Representations and Rijndael Descriptions[⋆]

Vincent Rijmen and Elisabeth Oswald

IAIK, Graz University of Technology,
Inffeldgasse 16a, A-8010 Graz, Austria
{vincent.rijmen, elisabeth.oswald}@iaik.tugraz.at

Abstract. We discuss different descriptions of Rijndael and its components and how to find them. The fact that it is easy to find equivalent descriptions for the Rijndael transformations, has been used for two different goals. Firstly, to design implementations on a variety of platforms, both efficient and resistant against side channel analysis. Secondly, to analyze the security of the cipher We discuss these aspects, give examples, and present our views.

1 Introduction

In this paper, we give an overview of recent developments in the study of Rijndael security and efficient Rijndael implementations. Central to many of these developments is the technique of changing representations, and therefore we take this as the central theme of our treatment here.

When we look at what has been published about Rijndael in the last couple of years, we see that most authors restrict in their studies the possible changes of representation to the set of polynomial bases in a finite field, e.g. selection of a different base element or the selection of a different reduction polynomial. However, finite fields have a much richer structure, e.g. they can also be described as vector spaces over the ground field. It is our belief that exploration of the vector space representation can bring us to new insights in both security and efficient implementation of the Rijndael.

We start this paper by setting the framework to study different representations and the resulting equivalent descriptions for Rijndael. Afterwards, we present the overview of recent results and place them in our framework.

2 Change of Representation: An Old Mathematical Technique

It is well-known that the choice of representation influences the complexity of most problems related to algebra. One example with application in cryptography

⋆ This research was supported financially by the A-SIT, Austria.

is given by elliptic curves. An arbitrary elliptic curve has a defining equation of the following form:

$$Ay^2 + Byx + Cy = Dx^3 + Ex^2 + Fx + G. \tag{1}$$

By choosing another representation, the defining equation can be transformed into the following form:

$$y^2 = x^3 + ax + b \ . \tag{2}$$

For all defining equations of the form (1), there is an equation of the form (2) defining an elliptic curve with the same mathematical properties, although both curves contain different points (x, y).

A second example is the gate complexity of a circuit that implements the squaring operation in a finite field with characteristic two. If the elements of the field are represented by their coordinates with respect to a normal basis, then the squaring operation corresponds to a simple rotation of the coordinates. In other representations, the squaring operation corresponds to a more complicated linear transformation of the coordinates.

The problem we address in this paper, is exactly the opposite problem. When given a Boolean transformation with fixed 'points' $(x, S(x))$, we want to find a simple algebraic description for this Boolean transformation.

3 Boolean Transformations and Algebras

In order to improve understanding of the issues related to equivalent descriptions, it is important to clearly define the terminology. We make a distinction between two mathematical concepts that are often used as synonyms. These concepts are an *abstract element of an algebra* on the one hand, and the *representation of the element* on the other hand. We start with a definition for an algebra.

An *algebra* consists of one or more sets of elements and one or more *operations* between the elements. We will consider here algebras that contain only one set of elements, denoted by $\mathcal{A}$. Furthermore, we will assume that the cardinality of $\mathcal{A}$ equals 2^n for some integer value n.

An m-ary operation b maps an input consisting of m elements of $\mathcal{A}$ to an output, which is also in $\mathcal{A}$.

$$b : \mathcal{A}^m \to \mathcal{A} : (x_1, x_2, \ldots, x_m) \mapsto b(x_1, x_2, \ldots, x_m) = y \tag{3}$$

A 1-ary operation is also called an (algebraic) function.

