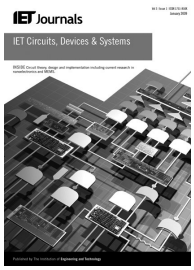
Techset Composition Ltd, Salisbury
Doc: {IEE}CDs/Articles/Pagination/CDS20110278.3d

Published in IET Circuits, Devices & Systems
Received on 13th September 2011
Revised on 11th January 2012
doi: 10.1049/iet-cds.2011.0278



ISSN 1751-858X

On the use of Karatsuba formula to detect errors in GF($(2^n)^2$) multipliers

S. Pontarelli A. Salsano

Department of Electronic Engineering, University of Rome 'Tor Vergata', Via del Politecnico 1, Rome 00133, Italy
E-mail: pontarelli@ing.uniroma2.it

Abstract: Galois fields are widely used in cryptographic applications. The detection of an error caused by a fault in a cryptographic circuit is important to avoid undesirable behaviours of the system that could be used to reveal secret information. One of the methods used to avoid these behaviours is the concurrent error detection. Multiplication in finite field is one of the most important operations and is widely used in different cryptographic systems. The authors propose in this study an error-detection method for composite finite-field multipliers based on the use of Karatsuba formula. The Karatsuba formula can be used in GF($(2^n)^2$) field to decrease the hardware complexity of the finite-field multiplier. The authors propose a novel finite-field multiplier with concurrent error-detection capabilities based on the Karatsuba formula. How the error-detection capabilities of this multiplier are able to face a wide range of fault-based attacks is also shown.

1 Introduction

The increasing scaling rate of the microelectronic technologies observed in the last years pushes for the use of fault-tolerant techniques, traditionally used in critical applications such as aerospace and avionics ones, and also in commercial ones. For example, the effects of the neutrons at sea levels when subnanometric microelectronics technologies are used seem to be not negligible. Also, the use of fault injection attack in cryptography is a reason to develop high reliable system when the secrecy is important. These are the main motivations to study the design of fault-tolerant circuits performing operations in finite fields. In fact, operations in finite fields are often used in cryptographic applications and in error detecting and correcting codes. The detection of a fault is useful to avoid an undesirable behaviour of the system in both these applications.

Recently, a number of papers have considered error detection in finite field arithmetic circuits, for example [1–10]. Also some methods for error detection in complex structures like Reed-Solomon encoders and decoders [11], in elliptic curve cryptography [12] or in the advanced encryption standard [13] or in the *S*-box block [14] have been recently proposed. One of the most important operations in finite fields is the multiplication. It requires a considerable amount of silicon area in the implementation of circuits for cryptographic applications, and therefore it is one of the blocks of the circuit most susceptible to faults. Moreover, other complex finite-field arithmetic operations such as inversion and exponentiation over binary extension fields can be performed by repeated multiplications. Almost all the proposed methods for error detection in finite-field multipliers are based on the use of parity bits [1, 4–10].

One parity check bit can detect any odd number of erroneous bits in the result of the finite-field multiplier, while it cannot directly detect an even number of erroneous bits. It must be noticed that, because of the sharing of the hardware resources in very large-scale integration (VLSI) implementation of finite-field multiplier, a single stuck-at fault affecting a gate of the circuit can produce multiple erroneous bits. Some extensions that use multiple parity bits have been proposed in [1, 9], but also in these multipliers the 100% fault coverage has not been proved. The use of these parity bits-based methods requires a fine tuning of the overall design flow. In fact, to guarantee the 100% fault coverage when one of more parity bits is used, some additional constraint on the logic optimisation must be added [14, 15]. These constraints should assure that no sharing occurs between logic gates belonging to the same parity group. The logic optimisation with this additional constraint has been called structure-constrained logic optimisation in [15]. This limitation is related to the used fault model (the classic stuck-at-0/1 fault model) that creates a strong connection between the fault-tolerance properties of the finite-field multiplier and the implemented circuit.

Moreover, while standard fault-tolerance techniques use the well-known single-fault assumption [16], in the case of fault injection attacks multiple faults can be induced to the cryptographic circuit [17]. For example, laser-based attacks [17] can induce multiple event upsets. With the technology shrinking, this effect will become more and more likely. The error-detection capabilities of circuits based on the use of parity bits can be further reduced when the attacker is able to induce multiple faults to the circuit under attack. In [17], also fault attacks based on power or clock glitches have been described. These faults are able to supersede the

error-detection capabilities of many circuits based on the single-fault assumption. However, this kind of induced fault occurs in a physically limited silicon area of the circuit, or, in the case of faults occurring to the clock and power lines, can be confined by using different power and clock lines for different areas of the circuit.

In this paper a high-level fault model is used, in order to develop (and to prove the effectiveness of) an error-detection method applicable also for this kind of faults. The achieved error-detection properties for the circuit under consideration are also less dependent from the implementation process with respect to the previously cited solutions.

