A Novel AES Cryptographic Core Highly Resistant to Differential Power Analysis Attacks

Felipe Ghellar[1] and Marcelo Soares Lubaszewski[2]

[1] Computer Science Deptartment, Federal University of Rio Grande do Sul, Porto Alegre, Brazil
[2] Electrical Engineering Deptartment, Federal University of Rio Grande do Sul, Porto Alegre, Brazil
e-mail: fghellar@inf.ufrgs.br

ABSTRACT

In this work, we present a novel core implementation of the Advanced Encryption Standard with an integrated countermeasure against side channel attacks, which can theoretically increase the complexity of a DPA attack by a factor of 240. This countermeasure is based on mathematical properties of the Rijndael algorithm, and retains compatibility with the published Standard. The entire system was designed from the ground up to allow the reutilization of the building blocks in many different combinations, thus providing for design space exploration. Synthesis results show that the protected core can perfectly meet the performance constraints of currently used smart cards.

Index Terms: AES, Rijndael, Isomorphisms, DPA.

1. INTRODUCTION

The need for secure communication dates back to the beginnings of human society. Ever since the first men started to gather in small groups, there was the need to transmit some kind of information from someone to someone else, while keeping it hidden from others. This need has accompanied us throughout History, and is still very much present today. Wireless networks, home banking, and e-commerce are examples of applications that require a high level of security. The current trend for mobile systems is pushing the market in the direction of ever smaller and more secure devices.

Although the main goal remains the same since many centuries, the way cryptography is done has significantly changed in the last decades. Security through obscurity is no more considered security at all. Today's security systems are based on open algorithms associated with strong, carefully selected secret keys.

In 2001, after an open selection process, the Rijndael algorithm [1] was announced by the US government as the new Advanced Encryption Standard (AES) [2], intended to supercede its predecessor, the Data Encryption Standard (DES). It has since been adopted internationally, and gained widespread use in a great number of cryptographic systems and devices.

Rijndael, like many modern symmetric ciphers, is built upon the iterative application of a sequence of simpler cryptographic primitives to the input data. Taken alone, none of these primitives would be able to provide security, but when combined and repeated enough times, they can achieve the level of diffusion and confusion required in today's cryptosystems.

The complexity of modern cryptographic algorithms requires that they be physically realized someway, either as a dedicated piece of hardware, or in the form of a sequence of instructions to be executed in a particular microprocessor. This opens the possibility for a specific type of attack against such devices, known altogether as physical attacks or implementation attacks [3]. Such attacks do not target the algorithm itself, but rather an implementation of it.

The fundamental threat behind implementation attacks is that they give access to intermediate values produced by the algorithm, which would not normally be available to the attacker. This in turn may lead to the extraction of the secret key, or any other sensitive information, stored in the device. Small, dedicated devices, such as security tokens and smart cards, due to their relatively lower complexity, are especially susceptible to such attacks [4].

Implementation attacks can be broadly classified as whether they are invasive or non-invasive, as well as active or passive. Invasive attacks usually require the removal of the encapsulation of the device and the exposure of its internals; an example of invasive attack is the insertion of microprobes in the device's data bus. Fault attacks are a group of active attacks which may be either invasive or non-invasive, depending on the method used to induce faults in the device; common forms of fault attacks include the use of laser beams, clock glitches, and power fluctuations. Finally, side channel attacks are a class of passive, non-invasive attacks which explore the information leakage through the device's intrinsic side channels, such as electromagnetic emanations, or the power and time consumed to perform a given operation. Contrarily to fault attacks and many forms of invasive attacks, side channel attacks cannot usually be detected by the device.

In this work, we present a novel core implementation of the Advanced Encryption Standard with an integrated countermeasure against side channel attacks, which can theoretically increase the complexity of a DPA attack by a factor of 240. This countermeasure is based on mathematical properties of the Rijndael algorithm, and retains compatibility with the published Standard.

The paper is organized as follows. Section 2. discusses some related work and clearly states the contribution of this work. Section 3. describes our proposed solution, after summarizing some important mathematical fundamentals and giving a description of the Rijndael algorithm and its mathematical properties. Section 4. presents some experimental results. Finally, conclusions are drawn in Section 5.

