AN FPGA IMPLEMENTATION OF RIJNDAEL: TRADE-OFFS FOR SIDE-CHANNEL SECURITY

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

* {*Nele.Mentens,Lejla.Batina,Bart.Preneel,Ingrid.Verbauwhede*}
@esat.kuleuven.ac.be
Kasteelpark Arenberg 10, 3001 Leuven, Belgium

Abstract: This work proposes a complete and side-channel proof solution for an FPGA implementation of AES. An unsecured implementation is extended to a secured version by using a masking algorithm. Our solution is implemented as an FPGA prototype, but in the future it can be easily used in a crypto-coprocessor on a smartcard.

Keywords: AES, FPGA implementation, composite fields, power analysis attacks, countermeasures, masking techniques

1. INTRODUCTION

The National Institute for Standard and Technology (NIST) has selected the Rijndael algorithm as the new Advanced Encryption Algorithm (AES) in 2000 (*FIPS Pub. 197: Specification for the AES* 2001). The AES algorithm is a block cipher with an SPN (Substitution Permutation Network) structure designed by Joan Daemen and Vincent Rijmen. The use of AES is required 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 old DES (Data Encryption Standard) algorithm and its variants (3DES, DESX etc.); it is expected that it will become soon the worldwide de facto standard block cipher for data encryption.

Since 2000, extensive research has been performed on AES implementations in software and hardware. Very efficient software implementations exist (around 15 cycles/byte) which are suitable for applications on PCs. Hardware implementations are especially important to facilitate high-speed implementations for network security applications. At the other end of the spectrum, there is a need for compact hardware implementations for smart cards and other hand-held devices. In these environments, speed, power consumption and resistance against side-channel attacks are important. There is a clear need for implementations that offer a good trade-off in this design space.

In this paper we present an FPGA implementation that has been improved to power analysis resistance. Namely, we have implemented the method of Akkar *et al.* (Akkar and Giraud 2001). Our secured architecture features an increase in gate area of less than 20%. The proposed secured architecture presents an interesting option for smart cards and other hand-held devices where a compact and side-channel secure solution is essential.

The remainder of this paper is organized as follows. In Section 2 some details on the AES algorithm are discussed. Section 3 lists previous work on hardware implementations of Rijndael. Section 4 describes a composite field based AES

implementation. In Section 5 the security with respect to power analysis attacks is discussed. Section 6 gives an improvement of our implementation to a side-channel analysis resistant design. Section 7 presents all results and compares our unsecured and secured implementations. Section 8 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 version of 128-bit key version of AES which has 10 rounds. Each round and the initial stage require in this case a 128-bit round key. In total 11 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. The round transformation consists of four different transformations: `ByteSub`, `ShiftRow`, `MixColumn` and `AddRoundKey`. They are performed in this order with the exception of 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), operating on bytes independently. The S-box is invertible and is constructed by the composition 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 8x8 fixed matrix and b is a 8x1 vector-matrix.

Further details on the AES algorithm can be found in (Daemen and Rijmen 2001, Daemen and Rijmen 2002).

3. PREVIOUS WORK

Many hardware architectures for Rijndael were proposed as either ASIC (Kuo and Verbauwhede 2001, Lu and Tseng 2002, Verbauwhede *et al.* 2003) or FPGA implementations (Alam *et al.* 2002, Chodowiec and Gaj 2003, Fischer and Drutarovský 2001, Gaj and Chodowiec 2001, McLoone and McCanny 2001, Standaert *et al.* 2003). 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.* (Satoh *et al.* 2001) and Wolkerstorfer et al. (Wolkerstorfer *et al.* 2002). In (Rudra *et al.* 2001) the use of the composite field $GF((2^4)^2)$ was also proposed but no hardware implementation was given.

Here we use the composite field $GF((2^4)^2)$, which was also the case for the work of Wolkerstorfer et al. (Wolkerstorfer *et al.* 2002) but with another representation. Also, we have implemented the masking as proposed by Akkar et al. (Akkar and Giraud 2001) in order to have a compact solution resistant against side-channel attacks.

