Low-Complexity Hardware Architecture of APN Permutations Using TU-Decomposition

Lilya Budaghyan, José L. Imaña, and Nikolay Kaleyski

***Abstract*— Functions with good cryptographic properties which are used as S-boxes in the design of block ciphers have a fundamental importance to the security of these ciphers since they determine the resistance to various kinds of cryptanalytic attacks. *Almost Perfect Nonlinear* (APN) functions provide the best possible resistance to differential cryptanalysis, which is one of the most efficient cryptographic attacks against block ciphers known to date. Furthermore, APN permutations are of particular interest in practice since many cipher designs require the S-box to be a permutation. In this paper, we present a low-complexity hardware architecture for the TU-decomposition of APN permutations, showing how Dillon's APN permutation can be decomposed in this way as a practically relevant example. The TU-decomposition of an m-bit permutation is based on the use of two $m/2$-bit keyed permutations (T and U) to reduce the complexity of the original permutation. Dillon's permutation on 6 bits is the only known APN permutation on an even number of bits, so its study is of fundamental interest. We present hardware theoretical complexities and experimental results obtained from FPGA and ASIC implementations for the proposed TU-decomposition hardware architecture. These complexities and results are compared with other hardware architectures given in the literature for the same function. From the comparisons, it can be observed that the TU-decomposition architecture presented here greatly outperforms other hardware approaches with respect to area, delay and area×delay complexities.**

***Index Terms*— Block cipher, *S-box*, almost perfect nonlinear (APN), TU-decomposition, finite field, HW architecture.**

I. Introduction

BLOCK ciphers are the most important cryptographic primitives in symmetric cryptography, and they allow the encryption and decryption of an input message using a shared secret key [1]. Block ciphers are essential building blocks of virtually all modern cryptographic protocols, and the design of secure and efficient block ciphers is therefore of crucial importance for any cryptography applications.

Block ciphers must be resistant against any cryptanalytic attacks that a third-party might employ, in order to guarantee security [2]. Furthermore, they must encrypt and decrypt efficiently in terms of time and memory, since in a typical cryptographic protocol it is more often than not the block ciphers that need to process the largest amount of data.

The design of block ciphers involves the interleaving of highly complex non-linear transformations (that provide the security of the cipher) with fast and easy to implement linear operations (that provide no security by themselves, but strengthen the effect of the non-linear transformations). The non-linear transformations are typically called *substitution boxes*, or *S-boxes*, and are typically modeled as functions that input and output sequences of bits [1]. Various properties and statistics can be computed for a given S-box which describe how well it resists different kinds of attacks. The security of a cipher can be evaluated by considering the properties of its underlying S-boxes. An S-box is a function which maps n input bits to m output bits. For this reason, S-boxes are also called (n, m)-*functions*. An important case is when $n = m$ and a sequence of bits is replaced with another sequence with the same length.

The resistance of an S-box to differential [3] and linear cryptanalysis [4] (two of the most efficient cryptographic attacks against block ciphers) is determined by its *differential uniformity* [5] and *nonlinearity*, respectively. In order to be resilient to these attacks, the differential uniformity of an S-box should be as low as possible while the nonlinearity should be as high as possible. When $n = m$, the functions having the lowest differential uniformity are called *Almost Perfect Nonlinear* (APN). In the case of nonlinearity, its optimal value is only known in the case of odd n. The functions achieving this highest nonlinearity are called *Almost Bent* (AB). It can be proved that any AB function is also APN [6], but an APN function is not necessarily AB, even in the case of odd n. The S-boxes are also frequently required to be bijective, such as in block ciphers designed using a *Substitution Permutation Network* (SPN). For example, the design of the Rijndael cipher, which was selected as the Advanced Encryption Standard (AES) [7], is based on an SPN and has a bijective $(8, 8)$-function as an S-box. This function is not APN; this is because at the time of writing, no APN permutations on 8 bits are known, so a

Manuscript received 28 February 2024; revised 5 June 2024; accepted 27 June 2024. Date of publication 11 July 2024; date of current version 27 November 2024. The work of Lilya Budaghyan was supported in part by the Trond Mohn Foundation under Grant "Construction of Optimal Boolean Functions" and in part by the Research Council of Norway under Grant 314395. The work of José L. Imaña was supported by MCIN/AEI/10.13039/501100011033 and in part by the "ERDF a way of making Europe" under Grant PID2021-123041OB-I00. This article was recommended by Associate Editor X. Zeng. *(Corresponding author: José L. Imaña.)*

Lilya Budaghyan and Nikolay Kaleyski are with the Selmer Center, Department of Informatics, University of Bergen, 5020 Bergen, Norway (e-mail: Lilya.Budaghyan@uib.no; Nikolay.Kaleyski@uib.no).

José L. Imaña is with the Department of Computer Architecture and Automation, Faculty of Physics, Complutense University of Madrid, 28040 Madrid, Spain (e-mail: jluimana@ucm.es).

Digital Object Identifier 10.1109/TCSI.2024.3421354

permutation with the next best differential uniformity is used instead.

APN permutations with an odd number of bits have been known for a long time [8], but it was believed that APN permutations with an even number of bits do not exist [9], [10] until Dillon et al. [11] found an APN permutation on six bits. Dillon's permutation is the only known APN permutation with an even number of bits (up to equivalence) and has also been used for the design of the FIDES lightweight authenticated encryption algorithm [12], so its study is of fundamental interest.

APN functions are mostly represented as polynomials over finite fields, mainly using a univariate polynomial representation [13] or using a composition of simpler functions [14], [15], [16], that could lead to different hardware implementation complexities. For functions operating on a small number of bits, hardware implementation can be done simply using a lookup table containing all values of the function. However, this approach can be infeasible (especially in the case of resource-constrained devices) for functions working on a large number of bits because the size of the lookup table grows exponentially with the dimension [17], [18], [19]. For this reason, it is important to consider other hardware architectures that lead to efficient implementations in terms of memory requirements.

Low-complexity non-APN S-boxes considering security properties as differential uniformity and nonlinearity, among others, have been reported in the literature. In [31] and [32], low area and highly secure lightweight 8-bit S-boxes based on field inversion over $\mathbb{F}_{2^8}$ were presented. Lighweight cryptographic 8-bit S-boxes based on the use of two low-complexity 4-bit S-boxes were also presented in [33]. Optimized fault-tolerant and error-correcting 4-bit S-boxes for cryptographic applications with multiple errors detection were also given in [34]. However, the above S-boxes do not have the lowest differential uniformity and therefore they are not APN permutations.

In this paper, we present a low-complexity hardware architecture for the TU-decomposition of APN permutations and demonstrate how to decompose Dillon's APN permutation. The new architecture is based on the decomposition of Dillon's APN into smaller 3-bit keyed permutations (T, U) over $\mathbb{F}_{2^3}$. This TU-decomposition [15], obtained applying methods of reverse engineering to S-boxes [16], [20], can produce important reductions on the hardware implementation complexity of Dillon's APN permutation in comparison with classical approaches based on its evaluation as a polynomial over $\mathbb{F}_{2^6}$ for a given input. We present hardware theoretical complexities and experimental results obtained from FPGA and ASIC implementations for the proposed TU-decomposition hardware architecture and we compare them with other hardware architectures given in the literature for the same permutation. From the results obtained, the TU-decomposition architecture presented here greatly outperforms all other hardware approaches in terms of area, delay and area×delay complexities. It is important to note that the hardware architecture presented here can easily be generalized to any permutation on an even number of bits, and is not restricted to just Dillon's permutation; the latter is merely chosen as a particularly relevant practical example for demonstrating our method.

The paper is organized as follows. Section II introduces the fundamental concepts and definitions used throughout the paper. Dillon's APN permutation is given in Section III. Section IV presents the procedure for the TU-decomposition of Dillon's APN permutation $g(x)$. The hardware architecture of the TU-decomposition of Dillon's APN permutation and the description of the different components are given in Section V. Theoretical complexity analysis of the hardware architecture presented here and its comparison with other approaches are given in Section VI. Section VII gives FPGA and ASIC implementation results and discussion. Finally, the conclusions and some potential directions for future work are given in Section VIII.

II. Notations and Definitions

Let $\mathbb{F}_2 = \{0, 1\}$ be the finite field with two elements (also referred to as the *binary field*) and let $\mathbb{F}_{2^m}$ be the *binary extension field* with 2^m elements. Let also $f(y) = \sum_{i=0}^{m} f_i y^i$ be a monic irreducible polynomial of degree m over the binary field, with $f_i \in \mathbb{F}_2$ for $i = 0, 1, \ldots, m$. Any element $x \in \mathbb{F}_{2^m}$ can be represented in the *standard* or *polynomial basis* $\{1, \rho, \ldots, \rho^{m-1}\}$ as $x = \sum_{i=0}^{m-1} x_i \rho^i = (1, \rho, \ldots, \rho^{m-1}) \cdot (x_0, \ldots, x_{m-1})^T$, where $x_i \in \mathbb{F}_2$ and ρ is a root of the irreducible polynomial $f(y)$. The coefficients $(x_0, \ldots, x_{m-1})$ are denoted as the coordinates of the element x with respect to the polynomial basis.

