Secure and Efficient Architectures for Single Exponentiations in Finite Fields Suitable for High-Performance Cryptographic Applications

Reza Azarderakhsh, *Member, IEEE,* Mehran Mozaffari-Kermani, *Member, IEEE,* and Kimmo Järvinen

***Abstract*—High performance implementation of single exponentiation in finite field is crucial for cryptographic applications such as those used in embedded systems and industrial networks. In this paper, we propose a new architecture for performing single exponentiations in binary finite fields. For the first time, we employ a digit-level hybrid-double multiplier proposed by Azarderakhsh and Reyhani-Masoleh for computing exponentiations based on square-and-multiply scheme. In our structure, the computations for squaring and multiplication are uniform and independent of the Hamming weight of the exponent; considered to have built-in resistance against simple power analysis attacks. The presented structure reduces the latency of exponentiation in binary finite field considerably and thus can be utilized in applications exhibiting high-performance computations including sensitive and constrained ones in embedded systems used in industrial setups and networks.**

***Index Terms*—Cryptography, double-hybrid multiplier, exponentiation, Gaussian normal basis (GNB), high-performance, security.**

I. Introduction

INFORMATION security in constrained applications such as embedded systems, industrial networks, and sensitive automation structures needs cryptographic measures implemented efficiently [2]–[6]. It is often desired to perform these computations in dedicated hardware platforms in order to achieve the required levels of performance. Exponentiation in finite fields is of much importance and is widely used in many sensitive applications such as the public key cryptography and the Reed–Solomon codes [7]. Consequently, many research works have been focused on low-complexity and high-performance designs of the arithmetic units for cryptosystems [8]–[17]. Single exponentiation is a computation of the form A^K, where A is a field element and K is a positive integer. There are several schemes for computing single exponentiations including the square-and-multiply method, the M-ary method, the sliding window method, the fixed-base comb method, and methods using precomputations (see [7], [18]–[23]). The square-and-multiply method is widely used in embedded devices as it is simple and has low memory requirements. However, this method has known weaknesses against side-channel attacks [e.g., simple power analysis (SPA)] and, hence, its modified versions, e.g., square-and-multiply-always method and Montgomery's powering ladder, are preferred instead [24], [25]. The original concept of power analysis attacks against exponentiations involved the consideration of distinguishable operations for multiplication and squaring [24]. The sequences of these operations depend on the bits of the exponents and an attacker can easily obtain the exponent by performing a SPA.

In this paper, we propose a new scheme to compute single exponentiations using a digit-level hybrid-double multiplier proposed in [1]. In [1], a hybrid-double multiplier is presented which has been employed for the computation of double-exponentiation. In this paper, we employed hybrid-double multiplier architecture and proposed a hardware architecture for the computation of single-exponentiation. We first decompose the exponent into two parts and perform a double exponentiation. Then, using the hybrid-double multiplier we perform multiplications concurrently and reduce the latency of computing single exponentiation considerably. We also analyze the resistivity of the architecture against side-channel attacks. As a result of this analysis, we conclude that the presented structure offers excellent protection against common side-channel attacks.

The remaining part of this paper is organized as follows. In Section II, we provide preliminaries of the multiplication over Gaussian normal basis (GNB) and revisit hybrid-double multipliers. In Section III, we propose a new architecture for computing single exponentiations by using hybrid-double multipliers. In Section IV, we investigate the resistivity of the presented structure against side-channel analysis attacks. Section V presents the results of our application-specific integrated circuit (ASIC) syntheses. Finally, we conclude this paper in Section VI.

Manuscript received July 11, 2014; revised October 22, 2014; accepted December 17, 2014. Date of publication January 6, 2015; date of current version February 17, 2015. This work was supported by the Texas Instruments Faculty Award granted to R. Azarderakhsh and M. Mozaffari-Kermani. The work of K. Järvinen was supported in part by the Department of Information and Computer Science, Aalto University, and in part by the Academy of Finland under Project 138358. This paper was recommended by Associate Editor R. Karri.

R. Azarderakhsh is with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: rxaeec@rit.edu).

M. Mozaffari-Kermani is with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: m.mozaffari@rit.edu).

K. Järvinen is with the Department of Information and Computer Science, Aalto University, Espoo 02150, Finland (e-mail: kimmo.jarvinen@aalto.fi).

Digital Object Identifier 10.1109/TCAD.2014.2387866

II. Preliminaries

In this section, we present the GNB. Moreover, hybrid-double digit-level GNB multiplier and exponentiation are discussed.

A. GNB

The GNB is a special class of normal basis which provides low-complexity multiplication matrix whenever $m \neq 0 \mod 8$.

Let us assume that m (field sizes) and T be positive integers such that $p = mT + 1$ is prime and $\gcd(Tm/k, m) = 1$, in which, we denote the multiplication order of 2 modulo p as k. Let α be a primitive $mT + 1$th root of unity in $GF(2^{Tm})$. Then, for any primitive Tth root of unity τ in $\mathbb{Z}_p$, $\beta = \sum_{i=0}^{T-1} \alpha^{\tau^i}$ generates a normal basis of $GF(2^m)$ over $GF(2)$ given by $N = \{\beta, \beta^2, \ldots, \beta^{2^{m-1}}\}$, which is called a GNB of type T [26]. Then, one can represent an element $A = (a_0, a_1, \ldots, a_{m-1}) \in GF(2^m)$, as $A = \sum_{i=0}^{m-1} a_i\beta^{2^i}$ with $a_i \in \{0, 1\}$. In GNB, squaring is right cyclic shift as $\beta^{2^m} = \beta$. It is noted that implementation of squaring is very efficient in hardware.

Consider A and B as two field elements which are represented by GNB over $GF(2^m)$. Then, addition is obtained by XORing field elements A and B as $A+B = \sum_{i=0}^{m-1}(a_i \oplus b_i)\beta^{2^i}$. The multiplication $C = A \times B$, is a bit complex over GNB which is based on a multiplication matrix $R_{(m-1)\times T}$.

