A Systematic Evaluation of Compact Hardware Implementations for the Rijndael S-Box

Nele Mentens*, Lejla Batina*, Bart Preneel, and Ingrid Verbauwhede

K.U. Leuven ESAT/COSIC, Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee, Belgium
{Nele.Mentens,Lejla.Batina,Bart.Preneel,Ingrid.Verbauwhede}
@esat.kuleuven.ac.be

Abstract. This work proposes a compact implementation of the AES S-box using composite field arithmetic in $\mathrm{GF}(((2^2)^2)^2)$. It describes a systematic exploration of different choices for the irreducible polynomials that generate the extension fields. It also examines all possible transformation matrices that map one field representation to another. We show that the area of Satoh's S-box, which is the most compact to our knowledge, is at least 5% away from an optimal solution. We implemented this optimal solution and Satoh's design using a 0.18 μm standard cell library.

Keywords: AES, S-box, inversion in $GF(2^n)$, composite fields, smart card implementation

1 Introduction

After an open competition ending in 2000, the National Institute for Standard and Technology (NIST) has selected the Rijndael block cipher as the new Advanced Encryption Standard (AES) [1]. The AES algorithm, designed by Joan Daemen and Vincent Rijmen, has an SPN (Substitution Permutation Network) structure. Its use is mandatory for the encryption of sensitive but unclassified US government information; in 2003 the US government has announced that it can also be used for encrypting secret and top secret information (for the last category key lengths of at least 192 bits need to be used). AES is currently replacing the Data Encryption Standard as the worldwide standard algorithm.

Since 2000, extensive research has been performed on AES implementations. In this article we are focusing on compact hardware implementations for mobile devices and smart cards, but our results can also be applied in high-speed pipelined implementations for network security and e-commerce applications. Note that the best known software implementations achieve about 15 cycles/byte on a modern PC.

* Lejla Batina and Nele Mentens are funded by research grants of the Katholieke Universiteit Leuven, Belgium. This work was supported by FWO project (G.0450.04).

A.J. Menezes (Ed.): CT-RSA 2005, LNCS 3376, pp. 323–333, 2005.

Design challenges for AES mainly lie in exploring all the options for the S-box design. The most common strategy to reduce the gate complexity consists of exploiting composite field arithmetic. By following that approach one still has several options to represent the finite field $GF(2^8)$. In this paper we represent $GF(2^8)$ as the composite field $GF(((2^2)^2)^2)$. In this way, we reduce the arithmetic in $GF(2^8)$ to operations in smaller fields. There are many ways to represent $GF(2^8)$ as $GF(((2^2)^2)^2)$. Choices need to be made with respect to the irreducible polynomials that are used to create the extension fields and with respect to the transformation matrices that map elements from one representation to the other. Exploring these two degrees of freedom we optimize the S-box of Satoh *et al.* [17], which is to our knowledge the most compact implementation today. Another area efficient implementation is the one of Wolkerstorfer *et al.* [20]. More precisely, according to Daemen and Rijmen [6], the number of kgates for the implementations of [17] and [20] are 5.4 and 5.7 respectively.

The remainder of this paper is organized as follows. In Sect. 2 some details on the AES algorithm are discussed. Section 3 lists previous work on hardware implementations of Rijndael. In Sect. 4 we explain our approach to minimize the area of the S-box and compare our new solution with the S-box of Satoh. Section 5 concludes the paper and outlines future work.

2 The AES Algorithm

Rijndael has a variable block and key length which can be 128, 192 or 256 bits; the AES standard includes only block lengths of 128 bits. In this implementation we focus on the 128-bit key version of AES which has 10 rounds. In this case, each round and the initial stage require a 128-bit round key. In total 10 sets of round keys are generated from the secret key by using the S-box. The input data is arranged as a table *i.e.,* a matrix of bytes. Figure 1 outlines the basic structure of the algorithm. The round transformation consists of four different transformations: `ByteSub`, `ShiftRow`, `MixColumn` and `AddRoundKey`. They are performed in this order with the exception of the final round which is slightly different. All transformations are based on byte-oriented arithmetic and `AddRoundKey` is a bitwise XOR operation. The transformations operate on the intermediate result, which is called the `State`.

The `ByteSub` transformation is a non-linear byte substitution also called S-box (substitution table). It operates on bytes independently. The S-box is invertible and consists of the following two transformations:

1. Inversion in the $GF(2^8)$ field, modulo the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$.
2. Affine transformation defined with: $Y = AX^{-1} + b$, where A is a 8×8 fixed matrix and b is a 8×1 vector-matrix.