A *Boolean vector* is a one-dimensional array of bits. By *Boolean transformation*, we mean a function S that maps a Boolean vector to another Boolean vector. For sake of simplicity, we will assume that the input and output vectors have equal size.

$$S : Z_2^n \to Z_2^n : \mathrm{x} \mapsto \mathrm{y} = S(\mathrm{x}) \tag{4}$$

A *representation* ρ maps the elements of $\mathcal{A}$ to n-bit Boolean vectors.

$$\rho : \mathcal{A} \to Z_2^n : x \to \rho(x) = \mathrm{x} \tag{5}$$

The inverse map of a representation is called a *labeling map*. A representation, or labeling, defines a map from the algebraic functions to the Boolean transformations:

$$R(b) = S_b \Leftrightarrow \forall x \in \mathcal{A} : \rho(b(x)) = S_b(\rho(x)) \ . \tag{6}$$

$$\begin{tikzcd}
\text{Algebra elements} \arrow[r, shift right, "\text{representation}"'] & \text{Boolean vectors} \arrow[l, shift right, "\text{labeling}"'] \\
\text{Operations} & \text{Boolean transformations} \arrow[l, "\text{description}"]
\end{tikzcd}$$

Fig. 1. Terminology w.r.t. algebra elements and Boolean vectors

3.1 Finding Descriptions

Finding an algebraic description for a Boolean transformation S can be done in three steps.

Initialization: Decide on an algebra to be used.
Labeling: Define a labeling map from the Boolean vectors to elements of the algebra.
Describing: Compute the description of the Boolean transformation S.

Before describing the steps in some more detail, we briefly return to the previous example. Suppose we have a Boolean transformation S that implements a rotation of the bits: $S(\mathrm{x}) = \mathrm{x} \lll 1$. Suppose further that we have chosen to use the finite field $GF(2^n)$ as the algebra to work in. Since S is linear over this field, it can always be described with a linear polynomial $l(x)$:

$$l(x) = \sum_{i=0}^{n-1} a_i x^{2^i} , \tag{7}$$

where the a_i are some constants. If the Boolean vectors x are considered as the representation of the coordinates of the elements x with respect to a normal basis, then the polynomial describing S can be as simple as $l(x) = x^2$.

Selecting an Algebra. For the values of n that are of practical importance, the number of algebras that can be defined, is very large: $(2^n)^{2^{2n}}$ different binary operations can be defined. However, the requirement to obtain a 'simple' description in practice limits the number of interesting algebras. The most natural choices are perhaps the vector space $(\mathrm{GF}(2))^n$ or the field $\mathrm{GF}(2^n)$. However, other algebras can be used as well, e.g. the algebra $< Z_{2^n+1} \backslash \{0\}, \times >$ as used in IDEA.

Selecting a Labeling. Once an algebra has been selected, the elements of the algebra have to be assigned to the Boolean vectors. We call this the *labeling* of the Boolean vectors. Let the labeling map be denoted by m:

$$m : (Z_2)^n \to \mathcal{A}. \tag{8}$$

The number of labeling maps equals $(2^n!)$. Note that this is more general than a change of basis in a finite field.

Computing the Representation. Once the algebra and the labeling map are defined, the input-output tuples of the transformation S can be translated into tuples of the algebra:

$$(x, S(x)) \to (a, b) = (m(x), m(S(x))). \tag{9}$$

The functional description of S in the new representation can be derived from the input-output tuples, using for instance the Lagrange interpolation formula, if it can be applied in the algebra selected.

4 Rijndael Descriptions

Equivalent descriptions can be investigated for any block cipher. Indeed, already in the 1980's, several results appeared about equivalent descriptions for the DES [6]. Afterwards, the topic seems to have died out in the field of symmetric cryptography. The selection of Rijndael to become the AES has triggered new research in this direction.

4.1 Number of Equivalent Descriptions

The design of Rijndael was made in the field GF(256). All design criteria were made and the selection of components was done with this field in mind. In order to be able to define the input-output behavior uniquely, one specific representation of the field elements had to be chosen. The choice was made to use a binary polynomial basis, with irreducible polynomial $p(t) = t^8 + t^4 + t^3 + t + 1$.

In [2], it was observed that 240 *equivalent ciphers* (i.e. alternative descriptions) can be generated by choosing one of the 30 irreducible binary polynomials of degree 8 and by choosing one of the 8 roots of this polynomial as generator.