In particular, since a composite field is used to design the finite-field multiplier, the error analysis is performed at level of the elements of the ground field, instead of the bit level usually used in the methods based on parity bits. This fault model works at a higher abstraction level with respect to the bit level of the classic stuck-at model, considering that any basic component of the finite-field multiplier can be affected by an error that change the value of the arithmetic operation that the component performs, by an unpredictable value. We can refer to a basic component, and to the corresponding basic arithmetic operation performed, as any addition or multiplication performed in the ground field. Therefore any addition or multiplication in the ground field can be affected by an error of any magnitude. We remark that this arithmetic-level fault model is a superset of the classic stuck-at model, since all the errors caused by a stuck-at fault affects only one basic component performing an arithmetic operation. Using this fault model, even if the implementation of the basic component chances, the error detection properties of the overall circuit remain unchanged, since the used fault model covers any fault that can occur in any implementation of the single basic component.

This fault model also allows to design multipliers that are robust with respect to attacks based on power and clock lines. In fact, the proposed method identifies some subset of the circuits, (the boxes in Fig. 3) that can be powered by different power sources and clocked by different clock distribution lines. If a fault occurs to the power or to the clock line, it behaves as an arithmetic-level fault, since it affects only one box of the multiplier, corresponding to the output of one basic arithmetic operation. Since the other boxes are not affected by the error caused by the induced fault, the system is able to detect also this kind of faults.

Our proposed technique is based on the use of Karatsuba multiplication that allows computing the multiplication in a composite field starting from smaller multiplications performed on the ground field. The Karatsuba method to speed up multiplication has been originally proposed for integer multiplication in 1962 in the paper [18], and has been extended to finite-field multiplications in composite fields in [19, 20] and generalised in [21]. Our method requires the computation of an additional multiplication in the ground field and produces two alternative values for the result of the multiplication. The two alternative results are equal to the multiplication result if no errors occur and are different from each other if an error caused by a fault inside the multiplier occurs. The paper is organised as follows: in Section 2 the use of the Karatsuba formula for multiplication in composite Galois field is described, while in Section 3 the implementation of a concurrent error detection Karatsuba multiplier is proposed and Section 4 demonstrates that a fault in one of the elements composing the multiplier can be detected comparing the two alternative values. Finally, conclusions are drawn in Section 5.

2 Karatsuba multiplication in GF($(2^n)^2$)

A GF(2^n) Galois field is a finite field of 2^n elements and is constructed using an irreducible polynomial $p(x)$ of degree n. GF(2^n) is defined as the quotient ring obtained by the reduction modulo $p(x)$ of the ring of polynomials with coefficient in GF(2)[x], that is, GF(2^n) $\cong$ GF(2)[x]/$p(x)$. The elements of GF(2^n) can be represented as polynomial of degree less than n with coefficients in GF(2). The construction of a composite field GF($(2^n)^2$) can be obtained in a similar way. We start from GF(2^n)[x], the polynomial ring with coefficient in GF(2^n), choose an irreducible polynomial $p(x) \in$ GF(2^n)[x] having the form $p(x) = x^2 + x + p_0$ and construct the field GF($(2^n)^2$) as the quotient ring GF(2^n)[x]/$p(x)$. This field is isomorphic to the field GF(2^{2n}). The field GF(2^n) over which the composite field is defined is called the ground field and the elements of GF($(2^n)^2$) can be represented as polynomial of degree less than 2 with coefficients in GF(2^n).

The multiplication of two elements $A(x)$ and $B(x)$ of a Galois field can be performed multiplying the polynomials and reducing modulo $p(x)$ the result of the multiplication. If we have $A(x)$, $B(x) \in$ GF($(2^n)^2$) and $p(x) = x^2 + x + p_0$, $A(x)$ and $B(x)$ can be represented as $A(x) = A_1x + A_0$ and $B(x) = B_1x + B_0$. The result of the multiplication is

$$\begin{aligned} C(x) &= A(x)B(x) \\ &= A_1B_1x^2 + (A_1B_0 + A_0B_1)x + A_0B_0 \\ &= \underbrace{(A_1B_1 + A_1B_0 + A_0B_1)}_{C_1}x + \underbrace{A_0B_0 + p_0A_1B_1}_{C_0} \\ &= C_1x + C_0 \end{aligned} \tag{1}$$

The multiplication of two elements of GF($(2^n)^2$) therefore requires four multiplications in the ground field GF(2^n), a constant multiplication for the coefficient p_0 and three additions. The schema of the multiplier in a composite field is shown in Fig. 1*a*.