Definition 1: Let $p := Tm + 1$. Then, every $k \in GF(p)\backslash\{0\}$ can be written uniquely as

$$k = 2^i u^j \mod p$$

where u is a primitive Tth root of unity, and $1 \le i < m-1$, $1 \le j < T$. Assume F to be the map such that $F(k) = F(2^i u^j) = i$ for all $k \in GF(p)\backslash\{0\}$. Then, we can obtain the first coordinate of C as

$$c_0 = a_0 b_1 + \sum_{i=1}^{m-1} a_i \left(\sum_{j=1}^{T} b_{R(i,j)} \right) \tag{1}$$

where $R(i, j)$, $0 \le R(i, j) \le m-1$, $1 \le i \le m-1$, and $1 \le j \le T$ denotes the (i, j)th element of $R_{(m-1)\times T}$ matrix. Then, one needs to shift the input operands A and B to obtain the other coordinates of C. The identity element of addition, i.e., 0, is $(0, 0, \ldots, 0, 0)$ and the identity element of multiplication, i.e., 1, is $(1, 1, \ldots, 1, 1)$ as $1 = \beta + \beta^2 + \beta^{2^2} + \cdots + \beta^{2^{m-1}}$.

B. Hybrid Double Digit Level GNB Multiplier

In [1], a new digit-level interleaving method for computing double multiplication (to perform $A \times B \times D$ on $GF(2^m)$ to get the result E) is presented over GNB. Based on this method, the hybrid-double multiplier performs two normal multiplications simultaneously with two interleaved digit-serial multipliers. The result of the double multiplication is computed after $M = \lceil m/d \rceil + 1$ clock cycles (in parallel) assuming that one clock cycle is required to load the output of the first multiplier to the input of the second multiplier. The architecture for this multiplier is shown in Fig. 1. As shown it is composed of a digit-level parallel-input and serial-out multiplier and a digit-level serial-input and parallel-output multiplier. For the operation presented hybrid-double multiplier, registers $\langle X \rangle$, $\langle Y \rangle$, and $\langle F \rangle$ required to be preloaded with operands A, B, and D, respectively, and the register $\langle Z \rangle$ should be cleared to zero, $0 \in GF(2^m)$. Then, the product of double multiplication E is available after M clock cycles in the register $\langle Z \rangle$. The area and time complexities of the presented structure for hybrid double multiplier are as given in the following proposition.

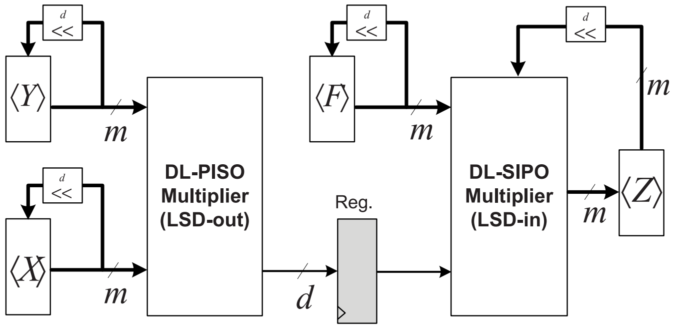

Fig. 1. Architecture of the hybrid-double multiplier [1].

Proposition 1: The architecture of hybrid double multiplier of [1] requires $\le 2v_s(T-1)+2dm-d$ XOR gates ($v_s = d(m-1) - (d(d-1)/2)$), $2dm$ AND gates, four registers (m-bit), and a d-bit register. Also, the critical path delay (CPD) for this multiplier structure is $t_{CP} = T_{\text{AND}} + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil) T_{\text{XOR}}$.

Then, one can obtain the time of multiplication of double multiplication based on the above proposition as $T = t_{CP} \times M$ [1]. For details about the operation of this multiplier, one can refer to [1].

C. Single Exponentiation

Assume P is an m-bit positive integer with binary expansion representation given as $P = (1P_{m-2} \cdots P_1 P_0)_2 = 2^{m-1} + \sum_{i=0}^{m-2} 2^i P_i$. Let $A \in GF(2^m)$, then the computation of A^P is called single exponentiation which can be computed as

$$A^P = A^{\left(2^{m-1} + \sum_{i=0}^{m-2} 2^i P_i\right)} = A^{2^{m-1}} \cdot \prod_{i=0}^{m-2} A^{2^i} P_i. \tag{2}$$

The most well-known scheme for computing (2) is called the square-and-multiply method (also called the binary method). It scans the bits of the exponent P one-by-one either from the left to the right (most significant bit (MSB)-first method) or from the right to the left [least significant bit (LSB)-first method] based on Horner's rule. Both schemes require $m - 1$ iterations. On each iteration, only squaring is computed if the bit is zero and both squaring and multiplication are computed if the bit is one. Therefore, the square-and-multiply method requires $m - 1$ squarings and $H(P) - 1$ field multiplications, where $H(P)$ is the Hamming weight of the binary representation of the exponent P.

Because the square-and-multiply method performs different operations depending on the values of the exponent bits, it is possible to extract the exponent by mounting a SPA attack that determines which operations take place during the computation. A simple method to fix this issue is to always compute multiplications but to save the results only if the bits are ones. This solution, called square-and-multiply-always, is costly and offers only limited protection because the attacker can reveal the exponent if he/she is able to distinguish when the result is saved and when not [7], [24]. Square-and-multiply-always is also unsafe against fault attacks. Faults during multiplications differentiate real multiplications from dummy multiplications because only faults in real multiplications corrupt the result of the single exponentiation. More sophisticated algorithms, such as Montgomery's ladder [25], solve the security issues but still come with a significant performance penalty.

D. Double Exponentiation

Assume that A and B are two finite field elements of binary characteristic field, $GF(2^m)$, and $K = (K_{m-1}\cdots K_1K_0)_2$ and $Q = (Q_{m-1}\cdots Q_1Q_0)_2$ represent two arbitrary positive integers. Then, the double exponentiation is the computation of A^KB^Q. It is a crucial operation for many applications in cryptography and coding (such as various methods for signature verifications) [25]. Computing double exponentiations based on traditional schemes, i.e., by computing two single exponentiations and multiplying their results, is not efficient. In [1], computing double exponentiations by using hybrid-double multipliers was presented for the first time. In the following, we employ double exponentiations along with hybrid-double multipliers in order to accelerate the computation of single exponentiations.