The schematic of the complete AES algorithm is shown in Fig. 1. Further details on the AES algorithm can be found in [4, 5].

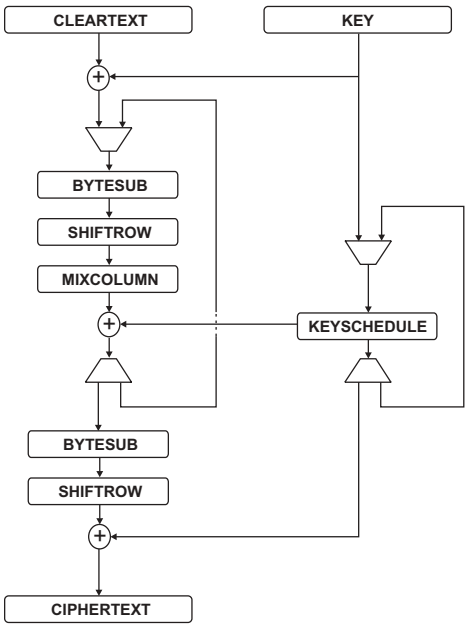

Fig. 1. Schematic of the AES encryption algorithm.

3 Previous Work

Many hardware architectures for Rijndael were proposed as either ASIC [11, 12, 19] or FPGA implementations [2, 3, 7, 8, 10, 14, 18]. Most of the known implementations, particularly the early ones, were quite simple and not small enough as they did not exploit composite field arithmetic. Among those who tried to produce a really small circuit we mention the work of Satoh *et al.* [17] and Wolkerstorfer *et al.* [21]. In [16] the use of the composite field $GF((2^4)^2)$ was also proposed but no hardware implementation was presented.

Satoh *et al.* introduced a new composite field $GF(((2^2)^2)^2)$ which resulted in an optimized S-box. More precisely, their S-box requires less than 1/4 of the size of one using a look-up table. This resulted in a compact AES implementation with a gate complexity of 5.4 kgates. To our knowledge this is the most compact architecture so far. Wolkerstorfer *et al.* used arithmetic in $GF((2^4)^2)$ to achieve an implementation with a gate count comparable to Satoh *et al.* (5.7 kgates). In the solution of Wolkerstorfer *et al.* inversion of two-term polynomials $a_h x + a_l \in GF((2^4)^2)$ involves only operations in $GF(2^4)$, which are easily computed using combinational logic. Macchetti and Bertoni [13] have described an ASIC implementation for the same composite field $GF((2^4)^2)$, but with a representation as given in [16]. The work of Chodowiec and Gaj [3] also offers a compact design that is targeting low-cost embedded applications. They used dedicated Block RAMs for the implementation of the S-boxes. Recently, the work

of Wu *et al.* [22] gives an area and delay reduction of 1/6 and 1/4 respectively compared to [21]. The proposed approach uses dual AES in combination with a composite field.

Here we use the composite field $GF(((2^2)^2)^2)$, which was only explored by Satoh *et al.* [17]. By a systematic exploration of all options we show that Satoh's S-box is at least 5% away from an optimal solution. The implementation of this optimal solution and the approach we use to explore all design possibilities is explained in the next section.

4 Hardware Implementation

In this section we examine the S-box of Satoh *et al.* [17] and we try to optimize it for area. Section 4.1 describes the approach we used to optimize Satoh's S-box. Section 4.2 presents our new S-box. Implementation results and comparison with Satoh's S-box are given in Sect. 4.3.

4.1 Theoretical Approach to Optimize the S-Box

Let us view $GF(2^{2m})$ as a field extension of degree 2 over $GF(2^m)$. The field $GF(2^{2m})$ is generated as an extension field of $GF(2^m)$ using an irreducible polynomial say $f(x) = x^2 + \alpha x + \beta$, where $\alpha, \beta \in GF(2^m)$. Then we have $GF(2^{2m}) = GF(2^m)[\omega]$, where ω is a root of $f(x)$ and $GF(2^{2m})$ can be viewed as a two-dimensional vector space over $GF(2^m)$. Hence, an arbitrary element $\Delta \in GF(2^{2m})$ can be written as $\Delta = \delta_1\omega + \delta_0$, where $\delta_1, \delta_0 \in GF(2^m)$. We want to calculate the inverse of Δ i.e. Δ^{-1} such that $\Delta \cdot \Delta^{-1} \equiv 1 \bmod f(x)$.