The Karatsuba formula computes the multiplication by using only three multipliers in the ground field. First, three intermediate values are computed

$$\begin{aligned} L &= A_0B_0 \\ H &= A_1B_1 \\ K &= (A_1 + A_0)(B_1 + B_0) \end{aligned}$$

For K the following equation holds

$$\begin{aligned} K &= (A_1 + A_0)(B_1 + B_0) \\ &= \underbrace{A_1B_1 + A_1B_0 + A_0B_1}_{C_1} + \underbrace{A_0B_0}_{L} \\ &= C_1 + L \end{aligned} \tag{2}$$

From the three intermediate values, C_1 and C_0 can be computed as

$$\begin{aligned} C_1 &= (K + L) \\ C_0 &= A_0B_0 + p_0A_1B_1 = L + p_0H \end{aligned}$$

doi: 10.1049/iet-cds.2011.0278

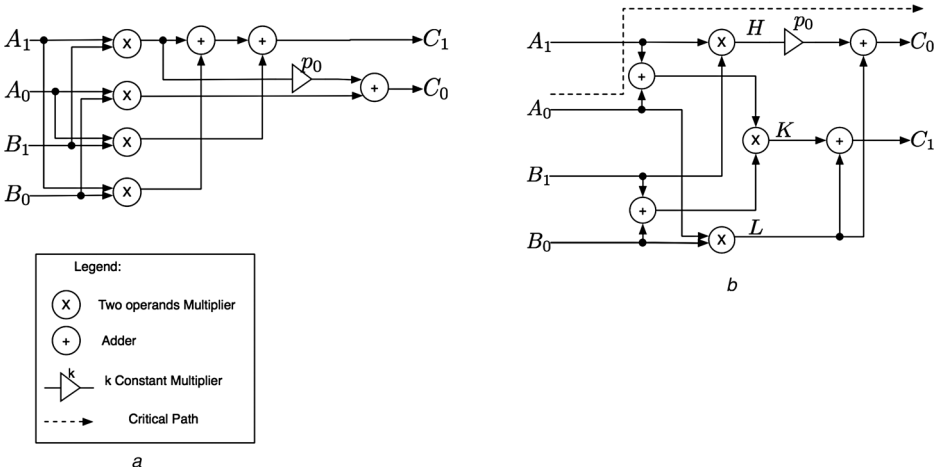


Fig. 1 *Schema of the multiplier*

a Standard multiplier
b Karatsuba multiplier

The final result C can be computed as

$$C = A(x)B(x) = (K + L)x + L + p_0H \tag{3}$$

The multiplication performed following (3) requires three multiplications, a constant multiplication for p_0 and four additions. The schema of the multiplier implementing (3) in presented in Fig. 1*b*. The implementation of a multiplier in a composite field GF($(2^n)^2$) as defined in (3) requires fewer resources than the implementation of a standard multiplier in the isomorphic field GF(2^n).

In fact, a multiplier in a finite field of degree m requires a number of AND and XOR gates that can be specified as [19–21]

$$\#\text{AND} \simeq m^2 \tag{4}$$

$$\#\text{XOR} \simeq m^2 \tag{5}$$

Readers interested to a detailed analysis of space and time complexity of this multiplier can refer to paragraph 6.1.2 (and specifically to Table 6.1) of the PhD thesis of Paar [19]. Here, we recall the main results related to this multiplier.

The space complexity of a multiplier realised in a composite field GF($(2^n)^2$) can be estimated as

$$\#\text{AND} \simeq 3n^2 \tag{6}$$

$$\#\text{XOR} \simeq 3n^2 + 4n \tag{7}$$

In fact, each multiplier requires n^2 AND and n^2 XOR and each adder requires n XOR. The constant multiplication can be realised, with the right choice of p_0, with a few XOR gates.

Fig. 1*b* also shows the critical path (the dash line) of the Karatsuba multiplier. This path is composed by two adders, one multiplier and one constant multiplier. The delay of an adder is simply the delay of a 2 inputs XOR gate, which in this paper will be indicated as T_X. The delay of the constant multiplier is strictly related to the choice of the p_0 value. For many values of p_0 the delay of the constant multiplier can be estimated as one XOR port, that is, $T_{p_0} = T_X$. Finally, the delay of the multiplier in the GF(2^n) field is $2\lceil \log_2 n \rceil T_X + T_A$, where T_A is the delay of a two-input AND gate [19]. The total delay is therefore

$$T = T_{p_0} + (2\lceil \log_2 n \rceil + 1)T_X + T_A \simeq (2\lceil \log_2 n \rceil + 2)T_X + T_A \tag{8}$$

In order to compare the complexity of the Karatsuba multiplier with the one of a standard multiplier, we report the space and delay complexity of a multiplier in the isomorphic field GF(2^n).

The number of required AND and XOR gates is

$$\#\text{AND} \simeq 4n^2 \tag{9}$$

$$\#\text{XOR} \simeq 4n^2 \tag{10}$$

since the degree of the field is $2n$. The use of the Karatsuba formula therefore allows saving the 25% of gates with respect to a standard implementation.