2. RELATED WORK

Differential Power Analysys (DPA) [5] and its more specialized variations [6, 7] are arguably the most powerful and hardest to prevent form of side channel attack known today. DPA is based on the statistical analysis of power traces collected while the device is performing a cryptographic operation – either encryption or decryption. In small microcontrollers, like the ones used in smart cards, the power traces can be easily correlated to the sequence of instructions being executed. With a little more effort, even the data being manipulated by such instructions can be inferred [8]. The same general technique can also be applied against cryptographic cores implemented as ASICs [9] or in FPGAs [10].

It is not physically possible to completely prevent the existence of side channels, as they are inherent to any physical device. Instead, the most effective countermeasures against side channel attacks are devised to try to weaken the correlation between the binary values being manipulated in the device and the information leaked via a particular side channel. In the case of DPA, the weaker the correlation between the binary values and the device's instantaneous power consumption, the more power traces the attacker needs to collect to be able to perform the required statistical analysis [11]. Rising this number to big enough values may turn the attack unfeasible or not worthy at all.

The investigation of varying Rijndael's internal data representation was introduced in [12]. Since then, many works were published exploring the use of different representations of Rijndael, with various goals. [13] investigates which one of the 240 representations mentioned in [12] leads to a better area efficiency when implemented in hardware. Other works intended to achieve protection against side channel attacks, while others yet used it as a means of assisting in cryptanalysis. A summary of several of those works can be found in [14].

In [15], the authors suggest and discuss the use of random representations as a protection mechanism against side channel attacks. They present a brief theoretical evaluation of two possible approaches to the generation of Rijndael representations, but no actual results are given. The main idea is to change, from one encryption/decryption operation to the next, the Rijndael representation, so that different binary values are internally processed. Doing that, the correlation between side channel information and the data manipulated inside the device is reduced, making it harder to extract the secret key or any other sensitive information.

An implementation of the Rijndael algorithm using selectable representations is presented in [16]. The implementation, named by the authors as Diversified AES (DAES), is targeted to an FPGA device with an embedded microprocessor core. Given the need for extra inputs, in addition to the secret key and the data to be processed, DAES is not directly compatible with other implementations of the AES.

In this work, 240 randomly selectable representations of Rijndael are used to protect an implementation of the Advanced Encryption Standard against side channel attacks. This countermeasure builds from the mathematical properties of the Rijndael algorithm discussed in [15] and, contrarily to [16], fully complies with the published Standard.

3. A PROTECTED AES ARCHITECTURE

In this section, we first summarize some mathematical fundamentals required to understand the mechanism we use to protect Rijndael against side channel attacks. Then, we give a description of the algorithm and discuss its mathematical properties. Finally, we define our threat model and describe the protected core implementing the Advanced Encryption Standard.

A. Mathematical Background

In this section, we give a brief overview of some concepts of Abstract Algebra required to understand the remainder of the work. For a more detailed coverage of the subject, we refer the reader to *e.g.* [17].

A *field* is an algebraic entity composed of a set of elements and two binary operations, addition and multiplication, such that, when applied to two elements of the given set, the result is also an element of the same set. These operations follow the properties listed in Table I. Given this definition, the set $\mathbb{Q}$ of the rational numbers, the set $\mathbb{R}$ of the real numbers, and the set **C** of the complex numbers are fields, while the set $\mathbb{N}$ of the natural numbers and the set $\mathbb{Z}$ of the integers are not.

A *finite field* is a field with a finite number of elements. The sets Z_n of the integers modulo n, where n is a prime number, and with addition and multiplication performed modulo n, are examples of finite fields.

A finite field F_1 with p^n pelements, where p is a prime number, can be equivalently viewed as a field by itself, or as an *extension* of a smaller field F_2 with p^m elements, being $m < n$. The field F_2 is then a *subfield* of F_1 , and p is the *characteristic* of both fields. There are several ways to express a field F_1 in terms of a subfield F_2 . One way is by building vectors composed of elements of F_2 , thus defining F_1 as a *vector space* over F_2 . Another way is by taking the elements of F_2 as coefficients of a polynomial, thus defining F_1 as a *polynomial space* over F_2 . In both cases, the number of elements of F_2 used to build one element of F_1 is known as the *order* of the extension field relative to the *base field*.

