A Low-Power and Secure AES Coprocessor Protecting Differential Power Attack with 0.18μm CMOS Smart Card Technology

Mi-Jung Noh
SoC R&D Center
Samsung Electronics Co.
Korea
+82-31-209-9780
mjnoh@samsung.com

Yoo-Jin Baek
SoC R&D Center
Samsung Electronics Co.
Korea
+82-31-209-9780
yoojin.baek@samsung.com

Kyoung-Moon Ahn
SoC R&D Center
Samsung Electronics Co.
Korea
+82-31-209-9499
kyoungmoon.ahn@samsung.com

Abstract

This paper presents an AES coprocessor with 25K gates which is secure against DPA. The overall current consumption is about 1mA at 5MHz clock frequency using Samsung smart card 0.18μm CMOS technology and the maximum throughput of 12.8Mbps has been achieved at 16MHz clock frequency. In comparison with previous designs, the proposed architecture achieves a considerable increase in security level by adopting a new masking method. It also significantly reduces the computing power and the area.

keyword : Advanced Encryption Standard, Differential Power Analysis, Masking Method, Hardware Architecture, VLSI

1 Introduction

Since AES (Advanced Encryption Standard)[1] was chosen by NIST (National Institute of Standards and Technology) as a new block encryption algorithm standard in November 2001, a number of AES hardware architectures have been proposed ([7], [8], [9], [10]). However previous architectures mainly focused on the throughput or the area and did not pay much attention to implementing securely against known cryptographic attacks. For example, the side-channel attacks such as SPA (Simple Power Analysis) and DPA (Differential Power Analysis)[2] can easily derive secret information inside a device, using the statistical relationship between the secret key and the power consumption curves derived during the cryptographic operations. Therefore, deploying countermeasures against these attacks have grown as one of the architectural key features in applications such as smart cards and multi-media cards.

This paper proposes an AES coprocessor which has some noticeable features. First, it adopts a new masking structure which gives a countermeasure against DPA. Another feature is that it maintains a small area, while it uses additional logic gates for the masking. The final feature is the low-power architecture suitable for card applications.

The paper is organized as follows. Section 2 gives a brief overview for DPA, the masking method and the AES algorithm. In Section 3, a new masking algorithm and the corresponding hardware-implementation are described. The implementation results of the proposed architecture are summarized and compared with previous AES hardware-implementations in Section 4. Concluding remarks can be found in Section 5.

2 Preliminaries

2.1 Differential Power Analysis and Masking Method

Differential Power Analysis (DPA), which was first introduced by Kocher et al. in [2], works by parti-

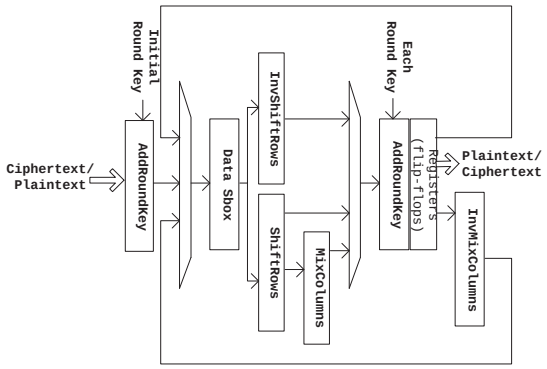

Figure 1: The Basic Datapath of AES

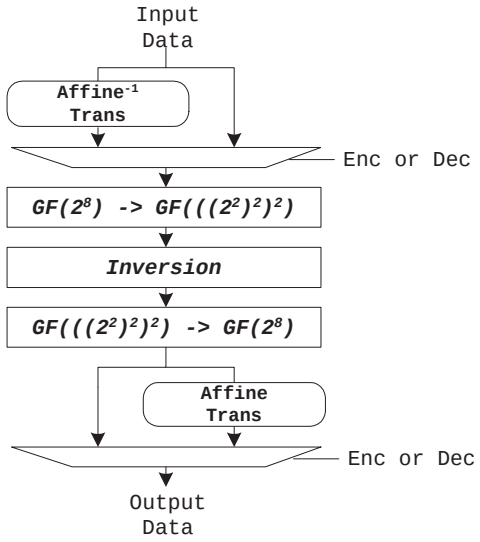

Figure 2: The Computation Sequence of S-box

tioning the set of manipulated plaintexts into two subsets based on some key-dependent bits. The idea is that for the rightly guessed subkey, power consumption and the data computed with the subkey are statistically related to each other. On the other hand, for the wrongly guessed subkey, the relation between those data are likely to look random. Hence, one hopes to detect the statistical relationship and obtain information about the secret key.