4. COMPOSITE FIELD IMPLEMENTATION

When discussing the efficiency of a hardware implementation of AES, an important role is played by the way `MixColumn` is implemented. However, we found the impact of the S-box design more substantial as inversion is the most expensive operation in hardware. Therefore, the key to a compact AES implementation is the reduction of the circuit size of the S-box. As already observed by many researchers, there exist different ways to implement the S-box. We make our division with respect to the field in which the inversion is computed and then related to the way in which the inversion is done. The inversion is computed either in the main field *i.e.* in $GF(2^8)$ or in some of the subfields ($GF(2^2)$ or $GF(2^4)$). Here we chose for inversion in $GF(2^4)$ in the form of a table look-up. This choice was important in order to facilitate our secured version *i.e.,* a masked S-box implementation which is resistant to a first-order DPA attack.

4.1 The Field $GF((2^4)^2)$

From finite field theory it is known that there exists a field $GF(p^k)$ of size p^k, for any prime p and for any positive integer k. Moreover, every finite field has this form. Let $GF(q)$ be a finite field of characteristic p (that is the smallest positive integer c such that $ce = 0$, where e is the unit element). Then $GF(q)$ can be considered as a k-dimensional vector-space over $GF(p)$. In this case we call $GF(q) = GF(p^k)$ an extension field of the field $GF(p)$. The special case where $k = mn$ an extension field $GF(p^k) = GF((p^n)^m)$ is sometimes called a composite field (Paar 1994).

As defined in (Paar 1994) two pairs $GF(2^4)$, $Q(y)$ and $GF((2^4)^2)$, $P(x)$ form a composite field if the irreducible polynomials $Q(y)$ and $P(x)$ are used to construct $GF(2^4)$ from $GF(2)$ and $GF((2^4)^2)$ from $GF(2^4)$ respectively. In this case there exists an isomorphism δ between the fields $GF(2^8)$ and $GF((2^4)^2)$, so $\delta : GF(2^8) \rightarrow GF((2^4)^2)$. The polynomials Q and P are chosen as follows: $Q(y) = y^4 + y + 1$ and $P(x) = x^2 + x + \lambda$, where $\lambda \in GF(2^4)$. Here, we chose $\lambda = \omega^{14} = (1001)$, where ω is a generator in $GF(2^4)$. In this case an isomorphism δ is determined by an 8x8 matrix $\mathbf{T}$

i.e., we have $H(x) = \mathbf{T} \cdot x$. More precisely, for an element $x \in GF(2^8)$:

$$x' = x^{-1}, \delta(x') = \mathbf{T}x' = \delta(x^{-1}) = (\mathbf{T}x)^{-1} .$$

Hence the matrices $\mathbf{T}$ and $\mathbf{T}^{-1}$ allow us to convert between representations.

The calculation of the inverse is then performed in the composite field $GF((2^4)^2)$. For $A(x) = a_1 x + a_0$, where $a_1, a_0 \in GF(2^4)$, the inverse A^{-1} is calculated as:

$$A^{-1} = a_1({a_1}^2\lambda + a_1 a_0 + {a_0}^2)^{-1}x + (a_0 + a_1)({a_1}^2\lambda + a_1 a_0 + {a_0}^2)^{-1}$$

4.2 Details of the Implementation

4.2.1. Implementation of AddRoundKey, MixColumn and ShiftRow. The `AddRoundKey`, operation is an addition of the `State` and the key. In hardware this is a simple bitwise XOR. The number of XOR gates needed is 128. The implementation of the `MixColumn` operation uses the fact that $03 = 02 + 01$. A multiplication with 02 is a shift to the left followed by a reduction with x^4+1. The `ShiftRow` operation shifts the elements in the `State`. This is just a connection of wires, which is costless in hardware.