III. New Architecture for Computing Single Exponentiation Using Hybrid-Double Multiplier

In this section, we propose an architecture for computing single exponentiations by using double exponentiations with a hybrid-double multiplier. Let A be a field element in $GF(2^m)$ over an even type-T GNB and let P be a positive m-bit integer and $P = (p_0, p_1, \ldots, p_{m-1})_2$ its binary representation. Assume that B is represented by $B = A^{2^{\lceil m/2\rceil}}$ and P is represented by

$$P = p_02^0 + p_12^1 + \cdots + p_{m-1}2^{m-1} = K + 2^{\lceil \frac{m}{2}\rceil}Q \qquad (3)$$

where

$$\begin{aligned} K &= k_0 + k_12 + \cdots + k_{\lceil \frac{m}{2}\rceil-1}2^{\lceil \frac{m}{2}\rceil-1}\\ Q &= q_0 + q_12 + \cdots + q_{\lceil \frac{m}{2}\rceil-1}2^{\lceil \frac{m}{2}\rceil-1}\\ k_i &= p_i\\ q_i &= p_{\lceil \frac{m}{2}\rceil+i}. \end{aligned}$$

The exponents K and Q are obtained by simply splitting P into two equally long halves which does not require any computation and, hence, does not result in extra latency compared to the straightforward exponentiation schemes, such as the square-and-multiply method. If m is an odd integer, the coefficient $q_{\lceil m/2\rceil-1}$ is appended by zeros at its most significant bit end. Thus, computing the single exponentiation of the form A^P yields the computation of the following double exponentiation:

$$\begin{aligned} A^P &= A^{K+2^{\lceil \frac{m}{2}\rceil}Q} = A^KB^Q\\ &= \left(A^{k_0}B^{q_0}\right)^{2^0}\left(A^{k_1}B^{q_1}\right)^{2^1}\cdots\left(A^{k_{\lceil \frac{m}{2}\rceil-1}}B^{q_{\lceil \frac{m}{2}\rceil-1}}\right)^{2^{\lceil \frac{m}{2}\rceil-1}} \end{aligned} \qquad (4)$$

where the bit-lengths of K and Q are half of the bit-length of P. In Algorithm 1, we propose a new algorithm to compute single exponentiation based on the split exponent employing a hybrid-double multiplier. The computation of double multiplication (Step 3.1 of Algorithm 1) needs to be performed efficiently using a hybrid-double multiplier. In Fig. 2, we propose an architecture for computing (4) based on the Algorithm 1. It is composed of three 4-to-1 and one 2-to-1 multiplexers, a hybrid-double multiplier, and successive squaring circuits. The successive squarers are only rewirings as the field elements are represented in normal basis. The computation of $B = A^{2^{\lceil m/2\rceil}}$ is a simple rewiring that performs the right cyclic shift by $B = (A \gg \lceil m/2\rceil)$ as shown in Fig. 2. The architecture starts from the LSB end of the integers K and Q, i.e., k_0 and q_0, respectively (Steps 1.3–1.6). Then, every consecutive two bits of both integers (k_iq_i and $k_{i+1}q_{i+1}$) are fed into the two 4-to-1 multiplexers on the top in Fig. 2 in order to select appropriate inputs for the digit-level hybrid-double multiplier. These inputs select the inputs of the hybrid-double multiplier (after performing successive squarings) to be 1, A, B, or AB. We note that if one would simply rotate the registers holding A, B, and AB, by i, then the successive squarers would, indeed, be only fixed rewirings.

Algorithm 1 Proposed Single-Exponentiation Algorithm

```
Inputs: $A \in GF(2^m)$ and $P = (p_0, p_1, \ldots, p_{m-1})$, $P > 1$.
Output: $C = A^P$.
1: initialize $B = A^{2^{\lceil \frac{m}{2}\rceil}}$ and
1.1: $K = k_0 + k_12 + \cdots + k_{\lceil \frac{m}{2}\rceil-1}2^{\lceil \frac{m}{2}\rceil-1}$, where $k_i = p_i$
1.2: $Q = q_0 + q_12 + \cdots + q_{\lceil \frac{m}{2}\rceil-1}2^{\lceil \frac{m}{2}\rceil-1}$,
        where $q_i = p_{\lceil \frac{m}{2}\rceil+i}$
1.3: if $(k_0q_0 = 00)$ $C_0 = 1$
1.4: if $(k_0q_0 = 01)$ $C_0 = B$
1.5: if $(k_0q_0 = 10)$ $C_0 = A$
1.6: if $(k_0q_0 = 11)$ $C_0 = AB$
2: for $i$ from 1 to $\lceil \frac{m}{2}\rceil - 1$ do
2.1: if $(k_iq_i = 00)$ $R_i = 1$, $R_{i+1} = 1$
2.2: if $(k_iq_i = 01)$ $R_i = B^{2^i}$, $R_{i+1} = B^{2^{i+1}}$
2.3: if $(k_iq_i = 10)$ $R_i = A^{2^i}$, $R_{i+1} = A^{2^{i+1}}$
2.4: if $(k_iq_i = 11)$ $R_i = (AB)^{2^{i+1}}$, $R_{i+1} = (AB)^{2^{i+1}}$
endfor
3: for $j$ from 1 to $(\lceil \frac{m}{2}\rceil - 1)/2$ do
3.1: $C_j = C_{j-1} \times R_i \times R_{i+1}$
endfor
3.2: $C = C_j$
4: return $C = A^P$
```

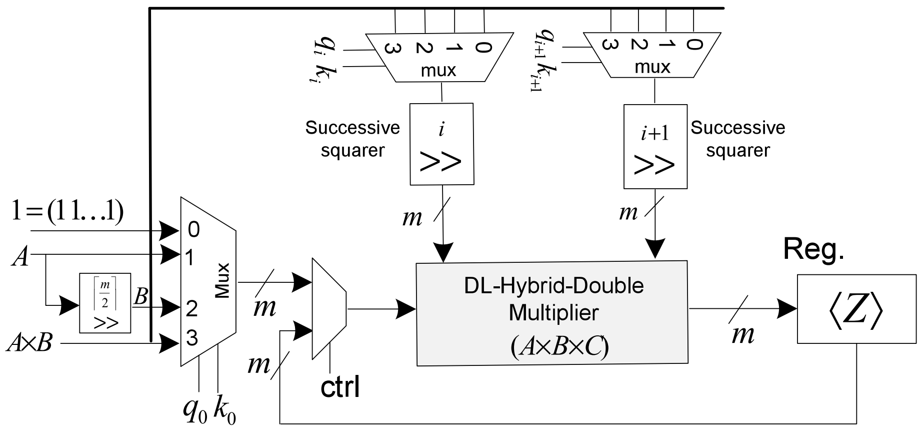

Fig. 2. Presented structure for computing single exponentiation using a hybrid-double multiplier.

