Generating S-Box Multivariate Quadratic Equation Systems And Estimating Algebraic Attack Resistance Aided By SageMath

A.-M. Leventi-Peetz[1] and J.-V. Peetz[2]

[1] Federal Office for Information Security,
Godesberger Allee 185–189, DE-53175 Bonn, Germany
anastasia-maria.leventi-peetz@bsi.bund.de

[2] DE-53343 Wachtberg, Germany
jvpeetz@web.de

June, 2015

Abstract

Methods are presented to derive with the aid of the computer mathematics software system SageMath the Multivariate Quadratic equation systems (MQ) for the input and output bit variables of a cryptographic S-box starting from its algebraic expressions. Motivation to this work were the results of recent articles [4, 5, 6] which we have verified and extended in an original way, to our knowledge, not yet published elsewhere. At the same time we present results contrary to the published ones which cast serious doubts on the suitability of previously presented formulas, supposed to quantify the resistance of S-boxes against algebraic attacks.

Key Words – Algebraic cryptanalysis, algebraic attack resistance, multivariate quadratic polynomial equation system, S-box, SageMath, polynomial quotient ring, SAT solver, Rijndael AES, Lagrange polynomial, algebraic expression.

1 Introduction

In this report we survey main results of the study of Jie Cui *et al.* [4] of the year 2014 in which the authors claim to have presented a new and concise

approach for generating the Multivariate Quadratic equation system (MQ) over $GF(2)$ according to the inverse transformation principle and the affine transformation of Rijndael S-box. Our contribution consists in the presentation of an independent way to automatically derive the same to those of Cui *et al.* as well as further implicit MQ for the S-box aided by the computer mathematics software system SageMath [1]. We investigate the estimation of the algebraic attack resistance and critically discuss the results stated by Cui *et al.*

SageMath (previously Sage or SAGE, System for Algebra and Geometry Experimentation[2]) is a free open-source software system for computer mathematics with features covering many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus.[2] It is licensed under the Gnu General Public License. It builds on top of many existing open-source packages: NumPy, SciPy, matplotlib, Sympy, PolyBoRi, Maxima, GAP, FLINT, R and many more. Their combined power is accessible through a common, Python-based language interface from the command line or a notebook in a web browser. Originally, it is designed by William Stein, still the leader of the SageMath project. He has also invented SageMathCloud [3] for collaborative computational mathematics in the cloud.

2 Principle of Rijndael S-box RD

We shall repeat the well known principles and algebraic properties [6] explored again in the study of Cui *et al.* [4]. Looking upon 8-bit bytes as elements in $GF(2^8)$, Rijndael's S-box is a mapping $S : GF(2^8) \longrightarrow GF(2^8)$ in form of a combination of an inverse function $I(\mathbf{x})$ which is a multivariate inverse modulo the irreducible polynomial $m(t) = t^8 + t^4 + t^3 + t + 1$ and an affine transformation function $A(\mathbf{x})$. $\mathbf{x}$ is a byte variable consisting of bits $x_i (i = 0, \ldots, 7)$, with x_7 symbolizing the most significant bit: $\mathbf{x} = \sum_{i=0}^{7} x_i t^i$. The modular inverse function $I(\mathbf{x})$ is defined as:

$$I(\mathbf{x}) = \mathbf{x}^{254} \bmod m(t) \tag{1}$$

i.e., the modular inverse of 0 is mapped to 0. According to the AES design[7], the affine transformation $A(\mathbf{x})$ can also be described as a modular polynomial multiplication followed by an addition (XOR) of a constant polynomial:

$$A(\mathbf{x}) = \mathbf{a}\,\mathbf{x} \bmod (t^8 + 1) + \mathbf{b} \tag{2}$$

with $\mathbf{a}$ = '1F' and $\mathbf{b}$ = '63'. A two-digit hexadecimal number stands for a *constant* byte, that is a polynomial in t, e.g, '63' for $t^6 + t^5 + t + 1$ or

in binary vector notation $[01100011]^T$. Equivalently, $A(\mathbf{x})$ can be written as matrix equation where the coefficients of the byte variables are regarded as vector components:

$$A(\mathbf{x}) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

The Rijndael S-box can be written as:

$$S_{\mathrm{RD}}(\mathbf{x}) = A \circ I = A(I(\mathbf{x}))$$

3 Cui's approach to generate an MQ for the Rijndael S-box

Cui *et al.* [4] (as Courtois and Pieprzyk [6] before them) utilize the Rijndael S-box composition of two functions to derive an MQ for it. With $\mathbf{x}$ the input and $\mathbf{z}$ the output value, and the intermediate variable $\mathbf{y} = I(\mathbf{x})$ they note: $\mathbf{z} = S_{\mathrm{RD}}(\mathbf{x}) = A(\mathbf{y}) = A(I(\mathbf{x}))$. Considering the inverse transformation $\mathbf{y} = I(\mathbf{x})$, obviously $\mathbf{xy} = 1$ when $\mathbf{x}$ not equal 0, which reads in polynomial form:

$$\left(\sum_{i=0}^{7} x_i t^i\right)\left(\sum_{j=0}^{7} x_j t^i\right) \bmod m(t) = \sum_{k=1}^{7} 0 \cdot t^k + 1 \tag{3}$$

The above modulo division is then analytically performed and a comparison of coefficients of terms of the same order in t^k, $0 \leq k \leq 7$, leads to the first eight multivariate quadratic equations for Rijndael S-box in the pages 2483 and 2484 of the paper of Cui *et al.* [4] The authors give all the steps and in-between results of the complete length of the calculation. They formulate and evaluate two additional equations of the byte variables to define the S-box completely. Doing so, Cui *et al.* replicate results already presented in 2002 by Courtois and Pieprzyk in the extended version of [6].

We will model and evaluate the same equations in the next section with the open source computer mathematics tool Sage.

4 Rijndael S-box coded in Sage

In order to work with polynomials like $\mathbf{x} = \sum_{i=0}^{7} x_i t^i$ Sage provides modules to construct rings of multivariate polynomials. In the previous section the polynomials **x**, **y**, and **z** were introduced. In line 1 of listing 1 a variable for the number of bits is defined for convenience. In line 2 a list of strings for the names of the coefficients of the three byte polynomials is generated (`['x0', 'x1', .., 'z7']`).

Listing 1: Byte polynomials over a quotient ring

```
1   nb = 8
2   varl = [c + str(p) for c in 'xyz' for p in range(nb)]
3   B = BooleanPolynomialRing(names = varl)
4   B.inject_variables()
5   P.<p> = PolynomialRing(B)
6   Byte.<t> = P.quotient_ring(p^8 + p^4 + p^3 + p + 1)
7   X = B.gens()[:nb]
8   Y = B.gens()[nb:2*nb]
9   x = sum([X[j]*t^j for j in range(nb)])
10  y = sum([Y[j]*t^j for j in range(nb)])
```

In line 3 a Boolean polynomial ring for these coefficients is constructed which assigns $GF(2)$ properties to them. In line 4 the coefficient names are made available as variables. In line 5 a polynomial ring over the Boolean polynomial ring `B` is constructed and from that, in line 6, the final quotient ring `Byte` with modulus $m(t)$. In lines 7 and 8, lists of coefficient variables of the byte polynomials are created for convenience. With the help of these lists, in lines 9 and 10, finally, the polynomials **x** and **y** are modeled.

