Hardware implementation of DBNS recoding for ECC processor

Thomas Chabrier, Danuta Pamula, Arnaud Tisserand

Hardware Implementation of DBNS Recoding for ECC Processor

Thomas Chabrier[1], Danuta Pamula[1,3] and Arnaud Tisserand[2,1]
IRISA[1], CNRS[2], INRIA Centre Rennes - Bretagne Atlantique, Univ. Rennes 1
6 rue Kérampont, BP 80518, F-22305 Lannion cedex, FRANCE
[3]Silesian University of Technology, ul. Akademicka 2A, 44-100 Gliwice, Poland.

thomas.chabier@irisa.fr, danuta.pamula@irisa.fr,
arnaud.tisserand@irisa.fr

***Abstract*—The paper presents arithmetic level protections for ECC processor against some side channel attacks. The proposed protection is based on random recodings of the secret key in the double base number system (DBNS). DBNS is a highly redundant and sparse number system. Here, the high redundancy level of DBNS is used to randomly modify on-the-fly the k_i digits during the scalar multiplication $[k]P$. The proposed solution leads to random numbers and orders of curve level operations (point addition, doubling and tripling) during the computation of $[k]P$ operations. Our random recoding method provides $[k]P$ computation time comparable to the best w-NAF recoding methods. But standard w-NAF recodings are deterministic ones while our solution is a random one.**

I. Introduction

Implementation of embedded cryptosystems has to face two types of attacks: theoretical attacks and physical attacks. The security against theoretical attacks is provided by the robustness of the mathematical problem behind the cryptosystem: the elliptic curve discrete logarithm problem (ECDLP) for elliptic curve cryptography (ECC) [1]. The security of the cryptosystem against physical attacks, which are considered as very strong threats in embedded security applications, should be provided by cryptosystem designers. Recently, very efficient side channel attacks (SCA) have been proposed such as power analysis [2] or electromagnetic radiations analysis.

Many countermeasures have been introduced to protect cryptosystems against some SCAs. Some methods have been proposed at the arithmetic level. For instance, addition chains [3], [4] allow to perform only one type of operation, point additions, during the scalar multiplication $[k]P$. Hence, addition chains prevent from some attacks based on simple power analysis (SPA).

In this paper, we propose on-the-fly random recodings of the scalar digits k_i using the double base number system (DBNS). The very high redundancy level of DBNS allows to randomly choose among several representations of the key digits k_i. Then the number and order of operations at the curve level (point addition, doubling and tripling) is randomized. This may prevent from some power or electromagnetic radiations attacks.

Notations and background on ECC and DBNS are provided in Sections II and III respectively. The proposed countermeasure method is described in Section V. Implementation details in FPGA and experimental results are reported in Sections VI and VII respectively.

II. Elliptic Curve Cryptography (ECC)

A complete presentation of ECC systems in presented in [1]. In this work, we use elliptic curves defined over the prime finite field $\mathbb{F}_p$. Arithmetic operations in $\mathbb{F}_p$ are performed modulo p a large (160–600 bits) prime number. Extension to ECC systems defined over the $\mathbb{F}_{2^m}$ finite field is simple. See [5] for finite fields basic theory and applications and [1, sec. 5.2] for hardware implementations of finite field arithmetic. An elliptic curve E over $\mathbb{F}_p$ is defined by the simplified Weierstrass equation (see [1, p. 78]):

$$E : y^2 = x^3 + ax + b,$$

where $(x, y) \in \mathbb{F}_p^2$ are the coordinates of point P on E, the curve parameters a and b, in $\mathbb{F}_p$, are such that $4a^2+27b^2 \neq 0$.

The addition (`ADD`) of two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ of E gives a third point $R = (x_3, y_3)$ also on E (the sum point $R = P + Q$). The computation of R coordinates involves arithmetic operations in $\mathbb{F}_p$. See [1, see p. 80] for a geometrical illustration. The addition of $P + P$ is called point doubling $2P$ (`DBL`), and is computed using a slightly algorithm. Point tripling $3P$ (`TPL`) can also be defined in order to speed up the scalar multiplication process.