4.2.2. Implementation of ByteSub. The structure of our `ByteSub` operation is shown in Fig. 1. The hardware blocks in this figure are explained in the remainder of this section.

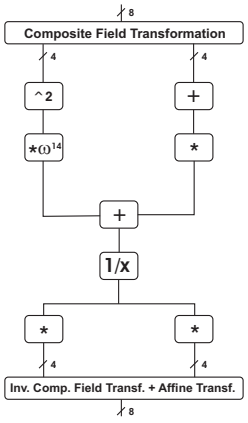

Fig. 1. Structure of the ByteSub implementation

An element in $GF(2^8)$ can be represented by elements in $GF(2^4)$. In (Rudra *et al.* 2001) x^4+x+1 is shown to be the most area efficient polynomial to define the field $GF(2^4)$. The generator ω of this field is (0000)0010. The 14th power of this element, $\omega^{14} = (0000)1001$, is used to define the irreducible polynomial of the upper field: $x^2 + x + \omega^{14}$. Out of 8 possible isomorphisms α is chosen as the most area efficient (Macchetti and Bertoni 2002, Rudra *et al.* 2001). This way, the transformation matrix is given by

$$T = \begin{bmatrix} 1&0&1&0&0&0&0&0 \\ 1&1&0&1&0&0&1&0 \\ 0&0&0&0&1&1&0&0 \\ 1&0&1&0&0&0&1&0 \\ 0&0&0&1&0&1&1&0 \\ 0&1&1&1&0&1&0&0 \\ 0&1&0&0&1&0&0&0 \\ 0&1&1&1&1&0&1&1 \end{bmatrix} .$$

This transformation needs 26 XOR-gates.

The inverse transformation can be obtained by calculating the inverse of T modulo 2. It can be combined with the affine transformation.

The multiplicative inverse in $GF(2^8)$ can be calculated as

$$B(x) = b_1 x + b_0 = \frac{a_1 x + (a_1 + a_0)}{a_0(a_1 + a_0) + a_1^2\omega^{14}} , \quad (1)$$

where $B(x) \in GF(2^8)$ and $a_1, a_0, b_1, b_0 \in GF(2^4)$. The hardware blocks needed for this operation are 1 inversion, 3 multiplications, 2 additions, 1 squaring and 1 multiplication with ω^{14}. To implement the inversion in $GF(2^4)$ a look-up table is used.

An addition in $GF(2^4)$ is a bitwise XOR using 4 XOR-gates. The `ByteSub` operation uses a standard multiplier with Mastrovito optimizations (Mastrovito 1991). Squaring in $GF(2^4)$ consists of 2 XOR-gates. The multiplication with ω^{14} needs one XOR-gate.

4.2.3. Implementation of KeySchedule Not many optimizations are possible for the `KeySchedule` operation, hence the `KeySchedule` is implemented in a straight-forward manner. This results in 136 XORs, 4 `ByteSub` operations and 1 free rotation.

4.2.4. Implementation of the controller A controller Finite State Machine (FSM) combines all the previously described operations to obtain the complete AES encryption. The FSM waits in the IDLE state for the START signal to come. The FSM can return to the IDLE state at any time when the RESET signal is set. After the START signal the plaintext and the key are read from the input registers. Next, the `AddRoundKey` operation is performed with the input key. This initialization takes 2 clock cycles. After that, the regular encryption round is performed 10 times.

One encryption round needs the following operations: 16 `ByteSub`, 4 `MixColumn`, 1 `AddRoundKey`, 1 `ShiftRow` and 1 `KeySchedule` block(s). To obtain a more compact implementation only 4 `ByteSub` blocks are used. In this way, the data are processed in 8 instead of 2 clock cycles. During the first clock cycle of each of the 4 cycles the output of the `ByteSub` operation is calculated, which is written in a register one clock cycle later. The other 3 operations are performed in 1 clock cycle. The 4 `ByteSub` blocks are used in the 7th clock cycle to perform the first step in the `KeySchedule` operation. During the 8th clock cycle the key for

the next round is calculated. The total number of clock cycles for one encryption round is 9.

