Using Normal Bases for Compact Hardware Implementations of the AES S-box

Svetla Nikova[1] and Vincent Rijmen[1,2] and Martin Schläffer[2]

[1]Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
[2]Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria

martin.schlaeffer@iaik.tugraz.at

Abstract. The substitution box (S-box) of the Advanced Encryption Standard (AES) is based on the multiplicative inversion $s(x) = x^{-1}$ in GF(256) and followed by an affine transformation in GF(2). The S-box is the most expansive building block of any hardware implementation of the AES, and the multiplicative inversion is the most costly step of the S-box transformation. There exist many publications about hardware implementations of the S-box and the smallest known implementations are based on normal bases. In this paper, we introduce a new method to implement the multiplicative inversion over GF(256) based on normal bases that have not been considered before in the context of AES implementations.

Keywords: AES, S-box, hardware implementation, normal basis

1 Introduction

The first efficient hardware implementation of the multiplicative inversion in GF(256) has been proposed by Rijmen [9] and first imlemented by Rudra et al. [10] and Wolkerstorfer et al. [13]. They decompose the elements of GF(256) into polynomials of degree 2 over the subfield GF(16). In the next step, the elements of GF(16) are further decomposed into polynomials of degree 4 over GF(2). The resulting operations in GF(2) work on bit-level and can be implemented in hardware using simple gates.

Satoh et al. [11] and further Mentens et al. [6] use the different *tower field decomposition* in their implementation. They first start by decomposing the elements of GF(256) into polynomials over GF(16) as well. But then the elements of the field GF(16) are further decomposed into polynomials over the subfield GF(4) before implementing the final operations in GF(2). In all these approaches the field elements are represented by using polynomial bases. In contrast, Canright has been able to further reduce the size of the S-box computation by using normal bases at all levels of the tower field decomposition [3,2].

In this paper, we propose a new way to implement the inversion of the AES S-box. We use normal bases as in the approach of Canright but do not decompose

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 236–246, 2008.
The original publication is available at http://www.springerlink.com/content/145t6x7144361tn1

the elements of GF(16) into elements of GF(4). Table 1 illustrates the relation between the four mentioned approaches. We show that our implementation of the AES S-box can be at least as compact as the one proposed by Canright when counting the required number of gates. Which of the approaches is the best depends not only on the used hardware technology, but also on the application of the circuit and the resulting hardware criteria like size, timing, or power consumption [12]. Therefore, we cannot make an unambiguous ranking of the different implementations, but we are convinced that our alternative has its merits, already simply because it increases the available options of a hardware designer.

Table 1. Four approaches to decompose elements of GF(256) into elements of smaller subfields.

	Decomposition of field elements	
	GF(256) → GF(16) → GF(2)	GF(256) → GF(16) → GF(4) → GF(2)
Polynomial bases	Rijmen [9], Rudra et al. [10], Wolkerstorfer et al. [13]	Satoh et al. [11], Mentens el al. [6]
Normal bases	this paper	Canright [3,2]

The paper is organized as follows. In Section 2 we briefly recall some properties of normal bases which are relevant for the implementation of the AES S-box. Subsequently, in Section 3 we study extensively the different normal bases of GF(16) over GF(2) and discuss the impact of the choice of basis on the structure and complexity of the multiplicative inversion over GF(16). Additional hardware related considerations are made in Section 4 and we conclude in Section 5

2 Normal bases

In this section we briefly summarize some properties of normal bases over finite fields [5]. The finite field GF(2^{mp}) is isomorphic to a p-dimensional vector space over GF(2^m). This implies that it is possible to construct a *basis* for GF(2^{mp}). A basis consists of p elements $\beta_0, \beta_2, \ldots, \beta_{p-1} \in \mathrm{GF}(2^{mp})$ such that all elements of GF(2^{mp}) can be written as a linear combination of the elements β_j, with all coefficients elements of GF(2^m). As in all vector spaces, there are many different choices possible for the basis, and the choice of basis may influences the complexity to describe transformations on the vector space.