We refer the reader to [1] for a general reference on (n, m)-functions and their cryptographic properties. A *vectorial Boolean function*, or (n, m)-function, is a function mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. Modern block ciphers include one or more vectorial Boolean functions as their only nonlinear components. That is the reason why (n, m)-functions are of fundamental importance in cryptography. If $n = m$, then any vectorial Boolean function can be uniquely expressed as a polynomial $S(x) = \sum_{i=0}^{2^m-1} s_i x^i$, with $s_i \in \mathbb{F}_{2^m}$, that is called the *univariate representation* of S. The *algebraic degree* of S is the largest binary weight (number of 1's in the binary representation) of any exponent i of x (with $s_i \neq 0$) in the univariate representation. *Affine* functions have an algebraic degree at most 1, while *quadratic* and *cubic* functions have algebraic degree 2 and 3, respectively. An affine function h satisfying $h(0) = 0$ is called *linear*.

The differential properties of an (n, m)-function g are given by its *Difference Distribution Table* (DDT). The DDT of g is the $2^n \times 2^m$ matrix Δ_g such that $\Delta_g(a, b) = \#\{x \in \mathbb{F}_2^n, D_a g(x) = b\}$, where $D_a g(x) = g(x + a) + g(x)$ is the *derivative* of g in direction $a \in \mathbb{F}_{2^n}$. The maximum coefficient in Δ_g (for all non-zero row indices a) is the *differential uniformity* of g and, if it is equal to δ, then g is said to be *differentially δ-uniform*. The lower the differential uniformity of an (n, n)-function, the stronger its resistance to differential cryptanalysis [3]. The lowest possible value of the differential uniformity is 2, and functions achieving this optimal value are called *Almost Perfect Nonlinear* (APN) functions. This is one of the reasons why the study of APN functions is of fundamental importance for the design and construction of

secure block ciphers [21], [22], [23], [24]. A well-established method for constructing block ciphers from S-boxes is the so-called Substitution Permutation Network (SPN). In order for an S-box to be used in an SPN, it must be a permutation. Thus, it is highly desirable to find APN permutations from the point of view of block cipher design.

With reference to the *nonlinearity* of and S-box g needed to resist linear cryptanalysis, it can be studied using the *Linear Approximation Table* (LAT). The LAT of an (n, m)-function g is the $2^n \times 2^m$ matrix Λ_g such that $\Lambda_g(a, b) = \#\{x \in \mathbb{F}_2^n, a \cdot x = b \cdot g(x)\} - 2^{n-1}$, where "$\cdot$" is the scalar product. The *nonlinearity* of an (n, m)-function g is $2^{n-1} - max(|\Lambda_g(a, b)|)$, for all non-zero row and column indices a and b. In order to resist linear attacks, the nonlinearity of an S-box must be high.

III. Dillon's APN Permutation

APN functions must be permutations in order to be used as S-boxes in Substitution Permutation Networks. While many functions are known to be APN permutations in $\mathbb{F}_2^m$ for m odd, it was long believed that there are no APN permutations for m even (known as the "big APN problem"). However, in 2010 mathematicians from the NSA (Dillon) presented what, to date, is the only known 6-bit APN permutation [11]. At present, Dillon's permutation is the only known APN permutation on an even number of bits n. We know that for even n less than 6 no APN permutations exist, while for even n greater than 6, the existence of APN permutations remains unresolved. Furthermore, Dillon's permutation has been used to design the lightweight authenticated cipher FIDES [12], so the efficiency of its hardware implementations and the study of its cryptographic properties are of great interest [25].

Dillon's APN permutation can be efficiently implemented in hardware by means of its decomposition into simpler functions [14], [15]. A hardware architecture and implementation of the univariate polynomial representation of Dillon's permutation $g(x)$, with $x \in \mathbb{F}_{2^6}$, was given in [14], where $g(x)$ was also presented as a composition of two functions f_1 and f_2^{-1}, i.e., $g = f_1 \circ f_2^{-1}$. The univariate representation of $g(x)$ given in [14] is $g(x) = \rho^{18}x^{57} + \rho^{22}x^{56} + \rho^{18}x^{50} + \rho^{22}x^{49} + \rho^{7}x^{48} + \rho^{18}x^{43} + \rho^{22}x^{42} + \rho^{44}x^{41} + \rho^{57}x^{40} + \rho^{18}x^{36} + \rho^{22}x^{35} + \rho^{22}x^{34} + \rho^{50}x^{33} + \rho^{24}x^{32} + \rho^{18}x^{29} + \rho^{57}x^{28} + \rho^{25}x^{25} + \rho^{18}x^{24} + \rho^{18}x^{22} + \rho^{57}x^{21} + \rho^{7}x^{20} + \rho^{18}x^{18} + \rho^{18}x^{17} + \rho^{18}x^{15} + \rho^{57}x^{14} + \rho^{44}x^{13} + \rho^{29}x^{12} + \rho^{11}x^{11} + \rho^{18}x^{10} + \rho^{24}x^{8} + \rho^{57}x^{7} + \rho^{22}x^{6} + \rho^{22}x^{5} + \rho^{3}x^{4} + \rho^{18}x^{3} + \rho^{13}x$, while the functions $f_1(x)$ and $f_2^{-1}(x)$ in [14] are $f_1(x) = x + \rho^7 x^8 + \rho^4 f(x) + \rho^{32} f(x)^8$ and $f_2^{-1}(x) = \rho x^{56} + \rho x^{49} + \rho^{22}x^{48} + \rho x^{42} + \rho^{23}x^{41} + \rho^{36}x^{40} + \rho x^{35} + \rho^{15}x^{34} + \rho^{29}x^{33} + \rho^{36}x^{28} + \rho^{36}x^{21} + \rho^{22}x^{20} + \rho^{36}x^{14} + \rho^{23}x^{13} + \rho^{8}x^{12} + \rho^{58}x^{8} + \rho^{36}x^{7} + \rho^{15}x^{6} + \rho x^{5} + \rho^{58}x$, respectively. For $f_1(x)$, the function $f(x) = \rho x^3 + \rho^5 x^{10} + \rho^4 x^{24}$ was used, where ρ is a root of the primitive pentanomial $f(y) = y^6 + y^4 + y^3 + y + 1$ over $\mathbb{F}_2$. It can be observed that Dillon's permutation $g(x)$ has algebraic degree 4, while the functions $f_1(x)$ and $f_2^{-1}(x)$ used to express $g(x) = f_1(f_2^{-1}(x))$ have algebraic degrees 2 and 3, respectively. The reduction of the algebraic degree of the functions involved in the representation implies a reduction of the complexity of the hardware implementation, so the study of the decomposition of Dillon's permutation is of fundamental interest. A look-up table of this permutation [14] in octal is given in Table I, where if the input x is given in octal as e.g. $x = cd = 26$ then $g(x) = 47$.

TABLE I
Dillon's Permutation $g(x)$ in Octal

		d							
		0	**1**	2	3	4	5	6	7
c	0	00	**30**	42	71	01	13	04	75
	1	62	**31**	22	35	02	36	65	52
	2	15	**20**	16	03	54	05	47	56
	3	40	**14**	70	64	43	11	66	33
	4	74	**06**	50	07	17	26	23	57
	5	73	**41**	27	77	61	72	63	55
	6	44	**37**	12	67	25	53	34	24
	7	60	**76**	51	21	10	46	45	32

Applying methods of reverse engineering to S-boxes [16], [20], a method to obtain a decomposition of Dillon's APN permutation relying on two 3-bit keyed permutations (T, U) was given in [15]. We follow this method and apply it to Dillon's permutation $g(x)$ in Table I to obtain a $TU - decomposition$ and present its hardware architecture. We then compare the complexity of the proposed architecture with the complexities of the corresponding architectures of the univariate and decomposed representations given in [14].

IV. TU-Decomposition of Dillon's APN Permutation

The procedure for the TU-decomposition of Dillon's APN permutation $g(x)$ has the following steps [15]:

- Obtain a permutation η by means of the creation of the LAT table for $g(x)$ and the construction of its Pollock's graphical representation.
- Apply the permutation η to the table $(x, g(x))$ with the inputs/outputs of $g(x)$ given in Table I to obtain a new table $(\eta(x), \eta(g(x)))$.
- Obtain the *keyed* permutations T from the table $(\eta(x), \eta(g(x)))$.
- Obtain the *keyed* permutations U from the tables $(x, g(x))$ and $(\eta(x), \eta(g(x)))$.