Now one can already evaluate the product **xy** in Sage with the commands:

```
E3 = x * y
eqs3 = E3.list()
latex(eqs3)
```

In the second line we used the `list()` attribute to get the coefficients of each power of t in equation `E3`. With the last command `latex(eqs3)` one gets a list of the equations printed in a form ready to copy-and-paste directly into a LaTeX document. Due to the usage of the quotient ring, `E3` is of degree 7, the length of list `eqs3` (the number of coefficients) is 8. With Sage we have gotten the following equations which compare with the system of equations with number (3) in the paper of Cui *et al.* [4]:

$$\begin{aligned} c_0 &= x_0y_0 + x_1y_7 + x_2y_6 + x_3y_5 + x_4y_4 + x_5y_3 + x_5y_7 + x_6y_2 \\ &\quad + x_6y_6 + x_6y_7 + x_7y_1 + x_7y_5 + x_7y_6 \end{aligned}$$

$$\begin{aligned} c_1 &= x_0y_1 + x_1y_0 + x_1y_7 + x_2y_6 + x_2y_7 + x_3y_5 + x_3y_6 + x_4y_4 \\ &\quad + x_4y_5 + x_5y_3 + x_5y_4 + x_5y_7 + x_6y_2 + x_6y_3 + x_6y_6 + x_7y_1 \\ &\quad + x_7y_2 + x_7y_5 + x_7y_7 \end{aligned}$$

$$
\begin{aligned}
c_2 &= x_0y_2 + x_1y_1 + x_2y_0 + x_2y_7 + x_3y_6 + x_3y_7 + x_4y_5 + x_4y_6 \\
&\quad + x_5y_4 + x_5y_5 + x_6y_3 + x_6y_4 + x_6y_7 + x_7y_2 + x_7y_3 + x_7y_6 \\
c_3 &= x_0y_3 + x_1y_2 + x_1y_7 + x_2y_1 + x_2y_6 + x_3y_0 + x_3y_5 + x_3y_7 \\
&\quad + x_4y_4 + x_4y_6 + x_4y_7 + x_5y_3 + x_5y_5 + x_5y_6 + x_5y_7 + x_6y_2 \\
&\quad + x_6y_4 + x_6y_5 + x_6y_6 + x_6y_7 + x_7y_1 + x_7y_3 + x_7y_4 + x_7y_5 \\
&\quad + x_7y_6 + x_7y_7 \\
c_4 &= x_0y_4 + x_1y_3 + x_1y_7 + x_2y_2 + x_2y_6 + x_2y_7 + x_3y_1 + x_3y_5 \\
&\quad + x_3y_6 + x_4y_0 + x_4y_4 + x_4y_5 + x_4y_7 + x_5y_3 + x_5y_4 + x_5y_6 \\
&\quad + x_6y_2 + x_6y_3 + x_6y_5 + x_7y_1 + x_7y_2 + x_7y_4 + x_7y_7 \\
c_5 &= x_0y_5 + x_1y_4 + x_2y_3 + x_2y_7 + x_3y_2 + x_3y_6 + x_3y_7 + x_4y_1 \\
&\quad + x_4y_5 + x_4y_6 + x_5y_0 + x_5y_4 + x_5y_5 + x_5y_7 + x_6y_3 + x_6y_4 \\
&\quad + x_6y_6 + x_7y_2 + x_7y_3 + x_7y_5 \\
c_6 &= x_0y_6 + x_1y_5 + x_2y_4 + x_3y_3 + x_3y_7 + x_4y_2 + x_4y_6 + x_4y_7 \\
&\quad + x_5y_1 + x_5y_5 + x_5y_6 + x_6y_0 + x_6y_4 + x_6y_5 + x_6y_7 + x_7y_3 \\
&\quad + x_7y_4 + x_7y_6 \\
c_7 &= x_0y_7 + x_1y_6 + x_2y_5 + x_3y_4 + x_4y_3 + x_4y_7 + x_5y_2 + x_5y_6 \\
&\quad + x_5y_7 + x_6y_1 + x_6y_5 + x_6y_6 + x_7y_0 + x_7y_4 + x_7y_5 + x_7y_7
\end{aligned} \tag{4}
$$

These equations are practically identical to equations (3) of the paper of Cui *et al.*, with the only difference being in the order of summands within each equation. The equations (4) are terms of the Sage *object* list `eqs3`. In the enumeration the zeroth term of the list (`eqs3[0]`) is the constant term of the polynomial, the first (`eqs3[1]`) is the coefficient of the term of first order and so on.

Cui *et al.* proceeded with the generation of the next set of equations for Rijndael's S-box, the affine transformation. From equation (2) setting $\mathbf{z} = A(\mathbf{y})$ it follows

$$
\mathbf{y} = \mathbf{a}^{255}(\mathbf{z} + \mathbf{b}) \bmod (t^8 + 1) \tag{5}
$$

since $\mathbf{a}^{255}\,\mathbf{a} \bmod (t^8+1) = 1$. Written in the bit variables equation (5) translates to:

$$\begin{cases} y_7 &= z_6 + z_4 + z_1 \\ y_6 &= z_5 + z_3 + z_0 \\ y_5 &= z_7 + z_4 + z_2 \\ y_4 &= z_6 + z_3 + z_1 \\ y_3 &= z_5 + z_2 + z_0 \\ y_2 &= z_7 + z_4 + z_1 + 1 \\ y_1 &= z_6 + z_3 + z_0 \\ y_0 &= z_7 + z_5 + z_2 + 1 \end{cases} \tag{6}$$

We can substitute for $\mathbf{y}$ equations (6) into (4) to get the final form of the first eight equations representing Rijndael S-box. In Sage the values substitution is accomplished with the help of a so called *dictionary*. By using equation (5) the code in Listing 2 generates this dictionary, called `eqs4`.

Listing 2: Generate a dictionary to substitute $\mathbf{y}$ variables

```
Baff.<u> = P.quotient_ring( p^8 + 1 )
Z = B.gens()[2*nb:][:nb]
z = sum([Z[j]*u^j for j in range(nb)])
a = u^4 + u^3 + u^2 + u + 1
b = u^6 + u^5 + u + 1
eqs4 = dict(zip(Y, (a^255 * (z + b)).list()))
```

The first line sets up a quotient ring modulo $t^8 + 1$. The next four lines define the byte variable $\mathbf{z}$ and the two constant polynomials $\mathbf{a}$ and $\mathbf{b}$ in this ring (with generator `u`). Equation (5) simply reads `a^255 * (z + b)` in the code. The dictionary is constructed in the last line. The result is shown in Listing 3.

Listing 3: Dictionary to substitute $\mathbf{y}$ variables

```
{y7: z6 + z4 + z1,
 y6: z5 + z3 + z0,
 y5: z7 + z4 + z2,
 y4: z6 + z3 + z1,
 y3: z5 + z2 + z0,
 y2: z7 + z4 + z1 + 1,
 y1: z6 + z3 + z0,
 y0: z7 + z5 + z2 + 1}
```

The substitution of $\mathbf{y}$ in equation (4) via the dictionary succeeds with the following:

```
eqs5 = [_.substitute(eqs4) for _ in eqs3]
```