2.1 Construction

A *normal basis* is constructed by choosing an element $\theta \in \mathrm{GF}(2^{mp})$ and setting $\beta_j = \theta^{2^{mj}}$. Not all elements of GF(2^{mp}) result in a basis, but there exist always

some suitable elements. Let now

$$x = \sum_{j=0}^{p-1} c_j \theta^{2^{mj}}, \quad c_j \in \mathrm{GF}(2^m).$$

We raise both sides to the power 2^m. This operation is linear over $\mathrm{GF}(2^{mp})$ and corresponds to the identity transformation for all elements in $\mathrm{GF}(2^m)$. Then we obtain

$$x^{2^m} = \sum_{j=0}^{p-1} (c_j)^{2^m} (\theta^{2^{mj}})^{2^m} = \sum_{j=0}^{p-1} c_j \theta^{2^{m(j+1)}} = \sum_{j=0}^{p-1} c_{j-1} \theta^{2^{mj}}.$$

In words, this corresponds to the following property.

Property 1 ([5]). If the elements of the finite field $\mathrm{GF}(2^{mp})$ are represented by p-dimensional vectors over $\mathrm{GF}(2^m)$ using a normal basis, then raising an element to the power 2^m corresponds to rotating the coordinates of the element by one position.

Let now $m = 1$ and consider the inversion map used in the AES S-box. We denote this map by s. Then we have:

$$\begin{aligned} \mathrm{s}(0) &= 0, \\ \mathrm{s}(x) &= x^{-1}, \quad x \neq 0. \end{aligned}$$

Equivalently, we can write: $\mathrm{s}(x) = x^{254}$. Clearly, $\mathrm{s}(x^2) = x^{508} = (\mathrm{s}(x))^2$. Hence we obtain the following property.

Property 2. If the elements of the finite field $\mathrm{GF}(2^{mp})$ are represented in a normal basis, then the inversion map $\mathrm{s}(x)$ is rotation invariant:

$$\mathrm{rot}(\mathrm{s}(x)) = \mathrm{s}(\mathrm{rot}(x)).$$

For the remainder of this section, we set $p = 2$ and let v be an element of $\mathrm{GF}(2^{2m})$ such that $v + v^\ell = 1$ with $\ell = 2^m$. Then $\{v, v^\ell\}$ is a normal basis of $\mathrm{GF}(2^{2m})$ and the coordinate vectors consist of two elements of $\mathrm{GF}(2^m)$. We denote by g the trace of v over $\mathrm{GF}(2^m)$:

$$g = v^2 + v, \tag{1}$$

and $g \in \mathrm{GF}(2^m)$. We show now that the use of a normal basis leads to simple formulas for products and inverses of elements.

2.2 Multiplication

Let (a, b) and (c, d) be the coordinates of two elements of $\mathrm{GF}(2^{2m})$. The coordinates of the product are given by the following formula:

$$\begin{aligned} (e, f) = (a, b) \times (c, d) &\Leftrightarrow ev + fv^\ell = acv^2 + (ad + bc)v^{\ell+1} + bdv^{2\ell} \\ &\Leftrightarrow \begin{cases} e = (a + b)(c + d)g + ac \\ f = (a + b)(c + d)g + bd \end{cases} \end{aligned}$$

Here the element g is defined by (1).

2.3 Inversion

Let (a, b) be the coordinates of an element of GF(2^{2m}). The coordinates of the inverse element are given by the following formula:

$$\begin{aligned}(c,d) = (a,b)^{-1} &\Leftrightarrow 1 = (av + bv^{\ell})(cv + dv^{\ell}) \\ &\Leftrightarrow 1 = acv^2 + (ad + bc)(v^2 + v) + bd(v^2 + 1) \\ &\Leftrightarrow \begin{cases} c = ((a+b)^2 g + ab)^{-1} b \\ d = ((a+b)^2 g + ab)^{-1} a \end{cases} \end{aligned} \tag{2}$$

Again the element g is defined by (1).

3 Implementing the AES inverse using normal bases

Canright has investigated all 432 possible tower field decompositions using polynomial and normal bases in his work [3,2]. He has obtained the smallest hardware implementation of the S-box by using normal bases at all three levels. However, he did not consider the decomposition of the field GF(16) into the subfield GF(2) using normal bases. In this section we derive this decomposition and present formulas for the inversion in GF(16) using the additional normal bases.

3.1 Inversion in GF(256)

Assume a normal basis $\{v, v^{16}\}$. Equation (2) suggests the following 3-stage approach to implement the inverse [2,3].

Step 1: Compute the product ab and add the result to $(a + b)^2 g$. Computing the product is a nonlinear operation and the remaining operations are linear.

Step 2: Compute the multiplicative inverse over GF(16) of the result of Step 1.

Step 3: Multiply the result of Step 2 once with a, and once with b.