TABLE I
CONTENTS OF THE REGISTER FOR COMPUTING $A^{104\,853}$ SINGLE EXPONENTIATION OVER $GF(2^{17})$ GIVEN IN EXAMPLE 1, WHERE $K = 405 = (110010101)_2$ AND $Q = 204 = (011001100)_2$

Iterations	i	k_i	q_i	$\langle Z \rangle$	Cost
initialize	0	1	0	A	no cost
1	1	0	0	$C_1 = A \times 1 \times (AB)^4$	1 double multiplication
	2	1	1	$= A^5B^4$	
2	3	0	1	$C_2 = A^5B^4 \times B^8 \times A^{16}$	1 double multiplication
	4	1	0	$= A^{21}B^{12}$	
3	5	0	0	$C_3 = A^{21}B^{12} \times 1 \times B^{64}$	1 double multiplication
	6	0	1	$= A^{21}B^{76}$	
4	7	1	1	$C_4 = A^{21}B^{76} \times (AB)^{128} \times A^{256}$	1 double multiplication
	8	1	0	$= A^{405}B^{204}$	

We provide the following example in order to better illustrate the operation of the presented structure.

Example 1: Let the element $A \in GF(2^{17})$ and $P = p_02^0 + p_12^1 + \cdots + p_{16}2^{16}$ be a positive 17-bit integer. Since $\lceil 17/2 \rceil = 9$, we set $B = A^{2^9}$, $K = k_02^0 + k_12^1 + \cdots + k_82^8$ and $Q = q_02^0 + q_12^1 + \cdots + q_82^8$, where $k_i = p_i$ for $0 \leq i \leq 8$, $q_i = p_{i+9}$ for $0 \leq i \leq 7$, and we zero-pad $q_8 = 0$. The exponentiation $C = A^P$ can be represented as $C = A^P = A^{K+2^9Q} = A^KB^Q$. Assume that we want to compute $A^{104\,853}$. We select $P = K + 2^9Q = 104\,853$, where $K = 405 = (110010101)_2$ and $Q = 204 = (011001100)_2$. Therefore, the computation of the single exponentiation is divided into computing a double exponentiation of the form $A^P = A^KB^Q = A^{11001100110010101} = A^{110010101}B^{011001100}$. In Table I, we provide the contents of the register $\langle Z \rangle$ in each iteration. First, we start from the LSB of the integers K and Q, i.e., $k_0 = 1$ and $q_0 = 0$, which select A to be the left input of the hybrid-double multiplier. Then, $q_1 = 0$ and $k_1 = 0$ are entered to the control input of the top multiplexer (left one) which select $1 \in GF(2^m) = (11, \ldots, 1)$ to be the input of the hybrid double multiplier. Next, $k_2 = 1$ and $q_2 = 1$ are fed to the control inputs of the multiplexer (top-right) to select AB to be shifted (as $AB \gg 2$) and fed into the third input of the hybrid-double multiplier. The hybrid double multiplier performs a double multiplication and the content of the register $\langle Z \rangle$ becomes $A \times 1 \times (AB)^4 = A^5B^4$ as illustrated in Table I. The process continues until it has processed the MSB of the elements K and Q. The result of the exponentiation becomes available after four iterations in the register $\langle Z \rangle = A^{405}B^{204} = A^{104\,853}$. In total, the proposed method requires $\lceil 9 - 1/2 \rceil = 4$ iterations to process 9 bits of the exponents K and Q, concurrently. This is possible due to the digit-level hybrid-double multiplier which performs two multiplications in each iteration with the latency of only one multiplication. Therefore, the presented structure performs only four double multiplications in computing a single exponentiation. The latency of the entire single exponentiation is approximately the same as the latency of four normal multiplications. Using the binary exponentiation method would yield a latency of eight normal multiplications.

A. Complexity Comparison and Discussion

The presented structure computes an exponentiation with the latency of $\lceil m - 1/4 \rceil$ double multiplications. We emphasize that the latency of double multiplication is the same as the one for single multiplication as described in Section II-B. In Table II, we compare our presented structure in terms of the latency and the required multipliers for computing single exponentiation to the leading ones available in [7], [24], [27], and [29]. As one can see in Table II, the latency of the presented structure is considerably less than those of the counterparts [7], [24], [27], [29]. Since the presented structure is scalable using the digit-size, one can also reduce the space complexity of the presented structure in comparison to the counterparts and perform a trade-off between the time and space complexities. The presented structure is approximately twice as fast with approximately twice the area which is a decent trade-off that is not possible with the current

TABLE II
COMPARISON OF VARIOUS SINGLE EXPONENTIATIONS ARCHITECTURES OVER $GF(2^m)$

Single-exponentiation							
Architecture	Basis	Latency (# of multiplications)	Mult. type	Area (# of mult.)	XOR	AND	Time
[7],[24],[29]	NB	$H(K) \approx m/2$	MO [30]	1	$dT(m-1)$	dm	$T_A + \lceil \log_2(Tm - T + 1) \rceil T_X$
[31]	NB	$H(K) + 1 \approx m/2 + 1$	MO [30]	2	$2dT(m-1)$	$2dm$	$T_A + \lceil \log_2(Tm - T + 1) \rceil T_X$
[31]	NB	$2H(K) + 1 \approx m + 1$	MO [30]	1	$dT(m-1)$	dm	$T_A + \lceil \log_2(Tm - T + 1) \rceil T_X$
This work	GNB	$\lceil \frac{m-1}{4} \rceil$	HD [1]	1	$\leq (2dm - d(d-1)) \times (T-1) + 2dm - d$	$2dm$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil) T_X$

schemes available in the literature once digit-sizes get large enough. Moreover, because the proposed scheme can employ smaller digit sizes in order to get the same latency, the actual implementation results are likely to be even better.

We are not aware of any implementations of binary field exponentiations over normal basis that would use the Chinese remainder theorem or nonadjacent form (NAF) and, hence, these comparisons are not possible. Using NAF is not an advantageous strategy for finite field exponentiations. NAF with signed bits is a good strategy, for instance, for elliptic curve cryptography because both addition (if the bit is $+1$) and subtraction (if the bit is -1) are equally fast in the additive group formed by the points on an elliptic curve. In the multiplicative groups of exponentiations, we need to compute a multiplication if the bit is $+1$ and a division if the bit is -1. Because division is significantly more expensive than multiplication, NAF is not directly applicable. This problem can be partly avoided with precomputations: for computing x^e, one precomputes $1/x$ and uses that in a left-to-right exponentiation algorithm whenever the bit is -1. Even this inversion is typically so expensive that it overweighs the benefits obtained from NAF. If x is fixed, then $1/x$ could be hardwired in the design and, then, NAF would be a good option. However, in this paper, we consider cases where x varies which rules out the NAF approach. Also, for the applications based on exponents of up to 200 bits and with free inversions in the group a typical choice is either the NAF or the signed sliding window method for a single exponentiation [31]. We should note that our presented method of computing single exponentiation has no restrictions in terms of the length of the exponent. Note that the Euclidean algorithms are in general slower than the square-and-multiply algorithms [31] as they require inversions.