Finally, the last round is performed. This round uses the same `AddRoundKey`, `ByteSub` and `ShiftRow` blocks as the regular rounds. The final round is executed in 9 clock cycles. The encryption ends with writing the ciphertext to the output register (1 clock cycle).

5. RESISTANCE AGAINST SIDE-CHANNEL ATTACKS

In this section we address side-channel security *i.e.,* resistance to power analysis (Kocher *et al.* 1999) and electromagnetic analysis attacks (EMA) (Gandolfi *et al.* 2001, Quisquater and Samyde 2001). These types of attacks, together with fault-analysis based attacks (Aumüller *et al.* 2002, Boneh *et al.* 1997, Joye *et al.* 1999), timing (Hachez *et al.* 1999, Kocher 1996) and other physical attacks such as probing attacks (Anderson and Kuhn 1997) are a major concern especially for wireless applications.

One of the first to realize the possibilities of "masking" as an efficient countermeasure was T. Messerges (Messerges 2000). But simple additive masking as XOR does not work for Rijndael as `ByteSub` is not a completely linear transformation. Therefore Akkar *et al.* proposed to use two masks, an additive and a multiplicative mask. This ensures that secret data that pass through `ByteSub` are always protected with a fresh mask that is a combination of two masks. The details of this method are explained in the following section. This paper demonstrates that the idea of Akkar et al. can be used in a practical hardware solution. The need of using two masks instead of one as proposed by Trichina et al. in (Trichina *et al.* 2002) has been discussed in (Akkar and Goubin 2004).

6. SIDE-CHANNEL RESISTANT AES IMPLEMENTATION

The masking applied in this implementation is based on (Akkar and Giraud 2001, Chari *et al.* 1999*a*, Chari *et al.* 1999*b*, Coron and Goubin 2000, Golić and Tymen 2002). This method, the Transformed Masking Method, is stated as being safe but practically infeasible because of the big area overhead. This section proves it is possible to implement the Transformed Masking Method in hardware with a limited area overhead.

6.1 Data Masking

In the masking method a random number called the *mask* is XOR-ed to the plaintext. The masked data are encrypted and afterwards the mask is removed to restore the ciphertext. To ensure security the data must remain masked throughout the whole encryption and the mask must be data-independent.

The hardest part to implement the masking method is the non-linear part of the implementation, the `ByteSub` block. Inside this block the Boolean additive mask X has to be replaced temporarily by a multiplicative mask Y. This Y is generated by a random number generator. The Transformed Masking Method protects a cryptographic circuit against first-order DPA attacks by hiding the actual values of the processed data throughout the whole implementation. The Transformed Masking Method does not protect the circuit against second order attacks, but increases the level of resistance against higher order attacks (Kocher *et al.* 1999, Messerges 2000).

6.2 Implementation of the Masking

As explained in the previous paragraph, the hardest part to apply the masking is the `ByteSub` block. As shown in Fig. 2 some conversions from additive to multiplicative masks are needed. The implementation of the Transformed Masking Method in the `ByteSub` block requires 6 of these conversions. This way, the extra cost of the masking will be 12 multipliers and 6 adders. The multipliers are standard multipliers with Mastrovito optimizations (Akkar and Giraud 2001) and the adders are bitwise XORs.