However, there are many more alternative representations possible. The field GF(256) is isomorphic to the field $(GF(2))^8$, which is also an 8-dimensional vector space. In this vector space, there are

$$\prod_{i=0}^{7}(2^8 - 2^i) \approx 2^{62} \tag{10}$$

different bases. Each base leads to a different labeling of bytes and hence a different description of Rijndael. In [3], 2040 bases with a special property are derived.

The authors combine these 2040 bases with the 30 irreducible polynomials in order to define 61200 equivalent cipher descriptions.

Equivalent descriptions can also be constructed by defining an arbitrary bijective labeling map m. There are 256! different such labeling maps and at least as many different descriptions. Finally, observe that labeling maps don't have to be bijective. Also injective maps can be used (cf. infra), and hence there are infinitely many labeling maps.

4.2 Useful Representations

The vast majority of the 256! equivalent descriptions of Rijndael will not result in any new insights. Indeed, only the 2^{62} descriptions that are constructed following a change of basis in the vector space $(GF(2))^8$, have the property that 'addition' corresponds to binary exclusive-or. In all other descriptions, the specification of addition will require the use of tables without any apparent structure.

The transformations MixColumns and AddRoundKey can be described by very simple operations when the default representation is used. This is a second reason to look for new representations 'close' to the default representation.

5 Descriptions Assisting Implementations

Alternative descriptions facilitating implementations are used mostly on constrained platforms: hardware and small processors. In environments with little constraints, the default description of Rijndael seems to be as good as any other one.

The addition of side-channel attack countermeasures usually decreases the performance and/or increases the cost of an implementation. Hence, the techniques developed to improve the performance of ordinary implementations in constrained environments, are usually also of use in side-channel attack resisting hardware.

5.1 Hardware Efficient Descriptions

As explained before, the use of alternative finite field representations in order to reduce the gate count of a circuit is a well-established technique. Several alternative representations have been proposed in the cryptographic literature, mainly in order to improve the implementation of the SubBytes transformation.

The first type of alternative representation is to label bytes as polynomials of degree smaller than 2, with coefficients in GF(16):

$$m : Z_2^8 \rightarrow GF(16)[t]/(t^2 + At + B) : x \mapsto m(x) = at + b, \tag{11}$$

with $a = m_1(x), b = m_2(x)$. The maps m_1, m_2 have to satisfy some conditions and the coefficients A, B are chosen such that the polynomial $t^2 + At + B$ is irreducible. Then, the transformation SubBytes can be described by one nonlinear formula of the form

$$(at+b)^{-1} = (b^2B + baA + a^2)^{-1}(bt + a + bA), \tag{12}$$

combined with linear and affine operations, which depend on the choice for A, B and the details of the maps $m_1(x), m_2(x)$. Descriptions of this type have been proposed in [12, 13, 14, 16].

The alternative representations mainly improve the gate complexity of the SubBytes step, while the complexity of the other steps remains the same, or deteriorates slightly. This results in different approaches. In the first approach, the SubBytes is implemented in the representation that is best for that step, and the other steps are implemented using another representation. This approach necessitates changes of representation in between steps [14, 16].

In the second approach, frequent changes of representation are avoided by adopting a 'compromise' representation, which improves the complexity of SubBytes, and doesn't increase the complexity of other steps too much [13].

In the third approach, an alternativ representation, which is best for the SubBytes step, is combined with an equivalent AES [17].

5.2 Representations Assisting SCA Countermeasures

Side-channel attacks are used to extract secret key material from real systems. It has been observed that computing hardware and software often leak information about secret keys used in cryptographic algorithms. This leakage comes from variations in execution time, power consumption, radiation, etc.

Many proposals for hardware designs that resist side-channel attacks, are based on *masking* techniques: the sensitive values are never manipulated directly, but only in blinded, or masked, form. The mask is a random value, which needs to be processed separately. Such a masking scheme can also be described as a secret sharing scheme, where the masked and the masked value are two *shares*. Hence, a masking scheme by itself can already be seen as an expanding alternative representation.

For linear operations, it is well-known what masking schemes to use and how to implement them. For the non-linear operation of Rijndael, there are several proposals.