The delay of the multiplier is

$$\begin{aligned} T &= 2\lceil \log_2 2n \rceil T_X + T_A \\ &= 2(\lceil \log_2 n \rceil + \log_2(2))T_X \\ &\quad + T_A = (2\lceil \log_2 n \rceil + 2)T_X + T_A \end{aligned} \tag{11}$$

and therefore is approximately equal to the delay of the Karatsuba multiplier.

3 Concurrent error-detection Karatsuba multiplier

3.1 Architecture of the concurrent error-detection Karatsuba multiplier

In this section, we show how to detect an error in a GF(2^n) multiplier realised using the Karatsuba formula. First, we define an additional intermediate result named K_α defined as

$$\begin{aligned} K_\alpha &= (A_0 + \alpha A_1)(B_0 + \alpha B_1) \\ &= \underbrace{A_0B_0}_{L} + \alpha(A_1B_0 + A_0B_1) + \alpha^2 \times \underbrace{A_1B_1}_{H} \end{aligned} \tag{12}$$

$\alpha \in$ GF(2^n) is an element of the ground field. From the

definition of K_α, we obtain

$$A_1B_0 + A_0B_1 = \alpha^{-1}(K_\alpha + L + \alpha^2 H) \tag{13}$$

and therefore C_1 can be computed as

$$\begin{aligned} C_1 &= \underbrace{A_1B_0 + A_0B_1}_{\alpha^{-1}\cdot(K_\alpha+L+\alpha^2H)} + \underbrace{A_1B_1}_{H} \\ &= \alpha^{-1}(K_\alpha + L + \alpha^2 H) + H \\ &= \alpha^{-1}(K_\alpha + L) + (\alpha + 1)H \end{aligned}$$

Now we compute two alternative values of the multiplication $A(x)\cdot B(x)$ as

$$C'(x) = (K + L)x + L + p_oH = C'_1x + C'_0 \tag{14}$$

$$\begin{aligned} C''(x) &= (\alpha^{-1}(K_\alpha + L) + (\alpha + 1)H)x + L + p_0H \\ &= C''_1x + C''_0 \end{aligned} \tag{15}$$

The computation of the two alternative values requires four multiplications in the ground field GF(2^n) and five constant multiplications. If no errors occur the two alternative values are identical, and $C(x) = C'(x) = C''(x)$. Instead, if an error occurs, the two alternative values are different and comparing $C'(x)$ and $C''(x)$ we can detect an error inside the Karatsuba multiplier. To achieve the error-detection property to the Karatsuba multiplier also the circuit performing the comparison between $C'(x)$ and $C''(x)$ should be designed to be able to detect errors inside itself. A two rail self-checking comparator [22] can be used to compare the values of $C'(x)$ and $C''(x)$ and to assure the detection of errors both in the Karatsuba multiplier and in the circuit performing the comparison. Finally, the value of α used in our method can be any element of the ground field GF(2^n), except the values of 0 and 1, and therefore can be chosen to reduce the hardware overhead of our proposed method. In particular, choosing $\alpha + 1 = p_0$ the resource usage of our method will require four multiplications in the ground field GF(2^n), four constant multiplications and nine additions. With this choice the value of p_0H can be used both for computing C'_0 and C''_0 and for the computation of C''_1. In Fig. 2 the structure of the proposed multiplier is presented.

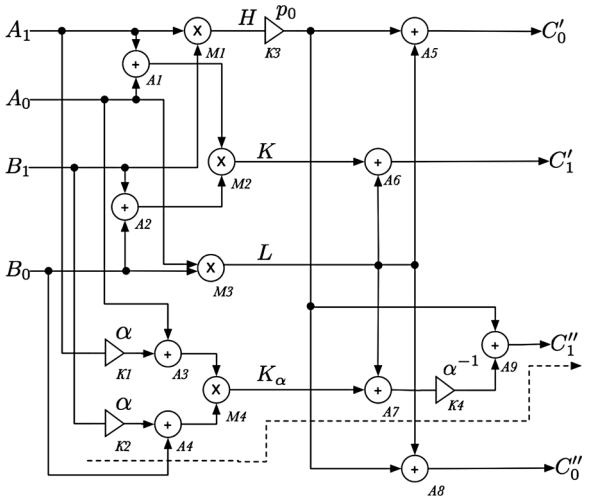


Fig. 2 *Concurrent error detection Karatsuba multiplier*

As an example, the multiplier over GF($(2^3)^2$) is explained below.

Example 1: The field polynomial of the ground field GF(2^3) is $Q(y) = y^3 + y + 1$. The elements of the ground field in the polynomial and exponential forms are shown in Table 1.

The composite field is generated by the polynomial $p(x) = x^2 + x + p_0$ with $p_0 = \omega^6$. Now we take two elements of the composite field $A(x) = \omega^5x + \omega^2$ and $B(x) = \omega^3x + \omega^4$ and compute the intermediate values L, H, K and K_α.