Note that Step 3 uses twice the same hardware. Hence, hardware can be saved by splitting Step 3 up into two steps using the same circuit. The computation of the multiplicative inverse over GF(16) can be computed by applying recursion, i.e. describing GF(16) as an extension field of GF(4) using the *tower field approach*. We have opted for a direct computation of the inverse over GF(16), as explained in the next section.

3.2 Normal bases in GF(16)

The field GF(16) can be described directly as an extension field of GF(2). This approach gives a 4-dimensional basis with coordinate elements from GF(2), i.e. $m = 1$ and $p = 4$. The field GF(16) counts exactly 8 solutions to the following equation:

$$\theta + \theta^2 + \theta^4 + \theta^8 = 1. \tag{3}$$

These 8 elements define 8 possible normal bases: $\{\theta, \theta^2, \theta^4, \theta^8\}$. Note that if θ_1 satisfies (3), then so do θ_1^2, θ_1^4 and θ_1^8, i.e. they define rotated versions of the same 4 base vectors. Thus the number of the normal bases reduces to two, as noted in [7].

It is called an *optimal normal base (ONB)* [7], because the complexity of the multiplication formula in this basis is minimal, i.e. equal to $2n - 1 = 7$ [4]. In the non-optimal normal basis (NB), the complexity of the multiplication formula equals 9 [8]. In those two cases multiplying $x = (x_0, x_1, x_2, x_3)$ with $y = (y_0, y_1, y_2, y_3)$ results in $z = (z_0, z_1, z_2, z_3)$, where

$$\begin{aligned} \text{NB:} \quad & z_3 = x_2y_3 + x_3y_2 + x_1y_3 + x_3y_1 + x_3y_0 + x_0y_3 + x_2y_2 + x_0y_1 + x_1y_0. \\ \text{ONB:} \quad & z_3 = x_3y_1 + x_0y_1 + x_0y_2 + x_1y_3 + x_1y_0 + x_2y_0 + x_2y_2. \end{aligned} \tag{4}$$

As noted by Paar [8] the multiplication in any normal basis is rotation symmetric, so the rest of the output bits z_0, z_1 and z_2 can be computed by rotating the input bits. Note that the multiplicative group of $GF(16)$ has order 15 and we know from group theory that this group is the direct product of two cyclic groups of order 3 and order 5. Remember that the intersection of these two subgroups is trivial.

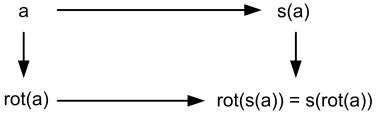
3.3 Inversion in GF(16)

We now study in detail how the choice of θ influences the complexity of the map $\mathrm{s}(x)$.

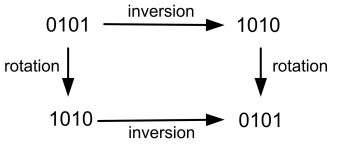
1. Equation (3) can be rewritten as

   $$(\theta + \theta^4)^2 + (\theta + \theta^4) = 1.$$

   It follows that for all θ satisfying (3), the elements $\theta + \theta^4$ and $\theta^2 + \theta^8$ are the roots of the polynomial $x^2 + x + 1$. The roots of this polynomial are exactly the two elements of order 3 in GF(16).
2. In a normal base, the zero element of GF(16) always has coordinates 0000, while the unit element always has 1111. Hence we always have $\mathrm{s}(0000) = (0000)$ and $\mathrm{s}(1111) = (1111)$.
3. It follows from Property 2 that x and $\mathrm{s}(x)$ must have coordinates with the same rotational symmetry. Hence, the inverse map must map elements with rotational symmetry to elements with the same symmetry. Hence we have the following commutative diagram:



   This implies that $\mathrm{s}(\{0101, 1010\}) = \{0101, 1010\}$. The inversion map has only two fixed points: the zero element and the unit element and it follows



that s(0101) = 1010 and s(1010) = 0101: We can conclude that 0101 and 1010 are the two elements of order 2. Together with the unit element, they form the cyclic subgroup of order 3 of the multiplicative group of $GF(16)$.

4. The remaining 12 elements of GF(16) can be partitioned into 3 sets with 4 elements each. The index of S_i denotes the number of ones in each representation:

$$S_1 = \{0001, 0010, 0100, 1000\}$$
$$S_2 = \{0011, 0110, 1100, 1001\}$$
$$S_3 = \{0111, 1110, 1101, 1011\}$$

Due to the rotational symmetry of s, this partitioning is consistent with s. If one element of S_i is mapped to an element of S_j, then all elements of S_i are mapped to elements of S_j, and since s is an involution, this also implies that all elements of S_j are mapped to elements of S_i.