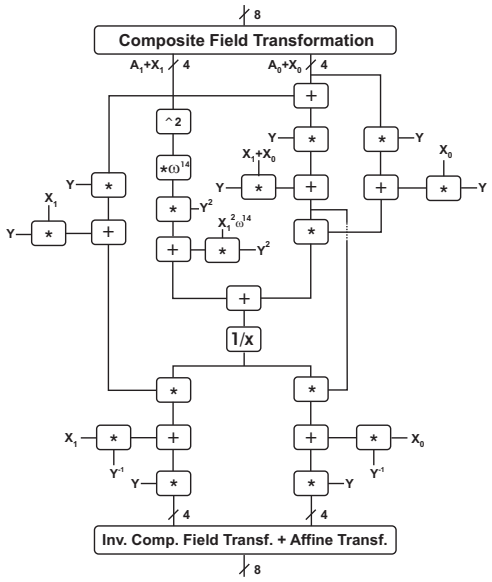

Fig. 2. Structure of the secure ByteSub implementation

7. RESULTS AND COMPARISON

We implemented our unsecured and secured solutions on a Xilinx XCV800-4-HQ240 FPGA. In this section we compare both implementations. Table 1 gives the result of this comparison.

Since the stress is on designing a compact solution that can fit on a smart card, the throughput is of less importance because all discussed implementations exceed the speed requirements for smart card implementations. The area overhead

Table 1. Comparison of the unsecured and the secured implementation

Impl.	unsecured	secured
Frequency (MHz)	33	23
Throughput (Mbit/s)	41	29
Number of CLBs	908	1113
Number of clock cycles	102	102

of the secured implementation compared to the unsecured version is less than 20%.

8. CONCLUSIONS AND FUTURE WORK

We designed a compact FPGA implementation of the AES algorithm. A second implementation extends the first one with some security measures against DPA attacks using the Transformed Masking Technique. The next step will be to make an ASIC implementation of our secured AES solution.

REFERENCES

Akkar, M.-L. and C. Giraud (2001). An implementation of DES and AES, secure against some attacks. In: *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (Ç. K. Koç, D. Naccache and C. Paar, Eds.). number 2162 In: *Lecture Notes in Computer Science*. Springer-Verlag. Paris, France. pp. 309–318.

Akkar, M.-L. and R. Bévan L. Goubin (2004). Two power analysis attacks against one-mask methods. In: *Proceedings of Fast Software Encryption (FSE)* (Bimal Roy and Willi Meier, Eds.). Lecture Notes in Computer Science. Springer-Verlag. New Delhi, India.

Alam, M., W. Badawy and G. Jullien (2002). A novel pipelined threads architecture for aes encryption algorithm. In: *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP)* (M. Schulte, S. Bhattacharyya, N. Burgess and R. Schreiber, Eds.). IEEE Computer Society Press. San Jose, CA, USA. pp. 296–302.

Anderson, R. and M. Kuhn (1997). Low cost attacks on tamper resistant devices. In: *Proceedings of 5th International Workshop on Security Protocols* (B. Christianson, B. Crispo, T. M. A. Lomas and M. Roe, Eds.). number 1361 In: *Lecture Notes in Computer Science*. Springer-Verlag. Paris, France. pp. 125–136.

Aumüller, C., P. Bier, W. Fischer, P. Hofreiter and J.-P. Seifert (2002). Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In: *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (B. S. Kaliski Jr., Ç. K. Koç and C. Paar, Eds.). number 2523 In: *Lecture Notes in Computer Science*. Springer-Verlag. Redwood Shores, CA, USA. pp. 260–275.

Boneh, D., R. A. DeMillo and R. J. Lipton (1997). On the importance of checking cryptographic protocols for faults (extended abstract). In: *Advances in Cryptology: Proceedings of EUROCRYPT'97* (W. Fumy, Ed.). number 1233 In: *Lecture Notes in Computer Science*. Springer-Verlag. Konstanz, Germany. pp. 37–51.

Chari, S., C. S. Jutla, J. R. Rao and P. Rohatgi (1999*a*). A cautionary note regarding evaluation of AES candidates on smart cards. In: *Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference*. Rome, Italy.

Chari, S., C. S. Jutla, J. R. Rao and P. Rohatgi (1999*b*). Towards sound approaches to counteract power-analysis attacks. In: *Advances in Cryptology: Proceedings of CRYPTO'99* (M. Wiener, Ed.). number 1666 In: *Lecture Notes in Computer Science*. Springer-Verlag. Santa Barbara, CA, USA. pp. 398–412.