It is worth mentioning that for embedded systems (often battery operated) there is an architectural advantage with the proposed method of computing single exponentiation here. The main objective of the proposed architecture is achieving high performance structures through reducing the latencies of computations. Our savings in the time of computations would make it possible to target high performance applications.

IV. SIDE-CHANNEL ANALYSIS OF THE PROPOSED ARCHITECTURE FOR SINGLE EXPONENTIATION

In side-channel attacks, an attacker analyzes, e.g., the power dissipation, electromagnetic radiation, or operation times of a cryptographic device in order to reveal secret parameters. The original concept of side-channel attacks against exponentiations involved the consideration of certain physical phenomena which allow differentiation between multiplication and squaring operations. The two most known attacks are the SPA and the differential power analysis (DPA) [32].

In this section, we survey the applicability of certain side-channel attacks against the presented structure. First, we focus on SPA which uses a single power trace in order to figure out the sequence of multiplications and squarings. Typically, it is assumed that the attacker is able to distinguish when multiplications and squarings are computed but that he/she cannot say anything about the values of the operands or results of these operations. Hence, the attacker can learn information about the value of the secret exponent only from the sequence of multiplications and squarings. The standard version of the proposed exponentiation algorithm computes a fixed number of squarings on each iteration. One hybrid-double multiplication is computed on each iteration expect when all bits processed during the iteration are zeros. This means that the attacker will not learn the values of the key bits if any of them is one, but if all of them are zeros, then a hybrid-double multiplication is not computed which is visible in a power trace and this immediately reveals the values of the processed key bits (all zeros) to an attacker. Hence, we point out that, in order to improve protection against these attacks, a hybrid-double multiplication must be computed also in the case where all bits processed in an iteration are zeros: $k_i = k_{i+1} = q_i = q_{i+1} = 0$. As a result of this dummy multiplication $\langle X \rangle \times 1 \times 1$, a hybrid-double multiplication is computed on each iteration of the algorithm regardless of the values of the key bits and the sequence of operations is constant. Consequently, when this countermeasure is implemented, the architecture is fully protected against traditional SPA which can distinguish multiplications and squarings from a power trace. The computation latency is also constant and, consequently, the architecture is fully protected against timing attacks. This countermeasure has negligible effect on performance because on average only $1/16$ of iterations require this dummy operation. This countermeasure comes without an area penalty because it does not require dummy registers or additional logic.

As pointed out above, the architecture offers excellent protection against basic side-channel attacks because the sequence of operations computed during an exponentiation is always the same regardless of the value of the secret exponent, i.e., $\lceil (m-1)/4 \rceil$ double multiplications. Next, we discuss certain more elaborated side-channel attacks and show that, although there are certain pitfalls which may leak information about the secret exponent to an attacker who can launch DPA attacks or highly accurate SPA attacks, the level of protection against these attacks is also very high. In many cryptosystems, e.g., digital signature algorithm (DSA) [33], exponentiations are computed using an ephemeral key which changes for each execution of the algorithm. This nullifies DPA attacks against exponentiations because the attacker can obtain only a single power trace. Even if DPA attacks are viable, known countermeasures, such as randomization of inputs (see [34]), apply without any changes also in this case, which thwarts DPA. Hence, in the following, we discuss only the resistivity against highly accurate SPA.

The potential weaknesses against highly accurate SPA stem from the fact that the inputs to the multiplexers remain constant throughout an exponentiation. If an attacker is able to exploit this feature and distinguish which input, 1, A, B, or $A \times B$, is used, then the attacker can find out information about the key bits. We make the widely-used assumption that a value x can be distinguished based on its Hamming weight $H(x)$. Because B is a rotation of A, $H(A) = H(B)$ which makes distinguishing between these two values very hard and it is safe to assume that it cannot be done via SPA. If A is a randomly selected element of $GF(2^m)$, then $H(A) \approx H(A \times B) \approx m/2$ with high probability and it will also be hard to distinguish these values from each other. Consequently, potential problems concentrate on the distinguishability of the value $1 = (111 \cdots 111)_2$, for which $H(1) = m$. Although we anticipate that even this value will be hard to distinguish via SPA, we make the pessimistic assumption that it is, indeed, possible and the attacker can distinguish these cases with 100% accuracy. We survey the consequences of this assumption next.

We divide this analysis in two cases: 1) the attacker is able to distinguish when a hybrid-double multiplication $\langle X\rangle \times 1 \times 1$ is computed and 2) the attacker is able to distinguish when either $\langle X\rangle \times \langle Y\rangle \times 1$ or $\langle X\rangle \times 1 \times \langle F\rangle$ are computed. The attacker immediately learns four bits of P (all zeros) when he/she encounters the case 1). On average, the attacker encounters the case 1) on every $1/16$ iterations. The case 2) divides into two sub-cases: 2a) where the attacker can distinguish which of the operands is 1 and 2b) where he/she can only say that one of the operands is 1 but does not know which one. The case 2a) immediately reveals two bits (both zeros) but leaves three options for the remaining two: (01), (10), and (11). The case 2b) does not directly reveal any bits but leaves only six options for the four bits: (0001), (0010), (0011), (0100), (1000), and (1100). On average, the attacker encounters the case 2) on every $3/8$ iterations.

As a result, we get the following effects on the security offered by one iteration. If the attacker can distinguish only the case 1), then the security per iteration is reduced from 4 bits to $\log_2(1/16 + 15/16 \times 16) \approx 3.91$ bits. If the attacker can distinguish the cases 1) and 2a), then the security is reduced to $\log_2(1/16 + 3/8 \times 3 + 9/16 \times 16) \approx 3.35$ bits. If the case is 2b) instead of 2a), then the reduction is slightly smaller: $\log_2(1/16 + 3/8 \times 6 + 9/16 \times 16) \approx 3.50$ bits. As a result of this analysis, we conclude that even in the worst case, the complexity is reduced only by approximately $2^{3.35-4} - 1 \approx -36\%$ and, hence, we are confident that the presented structure offers excellent protection against common passive side-channel attacks.