The result is again a list, the members of which give the first set of eight multivariate quadratic equations of the S-box by setting the byte variable product equal to 1. They read:

$$1 \;=\; x_0z_2 + x_0z_5 + x_0z_7 + x_0 + x_1z_1 + x_1z_4 + x_1z_6 + x_2z_0$$

$$
\begin{aligned}
& +x_2z_3+x_2z_5+x_3z_2+x_3z_4+x_3z_7+x_4z_1+x_4z_3+x_4z_6 \\
& +x_5z_0+x_5z_1+x_5z_2+x_5z_4+x_5z_5+x_5z_6+x_6z_0+x_6z_3 \\
& +x_6z_5+x_6z_6+x_6z_7+x_6+x_7z_2+x_7z_4+x_7z_5+x_7z_6+x_7z_7 \\
0 \;=\; & x_0z_0+x_0z_3+x_0z_6+x_1z_1+x_1z_2+x_1z_4+x_1z_5+x_1z_6 \\
& +x_1z_7+x_1+x_2z_0+x_2z_1+x_2z_3+x_2z_4+x_2z_5+x_2z_6 \\
& +x_3z_0+x_3z_2+x_3z_3+x_3z_4+x_3z_5+x_3z_7+x_4z_1+x_4z_2 \\
& +x_4z_3+x_4z_4+x_4z_6+x_4z_7+x_5z_0+x_5z_2+x_5z_3+x_5z_4 \\
& +x_5z_5+x_6z_1+x_6z_2+x_6z_3+x_6z_4+x_6z_7+x_6+x_7z_0 \\
& +x_7z_2+x_7z_3+x_7z_4+x_7 \\
0 \;=\; & x_0z_1+x_0z_4+x_0z_7+x_0+x_1z_0+x_1z_3+x_1z_6+x_2z_1 \\
& +x_2z_2+x_2z_4+x_2z_5+x_2z_6+x_2z_7+x_2+x_3z_0+x_3z_1 \\
& +x_3z_3+x_3z_4+x_3z_5+x_3z_6+x_4z_0+x_4z_2+x_4z_3+x_4z_4 \\
& +x_4z_5+x_4z_7+x_5z_1+x_5z_2+x_5z_3+x_5z_4+x_5z_6+x_5z_7 \\
& +x_6z_0+x_6z_2+x_6z_3+x_6z_4+x_6z_5+x_7z_1+x_7z_2+x_7z_3 \\
& +x_7z_4+x_7z_7+x_7 \\
0 \;=\; & x_0z_0+x_0z_2+x_0z_5+x_1z_6+x_1z_7+x_1+x_2z_5+x_2z_6 \\
& +x_3z_1+x_3z_5+x_3z_6+x_3+x_4z_0+x_4z_4+x_4z_5+x_5z_1 \\
& +x_5z_3+x_5z_6+x_5z_7+x_6z_0+x_6z_1+x_6z_2+x_6z_4+x_6z_5 \\
& +x_6+x_7z_0+x_7z_3+x_7z_6+x_7z_7 \\
0 \;=\; & x_0z_1+x_0z_3+x_0z_6+x_1z_0+x_1z_1+x_1z_2+x_1z_4+x_1z_5 \\
& +x_1z_6+x_2z_0+x_2z_3+x_2z_5+x_2z_6+x_2z_7+x_2+x_3z_2 \\
& +x_3z_4+x_3z_5+x_3z_6+x_3z_7+x_4z_3+x_4z_5+x_4+x_5z_1 \\
& +x_5z_2+x_5z_6+x_6z_0+x_6z_1+x_6z_5+x_6+x_7z_0+x_7z_1 \\
& +x_7z_6+x_7z_7+x_7 \\
0 \;=\; & x_0z_2+x_0z_4+x_0z_7+x_1z_1+x_1z_3+x_1z_6+x_2z_0+x_2z_1 \\
& +x_2z_2+x_2z_4+x_2z_5+x_2z_6+x_3z_0+x_3z_3+x_3z_5+x_3z_6 \\
& +x_3z_7+x_3+x_4z_2+x_4z_4+x_4z_5+x_4z_6+x_4z_7+x_5z_3 \\
& +x_5z_5+x_5+x_6z_1+x_6z_2+x_6z_6+x_7z_0+x_7z_1+x_7z_5+x_7 \\
0 \;=\; & x_0z_0+x_0z_3+x_0z_5+x_1z_2+x_1z_4+x_1z_7+x_2z_1+x_2z_3 \\
& +x_2z_6+x_3z_0+x_3z_1+x_3z_2+x_3z_4+x_3z_5+x_3z_6+x_4z_0 \\
& +x_4z_3+x_4z_5+x_4z_6+x_4z_7+x_4+x_5z_2+x_5z_4+x_5z_5 \\
& +x_5z_6+x_5z_7+x_6z_3+x_6z_5+x_6+x_7z_1+x_7z_2+x_7z_6
\end{aligned}
$$

$$
\begin{aligned}
0 &= x_0z_1 + x_0z_4 + x_0z_6 + x_1z_0 + x_1z_3 + x_1z_5 + x_2z_2 + x_2z_4 \\
&\quad + x_2z_7 + x_3z_1 + x_3z_3 + x_3z_6 + x_4z_0 + x_4z_1 + x_4z_2 + x_4z_4 \\
&\quad + x_4z_5 + x_4z_6 + x_5z_0 + x_5z_3 + x_5z_5 + x_5z_6 + x_5z_7 + x_5 \\
&\quad + x_6z_2 + x_6z_4 + x_6z_5 + x_6z_6 + x_6z_7 + x_7z_3 + x_7z_5 + x_7
\end{aligned} \tag{7}
$$

The last system of equations (7) is identical to the system (5) of Cui *et al.* Those equations with zero constant term (7 out of 8 above) are true with probability equal to 1. The 8th equation is true only when $\mathbf{x} \neq 0$, so that this equation is true with a probability 255/256. Furthermore for $\forall\, \mathbf{x} \neq 0$ $\mathbf{x} = \mathbf{x}^2\mathbf{y}$. Obviously this last equation is true also when $\mathbf{x} = 0$, so that one can write:

$$
\forall\, \mathbf{x} \in GF(2^8) \begin{cases} \mathbf{x} &= \mathbf{y}\mathbf{x}^2 \\ \mathbf{x}^2 &= \mathbf{y}^2\mathbf{x}^4 \\ & \vdots \\ \mathbf{x}^{128} &= \mathbf{y}^{128}\mathbf{x}^{256} = \mathbf{y}^{128}\mathbf{x} \end{cases} \tag{8}
$$

Cui *et al.* take the last of the above and write two symmetrical equations to generate an additional set of 16 equations for the Rijndael S-box. The equations they take are:

$$
\begin{cases} \mathbf{x}^{128} &= \mathbf{y}^{128}\mathbf{x} \\ \mathbf{y}^{128} &= \mathbf{x}^{128}\mathbf{y} \end{cases} \tag{9}
$$

We develop the two last equations to get the needed additional 16 equations for the implicated variables. We also substitute in these equations $\mathbf{y}$ by using (6) in form of the dictionary in listing 3.

```
E7 = x^128 + y^128 * x
eqs7 = [_.substitute(eqs4) for _ in E7.list()]
```

The result is a list with the following equations, which are practically the equations (7) of Cui *et al.* written in the reverse order than that of Cui's paper. Cui *et al.* begin with the expression corresponding to the highest order term of $E7$ while the sage list begins with the constant term.

$$
\begin{aligned}
0 &= x_0z_0 + x_0z_1 + x_0z_2 + x_0z_6 + x_1z_1 + x_1z_3 + x_1z_5 + x_1z_7 \\
&\quad + x_2z_3 + x_2z_4 + x_2z_5 + x_2z_6 + x_2z_7 + x_3z_2 + x_3z_3 + x_3z_5 \\
&\quad + x_3z_6 + x_3 + x_4z_0 + x_4z_1 + x_4z_3 + x_4z_4 + x_5z_0 + x_5z_1 \\
&\quad + x_5z_2 + x_5z_3 + x_5z_5 + x_5z_6 + x_5z_7 + x_5 + x_6z_0 + x_6z_3 \\
&\quad + x_6z_5 + x_6z_7 + x_7z_1 + x_7z_3 + x_7z_5 + x_7z_6 \\
0 &= x_0z_1 + x_0z_2 + x_0z_3 + x_0z_4 + x_0z_5 + x_0z_6 + x_0z_7 + x_0 \\
&\quad + x_1z_0 + x_1z_2 + x_1z_3 + x_1z_5 + x_1z_6 + x_1z_7 + x_2z_1 + x_2z_4
\end{aligned}
$$