We note that this same procedure generalizes to any function on an even number of bits, and Dillon's permutation is chosen as a particularly relevant example to illustrate the method.

A. Computation of the Permutation η

In order to determine the permutation η, the LAT of Dillon's APN S-box $g(x)$ must first be computed. Then the *Pollock's Pattern Recognition* [20] method is applied. This method is based on turning the LAT of the S-box to be analyzed (where values in the LAT are 0s, 4s and 8s) into a picture and then identifying structural properties by means of pattern finding. The picture obtained is similar to *Jackson Pollock*'s abstract drip paintings, hence the name of this method. The Pollock's representation [15] of the absolute value of the LAT of $g(x)$ for Table I is shown in Figure 1, where some patterns (row and columns) with only black and white colors can be found (white color corresponds with 0 value, grey with

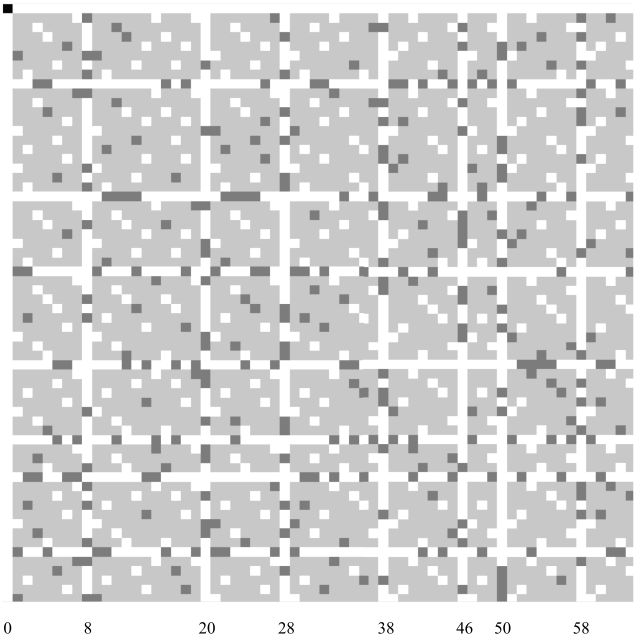

Fig. 1. Pollock's representation of the LAT of $g(x)$.

4 and black with 8). Furthermore, it can be observed that the columns with only black and white colors have abscissas $\{0, 8, 20, 28, 38, 46, 50, 58\}$ and that the binary representation of these numbers forms a linear subspace of $\mathbb{F}_{2^6}$ generated by the values $\{8, 20, 38\}$ [15]. Using the binary representations, this can be proven considering that $28 = 8 \oplus 20$, $46 = 8 \oplus 38$, $50 = 20 \oplus 38$ and $58 = 8 \oplus 20 \oplus 38$ (where $\oplus$ stands for the bitwise XOR). Then a permutation η can be constructed such that $\eta : 1 \rightarrow 8$, $\eta : 2 \rightarrow 20$, $\eta : 4 \rightarrow 38$ and then completing it in a natural way by setting $\eta : 8 \rightarrow 1$, $\eta : 16 \rightarrow 2$ and $\eta : 32 \rightarrow 4$ to obtain a linear permutation η of $\mathbb{F}_{2^6}$. The composition of the permutation η with the S-box will have the structural effect of grouping the black-and-white columns in the LAT [15].

B. Computation of $(\eta(x), \eta(g(x))$

The application of the above permutation η to Dillon's APN S-box $g(x)$ given in Table I by the pairs $(x, g(x))$ allows us to obtain the new table $(\eta(x), \eta(g(x))$ given in Table II with values in octal. Table II allows us to easily identify the property that if the last three bits of the input $\eta(x)$ are fixed and its first three bits take all possible 2^3 values, then the last three bits of the output $\eta(g(x))$ also take all possible values [15]. In Table II, for each column (last 3 bits of $\eta(x)$ fixed), the last 3 bits (rightmost digit) of the output $\eta(g(x))$ take all possible values.

As given in [15], a permutation π from $\mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ to itself can be decomposed using two *keyed* n-bit permutations T and U if the following property is fulfilled: fixing the lowest (rightmost) n-bits l of the input to any value and taking the highest (leftmost) n-bits h all possible 2^n values then the highest n-bits output of π takes all possible 2^n values. In this case, the permutation $\pi(h, l)$ is $\pi(h, l) = (T_l(h), U_{T_l(h)}(l))$. Figure 2 shows the TU-decomposition [15]. The above permutation η applied to Dillon's S-box $g(x)$ satisfies this property, so its *keyed* 3-bit permutations T_i and U_i of T and U, respectively, can be determined.

TABLE II
TABLE $(\eta(x), \eta(g(x))$ IN OCTAL

		d							
		0	1	2	3	4	5	6	7
c	0	00	22	57	04	41	33	42	06
	1	03	13	02	47	62	14	71	65
	2	05	70	25	15	20	26	63	07
	3	72	75	74	12	17	55	34	40
	4	54	01	73	16	43	30	10	24
	5	31	66	60	23	56	11	35	61
	6	76	64	46	50	45	52	36	32
	7	67	37	51	21	44	27	77	53

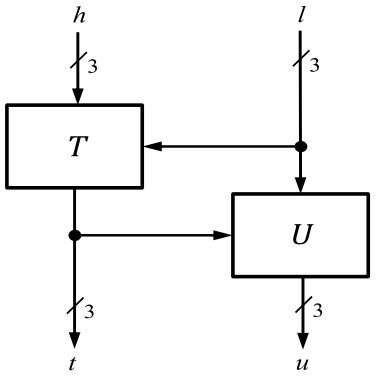

Fig. 2. TU decomposition.

TABLE III
KEYED T_i PERMUTATIONS OF T

	0	**1**	2	3	4	5	6	7
T_0	0	**3**	5	2	4	1	6	7
T_1	2	**3**	0	5	1	6	4	7
T_2	7	**2**	5	4	3	0	6	1
T_3	4	**7**	5	2	6	3	0	1
T_4	1	**2**	0	7	3	6	5	4
T_5	3	**4**	6	5	0	1	2	7
T_6	2	**1**	3	4	0	5	6	7
T_7	6	**5**	7	0	4	1	2	3

C. Computation of the Keyed Permutations T

The keyed permutations T_i, with $i \in \{0, \ldots, 7\}$, of T can be directly obtained from the octal Table II where, for a fixed column d, it can be observed that the rightmost digits of the outputs $\eta(g(x))$ for $c \in \{0, \ldots, 7\}$ take all possible values. Therefore, each column i in Table II determines the keyed permutations T_i, $i \in \{0, \ldots, 7\}$, as given in Table III, where T_i denotes the permutation corresponding to the key i.

D. Computation of the Keyed Permutations U

In order to determine the keyed permutations U_i, with $i \in \{0, \ldots, 7\}$, of U we use Dillon's S-box $g(x)$ given in Table I. Now using the keyed permutation T_i in Table III we annotate the values obtained for $T_V(M)$, where M is fixed ($M \in \{0, \ldots, 7\}$ is one of the columns in Table III) and V ranges from 0 to 7. The annotated values will be the inputs

TABLE IV
KEYED U_i PERMUTATIONS OF U

	0	1	2	3	4	5	6	7
U_0	0	7	6	5	2	3	1	4
U_1	3	0	5	2	4	1	**7**	6
U_2	7	2	**0**	1	**6**	5	4	3
U_3	**0**	**1**	7	2	4	3	6	5
U_4	5	6	7	0	4	**1**	3	2
U_5	0	7	2	1	4	5	3	**6**
U_6	7	6	4	1	5	2	3	0
U_7	6	3	5	**4**	1	2	7	0

for the keyed permutations U_i. Now in Table I with the pairs $(x, g(x))$ in octal, we fix the 3 least significant bits of the input x (the d columns) to the value M and let the 3 most significant bits of the input x (the c rows) take all possible values V (ranging from 0 to 7), i.e., $x = VM$ where M is fixed and V varies. Then we annotate the three least significant bits (rightmost digits) of the values $g(x) = g(VM)$ and denote these values as $g_r(VM)$. Finally the obtained values $g_r(VM)$ must be the outputs of the keyed permutations $U_{T_V(M)}(V)$, i.e., $U_{T_V(M)}(V) = g_r(VM)$ with V, M ranging from 0 to 7. The keyed permutations U_i, $i \in \{0, \ldots, 7\}$, of U are given in Table IV, where U_i denotes the permutation corresponding to the key i.