5.3 Additive Split

In order to implement a linear operation $L(x)$, the input x is represented by a tuple (p, q) with $p + q = x$. We call this the *additive split* of sensitive variables. In order to compute the tuple corresponding to $L(x)$, the linear operation is performed separately on each share. Indeed, we have that if $x = p + q$, then $L(x) = L(p) + L(q)$, and hence $L(x)$ is represented by $(L(p), L(q))$.

The addition of two sensitive variables can also be protected using the additive split. The result can be computed in a secure way by simply adding the coordinates of the corresponding tuples: if x is represented by (p, q) and y by (r, s), then $x + y$ can be represented by $(v, w) = (p + r, q + s)$ or $(v, w) = (p + s, q + r)$. In both representations, v and w are completely uncorrelated to the values of x and y.

The SubBytes step in the Rijndael round transformation consists of an affine operation and the multiplicative inverse map, or, more accurately, the power function map x^{254}. Protecting non-linear maps by means of an additive split is not a straightforward process.

In order to implement this map, two functions f, g are required, such that $x^{254} = f(p) + g(q)$ (where $x = p + q$). In order to have security against first-order side-channel attacks, the implementation of maps $f(p)$ and $g(q)$ should not produce intermediate results which correlate with $p + q$. It remains an open problem whether such maps can be defined.

As an alternative solution, other types of split have been proposed in the literature and we describe them in Section 5.4. In Section 5.5, we describe a recently developed method. By using a special representation of the field elements, it becomes possible to protect also the power function by means of an additive split.

5.4 Multiplicative Split

A multiplicative split was proposed in [1]. A byte x is mapped to (p, q) with $x = pq^{254}$. It can be seen that the tuple corresponding to x^{254} can be computed as $(p^{254}, q^{254}) = (q, p)$. Hence this representation would allow to implement the power function with a simple swap of registers. Alas, it seems from [1] that the requirement to change the representation from additive split to multiplicative split and vice versa, makes it necessary to compute the power functions of the two shares explicitly.

A more important disadvantage is the so-called zero multiplication problem, which refers to the fact that a multiplicative split fails to hide the zero value:

$$x = 0 \Leftrightarrow (p = 0 \text{ or } q = 0). \tag{13}$$

One approach to fix this problem is to introduce a second injective map, that maps the elements of GF(256) to a larger ring containing zero divisors [9].

5.5 Additive Split in Tower Fields

This technique has been described in [11]. It builds on the techniques explained in [16]. During all operations, the variables are protected by means of an additive split. In [4], an alternative method was developed, based on the same principles.

In the tower field representation, bytes are labeled as polynomials of degree smaller than 2. The two coefficients of the polynomials are elements of GF(16). For some of the computations, these coefficients in turn are labeled as polynomials of degree 2, with coefficients in GF(4).

$$m: \; Z_2^8 \rightarrow GF(16)[t]/(t^2 + At + B) : x \mapsto m(x) = at + b, \tag{14}$$
$$m': \; GF(16) \rightarrow GF(4)[u]/(u^2 + Cu + D) : y \mapsto m'(y) = cu + d, \tag{15}$$

with $a = m_1(x), b = m_2(x), c = m'_1(y), d = m'_2(y)$. The maps m_1, m_2, m'_1, m'_2 have to satisfy some conditions and the coefficients A, B, C, D are chosen such that the polynomials $t^2 + At + B$ and $u^2 + Cu + D$ are irreducible over GF(16), respectively GF(4), and lead to efficient arithmetic. Note that the labeling maps are all linear and hence they can be protected as explained in Section 5.3. The multiplicative inverse map can be described in two steps, which are explained below.

Step 1: GF(16). Firstly, (12) is used to describe the map using only the following operations: addition, multiplication and taking the multiplicative inverse. All these operations are in GF(16). The implementation of the multiplicative inverse is done in Step 2. The additions in GF(16) are protected as explained in Section 5.3.