We do not anticipate active side-channel attacks to pose a much more serious risk for the architecture. Inducing a fault at any point of the computation including dummy multiplications always corrupts the entire result and, hence, does not give any additional information to an attacker. If the attacker can freely choose the value of A, he/she could select A so that $H(1) = m$, $H(A) = H(B)$, and $H(A \times B)$ differ enough from each other which, consequently, would allow him/her to differentiate 1, A or B, and $A \times B$ from a power trace. If ha/she is able to do this with 100% accuracy, then it will have more serious effects on the security provided by each iteration: $\log_2(1/4 + 1/2 \times 2 + 1/4 \times 4) \approx 1.17$ bits which is an approximately $2^{1.17-4} - 1 \approx -86\%$ reduction. However, we foresee that this attack would be difficult to mount in practice because differentiation accuracy would be considerably smaller than 100% (e.g., only some percents at best). It is also notable that even if the attacker is able to launch this very sophisticated active side-channel attack with 100% accuracy, it offers less information about the secret exponent than the traditional SPA against typical implementations using square-and-multiply. Consequently, the architecture offers very good protection against side-channel attacks.

V. Implementations Results and Discussions

In a number of applications which require sensitive embedded architectures, e.g., for mobile *ad hoc* networks (MANETs) in industrial setups, one needs to tailor the structures to exhibit the required security mechanisms. Secure MANETs are becoming more and more widely implemented in the industry [35]. Nevertheless, the open medium and remote distribution of MANETs make it prone to different types of attacks (intrusion detection mechanisms may be needed). As such, any security breach could result in loss of assets and needs to be avoided. The proposed architectures in this paper can be used in a number of security mechanisms such as DSA and key agreement through Diffie–Hellman protocol.

In order to validate the feasibility of the proposed architectures in embedded hardware (ASIC-based embedded architectures which can be used in networks in many applications such as industrial networks), we have synthesized our proposed architectures. We note that we have chosen the ASIC platform based on the resources available to us; because our presented schemes are not dependent on the hardware platform, similar overheads are expected for field-programmable gate array (FPGA) designs. Through the performed ASIC synthesis process, the performance and synthesis metrics in terms of hardware and timing are derived. We have used the STM 65-nm standard-cell library in the Synopsys Design Compiler [36]. We have used typical corner case for synthesis.

We have presented the results of our syntheses for three single exponentiation architectures for $m = 571$ for different digit sizes in Table III. In this table, the areas (in terms of μm^2),

TABLE III
ASIC SYNTHESIS RESULTS FOR THE PRESENTED SINGLE EXPONENTIATION ARCHITECTURES FOR $m = 571$ FOR DIFFERENT DIGIT SIZES USING 65-NM CMOS STANDARD TECHNOLOGY

d	q	$q+1$	Single Exponentiation using a Hybrid-double multiplier						
			Total Latency $(L)^1$	CPD $[ns]$	Time $[\mu\text{s}]$	Area $[\mu\text{m}^2]$	Area kGE	Time×area $[\mu s\text{m}^2]$	Energy $[\mu J]$
13	44	45	6,435	1.88	12.1	619,248	298	7,492.9	16.2
22	26	27	3,861	2.31	8.91	1,015,358	507	9,046.8	27.5
26	22	23	3,289	2.57	8.45	1,192,370	596	10,075.5	32.3

1. $L = (q+1) \times \lceil \frac{m-1}{4} \rceil$.

the maximum working frequencies represented by CPD in terms of *ns*, the total number of clock cycles needed for a complete process, i.e., latencies, the total times (product of total latency and CPD in terms of ms), the product of time and area, i.e., time $\times$ area (in terms of μsm^2), and the energies (μJ) have been depicted. In order to make the area results transferable when switching technologies, we have provided the NAND-gate equivalency (in terms of kilo gate equivalent, denoted as kGE). This is performed using the area of a NAND gate in the utilized library which is 2.08 μm^2. We note that the presented results are for $d = 13, 22, 26$ and $q = 44, 26, 22$ and the total latencies (L) are derived through $L = (q+1) \times \lceil m - 1/4 \rceil$, where m is the field size ($m = 571$).

As seen in Table III, the latencies for $d = 13, 22, 26$ are 6435, 3861, and 3289, respectively, which implies that as we increase the digit sizes, we need lower number of clock cycles to finish the operations. However, this comes at the expense of higher CPD for these three architectures, i.e., 1.88 ns, 2.13 ns, and 2.57 ns, respectively. These result in the total times of 12.1 ms, 8.91 ms, and 8.45 ms for the aforementioned digit sizes (the total times are decreased as digit sizes go up).

As seen in this table, the areas in terms of μm^2 (kGE) for $d = 13, 22, 26$ are 619 248 μm^2 (298 kGE), 1 015 358 μm^2 (507 kGE), and 1 192 370 μm^2 (596 kGE), respectively. In other words, increasing the digit sizes results in higher hardware complexity. Finally, the time $\times$ area (in terms of μsm^2) is increased if the digit sizes are increased, i.e., 7492.9 μsm^2, 9046.8 μsm^2, and 10075.5 μsm^2 for the three architectures, respectively. We would like to point out that based on the requirements and constraints imposed, the performance/implementation metrics to achieve, and the resources available (overheads tolerated) one can tailor the presented architectures to be used in different embedded systems deployed in a variety of applications including industrial usage models.

Finally, one should note that the cost of implementing a digital signature scheme depends on various aspects but common to them is that the cost is dominated by the exponentiation operation (or scalar point multiplication in the case of systems based on elliptic curves). Because we do not target any specific signature scheme but rather provide a general solution that can be used in a multitude of systems that use exponentiations in binary fields over normal basis, we cannot provide any specific timings for a signature scheme. It is noted that the other costs of signature schemes usually comprise computing a hash of a message and its timing depends significantly on the length of the message being signed and some arithmetic that combines the results of suboperations (for instance, the results of the exponentiation with the hash and secret key). The costs of these operations are independent of the cost of the exponentiation which we focused on this paper. For instance, in [37], a comparison of implementing DSAs is done between binary fields and prime fields. It has been shown that binary fields outperform prime fields in terms of both time and energy consumptions (occupying similar silicon area). Therefore, our proposed method of computing exponentiation on binary finite field is a step forward to accelerate the computation of digital signature on binary fields targeting high performance applications. We will explore the implementation of a complete cryptosystem based on our proposed method in our future work. We will also explore fault analysis attack defense mechanisms in our future work building on our previous work on reliability in crypto-systems (see [38]–[43]).