$$
\begin{aligned}
& +x_2z_6 + x_2 + x_3z_2 + x_3z_4 + x_3z_7 + x_3 + x_4z_0 + x_4z_1 \\
& +x_4z_2 + x_4z_4 + x_4z_5 + x_4z_6 + x_5z_2 + x_5z_4 + x_5z_5 + x_5z_6 \\
& +x_5z_7 + x_6z_1 + x_6z_2 + x_6z_6 + x_7z_0 + x_7z_1 + x_7z_6 + x_7z_7 + x_7 \\
0 \;=\; & x_0z_0 + x_0z_1 + x_0z_3 + x_0z_4 + x_0z_5 + x_0z_6 + x_0z_7 + x_1z_1 \\
& +x_1z_2 + x_1z_3 + x_1z_4 + x_1z_5 + x_1z_6 + x_1z_7 + x_1 + x_2z_0 \\
& +x_2z_2 + x_2z_3 + x_2z_5 + x_2z_6 + x_2z_7 + x_2 + x_3z_1 + x_3z_4 \\
& +x_3z_6 + x_3 + x_4z_2 + x_4z_4 + x_4z_7 + x_4 + x_5z_0 + x_5z_1 \\
& +x_5z_2 + x_5z_4 + x_5z_5 + x_5z_6 + x_5 + x_6z_2 + x_6z_4 + x_6z_5 \\
& +x_6z_6 + x_6z_7 + x_7z_1 + x_7z_2 + x_7z_6 \\
0 \;=\; & x_0z_0 + x_0z_2 + x_0z_6 + x_1z_0 + x_1z_4 + x_1z_6 + x_1 + x_2z_1 \\
& +x_2z_2 + x_2 + x_3z_0 + x_3z_7 + x_4z_0 + x_4z_3 + x_4z_6 + x_5z_0 \\
& +x_5z_1 + x_5z_3 + x_5z_4 + x_5z_5 + x_5z_6 + x_6z_1 + x_6z_2 + x_6z_3 \\
& +x_6z_4 + x_6z_6 + x_6z_7 + x_6 + x_7z_1 + x_7z_2 + x_7z_3 + x_7z_4 \\
& +x_7z_7 + x_7 \\
0 \;=\; & x_0z_0 + x_0z_1 + x_0z_3 + x_0z_4 + x_1z_0 + x_1z_1 + x_1z_2 + x_1z_3 \\
& +x_1z_5 + x_1z_6 + x_1z_7 + x_1 + x_2z_0 + x_2z_3 + x_2z_5 + x_2z_7 \\
& +x_3z_1 + x_3z_3 + x_3z_5 + x_3z_6 + x_3 + x_4z_1 + x_4z_3 + x_4z_4 \\
& +x_4z_7 + x_4 + x_5z_1 + x_5z_2 + x_5z_5 + x_5z_7 + x_6z_1 + x_6z_4 \\
& +x_6z_6 + x_6z_7 + x_7z_2 + x_7z_4 + x_7z_5 + x_7z_7 \\
0 \;=\; & x_0z_2 + x_0z_3 + x_0z_5 + x_0z_6 + x_1z_0 + x_1z_1 + x_1z_3 + x_1z_4 \\
& +x_1 + x_2z_0 + x_2z_1 + x_2z_2 + x_2z_3 + x_2z_5 + x_2z_6 + x_2z_7 \\
& +x_3z_0 + x_3z_3 + x_3z_5 + x_3z_7 + x_3 + x_4z_1 + x_4z_3 + x_4z_5 \\
& +x_4z_6 + x_4 + x_5z_1 + x_5z_3 + x_5z_4 + x_5z_7 + x_5 + x_6z_1 \\
& +x_6z_2 + x_6z_5 + x_6z_7 + x_7z_1 + x_7z_4 + x_7z_6 + x_7z_7 \\
0 \;=\; & x_0z_3 + x_0z_4 + x_0z_5 + x_0z_6 + x_0z_7 + x_1z_2 + x_1z_3 + x_1z_5 \\
& +x_1z_6 + x_1 + x_2z_0 + x_2z_1 + x_2z_3 + x_2z_4 + x_3z_0 + x_3z_1 \\
& +x_3z_2 + x_3z_3 + x_3z_5 + x_3z_6 + x_3z_7 + x_3 + x_4z_0 + x_4z_3 \\
& +x_4z_5 + x_4z_7 + x_5z_1 + x_5z_3 + x_5z_5 + x_5z_6 + x_6z_1 + x_6z_3 \\
& +x_6z_4 + x_6z_7 + x_6 + x_7z_1 + x_7z_2 + x_7z_5 + x_7z_7 \\
0 \;=\; & x_0z_1 + x_0z_3 + x_0z_5 + x_0z_7 + x_1z_3 + x_1z_4 + x_1z_5 + x_1z_6 \\
& +x_1z_7 + x_1 + x_2z_2 + x_2z_3 + x_2z_5 + x_2z_6 + x_3z_0 + x_3z_1 \\
& +x_3z_3 + x_3z_4 + x_3 + x_4z_0 + x_4z_1 + x_4z_2 + x_4z_3 + x_4z_5
\end{aligned}
$$

$$
\begin{aligned}
&+x_4z_6 + x_4z_7 + x_5z_0 + x_5z_3 + x_5z_5 + x_5z_7 + x_5 + x_6z_1 \\
&+x_6z_3 + x_6z_5 + x_6z_6 + x_6 + x_7z_1 + x_7z_3 + x_7z_4 + x_7z_7 \qquad (10)
\end{aligned}
$$

The above equations (10) are identical with those in (7) of Cui *et al.* [4] with a couple of minimal differences which we attribute to typographical errors in the paper of Cui *et al.* These are the following: In both the second and third from the top equations (7) in the paper of Cui *et al.* there is missing one term each time: These are the terms: x_3z_7 and x_2z_3 respectively. In the seventh equation from the top of (7) there is a superfluous term: x_6.

Similarly, we write:

```
E8 = y^128 + x^128 * y
eqs8 = [_.substitute(eqs4) for _ in E8.list()]
```

and get the next block of eight equations for the Rijndael S-box. These are to compare with the system (8) of Cui *et al.*

$$
\begin{aligned}
0 = {} & x_0z_2 + x_0z_5 + x_0z_7 + x_0 + x_1z_1 + x_1z_2 + x_1z_5 + x_1z_6 \\
& +x_1z_7 + x_1 + x_2z_1 + x_2z_4 + x_2z_6 + x_3z_0 + x_3z_5 + x_4z_0 \\
& +x_4z_3 + x_4z_5 + x_5z_4 + x_5z_7 + x_6z_2 + x_6z_4 + x_6z_7 + x_7z_1 \\
& +x_7z_3 + x_7z_4 + x_7 + z_0 + z_1 + z_2 + z_6 + 1 \\
0 = {} & x_0z_0 + x_0z_3 + x_0z_6 + x_1z_0 + x_1z_1 + x_1z_3 + x_1z_5 + x_1 \\
& +x_2z_1 + x_2z_2 + x_2z_4 + x_2z_5 + x_2z_6 + x_2z_7 + x_2 + x_3z_0 \\
& +x_3z_1 + x_3z_2 + x_3z_6 + x_3z_7 + x_3 + x_4z_0 + x_4z_1 + x_4z_3 \\
& +x_4z_4 + x_4z_5 + x_4z_6 + x_5z_0 + x_5z_4 + x_5z_5 + x_5z_7 + x_6z_0 \\
& +x_6z_2 + x_6z_3 + x_6z_4 + x_6z_5 + x_6z_7 + x_7z_1 + x_7z_3 + x_7z_7 \\
& +x_7 + z_1 + z_2 + z_3 + z_4 + z_5 + z_6 + z_7 + 1 \\
0 = {} & x_0z_1 + x_0z_4 + x_0z_7 + x_0 + x_1z_1 + x_1z_4 + x_1z_5 + x_1z_6 \\
& +x_1z_7 + x_1 + x_2z_0 + x_2z_3 + x_2z_6 + x_3z_0 + x_3z_1 + x_3z_3 \\
& +x_3z_5 + x_3 + x_4z_1 + x_4z_2 + x_4z_4 + x_4z_5 + x_4z_6 + x_4z_7 \\
& +x_4 + x_5z_0 + x_5z_1 + x_5z_2 + x_5z_6 + x_5z_7 + x_5 + x_6z_0 \\
& +x_6z_1 + x_6z_3 + x_6z_4 + x_6z_5 + x_6z_6 + x_7z_0 + x_7z_4 + x_7z_5 \\
& +x_7z_7 + z_0 + z_1 + z_3 + z_4 + z_5 + z_6 + z_7 \\
0 = {} & x_0z_0 + x_0z_2 + x_0z_5 + x_1z_0 + x_1z_1 + x_1z_2 + x_1z_5 + x_1z_7 \\
& +x_1 + x_2z_6 + x_2z_7 + x_2 + x_3z_0 + x_3z_1 + x_3z_4 + x_3z_6 \\
& +x_3z_7 + x_3 + x_4z_5 + x_4z_6 + x_5z_0 + x_5z_1 + x_5z_3 + x_5z_4 \\
& +x_5z_5 + x_5z_7 + x_5 + x_6z_1 + x_6z_5 + x_6z_6 + x_6 + x_7z_0 \\
& +x_7z_2 + x_7z_3 + x_7z_4 + x_7z_6 + x_7z_7 + z_0 + z_2 + z_6
\end{aligned}
$$