For example, for $M = 1$, Table III shows in boldface the values of $T_0(1) = 3$, $T_1(1) = 3$, $T_2(1) = 2$, $T_3(1) = 7$, $T_4(1) = 2$, $T_5(1) = 4$, $T_6(1) = 1$, $T_7(1) = 5$, and Table I shows in boldface the values $g_r(V1)$, with V from 0 to 7, i.e., $g_r(01) = 0$, $g_r(11) = 1$, $g_r(21) = 0$, $g_r(31) = 4$, $g_r(41) = 6$, $g_r(51) = 1$, $g_r(61) = 7$, $g_r(71) = 6$. Now using the expression $U_{T_V(M)}(V) = g_r(VM)$, the following values for keyed permutations U_i can be obtained: $U_3(0) = 0$, $U_3(1) = 1$, $U_2(2) = 0$, $U_7(3) = 4$, $U_2(4) = 6$, $U_4(5) = 1$, $U_1(6) = 7$ and $U_5(7) = 6$. These values are represented in boldface in Table IV.

V. HARDWARE ARCHITECTURE OF THE TU-DECOMPOSITION OF DILLON'S APN PERMUTATION

The hardware architecture of Dillon's APN Permutation $g(x)$ using the above TU-decomposition is shown in Figure 3, where the dashed rectangle shows the TU-decomposition with the outputs swapped. It must be noted that this architecture generalizes to any even size of the input, and is not restricted to just Dillon's permutation; the latter is merely chosen as a particularly relevant practical example for demonstrating our method. The construction of the different building blocks is given in the following subsections.

A. Permutation η

The permutation η has the 6-bit input x and the 6-bit output $\eta(x)$. Using the binary representation of $x = \sum_{i=0}^{5} x_i 2^i$, we have that the coefficients x_i, with $i \in \{0, \ldots, 5\}$, have 2^i as their associated weights. In the same way, the 6-bit output $\eta(x) = \sum_{i=0}^{5} \eta_i 2^i$ has the coefficients η_i associated with weights 2^i. As given in Subsection IV-A, the permutation η can be constructed such that $\eta(1) = 8$, $\eta(2) = 20$, $\eta(4) = 38$, $\eta(8) = 1$, $\eta(16) = 2$ and $\eta(32) = 4$. Therefore,

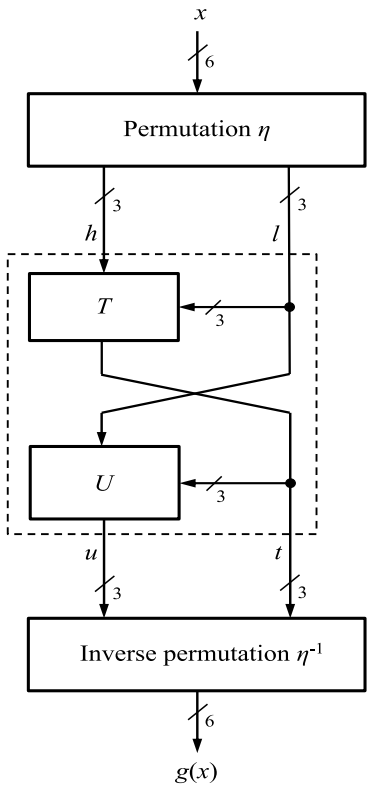

Fig. 3. Hardware architecture of TU-decomposition of Dillon's permutation.

the permutation $\eta(1) = 8$ connects the input x_0 (with weight $1 = 2^0$) with the output η_3 (with weight $8 = 2^3$). Similarly, $\eta(8) = 1$, $\eta(16) = 2$ and $\eta(32) = 4$ will connect x_3 with η_0, x_4 with η_1 and x_5 with η_2, respectively. Furthermore, $\eta(2) = 20$ will connect x_1 with both η_2 and η_4 ($20 = 2^2 + 2^4$). Finally, $\eta(4) = 38$ will connect x_2 with η_1, η_2 and η_5 ($38 = 2^1 + 2^2 + 2^5$). It is easy to determine the binary matrix of the linear permutation η that is given as follows:

$$\eta = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{1}$$

Therefore, the outputs are $\eta_0 = x_3$, $\eta_1 = x_4 \oplus x_2$, $\eta_2 = x_5 \oplus x_2 \oplus x_1$, $\eta_3 = x_0$, $\eta_4 = x_1$ and $\eta_5 = x_2$. The hardware architecture of the block implementing the permutation η is shown in Figure 4.

B. Keyed Permutation T

In order to compare different architectures of Dillon's APN permutation, polynomial expressions can be computed for the keyed permutations T_i, with $i \in \{0, \ldots, 7\}$, of T given in Table III. The 3-bit inputs to block T can be represented as finite field elements in $\mathbb{F}_{2^3}$ generated by the irreducible trinomial $f(y) = y^3 + y + 1$. In this case, the elements of $\mathbb{F}_{2^3}$ represented in the polynomials basis $\{1, \rho, \rho^2\}$, with ρ being a root of $f(y)$, are $\{0, 1, \rho, \rho+1, \rho^2, \rho^2+1, \rho^2+\rho, \rho^2+\rho+1\}$ where their binary strings can be represented as integers

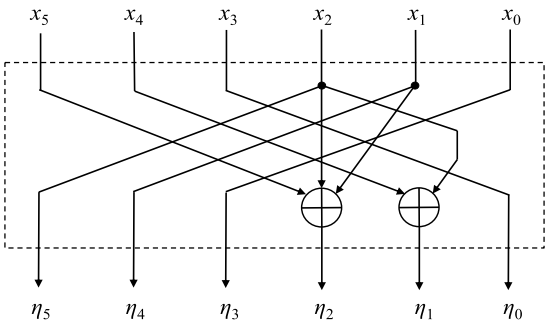

Fig. 4. Permutation η.

TABLE V
POLYNOMIAL REPRESENTATION OF KEYED T_i PERMUTATIONS

	Interpolation polynomial
T_0	$6x^6+7x^5+1x^4+6x^3+5x^2+0x+0$
T_1	$6x^6+7x^5+2x^4+6x^3+2x^2+6x+6$
T_2	$6x^6+7x^5+2x^4+6x^3+7x^2+7x+7$
T_3	$6x^6+7x^5+0x^4+6x^3+5x^2+1x+4$
T_4	$6x^6+7x^5+0x^4+6x^3+5x^2+1x+1$
T_5	$6x^6+7x^5+1x^4+6x^3+0x^2+1x+3$
T_6	$6x^6+7x^5+0x^4+6x^3+5x^2+1x+2$
T_7	$6x^6+7x^5+0x^4+6x^3+5x^2+1x+6$

$\{0, 1, 2, 3, 4, 5, 6, 7\}$, respectively. Now keyed permutations T_i can be represented by univariate polynomials over $\mathbb{F}_{2^3}$ using Lagrange interpolation. For $x \in \mathbb{F}_{2^3}$, these polynomials (with algebraic degree 2) are given in Table V where it can be observed that the terms $6x^6+7x^5+6x^3$ are common to every T_i and therefore key-independent. Furthermore, the selection of each of the permutations T_i depends on the *key* i, so a vectorial multiplexer can be used for their selection as shown in Figure 5(a).

The hardware architecture of the keyed permutations T_i can be determined as follows. The 3-bit input x to T_i, $i \in \{0, \ldots, 7\}$, is a finite field element in $\mathbb{F}_{2^3}$ generated by the irreducible trinomial $f(y) = y^3+y+1$, with ρ being a root of $f(y)$. From the univariate expressions given in Table V, it can be observed that the squares x^2 and x^4 must first be computed in such a way that the remaining powers x^3, x^5 and x^6 can be computed by the finite field $\mathbb{F}_{2^3}$ multiplication of these squares and/or the input x, i.e., $x^3 = x \cdot x^2$, $x^5 = x \cdot x^4$ and $x^6 = x^2 \cdot x^4$. Subsequently, the above powers of x must be multiplied by some element $\{0, 1, \rho, \rho+1, \rho^2, \rho^2+1, \rho^2+\rho, \rho^2+\rho+1\}$ (represented in Table V as integers) of $\mathbb{F}_{2^3}$ and finally XORed to compute the keyed permutation T_i.

The finite field multiplier over $\mathbb{F}_{2^3}$ used in this paper and, as an example, the hardware architecture of the keyed permutation T_0 are given in the following.