5. Assume now that $s(S_i) = S_i$. Then the rotational symmetry of s implies:

$$\exists r, \forall v \in S_i : s(v) = \mathrm{rot}_r(v).$$

Since s is an involution we have

$$\mathrm{rot}_r(\mathrm{rot}_r(v)) = \mathrm{rot}_{2r}(v) \equiv v.$$

This implies that $2r = 0 \pmod 4$. Since we cannot have new fixed points for s, $r \neq 0$ we get $r = 2$. We require that $s(S_2) \neq S_2$ because otherwise, we would get s(0011) = (1100), and the corresponding element satisfies $x^{-1} + x = 1$. Since this would mean that it is a root of the polynomial $x^2 + x + 1$ we get a contradiction because the roots of this polynomial have coordinates (0101) and (1010). Hence, there are only two possibilities left: either $s(S_1) = S_1$ and $s(S_2) = S_3$, or $s(S_1) = S_2$ and $s(S_3) = S_3$. Further analysis shows that

$$s(S_1) = S_1 \Leftrightarrow s(0001) = (0100),$$
$$s(S_3) = S_3 \Leftrightarrow s(0111) = (1101),$$

because other choices lead to violations of Property 2. For each of these possibilities, there are 4 possibilities left for $s(0, 0, 1, 1)$ and this choice determines completely the action of s on the coordinate vectors. Table 2 lists the 8 remaining candidates for the inversion map in GF(16). The cases A-D correspond to the 4 choices for $S_1 = s(S_2)$ and the cases E-H to the 4 choices for $S_3 = s(S_2)$.

Table 2. 8 candidates for the inversion map in GF(16). The first 4 cases have $s(S_2) = S_1$, the last 4 have $s(S_2) = S_3$.

x	s(x)							
	A	B	C	D	E	F	G	H
0000	0000	0000	0000	0000	0000	0000	0000	0000
1111	1111	1111	1111	1111	1111	1111	1111	1111
0101	1010	1010	1010	1010	1010	1010	1010	1010
1010	0101	0101	0101	0101	0101	0101	0101	0101
0001	0011	0110	1100	1001	0100	0100	0100	0100
0010	0110	1100	1001	0011	1000	1000	1000	1000
0100	1100	1001	0011	0110	0001	0001	0001	0001
1000	1001	0011	0110	1100	0010	0010	0010	0010
0011	0001	1000	0100	0010	0111	1011	1101	1110
0110	0010	0001	1000	0100	1110	0111	1011	1101
1100	0100	0010	0001	1000	1101	1110	0111	1011
1001	1000	0100	0010	0001	1011	1101	1110	0111
0111	1101	1101	1101	1101	0011	0110	1100	1001
1011	1110	1110	1110	1110	1001	0011	0110	1100
1101	0111	0111	0111	0111	1100	1001	0011	0110
1110	1011	1011	1011	1011	0110	1100	1001	0011

The two choices $S_1 = s(S_2)$ and $S_3 = s(S_2)$ can be restated as $S_3 = s(S_3)$ and $S_1 = s(S_1)$. Hence, the choice is which elements will have order 5 and will form together with the unit element the cyclic subgroup of order 5 of the multiplicative group of $GF(16)$.
Recall that the two cyclic groups of order 3 and order 5 build up the multiplicative group of $GF(16)$. Thus the fact that the cyclic group of order 3 is fixed and for the cyclic group of order 5 we have 2 choices implies that we have two choices for the multiplicative group of $GF(16)$ this naturally corresponds to the fact that we have only two normal bases in $GF(16)$.

6. Finally, we use the fact that s must satisfy

$$s(xy) = s(x)s(y), \forall x, y$$

to eliminate all but two of the candidate maps. We obtain that only case A and case H are inversion maps, corresponding to the optimal, respectively the normal base, mentioned before.