In ECC protocols, the main operation is the scalar multiplication $[k]P = P + P + \cdots + P$ (with $k - 1$ additions) where k is a given integer (the secret/public key) and P a point on the curve E. The scalar k is a t-bit integer, $k = (k_{t-1}, \ldots, k_1, k_0)_2$. There are several algorithms for scalar multiplication (see [1, sec. 3.3]). Figure 1 presents the simplest $[k]P$ algorithm.

The theoretical security of ECDLP comes from the fact that given two points P and Q such that $Q = [k]P$, finding the integer k is not feasible in practice (for well chosen curves), see [1, sec. 4.1] for details and theoretical attacks.

Side channel attacks allow to extract secret informations from running devices by measuring and analyzing some physical parameters (power consumption, electromagnetic radiations, computation time). For instance, recording and analyz-

INPUT: $k = (k_{t-1}, \ldots, k_1, k_0)_2$ and $P \in E$
OUTPUT: $Q = [k]P$

1: $Q \leftarrow \infty$
2: **for** i **from** 0 **to** $t-1$ **do**
3: **if** $k_i = 1$ **then** $Q \leftarrow Q + P$ (ADD)
4: $P \leftarrow 2P$ (DBL)
5: **return** Q

Fig. 1. Right to left version of the double-and-add algorithm computing the scalar multiplication $Q = [k]P$ (from [1, p. 96]).

ing the instantaneous power consumption of a circuit may lead to very efficient power attacks [2]. Simple power analysis (SPA, see [2, chap. 5]) directly uses the fact that basic implementations of point addition and point doubling operations have different power traces. In the algorithm presented in Fig. 1, the point addition operation $Q \leftarrow Q + P$ is only performed for odd bits of the scalar. Thus, the recorded power traces will directly show where/when are the ADD operations in the sequence of DBL ones. Algorithms such as the double-and-add method are prone to this type of attacks.

III. Double Base Number System (DBNS)

The double base number system (DBNS) simultaneously uses two bases for representing numbers [6]. In this work, we use bases 2 and 3. The integer x is represented by:

$$x = \sum_{i=0}^{n-1} s_i 2^{a_i} 3^{b_i},$$

where $s_i \in \{-1, 1\}$ and $(a_i, b_i) \in \mathbb{N}^+$. Triplets (s_i, a_i, b_i) are called $\{2,3\}$-terms and are the "digits" of the DBNS representation. DBNS is a sparse (i.e. the number of $\{2,3\}$-terms is small) and very redundant (i.e. some numbers have several representations) number system. For instance, the value 127 has 783 different DBNS representations (from [7]):

$$\begin{aligned} 127 &= 2^2 3^3 + 2^1 3^2 + 2^0 3^0 \\ &= 2^2 3^3 + 2^4 3^0 + 2^0 3^1 \\ &= 2^3 3^2 + 2^1 3^3 + 2^0 3^0 \\ &= \cdots \end{aligned}$$

Most of arithmetic operations (addition, multiplication, division) are very complex and costly in DBNS. But there are a few very cheap operations: multiplication by 2 and 3. Those operations are very useful in ECC.

As suggested in [8], we use k recoded using DBNS chains to perform the scalar multiplication operation $[k]P$. As stated in [8], the scalar multiplication is efficiently computed using DBNS representations with decreasing exponents[1]:

$$a_0 \geq \cdots \geq a_{n-1} \quad \text{and} \quad b_0 \geq \cdots \geq b_{n-1}.$$

[1] a_0 is considered the largest exponent of powers of 2, while a_{n-1} is the smallest one (the same notation applies for b_i exponents of powers of 3).

Let $t_i = 2^{a_i} 3^{b_i}$, using a decreasing sequence of exponents, the value $k = \sum_{i=0}^{n-1} t_i$ can be factorized by t_{n-1}:

$$k = \left(\sum_{i=0}^{n-2} s_i t'_i + s_{n-1} \right) \times t_{n-1},$$

where $t'_i = 2^{a_i - a_{n-1}} 3^{b_i - b_{n-1}}$. This recursive factorization is similar to the Horner scheme.