1) Multiplier Over $\mathbb{F}_{2^3}$: In this work, we have used the method given in [28] for the design of $\mathbb{F}_{2^m}$ multipliers for irreducible trinomials. Following [28], the product $C = A \cdot B$, with $A, B \in \mathbb{F}_{2^3}$, can be computed by using the functions $\mathbf{S_1} = a_0b_0$, $\mathbf{S_2} = (a_0b_1 + a_1b_0)$, $\mathbf{S_3} = a_1b_1 + (a_0b_2 + a_2b_0)$, $\mathbf{T_0} = (a_1b_2 + a_2b_1)$ and $\mathbf{T_1} = a_2b_2$, where $a_i, b_i \in \mathbb{F}_2$ are the coordinates of A and B, respectively. With these functions, the coordinate expressions of the product $C = A \cdot B$ over $\mathbb{F}_{2^3}$,

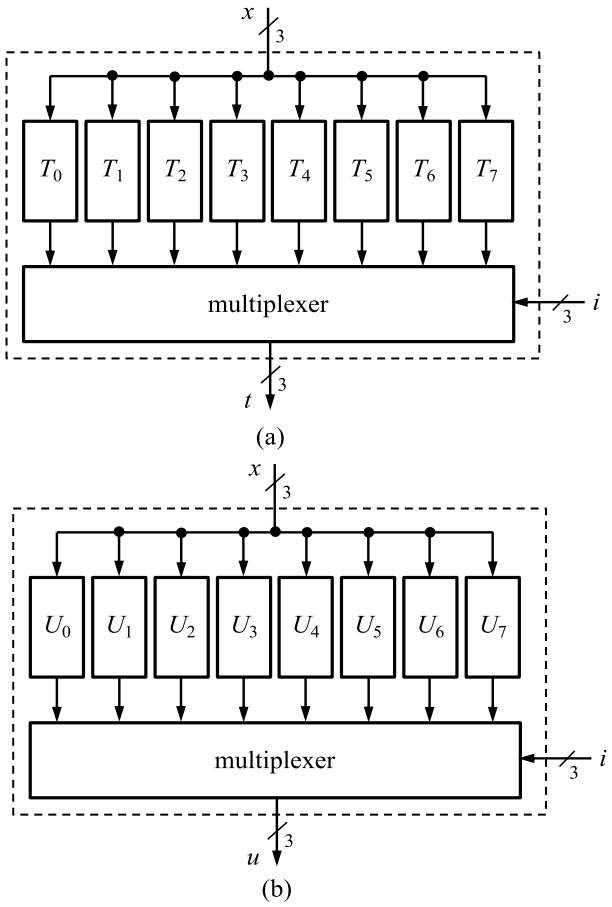

Fig. 5. (a) Keyed permutation T. (b) Keyed permutation U.

with $c_i \in \mathbb{F}_2$, are

$$\begin{aligned} c_0 &= \mathbf{S_1} + \mathbf{T_0} \\ c_1 &= \mathbf{S_2} + \mathbf{T_1} + \mathbf{T_0} \\ c_2 &= \mathbf{S_3} + \mathbf{T_1} \end{aligned} \tag{2}$$

2) Architecture of T_0: As given in Table V, the univariate expression of the keyed permutation T_0 is $T_0 = 6x^6 + 7x^5 + 1x^4 + 6x^3 + 5x^2 + 0x + 0$, where x is a finite field element in $\mathbb{F}_{2^3}$ generated by the irreducible trinomial $f(y) = y^3 + y + 1$, with ρ being a root of $f(y)$. In this case, the elements of $\mathbb{F}_{2^3}$ in the polynomials basis $\{1, \rho, \rho^2\}$ are $\{0, 1, \rho, \rho+1, \rho^2, \rho^2+1, \rho^2+\rho, \rho^2+\rho+1\}$, where their binary strings are represented by the integers $\{0, 1, 2, 3, 4, 5, 6, 7\}$, respectively.

For the input $x = x_2\rho^2 + x_1\rho + x_0$, with $x_i \in \mathbb{F}_2$, the squares x^2 and x^4 modulo $f(y)$ can be easily computed because $x^{2^i} = x_2\rho^{2\cdot 2^i} + x_1\rho^{1\cdot 2^i} + x_0\rho^{0\cdot 2^i}$ and the powers of ρ are reduced modulo $f(y)$ using the expressions $\rho^3 = \rho + 1$, $\rho^4 = \rho^2 + \rho$, $\rho^5 = \rho^2 + \rho + 1$, $\rho^6 = \rho^2 + 1$, $\rho^7 = 1$ and $\rho^8 = \rho$. Therefore the squares x^2 and x^4 modulo $f(y) = y^3 + y + 1$ are

$$x^2 = (x_2 + x_1)\rho^2 + x_2\rho + x_0 \tag{3}$$

$$x^4 = x_1\rho^2 + (x_2 + x_1)\rho + x_0 \tag{4}$$

The remaining powers x^3, x^5 and x^6 can be computed as $x^3 = x \cdot x^2$, $x^5 = x \cdot x^4$ and $x^6 = x^2 \cdot x^4$, where $\cdot$ corresponds to the use of the $\mathbb{F}_{2^3}$ multiplier given in equation (2).

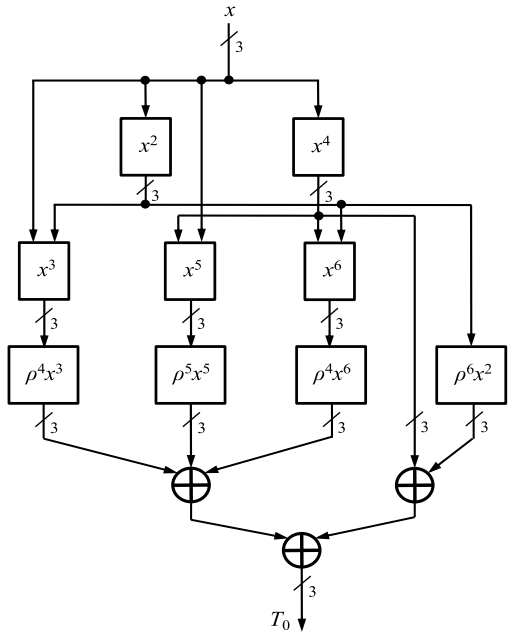

Fig. 6. Keyed permutation T_0.

TABLE VI
POLYNOMIAL REPRESENTATION OF KEYED U_i PERMUTATIONS

	Interpolation polynomial
U_0	$2x^6 + 5x^5 + 0x^4 + 4x^3 + 3x^2 + 1x + 0$
U_1	$2x^6 + 5x^5 + 7x^4 + 4x^3 + 5x^2 + 4x + 3$
U_2	$2x^6 + 5x^5 + 1x^4 + 4x^3 + 0x^2 + 1x + 7$
U_3	$2x^6 + 5x^5 + 6x^4 + 4x^3 + 6x^2 + 2x + 0$
U_4	$2x^6 + 5x^5 + 6x^4 + 4x^3 + 5x^2 + 5x + 5$
U_5	$2x^6 + 5x^5 + 1x^4 + 4x^3 + 3x^2 + 0x + 0$
U_6	$2x^6 + 5x^5 + 7x^4 + 4x^3 + 6x^2 + 5x + 7$
U_7	$2x^6 + 5x^5 + 0x^4 + 4x^3 + 0x^2 + 0x + 6$

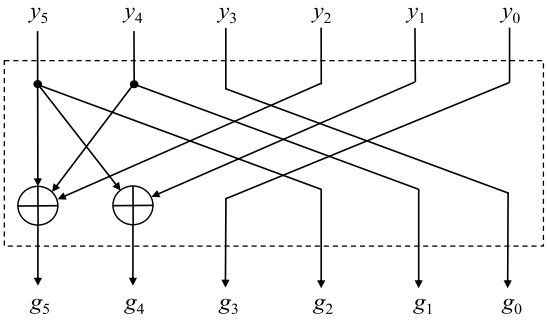

Fig. 7. Inverse permutation η^{-1}.

As previously given, the integers 7, 6, 5, 1 in the T_0 expression corresponds to the field elements $\rho^2 + \rho + 1 = \rho^5$, $\rho^2 + \rho = \rho^4$, $\rho^2 + 1 = \rho^6$, 1, respectively. Therefore, the products $7a$, $6a$ and $5a$, with $a = a_2\rho^2 + a_1\rho + a_0$, $a_i \in \mathbb{F}_2$, being a generic element in $\mathbb{F}_{2^3}$ can be computed as:

$$7a = \rho^5 a = (a_1 + a_0)\rho^2 + a_0\rho + (a_2 + a_1 + a_0) \quad (5)$$

$$6a = \rho^4 a = (a_2 + a_1 + a_0)\rho^2 + (a_1 + a_0)\rho + (a_2 + a_1) \quad (6)$$

$$5a = \rho^6 a = a_0\rho^2 + a_2\rho + (a_1 + a_0) \quad (7)$$

Figure 6 shows the architecture of the keyed permutation $T_0 = 6x^6 + 7x^5 + 1x^4 + 6x^3 + 5x^2 + 0x + 0 = 6x^6 + 7x^5 + x^4 + 6x^3 + 5x^2 = \rho^4x^6 + \rho^5x^5 + x^4 + \rho^4x^3 + \rho^6x^2$, where the key-independent addition $6x^6 + 7x^5 + 6x^3$ (common to every T_i) is previously XORed in order to be shared with the remaining T_i, $i = 1, \ldots, 7$ permutations.