Table 3 gives the truth tables for the Boolean functions that computes the rightmost bit f_0 of s(x) in the two cases:

$$(f_3, f_2, f_1, f_0) = s(x_3, x_2, x_1, x_0)$$

Due to the rotational symmetry of s, the other output bits f_3, f_2 and f_1 can be computed by rotating the input bits. The Algebraic Normal Form (ANF) of

Table 3. The truth tables for the Boolean functions computing the rightmost bit of s(x) in the two cases.

x	A (NB)	H (ONB)
0000	0	0
0001	1	0
0010	0	0
0011	1	0
0100	0	1
0101	0	0
0110	0	1
0111	1	1
1000	1	0
1001	0	1
1010	1	1
1011	0	0
1100	0	1
1101	1	0
1110	1	1
1111	1	1

each output bit f_0 is given by:

$$\begin{array}{ll} \text{NB:} & f_0 = x_0 + x_3 + x_0x_1 + x_1x_3 \qquad\quad + x_0x_1x_2 + x_0x_1x_3 + x_1x_2x_3 \\ \text{ONB:} & f_0 = x_1 \qquad + x_0x_3 + x_0x_2 + x_1x_3 + x_0x_1x_2 + x_0x_1x_3 + x_1x_2x_3 \end{array} \tag{5}$$

In the next section we show how the ANF can be simplified to reduce the number of operations and the resulting hardware size.

4 Hardware considerations

In this section we optimize the hardware implementation of the different normal bases regarding their size. We present the size of the inversion in GF(16) using the two normal bases and compare our results with the results of Canright.

4.1 Optimizing the implementation

In order to achieve minimal hardware implementations, the 4 formulae for the 4 output bits of the inversion in GF(16) need to be further optimized. We compute the four output bits in parallel and share intermediate terms between different functions. Hence, formulae which allow to share many terms, have an advantage.

Secondly, '+' operations (XOR) are usually very expensive in hardware implementations. For instance using the AMS $0.35\mu m$ technology [1], an XOR gate costs 2.33 times more than a NAND gate. Hence, the final implementation

formulae should contain as little '+' operations as possible. For instance, the inversion formulae using the optimal normal basis (5) can be rewritten as:

$$\begin{aligned}
f_0 &= \mathrm{NAND}(\mathrm{NOR}(\mathrm{NOR}(\mathrm{NAND}(x_0, \overline{x_2})), \mathrm{NAND}(x_3, \overline{x_1}),\\
&\quad \mathrm{NOR}(\mathrm{NOR}(\mathrm{NOR}(\overline{x_0}, x_3), x_1), \overline{x_2})), \mathrm{NAND}(x_1, \overline{x_3}))\\
f_1 &= \mathrm{NAND}(\mathrm{NOR}(\mathrm{NOR}(\mathrm{NAND}(x_3, \overline{x_1}), \mathrm{NAND}(x_2, \overline{x_0})),\\
&\quad \mathrm{NOR}(\mathrm{NOR}(\mathrm{NOR}(\overline{x_3}, x_2), x_0), \overline{x_1})), \mathrm{NAND}(x_0, \overline{x_2}))\\
f_2 &= \mathrm{NAND}(\mathrm{NOR}(\mathrm{NOR}(\mathrm{NAND}(x_2, \overline{x_0})), \mathrm{NAND}(x_1, \overline{x_3}),\\
&\quad \mathrm{NOR}(\mathrm{NOR}(\mathrm{NOR}(\overline{x_2}, x_1), x_3), \overline{x_0})), \mathrm{NAND}(x_3, \overline{x_1}))\\
f_3 &= \mathrm{NAND}(\mathrm{NOR}(\mathrm{NOR}(\mathrm{NAND}(x_1, \overline{x_3})), \mathrm{NAND}(x_0, \overline{x_2}),\\
&\quad \mathrm{NOR}(\mathrm{NOR}(\mathrm{NOR}(\overline{x_1}, x_0), x_2), \overline{x_3})), \mathrm{NAND}(x_2, \overline{x_0}))
\end{aligned} \tag{6}$$

These formulae use 16 NAND gates, 20 NOR gates, 20 NOT gates. When implementing the 4 output bits in parallel, the inverters $\overline{x_i}$ and the common terms $\mathrm{NAND}(x_i, \overline{x_{i+2}})$ can be shared between different output bits. Therefore, these 4 functions can finally be implemented using 20 NOR gates, 8 NAND gates and 4 NOT gates.

4.2 Counting gate equivalents

We refer to the size of the NAND gate by one gate equivalent (GE). In the AMS $0.35\mu m$ standard cell library [1] an XNOR gate corresponds to 2GE, an XOR gate to 2.33GE, an inverter (INV) to 0.65GE and a NOR gate to 1GE. These values give a total of 30.7GE for our best GF(16) inversion. This compares favorably to the best GF(16) inversion circuit reported by Canright, which uses 9 XOR and 10 NAND gates, corresponding to 31.0GE [2]. Referring to personal communication with Satoh, Canright equates XOR and XNOR gates to 1.75GE, and inverters to 0.75GE. Using these values, our best GF(16) inversion costs 31.0GE, but the cost of the best Canright implementation is reduced to 25.8GE.

