Standard - AES is examined by this method and also proven against a related-key differential attack. In conclusion, the permutation introduced by the CAESAR candidate Artemia is investigated.

Keywords: differential cryptanalysis, mixed-integer linear programming, MILP, AES, Artemia, related-key

1 Introduction

Encryption of data is an important component in a fast evolving digital world. Encryption in terms of symmetric primitives basically generates an unreadable ciphertext depending on a secret key. This ciphertext can then be transmitted over an insecure, interceptable channel without allowing eavesdroppers, who do not know the secret key, to read the original content, the plaintext. The legitimate receiver of this message, who knows the secret key, then is able to decrypt the ciphertext to obtain the readable plaintext.

Particularly since much digital communication is performed over the internet, an open network, encryption plays an important role in nearly every area that involves electronic data processing.

To keep the confidentiality of private messages, encryption algorithms, ciphers, are utilized to encrypt the message transmitted. Designers of encryption algorithms are interested in developing a cipher that is resistant against attacks that could exploit certain properties of the algorithm to reveal the plaintext message without knowing the secret key or to reveal the secret key itself. One of the most famous cryptanalytical attacks for symmetric block ciphers is differential cryptanalysis, which basically tracks differences of pairs of plaintexts through the cipher. Another one is linear cryptanalysis, which tries to find likely occuring linear approximations of particular functions in the algorithm. In this thesis we will focus on proving a ciphers security against differential cryptanalysis. Especially the differential cyptanalytical properties of block ciphers based on substitution and permutation networks are investigated.

There are also other security primitives besides the prominent symmetric block ciphers, such as hash functions, securing the integrity of a message, and asymmetric ciphers used for many purposes.

The remainder of this thesis is organized as follows: Section 2 will introduce the ideas of differential cryptanalysis and how it can be applied to reveal information of the secret key. The subsequent Section 3 is intended to discuss a generic method to reduce the proof of a lower bound of security of a cipher to an optimization problem. Also examples of modelling certain parts of an encryption algorithm to an optimization problem are given. The following Section 4 presents the first real application of this technique applied to the widely used Advanced Encryption Standard – AES. From this model the ciphers security against differential cryptanalysis with also allowing related-keys is deduced. In the last Section 5 the differential properties of the permutation Artemia used in a cipher participating in the CAESAR competition is investigated.

2 Differential Cryptanalysis

This section is intended to introduce the main concepts of differential cryptanalysis. Also we will start examining the properties of basic parts of a block cipher in terms of differential cryptanalysis and how it can be applied to reveal bits of the secret key of a toy cipher.

2.1 The Basic Idea

Differential cryptanalysis by Biham et al. [BS90] is one of the most commonly used attacks on a block cipher. Designing a new block cipher also takes into account the resistance against this attack. Suppose there are two plaintext messages P_1 and P_2 with a certain known difference $\Delta_1 = P_1 \oplus P_2$ in terms of the XOR operation, the addition in $\mathbb{F}_2$. The basic idea is to track the transformation of this difference in the input as it propagates through the cipher and to predict what difference can be expected on certain intermediate variables of the cipher with a particular probability. Hence a differential attack is mostly a **chosen plaintext attack** since the attacker wants to chose messages that have a fixed difference of own choice.

Many block ciphers, including AES, are designed after a special pattern, the so-called substitution permutation network (SPN), which mainly consists of key mixing, a linear transformation mainly for diffusion and non-linear functions that mostly happen to be substitution boxes (short: **S-boxes**) for confusion. This combination of substitution, permutation and key mixing is then applied several times and is referred to as a **round**.

S-boxes often are simple lookup tables which map a given value to another in a nonlinear way. In Section 2.1.3 these functions will be discussed extensively with regard to differential cryptanalysis.

2.1.1 Key Mixing

First, consider the key mixing: Suppose we have a state of the cipher, that gives the intermediate variables after a certain amount of rounds (and parts of the current round, e.g. diffusion layers) are applied, consisting of bits x_i, then the key bits k_i are introduced to the state by the XOR operation like $y_i = x_i \oplus k_i$. Consider a pair of states $x_{1,i}$ and $x_{2,i}$, then the difference of this pair is $\Delta x_i = x_{1,i} \oplus x_{2,i}$. We now want to know how this difference changes after the key is introduced. It follows that

$$\Delta y_i = y_{1,i} \oplus y_{2,i} = x_{1,i} \oplus k_i \oplus x_{2,i} \oplus k_i = x_{1,i} \oplus x_{2,i} = \Delta x_i \ ,$$

as we require the key bits to be the same in both encryptions. The difference passes the key mixing without any change.

2.1.2 Linear Transformation

A linear transformation is a mapping that is considered linear if certain properties hold: Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^m, x \mapsto f(x)$ be a mapping. If $f(a + \alpha b) = f(a) + \alpha f(b)$ holds $\forall a, b \in \mathbb{F}_2^n$ with $\alpha \in \mathbb{F}_2$ being an arbitrary constant, then the mapping f is considered a linear one. Therefore we can express the difference at the output exactly in terms of the difference at the input when considered bitwise. Let x_1, x_2 be the inputs of a linear transformation f with a particular difference $\Delta = x_1 \oplus x_2$. Then the difference on the output of the linear transformation is given by

$$f(x_2) \oplus f(x_1) = f(x_1 \oplus \Delta) \oplus f(x_2) = f(x_1) \oplus f(\Delta) \oplus f(x_1) = f(\Delta) \ .$$

But there are simplifications and other properties to be discussed in Section 3.

2.1.3 Nonlinear Transformations, S-box

The behaviour of differences passing through an S-box is different and more tricky to handle. Since the S-box is a nonlinear function, the difference on the output is not only dependent on the difference itself but also on the current state and thus also on the key bytes. However, we still want an expression on how a difference changes when an S-box is applied. Hence, a fruitful approach is to describe the differential behaviour of these nonlinear S-boxes in a probabilistic manner. Although we cannot state exactly how differences map to each other, we can state how they most likely will. To express how a difference α changes when an S-box is applied, we introduce the notation $\alpha \xrightarrow{S} \beta$ for the event that the S-box, denoted by S, transforms the input difference α to the output difference β which occurs with a certain probability $P(\alpha \xrightarrow{S} \beta) \geq 0$ and call it a **differential** through the given S-box. Obviously it is required to know the probability that this differential occurs. To achieve the knowledge of all differential probabilities it is necessary to investigate all possible inputs. Let again α denote a fixed difference and without loss of generality $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a nonlinear mapping, an S-box for example. For this fixed difference α, we also keep a counter $N_{\alpha,\beta}$ for every possible output difference β, logically there are 2^n counters for this fixed input difference α. Now let a and b be two n-bit input bitstrings and let a count from zero to its maximum value. To obtain this certain difference α we compute b to be $b = a \oplus \alpha$. Consider now the resulting difference on the output $\beta = S(a) \oplus S(b)$ and increase its associated counter. Every input difference maps to an output difference so the sum of all counters associated to one specific input difference has to be $\sum_{\beta \in \mathbb{F}_2^n} N_{\alpha,\beta} = 2^n$.

For simplification, it is assumed that all inputs to S-boxes are evenly distributed. Then it can be stated that a differential through an S-box has a probability

$$P(\alpha \xrightarrow{S} \beta) = \frac{N_{\alpha,\beta}}{2^n}.$$

Since there are 2^n possible input differences and 2^n possible output differences, it follows that the values of $N_{\alpha,\beta}$ can be stored in a quatratic matrix $N \in M(2^n \times 2^n)$ with integer values. This table or matrix N is referred to as the **differential distribution table** which is used in the following sections to attack a simple cipher.

Algorithm 1 Simple algorithm that gerenates the differential distribution table

$N \in M(2^n \times 2^n)$
$(N_{\alpha,\beta})_{(\alpha,\beta)\in\{0,1,\ldots,2^n-1\}^2} \leftarrow 0$
for $\alpha \in \mathbb{F}_2^n$ **do**
  for $a \in \mathbb{F}_2^n$ **do**
    $b \leftarrow a \oplus \alpha$
    $\beta \leftarrow S(a) \oplus S(b)$
    $N_{\alpha,\beta} \leftarrow N_{\alpha,\beta} + 1$
  end for
end for
return N

2.2 Differential Cryptanalysis and Key Recovery for a Simple Cipher

To clearify the elements of differential cryptanalysis, consider the toy example of **Cipher Two** in [KR11]. This cipher basically consists of two rounds, and a so-called pre-whitening before the first round.

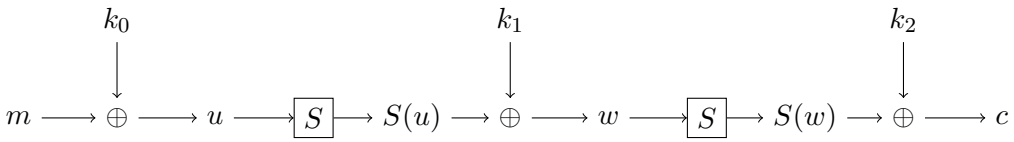


Figure 1: Structure of encryption with Cipher Two

2.2.1 Encryption, Decryption

In the following, let m_i and $k_{0,i}$ denote the i-th bit of message m and key k_0, respectively.

Given a message m of 4-bit length the cipher begins the encryption by adding the pre-whitening key k_0. The intermediate state u is given by $u_i = m_i \oplus k_{0,i}$. Then two rounds that are defined next take effect on this intermediate state.

A round of this cipher consists of the application of an S-box and an addition of a round key $k_{j,i}$, where j denotes the current round. For example, let u be the input to the round. Then a 4-bit S-box is applied to give $S(u)$. In the next step, the 4-bit round key is added to finally give the output of a round $w = S(u) \oplus k_{j,i}$.

Therefore, there are 3 round keys, each with a length of 4-bit. Since the cipher does not include a key scheduling algorithm, the round keys are completely independent and the total key length is 12-bit.

For decryption, we simply move backward through the cipher, adding the same round keys. It follows, that $c \oplus k_2 = S(w) \oplus k_2 \oplus k_2 = S(w)$. Finally, we compute the inverse mapping of the S-box S^{-1} to obtain the input to the last round w. Analogically the remaining rounds are decrypted.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	6	4	C	5	0	7	2	E	1	F	3	D	8	A	9	B

Table 1: S-box of Cipher Two (hexadecimal notation).

2.2.2 Creation of a Differential Trail

To develop a differential attack, it is necessary to address the differential properties of the used 4-bit S-box defined in Table 1. To get the most promising differential trail through the cipher, we need differences that pass the S-boxes with a high probability. Therefore, we need to compute the differential distribution table of the used 4-bit S-box. This table or matrix will have 16×16 entries in total with the resulting values given in Table 2. This matrix can easily be generated using for example Algorithm 1. Let

·	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	6	-	-	-	-	2	-	2	-	-	2	-	4	-
2	-	6	6	-	-	-	-	-	-	2	2	-	-	-	-	-
3	-	-	-	6	-	2	-	-	2	-	-	-	4	-	2	-
4	-	-	-	2	-	2	4	-	-	2	2	2	-	-	2	-
5	-	2	2	-	4	-	-	4	2	-	-	2	-	-	-	-
6	-	-	2	-	4	-	-	2	2	-	2	2	2	-	-	-
7	-	-	-	-	-	4	4	-	2	2	2	2	-	-	-	-
8	-	-	-	-	-	2	-	2	4	-	-	4	-	2	-	2
9	-	2	-	-	-	2	2	2	-	4	2	-	-	-	-	2
A	-	-	-	-	2	2	-	-	-	4	4	-	2	2	-	-
B	-	-	-	2	2	-	2	2	2	-	-	4	-	-	2	-
C	-	4	-	2	-	2	-	-	2	-	-	-	-	-	6	-
D	-	-	-	-	-	-	2	2	-	-	-	-	6	2	-	4
E	-	2	-	4	2	-	-	-	-	-	2	-	-	-	-	6
F	-	-	-	-	2	-	2	-	-	-	-	-	-	10	-	2

Table 2: Differential distribution table of the used 4-bit S-box. Input difference gives the row index and the output difference gives the column index (hexadecimal notation).

us examine the basic features of a differential distribution table in general: Inspect the first row with zero input difference. Since zero input difference means that both input strings are the same, they have to map to the same output resulting in zero output difference for all possible inputs. Therefore the first entry in the first row is 16 and the other columns are zero.

Consider two inputs that have a difference. The S-box is required to be invertible and hence $S(x)$ is an injective mapping. This causes that two inputs cannot map to the same outputs and due to this fact the entries in the first column are zero except the first.

The symmetry of the difference operation $a \oplus b = b \oplus a$ also causes that there are only even numbers in the table.

An attacker would now look for high entries in this table because its associated differential characteristics have the highest probabilty. The probability that an input difference of hexadecimal F maps to an output difference of hexadecimal D is $P\Big((1,1,1,1) = \alpha \xrightarrow{S} \beta = (1,1,1,0)\Big) = \frac{10}{16} = \frac{5}{8}$ and is very likely to occur. Note that no real cipher would use an S-box with such high differential probabilities. This is only a toy example.

An attacker therefore could require the encryption of message pairs with a difference of hexadecimal F, and since we know that key mixing has no effect on the difference,

if the same key is used in both encryptions, we have that $P(\Delta m \to \Delta w) = \frac{5}{8}$ with $\Delta w = (1, 1, 1, 0)$. This high probability could then be exploited by an attacker to reveal bits of the secret key of the last round k_2.