C. Keyed Permutation U

In a similar way as in Subsection V-B, univariate polynomials over $\mathbb{F}_{2^3}$ using Lagrange interpolation can be computed for the keyed permutations U_i, with $i \in \{0, \ldots, 7\}$, of U given in Table IV. These expressions (with algebraic degree 2) are given in Table VI, where it can also be observed that the terms $2x^6 + 5x^5 + 4x^3$ are common to every U_i and therefore key-independent. The selection of each of the permutations U_i depends on the *key* i, so a vectorial multiplexer can also be used for their selection as shown in Figure 5(b).

The hardware architecture of the keyed permutations U_i can be determined in a similar way as shown in subsection V-B for the T_i permutations.

D. Inverse Permutation η^{-1}

The inverse permutation η^{-1} has a 6-bit input y and the 6-bit output $\eta^{-1}(y)$. As shown in Figure 3, this block supplies the output of Dillon's permutation $g(x)$, so $\eta^{-1}(y) = g(x)$. The inputs and outputs can be represented by their coefficients $(y_5, y_4, y_3, y_2, y_1, y_0)$ and $(g_5, g_4, g_3, g_2, g_1, g_0)$, respectively. The inverse permutation η^{-1} can be determined by simply computing the inverse of the binary matrix of η given in equation (1). The inverse matrix is as follows:

$$\eta^{-1} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \quad (8)$$

and therefore the outputs are given as $g_0 = y_3$, $g_1 = y_4$, $g_2 = y_5$, $g_3 = y_0$, $g_4 = y_1 \oplus y_5$ and $g_5 = y_2 \oplus y_4 \oplus y_5$. The hardware architecture of the block implementing the inverse permutation η^{-1} is shown in Figure 7.

VI. THEORETICAL COMPLEXITY ANALYSIS

Area and time theoretical complexities of Dillon's APN Permutation $g(x)$ using the TU-decomposition can be obtained from the complexities of the modules described in Section V. Area complexity is determined by the number of 2-input logic gates (AND, OR, XOR) and time complexity corresponds to the maximum number of 2-input logic gates that a signal must traverse from input to output, given in terms of T_{AND}, T_{OR} and T_{XOR} (delay of 2-input AND, OR and XOR gates, respectively).

A. Area Complexity

Using the modules described in Section V, the area complexity can be computed as follows:

TABLE VII
THEORETICAL COMPLEXITIES OF DILLON'S PERMUTATION

	AND	XOR	OR	Delay
Univariate polynomial [13]	1764	2709	–	$2T_{AND} + 24T_{XOR}$
$f_1 \circ f_2^{-1}$ [14]	756	1167	–	$3T_{AND} + 42T_{XOR}$
Univariate polynomial [14]	1152	1605	–	$2T_{AND} + 24T_{XOR}$
TU-decomposition	198	219	42	$5T_{AND} + 6T_{OR} + 11T_{XOR}$

- The permutation block η shown in Figure 4 involves the use of one 3-input XOR and one 2-input XOR gates, so the area complexity will be three 2-input XOR gates.
- The keyed permutation T block given in Subsection V-B involves the implementation of the eight permutations T_i and a vectorial multiplexer with three control inputs, eight 3-bit data inputs and one 3-bit output.
  The area complexity of T_0 can be computed using equations (3)–(7) and Figure 6.
  For a generic $a = a_2\rho^2 + a_1\rho + a_0$, $a_i \in \mathbb{F}_2$, in $\mathbb{F}_{2^3}$ and in a similar way as given in Subsection V-B, the following expression is also needed for the computation of the remaining T_i, $i = 1, \ldots, 7$, permutations:

$$2a = \rho a = a_1\rho^2 + (a_2 + a_0)\rho + a_2 \tag{9}$$

  Furthermore, it can be observed that the implementation of T_i permutations require the use of three $\mathbb{F}_{2^3}$ multipliers (for the computation of $x^3 = x \cdot x^2$, $x^5 = x \cdot x^4$ and $x^6 = x^2 \cdot x^4$). From equations (2), a finite field multipler requires 9 AND and 8 XOR gates, so 27 AND and 24 XOR gates are needed for the three multipliers.
  Using the expressions given in Table V, it can be observed that the key-independent addition $6x^6 + 7x^5 + 6x^3$ can be shared among all the T_i permutations.
  It can also be observed that the addition of the last term in expressions given in Table V is simply given by the 3-bit bitwise XOR of the corresponding terms $1 =$ "001", $2 = \rho =$ "010", $3 = \rho + 1 =$ "011", $4 = \rho^2 =$ "100", $6 = \rho^2 + \rho =$ "110" and $7 = \rho^2 + \rho + 1 =$ "111".
  With respect to the vectorial multiplexer with three control inputs, eight 3-bit data inputs and one 3-bit output, it is implemented with three MUXs 8:1 in parallel, with a total complexity of 72 AND and 21 OR 2-input gates.
  Finally, using the above considerations, it can be proven that the area complexity of the keyed permutation T block is given by 99 AND, 21 OR and 101 XOR gates.
- The keyed permutation U block given in Subsection V-C involves the implementation of the eight permutations U_i and a vectorial multiplexer with three control inputs, eight 3-bit data inputs and one 3-bit output.
  From Table VI, it can be observed that the key-independent addition $2x^6 + 5x^5 + 4x^3$ can be shared among all the U_i permutations. Furthermore, using the following expressions for a generic $a \in \mathbb{F}_{2^3}$

$$3a = (a_2 + a_1)\rho^2 + (a_2 + a_1 + a_0)\rho + (a_2 + a_0) \tag{10}$$

$$4a = (a_2 + a_0)\rho^2 + (a_2 + a_1)\rho + a_1 \tag{11}$$

  and using similar considerations as for the keyed permutation T block, it can be proven that the area complexity of the keyed permutation U block is given by 99 AND, 21 OR and 112 XOR gates.
- The inverse permutation block η^{-1} shown in Figure 7 requires the use of one 3-input XOR and one 2-input XOR gates, so the area complexity is three 2-input XOR gates.

The combination of the above complexities gives a total area complexity of 198 AND, 42 OR and 219 XOR gates for Dillon's APN Permutation $g(x)$ using the TU-decomposition as shown in Table VII.

B. Time Complexity

From the previous descriptions of the different modules, the time complexity can be determined as follows:

- The delay of the permutation block η is given by $2T_{XOR}$.
- The time complexity of the keyed permutation T block is given by the maximum delay of the T_i, $i = 0, \ldots, 7$, permutations and the delay of the vectorial multiplexer. It can be proven that the maximum delay corresponds to the permutations T_1, T_2, T_5 and is given by $T_{AND} + 11T_{XOR}$. Furthermore, the delay of the vectorial multiplexer is $2T_{AND} + 3T_{OR}$. Therefore, the time complexity of the keyed permutation T block is $3T_{AND} + 3T_{OR} + 11T_{XOR}$.
- The time complexity of the keyed permutation U block is determined in a similar way as the previously computed keyed permutation T. It can be proven that the maximum delay of the U_i, $i = 0, \ldots, 7$, permutations corresponds to U_1, U_2, U_3, U_4, U_6 and is given by $T_{AND} + 10T_{XOR}$. Therefore, the time complexity of the keyed permutation U block is $3T_{AND} + 3T_{OR} + 11T_{XOR}$.
- The delay of the inverse permutation block η^{-1} is given by $2T_{XOR}$.

It can be observed that the keyed permutations T_i and U_i from blocks T and U, respectively, can be computed in parallel. However, as shown in the hardware architecture of the TU-decomposition given in Figure 3, there is a dependency for the computation of the U block: the control inputs to the vectorial multiplexer in the U block are the outputs of the T block. Therefore, in order to determine the delay of the TU-decomposition, the time complexity of the keyed permutation T block $(3T_{AND} + 3T_{OR} + 11T_{XOR})$ must be added with the delay of the U block vectorial multiplexer $(2T_{AND} + 3T_{OR})$. Finally, the overall time complexity of the Dillon's APN Permutation $g(x)$ using the TU-decomposition is $5T_{AND} + 6T_{OR} + 11T_{XOR}$, as shown in Table VII.

C. Previous Architectures

Hardware architectures and implementations of the Dillon's 6-bit APN Permutation $g(x)$ in Table I were given in [14],

where the univariate polynomial representation of $g(x)$ and the composition of two functions $f_1(x)$ and $f_2^{-1}(x)$ were considered. As given in Section III, it was shown in [14] that the univariate representation of Dillon's permutation $g(x)$ has algebraic degree 4, while the functions $f_1(x)$ and $f_2^{-1}(x)$ used to express $g(x) = f_1(f_2^{-1}(x))$ have algebraic degrees 2 and 3, respectively. In both cases, the primitive pentanomial $f(y) = y^6 + y^4 + y^3 + y + 1$ over $\mathbb{F}_2$ was used. In [14], the total number of 2-input AND gates and time delays were theoretically computed using the complexities of the modules that constitute the different architectures while that the total number of 2-input XOR gates was given by the synthesis tool used for the implementation. The theoretical complexities and hardware FPGA implementations given in [14] show that the reduction of the algebraic degree of the functions involved in the representation implies a reduction of the complexity of the hardware implementation.