The multiplicative inverse of $\Delta \in GF(2^{2m})$ can therefore be computed as:

$$\Delta^{-1} = (\delta_1\omega + \delta_0)^{-1} = \delta_1(\delta_1^2\beta + \delta_1\delta_0\alpha + \delta_0^2)^{-1}\omega + (\delta_0 + \delta_1\alpha)(\delta_1^2\beta + \delta_1\delta_0\alpha + \delta_0^2)^{-1} \quad (1)$$

This equation consists of operations which can be performed in the subfield $GF(2^m)$ [9].

Equation (1) can be used recursively to find the inverse in $GF(((2^2)^2)^2)$. $GF(((2^2)^2)^2)$ is a field extension of degree 2 over $GF((2^2)^2)$ constructed using the irreducible polynomial $P(x) = x^2 + p_1x + p_0$, where $p_1, p_0 \in GF((2^2)^2)$. Let us call a root of P also x. $GF((2^2)^2)$ is a field extension of degree 2 over $GF(2^2)$ using the irreducible polynomial $Q(y) = y^2 + q_1y + q_0$, with y a root of the polynomial and $q_1, q_0 \in GF(2^2)$. $GF(2^2)$ is a field extension of degree 2 over $GF(2)$ using the irreducible polynomial $R(z) = z^2 + z + 1$, with root z.

In Satoh *et al.* the following choices are made for the coefficients of the irreducible polynomials: $p_1 = 1 = \{0001\}_2$, $p_0 = \lambda = (z+1)y = \{1100\}_2$, $q_1 = 1 = \{01\}_2$ and $q_0 = \phi = z = \{10\}_2$. Equation (1) is implemented as

$$\begin{aligned}
&\Delta_2 = \delta_{21}x + \delta_{20} \in GF(((2^2)^2)^2) : \\
&\quad \Delta_2^{-1} = (\delta_{21}x + (\delta_{21} + \delta_{20})) \cdot (\lambda\delta_{21}^2 + (\delta_{21} + \delta_{20})\delta_{20})^{-1}\,, \qquad (2)\\
&\Delta_1 = \delta_{11}y + \delta_{10} \in GF((2^2)^2) : \\
&\quad \Delta_1^{-1} = (\delta_{11}y + (\delta_{11} + \delta_{10})) \cdot (\phi\delta_{11}^2 + (\delta_{11} + \delta_{10})\delta_{10})^{-1}\,.
\end{aligned}$$

Inversion in $GF(2^2)$ requires only one addition:

$$\Delta_0 = \delta_{01}z + \delta_{00} \in GF(2^2) : \Delta_0^{-1} = \delta_{01}z + (\delta_{01} + \delta_{00})\,. \qquad (3)$$

The inversion in $GF(2^8)$ is finally decomposed into operations in $GF(2^2)$. Therefore a transformation is needed to transform a representation in $GF(2^8)$ to a representation in $GF(((2^2)^2)^2)$. In [15], Paar explains how a matrix can be created to perform this transformation. Different choices for the irreducible polynomials $P(x)$ and $Q(y)$ lead to different transformation matrices. For every combination of $P(x)$ and $Q(y)$ there are 8 possibilities for the transformation matrix. For hardware implementations, the most area efficient transformation matrix is the one that has the least '1' entries, because this number determines the XOR gate count for the transformation. After performing the inversion using the $GF(((2^2)^2)^2)$ representation we need to go back to the $GF(2^8)$ representation using the inverse of the transformation matrix. This matrix can be combined with the affine transformation matrix at the end of the S-box.

We stick to the choice of Satoh *et al.* to make $p_1 = q_1 = 1$ and $q_0 = \phi = z$. Based on (2) and the fact that the transformation matrix depends on $P(x)$ and $Q(y)$, we conclude that the hardware complexity of the circuit depends on the choice of $p_0 = \lambda$. That is why we explored all values of λ to determine the most compact solution for the S-box. There are 8 choices for λ. The two elements that determine the hardware complexity of the circuit are:

- the number of gates in the constant multiplication with λ in $GF((2^2)^2)$,
- the number of '1' entries in the transformation matrix and in the combination of the inverse transformation matrix with the affine transformation matrix.