Suppose n_m pairs of messages with a difference of hexadecimal F are required to be encrypted. To find out the bits of k_2, we are going to try out every possible value and keep a counter associated to every possible key.

Let m_1 and m_2 be two arbitrary messages with the required difference and c_1 and c_2 their associated ciphertexts. We now proceed to compute backwards from the ciphertexts. Let $k_2^{(i)}$ denote an attempt value for the last round key. It follows that $S(w_1^{(i)}) = c_1 \oplus k_2^{(i)}$ and since the used S-box is invertible, we have that $w_1^{(i)} = S^{-1}(c_1 \oplus k_2^{(i)})$. The same computation is done for c_2 to get $w_2^{(i)}$. In the next step $\Delta w^{(i)} = w_1^{(i)} \oplus w_2^{(i)}$ is calculated, and if this difference matches hexadecimal D, the counter for this key is incremented. This is repeated n_m times for every message pair and also for every possible key.

Since we assumed the messages to be evenly distributed, we would therefore expect the counter for the right key to have a count of approximatly $n_m \cdot \frac{5}{8}$. Keys whose associated counters are in a close range of the expected differential probability are called candidate keys.

Once k_2 is fixed, one can proceed to determine the key bits of k_1. To do so, we again compute backwards and try for each possible value of k_1. Compute $u_1^{(i)} = S^{-1}(S^{-1}(c_1 \oplus k_2) \oplus k_1^{(i)})$ and $u_2^{(i)}$ analogously. We compute $\Delta u^{(i)} = u_1^{(i)} \oplus u_2^{(i)}$. Since we know that $\Delta u = \mathrm{F}$ with probability 1, we can remove every key that would give a difference $\Delta u^{(i)}$ different from hexadecimal F for any message pair.

If k_1 is fixed, k_0 would be revealed by simple trying. Differential cryptanalysis revealed all secrect round keys of CipherTwo requiring about $\frac{1}{P_{\text{diff}}} = \frac{8}{5}$ chosen plaintext pairs.

To review differential cryptanalysis, note that every S-box would give more uncertainty to a differential trail and would therefore make key recovery more difficult. To prove that a cipher is resistant against differential attacks, it has to be shown that more chosen plaintext pairs are required for analysis than attempts for the key in a bruteforce search. A possible approach is to show that in the best differential trail, the differential probability is below a given threshold. A differential trail's probability can be approximated by the probability of the most likely differential through an S-box to the power of the number of S-boxes through which a difference passes. These S-boxes are referred to as active ones. An approach to find the best trail is discussed in the next section.

3 Mixed-Integer Linear Programming

In order to get the best differential trail which utilizes the fewest active S-boxes, we reduce this problem to an integer optimization problem.

Linear programming names a method to optimize (minimize or maximize) a given linear objective function $\sum_{i \in I} c_i x_i$, with c_i beeing constants, under constraints expressed in linear inequalities, for example:

$$b_l \leq \sum_{i \in J} r_i x_i \leq b_u \ .$$

In some applications it is also necessary to require that some of the variables x_i are

integers: $\{x_i | i \in M_{\text{Integers}}\} \subset \mathbb{Z}$. Problems of this character are then called **Mixed-Integer Linear Programs**.

We now want to use this method to create a model based on mixed-integer linear programming that gives as a result the minimum number of active S-boxes.

In order to set up an objective function to be minimized, we introduce variables that are 1 if and only if the corresponding S-box is active and 0 otherwise. The objective function would then simply consist of the sum of all those variables that determine if an S-boxes is active. If the trail of minimum active S-boxes is obtained by the solver, the result can be used to **prove security bounds against differential cryptanalysis** for a cipher, since we know that the solution of the linear program forms the most probable differential trail and utilizes a particular amount of S-boxes that causes the probability of a differential trail to be under a certain limit. To remark, note that the probability of a differential is not the same as the probability of a trail. A differential is considered as a difference that transits to another at the output of a certain point, whereas the differential trail is the exact path of differences through the cipher. Nicky Mouha et al. [MWGP11] described a framework for exactly that purpose.

3.1 1-bit XOR Described in Terms of Integer Linear Programming

To start with an example consider the 1-bit exclusive or operation. XOR denoted by $\oplus$ is linear in $\mathbb{F}_2^n$, but the linear equations introduced for mixed-integer linear programming are meant to be linear in $\mathbb{Z}$. Let x_1, x_2 and y be the input and the output respectively so that $y = x_1 \oplus x_2$. Let $\Delta x_1, \Delta x_2, \Delta y$ denote the differences of the corresponding bits in a pair of XOR operations. Examine the expression $\Delta x_1 + \Delta x_2 + \Delta y =: K$ when $+$ denotes the usual addition in $\mathbb{Z}$. It holds that $K = 2d$ where d is a dummy variable securing that the all zero case is possible when $d = 0$. Note that this sum is always two since if we change only one input variable the output also changes. Changing both input variables is the only way that does not change the output apart from the trivial case.

3.2 Truncated Differences

Modelling in a bitwise fashion usually leads to a model with a huge amount of variables and constraints and is therefore hard to solve. A different approach which simplifies computation is to use truncated differences. Let Δx_i denote if the i-th bit in a byte or generally a vector containing n bits, referred to as a word of n-bit length, has a difference. Suppose $\Delta x = \max\{\Delta x_i | i \in \mathbb{N}_0 \wedge 0 \leq i < n\}$ is one if any bit has a difference and zero if the word stays the same.

Let $\Delta x_1, \Delta x_2$ and Δy indicate if there are differences in input and output, respectively. To clearify, these variables are binary and can only take values of 0 if the word stays the same and 1 otherwise. We can now describe the XOR operation in a very similar way as in a bitwise formulation, but since there is no information about which bits have a difference, it is also possible that all three words have a difference. The

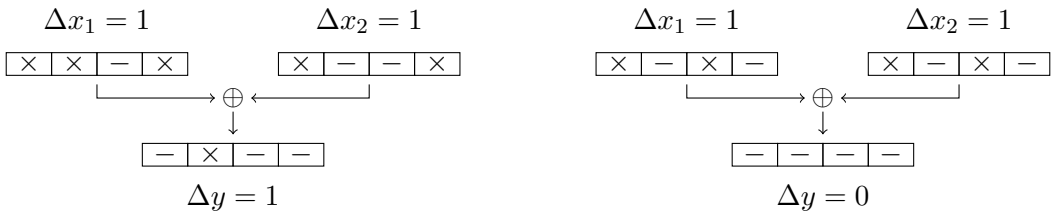


Figure 2: Exclusive or operation on 2 nibbles of 4-bits length visualizing truncated differences.

obtained inequalities are formulated in $\mathbb{Z}$.

$$
\begin{aligned}
& 2d \leq \Delta x_1 + \Delta x_2 + \Delta y \leq 3d \\
& \Leftrightarrow \\
& \Delta x_1 + \Delta x_2 + \Delta y \geq 2d \\
& \wedge\, d \geq \Delta x_1 \\
& \wedge\, d \geq \Delta x_2 \\
& \wedge\, d \geq \Delta y
\end{aligned}
$$

Where d is again a dummy variable that is zero if no input or output is active, demanding that all involved bytes or words are zero. d is one if any input is active and then requires the sum to be greater than two, which we refer to as the **differential branch number**. This number is defined as the minimum number of bytes or words in input and output that contain differences excluding the trivial case when all of them are the same.

3.3 Arbitrary Linear Transformations

Suppose there is an arbitrary linear transformation $T(\cdot)$ with n input and output words. Again consider the differences truncated on the given word size. Let Δx_i and Δy_i denote the input and output word differences. The differential branch number notated as $\mathcal{B}_D$ can now formally be defined as

$$
\mathcal{B}_D = \min_{\exists \Delta x_i, \Delta y_i \neq 0} \left\{ \sum_{i=0}^{n-1} \Delta x_i + \sum_{i=0}^{n-1} \Delta y_i \right\}.
$$

The differential behaviour of this linear transformation is easily described using the differential branch number. Two linear inequalities are obtained, but it is also possible to formulate it using simpler inequalities as stated in Section 3.2.

$$
\mathcal{B}_D \cdot d \leq \sum_{i=0}^{n-1} (\Delta x_i + \Delta y_i) \leq 2n \cdot d
$$

where d is a dummy binary variable that is required to be one if any involved word is active (right inequality), securing then that the sum of active words to be greater or equal than the differential branch number (left inequality).

4 Active S-boxes in AES-128

The **Advanced Encryption Standard** [DR02] abbreviated as **AES** is one of the most important ciphers at the time of writing and is also known as the **Rijndael** algorithm. The following introductory subsections are intended to give an overview on the encyption algorithm of AES. Subsequently, security bounds for AES are computed using a MILP model, for both a normal key model as well as a related-key model.

4.1 Structure of Encryption of AES

The encryption of AES is organized in rounds. The state size, the length of an input block to be encrypted, is given by 16 bytes and does not change by the operations within a round. AES is designed as a substitution and permutation network (SPN). Therefore, a round includes the application of an S-box together with a linear transformation. Rijndael is defined with various key sizes, including 128-bit, 192-bit or 256-bit. Depending on the used size, Rijndael applies a round 10, 12 or 14 times respectively.

As an introduction we apply MILP to compute the minimum of active S-boxes of AES. In order to keep this computation efficient, we consider truncated differences on bytes. Since AES is a byte-oriented cipher and we aim to get a lower bound for the count of active S-boxes, this seems to be a feasible simplification of an exact bitwise model of AES.

The basic AES round update includes following operations:

1. SubBytes: On every byte of the state the AES S-box will be applied

2. ShiftRows: Consider the state as a 4×4 matrix of bytes. Then every row is shifted circularly to the left by its row index. For example consider the state matrix S_b before and after shift rows.

$$S_b = \begin{pmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{pmatrix} \xrightarrow{SR} \begin{pmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_5 & b_9 & b_{13} & b_1 \\ b_{10} & b_{14} & b_2 & b_6 \\ b_{15} & b_3 & b_7 & b_{11} \end{pmatrix} =: SR(S_b)$$

3. MixColumns is a linear transformation over $\mathbb{F}_{2^8}$ with the irreducible Rijndael polynomial

$$R_p = X^8 + X^4 + X^3 + X + 1$$

and can be written as a 4×4 matrix that is premultiplied with the state.

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

Let $SR(S_b)$ and S_a be state matrices before and after mix columns respectively. We then have

$$S_a = M \cdot SR(S_b)$$

Every column vector is premultiplied with M and gives a transformed column vector.

4. AddRoundkey: The round key supplied by the key scheduling algorithm is added to the state. Addition in terms of the XOR operation $\oplus$.

We now have investigated all operations in a round and are therefore ready to describe a linear model of AES-128 (key size 128-bit) to optimize the active S-boxes to a minimum.

Encryption over n rounds	
AddRoundKey	pre-whitening
SubBytes ShiftRows MixColumns AddRoundKey	for $n-1$ rounds
SubBytes ShiftRows AddRoundKey	final round

Table 3: Structure of encryption over n rounds

4.2 Building a model with linear inequalities

Let $x_i^{(r)}$ be the difference of the state byte with index i in beginning of round r. As we consider truncated differences these variables hold 1 if the corresponding byte has any difference and 0 otherwise.

$$x_i^{(r)} = \begin{cases} 0 & \text{not any difference in byte } i \\ 1 & \text{any difference in byte } i \end{cases}$$

It follows that the differences $x_i^{(r)}$ are invariant related to the application the S-boxes, because they are injective. Therefore, a difference in the inputs of the S-boxes always results in a difference in the outputs.

Consider the ShiftRows operation. As stated before, state bytes are circularly shifted to the left by the corresponding row index. Logically, the differences in the bytes stay the same, but they are rearranged. To illustrate the point, consider the difference state matrix X_b before and after ShiftRows:

$$X_b = \begin{pmatrix} x_0^{(r)} & x_4^{(r)} & x_8^{(r)} & x_{12}^{(r)} \\ x_1^{(r)} & x_5^{(r)} & x_9^{(r)} & x_{13}^{(r)} \\ x_2^{(r)} & x_6^{(r)} & x_{10}^{(r)} & x_{14}^{(r)} \\ x_3^{(r)} & x_7^{(r)} & x_{11}^{(r)} & x_{15}^{(r)} \end{pmatrix} \xrightarrow{SR} \begin{pmatrix} x_0^{(r)} & x_4^{(r)} & x_8^{(r)} & x_{12}^{(r)} \\ x_5^{(r)} & x_9^{(r)} & x_{13}^{(r)} & x_1^{(r)} \\ x_{10}^{(r)} & x_{14}^{(r)} & x_2^{(r)} & x_6^{(r)} \\ x_{15}^{(r)} & x_3^{(r)} & x_7^{(r)} & x_{11}^{(r)} \end{pmatrix} =: SR(X_b)$$

Now consider the MixColumns operation. Let $y_i^{(r)}$ denote the difference after the linear transformation M with same indexing as $x_i^{(r)}$.