Various countermeasures against DPA have been proposed. For example, introducing noise power, filtering power signal and randomizing execution orders are examples of hardware countermeasures against DPA. Also, data randomization techniques including the masking method[4] are well-known algorithmic countermeasures against DPA.

The masking method works as follows: first, scramble the plaintext with a random value (the masking phase), and then perform the cryptographic operation with the scrambled data, the subkey and the random value. Finally, we descramble the resulting data (the unmasking phase) to get the required ciphertext. The masking and the unmasking phase usually use the eXclusive-OR operation.

Now, suppose that a function $f : \{0,1\}^k \rightarrow \{0,1\}^n$ be given. Then, applying the masking method to f means that we should construct an efficiently computable (probabilistic) function $F : \{0,1\}^{2k} \rightarrow \{0,1\}^{2n}$ with the following properties: given any input $(x' = x \oplus r, r) \in \{0,1\}^k \times \{0,1\}^k$, F must output $F(x', r) = (f(x) \oplus s, s)$ for some random $s \in \{0,1\}^n$. For example, if f is a linear function, we can set $F(x', r) = (f(x'), f(r))$, because

$$\begin{aligned} f(x') \oplus f(r) &= f(x \oplus r) \oplus f(r) \\ &= f(x) \oplus f(r) \oplus f(r) \\ &= f(x). \end{aligned} \quad (1)$$

Therefore, if we apply the same hardware block for f twice, the masking method can be easily implemented in the case of linear functions. However, some special idea is required in the nonlinear functions. In this paper, we introduce a new masking architecture for the finite field multiplication, which is the unique nonlinear function in AES. Also, we propose an AES coprocessor realizing the new masking method. The masking method needs additional logic gates and power consumption but we minimize its overhead.

2.2 AES Overview

AES is a symmetric block cipher of SPN (Substitution Permutation Network) type. It uses keys of 128-, 192- and 256-bit to encrypt data blocks of 128-bit length. Also, it uses a round function that is composed of four different byte-oriented transformations: byte substitution using a substitution table (S-box), shifting rows of the state by different offsets, mixing the data within each column of the

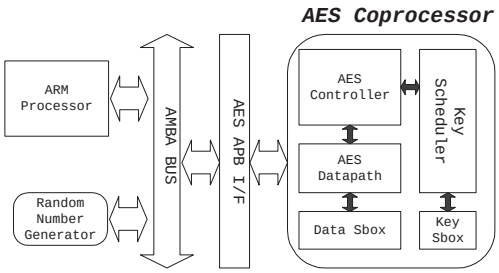

Figure 3: AES Top Configuration

state, and adding a round key to the state. The round function is executed ten/twelve/fourteen times repeatedly, depending on the key size. The basic datapath architecture of AES is shown in Figure 1.

Among the functions used in AES, S-box is the only nonlinear function and is defined to be the multiplicative inversion in the finite field GF(2^8), followed by an affine transformation. The field GF(2^8) has the defining irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$ over GF(2).

One approach for designing an S-box circuit is to construct a multiplicative inversion circuit and an affine transformation circuit independently, and then to connect these two circuits in serial. Various methods for constructing compact inversion circuits over GF(2^8) have been introduced. In particular, the composite field inversion [5] is effective for GF(2^8) and it can be used to create compact AES implementations [6][7]. The idea of composite field inversion is as follows: first, map a element of GF(2^8) to that of GF($(((2^2)^2)^2$) using an appropriate group isomorphism and then apply the inversion operation in GF($(((2^2)^2)^2$). Finally, we transform the result to an element in GF(2^8) using the inverse isomorphism. These processes are described in Figure 2.