The *Galois field* GF(2) is the finite field composed of two elements, 0 and 1, and with addition and multiplication performed modulo 2, as shown in Table II. The Galois field GF(2^8), also known as GF(256), is the finite field of 256 elements, which can also be viewed as an extension of order 8 of the base field GF(2). The notation GF(2^8) emphasizes this property. Examples of different notations for elements of GF(2^8) are given in Table III. Any of the notations can be freely chosen, according to the situation.

When viewed as a polynomial space over GF(2), GF(2^8) is characterized by an *irreducible polynomial* of degree 8 and a *generator element* associated to that polynomial. A generator element is an element of the given field with the property that all non-zero elements of the field can be obtained by taking powers of the generator element, modulo the irreducible polynomial. Multiplication of GF(2^8) elements is performed via polynomial multiplication, modulo the irreducible polynomial.

Table II: Addition and multiplication in *GF* (2)

Addition	Multiplication
0 + 0 = 0	0 . 0 = 0
0 + 1 = 1	0 . 1 = 0
1 + 0 = 1	1 . 0 = 0
1 + 1 = 0	1 . 1 = 1

Table III: Common notations for *GF* (2^8) elements

Hex	Binary	Vector	Polynomial
03	00000011	(0,0,0,0,0,0,1,1)	$x + 1$
85	10000101	(1,0,0,0,0,1,0,1)	$x^7 + x^2 + 1$
A2	10100010	(1,0,1,0,0,0,1,0)	$x^7 + x^5 + x$

All finite fields with a given number of elements are *isomorphic* to a Galois field with the same number of elements. Hence, all such fields are mathematically the very same field, only with a different *representation*. It is always possible to define an invertible *mapping function* to convert between any two of those representations.

B. The Rijndael Algorithm and Isomorphisms

The Rijndael cryptographic algorithm, as standardized in [2], is a block cipher with a fixed data block size of 128 bits and a configurable key size of 128, 192 or 256 bits. For both encryption and decryption, the input data is first stored in a four-by-four array of bytes, known as the State matrix. The State is then processed in a sequence of rounds, the number of which depends on the size of the key, being 10, 12, and 14 for 128-, 192-, and 256-bit keys, respectively. A key scheduling algorithm expands the input key into 10, 12 or 14 subkeys, according on the number of rounds, of 128 bits each (the same size of the data block). One subkey is used for each round. For the purpose of this work, and without any loss of generality, only the encryption process for 128-bit keys will be considered hereafter. A simplified block diagram of the AES algorithm is shown in Figure 1.

Each round in Rijndael has four different steps that perform operations on the State. The first step is a

Table I. Properties of field operations

Property	Addition	Multiplication
Associativity	a + (b + c) = (a + b) + c	a · (b · c) = (a · b) · c
Comutativity	a + b = b + a	a · b = b · a
Existence of Identity Element	a + 0 = a	a · 1 = a
Invertibility	a + (–a) = 0	a · a^{-1} = 1
Distributivity of Multiplication over Addition	a · (b + c) = a · b + a · c	

byte substitution operation, called SubBytes, where each byte in the array is updated using a fixed substitution table (SBox). The second operation, called ShiftRows, acts on the rows of the State matrix, cyclically shifting the bytes in each row by a different number of positions. In the third step, called MixColumns, the four bytes in each column are intermixed to form another set of four bytes, in such a way that each new byte is a combination of all the old ones. Finally, in the fourth step, AddRoundKey, the subkey for the respective round is combined with the State. The last round of the algorithm suppresses the MixColumns operation.

Rijndael has a strong arithmetic foundation. Bytes are considered as elements of the finite field GF(2^8) represented by the irreducible polynomial $m(x)= x^8 + x + x^3 + x + 1 = \{11B\}$ and the generator element $\beta = x + 1 = \{03\}$. Likewise, all the round transformations which operate over the State matrix are actualy algebraic operations defined in or over this field.