In terms of our linear model, we describe the MixColumns operation by its branch number of $\mathcal{B}_D = 5$, as we consider that every 4×1 column vector is premultiplied with the matrix M to give the associated transformed column vector. Let a and b be arbitrary

difference vectors truncated to bytes. Then the branching condition can be notated in terms of linear programming as following:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = b = M \cdot a = M \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

$$\sum_{i \in I} a_i + \sum_{i \in I} b_i \geq \mathcal{B}_D \cdot d, \qquad I = \{0, 1, 2, 3\}$$

$$\sum_{i \in I} a_i + \sum_{i \in I} b_i \leq 2 \cdot |I| \cdot d$$

In summary, the differences propagate through one round of the cipher as shown beneath.

$$\begin{pmatrix} x_0^{(r)} & x_4^{(r)} & x_8^{(r)} & x_{12}^{(r)} \\ x_1^{(r)} & x_5^{(r)} & x_9^{(r)} & x_{13}^{(r)} \\ x_2^{(r)} & x_6^{(r)} & x_{10}^{(r)} & x_{14}^{(r)} \\ x_3^{(r)} & x_7^{(r)} & x_{11}^{(r)} & x_{15}^{(r)} \end{pmatrix} \xrightarrow{SB} \begin{pmatrix} x_0^{(r)} & x_4^{(r)} & x_8^{(r)} & x_{12}^{(r)} \\ x_1^{(r)} & x_5^{(r)} & x_9^{(r)} & x_{13}^{(r)} \\ x_2^{(r)} & x_6^{(r)} & x_{10}^{(r)} & x_{14}^{(r)} \\ x_3^{(r)} & x_7^{(r)} & x_{11}^{(r)} & x_{15}^{(r)} \end{pmatrix}$$

$$\xrightarrow{SR} \begin{pmatrix} x_0^{(r)} & x_4^{(r)} & x_8^{(r)} & x_{12}^{(r)} \\ x_5^{(r)} & x_9^{(r)} & x_{13}^{(r)} & x_1^{(r)} \\ x_{10}^{(r)} & x_{14}^{(r)} & x_2^{(r)} & x_6^{(r)} \\ x_{15}^{(r)} & x_3^{(r)} & x_7^{(r)} & x_{11}^{(r)} \end{pmatrix} \xrightarrow{MC} \begin{pmatrix} y_0^{(r)} & y_4^{(r)} & y_8^{(r)} & y_{12}^{(r)} \\ y_1^{(r)} & y_5^{(r)} & y_9^{(r)} & y_{13}^{(r)} \\ y_2^{(r)} & y_6^{(r)} & y_{10}^{(r)} & y_{14}^{(r)} \\ y_3^{(r)} & y_7^{(r)} & y_{11}^{(r)} & y_{15}^{(r)} \end{pmatrix}$$

Finally, the linear program that is obtained by the linear describtion of the cipher's differential behaviour is given by:

$$\text{minimize objective:} \quad \sum_{r \in R} \sum_{i \in J} x_i^{(r)}$$

subject to:

$$\mathcal{B}_D \cdot d_j^{(r)} \leq \sum_{i \in I} x_{(5i+4j) \bmod 15}^{(r)} + \sum_{i \in I} y_{i+4j}^{(r)} \leq 2|I| d_j^{(r)}, \qquad \forall r \in R, \forall j \in I$$

$$x_j^{(r+1)} = y_j^{(r)}, \qquad \forall j \in J$$

$$I = \{0, 1, 2, 3\}, \quad J = \{0, 1, ..., 15\}, \quad R = \{1, 2, ..., n_{\text{rounds}}\}$$

Note that R ranges from 1 to n_{rounds} because in Pre-whitening there are no S-boxes. Obviously, one is able to shorten the linear program given above by immediately substituting $x_j^{(r+1)}$ for $y_j^{(r)}$. Later we will benefit from the expanded form with y. After typing in all constraints that describe AES-128, we can conclude the minimum number of active S-boxes on AES-128 to be **55**.

The same results as given by Mouha et al. [MWGP11] are obtained. The solver used for this optimization problem was `Gurobi`, which is free to students and researchers. The consumed time for the computations in Table 4.2 did not exceed a second on a quad core CPU.

4.3 Related-key differential trail on AES-128

AES-128 uses 10 full rounds with every round requiring a key of 16 bytes or equivalently 128 bits length. Therefore, the key schedule has to generate 10 additional keys, since 11 keys (pre-whitening) are needed.

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14
min. active S-boxes	1	5	9	25	26	30	34	50	51	55	59	75	76	80

Table 4: Minimum number of active S-boxes in N rounds

In order to obtain fewer active S-boxes on AES-128, we allow differences in the used key. Our intention is to provide a security proof against differential cryptanalysis even under related-key attacks. We consider a truncated model on bytes and extend the model with the key schedule algorithm.

The key schedule for AES works in a similar way for all different key sizes. The key for the next round is iteratively generated using the current key. To get a better understanding of the key scheduling of AES-128 take a look on Figure 4.3.

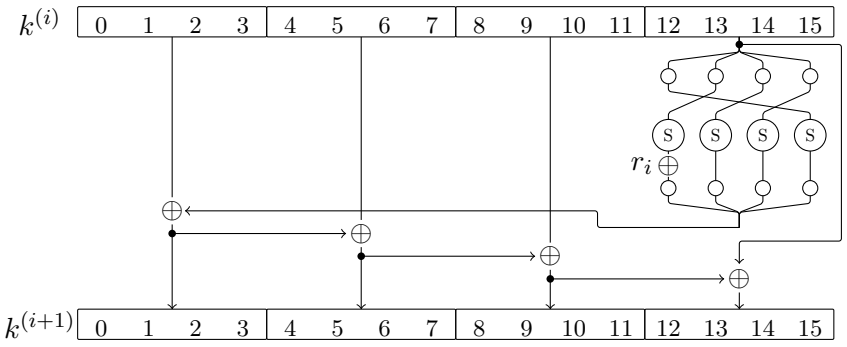


Figure 3: Key schedule algorithm

It is obviously useful to describe the key schedule in terms of 4-byte words. Let these words of the current round key be $w_j^{(i)} = k_{4j}^{(i)}||k_{4j+1}^{(i)}||k_{4j+2}^{(i)}||k_{4j+3}^{(i)}$. We further define the application of the AES S-box on a word to be the concatenation of the application of the S-box to the single bytes:

$$S(w_j^{(i)}) = S(k_{4j}^{(i)})||S(k_{4j+1}^{(i)})||S(k_{4j+2}^{(i)})||S(k_{4j+3}^{(i)}).$$

Then the words of the next round key are given by following equations.

$$\begin{aligned} w_0^{(i+1)} &= w_0^{(i)} \oplus S(w_3^{(i)} \lll 8) \oplus r_i||0||0||0 \\ w_j^{(i+1)} &= w_j^{(i)} \oplus w_{j-1}^{(i)} \qquad \forall j \in \{1,2,3\} \end{aligned}$$

where r_i is a fixed round constant of one byte size. Now since r_i is a fixed constant, it has no effect on the differences passing through the key schedule. Also note that for generating the first key word, a shift by one byte to the left takes place. The fact that the AES S-box is a mapping $S : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ is responsible that the property $S(w_j^{(i)} \lll 8) = S(w_j^{(i)}) \lll 8$ holds. Therefore the order of byte rotation and application of the S-boxes is irrelevant.

We are now ready to describe a linear model of AES extended by the key schedule, which also can include active S-boxes. This will lead to a minimum number of active S-boxes using differential attacks and allowing differences in the key. We expect the number of active S-boxes to be less because the solver has more options to optimize the differential trail, which can now include the key schedule.

Let $\Delta k_i^{(r)}$ be the difference variables of the key byte i in round r.

$$\Delta k_i^{(r)} = \begin{cases} 0 & \text{no difference in the key byte } i \text{ of round } r \\ 1 & \text{otherwise} \end{cases}$$

Using the description of the key schedule and the XOR branch number $\mathcal{B}_{\text{XOR}} = 2$ it follows that

$$\begin{aligned} \mathcal{B}_{XOR} \cdot d_i^{(r)} = 2d_i^{(r)} \leq \Delta k_i^{(r)} + \Delta k_{12+(i+1) \bmod 4}^{(r)} + \Delta k_i^{(r+1)} \leq 3d_i^{(r)} \qquad & \forall i \in \{0,1,2,3\} \\ \mathcal{B}_{XOR} \cdot d_j^{(r)} = 2d_j^{(r)} \leq \Delta k_j^{(r)} + \Delta k_{j-4}^{(r+1)} + \Delta k_j^{(r+1)} \leq 3d_j^{(r)} \qquad & \forall j \in \{4,5,...,15\} \end{aligned}$$

are the linear equations that model the key schedule, where $d_i^{(r)}$ are some dummy variables securing the branch condition for every single linear inequality.

By introducing the key mixing XOR equations, we are then ready to compose the complete linear model for the active S-boxes when allowing differences in the key.

$$\text{minimize objective:} \quad \sum_{r \in R} \left(\sum_{i \in J} x_i^{(r)} + \sum_{i=12}^{15} \Delta k_i^{(r)} \right)$$

subject to:

$$\mathcal{B}_D \cdot d_j^{(r)} \leq \sum_{i \in I} x_{(5i+4j) \ mod \ 15}^{(r)} + \sum_{i \in I} y_{i+4j}^{(r)} \leq 2 \cdot |I| \cdot d_j^{(r)} \qquad \forall r \in R, \forall j \in I$$

(mix columns)

$$\begin{aligned} \mathcal{B}_{XOR} \cdot d_{k|i}^{(r)} \leq \Delta k_i^{(r)} + \Delta k_{12+(i+1) \ mod \ 4}^{(r)} + \Delta k_i^{(r+1)} \leq 3d_{k|i}^{(r)} \qquad & \forall i \in \{0,1,2,3\} \\ \mathcal{B}_{XOR} \cdot d_{k|j}^{(r)} \leq \Delta k_j^{(r)} + \Delta k_{j-4}^{(r+1)} + \Delta k_j^{(r+1)} \leq 3d_{k|j}^{(r)} \qquad & \forall j \in \{4,5,...,15\} \end{aligned}$$

(key schedule)

$$\mathcal{B}_{XOR} \cdot d_{m|j}^{(r)} \leq x_j^{(r+1)} + y_j^{(r)} + \Delta k_j^{(r)} \leq 3d_{m|j}^{(r)} \qquad \forall j \in J$$

(key mixing)

$$I = \{0,1,2,3\}, \quad J = \{0,1,...,15\}, \quad R = \{1,2,...,n_{\text{rounds}}\}$$

To create all equations automatically, a simple `sage`-script is used and to solve this model, we again, use the `Gurobi` solver.

The objectives minimum is then given by **25** active S-boxes with a related-key model. By analyzing the differential distribution table of the AES S-box, the maximum differential propability is given by $\frac{4}{256} = 2^{-6}$. In conclusion, we obtain a maximum differential propability of $(2^{-6})^{25} = 2^{-150} < 2^{-128}$, which proves the cipher's resistance against a related-key differential attack.

To further discuss the best differential trail that was given by the solver, consider Figure 4 for the trail through the encryption rounds. Figure 5 shows the trail through the key scheduling algorithm.

Let $\Delta k_{4...7}^{(0)} =: \Delta$. It follows that $\Delta k_{8...11}^{(0)} = \Delta$, since they cancel out in the following XOR operation. After $\Delta k^{(2)}$ the same difference Δ passes through the S-boxes. Let Δ' denote the output difference of the byte rotated S-boxes. We can now conclude $\Delta = \Delta'$ to be the same difference, because they again cancel out in an exclusive or operation. These differences pass the key schedule in a linear way up to and including $k^{(6)}$.

Now let Δ'' denote the substituted and rotated difference word $\Delta k^{(6)}_{12\ldots15}$. Consequently it follows that $\Delta''' := \Delta'' \oplus \Delta \neq 0$, since $\Delta k^{(7)}_{0\ldots3}$ is different from zero. But we also have that $\Delta''' \oplus \Delta = 0$. This is a contradiction since $\Delta''' \oplus \Delta = \Delta'' \neq 0 = \Delta''' \oplus \Delta \quad \lightning$. The trail found by the solver is valid in terms of the linear model, but invalid in terms of the real behaviour of the differences considered on exact bit level. While this trail is not consistent, that does not imply that there is no other valid trail with 25 active S-boxes.

Although the lower bound of active S-boxes proves the cipher's resistance against a related-key differential attack, one could improve the linear model by introducing an exact bitwise representation of AES-128. The idea is to model the linear layers as they are. The application of an S-box would then allow the bit pattern to change to any other. Bitwise models were also introduced by Sun et al. [SHW+14]. Allowing the bit pattern to change to any other would also allow impossible differential trails. To prevent this from happening, there is the possibility to introduce addional constraints (linear inequalities) that prohibit these trails. As stated in by Sun et al. [SHW+14] a systematic approach to create such additional constraints is to generate the convex hull of all possible differentials through the AES S-box. In general the convex hull of an S-box consists of hundreds or thousands of inequalities, making it impossible to wipe out all of the impossible differentials. It then is good practice to add only such inequalities that prohibit the most impossible differentials. In `sage` there is already a mechanism which can generate the convex hull, see especially Section 5.7, and as part of this project, we tried to compute it for the AES S-box, but the computation in `sage` is not parallelized. Therefore no results were achieved and no convex hull was obtained. Since the security bound is proven, there is no special reason to further pursue the computation of the convex hull.