In this paper, we propose a new S-box architecture with an additional masking component, using composite field arithmetic. Even though using composite field arithmetic in an S-box implementation is not a new idea[6][7], our architecture gives a special contribution in that we added a new masking logic in it.

Round	10/12/14
Mode	ECB
Data	128 bit
Key	128/192/256 bit
Attack Proof	Yes
Clk/Round	16

Table 1: Our AES Processor Configuration

3 Our AES Hardware Architecture

3.1 Overall Structure

The basic configuration of our AES module is shown in Figure 3. It is composed of 6 modules, say, a controller, an on-the-fly key scheduler, a datapath module, a key S-box module, a data S-box module, and an interface module. It is designed to operate in the typical AMBA bus configuration. Therefore, it requires three inputs from a main processor (for example, ARM) to encrypt or decrypt, that is, an input data (plaintext or ciphertext), a random number used in the masking method, and a secret key. The 128-bit random number is produced by the external random number generator, which is used for scrambling the input data block. All operations inside the processor are executed on the masked data and the corresponding random value. Hence no correlation exists between inside operations and power curves acquired externally. Our AES processor characteristics are shown in Table 1.

3.2 Our S-box Implementation

In our design, same AES hardware blocks except S-box module are used for a masked data and a random value in parallel. But we give special care for non-linear S-box processing. As noted above, S-box is constructed by composing the multiplicative inversion in GF(2^8) and the following affine

transformation over GF(2):

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

Now, viewing GF(2^8) as GF((2^4)^2) via some explicit isomorphism, $g \in \mathrm{GF}(2^8)$ can be considered as $g = (a, b)$ for some $a, b \in \mathrm{GF}(2^4)$. Hence $g^{-1} = (a, b)^{-1} \in \mathrm{GF}(2^8)$ can be computed by

$$(a, b)^{-1} = \frac{1}{a^2\lambda \oplus b(a \oplus b)}(a, a \oplus b), \qquad (2)$$

where $\lambda = (1100)_2 \in \mathrm{GF}(2^4)$ and all the operations like $a^2, a^2\lambda, b(a \oplus b)$ and $(a^2\lambda \oplus b(a \oplus b))^{-1}$ take place in GF(2^4). So the inverse operation in GF(2^8) can be computed using only three multiplications, one squaring, one constant multiplication by λ, three eXclusive-OR operations, and one inversion over GF(2^4). Note that squaring and constant multiplication by λ in GF(2^4) are linear functions so we can easily apply the masking method to those functions.

Now, repeating the above process, GF(2^4) can be viewed as GF((2^2)^2) and each element in GF(2^4) can be written as a pair of elements in GF(2^2). So, $a^{-1} = (a_1, a_2)^{-1} \in \mathrm{GF}(2^4)$ can be computed by

$$a^{-1} = \frac{1}{a_1^2\phi \oplus a_2(a_1 \oplus a_2)}(a_1, a_1 \oplus a_2), \qquad (3)$$

where $\phi = (10)_2 \in \mathrm{GF}(2^2)$ and all the operations like $a_1^2, a_1^2\phi, a_2(a_1 \oplus a_2)$ and $(a_1^2\phi \oplus a_2(a_1 \oplus a_2))^{-1}$ occur in GF(2^2). Also, the multiplication in GF(2^4) can be realized by using the operations over GF(2^2) as follows: for $a = (a_1, a_2), b = (b_1, b_2) \in \mathrm{GF}(2^4)$ and $a_1, a_2, b_1, b_2 \in \mathrm{GF}(2^2)$

$$ab = ((a_1 \oplus a_2)(b_1 \oplus b_2) \oplus a_2b_2, a_1b_1\phi \oplus a_2b_2). \quad (4)$$

Hence multiplication and inversion over GF(2^4) can be implemented using multiplications, squaring, constant multiplications by ϕ, and inversion over GF(2^2). Note that inversion in GF(2^2) is a linear function because $a^{-1} = a^2$ for all $a \in$ GF(2^2). Also, squaring and constant multiplication by ϕ in GF(2^2) are linear functions so we can easily apply the masking method to those functions. Consequently, if we can apply the masking method to the multiplication in GF(2^2), we can apply it to the the inversion in GF(2^4). Hence our paper mainly focuses on proposing multiplications with masking over GF(2^2).