The multiplication of two sensitive variables a and b, that are represented by (p, q) and (r, s), is implemented by multiplying the coordinates p and r, and adding so-called *correction terms* c_i. The correction terms are defined in such a way that they can be computed without producing intermediate results that correlate to a, b, ab, $a + b$, a^2, b^2, or any other value that could leak information about the sensitive variables to the attacker. The result is a representation of the following form:

$$(v, w) = (pq + \sum_i c_i, q). \tag{16}$$

Step 2: GF(4). The second step is similar to the first step, but operating on smaller fields. A formula very similar to (12) describes how the multiplicative inverse in GF(16) can be computed using operation in GF(4) only.

Addition and multiplication in GF(4) are implemented and protected as described for Step 1. For the implementation of the multiplicative inverse, the following fact is used.

For all $x \in \mathrm{GF}(4)$, it holds that $x^{-1} = x^2$, hence taking the multiplicative inverse is a linear operation, that can be protected by means of an additive split, as described in Section 5.3.

6 Representations Assisting Cryptanalysis

Several alternative descriptions have been derived, showing that more elegant, more structured and more *simple* sets of equations defining Rijndael can be constructed. Although several of them seem a promising start for an attack, no breakthrough has been demonstrated yet.

6.1 BES

Murphy and Robshaw [10] define the block cipher BES, which operates on data blocks of 128 bytes instead of bits. According to Murphy and Robshaw, the algebraic structure of BES is even more elegant and simple than that of Rijndael. Furthermore, Rijndael can be *embedded* into BES. There is a map ϕ such that:

$$\mathrm{Rijndael}(x) = \phi^{-1}\left(\mathrm{BES}\left(\phi(x)\right)\right). \tag{17}$$

The map ϕ can also be seen as an injective labeling of the inputs of Rijndael.

Murphy and Robshaw proceed with some observations on the properties of BES. However, these properties of BES do not apply to Rijndael.

6.2 Redundant S-Boxes

Any 8×8-bit S-box can be considered as a composition of 8 Boolean functions sharing the same 8 input bits. J. Fuller and W. Millan observed that the S-box of Rijndael can be described using one Boolean function only [8]. The 8 Boolean functions can be described as

$$f_i(x_1, \ldots, x_8) = f(g_i(x_1, \ldots, x_8)) + c_i, \qquad i = 1, \ldots, 8, \tag{18}$$

where the function f is the only nonlinear function, the g_i are affine functions and the c_i are constants.

6.3 Continued Fractions

Ferguson, Schroeppel and Whiting [7] derive a closed formula for Rijndael that can be seen as a generalization of continued fractions. Any byte of the intermediate result after 5 rounds can be expressed as follows.

$$x = K + \sum \cfrac{C_1}{K^* + \sum \cfrac{C_2}{K^* + \sum \cfrac{C_3}{K^* + \sum \cfrac{C_4}{K^* + \sum \cfrac{C_5}{K^* + p_*^*}}}}} \tag{19}$$

Here every K is some expanded key byte, each C_i is a known constant and each $*$ is a known exponent or subscript, but these values depend on the summation variables that enclose the symbol.

A fully expanded version of (19) has 2^{25} terms. It is currently unknown what a practical algorithm to solve this type of equations would look like.

6.4 XL and XSL Methods

XL and XSL are new methods to solve nonlinear algebraic equations [5, 15]. The effectiveness and efficiency of these methods remain a topic of debate. It seems plausible that the complexity of the methods is influenced by the description that is chosen. For instance, the authors of [10] claim that using their representation, the complexity of the XSL method decreases significantly. More details about the methods can be found elsewhere in these proceedings.

7 Conclusions and Perspective on the Future

Compared with other symmetric ciphers, the design of Rijndael shows a remarkable level of mathematical abstraction. The rich structure, and in particular the ease with which equivalent descriptions can be constructed, on the one hand has caused worries with some people fearing for its long-term security. On the other hand, the presence mathematical structure has certainly greatly facilitated the development of efficient implementations on constrained environments and environments where protection measures against side-channel attacks are required.

We expect that the continuation and generalization of this research will lead to better insights in the security of Rijndael and even more efficient implementations. Furthermore, we expect that many results will be applicable to other ciphers as well, leading to an increased understanding about the process of designing secure symmetric primitives.