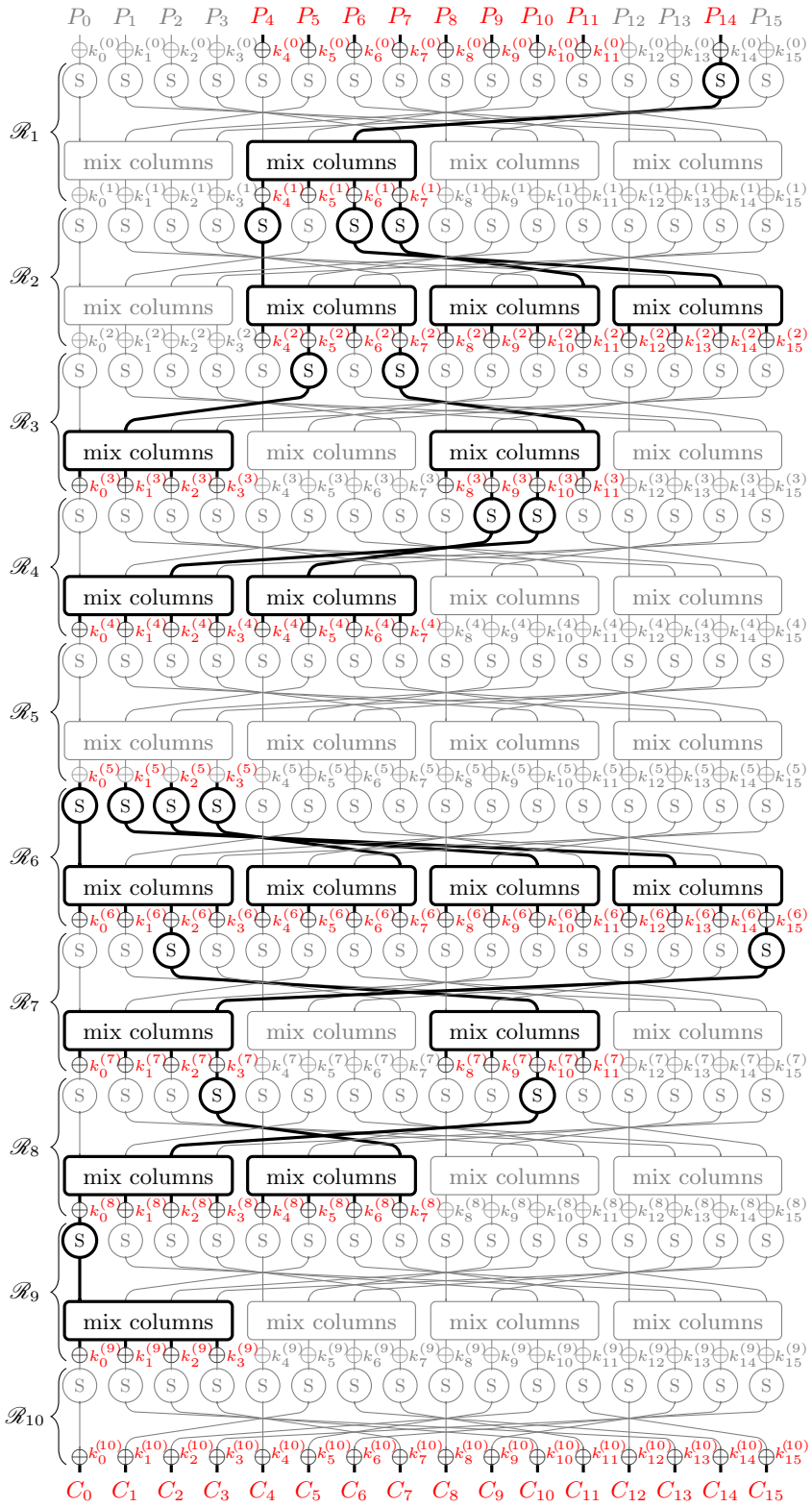


Figure 4: Visualizing differential trail through AES 128

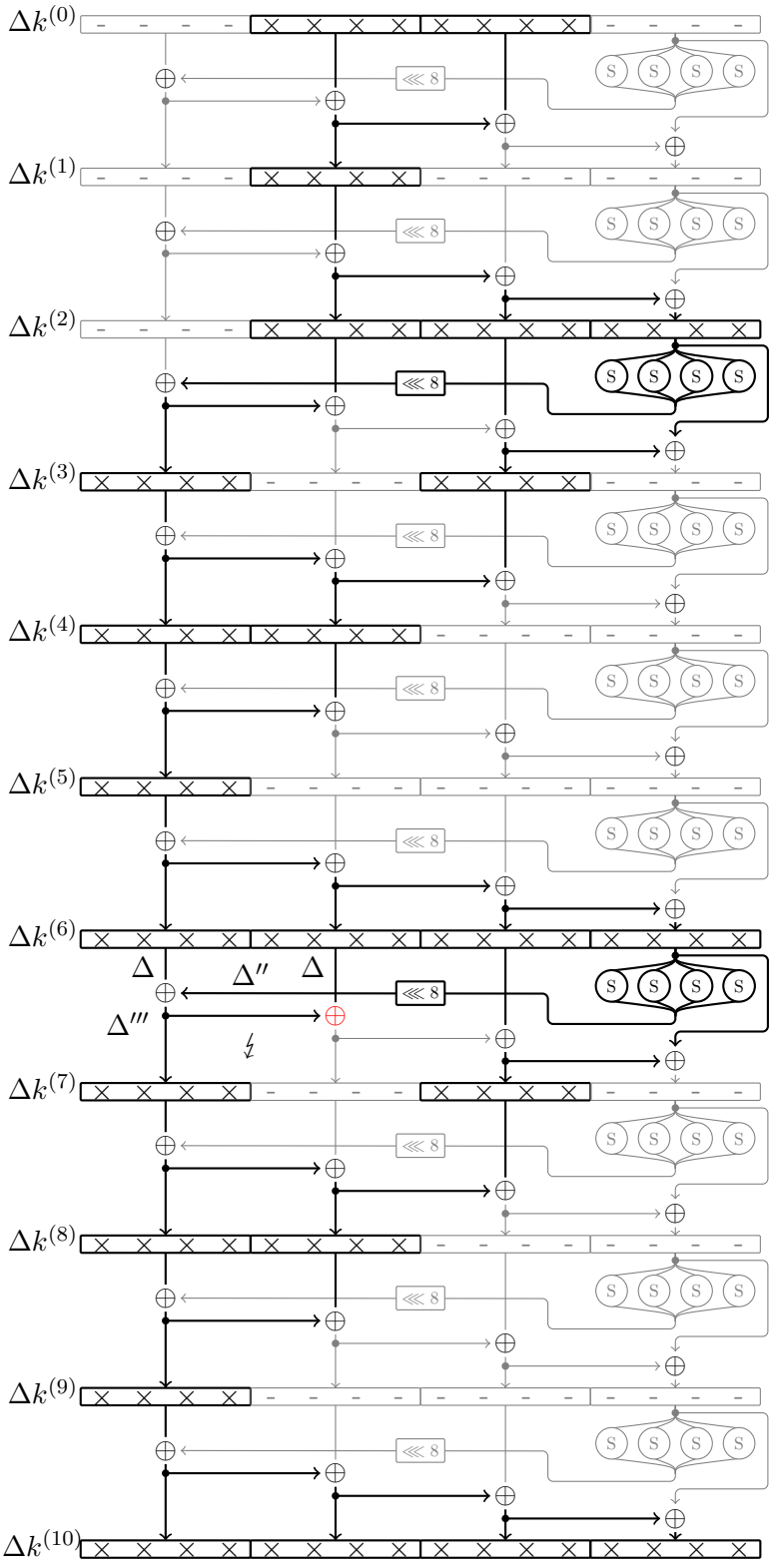


Figure 5: Schematic key schedule AES 128

5 Active S-boxes in Artemia

5.1 Description of Artemia

As part of this project the MILP technique was also applied on a permutation named Artemia. Artemia is an interesting permutation that was introduced in an authenticated cipher as a candidate of the CAESAR competition.

The permutation is specified for block sizes of 512 and 256 bits and can be representated as mappings $\mathbb{F}_2^{512} \to \mathbb{F}_2^{512}$ and $\mathbb{F}_2^{256} \to \mathbb{F}_2^{256}$ respectively. To firstly discuss the properties of Artemia with input size of 512 consider the structural representation given in Figure 6. At the beginning of each round a round dependent constant is added to the current state. Then the state is divided into four words of 128-bit length and a recursive linear transformation $D1$ is applied, generating four words of output. The AES S-box is now applied on each of the output words on every byte.

Next consider these substituted 128-bit words. Each of these is again divided into 4 32-bit words and transformed by a recursive linear layer $D2$, generating 4 32-bit ouput words each. The AES S-box is now again applied on each of the 32-bit words bytewise given by each $D2$ block.

Then these substituted 32-bit words on the output of $D2$ are again divided by 4 to give 4 single bytes that are now transformed by a recursive linear layer $D3$. To finish the permutation, the AES S-box is applied one last time on each byte of the state.

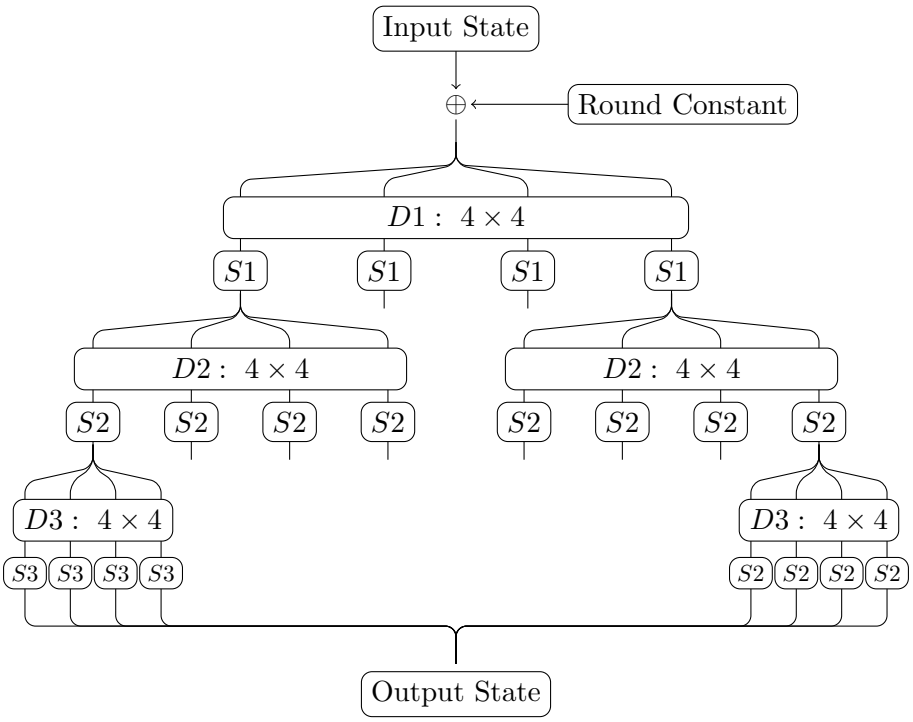


Figure 6: Artemia round with 512-bit input and output size

5.2 Linear Transformations $D1$, $D2$, $D3$

The linear transformations used in Artemia are of recursive nature and take 4 input words to generate 4 output words of the same size. The properties of recursive diffusion layers like these are discussed by Sajadieh et al. [SDMS15], also a proof of its perfect branching number is given by [SDMS15, Theorem 7].

5.2.1 Linear Transformation $D1$

Let X_i denote the input words and Y_i denote the output words. Each word in transformation $D1$ is of 128-bit size. Then the transformation is given by

$$\begin{aligned} Y_0 &= X_0 \oplus X_2 \oplus X_3 \oplus L(X_1 \oplus X_3) \\ Y_1 &= X_1 \oplus X_3 \oplus Y_0 \oplus L(X_2 \oplus Y_0) \\ Y_2 &= X_2 \oplus Y_0 \oplus Y_1 \oplus L(X_3 \oplus Y_1) \\ Y_3 &= X_3 \oplus Y_1 \oplus Y_2 \oplus L(Y_0 \oplus Y_2) \end{aligned}$$

where $L(X)$ is a linear mapping $L(X) = (X \ll 1) \oplus (X \gg 3)$.

5.2.2 Linear Transformation $D2$

This transformation is of identical structure of $D1$ except that the input and output word size is 32-bit.

5.2.3 Linear Transformation $D3$

In the $D3$ diffusion layer the input and output words are of the size of a byte. The equations describing the output are the same as in $D1$ and $D2$. However in contrast to the previous diffusion layers the linear mapping $L(X)$ is now given by

$$L(X) = (X \oplus (X \ll 1)) \lll 1.$$

5.3 A simple approach

In the Artemia submission paper by Alizadeh et al. [JAB14], the minimum number of active S-boxes in two rounds are given by **45**, reasoned by a clever argumentation: Suppose that a recursive $D3$ diffusion layer has been active. Since the branch numbers of all linear transformations are $\mathcal{B}_D = 5$ an active $D3$ layer guarantees at least five active S-boxes in $S3$ and $S2$. By implication an active $S2$ is caused by an active $D2$ and guarantees five active S-boxes in $S2$ and $S1$ for its part. As a result an active 128-bit word at the output of $D1$ in round i causes **9** active S-boxes in round i. Because an active word on the input of $D1$ in round i results in at least nine active S-boxes in the previous round $i-1$ and the branch number of $D1$ is given by five, this guarantees a minimum number of **45** active S-boxes in two rounds.

To automatically prove this statement we create a linear model of Artemia. In contrast to the linear model of AES it is now necessary to change the differential activity through the different word sizes in the different layers of the permutation.