We obtain $L = \omega^6$, and $H = \omega^8 = \omega$. For K we obtain $K = (\omega^5 + \omega^2)(\omega^3 + \omega^4) = \omega^3\omega^6 = \omega^9 = \omega^2$.

For K_α we use $\alpha = \omega^6 + 1 = \omega^2$ and we obtain $K_\alpha = (\omega^2 + \omega^2\omega^5)(\omega^4 + \omega^2\omega^3) = (\omega^2 + \omega^7)(\omega^4 + \omega^5) = (\omega^2 + 1)(1) = \omega^6$.

From the values of L, H, K and K_α we compute $C'(x)$ and $C''(x)$ obtaining the following results

$$\begin{aligned} C'(x) &= (K + L)x + L + p_0H \\ &= (\omega^2 + \omega^6)x + \omega^6 + \omega^6\omega \\ &= (1)x + \omega^6 + \omega^7 = x + \omega^2 \end{aligned}$$

while for $C''(x)$ we obtain

$$\begin{aligned} C''(x) &= (\alpha^{-1}(K_\alpha + L) + (\alpha + 1)H)x + L + p_0H \\ &= (\omega^{-2}(\omega^6 + \omega^6) + \omega^6\omega)x + \omega^2 \\ &= (0 + \omega^7)x + \omega^2 \\ &= (0 + 1)x + \omega^2 \end{aligned}$$

And the two results are identical as expected.

3.2 Resource estimation

An estimation of the resource of the multiplier shown in Fig. 2*b* can be done starting from the (4) and (5) reported in Section 2. The four multipliers require approximately $4n^2$ AND and XOR, while the nine adders require $9n$ XOR gates. The area of the constant multiplier can be estimated as n. In fact, the data of Table 6.1 of [19] gives a number of XOR for p_0 that is usually less than n and is equal to n in the worst cases. The overhead with respect to the basic implementation without error-detection capabilities is about the 33%, while the duplication and comparison approach will require more than 100% of overhead.

In order to evaluate the delay of the multiplier, its critical path (the dash line) has been depicted in Fig. 2. This path is composed by three adders, one multiplier and two constant

Table 1 Elements of GF(2^3)

Integer	Polynomial	Exponential
0	0, 0, 0	0
1	0, 0, 1	ω^0
2	0, 1, 0	ω^1
3	0, 1, 1	ω^3
4	1, 0, 0	ω^2
5	1, 0, 1	ω^6
6	1, 1, 0	ω^4
7	1, 1, 1	ω^5

doi: 10.1049/iet-cds.2011.0278

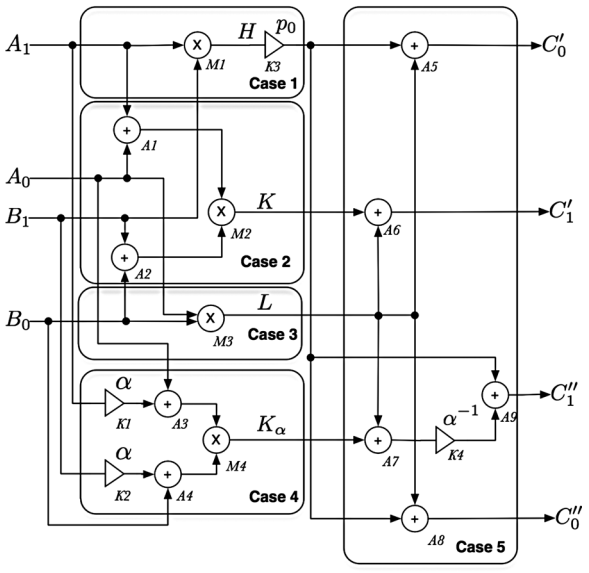


Fig. 3 *Five cases of the error analysis for the concurrent error detection Karatsuba multiplier*

multipliers. The delay of the adders is $3T_X$, the delay of the multiplier in the GF(2^n) field is $2\lceil \log_2 n \rceil T_X + T_A$ and the delay of each constant multiplier can be estimated as T_X. In fact, since the only restriction to the value of α is that it must be different from 0 and 1, it is easy to choose a value of α such that the constant multiplications for α and for α^{-1} require a minimum delay. Therefore the total delay can be estimated as

$$\begin{aligned} T &= (5 + 2\lceil \log_2 n \rceil)T_X + T_A \\ &= (5 + 2\lceil \log_2 m/2 \rceil)T_X + T_A \\ &= (3 + 2\lceil \log_2 m \rceil)T_X + T_A \end{aligned} \quad (16)$$

Our method allows superseding some of the disadvantages of the methods based on single and/or multiple parity bits. For example, in [1] only one parity bit is used and therefore this method can detect only the presence of an odd number of erroneous bits. Also with more than one parity bit [9], the 100% fault coverage is not assured. Moreover, the area overhead of the methods using multiple parity bits is linearly dependent by the number of parity bits used. With a limited number of parity bits, the overhead in percentage is less than the one proposed in this paper. Also in the other papers like [5, 7, 9] the overhead in percentage is very low. However, with respect to the total number of gates, our method achieves similar results compared with the others. In fact, the number of gates of the concurrent error-detection Karatsuba-based multiplier is very close to the number of gates of a direct implementation of a GF(2^{2n}) standard multiplier.