For every λ, Table 1 gives the number of 2-input XORs for the constant multiplication and the total number of '1' entries for every option of the transformation matrix. Out of 8 possible transformation matrices for every λ, the one that gives the least total number of '1' entries is given in the last column.

The values in the table are depicted in Fig. 2.

From Table 1 and Fig. 2 we conclude that the solution of Satoh *et al.*, which has $\lambda = (z+1)y$ uses the most area efficient constant multiplication. The transformation matrix they chose gives a total number of '1' entries equal to 61. Their implementation can be made more efficient by choosing the most compact transformation matrix which leads to a total number of '1' entries equal to 59. But the most optimal solution ("best case") would be to change the implementation even more by taking $\lambda = zy$. The constant multiplication requires only 1 XOR more than Satoh's constant multiplication, but the total number of '1' entries in the matrices is reduced by 5. On the other hand, the design with a maximized

Table 1. Comparison of the hardware complexity when using different polynomials $x^2 + x + \lambda$ for the generation of $GF(((2^2)^2)^2)$.

λ	$\{\lambda\}_2$	# XORs in $\star\lambda$	# '1' in matrices	min. # '1'
$(z+1)y+z$	$\{1110\}_2$	4	57, 58, 59, 60, 62, 63, 63, 67	57
zy	$\{1000\}_2$	4	54, 57, 59, 59, 61, 62, 63, 66	54
$zy+(z+1)$	$\{1011\}_2$	5	59, 63, 63, 64, 65, 65, 67, 71	59
$zy+z$	$\{1010\}_2$	4	56, 59, 59, 61, 61, 66, 70, 71	56
$(z+1)y$	$\{1100\}_2$	3	59, 61, 62, 63, 63, 64, 66, 69	59
$(z+1)y+1$	$\{1101\}_2$	4	60, 61, 62, 62, 63, 64, 65, 68	60
$zy+1$	$\{1001\}_2$	5	55, 59, 59, 61, 62, 63, 65, 67	55
$(z+1)y+(z+1)$	$\{1111\}_2$	3	59, 59, 61, 62, 64, 64, 66, 72	59

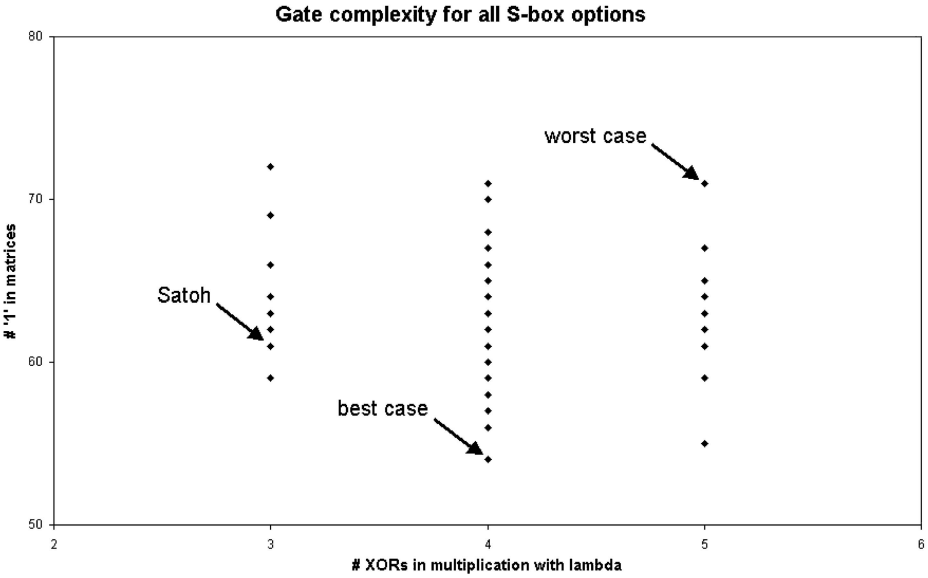

Fig. 2. Graphical representation of the values in Table 1. The arrows point at Satoh's S-box, the S-box with minimized gate count ("best case") and the S-box with maximized gate count ("worst case").

gate count ("worst case") uses 5 XOR gates for the constant multiplication and 71 '1' entries in the matrices.

The implementation of the new optimized S-box is explained in the next section. Implementation results for Satoh's S-box, the "best case" and the "worst case" are given in Sect. 4.3.

4.2 Implementation of the New Optimized S-Box

Figure 3 shows the structure of the S-box implementation.