Let $x_{\ell,i}^{(r)}$ denote the difference of the word around the diffusion layer $D[\ell+1]$ with index i in round r. Since there are 4 $D2$ and 16 $D3$ diffusion layers the input words of

all diffusion blocks are indexed from left to right. Then the output words of all blocks are indexed from left to right.

For example $x_{1,0}^{(r)}$ denotes the first input word of the first $D2$ block, $x_{1,16}^{(r)}$ denotes the first output word of the first $D2$ and $x_{1,4}^{(r)}$ denotes the first input word of the second $D2$ block.

As we know the branch number of all linear transformations we proceed to add the branch conditions of each diffusion block to the linear model.

$$\mathcal{B}_D \cdot d_{\ell,j}^{(r)} \leq \sum_{i \in I} x_{\ell,4j+i}^{(r)} + \sum_{i \in I} x_{\ell,4^{\ell+1}+4j+i}^{(r)} \leq 2|I| d_{\ell,j}^{(r)} \quad j \in \{0, 1, ..., 4^l - 1\}$$
$$r \in R = \{0, 1, ..., n_{\text{rounds}}\}, \quad \ell \in \{0, 1, 2\}, \quad I = \{0, 1, 2, 3\}$$

ℓ denotes the layer under consideration ($D1$: $l = 0$, $D2$: $l = 1$, ...) and i gives the word index at this block. Since there are $4^{\ell+1}$ input words in a layer with index ℓ, we have to add this to the index when accessing the output words.

Consider the different word sizes. It is required to expand the differential activity of a word of size s to the activity of 4 words of size $\frac{s}{4}$. Let a be the word that is 4 times bigger as each of the 4 words b_i. To connect a and $b_0||b_1||b_2||b_3$, certain conditions have to be fulfilled. Suppose a is active, it is now required that one of the b_i words is active. On the other hand suppose that at least on of the b_i is active, it follows that a must be active too. In terms of linear inequalities this behaviour can be modelled with the following constraint.

$$a \leq \sum_{i=0}^{3} b_i \leq 4a$$

Putting all constraints together a MILP model for Artemia is obtained. The objective to minimize is again the number of active S-boxes and is given by

$$\sum_{r \in R} \Big(\sum_{i=0}^{127} x_{2,i}^{(r)} + \sum_{i=0}^{15} x_{1,i} \Big).$$

The results obtained by this model are exactly the stated bounds by the CAESAR submission paper [JAB14]. Note that the computation of the model of each round count

N	1	2	3	4	5
$min(s_n)$	9	45	54	90	99

Table 5: Minimum number of active S-boxes in N rounds of Artemia

stated in the table was almost instantly with the `Gurobi` solver, whereas `GLPK` did not find the optimal solution for 2 rounds in 11 hours.

The differential trail found by this model happens to be concatenable and the number of active S-boxes is then given by the relation

$$N(n) = \begin{cases} 45n + 9 & n \text{ odd} \\ 45n & n \text{ even} \end{cases}$$

This motivates the design of a more precise model that also pays attention to more detailed properties of the linear transformations $D1, D2, D3$.

5.4 Properties of the Linear Transformation

To get clearer insights in how the linear transformation $D[1,2,3]$ works, it is necessary to investigate its definition.

Since D is given in recursive description

$$\begin{aligned}
Y_0 &= X_0 \oplus X_2 \oplus X_3 \oplus L(X_1 \oplus X_3) \\
Y_1 &= X_1 \oplus X_3 \oplus Y_0 \oplus L(X_2 \oplus Y_0) \\
Y_2 &= X_2 \oplus Y_0 \oplus Y_1 \oplus L(X_3 \oplus Y_1) \\
Y_3 &= X_3 \oplus Y_1 \oplus Y_2 \oplus L(Y_0 \oplus Y_2)
\end{aligned}$$

we substitute backwards to get the equations for the outputs that are only dependent of the inputs:

$$\begin{aligned}
Y_0 &= X_0 \oplus L(X_1) \oplus X_2 \oplus X_3 \oplus L(X_3) \\
Y_1 &= X_0 \oplus L(X_0) \oplus X_1 \oplus L(X_1) \oplus L^2(X_1) \oplus X_2 \oplus L^2(X_3) \\
Y_2 &= L^2(X_0) \oplus X_1 \oplus L(X_1) \oplus L^3(X_1) \oplus X_2 \oplus L(X_2) \oplus X_3 \oplus L^2(X_3) \oplus L^3(X_3) \\
Y_3 &= X_0 \oplus L^2(X_0) \oplus L^3(X_0) \oplus L(X_1) \oplus L^2(X_1) \oplus L^3(X_1) \oplus L^4(X_1) \\
&\quad \oplus L(X_2) \oplus L^2(X_2) \oplus L^2(X_3) \oplus L^4(X_3),
\end{aligned}$$

where $L^2(X) \Leftrightarrow L(L(X))$. Also keep in mind that, since the linear function L is linear, it follows that $L(a) \oplus L(b) = L(a \oplus b)$. Let D_{512} denote $D1$, since the input and output size is of 512-bits length. Equivalently D_{128} denotes $D2$ and D_{32} denotes $D3$.

Since D is a linear transformation we are able to express the mapping with a matrix

$$\begin{aligned}
&D_s : \mathbb{F}_2^s \to \mathbb{F}_2^s \\
&X \mapsto Y = D_s(X) = M_s \; X
\end{aligned}$$

$$\text{where } X = (X_0, X_1, X_2, X_3)^T \text{ and } Y = (Y_0, Y_1, Y_2, Y_3)^T$$

X and Y are then considered as vectors in $\mathbb{F}_2^s$. Let I_M denote the identity matrix in vector space $F_2^{s/4}$ and L_M denote the associated matrix of the linear function $L(X_i) = L_M \cdot X_i = (X_i \ll 1) \oplus (X - i \gg 3)$. We then have for example for $Y_0 = X_0 \oplus L_M \cdot X_1 \oplus X_2 \oplus X_3 \oplus L_M \cdot X_3$. It follows that $Y_0 = X_0 \oplus L_M \cdot X_1 \oplus X_2 \oplus (I_M \oplus L_M) \cdot X_3$ since we deal with linear operations. The linear transformation matrix can now be easily described using these submatrices of a fourth of the size. M_s is obtained to be

$$M_s = \begin{pmatrix}
I_M & L_M & I_M & I_M + L_M \\
I_M + L_M & I_M + L_M + L_M^2 & I_M & L_M^2 \\
L_M^2 & I_M + L_M + L_M^3 & I_M + L_M & I_M + L_M^2 + L_M^3 \\
I_M + L_M^2 + L_M^3 & L_M + L_M^2 + L_M^3 + L_M^4 & L_M + L_M^2 & L_M^2 + L_M^4
\end{pmatrix}$$

where each block is of size $\frac{s}{4} \times \frac{s}{4}$.

To examine the properties of function $L(X_i) = (X_i \ll 1) \oplus (X_i \gg 3)$ in D[1,2] we establish its matrix notation. Let without loss of generality $s = 128$ ($D2$) for example and $X = (x_0, x_1, x_2, ..., x_{127})^T \in \mathbb{F}_2^{128}$. It follows that L_M for $D2$ is of size 32×32. Consider the shift to the left by one position, this means that x_i is being replaced with

x_{i+1}. Shift to the right by three positions causes x_3 to be replaced by x_0 for example. To combine these observations consider following matrix representation of L_M in $D2$. It is important, that this matrix multiplication is done in a vector space over a finite field $\mathbb{F}_2^n$.

$$Y = L_M \cdot X$$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ \vdots \\ y_{125} \\ y_{126} \\ y_{127} \end{pmatrix} = \left(\begin{array}{cccccccc|cccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & & & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & & & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \dots & & & \dots & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \dots & & \dots & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & & & & & & & & \ddots & \ddots & \ddots & & \vdots \\ 0 & 0 & \dots & & & \dots & 0 & 1 & 0 & 0 & 0 & 1 & 0 & \\ 0 & 0 & \dots & & & & \dots & 0 & 1 & 0 & 0 & 0 & 1 & \\ 0 & 0 & \dots & & & & & \dots & 0 & 1 & 0 & 0 & 0 & \end{array}\right) \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ \vdots \\ x_{125} \\ x_{126} \\ x_{127} \end{pmatrix}$$

The first 8 rows form the first byte of the output word of 4 byte size of the linear function $Y = L(X)$. Note that the second byte has an effect on the last bit of the first byte. It follows that if $x_{0\dots7}$ has no difference it may be that $y_{0\dots7}$ contains a difference when x_8 was active.

Now consider the box right below. We can conclude that the activity of $y_{8\dots15}$ is dependent on the bytes $x_{0\dots7}, x_{8\dots15}, x_{16\dots23}$ or more exactly on the bits $x_{5\dots16}$.

To examine which bytes of input influence what bytes on the output, truncating this matrix on byte blocks gives insight on the dependencies. Also we want this truncated matrix to be in an integer space, as we want to investigate the constraints of the differences in terms of linear inequalities. Consider again $Y = L_M \cdot X$. As a result of the previous observations, it follows that the inequalities beneath have to hold in $\mathbb{Z}^n$.

$$\begin{pmatrix} \Delta y_{0\dots7} \\ \Delta y_{8\dots15} \\ \Delta y_{16\dots23} \\ \Delta y_{24\dots31} \\ \Delta y_{32\dots39} \\ \vdots \\ \Delta y_{120\dots127} \end{pmatrix} \leq \begin{pmatrix} 1 & 1 & 0 & 0 & \dots & & & & \dots & 0 \\ 1 & 1 & 1 & 0 & 0 & \dots & & & \dots & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & \dots & & \dots & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & \dots & 0 \\ \vdots & & & \ddots & \ddots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} \Delta x_{0\dots7} \\ \Delta x_{8\dots15} \\ \Delta x_{16\dots23} \\ \Delta x_{24\dots31} \\ \Delta x_{32\dots39} \\ \vdots \\ \Delta x_{120\dots127} \end{pmatrix}$$

In $D1$ a similar structure can be observed, the only difference is the 4 times bigger size of the involved matrices.

Since we also have multiple applications of $L(X)$ in the definition of the linear transformations it is advantageous to shortly discuss the effects of neighbouring bytes.

$$\begin{aligned} L^2(X) &:= L\big(L(X)\big) = L\big((X \ll 1) \oplus (X \gg 3)\big) \\ &= \Big(\big((X \ll 1) \oplus (X \gg 3)\big) \ll 1 \oplus \big((X \ll 1) \oplus (X \gg 3)\big) \gg 3\Big) \\ &= (X \ll 2) \oplus (X \gg 2) \oplus \big((X \ll 1) \gg 3\big) \oplus (X \gg 6) \end{aligned}$$

Clearly the influence of adjacent bytes gets higher. By induction we can also conclude that for L^3 and L^4 not only the byte to the left has an effect but in addition also the second byte to the left. When again $Y = L_M^3 \cdot X$ or $Y = L_M^4 \cdot X$, we get the following inequalities in an integer space:

$$\begin{pmatrix} \Delta y_{0\ldots7} \\ \Delta y_{8\ldots15} \\ \Delta y_{16\ldots23} \\ \Delta y_{24\ldots31} \\ \Delta y_{32\ldots39} \\ \vdots \\ \Delta y_{120\ldots127} \end{pmatrix} \leq \begin{pmatrix} 1 & 1 & 0 & 0 & \ldots & & & & \ldots & 0 \\ 1 & 1 & 1 & 0 & 0 & \ldots & & & \ldots & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & \ldots & & \ldots & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & \ldots & \ldots & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & \ldots & 0 \\ \vdots & & & \ddots & \ddots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \Delta x_{0\ldots7} \\ \Delta x_{8\ldots15} \\ \Delta x_{16\ldots23} \\ \Delta x_{24\ldots31} \\ \Delta x_{32\ldots39} \\ \vdots \\ \Delta x_{120\ldots127} \end{pmatrix}$$

See also Figure 7 to visualize.

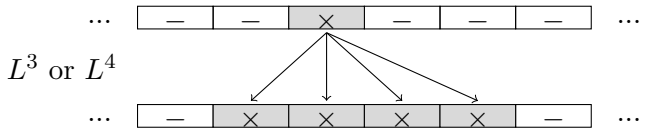


Figure 7: Possible propagation of a difference passing the linear function L 3 to 4 times.

The given dependencies already give a moderately accurate description on how differences can pass through the linear function. Still it is not exactly known how its differences can cancel out or pass into adjacent bytes when more than one byte is active. Since for $D2$ $L(X)$ operates on 32-bit words we can calculate every possible combination. To do so we simply let a 32-bit variable count from zero to $2^{32} - 1$ and give the counter value as input to $L(X)$. Then we keep track of which truncated difference pattern on the input maps to which truncated difference pattern on the output of this linear function L. Since all possible input combinations are tried, we obtain a complete table of which differentials considered as truncated differentials on byte level are possible through this function. Like a differential distribution table, this table lists all possible active input to output patterns but in a bytewise fashion. Note that 2^{32} possible combinations is moderately a little amount that does not require a parallel computation on today's standard computers. To summarize the results, one active byte can pass differences into all adjacent bytes after application of $L(X)$, also it is possible that if 3 bytes are active, after $L(X)$ is applicated there are only 2 or 1 active bytes remaining. Notable is that 2 active bytes can never change to one active byte. We again notate the activity of these 4 bytes in a hexadecimal way so that C for example means that the bytes with index 0 and 2 are active. The results are given in Table 6.