VI. CONCLUSION

In this paper, we presented a new scheme for computing single exponentiations employing a hybrid-double multiplier over GNB. The presented structure reduces the latency of computing single exponentiations significantly in comparison to the previously known works available in the literature which are suitable for security mechanisms in embedded systems and industrial networks. The presented structure has applications in high performance computations of single exponentiations for coding and cryptography. Moreover, we investigated the applicability of side-channel attacks and we concluded that the presented structure offers excellent protection against common side-channel attacks. The presented structure could also be used for the case where the field elements are represented using polynomial basis and this would result in a similar acceleration for the computation of single exponentiations. For future work, we believe that it would be interesting to perform a comparative power analysis of the proposed exponentiation scheme, a step-forward toward providing security for industrial/personal setups.

REFERENCES

[1] R. Azarderakhsh and A. Reyhani-Masoleh, "Low-complexity multiplier architectures for single and hybrid-double multiplications in Gaussian normal bases," *IEEE Trans. Comput.*, vol. 62, no. 4, pp. 744–757, Apr. 2013.

[2] M. Mozaffari-Kermani, M. Zhang, A. Raghunathan, and N. K. Jha, "Emerging frontiers in embedded security," in *Proc. IEEE Int. Conf. Embedded Syst. VLSI Design*, Pune, India, Jan. 2013, pp. 203–208.

[3] M. Zhang, M. Mozaffari-Kermani, A. Raghunathan, and N. K. Jha, "Energy-efficient and secure sensor data transmission using encompression," in *Proc. IEEE Int. Conf. Embedded Syst. VLSI Design*, Pune, India, Jan. 2013, pp. 31–36.

[4] Y. Xu *et al.*, "Distributed device networks with security constraints," *IEEE Trans. Ind. Informat.*, vol. 1, no. 4, pp. 217–225, Nov. 2005.

[5] A. Abu-Mahfouz and G. Hancke, "Distance bounding: A practical security solution for real-time location systems," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 16–27, Feb. 2013.

[6] B. Fabian, T. Ermakova, and C. Muller, "SHARDIS: A privacy-enhanced discovery service for RFID-based product information," *IEEE Trans. Ind. Informat.*, vol. 8, no. 3, pp. 707–718, Aug. 2012.

[7] J. Cheon, S. Jarecki, T. Kwon, and M. Lee, "Fast exponentiation using split exponents," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1816–1826, Mar. 2011.

[8] H. Fan and Y. Dai, "Fast bit-parallel $GF(2^n)$ multiplier for all trinomials," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 485–490, Apr. 2005.

[9] H. Fan and M. Hasan, "Subquadratic computational complexity schemes for extended binary field multiplication using optimal normal bases," *IEEE Trans. Comput.*, vol. 56, no. 10, pp. 1435–1437, Oct. 2007.

[10] R. Farashahi and M. Joye, "Efficient arithmetic on Hessian curves," in *Proc. 13th Int. Conf. Pract. Theory Public Key Cryptography (PKC)*, Piscataway, NJ, USA, 2010, pp. 243–260.

[11] A. Hariri and A. Reyhani-Masoleh, "Bit-serial and bit-parallel Montgomery multiplication and squaring over $GF(2^m)$," *IEEE Trans. Comput.*, vol. 58, no. 10, pp. 1332–1345, Oct. 2009.

[12] J. Imana, R. Hermida, and F. Tirado, "Low complexity bit-parallel multipliers based on a class of irreducible pentanomials," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 12, pp. 1388–1393, Dec. 2006.

[13] J. Adikari, V. S. Dimitrov, and L. Imbert, "Hybrid binary-ternary number system for elliptic curve cryptosystems," *IEEE Trans. Comput.*, vol. 60, no. 2, pp. 254–265, Feb. 2011.

[14] J. Adikari, V. S. Dimitrov, and K. U. Järvinen, "A fast hardware architecture for integer to τNAF conversion for Koblitz curves," *IEEE Trans. Comput.*, vol. 61, no. 5, pp. 732–737, May 2012.

[15] C. Doche and L. Imbert, "Extended double-base number system with applications to elliptic curve cryptography," in *Proc. Int. Conf. Cryptol. India (INDOCRYPT)*, Kolkata, India, 2006, pp. 335–348.

[16] M. Cenk, C. Nègre, and M. A. Hasan, "Improved three-way split formulas for binary polynomial and Toeplitz matrix vector products," *IEEE Trans. Comput.*, vol. 62, no. 7, pp. 1345–1361, Jul. 2013.

[17] M. Cenk, C. Nègre, and M. A. Hasan, "Improved three-way split formulas for binary polynomial multiplication," in *Proc. Conf. Sel. Areas Cryptography*, Toronto, ON, Canada, 2011, pp. 384–398.

[18] P. de Rooij, "Efficient exponentiation using procomputation and vector addition chains," in *Proc. Int. Conf. EUROCRYPT*, vol. 950. Perugia, Italy, 1995, pp. 389–399.

[19] Y. Han, "High-performance computing VLSI system for bit-parallel exponentiation in $GF(2^m)$," *Int. J. Electron.*, vol. 19, no. 5, pp. 243–252, 2013.

[20] M. A. G. Martinez, G. Morales-Luna, and F. Rodríguez-Henríquez, "Hardware implementation of the binary method for exponentiation in $GF(2^m)$," in *Proc. 4th Mex. Int. Conf. Comput. Sci. (ENC)*, Tlaxcala, Mexico, 2003, pp. 131–134.

[21] N. C. Cortés, F. Rodríguez-Henríquez, and C. A. C. Coello, "On the optimal computaion of finite field exponentiation," in *Proc. Ibero-Amer. Conf. Adv. Artif. Intell. (IBERAMIA)*, Puebla, Mexico, 2004, pp. 747–756.

[22] N. C. Cortés, F. Rodríguez-Henríquez, and C. A. C. Coello, "An artificial immune system heuristic for generating short addition chains," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 1–24, Feb. 2008.

[23] E. F. Brickell, D. M. Gordon, K. S. McCurley, and D. B. Wilson, "Fast exponentiation with precomputation (extended abstract)," in *Proc. Int. Conf. EUROCRYPT*, vol. 658. Balatonfüred, Hungary, 1993, pp. 200–207.

[24] N. Homma, A. Miyamoto, T. Aoki, A. Satoh, and A. Shamir, "Comparative power analysis of modular exponentiation algorithms," *IEEE Trans. Comput.*, vol. 59, no. 6, pp. 795–807, Jun. 2010.

[25] P. L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," *Math. Comput.*, vol. 48, no. 117, pp. 243–264, 1987.

[26] A. Menezes *et al.*, *Applications of Finite Fields*. Boston, MA, USA: Kluwer, 1993.