For instance, $k = 15679$ can be recoded in DBNS with and without decreasing exponents:

$$\begin{aligned} 15679 &= 2^6 3^5 + 2^2 3^3 + 2^1 3^2 + 2^0 3^0 \\ &= 2^1 3^2 \left(2^1 3^1 \left(2^4 3^2 + 1\right) + 1\right) + 1 \end{aligned}$$

Without decreasing exponents, the computation cost of $[15679]P$ is $3 \cdot \texttt{ADD} + 9 \cdot \texttt{DBL} + 10 \cdot \texttt{TPL}$ operations while with decreasing exponents this cost reduces to only $3 \cdot \texttt{ADD} + 6 \cdot \texttt{DBL} + 5 \cdot \texttt{TPL}$ operations.

In [8], a scalar multiplication algorithm dedicated to DBNS representation with decreasing exponents is proposed. This algorithm is given in Fig. 2. The cost of this algorithm is $(n-1) \cdot \texttt{ADD} + a_0 \cdot \texttt{DBL} + b_0 \cdot \texttt{TPL}$ operations.

INPUT: $k = \sum_{i=0}^{n-1} s_i 2^{a_i} 3^{b_i}$ and $P \in E$
OUTPUT: $Q = [k]P$

1: $Q \leftarrow s_0 P$
2: **for** i **from** 0 **to** $n-2$ **do**
3: $u \leftarrow a_i - a_{i+1}$, $v \leftarrow b_i - b_{i+1}$
4: $Q \leftarrow [2^u]Q$, $Q \leftarrow [3^v]Q$
5: $Q \leftarrow Q + s_{i+1} P$
6: $Q \leftarrow [2^{a_{n-1}}]Q$, $Q \leftarrow [3^{b_{n-1}}]Q$
7: **return** Q

Fig. 2. Right to left scalar multiplication algorithm $[k]P$ with k represented in DBNS with decreasing exponents (from [8]).

If the last exponents of powers of 2 and 3 (respectively a_{n-1} and b_{n-1}) are equal to 0 then line 6 must not be executed.

IV. True Random Number Generator (TRNG)

TRNGs use a physical noise source such as radioactive decay, meta-stability, thermal noise or jitter variations in free running oscillators, to produce a random signal. Here, we use in-house TRNGs with on-line randomness monitoring to provide the random bits required for the selection of DBNS recoding rules. This kind of TRNG is based on oscillators sampling for the noise source (random jitter produced by one or several free running oscillators). See [9], [10] for details.

V. Proposed Arithmetic Countermeasure

During the scalar multiplication $[k]P$, the point level operations (i.e. ADD and DBL) are scheduled in a deterministic way based on k digits. NAF or w-NAF standard recodings produce deterministic schedules (see [1, p. 98]).

In this work, we use random representations of k in DBNS. This is possible because the DBNS representation is extremely

redundant. The conversion from the standard binary representation to DBNS is done on-the-fly. Then random recodings of are applied using the following identities:

$$\begin{aligned} \mathrm{I}_1: \quad 1+2 &= 3 \\ \mathrm{I}_2: \quad 1+3 &= 2^2 \\ \mathrm{I}_3: \quad 1+2^3 &= 3^2 \\ \mathrm{I}_4: \quad 1+1 &= 2 \end{aligned}$$

Those identities can be applied in two different ways: one can reduce (e.g. $1+2 \rightarrow 3$) or expand (e.g. $3 \rightarrow 1+2$) DBNS terms. Reduction of the number of DBNS terms may increase the $[k]P$ computation speed due to the fact that it reduces the number of point-level operations. However if the number of $\{2,3\}$-terms is too small, reduction rules based on the previous identities may not be applied.

Expansions of DBNS terms may introduce more randomness in the recoded scalar k representation, and consequently in the power traces. There should be a trade-off between the computation time and the amount of randomness introduced by expansions.