·	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1 1	1 -	- -	- -	- -	- -	- -	- -	1 -	1 -	- -	- -	- -	- -	- -	- -
1	- -	1 1	1 -	1 1	- -	- -	- -	- -	- -	1 -	1 -	1 -	- -	- -	- -	- -
2	- -	- -	1 1	1 1	- -	- -	1 1	1 1	- -	- -	1 -	1 -	- -	- -	1 -	1 -
3	- -	- -	1 1	1 1	1 -	1 1	1 1	1 1	- -	- -	1 -	1 -	1 -	1 -	1 -	1 -
4	- -	- -	- -	- -	1 1	1 -	1 1	1 -	- -	- -	- -	- -	1 1	1 -	1 1	1 -
5	- -	- -	- -	- -	- -	1 1	1 -	1 1	- -	- -	- -	- -	- -	1 1	1 -	1 1
6	- -	- -	- -	- -	- -	1 1	1 1	1 1	- -	- -	1 1	1 1	- -	1 1	1 1	1 1
7	- -	- -	- -	- -	1 1	1 1	1 1	1 1	1 -	1 1	1 1	1 1	1 1	1 1	1 1	1 1
8	- -	- -	- -	- -	1 -	1 -	- -	- -	1 1	1 -	- -	- -	1 1	1 -	- -	- -
9	- -	- -	- -	- -	- -	1 -	1 -	1 -	- -	1 1	1 -	1 1	- -	1 1	1 -	1 1
A	- -	- -	- -	- -	- -	- -	1 -	1 -	- -	- -	1 1	1 1	- -	- -	1 1	1 1
B	- -	- -	- -	- -	1 -	1 -	1 -	1 -	- -	- -	1 1	1 1	1 -	1 1	1 1	1 1
C	- -	- -	1 -	1 -	1 1	1 -	1 1	1 -	- -	- -	1 1	1 -	1 1	1 -	1 1	1 -
D	- -	- -	1 -	1 -	- -	1 1	1 -	1 1	- -	- -	1 -	1 1	- -	1 1	1 -	1 1
E	- -	1 -	1 1	1 1	- -	1 1	1 1	1 1	- -	1 1	1 1	1 1	- -	1 1	1 1	1 1
F	1 -	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1

Table 6: All possible transition patterns through L in $D1$ when considering blocks of four bytes (first element) and through L in $D2$ (second element)

To get also all allowed patterns for the linear function L in $D1$ it would be required to compute all 2^{128} combinations of an input word that is of size 128-bit. Now that this is a unconceivable task for standard computers in appropriate time by now, a simplification brings some results. We approximate differential properties of the linear function by considering blocks of 4 bytes length as instead the whole 16 bytes. In addition to this, it is also required to investigate the influence of the neighbouring bytes to this 4 byte block. Therefore, we not only loop through all combinations of this block, but also through all possible neighbouring bits that might influence the output of this 4 byte block. The linear function shifts one time to the left and also 3 times to the right, consequently, we have to consider also one more bit to the right and 3 more bits to the left. Therefore, we have to loop through 2^{36} possible combinations, because of the 4 additional bits, which do not influence the actual truncated difference pattern on the input, but might have an effect on the difference on the output. Subsequently these results can be concatenated together to give a 16 byte pattern. Note that this is only an approximation and allows also some of the impossible differentials. This requires to compute all combinations of a total of 36 bits, therefore a multithreaded program using POSIX threads was developed and executed on an 8 core computer. This still requires the double of the time needed for the computations of all patterns of L in $D2$.

For completeness, the table of allowed input to output patterns for L in an approximation of 4 bytes in $D1$ and also exactly in $D2$ is given in Table 6. Each cell contains two values, the first corresponding to L in $D1$ and the second for L in $D2$. To denote an impossible change of a pattern we use -, a possible transition is denoted by 1. As expected L in $D1$ has all ones of $D2$, but also some additional ones caused by a possible effect of bytes neighboring the considered four byte block. The input difference pattern gives the row index, the output difference pattern refers to the column index. Subsequently it is now conceivable to create a more detailed model based on these observations.

5.5 A More Sophisticated Model

Since we only defined differential activity based on words in the last model, we want to get more in detail and use a description of Artemia on byte level. Also the AES S-box is used and acts on bytes instead of those big words used in the first model. The basic idea is to model out the XOR operations used in the linear transformation and also the allowed transitions of active bytes in its linear function L. Also the conditions for the branch number of the linear transformations are included, therefore it is also required to combine active bytes to active words that are used in the inequalities for the branch condition.

This time let $x_{\ell,i}^{(r)}$ denote the differential activity of the i-th output byte of the linear transformation $D[\ell + 1]$ in round r. Again bytes are indexed from left to right without regards that they may belong to different blocks of linear transformations (As for example given by the 4 $D2$ blocks).

5.5.1 Connecting Active Bytes to Corresponding Active Words to Secure the Branch Condition

Let $t_i^{(r)}$ denote if the words corresponding to the inputs and outputs of diffusion layer $D1$ has any difference and subsequently have to meet the same inequalities as given in Section 5.3. Additionally the words $t_i^{(r)}$ have to be connected with the bytes describing themselves. To do so we introduce similar inequalities as those when changing the word

sizes in the simple model.

$$\text{input } D1\text{: } t_i^{(r)} \leq \sum_{c=0}^{15} x_{2,16i+c}^{(r-1)} \leq 16 t_i^{(r)}$$
$$\text{output } D1\text{: } t_{4+i}^{(r)} \leq \sum_{c=0}^{15} x_{0,16i+c}^{(r)} \leq 16 t_{4+i}^{(r)}$$
$$\mathcal{B}_D \cdot p_{0,0}^{(r)} \leq \sum_{j=0}^{7} t_j^{(r)} \leq 8 p_{0,0}^{(r)}$$
$$\forall i \in \{0,1,2,3\}$$

These inequalities secure the branch condition of $D1$, where $p_{\ell,j}^{(r)}$ is the dummy variable for the j-th block of $D([\ell+1]$ in round r. Afterwards consider the 4 $D2$ blocks. Let $g_{b,i}^{(r)}$ denote the input and output words for the b-th $D2$ block. The index i counts again from input to output. Similar inequalities are derived.

$$g_{b,i}^{(r)} \leq \sum_{c=0}^{3} x_{0,16b+4i+c}^{(r)} \leq 4 g_{b,i}^{(r)}$$
$$g_{b,4+i}^{(r)} \leq \sum_{c=0}^{3} x_{1,16b+4i+c}^{(r)} \leq 4 g_{b,i}^{(r)}$$
$$\mathcal{B}_D \cdot p_{1,b}^{(r)} \leq \sum_{j=0}^{7} g_{b,j}^{(r)} \leq 8 p_{1,b}^{(r)}$$
$$\forall b, i \in \{0,1,2,3\}$$

And lastly the common inequalities for $D3$, since this transformation has a word size of a byte and therefore nothing needs to be converted in its word size.

$$\mathcal{B}_D \cdot p_{2,b}^{(r)} \leq \sum_{i=0}^{3} x_{1,4b+i}^{(r)} + \sum_{i=0}^{3} x_{2,4b+i}^{(r)} \leq 8 p_{2,b}^{(r)}$$
$$\forall b \in \{0,1,...,15\}, \forall i \in \{0,1,2,3\}$$

All these inequalities serve as constraints so that the branch conditions are met.

5.5.2 Byte Level Modelling of XOR Operations in the Linear Transformations

Now this model is extended by inequalities which describe the actual XOR operations and linear functions that are utilized by each diffusion layer. As a first step consider the unrecursive definition equations for the diffusion layers. For example consider $Y_0 = X_0 \oplus X_2 \oplus X_3 \oplus L(X_1 \oplus X_3)$. The XOR is subsequently modelled on each byte of the word X_0. The part that gives more constraints to the model is the linear function L. As already mentioned in previous sections the differential activity can pass into adjacent bytes. Let $\mathcal{L}_{\ell,bl,by}^{(r)}(X)$ denote if the by-th byte of the output of the linear function of the bl-th diffusion block in layer ℓ in round r has a difference. Also $(\mathcal{L}^2)$ denotes the output

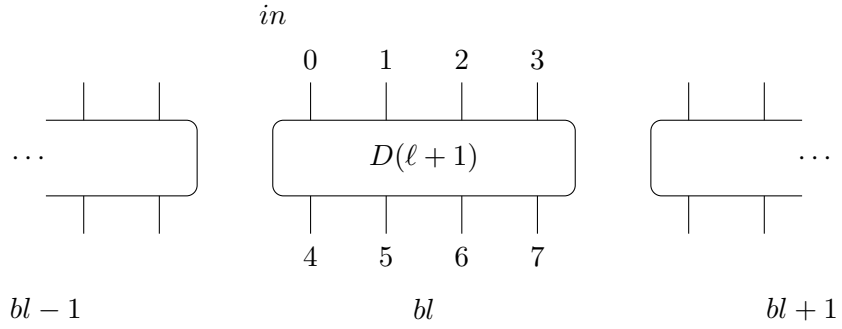


Figure 8: Indexing of the input and output around a diffusion block

differences of the twofold application of L. Starting with $D1$, the following inequalities characterizing the XORs in the linear transformation are obtained:

$$
\begin{aligned}
&2d_{0,i}^{(r)} \leq x_{0,i}^{(r)} + x_{2,i}^{(r-1)} + x_{2,32+i}^{(r-1)} + x_{2,48+i}^{(r-1)} + \mathcal{L}_{0,0,i}^{(r)}(X_1 \oplus X_3) \leq 5d_{0,i}^{(r)} \\
&2d_{0,16+i}^{(r)} \leq x_{0,16+i}^{(r)} + x_{2,i}^{(r-1)} + x_{2,16+i}^{(r-1)} + x_{2,32+i}^{(r-1)} + \mathcal{L}_{0,0,i}^{(r)}(X_0 \oplus X_1) \\
&\quad + (\mathcal{L}^2)_{0,0,i}^{(r)}(X_1 \oplus X_3) \leq 6d_{0,16+i}^{(r)} \\
&2d_{0,32+i}^{(r)} \leq x_{0,32+i}^{(r)} + x_{2,16+i}^{(r-1)} + x_{2,32+i}^{(r-1)} + x_{2,48+i}^{(r-1)} + \mathcal{L}_{0,0,i}^{(r)}(X_1 \oplus X_2) + (\mathcal{L}^2)_{0,0,i}^{(r)}(X_0 \oplus X_3) \\
&\quad + (\mathcal{L}^3)_{0,0,i}^{(r)}(X_1 \oplus X_2) \leq 7d_{0,32+i}^{(r)} \\
&2d_{0,48+i}^{(r)} \leq x_{0,48+i}^{(r)} + x_{2,i}^{(r-1)} + \mathcal{L}_{0,0,i}^{(r)}(X_1 \oplus X_2) + (\mathcal{L}^2)_{0,0,i}^{(r)}(X_0 \oplus X_1 \oplus X_2 \oplus X_3) \\
&\quad + (\mathcal{L}^3)_{0,0,i}^{(r)}(X_0 \oplus X_1) + (\mathcal{L}^4)_{0,0,i}^{(r)}(X_1 \oplus X_3) \leq 6d_{0,48+i}^{(r)} \\
&\forall i \in \{0, 1, ..., 15\}
\end{aligned}
$$

$D2$ is described in a similar way but obviously with different indices. For improved legibility, we define a function that gives the right difference variable when the round r, layer ℓ, block bl, input word in and byte by is known.

$$
db(r, \ell, bl, in, by) := \begin{cases} x_{2-2\lfloor in/4 \rfloor, 16(in \bmod 4)+by}^{(r-1+\lfloor in/4 \rfloor)} & \ell = 0 \\ x_{\ell-1+\lfloor in/4 \rfloor, 4^{3-\ell}bl+4^{2-\ell}(in \bmod 4)+by}^{(r)} & \ell = 1 \end{cases}
$$

The definition of this function helps to formulate the constraints more clearly. Note that the XOR constraints for the $D3$ blocks are not needed since the word size equals to one byte and are therefore modelled with the branch condition only. To summarize

we get inequalities describing all XOR operations in $D1$ and $D2$ as given below.

$$
\begin{aligned}
&2d^{(r)}_{\ell,4^{3-\ell}bl+by} \leq \sum_{j\in\{4,0,2,3\}} db(r,\ell,bl,j,by) + \mathcal{L}^{(r)}_{\ell,bl,by}(X_1 \oplus X_3) \leq 5d^{(r)}_{\ell,4^{3-\ell}bl+by} \\
&2d^{(r)}_{\ell,4^{3-\ell}bl+4^{2-\ell}+by} \leq \sum_{j\in\{5,0,1,2\}} db(r,\ell,bl,j,by) + \mathcal{L}^{(r)}_{\ell,bl,by}(X_0 \oplus X_1) \\
&\quad + (\mathcal{L}^2)^{(r)}_{\ell,bl,by}(X_1 \oplus X_3) \leq 6d^{(r)}_{\ell,4^{3-\ell}bl+4^{2-\ell}+by} \\
&2d^{(r)}_{\ell,4^{3-\ell}bl+2\cdot 4^{2-\ell}+by} \leq \sum_{j\in\{6,1,2,3\}} db(r,\ell,bl,j,by) + \mathcal{L}^{(r)}_{\ell,bl,by}(X_1 \oplus X_2) \\
&\quad + (\mathcal{L}^2)^{(r)}_{\ell,bl,by}(X_0 \oplus X_3) + (\mathcal{L}^3)^{(r)}_{\ell,bl,by}(X_1 \oplus X_3) \leq 7d^{(r)}_{\ell,4^{3-\ell}bl+2\cdot 4^{2-\ell}+by} \\
&2d^{(r)}_{\ell,4^{3-\ell}bl+3\cdot 4^{2-\ell}+by} \leq \sum_{j\in\{7,0\}} db(r,\ell,bl,j,by) + \mathcal{L}^{(r)}_{\ell,bl,by}(X_1 \oplus X_2) \\
&\quad + (\mathcal{L}^2)^{(r)}_{\ell,bl,by}(X_0 \oplus X_1 \oplus X_2 \oplus X_3) + (\mathcal{L}^3)^{(r)}_{\ell,bl,by}(X_0 \oplus X_1) \\
&\quad + (\mathcal{L}^4)^{(r)}_{\ell,bl,by}(X_1 \oplus X_3) \leq 6d^{(r)}_{\ell,4^{3-\ell}bl+3\cdot 4^{2-\ell}+by} \\
&\forall r \in R, \forall \ell \in \{0,1\}, \forall bl \in \{0,1,...,4^l-1\}, \forall by \in \{0,1,...,4^{2-l}-1\}
\end{aligned}
$$

5.5.3 Byte Level Description of the used Linear Function L

The last part of the model consists of formulating inequalities that describe the linear function L. The approach is given by the following ideas: First build constraints for the XOR operations, which gives the input to L, then we use the allowed pattern table (all possible differentials through L) introduced in the previous section to restrict impossible transitions of differential activity through L. To build up inequalities that describe this table a similar approach as given by Sun et al. [SHW+14] is used. The differentials through this linear function L are considered as cornerpoints of an 8 dimensional hypercube, with the first 4 coordinates given by the input byte differences and the last 4 coordinates given by the output byte differences. This subset of cornerpoints describes a convex set and therefore, this set can be described by inequalities that we want to use as constraints in the linear model.