$$
\begin{aligned}
0 &= x_0z_1 + x_0z_3 + x_0z_6 + x_1z_0 + x_1z_2 + x_1z_3 + x_1 + x_2z_0 \\
&\quad + x_2z_1 + x_2z_2 + x_2z_4 + x_2z_5 + x_2z_6 + x_3z_1 + x_3z_2 + x_3z_7 \\
&\quad + x_3 + x_4z_0 + x_4z_3 + x_4z_5 + x_4z_6 + x_4z_7 + x_4 + x_5z_0 \\
&\quad + x_5z_1 + x_5z_6 + x_5 + x_6z_2 + x_6z_4 + x_6z_5 + x_6z_6 + x_6z_7 \\
&\quad + x_7z_0 + x_7z_5 + x_7z_7 + z_0 + z_1 + z_3 + z_4 \\
0 &= x_0z_2 + x_0z_4 + x_0z_7 + x_1z_1 + x_1z_3 + x_1z_4 + x_1 + x_2z_1 \\
&\quad + x_2z_3 + x_2z_6 + x_3z_0 + x_3z_2 + x_3z_3 + x_3 + x_4z_0 + x_4z_1 \\
&\quad + x_4z_2 + x_4z_4 + x_4z_5 + x_4z_6 + x_5z_1 + x_5z_2 + x_5z_7 + x_5 \\
&\quad + x_6z_0 + x_6z_3 + x_6z_5 + x_6z_6 + x_6z_7 + x_6 + x_7z_0 + x_7z_1 \\
&\quad + x_7z_6 + x_7 + z_2 + z_3 + z_5 + z_6 \\
0 &= x_0z_0 + x_0z_3 + x_0z_5 + x_1z_4 + x_1z_7 + x_2z_2 + x_2z_4 + x_2z_7 \\
&\quad + x_3z_1 + x_3z_3 + x_3z_4 + x_3 + x_4z_1 + x_4z_3 + x_4z_6 + x_5z_0 \\
&\quad + x_5z_2 + x_5z_3 + x_5 + x_6z_0 + x_6z_1 + x_6z_2 + x_6z_4 + x_6z_5 \\
&\quad + x_6z_6 + x_7z_1 + x_7z_2 + x_7z_7 + x_7 + z_3 + z_4 + z_5 + z_6 + z_7 \\
0 &= x_0z_1 + x_0z_4 + x_0z_6 + x_1z_0 + x_1z_5 + x_2z_0 + x_2z_3 + x_2z_5 \\
&\quad + x_3z_4 + x_3z_7 + x_4z_2 + x_4z_4 + x_4z_7 + x_5z_1 + x_5z_3 + x_5z_4 \\
&\quad + x_5 + x_6z_1 + x_6z_3 + x_6z_6 + x_7z_0 + x_7z_2 + x_7z_3 + x_7 \\
&\quad + z_1 + z_3 + z_5 + z_7
\end{aligned} \tag{11}
$$

Here we have a couple of discrepancies which we again attribute to typographical mistakes in the reference paper. Our first equation in the set (11) is one-term shorter (has 32 terms) than the corresponding equation in (8) of Cui *et al.* [4] which has 33 terms. The term z_7 on the right side of Cui's equivalent equation is wrong and should be removed. We remind again that our first equation in (11) is the last in (8) of the reference paper, as Cui's equations are written in reverse than our order. Also Cui's third equation counting from the bottom (equivalent to our third from the top of (11)) has minor mistakes in comparison to our equation: This equation contains wrongly the terms x_2 and x_7 while the terms x_0, x_4 and x_5 contained in our corresponding equation are missing. In the fourth from the bottom equation in Cui's set (8) there is missing one term x_3z_7, and so the equation has one term less than our corresponding equation. The rest of terms and equations are identical.

Using the Sage model of this S-box it is easy to count the number of equations, the number of terms in each equation and determining the minimal and maximal number of terms, as well as the total number of different terms, as shown in Listing 4.

Listing 4: Survey of first S-box equation system

```
mq1 = eqs5[1:] + eqs7 + eqs8
len(mq1)
lmon1 = [len(_.monomials()) for _ in mq1]
min(lmon1)
max(lmon1)
len(set(flatten([_.monomials() for _ in mq1])))
```

As mentioned above, the first equation is discarded since it is only true with probability 255/256 (false if $\mathbf{x} = 0$). This gives for the Rijndael S-box 23 equations, with between 28 and 49 monomials per equation and, in total, 81 different monomials.

Finding the 256 solutions of this equation system representing the value table of the byte S-box with a SAT solver in Sage is accomplished with the following two lines of code:

Listing 5: SAT solver usage

```
from sage.sat.boolean_polynomials import solve as solve_sat
%time r = solve_sat(eqs5[1:] + eqs7 + eqs8, n=infinity)
```

This takes ca. 0.5 s CPU time on a decent computer (2.8 GHz CPU, 8 GB RAM).

Courtois and Pieprzyk [6] state that these 23 equations are linearly independent. Nonetheless, using the SAT solver it can be demonstrated that the last 16 equations (9) only are already sufficient to establish just the S-box value table of 256 solutions. For the solution of the 16 equation system the SAT solver takes ca. 0.7 s CPU time on the same computer.

Listing 6: Survey of shorter S-box equation system

```
mq2 = eqs7 + eqs8
len(mq2)
lmon2 = [len(_.monomials()) for _ in mq2]
min(lmon2)
max(lmon2)
len(set(flatten([_.monomials() for _ in mq2])))
```

The 16 equations describing the Rijndael S-box have between 28 and 49 monomials per equation and, in total, 81 different monomials.

5 Algebraic attacks and S-box optimization

For quantifying the resistance against algebraic attacks for r equations in t terms over $GF(2^n)$ Cui *et al.* [4] have used the criterion of Cheon and Lee [9] which defines the Resistance against Algebraic Attacks (RAA) Γ as:

$$\Gamma = \left(\frac{t-r}{n}\right)^{\lceil (t-r)/n \rceil} \tag{12}$$

Courtois and Pieprzyk [6] use another criterion

$$\Gamma_{\mathrm{CP}} = \left(\frac{t}{n}\right)^{\lceil t/r \rceil} \tag{13}$$

(note the brackets $\lceil\ \rceil$ in the exponent indicating the ceiling function).[1] The value of these criteria should reflect the difficulty of solving multivariate equations. For the Rijndael S-box we counted 23 equations and, in total, 81 different monomials. Therefore, it has $\Gamma = (29/4)^8 \approx 2^{22.9}$ and $\Gamma_{\mathrm{CP}} = (81/8)^4 \approx 2^{13.4}$. The 16 equations system has $\Gamma = (65/8)^9 \approx 2^{27.2}$ and $\Gamma_{\mathrm{CP}} = (81/8)^6 \approx 2^{20.0}$. Compared with the relation of the computational effort of the SAT solver for the MQ with 23 and 16 equations, the values for the 16 equation system seem exaggerated.