[27] L. A. Tawalbeh and S. Sweidan, "Hardware design and implementation of *ElGamal* public-key cryptography algorithm," *Inf. Security J. Glob. Perspect.*, vol. 19, no. 5, pp. 243–252, 2010.

[28] C. Lee, J. Lin, and C. Chiou, "Scalable and systolic architecture for computing double exponentiation over $GF(2^m)$," *Acta Appl. Math.*, vol. 93, no. 1, pp. 161–178, 2006.

[29] J.-L. Massey and J.-K. Omura, *Computational Method and Apparatus for Finite Field Arithmetic*. Berlin, Germany: Springer, 1986.

[30] C. Wang and D. Pei, "A VLSI design for computing exponentiations in $GF(2^m)$ and its application to generate pseudorandom number sequences," *IEEE Trans. Comput.*, vol. 39, no. 2, pp. 258–262, Feb. 1990.

[31] M. Stam and A. K. Lenstra, "Efficient subgroup exponentiation in quadratic and sixth degree extensions," in *Proc. 4th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, London, U.K., 2002, pp. 318–332.

[32] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. 9th Annu. Int. Cryptol. Conf. (CRYPTO)*, Santa Barbara, CA, USA, 1999, pp. 388–397.

[33] *Digital Signature Standard (DSS)*, NIST Standard FIPS PUB 186-4, Jul. 2013.

[34] J.-S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Cryptographic Hardware Embedded Systems* (Lecture Notes in Computer Science 1717). Berlin, Germany: Springer, 1999, pp. 292–302.

[35] E. M. Shakshuki, N. Kang, and T. R. Sheltami, "EAACK—A secure intrusion detection system for MANETs," *IEEE Trans. Ind. Electron.*, vol. 60, no. 3, pp. 1089–1098, Mar. 2013.

[36] (Jan. 2015). *Synopsys*. [Online]. Available: http://www.synopsys.com

[37] E. Wenger and M. Hutter, "Exploring the design space of prime field vs. binary field ECC-hardware implementations," in *Proc. 16th Nord. Conf. Secure IT Syst. Inf. Security Technol. Appl. (NordSec)*, Tallinn, Estonia, 2011, pp. 256–271.

[38] M. Mozaffari-Kermani, R. Azarderakhsh, and A. Aghaie, "Reliable and error detection architectures of Pomaranch for false-alarm-sensitive cryptographic applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, to be published.

[39] M. Mozaffari-Kermani, K. Tian, R. Azarderakhsh, and S. Bayat-Sarmadi, "Fault-resilient lightweight cryptographic block ciphers for secure embedded systems," *IEEE Embedded Syst. Lett.,* vol. 6, no. 4, pp. 89–92, Dec. 2014.

[40] S. Bayat-Sarmadi, M. Mozaffari-Kermani, and A. Reyhani-Masoleh, "Efficient and concurrent reliable realization of the secure cryptographic SHA-3 algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.,* vol. 33, no. 7, pp. 1105–1109, Jul. 2014.

[41] M. Mozaffari-Kermani, R. Azarderakhsh, C. Lee, and S. Bayat-Sarmadi, "Reliable concurrent error detection architectures for extended Euclidean-based division over $GF(2^m)$," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,* vol. 22, no. 5, pp. 995–1003, May 2014.

[42] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the advanced encryption standard," *IEEE Trans. Comput.,* vol. 59, no. 5, pp. 608–622, May 2010.

[43] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Parity-based fault detection architecture of S-box for advanced encryption standard," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Arlington, VA, USA, Oct. 2006, pp. 572–580.

Reza Azarderakhsh (M'12) received the B.Sc. and M.Sc. degrees in electrical and electronic engineering and computer engineering from the Sharif University of Technology, Tehran, Iran, in 2002 and 2005, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Western Ontario, London, ON, Canada, in 2011.

He joined the Department of Electrical and Computer Engineering, University of Western Ontario, as a Limited Duties Instructor, in 2011. He was a Natural Sciences and Engineering Research Council of Canada (NSERC) Post-Doctoral Research Fellow at the Center for Applied Cryptographic Research and the Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada. He is currently a faculty member with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY, USA. His current research interests include finite field and its application, elliptic curve cryptography, and pairing based cryptography.

Dr. Azarderakhsh was the recipient of the prestigious NSERC Post-Doctoral Research Fellowship in 2012.

Mehran Mozaffari-Kermani (M'11) received the B.Sc. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2005, and the M.E.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON, Canada, in 2007 and 2011, respectively.

He joined the Advanced Micro Devices, Markham, ON, Canada, as a Senior ASIC/Layout Designer, where he was integrating sophisticated security/cryptographic capabilities into a single accelerated processing unit. In 2012, he joined the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, as a Natural Sciences and Engineering Research Council of Canada Post-Doctoral Research Fellow. He is currently a faculty member with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY, USA. His research was funded by the Texas Instruments Faculty Award. His current research interests include emerging security/privacy measures for deeply embedded systems, cryptographic hardware systems, fault diagnosis and tolerance in cryptographic hardware, very large-scale integration reliability, and low-power secure and efficient FPGA and ASIC designs.

Dr. Mozaffari-Kermani was the recipient of the prestigious Natural Sciences and Engineering Research Council of Canada Post-Doctoral Research Fellowship in 2011 and the Texas Instruments Faculty Award in 2014. He serves as the Lead Guest Editor of the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING for the Special Issue of Emerging Security Trends for Deeply-Embedded Computing Systems 2014 and 2015. He is currently serving as the Technical Committee Member for a number of related conferences, including DFT, FDTC, RFIDsec, LightSEC, and WAIFI.

Kimmo Järvinen received the M.Sc. (Tech.) and the D.Sc. (Tech.) degrees from the Helsinki University of Technology (TKK), Espoo, Finland, in 2003 and 2008, respectively, both in electrical engineering.

He was at the Signal Processing Laboratory, TKK, from 2002 to 2008. From 2008 to 2014, he was at the Cryptology Group, Department of Information and Computer Science, Aalto University, Espoo. He had a three-year Post-Doctoral Researcher's project funded by the Academy of Finland, from 2011 to 2014. In 2014, he joined the COSIC Group of KU Leuven ESAT, Leuven, Belgium, where he currently has an FWO Pegasus Marie Curie Fellowship. His current research interests include efficient and secure realization of cryptosystems, elliptic curve cryptography, finite field arithmetic, general computer arithmetic, and FPGAs.