Furthermore, the previous identities can be combined with signed digits version of the DBNS representation. Thus there are two recoding rules for each identity:

$$\mathrm{I}_1 \Rightarrow \begin{cases} 2^{i+1}3^{j-1} + 2^i3^{j-1} = 2^i3^j & \mathrm{R}_1 \\ 2^{i-1}3^{j+1} - 2^{i-1}3^j = 2^i3^j & \mathrm{R}_2 \end{cases}$$

$$\mathrm{I}_2 \Rightarrow \begin{cases} 2^{i-2}3^{j+1} + 2^{i-2}3^j = 2^i3^j & \mathrm{R}_3 \\ 2^{i+2}3^{j-1} - 2^i3^{j-1} = 2^i3^j & \mathrm{R}_4 \end{cases}$$

$$\mathrm{I}_3 \Rightarrow \begin{cases} 2^{i+3}3^{j-2} + 2^i3^{j-2} = 2^i3^j & \mathrm{R}_5 \\ 2^{i-3}3^{j+2} - 2^{i-3}3^j = 2^i3^j & \mathrm{R}_6 \end{cases}$$

$$\mathrm{I}_4 \Rightarrow \begin{cases} 2^{i-1}3^j + 2^{i-1}3^j = 2^i3^j & \mathrm{R}_7 \\ 2^{i+1}3^j - 2^i3^j = 2^i3^j & \mathrm{R}_8 \end{cases}$$

Figure 3 illustrates some examples of DBNS recodings for the scalar $k = 140400$.

In some cases, some rules cannot be applied due to neighboring $\{2,3\}$-terms. For instance, if two consecutive terms, $k_i = (s_i, a_i, b_i)$ and $k_{i-1} = (s_{i-1}, a_{i-1}, b_{i-1})$, share the same exponent of the power of 2, i.e. $a_i = a_{i-1}$, then reductions based on rule R_1 cannot be applied.

Even for expansions, some problems may occur due to the fact that the sequence of exponents must be kept decreasing. In order to always ensure that the rules produce only decreasing sequences of exponents, three consecutive $\{2,3\}$-terms must be read for each rule application.

Below we provide a complete example of a situation where some rules should be discarded.

We use the k value presented in example Fig. 3. Starting with the initial DBNS representation $k = 140400 = 2^73^7 - 2^73^6 - 2^73^5 - 2^63^5 + 2^43^3$ (which is recoding 7 in Fig. 3), one may think that the reduction $2^73^6 + 2^73^5 \xrightarrow{\mathrm{R}_3} 2^93^5$ can be applied. But for this specific value of k, this reduction would produce a non-decreasing sequence of exponents. Indeed the new recoded DBNS representation would be $2^73^7 - 2^93^5 - 2^63^5 + 2^43^3$ which is not a decreasing exponents sequence of $\{2,3\}$-terms. This last DBNS representation would give the following computation sequence $2^43^4(2^23^1(2^13^0(-2^23^0 + 2^03^1) - 1) + 1)$.

VI. Implementation

Figure 4 presents the architecture of the implemented recoding unit. This unit on-the-fly recodes the scalar k using randomly chosen DBNS representations. The scalar k, represented in DBNS, is stored in a specific register as a sequence of $\{2,3\}$-terms (s_i, a_i, b_i). Two specific blocks check which rules, R_1 to R_8, can be applied. One block is dedicated to reductions and the other one to expansions. Both blocks take three consecutive $\{2,3\}$-terms as inputs. Using 3 random bits produced by a TRNG, one rule is randomly selected among the set of allowed rules and then applied. Once a rule is selected, the block checks if the application of rules produces a valid sequence of exponents.

In case of a reduction, one of the original two terms is modified accordingly to the selected rule (addition/subtraction on the exponents and sign adjustment), while the other term is deleted from the DBNS representation. In case of an expansion, the block checks that inserting a new term still lead to decreasing sequence of exponents.