To generate a *harder* to solve equation system Cui *et al.* [4] have introduced a more complicated Rijndael S-box structure which they name Affine-Inverse-Affine (AIA) structure. This S-box will be explored in detail in the next two sections.

6 MQ of the AIA structure S-box in Sage

In Cui, Huang, *et al.* (2011) [5], a new Rijndael S-box structure named *Affine-Inverse-Affine* (AIA) is designed in order to try to increase the algebraic complexity of said S-box.

A different affine transformation (2) with $\mathbf{a}$ = '5B' and $\mathbf{b}$ = '5D' is chosen. This transformation is applied before and after the inversion step:

$$S_{\mathrm{AIA}}(\mathbf{x}) = A \circ I \circ A = A(I(A(\mathbf{x})))$$

Cui *et al.* [4] derive a multivariate quadratic equation system of S_{AIA} using the coefficients of the polynomial expression of the S-box. They write up the equation system with indices for rounds and input bytes for the AES algorithm (without further using them) which will, for clarity, be omitted in what follows. As before, by $\mathbf{x}$ is denoted the input byte variable of the S-box function. Intermediate variables are denoted by $\mathbf{y}_0$, $\mathbf{y}_1$, ..., $\mathbf{y}_{253}$ and the output variable by $\mathbf{z}$. According to the polynomial expression of the new AIA S-box, the S-box transformation can be described by the following quadratic equations over GF(2^8):

$$\begin{cases} \mathbf{x}\,\mathbf{y}_0 = 1 \\ \mathbf{y}_m\,\mathbf{y}_0 = \mathbf{y}_{m+1}\,, \text{ for } 0 \le m \le 252 \text{ and } \mathbf{y}_{253} = \mathbf{x} \\ \mathbf{z} = g\,(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{252}, \mathbf{x}) \end{cases} \tag{14}$$

[1] Parameter n could be interpreted as number of dependent variables, see section 6.

Cui *et al.* [4] define the function g by the polynomial expression of their S-box. They state to list its coefficients in their Table 1 which is taken from Cui, Huang, *et al.* [5] and coincides with Table 3 in the latter. But inexplicably, these coefficients do not correspond to the polynomial expression of their S-box S_{AIA}. Although, Cui, Huang, *et al.* [5] list in their preceding Table 2, correctly, the output values of S_{AIA}. We have calculated the coefficients for the polynomial expression of S_{AIA} (its Lagrange polynomial) in Sage (in Listing 8 the result is shown). Thereby, the function g reads

$$\begin{aligned} &g\left(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{252}, \mathbf{x}\right) = \\ &\quad \text{'FA'} + \text{'12'}\,\mathbf{x} + \text{'26'}\,\mathbf{y}_{252} + \ldots + \text{'E5'}\,\mathbf{y}_2 + \text{'A9'}\,\mathbf{y}_1 + \text{'A6'}\,\mathbf{y}_0 \end{aligned} \tag{15}$$

This equation system can be modeled in Sage with the help of the preparatory code shown in Listing 7:

Listing 7: Preparation for equation system of AIA S-box

```
1   nb = 8
2   ny = 253
3   varlxz = [c + str(p) for c in 'xz' for p in range(nb)]
4   varly = ['y' + str(p) for p in range(nb*ny)]
5   B = BooleanPolynomialRing(names = varlxz + varly)
6   B.inject_variables()
7   P.<p> = PolynomialRing(B)
8   Byte.<t> = P.quotient_ring( p^8 + p^4 + p^3 + p + 1 )
9   X = B.gens()[:nb]
10  Z = B.gens()[nb:][:nb]
11  YY = [B.gens()[(2+m)*nb:][:nb] for m in range(ny)]
12  x = sum([X[j]*t^j for j in range(nb)])
13  z = sum([Z[j]*t^j for j in range(nb)])
14  yy = [sum([_Y[j]*t^j for j in range(nb)]) for _Y in YY]
```

In lines 3 and 4 lists of strings for the names of coefficients for the byte variables $\mathbf{x}$, $\mathbf{z}$, and $\mathbf{y}_0, \ldots, \mathbf{y}_{252}$ are generated.

Then, as in Listing 1, in line 5 a Boolean polynomial ring for these coefficients is constructed, assigning $GF(2)$ properties to them, in line 6 the coefficients are made available as variables and in line 7 a polynomial ring over the Boolean polynomial ring `B` is constructed and from that in line 8, eventually, the quotient ring `Byte` with modulus $m(t)$.

In lines 9 to 11 lists of the coefficient variables of the byte polynomials are created for convenience. For the $\mathbf{y}$-variables the coefficients are grouped byte-wise in sub-lists. With the help of these lists in lines 12 to 14, finally, the polynomials of the byte variables are modeled. For the $\mathbf{y}$-variables a list of polynomials is used.

In the next Sage code block (Listing 8), the coefficients of the polynomial expression of S_{AIA} are given in hexadecimal notation as a list of strings which is transformed to a list of the constant polynomials that enter g (15):

Listing 8: Generating constant polynomials of AIA S-box

```
sbt = [
 'FA 12 26 E7 9A C7 DB 79 56 01 D3 59 52 ED 97 C9',
 '47 46 FC 7C 5A 50 49 BF F4 F8 63 C8 82 1B EE 74',
 '3B 5D F8 02 2D 64 1A 15 BA DB 59 FE FB D6 97 FF',
 'AB 3F B4 09 32 77 AB 52 4D 96 D5 BB DE 30 DE 05',
 '62 23 7C 69 66 75 9F E9 9B 60 88 2F D1 8F 09 F4',
 '1E EF C4 48 0D A5 AE 7A 38 9B 71 F2 9F 44 B3 99',
 '20 C5 13 12 19 C2 5F 5B AD FA D5 49 7B F8 16 07',
 'B6 75 E9 B0 CA E8 83 C1 4E 75 C5 5E 91 07 86 BF',
 '6F C2 25 35 D3 7F CC 0D AC 7A C9 EC D2 3F C3 21',
 '7E A9 2A 6D A8 66 F8 7D D2 1B FE CD 58 64 25 DA',
 'AE 49 2D 4F 0C 74 F2 42 4A 87 42 9B 83 50 F1 91',
 'C1 02 4F 2A C9 19 37 59 D5 74 8D 0B 20 C5 AF 28',
 '47 FB 09 87 10 6A 3B C8 8B 08 5B 8B 13 0E 73 7E',
 'FA 45 85 18 D5 90 4E 71 E6 F2 BF EE 30 E9 99 54',
 '30 63 8F 03 92 91 0C 43 09 66 E5 76 6A 93 87 E4',
 '6C 6A 87 A1 CB 64 AA 5C FB 05 5A DE E5 A9 A6 00']

sbt = flatten([_.split(' ') for _ in sbt])
sbp = [Byte(list(ZZ('0x'+_).str(2)[::-1])) for _ in sbt]
```

In the last line of this code block each two-digit hexadecimal number in the table represented by a two character string is converted into a decimal number by the code fragment `ZZ('0x'+_)`. By appending `.str(2)[::-1]` it is transformed into a string of 0s and 1s for the big-endian binary representation of the hexadecimal number. The string is split into a list of single letters which as argument of the `Byte`-ring gives the corresponding constant polynomial.