Consequently it is required to convert this vertex representation (V-Representation) of the convex hull into a representation using inequalities (H-Representation). A convenient way to do this, is utilizing the built in class `Polyhedron` in the mathematical package `sage`. The allowed difference patterns serve as input and the class computes, though not needed because these points already describe a convex set, the convex hull represented, as required, by inequalities. Then the obtained inequalities connect the results of the inner XOR operations of L to the resulting byte differences of L restricting impossible truncated difference patterns on byte level.

Let a_i denote the i-th byte difference of the input to the linear function and ℓ_i the analogical output. The form of the given inequalities are mentioned below.

$$
\begin{aligned}
&s := (a_0, a_1, a_2, a_3, \ell_0, \ell_1, \ell_2, \ell_3) \quad \text{cornerpoint of a hypercube} \\
&\text{inequalities obtained by } \texttt{sage} \text{ are given by a vector } (u_0, u_1, ..., u_8) \\
&u_0 + \langle (u_1, u_2, ..., u_8), (a_0, a_1, a_2, a_3, l_0, l_1, l_2, l_3) \rangle \geq 0 \\
&\text{with } \langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \text{ the standard inner product}
\end{aligned}
$$

By implication every allowed difference transition pattern fulfills all given inequalities of the convex hull and not allowed patterns contradict at least one inequality. The complete convex hull may be expressed by a large amount of inequalities so that it is inadvisable to put them all in the linear model for the sake of computation time. As instead, inequalities that remove the most impossible difference transitions are picked and put into the model as in [SHW+14, Algorithm 1]. This comes in handy since there are 158 inequalities generated for all allowed difference transition patterns in $D2$ and 38 in the 4 byte approximated $D1$.

In the definition of the linear transformation expressed nonrecursive, there are certain XOR results, which serve as input to the linear function L, are required.

The needed results are: $(X_1 \oplus X_3), (X_0 \oplus X_1), (X_1 \oplus X_2), (X_0 \oplus X_3), (X_0 \oplus X_1 \oplus X_2 \oplus X_3)$. Therefore new variables that express the differential active state of these operations are introduced. Let $xo^{(r)}_{\ell,bl,by}(X_i \oplus X_j \oplus ... \oplus X_k)$ denote if the by-th byte of the operation $X_i \oplus X_j \oplus ... \oplus X_k$ in layer ℓ of block bl in round r has a difference. Expressed in terms of linear programming the required XOR results, which serve as input to the linear function L in the linear transformation D, are given beneath.

$$
\begin{aligned}
&2 \cdot dxo^{(r)}_{\ell,bl,by}(X_1 \oplus X_3) \leq xo^{(r)}_{\ell,bl,by}(X_1 \oplus X_3) + db(r,\ell,bl,1,by) \\
&\quad + db(r,\ell,bl,3,by) \leq 3 \cdot dxo^{(r)}_{\ell,bl,by}(X_1 \oplus X_3) \\
&2 \cdot dxo^{(r)}_{\ell,bl,by}(X_0 \oplus X_1) \leq xo^{(r)}_{\ell,bl,by}(X_0 \oplus X_1) + db(r,\ell,bl,0,by) \\
&\quad + db(r,\ell,bl,1,by) \leq 3 \cdot dxo^{(r)}_{\ell,bl,by}(X_0 \oplus X_1) \\
&2 \cdot dxo^{(r)}_{\ell,bl,by}(X_1 \oplus X_2) \leq xo^{(r)}_{\ell,bl,by}(X_1 \oplus X_2) + db(r,\ell,bl,1,by) \\
&\quad + db(r,\ell,bl,2,by) \leq 3 \cdot dxo^{(r)}_{\ell,bl,by}(X_1 \oplus X_2) \\
&2 \cdot dxo^{(r)}_{\ell,bl,by}(X_0 \oplus X_3) \leq xo^{(r)}_{\ell,bl,by}(X_0 \oplus X_3) + db(r,\ell,bl,0,by) \\
&\quad + db(r,\ell,bl,3,by) \leq 3 \cdot dxo^{(r)}_{\ell,bl,by}(X_0 \oplus X_3) \\
&2 \cdot dxo^{(r)}_{\ell,bl,by}(X_0 \oplus X_1 \oplus X_2 \oplus X_3) \leq xo^{(r)}_{\ell,bl,by}(X_0 \oplus X_1 \oplus X_2 \oplus X_3) \\
&\quad + db(r,\ell,bl,0,by) + db(r,\ell,bl,1,by) + db(r,\ell,bl,2,by) \\
&\quad + db(r,\ell,bl,3,by) \leq 3 \cdot dxo^{(r)}_{\ell,bl,by}(X_0 \oplus X_1 \oplus X_2 \oplus X_3) \\
&\forall r \in R, \forall \ell \in \{0,1\}, \forall bl \in \{0,1,...,4^{\ell}-1\}, \forall by \in \{0,1,...,4^{2-\ell}-1\}
\end{aligned}
$$

With dxo being a usual dummy variable.

5.5.4 Adding Inequalities of the Convex Hull to the Model

The linear model is then completed by setting up the constraints for the variables of $\mathcal{L}$. Additionally we define some abbreviations to give a short notation.

$$
\begin{aligned}
&v := (u_1, u_2, ..., u_8) \\
&vxo^{(r)}_{\ell,bl,i}(X) := (xo^{(r)}_{\ell,bl,4i}(X), xo^{(r)}_{\ell,bl,4i+1}(X), xo^{(r)}_{\ell,bl,4i+2}(X), xo^{(r)}_{\ell,bl,4i+3}(X)) \\
&vl^{(r)}_{\ell,bl,i}(X) := (\mathcal{L}^{(r)}_{\ell,bl,4i}(X), \mathcal{L}^{(r)}_{\ell,bl,4i+1}(X), \mathcal{L}^{(r)}_{\ell,bl,4i+2}(X), \mathcal{L}^{(r)}_{\ell,bl,4i+3}(X)) \\
&vco^{(r)}_{\ell,bl,i}(X) := (vxo^{(r)}_{\ell,bl,i}(X), vl^{(r)}_{\ell,bl,i}(X))
\end{aligned}
$$

Also let U denote the set of linear inequalities of the convex hull of the approximation of L in D1. In the same sense let V denote the set of the linear inequalities of the convex hull of L in D2. Finally E is a set that contains all required XOR operations. We therefore obtain the simple notation of linear inequalities describing the results of an application of L.

$$
\begin{aligned}
&u_0 + \langle v, vco^{(r)}_{l,bl,i}(op)\rangle \geq 0 \\
&\forall r \in R, \forall op \in E, \forall \ell \in \{0,1\}, \forall bl \in \{0,1,...,4^\ell - 1\}, \\
&\forall i \in \{0,1,...,4^{1-\ell} - 1\}, \forall v \in \begin{cases} U & \ell = 0 \\ V & \ell = 1 \end{cases}
\end{aligned}
$$

The very last inequalities for this model are then given by the two-, three- and fourfold applications of L. Consider tuples of the type (X, n) which hold a certain XOR operation in the first element X and in the second element n the number of times L will be applied on that result. Suppose M as the set of all tuples of L applicated on X required by the diffusion layers. We can analogically conclude several inequalities.

define the notation:

$$(vl^n)^{(r)}_{\ell,bl,i}(X) := ((\mathcal{L}^n)^{(r)}_{\ell,bl,4i}(X), (\mathcal{L}^n)^{(r)}_{\ell,bl,4i+1}(X), (\mathcal{L}^n)^{(r)}_{\ell,bl,4i+2}(X), (\mathcal{L}^n)^{(r)}_{\ell,bl,4i+3}(X))$$

to give:

$$
\begin{aligned}
&u_0 + \left\langle v, \left((vl^{n-1})^{(r)}_{\ell,bl,i}(op), (vl^n)^{(r)}_{\ell,bl,i}(op)\right)\right\rangle \geq 0 \\
&\forall r \in R, \forall (op, n) \in \{(o,i) | (o, i+j) \in M \wedge i, j \in \mathbb{N}\}, \forall \ell \in \{0,1\}, \\
&\forall bl \in \{0,1,...,4^\ell - 1\}, \forall i \in \{0,1,...,4^{1-\ell} - 1\}, \forall v \in \begin{cases} U & \ell = 0 \\ V & \ell = 1 \end{cases}
\end{aligned}
$$

All inequalities combined form the more detailed model on byte level.

5.6 Results of the more detailed Model

Subsequently the constraints of this model are implemented using `sage` to be solved by the `Gurobi` solver. We start by examining the results for one round of Artemia. `Gurobi` found the minimum number of active S-boxes to be **9**. Therefore we were not able to obtain a higher bound of minimum active S-boxes in one round, although the constraints are much more detailed. Consider Figure 9 for the actual trail. In our description based on bytes and the allowed difference transition patterns of L, given solution is plausible.

To further evaluate the solution, we consider the first $D2$ diffusion block. The input consists of 4 active bytes, which give $2^{32} - 15$ possible combinations in total. It is a possible task to loop through all these combinations to confirm this to be a possible truncated differential or prove it an impossible one. A short C program proves, that the truncated differential $(4888_{16}) \xrightarrow{D2} (4000_{16})$ is impossible. Although this trail is not achievable, this does not implicate that no other possible trail with 9 active S-boxes exists. Similar investigations could be done for the diffusion layer $D1$, but since there are 9 active bytes as input to this layer, it is not achievable to try all possible combinations in suitable time. However, it should also be noted, that this certain input pattern of the first $D2$ block has only a minimum branch number of 5 in only 40 cases out of all possible combinations.

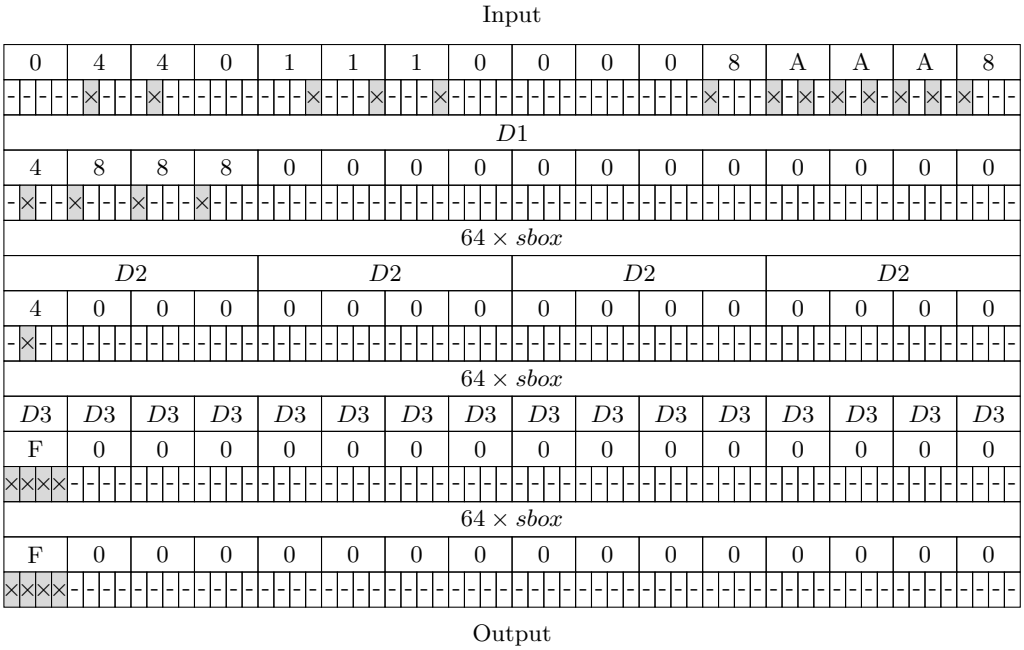


Figure 9: Active bytes in one round of the permutation Artemia

In conclusion, we also run the model for two rounds of the permutation to get the result of **45** minimum active S-boxes, identically to the simple model. The exact trail is given in Figure 10.