In Table 2 the complexity in terms of number of gates and the delay of our implementation are reported and compared with the ones of the multipliers proposed in [7] for fields of type GF(2^{2m}) with $m = 2n$. The value of h in Table 2 is the hamming weight of the generator polynomial. Therefore $h \geq 3$, since the generator polynomial is at least a trinomial.

From the data of Table 2, it can be seen that both the area occupation and the delay of our method is comparable with the ones of the multipliers based on parity check bits. For the space complexity, in all the implementation taken into account the quadratic factor m^2 dominates and therefore, the area occupation of all the implementation is similar when the value of m increases. For the time complexity, the traditional bit parallel is the one with the worst performances, since its complexity is dominated by m, while the low-complexity bit parallel and the Karatsuba multipliers have the same delays, since when m increases, the value of $\lceil \log_2(m^2 - m) \rceil$ tends to $2\lceil \log_2 m \rceil$, that is same value of the formula for the evaluation of the Karatsuba multiplier delay.

4 Error detection in the concurrent error-detection Karatsuba multiplier

To demonstrate the ability of our method to detect an error inside the multiplier, we analyse all the cases in which an error can occur. We suppose that a fault affects one of the basic arithmetic operations composing the concurrent error-detection Karatsuba multiplier. From Fig. 2 the basic arithmetic operations are the four multipliers named *Mi*, the four constant multipliers named *Ki* and the nine adders named *Ai*. The erroneous result can affect only one of the two final alternative values $C'(x)$ and $C''(x)$ or both the alternative values with a different magnitude. In particular, it will be shown that an error affecting the value of L compromises both the values of C'_1 and C''_1, but the circuit is still able to detect an error in L comparing those values, since the error in C'_1 and C''_1 differs by a multiplicative factor α^{-1}. We examine what happen when an error occurs dividing the analysis in five different cases. The first four cases correspond to an error in one of the intermediate values H, K, L, K_α, while the last case analyses an error affecting one output. The analysis shows that it is always possible to detect a fault in each of the basic arithmetic operations composing the multiplier. The analysis of the final self-checking comparator is omitted, since its behaviour is well known [22]. Fig. 3 shows the division of the circuit in the five different cases. In order to prevent fault attacks based on power glitches, each box in which the circuit in Fig. 3 should be powered by a different power source. In the same way, if a pipelined multiplier is used, a different clock distribution for each box allows preventing fault attacks based on the clock source.

Case 1: Error in p_0H. This computation is performed by the multiplier *M*1 and the constant multiplier *K*3. A fault in one of these elements affects the outputs C'_0, C''_1 and C''_0. The error can be detected since the values of C'_1 and C''_1 are different. In fact, C'_1 is error-free while C''_1 is erroneous.
Case 2: Error in computation of K. K is computed using the adders *A*1, *A*2 and the multiplier *M*2. This value is used

Table 2 Complexities and delay of finite-field multiplier with fault-detection capability

	Traditional bit parallel [7]	Low-complexity bit parallel [7]	Karatsuba
#AND	$m^2 + m$	$m^2 + 1$	m^2
#XOR	$m^2 + (h+1) \times m - h$	$m^2 + h \times (m - h + 2)$	$m^2 + 5m$
delay	$T_A + (m + \lceil \log_2(m+1) \rceil)T_X$	$T_A + (2h - 3 + \lceil \log_2(m^2 - m) \rceil)T_X$	$T_A + (3 + 2\lceil \log_2 m \rceil)T_X i$

only to compute C'_1 and therefore an error affects only the result of the $C'(x)$ computation, while the correct value of $C(x)$ is $C''(x)$. The comparison between $C'(x)$ and $C''(x)$ allow the detection of the error.

Case 3: Error in L. This computation is performed by $M3$. All the four outputs of the circuit can be affected by the error. However, now we will show how the magnitude of the error differs between C'_1 and C''_1 allowing the detection on the error comparing the two alternative values.

Let us define the error on L as $\bar{L} = L + e$, where e is an element of the ground field different from zero, L is the correct value of the computation and $\bar{L}$ is the value affected by an error resulting from the wrong computation of A_0B_0. The value of C'_1 is

$$C'_1 = K + \bar{L} = K + L + e = C_1 + e \tag{17}$$

whereas the value of C''_1 is

$$\begin{aligned} C''_1 &= \alpha^{-1}(K_\alpha + \bar{L}) + (\alpha + 1)H \\ &= \alpha^{-1}e + \alpha^{-1}(K_\alpha + L) + (\alpha + 1)H \\ &= \alpha^{-1}e + C_1 \end{aligned} \tag{18}$$

It can be noticed that imposing $\alpha^{-1} \neq 1$ then $C'_1 \neq C''_1$ therefore the error can be detected comparing these values.