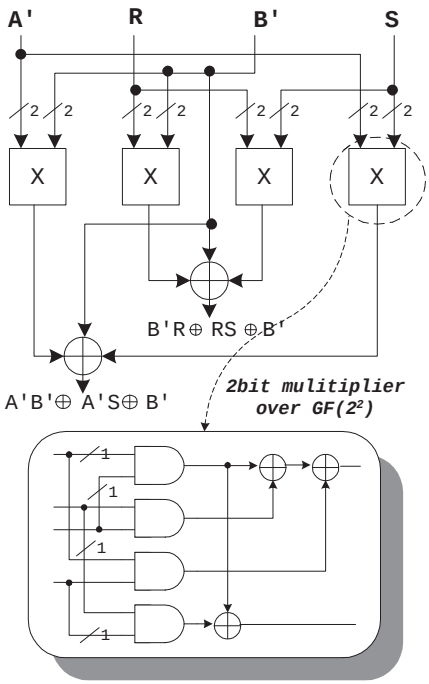

Figure 4: Multiplier Circuit over GF(2^2) with Mask

Proposed Masking Method: multiplication in GF(2^2)
Input : $(A' = A \oplus R, R), (B' = B \oplus S, S)$
Output : $(A'B' \oplus A'S \oplus B', B'R \oplus RS \oplus B')$

The above method can be justified as follows:

$$\begin{aligned} & (A'B' \oplus A'S \oplus B') \oplus (B'R \oplus RS \oplus B') \\ = \; & ((A \oplus R)(B \oplus S) \oplus (A \oplus R)S \oplus (B \oplus S)) \\ & \oplus((B \oplus S)R \oplus RS \oplus (B \oplus S)) \\ = \; & AB \end{aligned} \qquad (5)$$

Finally, since $B'R \oplus RS \oplus B'$ is uniformly distributed in GF(2^2) provided that A', B', R and

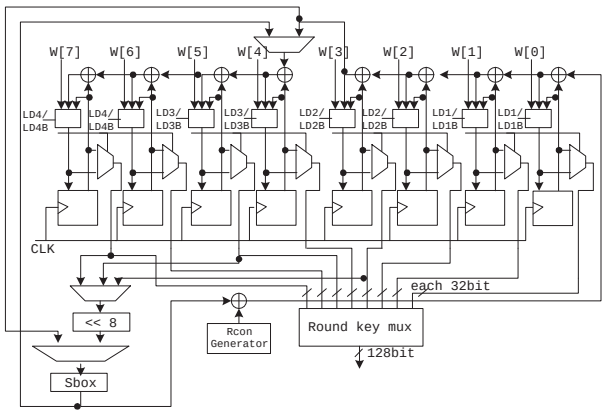

Figure 5: The On-the-fly Key Scheduler

S are randomly chosen, the proposed masking method is random and therefore it gives a good countermeasure against DPA.

Note that the proposed multiplier with mask in GF(2^2) requires only four 2-bit multipliers and four 3-input eXclusive-OR gates. This hardware architecture is shown in Figure 4.

3.3 Our Round Key Scheduler

Our round key scheduler performs the on-the-fly key expansion routine to generate each round key. In the module, there is only one 256 bit key register in which several different key sizes can be implemented. Sharing the key register saves a significant amount of area. Also key register has a special load enable circuit that reduces overall power consumption by minimizing toggling rates of flip-flops. Our key scheduler is shown in Figure 5.

4 Comparisons with Previous Works

4.1 Our Implementation Results

The architecture described in Section 3 has been implemented by using 0.18μm CMOS Smart Card technology and 25.4Kgates was obtained with about 1mA power consumption under the worst-case conditions. The detailed gate count of each component and performance are listed in Table 2 and Table 3. The power measurement result is also described in Table 4.