Another univariate representation of Dillon's 6-bit APN permutation with algebraic degree 4 was given in [13] for the primitive pentanomial $f(y) = y^6+y^4+y^3+y+1$. In this case, the total number of 2-input AND and XOR gates and time delays were theoretically computed using the complexities of the modules that constitute the different architectures. Theoretical complexities and hardware implementation results were also given in [13]. To the best of our knowledge, no other hardware architectures of 6-bit Dillon's APN permutation have been given in the literature. For this reason, although Dillon's univariate representation given in [13] is different from the one used in this work and in [14], it is included for comparison in Subsection VI-D and in Section VII.

Other low-complexity architectures of S-boxes considering security properties as differential uniformity and nonlinearity, among others, have been reported in the literature. In [31] and [32], low area and highly secure lightweight 8-bit S-boxes based on field inversion over $\mathbb{F}_{2^8}$ were presented. Lighweight cryptographic 8-bit S-boxes based on the use of two low-complexity 4-bit S-boxes were also given in [33]. Optimized fault-tolerant and error-correcting non-APN 4-bit S-boxes for cryptographic applications with multiple errors detection were also given in [34]. As the above highly efficient S-boxes exhibit differential uniformities greater than 2 (i.e. they are non-APN permutations) and they are not 6-bit S-boxes, they have not been compared with the 6-bit Dillon's APN S-boxes in Subsection VI-D and Section VII.

D. Comparison of Theoretical Complexities

Table VII shows the theoretical complexities of the different hardware architectures of 6-bit Dillon's APN Permutation $g(x)$ given in [13] and [14] and those presented in this paper.

From Table VII, it can be observed that the TU-decomposition architecture here presented exhibits a reduction in the number of 2-input AND gates of 73.8% and 82.81% with respect to the composition and univariate architectures given in [14], respectively, and a reduction of 81.2% and 86.36% in the number of XOR gates with respect to the composition and univariate architectures, respectively. However, the TU-decomposition includes 42 OR gates that are not needed in the other architectures.

TABLE VIII
FPGA EXPERIMENTAL RESULTS

	LUTs	Delay (ns)	LUTs × Delay
Univariate polynomial [13]	6	5.631	33.79
$f_1 \circ f_2^{-1}$ [14]	6	5.645	33.87
Univariate polynomial [14]	6	5.651	33.91
TU-decomposition	6	5.632	33.80

With respect to the theoretical delay, the TU-decomposition has increases of 67% and 150% in T_{AND} and reductions of 73.81% and 54.17% in T_{XOR} with respect to the composition and univariate architectures given in [14], respectively. TU-decomposition also has an additional delay $6T_{OR}$ not included in the other architectures. However, 2-input XOR gates have approximately double delay compared to AND and OR gates, so the TU-decomposition architecture also exhibits a reduced time complexity.

Table VII also includes the theoretical complexities of the different univariate representation of Dillon's APN permutation given in [13] for the primitive pentanomial $f(y) = y^6 + y^4+y^3+y+1$. In this case, the TU-decomposition architecture presented in this paper exhibits a reduction in the number of AND and XOR gates of 88.8% and 91.9%, respectively, with respect to the univariate architecture given in [13]. With respect to the theoretical delay, the TU-decomposition exhibits the same results for [13] as those given for [14].

VII. HARDWARE IMPLEMENTATIONS

In order to compare the new TU-decomposition architecture with the other two architectures of Dillon's permutation given in [14] (univariate polynomial and composition of two functions) and with the different univariate representation of Dillon's APN permutation given in [13], we have performed FPGA implementations of the hardware architecture presented in Section V using the univariate polynomials for the T_i and U_i permutations (referred to as TU-decomposition in Table V) given in Table V and Table VI, respectively.

The architectures have been described in VHDL, synthesized and implemented on Xilinx FPGA Artix-7 XC7A12T-3-CPG238 using VIVADO 2021.2. The experimental post-place and route results obtained are shown in Table VIII, where it can be observed that due to the small size of the Dillon's 6-bit permutation, the synthesis tool can optimize the designs and therefore all the implementations fit in only 6 LUTs (Lookup Tables). Furthermore, the delay (in nanoseconds) and the $area \times delay$ metrics are quite similar for all the implementations.

For this reason, and in order to highlight the differences shown in Table VII between the different architectures, the VHDL hardware descriptions have also been synthesized for ASIC with Synopsys Design Compiler [30] using the TSMC (Taiwan Semiconductor Manufacturing Company) 45 nm. CMOS library "tcbn45gsbwp12tml.db". Design Compiler tool enables concurrent optimization of timing, area, power and test, and its results are correlated within 10% of ASIC physical implementation [30]. Table IX compares the experimental results obtained with Synopsys DC, where *Cells* stands for

TABLE IX
SYNOPSYS EXPERIMENTAL RESULTS

	Cells	GE	Area (μm^2)	Delay (ns)	A×D ($\mu m^2 \cdot ns$)	Power (mW)
Univariate poly. [13]	3961	12250	7923.42	0.86	6814.14	13.54
$f_1 \circ f_2^{-1}$ [14]	1719	5634	3644.19	1.00	3644.19	5.77
Univariate poly. [14]	2467	7674	4963.66	0.71	3524.20	8.33
TU-decomposition	273	867	560.48	0.48	269.03	0.98

the number of cells used for the implementation, *GE* stands for the number of Gate Equivalent 2-input NAND gates, *Area* represents the combinational area in μm^2, *Delay* is the propagation delay in ns, $A \times D$ represents the $Area \times Delay$ given in $\mu m^2 \cdot ns$, and *Power* represents the energy consumption in mW. From the results obtained in Table IX, it can be observed that the TU-decomposition hardware architecture greatly outperforms the area and time complexities of the univariate polynomial and composition of two functions architectures of Dillon's permutation given in [14]. Considering the number of cells, the TU-decomposition architecture exhibits reductions of 84.1% and 88.9% with respect to the composition $f_1 \circ f_2^{-1}$ and univariate architectures, respectively. Considering the number of GE, the TU-decomposition architecture exhibits reductions of 84.6% and 88.7% with respect to the composition $f_1 \circ f_2^{-1}$ and univariate architectures, respectively. Considering the area, the TU-decomposition architecture presents reductions of 84.6% and 88.7% with respect to the composition and univariate architectures, respectively. With respect to the delay, the TU-decomposition exhibits reductions of 52.0% and 32.4% with respect to the composition $f_1 \circ f_2^{-1}$ and univariate architectures, respectively. For the $Area \times Delay$ metrics, the TU-decomposition presents reductions of 92.6% and 92.4% in comparison with the composition $f_1 \circ f_2^{-1}$ and univariate architectures, respectively. Considering the power consumption, the TU-decomposition architecture presents reductions of 83.0% and 88.2% with respect to the composition and univariate architectures in [14], respectively.

With respect to the comparison of the TU-decomposition with the different univariate representation of Dillon's permutation given in [13], the reductions in number of cells, area, delay, $Area \times Delay$ and power consumption are 93.1%, 92.9%, 44.2%, 96.1% and 92.8%, respectively.

Therefore the use of the TU-decomposition for Dillon's APN Permutation can provide hardware implementations with very low complexities in comparison with other approaches.

VIII. CONCLUSION

In this paper, we have presented a low-complexity hardware architecture for the TU-decomposition of APN permutations, using Dillon's APN as a target for this approach. The TU-decomposition can produce significant reductions on the hardware implementation complexity of APN permutations. A detailed description of the different components for the Dillon's APN permutation based on its decomposition into 3-bit keyed permutations (T, U) over $\mathbb{F}_{2^3}$ has been given. We have also presented hardware theoretical complexities and results obtained from FPGA and ASIC implementations for the TU-decomposition hardware proposed architecture. From the results obtained and their comparison with other hardware architectures given in the literature, we have observed that the TU-decomposition architecture presented here greatly outperforms all other hardware approaches in terms of area, delay and area×delay complexities. Furthermore, the hardware architecture presented here can easily be generalized to any permutation on an even number of bits, and is not restricted to just Dillon's permutation; the latter is merely chosen as a particularly relevant practical example for demonstrating our method. The decomposition of Dillon's APN permutation can lead to important reductions of hardware complexity, so our future work will be focused on further study the TU-decomposition of this and other S-boxes.