Chodowiec, P. and K. Gaj (2003). Very compact FPGA implementation of the AES algorithm. In: *Proceedings of 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (C. Walter, Ç. K. Koç and C. Paar, Eds.). number 2779 In: *Lecture Notes in Computer Science*. Springer-Verlag. Cologne, Germany. p. 319333.

Coron, J.-S. and L. Goubin (2000). On boolean and arithmetic masking against differential power analysis. In: *Proceedings of 2nd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (Ç. K. Koç and C. Paar, Eds.). number 1965 In: *Lecture Notes in Computer Science*. Springer-Verlag. Worcester, Massachusetts, USA. pp. 231–237.

Daemen, J. and V. Rijmen (2001). AES proposal: Rijndael.

Daemen, J. and V. Rijmen (2002). *The design of Rijndael: AES–The Advanced Encryption Standard*. Springer-Verlag.

FIPS Pub. 197: Specification for the AES (2001).

Fischer, V. and M. Drutarovský (2001). Two methods of Rijndael implementation in reconfigurable hardware. In: *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (Ç. K. Koç, D. Naccache and C. Paar, Eds.). number 2162 In: *Lecture Notes in Computer Science*. Springer-Verlag. Paris, France. p. 7792.

Gaj, K. and P. Chodowiec (2001). Fast implementation and fair comparison of the final candidates for advanced encryption standard using field programmable gate arrays. In: *Proceedings of RSA Security Conference: Topics in Cryptology - CT-RSA* (D. Naccache, Ed.). number 2020 In: *Lecture Notes in Computer Science*. Springer-Verlag. San Francisco, CA, USA.

Gandolfi, K., C. Mourtel and F. Olivier (2001). Electromagnetic analysis: Concrete results. In: *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (Ç. K. Koç, D. Naccache and C. Paar, Eds.). number 2162 In: *Lecture Notes in Computer Science*. Springer-Verlag. Paris, France. pp. 255–265.

Golić, J. D. and C. Tymen (2002). Multiplicative masking and power anaylsis of AES. In: *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (B. S. Kaliski Jr., Ç. K. Koç and C. Paar, Eds.). number 2535 In: *Lecture Notes in Computer Science*. Springer-Verlag. Redwood Shores, CA, USA. pp. 198–212.

Hachez, G., F. Koeune and J.-J. Quisquater (1999). Timing attack: what can be achieved by a powerful adversary?. In: *Proceedings of the 20th symposium on Information Theory in the Benelux* (A. Barbé, E. C. van der Meulen and P. Vanroose, Eds.). pp. 63–70.

Joye, M., A. K. Lenstra and J.-J. Quisquater (1999). Chinese remaindering based cryptosystem in the presence of faults. *Journal of Cryptology* **4**(12), 241–245.

Kocher, P. (1996). Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In: *Advances in Cryptology: Proceedings of CRYPTO'96* (N. Koblitz, Ed.). number 1109 In: *Lecture Notes in Computer Science*. Springer-Verlag. Santa Barbara, CA, USA. pp. 104–113.

Kocher, P., J. Jaffe and B. Jun (1999). Differential power analysis. In: *Advances in Cryptology: Proceedings of CRYPTO'99* (M. Wiener, Ed.). number 1666 In: *Lecture Notes in Computer Science*. Springer-Verlag. Santa Barbara, CA, USA. pp. 388–397.

Kuo, H. and I. Verbauwhede (2001). Architectural optimization for a 1.82Gbits/sec VLSI implementation of the AES rijndael algorithm. In: *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (Ç. K. Koç, D. Naccache and C. Paar, Eds.). number 2162 In: *Lecture Notes in Computer Science*. Springer-Verlag. Paris, France. p. 5164.