Components	Gate counts	%
Controller	4,159	16.4
Datapath	10,451	41.1
Key Scheduler	8,795	34.6
S-box data	1,323	5.2
S-box key	343	0.13
Interface	352	0.14
Total	25,423	100

Table 2: Components and Their Complexity

Clock Frequency (Clock period)	Throughput (128/192/256bit key)
5MHz(200ns)	4/3.3/2.5 Mbps
10MHz(100ns)	8/6.7/5 Mbps
15MHz(67ns)	11.9/10.0/8.5 Mbps

Table 3: Throughput Performance

4.2 Comparison with Related Works

So far, a lot of hardware architectures have been proposed ([7], [8], [9], [10]). But most of them mainly focus on increasing the throughput or decreasing the area and not much attention has been paid to power consumption. So, there are a lot of difficulties in comparing our design with other AES architectures. Also the difference in the used technologies does not allow reasonable comparisons for the maximum frequencies or the throughput. And the presented architecture differs significantly from other proposals in that of its masking feature and relatively low power consumption. Therefore a meaningful comparison would be possible only if the same technology is used with additional masking blocks.

5 Conclusion

AES coprocessor with a countermeasure against DPA is proposed and its overall current consumption is about 1mA at 5MHz clock frequency by us-

Components	value (μA)	value /Total
Controller	146.322	15.893
Datapath	285.967	31.061
Key Scheduler	153.718	16.696
S-box data	316.942	34.425
S-box key	17.076	1.855
Interface	0.640	0.070
Intrinsic Current	747.933 μA	70.07%
Switching Current	319.437 μA	29.93%
Total Current	1067.370 μA	

Table 4: Power Measurement Result

ing Samsung smart card 0.18μm CMOS technology. The resulting VLSI circuits achieve data rates up to 4Mbps keeping power consumption of 1mA and maximum throughput of 11.9Mbps at 16Mhz clock frequency. In comparison with the previous designs, the proposed architecture achieves a considerable increase in security level by adopting a new masking method. It also significantly reduces the computing power and area leading to a better overall performance in Smart Card applications.

References

[1] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 197, Anouncing the Advanced Encryption Standard(AES)," http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.

[2] P. Kocher, J. Jaffe and B. Jun, "Differential power analysis", *Proceedings of Crypto '99*, LNCS vol. 1666, Springer-Verlag, 1999, pp. 388–397.

[3] J.-S. Coron and L. Goubin, "On boolean and arithmetic masking against differential power analysis", *Proceedings of Cryptographic Hardware and Embedded Systems: CHES 2000*, LNCS vol. 1965, Springer-Verlag, 2000, pp. 231–237.

[4] T. Messerges, "Securing the AES fialists against power analysis attacks", *Proceedings of Fast Software EncryptionWorkshop 2000*, LNCS vol. 1978, Springer-Verlag, 2000, pp. 150–165.

[5] J. Guajardo and C. Paar, "Efficient Algorithms for Elliptic Curve Cryptosystems", *Proceedings of 17th Annual Intl. Cryptology Conf. : CRYPTO'97*, LNCS Vol.1294, pp. 342-356, 1997.

[6] A. Rudra, et al., "Efficient Rijndael Encryption Implementation with Composite Field Arithmetic", *Proceedings of Cryptographic Hardware and Embedded Systems: CHES 2001*, LNCS vol. 2162, Springer-Verlag, 2001, pp. 175–188.

[7] A. Satoh, S. Morioka, K. Takano, and S.Munetoh, "A compact Rijndael Hardware Architecture with S-Box Optimization", *Proceedings of ASIACRYPT 2001*, LNCS Vol. 2248, pp. 239–254, 2001.

[8] S. Mangard, M. Aigner and S. Dominikus, "A highly Regular and Scalable AES Hardware Architectur", *IEEE Transactions on Computers*, Vol. 52, No. 4, pp. 483–491, 2003.

[9] N. Sklavos and O. Koufopavlou, "Architectures and VLSI Implementations of the AES-Proposal Rijndael", *IEEE Transactions on Computers*, Vol. 51, No. 12, pp. 1454–1459, 2002.

[10] H. Kuo and I. Verbauwhede, "Architectural Optimization for a 1.82 Gbits/sec VLSI Implementation of the AES Rijndael Algorithm", *Workshop on Cryptographic Hardware and Embedded Systems : CHESS 2001*, pp. 68–80, 2001.