The SubBytes operation is composed of a multiplicative inversion in GF(2^8) followed by an affine transformation, which is a linear operation of the form $Ax + c$, where x and c are eight-element binary vectors and A is an eight-by-eight binary matrix. The multiplicative inversion is relative to the field polynomial $m(x)$, and the constants A and c involved in the affine transformation are dependent on the field polynomial and the generator element.

MixColumns uses the four bytes of each column of the State matrix to build a four-term polynomial with coefficients in GF(2^8). This polynomial is then multiplied over GF(2^8) by the constant polynomial $C(x) = 03x^3 + 01x^2 + 01x+ 02$, modulo x^4+1. The resulting four-term polynomial is split back into four bytes and stored in the same column of the State matrix.

AddRoundKey simply adds each byte of the round key to the corresponding byte of the State matrix. As additions in GF(2^8) always result in field elements, there is no need for reduction of the result.

ShiftRows is an operation which takes each row of the State matrix and cyclically shifts its bytes by different amounts. It also has an algebraic description, but for a hardware implementation it is more effective to simply implement it as routing.

The investigation of Rijndael isomorphisms was introduced in [12]. Two ciphers are isomorphic if they produce the same output for the same input. There are 30 irreducible polynomials of degree 8 over GF(2), and each has 8 generator elements associated to it. This brings the possibility to build 240 representations of the GF(2^8) field used in Rijndael. Adding a mapping function to the beginning of the algorithm, an inverse map to the end, and adapting the round transformations to use the new representation, the same number of isomorphic Rijndael ciphers can be obtained.

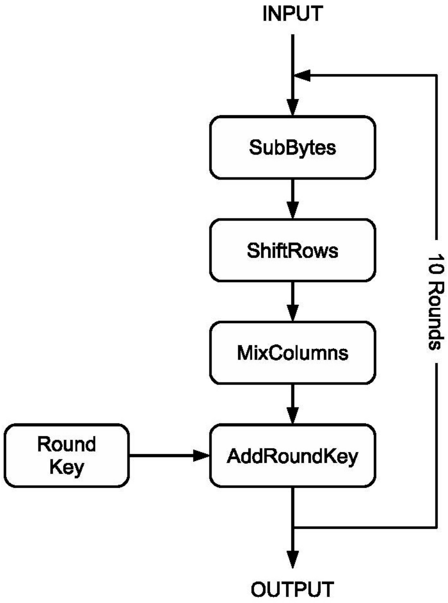

Figure 1. Simplified block diagram of the AES algorithm.

C. Threat Model

To be able to asses the level of security of any given system, one needs to first define the threat model to be considered. In other words, one has to clearly understand what capabilities the potential adversary has.

As an analogy, a small fence around a house may be enough to keep street dogs away from the frontyard, but it surely cannot keep a person from getting there. That would require at least a higher fence, and maybe a padlock. A couple of locks on the door can go a long way to protect the house against uninvited visitors, but determined burglars will still find a way to break in.

The effectiveness of any security system, no matter how well-thought and how expensive it may be, is limited by the level of determination and the budget of whoever is trying to break it. No fence, lock, or wall can stop someone with a backhoe. If the value of what is inside the house is worth the cost of a backhoe, it might be better to build a bunker.

The same idea applies to cryptographic systems. The level of security needed, and the *actual* level of security achieved, are dependent on whom we are trying to protect against. Who our adversaries are, how determied they are, and what their budget is.

In the case of smart cards and other similar security tokens, the user is in possession of the device and has full control over its operation. This includes not only the data inputs, but also the power and clock lines, not to mention disassembing the device. The user can also monitor and record any kind of information about the operation of the device, such as power traces.