However, standard cell libraries provide additional operations other than the basic Boolean operations INV, NAND, NOR, XNOR and XOR. For example, there are extended operations with more than one input which can be used to improve the 3-input NOR terms of (6). Additionally, the AMS $0.35\mu m$ library provides specialized standard cells like AND-OR-INVERT (AOI) which compute $Q = A.B+C+1$, or OR-AND-INVERT (OAI) which compute $Q = (A+B).C+1$. Both can be implemented using only $1.33GE$ [1] and have far less size than the XOR or XNOR gates. Using these additional cells, we have been able to improve the GF(16) inversion of Canright to $26.7GE$ and the inversion based on the optimal normal base formulae to 22.7GE. Table 4 gives an overview of these results.

5 Conclusion

In this paper, we discussed alternative constructions for the AES S-box using normal bases for GF(16) over GF(2). We worked out example implementations showing that our normal bases can compete with the results of Canright. Of

Table 4. Equivalent gate costs for the implementation of several GF(16) inversions.

Design	Canright's GE [3]	basic standard cells [1]	all standard cells [1]
Canright	25.8	31.0	26.7
NB	34.3	34.0	24.7
ONB	31.0	30.7	22.7

course, the final size of the S-box depends on the size of the multiplication in GF(16) and on the complexity of the basis transformations as well. As shown in Section 4.2, also the target technology influences the final count on the implementation cost. Therefore, our normal bases should at least be considered when designing small AES S-box implementations.

We did not check all possible cases, since the result can only be a specialized implementation for a single target technology. Further, the best basis does not only depend on the hardware size but on other optimization constraints such as low power, timing and throughput as well. However, using our normal bases new promising alternatives for hardware designers of compact AES S-boxes are available.

References

1. Austria Microsystems. Standard Cell Library $0.35\mu m$ CMOS (C35). Available online at `http://asic.austriamicrosystems.com/databooks/c35/databook_c35_33`.
2. David Canright. A very compact Rijndael S-box, May 2005. Available online at `http://web.nps.navy.mil/~dcanrig/pub`.
3. David Canright. A Very Compact S-Box for AES. In Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *LNCS*, pages 441–455. Springer, 2005.
4. Certicom. F_{2^4} with Optimal Normal Basis Representation. Available online at `http://www.certicom.com/index.php?action=ecc_tutorial,math9_1`.
5. Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications*. Cambridge University Press, New York, NY, USA, 1986.
6. Nele Mentens, Lejla Batina, Bart Preneel, and Ingrid Verbauwhede. A Systematic Evaluation of Compact Hardware Implementations for the Rijndael S-Box. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *LNCS*, pages 323–333. Springer, 2005.
7. R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson. Optimal Normal Bases in $GF(p^n)$. In *Discrete Appl. Math.*, volume 22, pages 149–161. Elsevier Science Publishers B. V., 1989.
8. Christof Paar. *Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields*. PhD thesis, Institute for Experimental Mathematics, University of Essen, 1994.
9. Vincent Rijmen. Efficient Implementation of the Rijndael S-box, 2000. Available online at `www.iaik.tugraz.at/RESEARCH/krypto/AES/old/~rijmen/rijndael/sbox.pdf`.

10. Atri Rudra, Pradeep K. Dubey, Charanjit S. Jutla, Vijay Kumar, Josyula R. Rao, and Pankaj Rohatgi. Efficient rijndael encryption implementation with composite field arithmetic. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES*, volume 2162 of *LNCS*, pages 171–184. Springer, 2001.
11. Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh. A Compact Rijndael Hardware Architecture with S-Box Optimization. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *LNCS*, pages 239–254. Springer, 2001.
12. Stefan Tillich, Martin Feldhofer, Johann Großschädl, and Thomas Popp. Area, Delay, and Power Characteristics of Standard-Cell Implementations of the AES S-Box. *Journal of Signal Processing Systems*, 50(2):251–261, January 2008.
13. Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger. An ASIC Implementation of the AES SBoxes. In Bart Preneel, editor, *CT-RSA*, volume 2271 of *LNCS*, pages 67–78. Springer, 2002.