Case 4: Error in computation of K_α. K_α is computed using the constant multipliers $K1$ and $K2$, the adders $A3$ and $A4$ and the multiplier $M4$. K_α is used only to compute C''_1 and therefore an error affects only the result of the $C''(x)$ computation, while the correct value of $C(x)$ is $C(x) = C'(x)$. The comparison between $C'(x)$ and $C''(x)$ allows the detection of the error.

Case 5: Error in computation of C'_0, C'_1, C''_1 and C''. An error in one of the outputs is due to a fault in the constant multiplier $K4$ or in one of the adders $A5$, $A6$ and $A7$, $A8$, $A9$. Here, an error affects only to one output, therefore can be detected comparing the erroneous output with the corresponding unaffected one.

To better describe the behaviour of the concurrent error-detection multiplier, two examples are reported below. Since the cases from 1 to 4 are simple to understand because the error affects only one of the components of $C'(x)$ and $C''(x)$, we provide these examples for explaining cases 1 and 3.

Example 2: Also in this example we use the same data of Example 1 and we suppose an error in H changing its value from ω to 0. We compute $C'(x)$ and $C''(x)$ obtaining the following results

$$\begin{aligned} C'(x) &= (K + L)x + L + p_0H \\ &= (\omega^2 + \omega^6)x + \omega^6 + \omega^6 \times 0 \\ &= \underbrace{(1)}_{C'_1} \times x + \underbrace{\omega^6}_{C'_0} \end{aligned}$$

$$\begin{aligned} C''(x) &= (\alpha^{-1}(K_\alpha + L) + (\alpha + 1)H)x + L + p_0H \\ &= (\omega^{-2}(\omega^6 + \omega^6) + \omega^6 \times 0)x + \omega^6 \\ &= \underbrace{(0 + 0)}_{C''_1}x + \underbrace{\omega^6}_{C''_0} \end{aligned}$$

Here, we obtain that $C''_0 = C'_0 \neq C_0$ and $C''_1 \neq C'_1$. The comparison between C''_1 and C'_1 allows detecting the error.

Example 3: Using the same composite field GF($(2^3)^2$) and the same values for $A(x)$ and $B(x)$ of Example 1, we suppose an error in L changing its value from ω^6 to ω^5. The error is $e = \omega$.

Now we compute $C'(x)$ and $C''(x)$ obtaining the following results

$$\begin{aligned} C'(x) &= (K + L)x + L + p_0H \\ &= (\omega^2 + \omega^5)x + \omega^5 + \omega^6\omega = \omega^3x + \omega^5 + \omega^7 \\ &= \underbrace{\omega^3}_{C'_1} \times x + \underbrace{\omega^4}_{C'_0} \end{aligned}$$

$$\begin{aligned} C''(x) &= (\alpha^{-1}(K_\alpha + L) + (\alpha + 1)H)x + L + p_0H \\ &= (\omega^{-2}(\omega^6 + \omega^5) + \omega^6\omega)x + \omega^5 + \omega^7 \\ &= (\omega^{-2}\omega + 1)x + \omega^4 \\ &= (\omega^6 + 1)x + \omega^4 \underbrace{\omega^2}_{C''_1} \times x + \underbrace{\omega^4}_{C''_0} \end{aligned}$$

The results of C''_1 and C'_1 are different from each other and are different from the right result $C_1 = 1$. In particular C'_1 and C''_1 assume the values of $C'_1 = 1 + \omega = \omega^3$ and $C''_1 = 1 + \omega^5 \cdot \omega = \omega^2$ as defined by (14) and (15).

5 Conclusions

In this paper, a technique for designing online fault-detection multiplier for composite finite fields has been proposed. The method uses an additional multiplication in the ground field and computes two alternative values of the multiplication result. The computation of the two values is performed by using the Karatsuba formula for composite finite field. The method assures that any error in the multiplier affects the two alternative values in a different way, allowing the detection of an error simply comparing the two results. The method requires only a small overhead and can detect any faults causing an error on one of the basic components composing the finite-field multiplier. Since this multiplier is also able to detect fault affecting multiple bits, or affecting clock and power lines, it is particularly suitable to protect cryptographic applications against fault-based side channel attacks.