In this scenario, the aim of the adversary is usually to try to retrieve some kind of secret information which is stored in the device. In our case, we consider

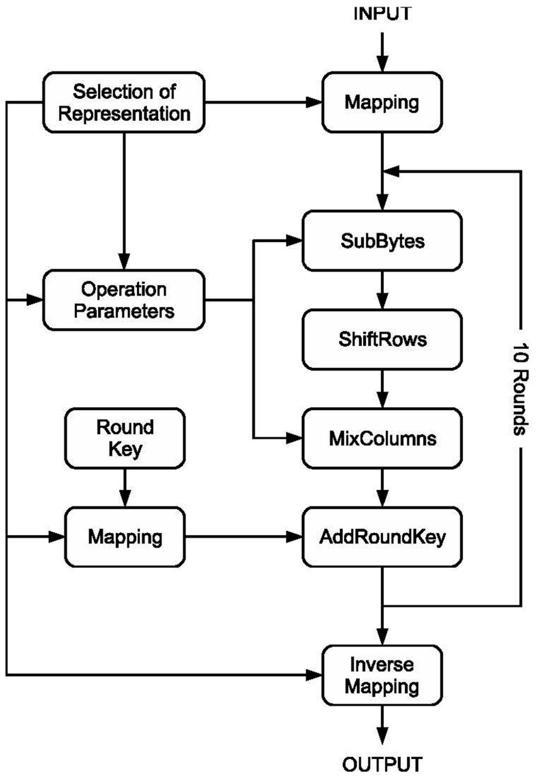

Figure 2. Block diagram of the proposed implementation

that this iformation is the secret key for the Rijndael algorithm, which, for the purpose of this work, is stored in the device in its expanded form.

D. Proposed Implementation

In our AES core implementation, any one of the 240 representations of Rijndael mentioned above may be randomly selected for protection against side channel attacks. This countermeasure is based on the first approach suggested in [15], and offers a protection degree against DPA up to 240 times higher than in an unprotected AES implementation. Additionally, our AES core fully complies with the Standard, since, differently from the implementation proposed in [16], no additional parameters than the secret key and the data to be processed are needed. A block diagram of our protected AES architecture is presented in Figure 2.

Modularity is a key aspect in the implementation of our protected AES core. The entire system was designed from the ground up to allow the reutilization of the building blocks in many different combinations, thus providing for design space exploration.

Since Rijndael is essentially made up of algebraic operations, the most basic modules are those that perform arithmetic in GF(2^8). Thus we first built an adder, a multiplier, and an inverter, all designed to be representation-independent. We also built modules to perform other operations with GF(2^8) elements, such as the affine transformation used in SubBytes, and the field mappings.

All of Rijndael's round transformations can be easily built from these few basic modules. As stated in Section B., Sub-Bytes is made up of an inversion followed by an affine transformation, while AddRoundKey uses only additions. The polynomial multiplication from MixColumns is broken down into GF(2^8) additions and multiplications. Only ShiftRows does not use any of the previous modules, as it is implemented as simple routing.

The building blocks of the SubBytes and MixColumns operations allow for non-fixed cipher parameters. In Sub-Bytes, the irreducible polynomial used to calculate the inversion in GF(2^8) and the binary matrix and vector involved in the affine transformation are provided as inputs to the operational module. In MixColumns, the polynomial C(x), used to intermix the bytes in each column of the State, and the irreducible polynomial, used to perform multiplications in GF(2^8), are also inputs to that module.

A mapping module is introduced at the input to convert the data to be processed from the standard Rijndael representation to the selected one. To perform the conversion, a GF(2^8) element, taken as an eight-dimensional vector of GF(2) elements, is multiplied by an eight-by-eight binary matrix, giving another eight-dimensional vector, which contains the new representation for the GF(2^8) element. Such mapping function is fundamentally the same transformation as performed in a change of base in Linear Algebra. An inverse mapping module is added before the data is output to convert it back to the standard representation. As the fundamental objective of an implementation attack is to recover the secret key from the device, we do not provide an input port for the key. We rather assume that it is already expanded and stored inside the device, as is the case for smart cards and other security tokens.

The operation of the system is as follows. First, one of the 240 possible representations, consisting of an irreducible polynomial and one of the eight generator elements associated to it, is randomly selected. In our prototype, a linear feedback shift register (LFSR) is used as a random number generator for proof of concept. As this LFSR implements a primitive polynomial, all 240 representations are visited before one representation is repeated. An LFSR should not be used in production, though, since its pseudo-randomness and high mathematical predictability cannot provide security against statistical attacks such as DPA. For production, it is better to use an specialized IP core with stronger randomness characteristics, such as the one in [18]. Our modular design allows for easy integration of such solution.