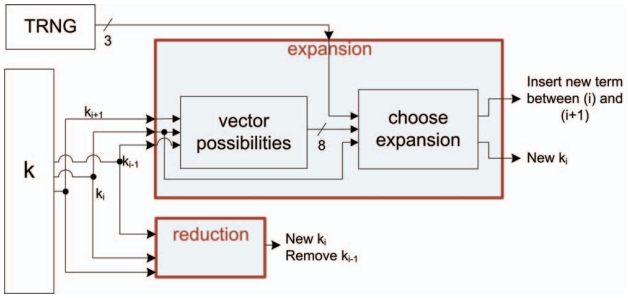

Fig. 4. Architecture of on-the-fly recoding unit with random DBNS representations.

The proposed recoding unit has been described in VHDL and implemented in a XC5VLX50T FPGA using the ISE version 11.4 design environment from Xilinx. The results reported in Table I have been obtained with standard efforts for both synthesis and place-and-route steps. The area required by the recoding unit is very small compared to the total area of the complete ECC processor. The recoding unit can work at a clock frequency greater than 200 MHz which is faster than the clock frequency of our arithmetic units in $\mathbb{F}_p$.

block	# registers	# LUTs	max. freq. [MHz]
expansion	93	352	283.5
reduction	26	101	234.6

TABLE I
Implementation results of the recoding unit on a XC5VLX50T FPGA (with 28 800 registers and 28 800 LUTs).

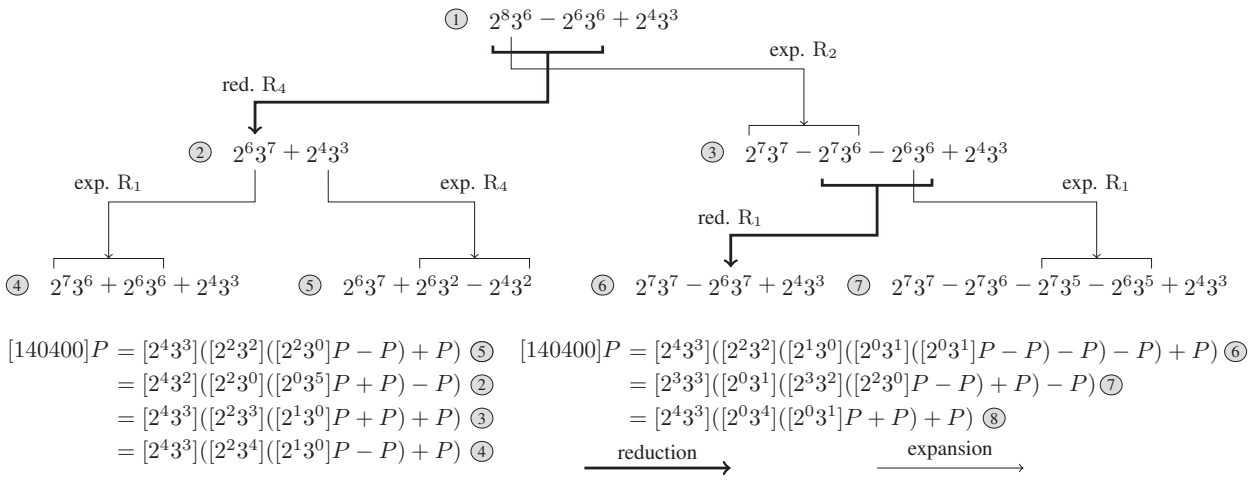

Fig. 3. Examples of some possible DBNS recodings for $k = 140400$.

VII. Validation and Comparison

In order to mathematically validate our method, we used comparisons to software results provided by PARI/GP which is a computer algebra system with many powerful number theory functions. We have used the software environment developed for the mathematical validation to experimentally evaluate the computation time using the proposed random recoding method. All obtained results have been compared to standard recoding methods (NAF, w-NAF see [1, p. 98]).