6 References

1 Bayat-Sarmadi, S., Hasan, M.A.: 'Concurrent error detection of polynomial basis multiplication over extension fields using a multiple-bit parity scheme'. Proc. 20th IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems, DFT'05, October 2005, pp. 102–110

2 Chiou, C.W., Lee, C.Y., Deng, A., Lin, J.: 'Concurrent error detection in Montgomery multiplication over GF(2^n)', *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 2006, **89-A**, (2), pp. 566–574

3 Gaubatz, G., Sunar, B.: 'Robust finite field arithmetic for fault-tolerant public-key cryptography'. Proc. Third Int. Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC'06, 2006, pp. 196–210

4 Fenn, S., Gossel, M., Benaissa, M., Taylor, D.: 'On-line error detection for bit-serial multipliers in GF(2^m)', *J. Electron. Test.*, 1998, **13**, (1), pp. 29–40

doi: 10.1049/iet-cds.2011.0278

5 Lee, C.Y., Chiou, C.W., Lin, J.: 'Concurrent error detection in a polynomial basis multiplier over GF(2^n)', *J. Electron. Test.*, 2006, **22**, (2), pp. 143–150

6 Lee, C.Y., Chiou, C.W., Lin, J.: 'Concurrent error detection in a bit-parallel systolic multiplier for dual basis of GF(2^n)', *J. Electron. Test.*, 2005, **21**, (5), pp. 539–549

7 Reyhani-Masoleh, A., Anwar Hasan, M.: 'Fault detection architectures for field multiplication using polynomial bases', *IEEE Trans. Comput.*, 2006, **55**, (9), pp. 1089–1103

8 Hariri, A., Reyhani-Masoleh, A.: 'Fault detection structures for the Montgomery multiplication over binary extension fields'. Proc. Fourth Int. Workshop on Fault Diagnosis and Tolerance in Cryptography, 2007, FDTC'07, Vienna, Austria, 10 September 2007, pp. 37–46

9 Bayat-Sarmadi, S., Hasan, M.A.: 'On concurrent detection of errors in polynomial basis multiplication', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2007, **15**, (4), pp. 413–426

10 Chelton, W., Benaissa, M.: 'Concurrent error detection in GF(2^m) multiplication and its application in elliptic curve cryptography', *IET Circuits Devices Syst.*, 2008, **2**, (3), pp. 289–297

11 Cardarilli, G.C., Pontarelli, S., Re, M., Salsano, A.: 'Concurrent error detection in Reed-Solomon encoders and decoders', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2007, **15**, (7), pp. 842–846

12 Mozaffari Kermani, M., Reyhani-Masoleh, A.: 'A structure-independent approach for fault detection hardware implementations of the advanced encryption standard'. Fourth Int. Workshop on Fault Diagnosis and Tolerance in Cryptography, 2007, FDTC 2007, Vienna, Austria, 10 September 2007

13 Mozaffari Kermani, M., Reyhani-Masoleh, A.: 'Parity prediction of s-box for AES'. Proc. Canadian Conf. on Electrical and Computer Engineering, CCECE 2006, 2006, pp. 2357–2360

14 Bolchini, C., Salice, F., Sciuto, D.: 'A novel methodology for designing TSC networks based on the parity bit code'. European Design and Test Conf., ED&TC 97, 17–20 March 1997, pp. 440–444

15 Touba, N.A., McCluskey, E.J.: 'Logic synthesis techniques for reduced area implementation of multilevel circuits with concurrent error detection'. Proc. 1994 IEEE/ACM Int. Conf. on Computer-Aided Design, 1994, pp. 651–654

16 Wang, L.T., Wu, C.W., Wen, X.: 'VLSI test principles and architectures: design for testability' (Morgan Kaufmann, 2006)

17 Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: 'The sorcerers apprentice guide to fault attacks', IACR Cryptology ePrint Archive. Available at: http://eprint.iacr.org/2004/, 2004

18 Karatsuba, A., Ofman, Y.: 'Multiplication of many-digital numbers by automatic computers', *Doklady Akad. Nauk SSSR*, 1962, **145**, pp. 293–294

19 Paar, C.: 'Efficient VLSI architectures for bit-parallel computation in Galois fields'. PhD thesis, (English translation), Institute for Experimental Mathematics, University of Essen, Essen, Germany, June 1994

20 Paar, C.: 'A new architecture for a parallel finite field multiplier with low complexity based on composite fields', *IEEE Trans. Comput.*, 1996, **45**, (7), pp. 856–861

21 Sunar, B.: 'A generalized method for constructing subquadratic complexity GF(2k) multipliers', *IEEE Trans. Comput.*, 2004, **53**, (9), pp. 1097–1105

22 Lala, P.K.: 'Self-checking and fault-tolerant digital design' (Morgan Kaufmann, San Francisco, 2001)

23 Afanasyev, V.B.: 'On the complexity of finite field arithmetic'. Proc. Fifth Joint Soviet-Swedish Int. Workshop on Information Theory, Moscow, USSR, January 1991, pp. 9–12 **Q3**

www.ietdl.org

CDS20110278

Author Queries

S. Pontarelli, A. Salsano

Q1 Please check the initial citation of Fig. 3.

Q2 Please check the inserted main caption for Fig 1.

Q3 Reference [23] is uncited. Please cite or delete from the list.