With these preparations the equation system (14), (15) of the AIA S-box (equation 9 in Cui *et al.* [4]) can be modeled in Sage as shown in Listing 9.

Listing 9: Equation system of AIA S-box in Sage

```
g = sbp[0] + sbp[1] * x \
    + sum(sbp[2+m] * yy[ny-1-m] for m in range(ny))

E9 = [x * yy[0] + 1]
E9.extend( yy[m] * yy[0] + yy[m+1] for m in range(ny-1) )
E9.append( yy[252] * yy[0] + x )
E9.append( z + g )
eqs9 = flatten([_.list() for _ in E9])
```

Using this Sage model it is easy to count the number of equations, the number of terms in each equation and determine the minimal and maximal number of terms, as well as the total number of different terms, as shown in Listing 10.

Listing 10: Survey of AIA S-box equation system

```
len(eqs9[1:])
lmon9 = [len(_.monomials()) for _ in eqs9[1:]]
min(lmon9)
max(lmon9)
len(set(flatten([_.monomials() for _ in eqs9[1:]])))
```

The first equation is discarded since it is only true with probability 255/256 (false if $\mathbf{x} = 0$).

This gives for the new AIA S-box 2,039 equations, with between 3 and 1,034 monomials per equation. These equations have in total 18,232 different monomials. In order to estimate the resistance against algebraic attacks, also the intermediate variables were taken into account by interpreting the parameter n in the definitions (12) and (13) as the number of dependent variables. Hence, the number 8×254 is used as n, and not only 8. This results in $\Gamma = (16,193/2,032)^8 \approx 2^{24.0}$ and $\Gamma_{\mathrm{CP}} = (18,232/2,032)^9 \approx 2^{28.5}$ as estimation for RAA.[2]

Also, the CPU time to evaluate all 256 solutions of this MQ with a SAT solver is ca. 7 s, which is 14 times the time used for the original Rijndael S-box.

Cui *et al.* [4] count a totally different number of equations and terms based on the byte variables, not on their polynomial coefficients, which contradicts the scheme applied to the Rijndael S-box with which they compare, and therefore, is misleading.

Further, this method used by Cui *et al.* [4] to derive an MQ for their AIA S-box applies to any S-box using the coefficients of its polynomial expression. Therefore, the resulting numbers of equations and terms is deceptive as criterion for the estimation of algebraic attack resistance and inapt to differentiate the quality of byte S-boxes. To substantiate this point, we have derived such an MQ for the original Rijndael S-box by using its polynomial expression in equation (14). The function g then reads

$$\begin{aligned} &g_{\mathrm{SRD}}(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{252}, \mathbf{x}) = \\ &\quad \text{'63'} + \text{'8F'}\,\mathbf{y}_{127} + \text{'B5'}\,\mathbf{y}_{63} + \text{'01'}\,\mathbf{y}_{31} + \text{'F4'}\,\mathbf{y}_{15} + \text{'25'}\,\mathbf{y}_7 + \\ &\quad \text{'F9'}\,\mathbf{y}_3 + \text{'09'}\,\mathbf{y}_1 + \text{'05'}\,\mathbf{y}_0 \end{aligned} \tag{16}$$

The same Sage code (Listings 7, 8, and 9) was used with an adapted table of the polynomial expression coefficients given in Listing 11.

Listing 11: Polynomial expression coefficients of RD S-box

```
sbt = [
 '63 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 8F',
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
```

[2] Formal application of $n = 8$ yields unlikely high values: $\Gamma \approx 2^{22,241}$ and $\Gamma_{\mathrm{CP}} \approx 2^{100}$.

```
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B5',
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00',
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01',
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F4',
'00 00 00 00 00 00 00 25 00 00 00 F9 00 09 05 00']
```

It is found that this MQ of the Rijndael S-box has the same number of equations with the same number of different monomials as the MQ of S_{AIA} resulting in equally high, misleading Γ values. Also, the SAT solver needs the same 7 s CPU time to find the solutions of this MQ.

In contrast, we will show in the next section that it is easy (at least by using computer mathematics) to derive a much simpler MQ for the AIA S-box which shows that its resistance against algebraic attacks according to the effort of a SAT solver is not that much greater than that of the original Rijndael S-box. But the RAA criteria seem to exaggerate the hardness of that simpler MQ.

7 Concise MQ for the AIA S-box in Sage

Building on the Sage code presented so far we derive a much simpler MQ for the AIA S-box. Its resistance against algebraic attacks according to the RAA criterion is greater than that of the original Rijndael S-box but over-estimates the time it takes to solve the system with a SAT solver.

For the inversion step we now use, temporarily, two intermediate byte variables $\mathbf{y}_0$ and $\mathbf{y}_1$, named `yy[0]` and `yy[1]` in the Sage code. Their coefficients shall be $y_0, \ldots, y_7$ and $y_8, \ldots, y_{15}$, respectively. The three steps of the AIA S-box are

$$\mathbf{z} = A(\mathbf{y}_1)\,,\ \mathbf{y}_1 = I(\mathbf{y}_0)\,,\ \mathbf{y}_0 = A(\mathbf{x})$$

Beginning with the inversion step we first model

$$\begin{cases} \mathbf{y}_0^{128} &= \mathbf{y}_1^{128}\mathbf{y}_0 \\ \mathbf{y}_1^{128} &= \mathbf{y}_0^{128}\mathbf{y}_1 \\ \mathbf{y}_0^{3} &= \mathbf{y}_0^{4}\mathbf{y}_1 \\ \mathbf{y}_1^{3} &= \mathbf{y}_1^{4}\mathbf{y}_0 \end{cases} \tag{17}$$

The last two equations in (17) are the only other additional (fully quadratic) MQ (besides $\mathbf{y}_0\mathbf{y}_1 = 1$) for the inversion as stated already by Courtois and Pieprzyk in the extended version of [6] (compare also Cheon and Lee [9]). These additional equations are necessary to completely define the S-box S_{AIA}. Without them the system is under-defined, as, for example, the solution with a SAT solver shows. In Sage the equations (17) read

```
E10 = yy[0]^128 + yy[1]^128 * yy[0]
E11 = yy[1]^128 + yy[0]^128 * yy[1]
E12 = yy[0]^3 + yy[0]^4 * yy[1]
E13 = yy[1]^3 + yy[1]^4 * yy[0]
```

The linear transformations according to equation (2) are

$$\mathbf{y}_0 = \mathbf{a}\,\mathbf{x} \bmod (t^8 + 1) + \mathbf{b} \tag{18}$$

$$\mathbf{y}_1 = \mathbf{a}^{255}(\mathbf{z} + \mathbf{b}) \bmod (t^8 + 1) \tag{19}$$

with $\mathbf{a} =$ '5B' and $\mathbf{b} =$ '5D'. In Sage we formulate

Listing 12: Generate dictionary to substitute $\mathbf{y}$ variables

```
1  Baff.<u> = P.quotient_ring( p^8 + 1 )
2  a = Baff(list(ZZ('0x5B').str(2)[::-1]))
3  b = Baff(list(ZZ('0x5D').str(2)[::-1]))
4  x = sum([X[j]*u^j for j in range(nb)])
5  z = sum([Z[j]*u^j for j in range(nb)])
6  eqs14 = dict(zip(YY[0], a * x + b))
7  eqs14.update(zip(YY[1], a^255 * (z + b)))
```

The right hand side of equation (18) reads `a * x + b` in line 6 of Listing 12 and the rhs of (19) is `a^255 * (z + b)` in line 7. Both their coefficients are inserted into the same dictionary `eqs14`. The result is shown in Listing 13.