References

1. Mehdi-Laurant Akkar and Christophe Giraud, "An implementation of DES and AES secure against some attacks," *CHES 2001*, LNCS 2162, Springer-Verlag, 2001, pp. 309–318.
2. Elad Barkan and Eli Biham, "In how many ways can you write Rijndael?", Advances in Cryptology — Asiacrypt 2002, LNCS 2051, Springer-Verlag, 2002, pp. 160–175.
3. Alex Biryukov, Christophe De Canniére, An Braeken and Bart Preneel, "A toolbox for cryptanalysis: linear and affine equivalence algorithms," *Advances in Cryptology — Eurocrypt 2003*, LNCS 2656, Springer-Verlag, 2003, pp. 33–50.
4. Johannes Blömer, Guajardo Merchan and Volker Krummel, "Provably secure masking of AES", *Selected Areas in Cryptography - SAC'04*, LNCS, Springer-Verlag, to appear.
5. Nicolas T. Courtois and Josef Pieprzyk, "Cryptanalysis of block ciphers with overdefined systems of equations," *Advances in Cryptology — Asiacrypt '02*, LNCS 2501, Springer-Verlag, 2003, pp. 267–287.
6. Marc Davio, Yvo Desmedt, Marc Fosséprez, René Govaerts, Jan Hulsbosch, Patrik Neutjens, Philippe Piret, Jean-Jacques Quisquater, Joos Vandewalle, and Pascal Wouters, "Analytical characteristics of the DES," *Advances in Cryptology — Crypto '83*, Plenum Press, 1984, pp. 171–202.
7. Niels Ferguson, Richard Schroeppel, and Doug Whiting, "A simple algebraic representation of Rijndael," *Selected Areas in Cryptography SAC01*, LNCS 2259, Springer-Verlag, 2001, pp. 103-111.
8. Joanne Fuller and William Millan, "On linear redundancy in S-boxes," *Fast Software Encryption '03*, LNCS 2887, Springer-Verlag, 2003, pp. 74–86.
9. Jovan Dj. Golic and Christophe Tymen, "Multiplicative masking and power analysis of AES," *CHES 2002*, LNCS 2535, Springer-Verlag, 2003, pp. 198–212.
10. Sean Murphy and Matthew J.B. Robshaw, "Essential algebraic structure within the AES", *Advances in Cryptology — Crypto 2002*, LNCS 2442, Springer-Verlag, 2002, pp. 17–38.
11. Elisabeth Oswald, Stefan Mangard and Norbert Pramstaller, "Secure and efficient masking of the AES: a mission impossible?," *Technical report IAIK-TR 2003/11/1*, available from http://eprint.iacr.org/2004/134.pdf.
12. Vincent Rijmen, "Efficient implementation of the Rijndael S-box," available from http://www.esat.kuleuven.ac.be/∼rijmen/rijndael/sbox.pdf, 2000.

13. Atri Rudra, Pradeep K. Dubey, Charanjit S. Jutla, Vijay Kumar, Josyula R. Rao, and Pankaj Rohatgi, "Efficient Rijndael encryption implementation with composite field arithmetic," *CHES 2001*, LNCS 2162, Springer-Verlag, 2001, pp. 171–184.
14. Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh, "A compact Rijndael hardware architecture with S-box optimization," *Advances in Cryptology, Asiacrypt 2001*, LNCS 2248, Springer-Verlag, 2001, pp. 239–254.
15. Adi Shamir, Jacques Patarin, Nicolas Courtois and Alexander Klimov, "Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations", *Advances in Cryptology — Eurocrypt 2000,* LNCS 1807, Springer-Verlag, 2000, pp. 392–407.
16. Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger, "An ASIC implementation of the AES S-boxes," *Topics in Cryptology — CT-RSA 2002*, LNCS 2271, Springer-Verlag, 2002, pp. 67–78.
17. Shee-You Wu, Shih-Chuan Lu, and Chi Sung Laih, "Design of AES Based on Dual Cipher and Composite Field," *Topics in Cryptology — CT-RSA 2004*, LNCS 2964, Springer-Verlag, 2004, pp. 25–38