The experiments reported below have been realized using the curve P-224 provided by NIST (FIPS 186-2), see [1, appendix A.2.1] for details. The prime field $\mathbb{F}_p$ is defined with $p = p_{224} = 2^{224} - 2^{96} + 1$. There exists many ways to represent points of an elliptic curve. We use Jacobian coordinates (denoted $\mathcal{J}$) in order to speed up the $[k]P$ computation (see [1, sec. 3.2.1]). We also use mixed coordinates Jacobian with affine ones (denoted $\mathcal{A}$). Table II summarizes the cost of the main elliptic curve operations (see [7] for details). $\texttt{ADD}^{\mathcal{J}+\mathcal{A}}$ is the cost of point addition in mixed coordinates ($\mathcal{J} + \mathcal{A} \to \mathcal{J}$). $x - \texttt{DBL}^{\mathcal{J}}$ and $x - \texttt{TPL}^{\mathcal{J}}$ are respectively the cost of x doubling operations and x tripling operations when Jacobian coordinates are used.

curve operation	cost
$\texttt{ADD}^{\mathcal{J}+\mathcal{A}}$	$8 \cdot \texttt{M} + 3 \cdot \texttt{S}$
$x - \texttt{DBL}^{\mathcal{J}}$	$4x \cdot \texttt{M} + (4x + 2) \cdot \texttt{S}$
$x - \texttt{TPL}^{\mathcal{J}}$	$(11x - 1) \cdot \texttt{M} + (4x + 2) \cdot \texttt{S}$

TABLE II
Cost of elliptic curve operations in Jacobian coordinates for curves defined over $\mathbb{F}_p$ (from [7]).

The experimental setup was as follows. 1000 scalars k have been randomly chosen in the range $2^{\lceil log_2 p \rceil - 1} < k < p - 1$. So $k = (1, k_{n-2}, \ldots, k_1, k_0)$.

For each scalar k, the binary representation of k is converted to DBNS using the algorithm provided in [11]. Then 10 000 random recodings are applied to k in DBNS. For each recoded DBNS sequence, the number of curve level operations (ADD, DBL, TPL) is recorded. The number of times expansion and reduction rules can be applied are also recorded.

Table III presents the obtained statistics for 1000 scalars k and 10 000 random recodings for each scalar. The average value and the standard deviation of number of times each rule can applied are reported for reductions and for expansions.

Rules R_7 and R_8 are the most often used rules both for reduction and expansion. Note that given three successive $\{2, 3\}$-terms, in 6.7 % of cases no rule can be applied for those three terms.

The average length of the obtained DBNS expansion is 65.5 $\{2, 3\}$-terms with a standard deviation equal to 4.1. The maximal exponent of powers of 2 and 3 are 112 and 71 respectively. Then the average cost for computing $[k]P$ using the random DBNS recoding is

$$65.5 \cdot \texttt{ADD} + 112 \cdot \texttt{DBL} + 71 \cdot \texttt{TPL}.$$

We compared the $[k]P$ computation cost using our random DBNS recoding to the results based on standard methods. The corresponding results are reported in Table IV where M and S are respectively the cost of one multiplication and one square in $\mathbb{F}_p$. We use the standard cost approximation $\texttt{S} \approx 0.8 \cdot \texttt{M}$. The reported results show that our solution provide $[k]P$ operations a computation time similar to state-of-the-art 4-NAF recoding method but with the advantage of having a randomized behavior.

VIII. Conclusion

In this paper, preliminary results for arithmetic level protections against some side channel attacks are reported. The proposed countermeasure uses random and on-the-fly recodings of the secret key k in DBNS during the scalar multiplication $[k]P$. Starting from a scalar k, our method randomly provides different DBNS representations of the recoded scalar k. Then the number and the order of point level operations (addition,

	rules	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
reductions	avg.	594.4	750.3	544.1	450.0	157.5	166.8	975.9	1010.9
	std. dev.	48.5	81.0	62.8	46.9	34.9	38.3	67.8	86.6
	avg. [%]	12.8	16.1	11.7	9.7	3.4	3.6	21.0	21.7
expansions	avg.	561.8	743.0	537.4	437.8	150.6	163.2	1039.2	1031.8
	std. dev.	45.4	82.3	62.8	47.4	34.3	37.3	71.8	88.6
	avg. [%]	12.0	15.9	11.5	9.4	3.2	3.5	22.3	22.1