Next, appropriate parameters for the round transformations are derived for the selected representation, and fed to the corresponding operational units. The construction of all those parameters involves only the same basic arithmetic operations described previously. Mapping and inverse mapping matrices are generated, and the input data is converted to the internal representation. The usual 10 rounds of transformations are applied, and after that the data is converted back to the standard representation. The output is the same as if it were produced by a standard AES core.

In this system, each time a new encryption operation is performed, a different, randomly selected representation is used internally. The main differences between any two representations are the binary values assigned to the field elements. Since different binary values are used each time, power traces collected for DPA will be weakly correlated to the data being manipulated, even if many runs are performed for the same inputs. The weaker the correlation between the binary values and the device's instantaneous power consumption, the more power traces the attacker needs to collect to be able to perform the required statistical analysis. Additionally, considering the hypothesis that all representations are used before one of them is repeated, this number rises by a factor of 240, which is likely to turn the attack unfeasible or, at least, too expensive.

4. EXPERIMENTAL RESULTS

From the description of the proposed AES implementation given in the previous section, one can easily conclude that area and performance penalties will be associated to the resulting protected core. To evaluate the price to pay for such a high degree of protection, two versions of the AES encryption process using the basic modules described in Section D. were built and are described next.

The first version utilizes one clock cycle for each of the 10 encryption rounds, plus one cycle for input and another for output. In the second version, one clock cycle is used per round transformation (thus four cycles per round), plus one cycle for input and one for output, which amounts to 42 clock cycles. In both versions, the same building blocks were used, and only the top of hierarchy was changed. For each version, two equivalent cores were built for comparison purposes, one with and another without the isomorphic operations (protection).

Table IV summarizes the synthesis results using the commercial tool LeonardoSpectrum [19], with its default settings and sample technology file (SCL05u). First of all, note that version 2 of the unprotected core presents the best performance result, but pays with additional area in comparison to the unprotected version 1. Next, as shown by the area increase and frequency decrease columns, it is natural that the isomorphism-enabled cores, having such a higher protection level, use more area and be slower than their unprotected counterparts. On the other hand, note that the version 2 is the one that presents the least relative implementation penalties for protection. This is due to the increasing design complexity of version 2 for performance improvement. Finally, even with such a decrease in speed due to the embedded protection, any of the two versions still perfectly fits in the design of most smart cards, where the working frequency is usually around 5 MHz.

5. CONCLUSIONS AND FUTURE WORK

We presented a novel AES core with an integrated countermeasure against side channel attacks, which can increase the complexity of a DPA attack by a factor of up to 240. This countermeasure is based on mathematical properties of the Rijndael algorithm, and retains compatibility with the published Standard.

Our modular design permits the construction of actual cores with very different area and speed characteristics, while still keeping a high protection level. The two implementations given in Section 4. were to illustrate that exploring the design space may improve area and speed figures, even if one knows beforehand that the price to pay for the degree of protection proposed in this work cannot be low.

Obviously, there is still much room, beyond these two examples, to explore the design space. For instance, we have not explored in this work the possibility of reducing the area and performance penalties by playing with the degree of protection of the core. It is likely that reducing the number of possible representations of Rijndael implemented in the core, the hardware complexity will be reduced and, despite the protection reduction, better cost figures will be obtained.

Future work includes experiments in this direc-

Table IV. Synthesis results for area (gates) and frequency (MHz)

Core Version	Area	Area Increase	Freq.	Freq. Decrease
Version 1				
Unprotected 12-cycle	32.7k		94.2	
Protected 12-cycle	96.5k	295%	37.6	60%
Version 2				
Unprotected 42-cycle	41.7k		105.3	
Protected 42-cycle	105.5k	253%	62.3	41%

tion, as well as the investigation of other forms of isomorphisms, as mentioned in [15] and [20], and stronger protection against more advanced variations of the DPA attack.

REFERENCES

[1] J. Daemen and V. Rijmen, *The Design of Rijndael*. Springer, 2002.