Listing 13: Combined dictionary to substitute $\mathbf{y}$ variables

```
{y15: z4 + z5 + z6,
 y14: z3 + z4 + z5,
 y13: z2 + z3 + z4 + 1,
 y12: z1 + z2 + z3,
 y11: z0 + z1 + z2,
 y10: z0 + z1 + z7 + 1,
 y9: z0 + z6 + z7,
 y8: z5 + z6 + z7 + 1,
 y7: x1 + x3 + x4 + x6 + x7,
 y6: x0 + x2 + x3 + x5 + x6 + 1,
 y5: x1 + x2 + x4 + x5 + x7,
 y4: x0 + x1 + x3 + x4 + x6 + 1,
 y3: x0 + x2 + x3 + x5 + x7 + 1,
 y2: x1 + x2 + x4 + x6 + x7 + 1,
 y1: x0 + x1 + x3 + x5 + x6,
 y0: x0 + x2 + x4 + x5 + x7 + 1}
```

Applying this substitutions in Sage and, thereby, getting rid of the intermediate byte variables is straight forward

```
eqs10 = [_.substitute(eqs14) for _ in E10.list()]
eqs11 = [_.substitute(eqs14) for _ in E11.list()]
eqs12 = [_.substitute(eqs14) for _ in E12.list()]
eqs13 = [_.substitute(eqs14) for _ in E13.list()]
```

and gives already the final, concise MQ for the S-box S_{AIA}. For the resulting equation system we count the numbers of equations and monomials in Sage.

Listing 14: Survey of short AIA S-box MQ

```
mq5 = eqs10 + eqs11 + eqs12 + eqs13
```

```
len(mq5)
lmon5 = [len(_.monomials()) for _ in mq5]
min(lmon5)
max(lmon5)
len(set(flatten([_.monomials() for _ in mq5])))
```

This MQ has 32 equations, with between 33 and 60 monomials per equation. These equations have in total 137 different monomials. This makes according to definitions (12) and (13) $\Gamma = (105/8)^{14} \approx 2^{52}$ and $\Gamma_{\mathrm{CP}} = (137/8)^5 \approx 2^{20.5}$.

Nonetheless, for this MQ with 32 equations the SAT solver takes ca. 0.8 s CPU time on the same computer (2.8 GHz CPU, 8 GB RAM) to find all and only 256 solutions. Clearly, the values of the hardness criteria do not correlate well with the effort of the SAT solver for this MQ.

For comparison and as an additional reference value for the RAA estimation, we have derived such an MQ with 32 equations for the original Rijndael S-box using the four equations (17) (replacing $\mathbf{y}_0$ by $\mathbf{x}$ and $\mathbf{y}_1$ by $\mathbf{y}$) and its affine transformation (5) (with $\mathbf{a} =$ '1F', $\mathbf{b} =$ '63'). This MQ has the same number of equations and the same number of different monomials, thus, the same values for the hardness criteria as that of S_{AIA}. The solution with the SAT solver of this MQ for the original Rijndael S-box takes the same CPU time as the solution of the 32 equation MQ of S_{AIA}, namely, ca. 0.8 s. This substantiates the doubt about the practicality of the hardness criteria.

8 Conclusion

SageMath is a very appropriate, powerful computer mathematics tool to analyze cryptographic problems formulated with byte variables as polynomials in a quotient ring. Sage draws its strength in this area mainly from the integration of the PolyBoRi library [1, 10].

We have used Sage to demonstrate how to produce various polynomial Multivariate Quadratic equations systems (MQ) for the Rijndael S-box and similar S-boxes in a simple and straightforward manner. Using the flexible structures and interface of Sage one can easily count the parameters of the resulting polynomial systems, like the number of different monomials in the system, the length of the equations, the frequency of the appearance of certain terms or variables in equations etc. We have used this facility to derive the necessary inputs for the estimation of the Resistance against Algebraic Attacks (RAA) according to Cui *et al.* [4], as well as to Courtois and Pieprzyk [6]. Parallelly, we performed numerical experiments by solving the corresponding MQ with a SAT solver in order to check if the formulas for the RAA predict a scaling accordingly to the solver efforts.

Our results in this respect are revealing. We couldn't validate the pre-

dicted reduced hardness for a system to solve with increasing number of equations and number of independent monomials in the polynomial systems which both the formulas forecast. There is in fact a slight increased solver effort when one reduces from the 23 Rinjndael s-box equations to the 16 but quantitatively this is badly represented in both formulas.

Cui *et al.* constructed a complicated algebraic expression for a new Rijndael S-box (starting from its Lagrange polynomial expression with 255 coefficients) which necessarily leads to a great number of equations and independent monomials in the resulting MQ. On this basis they thought they have demonstrated a remarkable new S-box practically not possible to solve according to the here discussed and by us criticized as inappropriate RAA formulas. However there are gaps in their concept arising from inconsistency in their comparison principle as well as the lack of thoroughness in the investigation of the properties of the new algebraic expression which we showed can be equivalently written in a much simpler form leading to a polynomial system nearly as easy solved as that of the original Rijndael S-box.

We also mapped the original Rijndael S-box with its 9 Lagrange coefficients on the AIA form of Cui *et al.* which gave us as result the same *huge* number of variables and multitude of polynomials which should be a prove that this way to create especially hard cryptographic S-boxes is not the right one.

In this way we showed that the, by Cui *et al.* so called, *improved* AIA S-box is in fact at most marginally an improvement.

Table 1 gives a survey of the MQ and the results of the algebraic attack

S-Box MQ	RD	RD	RD	RD	AIA	AIA
# equations	23	16	32	2,039	2,039	32
# monomials	81	81	137	18,232	18,232	137
# dependent variables	8	8	8	2,032	2,032	8
$\log_2(\Gamma)$	22.9	27.2	52.0	24.0	24.0	52.0
$\log_2(\Gamma_{\mathrm{CP}})$	13.4	20.0	20.5	28.5	28.5	20.5
SAT solver CPU time	0.5 s	0.7 s	0.8 s	7 s	7 s	0.8 s

Table 1: Survey of S-box MQ and estimations of their RAA.

resistance estimations presented in this work.

We conclude, that in order to assess the resistance of an S-box against algebraic attacks it is not sufficient to derive some multivariate quadratic equation system and analyze it. To really find an estimation one has to show that the derived MQ is optimal for solving and, thus, attacking it.

References

[1] William A. Stein *et al.*, *Sage Mathematics Software (Version 6.7). The Sage Development Team*, 2015, http://sagemath.org.

[2] Wikipedia: SageMath, Sage (mathematics software), https://en.wikipedia.org/wiki/SageMath.

[3] SageMathCloud, collaborative computational mathematics and course management; founder and main architect is William Stein, https://cloud.sagemath.com/.

[4] Jie Cui, Hong Zhong, Jiankai Wang and Runhua Shi, *Generation and Optimization of Rijndael S-box Equation System*, Information Technology Journal, 13: 2482–2488, 2014.

[5] Jie Cui, Liusheng Huang, Hong Zhong, Chinchen Chang and Wei Yang, *An improved AES S-Box and its performance analysis*, Int. J. Innovative Comput. Inform. Control, 7: 2291–2302, 2011.

[6] Nicolas T. Courtois and Josef Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, Proc. of Asiacrypt 2002, LNCS 2501, Springer-Verlag, 267–287, 2002.
An extended version of this paper is available on eprint.iacr.org http://eprint.iacr.org/2002/044.

[7] Joan Daemen and Vincent Rijmen, *The design of Rijndael: AES – the Advanced Encryption Standard*, Springer-Verlag, 2002, http://jda.noekeon.org/.

[8] *Specification for the Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication 197, 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[9] Jung Hee Cheon and Dong Hoon Lee, *Resistance of S-Boxes against Algebraic Attacks*, Lecture Notes in Computer Science 3017, 83–93, 2004.

[10] Michael Brickenstein and Alexander Dreyer, *PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials*, Journal of Symbolic Computation 44, 1326–1345, 2009, http://dx.doi.org/10.1016/j.jsc.2008.02.017.