Lu, C.-C. and S.-Y. Tseng (2002). Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter. In: *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP)* (M. Schulte, S. Bhattacharyya, N. Burgess and R. Schreiber, Eds.). IEEE Computer Society Press. San Jose, CA, USA. pp. 277–285.

Macchetti, M. and G. Bertoni (2002). Hardware implementation of the Rijndael Sbox: A case study. *ST Journal of system research* (0), 84–91.

Mastrovito, E. D. (1991). VLSI Architectures for Computations in Galois Fields. PhD thesis. Department of Electrical Engineering, Linkøping University, S-581 83 Linkøping, Sweden.

McLoone, M. and J.V McCanny (2001). High performance single-chip FPGA Rijndael algorithm implementations. In: *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (Ç. K. Koç, D. Naccache and C. Paar, Eds.). number 2162 In: *Lecture Notes in Computer Science*. Springer-Verlag. Paris, France. p. 6576.

Messerges, T. S. (2000). Securing the AES finalists against power analysis attacks. In: *Proceedings of 7th International Workshop on Fast Software Encryption Workshop (FSE)* (B. Schneier, Ed.). number 1978 In: *Lecture Notes in Computer Science*. Springer-Verlag. New York, NY, USA.

Paar, C. (1994). Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields. PhD thesis. Institute for Experimental Mathematics, University of Essen, Germany.

Quisquater, J.-J. and D. Samyde (2001). Electromagnetic analysis (ema): Measures and couter-measures for smard cards. In: *Smart Card Programming and Security (E-smart 2001)* (I. Attali and T. P. Jensen, Eds.). Vol. 2140 of *Lecture Notes in Computer Science*. Springer-Verlag. pp. 200–210.

Rudra, A., P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao and P. Rohatgi (2001). Efficient Rijndael encryption implementation with composite field arithmetic. In: *Proceedings of 3rd International Workshop on Cryptograpic Hardware and Embedded Systems (CHES)* (Ç. K. Koç, D. Naccache and C. Paar, Eds.). number 2162 In: *Lecture Notes in Computer Science*. Springer-Verlag. Paris, France. p. 171 184.

Satoh, A., S. Morioka, K. Takano and S. Munetoh (2001). A compact Rijndael hardware architecture with s-box optimization. In: *Proceedings of Advances in Cryptology - ASIACRYPT: 7th International Conference on the Theory and Application of Cryptology and Information Security* (C. Boyd, Ed.). number 2248 In: *Lecture Notes in Computer Science*. Springer-Verlag. Gold Coast, Australia. pp. 239–254.

Standaert, F.-X., G. Rouvroy, J.-J. Quisquater and J.-D. Legat (2003). Efficient implementation of rijndael encryption in reconfigurable hardware: Improvements and design tradeofs. In: *Proceedings of 5th International Workshop on Cryptograpic Hardware and Embedded Systems (CHES)* (C. Walter, Ç. K. Koç and C. Paar, Eds.). number 2779 In: *Lecture Notes in Computer Science*. Springer-Verlag. Cologne, Germany. pp. 334–350.

Trichina, E., D. De Seta and L. Germani (2002). Simplified adaptive multiplicative masking for AES. In: *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (B. S. Kaliski Jr., Ç. K. Koç and C. Paar, Eds.). number 2535 In: *Lecture Notes in Computer Science*. Springer-Verlag. Redwood Shores, CA, USA. pp. 187–197.

Verbauwhede, I., P. Schaumont and H. Kuo (2003). Design and performance testing of a 2.29-Gb/s Rijndael processor. *IEEE Journal of Solid-State Circuits* **38**(3), 569–572.

Wolkerstorfer, J., E. Oswald and M. Lamberger (2002). An ASIC implementation of the AES s-boxes. In: *Proceedings of the RSA Conference - Topics in Cryptography (CT-RSA)* (B. Preneel, Ed.). number 2271 In: *Lecture Notes in Computer Science*. Springer-Verlag. San Jose, USA. pp. 67–78.