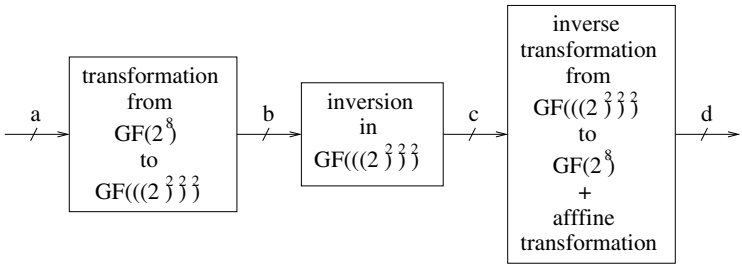

Fig. 3. Structure of the S-box implementation.

The transformation used here is

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1\,0\,1\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,1\,1 \\ 0\,0\,1\,0\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0\,0\,1\,1 \\ 0\,1\,0\,0\,0\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,1\,0 \\ 0\,1\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,0\,0\,1\,0\,1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix},$$

where $a_i, b_i \in GF(2)$ are the coefficients of $a \in GF(2^8)$, $b \in GF(((2^2)^2)^2)$ respectively.
This results in the following combination of the inverse transformation with the affine transformation

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix} = \begin{bmatrix} 1\,0\,1\,0\,1\,1\,0\,0 \\ 1\,1\,1\,0\,0\,0\,0\,0 \\ 1\,1\,0\,0\,0\,0\,0\,0 \\ 1\,0\,1\,0\,1\,1\,1\,0 \\ 1\,0\,0\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,0\,0 \\ 0\,0\,0\,0\,1\,0\,1\,1 \\ 0\,0\,1\,0\,1\,0\,1\,0 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix},$$

where $c_i, d_i \in GF(2)$ are the coefficients of $c \in GF(((2^2)^2)^2)$, $d \in GF(2^8)$ respectively.

The total number of '1' entries in both 8×8 matrices is equal to 54. The addition of the column vector in the affine transformation is fixed and hence does not have to be considered for optimization. The number of '1' entries in the matrices in Satoh's implementation is equal to 61. Implementing the matrices in a straightforward way, the number of XORs would be equal to the number of '1' entries minus the number of rows in the matrices. This would lead to an XOR gate count of 38 and 45 for our and Satoh's S-box respectively, which results in a

reduction of 7 XOR gates. By finding common terms in the XOR equations and exploring some rools of logic it is possible to reduce the number of XOR gates. We leave this to a synthesis tool and give results on the final implementations in Sect. 4.3.

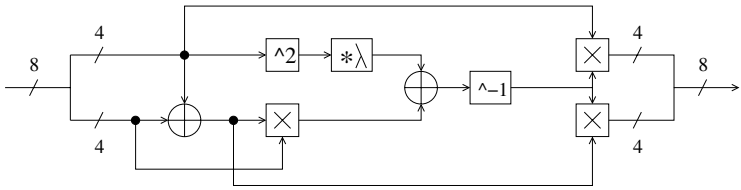

Fig. 4. Structure of the S-box implementation.

The inversion in $GF(((2^2)^2)^2)$ has many levels of hierarchy. At the highest level the architecture looks the same as Satoh's architecture (see Fig. 4). At the next level of hierarchy, the only difference with Satoh's design is the implementation of the constant multiplication with λ. Figure 5 gives the gate-level implementation of both Satoh's (top) and our (bottom) constant multiplication. As can be seen, our constant multiplication requires one extra XOR gate compared to Satoh's implementation.