TABLE III
EXPERIMENT RESULTS FOR THE NUMBER OF TIMES EACH RULE CAN BE APPLIED.

method	total cost	approx. cost
NAF	$1500 \cdot \mathtt{M} + 1575 \cdot \mathtt{S}$	$\approx 2760 \cdot \mathtt{M}$
3-NAF	$1354 \cdot \mathtt{M} + 1524 \cdot \mathtt{S}$	$\approx 2573 \cdot \mathtt{M}$
4-NAF	$1284 \cdot \mathtt{M} + 1494 \cdot \mathtt{S}$	$\approx 2479 \cdot \mathtt{M}$
DBNS recoding	$1752 \cdot \mathtt{M} + 930 \cdot \mathtt{S}$	$\approx 2496 \cdot \mathtt{M}$

TABLE IV
COST COMPARISON OF THE PROPOSED METHOD FOR THE COMPUTATION OF $[k]P$ TO STANDARD METHODS.

doubling and tripling) is randomized during the scalar multiplication $[k]P$.

An on-the-fly recoding unit in DBNS has been implemented in FPGA. The cost of this unit is very small compared to the total cost of a complete ECC processor both for clock frequency and silicon area aspects. Our countermeasure provides randomized scalar multiplications at the speed of the best unprotected algorithms.

In the future, we plan to integrate this protection scheme into an ECC processor under development and experimentally evaluate its robustness using power attacks. We also plan to work on other recoding identities and rules such as $2^2 + 2 = 3^2 - 3$.

ACKNOWLEDGMENT

This work has been supported in part by *Région Bretagne* and *Conseil Général des Côtes d'Armor* (ROBUSTA grant).

REFERENCES

[1] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Springer, 2004.
[2] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.
[3] A. Byrne, N. Meloni, A. Tisserand, E. M. Popovici, and W. P. Marnane, "Comparison of simple power analysis attack resistant algorithms for an elliptic curve cryptosystem," *Journal of Computers*, vol. 2, no. 10, pp. 52–62, 2007.
[4] A. Byrne, F. Crowe, W. P. Marnane, N. Meloni, A. Tisserand, and E. M. Popovici, "SPA resistant elliptic curve cryptosystem using addition chains," *Int. J. High Performance Systems Architecture*, vol. 1, no. 2, pp. 133–142, Oct. 2007.
[5] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*, 2nd ed. Cambridge University Press, 1994.
[6] V. Dimitrov, G. Jullien, and W. Miller, "Theory and applications of the double-base number system," *IEEE Transactions on Computers*, vol. 48, no. 10, pp. 1098–1106, Oct. 1999.
[7] V. Dimitrov, L. Imbert, and P. K. Mishra, "The double-base number system and its application to elliptic curve cryptography," *Mathematics of Computation*, vol. 77, no. 262, pp. 1075–1104, Apr. 2008.
[8] ——, "Efficient and secure elliptic curve point multiplication using double-base chains," in *Proc. 11th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, ser. LNCS, vol. 3788. Chennai, India: Springer, Dec. 2005, pp. 59–78.
[9] R. Santoro, A. Tisserand, O. Sentieys, and S. Roy, "Arithmetic operators for on-the-fly evaluation of TRNGs," in *Proc. Advanced Signal Processing Algorithms, Architectures and Implementations XVIII*, vol. 7444. San Diego, CA, U.S.A.: SPIE, Aug. 2009.
[10] R. Santoro, O. Sentieys, and S. Roy, "On-the-fly evaluation of FPGA-based true random number generator," in *Proc. Int. Symp. on VLSI (ISVLSI)*. IEEE Computer Society, May 2009, pp. 55–60.
[11] C. Doche and L. Imbert, "Extended double-base number system with applications to elliptic curve cryptography," in *Proc. 7th International Conference on Cryptology (INDOCRYPT)*, ser. LNCS, vol. 4329. Kolkata, India: Springer, Dec. 2006, pp. 335–348.