REFERENCES

[1] C. Carlet., *Boolean Functions for Cryptography and Error Correcting Codes*. Cambridge, U.K.: Cambridge Univ. Press, 2010, ch. 8.

[2] C. de Canniere, A. Biryukov, and B. Preneel, "An introduction to block cipher cryptanalysis," *Proc. IEEE*, vol. 94, no. 2, pp. 346–356, Feb. 2006.

[3] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *J. Cryptol.*, vol. 4, no. 1, pp. 3–72, Jan. 1991.

[4] M. Matsui, "Linear cryptoanalysis method for DES cipher," in *Advances in Cryptology*, vol. 765. Cham, Switzerland: Springer, 1994, pp. 386–397.

[5] K. Nyberg, "Differentially uniform mappings for cryptography," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 765. Cham, Switzerland: Springer, May 1994, pp. 55–64.

[6] F. Chabaud and S. Vaudenay, "Links between differential and linear cryptanalysis," in *Proc. Workshop Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1994, pp. 356–365.

[7] *Advanced Encryption*, Standard FIPS 197, 2001.

[8] K. Nyberg and L. R. Knudsen, "Provable security against differential cryptanalysis," in *Advances in Cryptology—CRYPTO' 92*. Cham, Switzerland: Springer, 1993, pp. 566–574.

[9] X.-D. Hou, "Affinity of permutations on $\mathbb{F}_2^n$," *Discrete Applied Mathematics*, vol. 154, pp. 313–325, 2006.

[10] M. Calderini, M. Sala, and I. Villa, "A note on APN permutations in even dimension," *Finite Fields Their Appl.*, vol. 46, pp. 1–16, Jul. 2017.

[11] J. F. Dillon, "APN polynomials: An update," in *Proc. Int. Conf. Finite Fields Appl.*, Jul. 2009, pp. 1–82.

[12] B. Bilgin, A. Bogdanov, M. Knezevic, F. Mendel, and Q. Wang, "Fides: Lightweight authenticated cipher with side-channel resistance for constrained hardware," in *Proc. 15th Int. Workshop*, 2013, pp. 142–158.

[13] J. L. Imaña, N. Kaleyski, and L. Budaghyan, "Hardware architecture of Dillon's APN permutation for different primitive polynomials," *Microprocessors Microsystems*, vol. 103, pp. 1–10, Nov. 2023.

[14] J. L. Imaña, L. Budaghyan, and N. Kaleyski, "Decomposition of Dillon's APN permutation with efficient hardware implementation," in *Proc. Int. Workshop Arithmetic Finite Fields*, Jan. 2023, pp. 250–268.

[15] L. Perrin, A. Udovenko, and A. Biryukov, "Cryptanalysis of a theorem: Decomposing the only known solution to the big APN problem," in *Proc. 36th Int. Cryptol. Conf.*, Aug. 2016, pp. 93–122.

[16] A. Biryukov, L. Perrin, and A. Udovenko, "Reverse-engineering the S-box of streebog, kuznyechik and STRIBOBr1," in *Proc. 35th Int. Conf. Theory Appl. Cryptograph. Techn.*, May 2016, pp. 372–402.

[17] C. A. Wood, S. P. Radziszowski, and M. Lukowiak, "Constructing large S-boxes with area minimized implementations," in *Proc. IEEE Mil. Commun. Conf.*, Oct. 2015, pp. 49–54.

[18] G. Werner, S. Farris, A. Kaminsky, M. Kurdziel, M. Lukowiak, and S. Radziszowski, "Implementing authenticated encryption algorithm MK-3 on FPGA," in *Proc. IEEE Mil. Commun. Conf.*, Nov. 2016, pp. 1225–1230.
[19] D. F. Stafford, "Evaluating performance and efficiency of a 16-bit substitution box on an FPGA," M.S. thesis, Dept. Comput. Eng., Rochester Inst. Technol., New York, NY, USA, 2021.
[20] A. Biryukov and L. Perrin, "On reverse-engineering S-boxes with hidden design criteria or structure," in *Proc. 35th Int. Cryptol. Conf.*, Aug. 2015, pp. 116–140.
[21] L. Budaghyan and C. Carlet, "Classes of quadratic APN trinomials and hexanomials and related structures," *IEEE Trans. Inf. Theory*, vol. 54, no. 5, pp. 2354–2357, May 2008.
[22] L. Budaghyan, T. Helleseth, and N. Kaleyski, "A new family of APN quadrinomials," *IEEE Trans. Inf. Theory*, vol. 66, no. 11, pp. 7081–7087, Nov. 2020.
[23] L. Budaghyan, *Construction and Analysis of Cryptographic Functions*. Cham, Switzerland: Springer, 2015.
[24] C. Carlet, *Boolean Functions for Cryptography and Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2021.
[25] M. Calderini, "On the EA-classes of known APN functions in small dimensions," *Cryptography Commun.*, vol. 12, no. 5, pp. 821–840, Sep. 2020.
[26] K. Browning, J. Dillon, M. McQuistan, and A. Wolfe, "An APN permutation in dimension six," in *Finite Fields: Theory and Applications* (Contemporary Mathematics), vol. 518. Providence, RI, USA: American Mathematical Society, 2010, pp. 33–42, doi: 10.1090/conm/518.
[27] Comput. Algebra Group. *Magma Computational Algebra System*. Accessed: Jan. 15, 2024. [Online]. Available: http://magma.maths.usyd.edu.au/magma/
[28] J. L. Imana, J. M. Sanchez, and F. Tirado, "Bit-parallel finite field multipliers for irreducible trinomials," *IEEE Trans. Comput.*, vol. 55, no. 5, pp. 520–533, May 2006.
[29] J. L. Imana, "Efficient polynomial basis multipliers for type-II irreducible pentanomials," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 11, pp. 795–799, Nov. 2012.
[30] *Synopsis Design Compiler*. Accessed: Dec. 19, 2023. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html
[31] B. Rashidi, "Lightweight 8-bit S-box and combined S-box/S-box-1 for cryptographic applications," *Int. J. Circuit Theory Appl.*, vol. 49, no. 8, pp. 2348–2362, Aug. 2021.
[32] B. Rashidi, "Compact and efficient structure of 8-bit S-box for lightweight cryptography," *Integr., VLSI J.*, vol. 76, pp. 172–182, Jan. 2021.
[33] B. Rashidi, "Lightweight cryptographic S-boxes based on efficient hardware structures for block ciphers," *ISC Int. J. Inf. Secur.*, vol. 15, no. 1, pp. 137–151, 2023.
[34] B. Rashidi, "Fault-tolerant and error-correcting 4-bit S-boxes for cryptography applications with multiple errors detection," *J. Supercomput.*, vol. 80, no. 1, pp. 1464–1490, 2024.

Lilya Budaghyan received the Ph.D. degree from the University of Magdeburg, Germany, in 2005, and the Habilitation degree from the University of Paris 8, France, in 2013. Since 2018, she has been a member with Norwegian Academy of Technological Sciences (NTVA). She is currently a Professor and the Head of the Selmer Center in Secure Communication, Department of Informatics, University of Bergen, Norway. She conducted her research with Yerevan State University, Armenia; the University of Trento, Italy; and Telecom ParisTech, France. Her research interests include cryptographic Boolean functions and discrete structures and their applications. She was a recipient of the Trond Mohn Foundation Award in 2016, the Young Research Talent Grant from Norwegian Research Council in 2014, the Post-Doctoral Fellowship Award from the Foundation of Mathematical Sciences of Paris in 2012, and the Emil Artin Junior Prize in Mathematics in 2011.

José L. Imaña received the M.Sc. and Ph.D. degrees in physics from the Complutense University of Madrid, Madrid, Spain, in 1989 and 2003, respectively. He was an Electronic Design Engineer with Madrid Institute of Technology, Spain. He is currently with the Department of Computer Architecture and Automation, Complutense University of Madrid, where he was promoted to an Associate Professor with a tenure in 2006. His research interests include algorithms and VLSI architectures for computations in finite fields, computer arithmetic, cryptographic hardware, and post-quantum cryptography. He has been the Promoter and the Co-Founder of the International Workshop on the Arithmetic of Finite Fields (WAIFI).

Nikolay Kaleyski received the bachelor's and master's degrees in theoretical computer science from Charles University, Prague, in 2014 and 2016, respectively, and the Ph.D. degree from the Selmer Centre, University of Bergen, in 2021. In addition to his research activities, he actively reviews articles for several international journals and takes part in outreach and propagational activities for the Selmer Centre and the Department of Informatics, University of Bergen, where he is currently a tenure track Associate Professor. His research interests include classes of cryptographically optimal functions over finite fields, including APN functions, AB functions, and planar functions, and mathematical constructions and related computational and algorithmic questions.