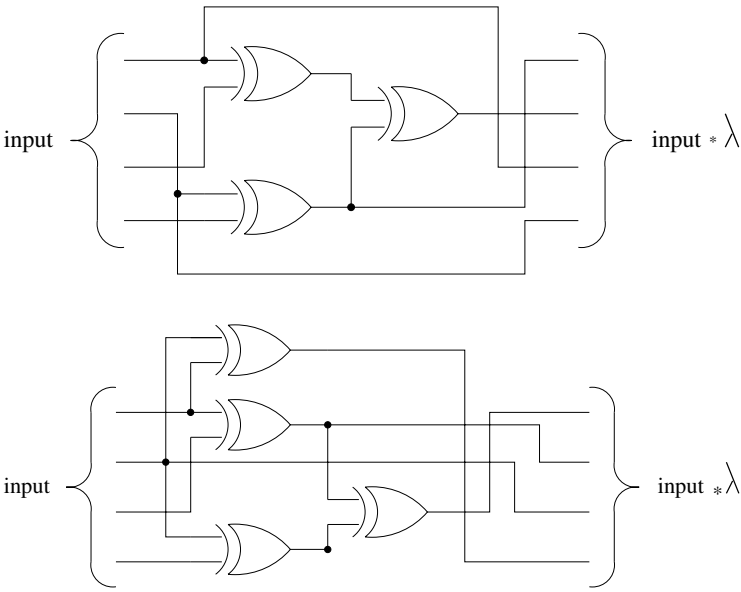

Fig. 5. Gate-level implementation of Satoh's (top) and our (bottom) constant multiplication with λ.

We can summarize this section by stating that, in a straightforward implementation, our S-box would require 6 XOR gates less than Satoh's S-box (7 less for the implementation of the matrices and 1 extra for the multiplication with λ). The implementation results of both approaches after synthesis are compared in the next section. To show the upper bound of the gate complexity, we also implemented the "worst case" S-box and included it in the comparison.

4.3 Implementation Results and Comparison

We implemented both our new optimized S-box and Satoh's S-box using a 0.18 μm CMOS standard cell library. To show that the area is sensitive to the choice of the polynomials and the transformation matrix we also implemented the "worst case" S-box (with maximized gate count). All three implementations are synthesized with a rather slow target delay of 10 ns. Table 2 gives the number of gates (in equivalent number of 2-input NAND gates) for all designs.

Table 2. Area comparison of our S-box with minimized/maximized gate count and Satoh's S-box (in equivalent number of 2-input NAND gates).

	min. area	Satoh	max. area
number of gates	272	286	297

Satoh *et al.* implemented their S-box using a 0.11 μm CMOS standard cell library, which resulted in 294 gates with a delay of 3.69 ns. This corresponds to 286 gates in 0.18 μm CMOS with a maximum delay of 10 ns. The table shows our new S-box has a 5% area reduction compared to Satoh's S-box. This is equivalent to the expected reduction of 6 XOR gates. The table also shows that a bad choice for the polynomials and the transformation matrix can lead to an area enlargement of 9%.

5 Conclusions and Future Work

We explored various options for low gate counts in the design of the AES S-box. We used the architecture of Satoh as a reference and we showed that it is 5% away from an optimal solution. Furthermore, we proved that the "worst case" S-box leads to a 9% area increase. We optimized Satoh's S-box by choosing the irreducible polynomial and transformation matrix that lead to the most compact solution.

There exists a possibility that a more compact S-box can be achieved by choosing an irreducible polynomial $P(x) = x^2 + p_1 x + p_0$ with $p_1 \neq 1$. The high level architecture of Satoh does not stay the same in this case. On the other hand, in this work we only considered the gate complexity of the S-box while encryption is done. The same strategy can be applied to the decryption operation as well.

Acknowledgements

For this work we used the Magma software package. We thank Jasper Scholten from COSIC, Katholieke Universiteit Leuven for his help.