[2] National Institute of Standards and Technologies, *Advanced Encryption Standard (FIPS 197)*, 2001.

[3] R. Anderson and M. Kuhn, "Tamper Resistance – a Cautionary Note," in *Proc. 2nd USENIX Workshop on Electronic Commerce*, pp. 1–11, USENIX Association, 1996.

[4] E. Hess, N. Janssen, B. Meyer, and T. Schütze, "Information Leakage Attacks Against Smart Card Implementations of Cryptographic Algorithms and Countermeasures – A Survey," in *Proc. EUROSMART Security Conference*, pp. 55–64, 2000.

[5] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology – CRYPTO '99*, vol. 1666 of *LNCS*, pp. 388–397, Springer, 1999.

[6] S. Chari, J. Rao, and P. Rohatgi, "Template Attacks," in *Cryptographic Hardware and Embedded Systems – CHES 2002*, vol. 2523 of *LNCS*, pp. 13–28, Springer, 2002.

[7] W. Schindler, K. Lemke, and C. Paar, "A Stochastic Model for Differential Side Channel Cryptanalysis," in *Cryptographic Hardware and Embedded Systems – CHES 2005*, vol. 3659 of *LNCS*, pp. 30–46, Springer, 2005.

[8] P. Fahn and P. Pearson, "IPA: A New Class of Power Attacks," in *Cryptographic Hardware and Embedded Systems – CHES '99*, vol. 1717 of *LNCS*, pp. 173–186, Springer, 1999.

[9] S. Örs, F. Gürkaynak, E. Oswald, and B. Preneel, "Power-Analysis Attack on an ASIC AES implementation," *Proc. International Conference on Information Technology – Coding and Computing – ITCC 2004*, vol. 2, 2004.

[10] S. Örs, E. Oswald, and B. Preneel, "Power-Analysis Attacks on an FPGA – First Experimental Results," in *Cryptographic Hardware and Embedded Systems – CHES 2003*, vol. 2779 of *LNCS*, pp. 35–50, Springer, 2003.

[11] T. Messerges, E. Dabbish, and R. Sloan, "Investigations of Power Analysis Attacks on Smartcards," in *Proc. USENIX Workshop on Smartcard Technology – Smartcard '99*, pp. 151–162, USENIX Association, 1999.

[12] E. Barkan and E. Biham, "In How Many Ways Can You Write Rijndael?," in *Advances in Cryptology – Asiacrypt 2002*, vol. 2501 of *LNCS*, pp. 160–175, Springer, 2002.

[13] S.-Y. Wu, S.-C. Lu, and C. S. Laih, "Design of AES Based on Dual Cipher and Composite Field," in *Topics in Cryptology – CT-RSA 2004*, vol. 2964 of *LNCS*, pp. 25–38, Springer, 2004.

[14] V. Rijmen and E. Oswald, "Representations and Rijndael Descriptions," in *AES 2004 – 4th International Conference*, vol. 3373 of *LNCS*, pp. 148–158, Springer, 2005.

[15] A. Rostovtsev and O. Shemyakina, "AES side channel attacks protection using random isomorphisms," *Cryptology ePrint Archive, Report 2005/087*. Available from: http://eprint.iacr.org/2005/087.

[16] M.-H. Jing, Z.-H. Chen, J.-H. Chen, and Y.-H. Chen, "Reconfigurable system for high-speed and diversified AES using FPGA," *Microprocessors and Microsystems*, vol. 31, no. 2, pp. 94–102, 2007.

[17] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*. Cambridge University Press, 2 ed., 1994.

[18] B. Barak, R. Shaltiel, and E. Tromer, "True Random Number Generators Secure in a Changing Environment," in *Cryptographic Hardware and Embedded Systems – CHES 2003*, vol. 2779 of *LNCS*, pp. 166– 180, Springer, 2003.

[19] Mentor Graphics, *LeonardoSpectrum*, 2006. http://www.mentor.com/products/.

[20] H. Raddum, "More Dual Rijndaels," in *AES 2004 – 4th International Conference*, vol. 3373 of *LNCS*, pp. 142–147, Springer, 2005.