Given trail is also plausible, when we consider the byte model only.

On the other side, we again examine the differential of the first $D2$ block $(4442_{16}) \xrightarrow{D2} (0005_{16})$. As in the previous trail of one round, we try every possible bit combination on the active input bytes and analyze the output. It turns out, that there does not exist an input corresponding to this active bytes that cause active bytes on the output as given by this solution. Therefore, $(4442_{16}) \xrightarrow{D2} (0005_{16})$ is an impossible differential. Also, an interesting observation is, that this pattern of active inputs leads to a minimum of 5 active branches only in **15** cases out of $2^{32} - 15$ possible inputs, much fewer than in the previous differential.

The fact, that the more detailed model does not improve the bounds and also contains impossible differentials, motivates the design of further constraints. It would be conceivable to generate a table of all possible differentials through the diffusion layers with a similar approach as used in Section 5.4 and Section 5.5, where inequalities restricting impossible differentials are generated by introducing convex hulls to this problem. Since the inputs to the diffusion layers $D1$ and $D2$ are of a great size, 512 and 128-bit respectively, it is not conceivable to generate a table including all possible differentials required to generate inequalities restricting impossible differentials.

This motivates a further detailed model in a bitwise fashion investigated in Section 5.7.

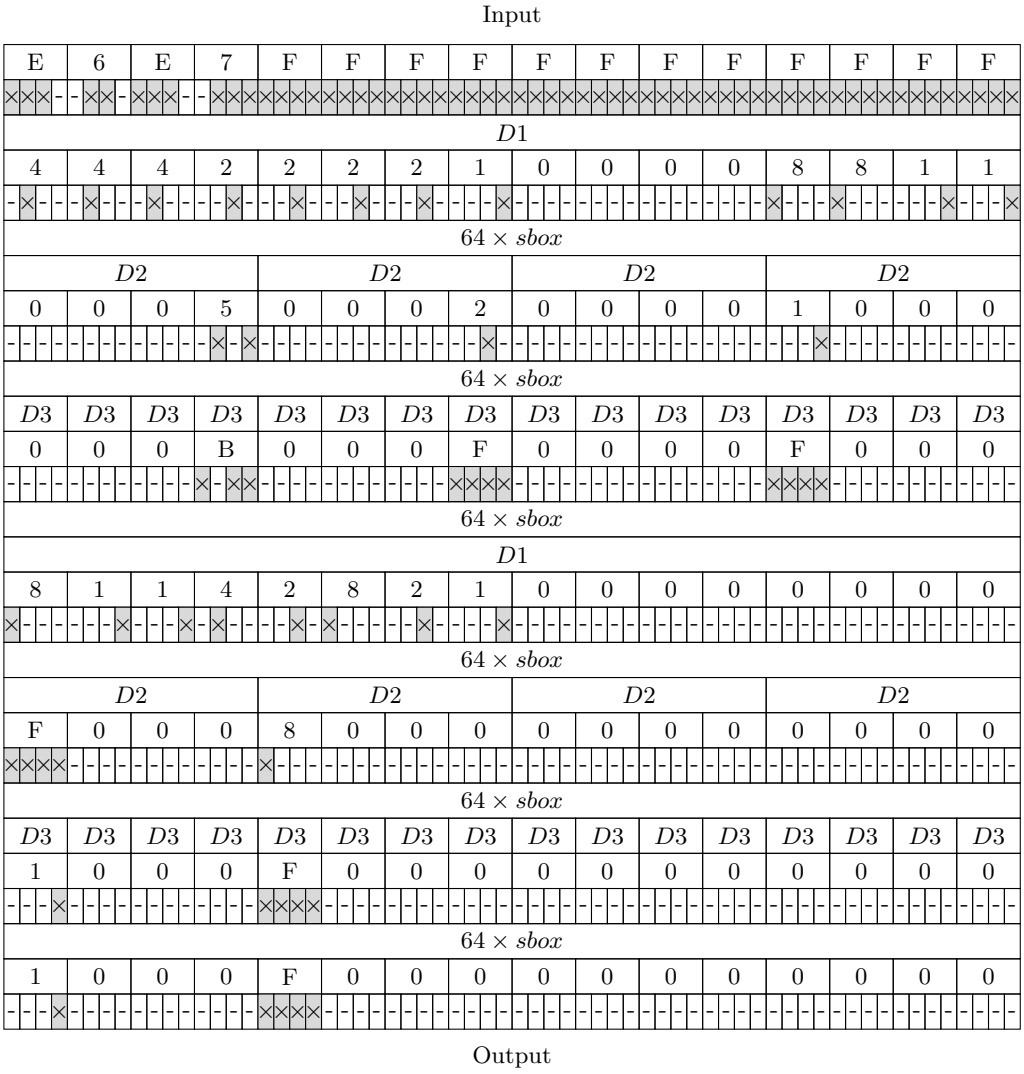


Figure 10: Active bytes in two rounds of the permutation Artemia

5.7 Binary Model

The most detailed approach is to model each of the linear transformations including the inner linear function exactly considering differences on bit level. Also we would expect that this exact representation will lead to a high bound for the amount of active S-boxes. As a result of the fine and accurate modelling, a model with a large amount of variables and constraints will be obtained.

The matrix representation of the linear diffusion layers seems to be a conventient representation to be utilized in the linear program. At this point, it is also necessary to discuss, how the differences in bit form are handled when an S-box is applied. Since Artemia uses the AES S-box, which is a non-linear mapping on strings of 8-bit length, the difference pattern in the bits can transit to every other as long as an active byte stays active. On the other hand, we know that there are plenty of impossible differentials through S-boxes. In our first approach we will allow every differential through the S-box, since the bitwise model already generates a great amount of variables and constraints.

We begin to define how the variables describing the difference of each bit are indexed. Let $x_{\ell,i}^{(r)}$ denote the difference of the i-th bit in the input of the layer ℓ, which consists of linear transformations $D(\ell+1)$, in round r and let $y_{\ell,i}^{(r)}$ denote the difference on the output in the same way. Suppose that M_s is again the matrix expressing the linear transformation with a total input of s bit length. The entry in the i-th row and j-th column is denoted by $(M_s)_{i,j}$. We obtain:

$$2d_{\ell,i}^{(r)} = y_{\ell,i}^{(r)} + \sum_{j=0}^{s-1} (M_s)_{i,j} \cdot x_{\ell,j}^{(r)}$$
$$\forall s \in \{512, 128, 32\}, \forall r \in R, \forall \ell \in \{0,1,2\}, \forall i \in \{0,1,...,511\}$$

d is again a dummy variable. Given equations already describe all diffusion layers. We now face to the description of the S-boxes. It is only necessary to connect the activity of bytes, without regard to the difference pattern of the bits.

$$da_{\ell,b}^{(r)} \leq \sum_{i=0}^{7} x_{\ell,8b+i}^{(r)} \leq 8da_{\ell,b}^{(r)}$$
$$da_{\ell,b}^{(r)} \leq \sum_{i=0}^{7} y_{\ell,8b+i}^{(r)} \leq 8da_{\ell,b}^{(r)}$$
$$\forall r \in R, \forall \ell \in \{0,1,2\}, \forall b \in \{0,1,...,63\}$$

Finally, we have to connect the output of a round to the input of the next round.

$$y_{2,i}^{(r-1)} = x_{0,i}^{(r)} \qquad \forall r \in R, \forall i \in \{0,1,...,511\}$$

Since we want to determine the minimum number of active S-boxes, the objective is given by the sum of all da, which expresses the active bytes in the concerning rounds.

This model for one round is then executed on a machine with 24 cores clocked at 3.47 GHz utilizing the `CPLEX` optimizer. No results could be obtained, as the linear program is far to big and therefore, infeasible to solve.

To improve performance, the model can be altered in the last layer. Since we allow every differential through the S-boxes, the binary modelling of $D3$ can be replaced by the branch condition. Although this measure shortens the linear program, the problem is still too big to be solved on this multicore machine for one round of the permutation.

A further approach is to not execute a linear program for a complete round, but instead one with a $D2$ block and its subsequent $D3$ blocks. Argumenting with the branch condition for $D1$, we therefore could conclude the minimum number of active S-boxes for 2 rounds, by 5 times the result of this smaller model. The same reasoning was also used in [JAB14]. To summarize: This model would consist of the inequalities describing one $D2$ layer in a binary exact way and 4 $D3$ blocks attached to it described by the branch condition. As an objective, the sum of active bytes in the input and output of $D2$ plus the active bytes in the output of $D3$ is used.

Executed on the same machine as above, we were able to obtain a lower bound of active S-boxes to be **9** after 10 days of computation. As a result, we cannot conclude better bounds for the amount of active S-boxes in 2 rounds of the permutation by this model.

As we still want to achieve a greater lower bound of active S-boxes, further ideas have to be pursued. The next conceivable approach on this problem, without trying to

run the complete model, is to describe $D3$ exactly in terms of the linear transformation matrix and at the same time restricting impossible differentials through the connecting S-boxes from $D2$ to $D3$. To get inequalities restricting impossible differentials, we again utilize `sage` and its class `Polyhedron`. This class would then compute the convex hull of all possible differentials, represented as strings of 16 bit length. No results could be obtained, as there is a large amount of possible differentials and the `Polyhedron` class is not parallelized. After 4 days, the computation was finally cancelled.

This class first computes the convex hull of the given points, but since we only have a subset of the cornerpoints of a hypercube, this is not necessary. These points already describe a convex set. The faster way therefore is to utilize an algorithm, that does not compute the convex hull of the points, but changes the vertex representation (V-representation) of this convex set to the representation with inequalities (H-representation). A program, which takes this task, is called `lrs` by Avis [Avi]. It takes the V-representation of a convex hull as input to transform it to the H-representation. In order to get all inequalities restricting impossible differentials, the program is executed. But since it is also not parallelized, we were not able to achieve the result in suitable time. However, there also exists a parallelized version of `lrs` that uses `MPI - Message Passing Interface` for communication: `mplrs`. It promises a nearly linear speedup and is perfectly applicable to problems of this character, the computation of inequalities that restrict impossible differentials through S-boxes. In this case, this does motivate the further investigation of active S-boxes on Artemia, but since the computation of active S-boxes on a single $D2$ layer with the $D3$ modelled with the branch condition already takes unconveniently long time, the time consumed by a model with all binary described, extended with inequalities restricting impossible differentials through the AES S-box, would be vast. However, it is conceivable that this not the case at some point in the future.

6 Conclusion

This thesis described the basic ideas of differential cryptanalysis as also Mouha et al.'s framework for generating a MILP model to find the best trail through a cipher. These methods were then applied to the Advanced Encryption Standard - AES, both using a static key model as well as in a related-key scenario. The results showed that a related-key model leads to a significant lower amount of active S-boxes and therefore, to a higher probable trail. Yet the amount of active S-boxes proves the ciphers resistance against differential cryptanalysis, even with related-keys.

Additionally, the permutation introduced by the CAESAR candidate Artemia was analyzed. We confirmed the bound of 45 active S-boxes over 2 rounds. Then, we investigated several refined models to establish the validity of these bounds for a byte-level and a bit-level model. We proposed different modelling approaches based on branch numbers, a matrix model of the linear function, and automatically approximated convex hulls. Unfortunately, we found, that the lower bound of S-boxes could not be improved when a more detailed model was used. Therefore, we could not reduce the required rounds for ensured safety against differential cryptanalysis. But since the obtained trails contain impossible differentials through the linear layer, it can be assumed, that the lower bound of S-boxes is greater than the results. Ways to find better bounds were introduced, in particular a binary model would likely give the best results, but is clearly infeasible. Also inequalities restricting impossible differentials could improve results. Methods to obtain these inequalities were presented. This binary model motivates future work as it gives as a result the exact trail through the cipher.

References

[All97] Michael Alley. *The Craft of Scientific Writing*. Springer Verlag, 3rd edition, June 1997.

[Avi] David Avis. lrs: A revised implementation of the reverse search vertex enumeration algorithm, may 1998.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In Menezes and Vanstone [MV91], pages 2–21.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[JAB14] Mohammad Reza Aref Javad Alizadeh and Nasour Bagheri. Artemia v1.1. 2014.

[KR11] Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011.

[MV91] Alfred Menezes and Scott A. Vanstone, editors. *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*. Springer, 1991.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In Wu et al. [WYL12], pages 57–76.

[Osw04] Elisabeth Oswald. Introduction to Information Security – Syllabus. Available online at `http://www.iaik.tugraz.at/content/teaching/`, October 2004.

[SDMS15] Mahdi Sajadieh, Mohammad Dakhilalian, Hamid Mala, and Pouyan Sepehrdad. Efficient recursive diffusion layers for block ciphers and hash functions. *J. Cryptology*, 28(2):240–256, 2015.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In Sarkar and Iwata [SI14], pages 158–178.

[SI14] Palash Sarkar and Tetsu Iwata, editors. *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*. Springer, 2014.

[WYL12] Chuankun Wu, Moti Yung, and Dongdai Lin, editors. *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*. Springer, 2012.