References

1. FIPS Pub. 197: Specification for the AES, Nov. 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.
2. M. Alam, W. Badawy, and G. Jullien. A novel pipelined threads architecture for aes encryption algorithm. In M. Schulte, S. Bhattacharyya, N. Burgess, and R. Schreiber, editors, *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP)*, pages 296–302, San Jose, CA, USA, July 17-19 2002. IEEE Computer Society Press.
3. P. Chodowiec and K. Gaj. Very compact FPGA implementation of the AES algorithm. In C. Walter, Ç. K. Koç, and C. Paar, editors, *Proceedings of 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2779 in Lecture Notes in Computer Science, pages 319–333, Cologne, Germany, September 7-10 2003. Springer-Verlag.
4. J. Daemen and V. Rijmen. AES proposal: Rijndael, September 2001. http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf.
5. J. Daemen and V. Rijmen. *The design of Rijndael: AES–The Advanced Encryption Standard*. Springer-Verlag, 2002.
6. J. Daemen and V. Rijmen. Security of a wide trail design. In A. Menezes and P. Sarkar, editors, *Proceedings of Third International Conference on Cryptology in India*, volume 2551 of *LNCS*, pages 1–11, Hyderabad, India, December 16-18 2002. Springer-Verlag. Invited talk.
7. V. Fischer and M. Drutarovský. Two methods of Rijndael implementation in reconfigurable hardware. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, page 77–92, Paris, France, May 13-16 2001. Springer-Verlag.
8. K. Gaj and P. Chodowiec. Fast implementation and fair comparison of the final candidates for advanced encryption standard using field programmable gate arrays. In D. Naccache, editor, *Proceedings of RSA Security Conference: Topics in Cryptology - CT-RSA*, number 2020 in Lecture Notes in Computer Science, San Francisco, CA, USA, April 8-12 2001. Springer-Verlag.
9. J. Guajardo and C. Paar. Efficient algorithms for elliptic curve cryptosystems. In B. S. Kaliski Jr., editor, *Advances in Cryptology: Proceedings of CRYPTO'97*, number 1294 in Lecture Notes in Computer Science, pages 342–356. Springer-Verlag, 1997.
10. K. Jdrvinen, M. Tommiska, and J. Skyttd. A fully pipelined memoryless 17.8 Gbps AES-128 encyptor. In *Proceedings of the 11th ACM International Symposium on Field-Programmable Gate Arrays (FPGA)*, Monterey, CA, USA, February 23-25 2003.
11. H. Kuo and I. Verbauwhede. Architectural optimization for a 1.82Gbits/sec VLSI implementation of the AES rijndael algorithm. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, pages 51–64, Paris, France, May 13-16 2001. Springer-Verlag.

12. C.-C. Lu and S.-Y. Tseng. Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter. In M. Schulte, S. Bhattacharyya, N. Burgess, and R. Schreiber, editors, *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP)*, pages 277–285, San Jose, CA, USA, July 17-19 2002. IEEE Computer Society Press.
13. M. Macchetti and G. Bertoni. Hardware implementation of the Rijndael Sbox: A case study. *ST Journal of system research*, (0):84–91, 2002.
14. M. McLoone and J.V McCanny. High performance single-chip FPGA Rijndael algorithm implementations. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, pages 65–76, Paris, France, May 13-16 2001. Springer-Verlag.
15. C. Paar. *Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields*. PhD thesis, Institute for Experimental Mathematics, University of Essen, Germany, 1994.
16. A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi. Efficient Rijndael encryption implementation with composite field arithmetic. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptograpic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, pages 171–184, Paris, France, May 14-16 2001. Springer-Verlag.
17. A. Satoh, S. Morioka, K. Takano, and S. Munetoh. A compact Rijndael hardware architecture with S-Box optimization. In C. Boyd, editor, *Proceedings of Advances in Cryptology - ASIACRYPT: 7th International Conference on the Theory and Application of Cryptology and Information Security*, number 2248 in Lecture Notes in Computer Science, pages 239–254, Gold Coast, Australia, December 2001. Springer-Verlag.
18. F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. Efficient implementation of rijndael encryption in reconfigurable hardware: Improvements and design tradeoffs. In C. Walter, Ç. K. Koç, and C. Paar, editors, *Proceedings of 5th International Workshop on Cryptograpic Hardware and Embedded Systems (CHES)*, number 2779 in Lecture Notes in Computer Science, pages 334–350, Cologne, Germany, September 7-10 2003. Springer-Verlag.
19. I. Verbauwhede, P. Schaumont, and H. Kuo. Design and performance testing of a 2.29-Gb/s Rijndael processor. *IEEE Journal of Solid-State Circuits*, 38(3):569–572, March 2003.
20. J. Wolkerstorfer. Dual-field arithmetic unit for $GF(p)$ and $GF(2^m)$. In B. S. Kaliski Jr., Ç. Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2523 in Lecture Notes in Computer Science, pages 500–514, Redwood Shores, CA, USA, August 13-15 2002. Springer-Verlag.
21. J. Wolkerstorfer, E. Oswald, and M. Lamberger. An ASIC implementation of the AES S-Boxes. In B. Preneel, editor, *Proceedings of the RSA Conference - Topics in Cryptography (CT-RSA)*, number 2271 in Lecture Notes in Computer Science, pages 67–78, San Jose, USA, February 18-22 2002. Springer-Verlag.
22. S.-Y. Wu, S.-C. Lu, and C. S. Laih. Design of AES based on dual cipher and composite field. In T. Okamoto, editor, *Proceedings of RSA Cryptographers' Track*, number 2964 in Lecture Notes in Computer Science, pages 25–38, San Fransisco, USA, February 23-27 2004. Springer-Verlag.