This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



# Extended Generalized Feistel Networks using Matrix Representation to Propose a New Lightweight Block Cipher: LILLIPUT

Thierry P. Berger, Julien Francq, Marine Minier and Gaël Thomas

***Abstract*—While Generalized Feistel Networks (GFNs) have been widely studied in the literature as a building block of a block cipher, we recall in this paper the results of [1] where a unified vision to easily represent them through a matrix representation is proposed. We also introduce a new class of such schemes called Extended Generalized Feistel Networks well suited for cryptographic applications.**

**We instantiate this particular construction into a lightweight block cipher called LILLIPUT analyzing its security and its hardware performances.**

***Index Terms*—Block Ciphers, Generalized Feistel Networks, Matrix Representation, Lightweight Design Proposal, Security Analysis, ASIC implementation.**

## I. INTRODUCTION

In this article, we first recall the results of [1] concerning a matrix representation of GFNs that allows to directly study diffusion properties of GFNs. Then, we extend this matrix representation to introduce a new Feistel network class called Extended Generalized Feistel Networks (EGFNs) with good diffusion properties, already presented in [1]. We provide generic security bounds for this new class of Feistel network and we finally instantiate a particular EGFN into a new particular lightweight block cipher called LILLIPUT. We thus study the security properties of this proposal and present its hardware implementation that compares well to other lightweight ciphers (when implemented in a round-based and in a serialized fashion) especially when both encryption and decryption are necessary on a low-power 65 nm standard-cell library.

This paper is organized as follows: Section II contains the previous works in the litterature regarding GFNs. Section III gives the matrix representation of a GFN, its link with diffusion and shows how each possible GFN can be represented using a particular matrix. Section IV extends GFNs into EGFNs and contains a particular EGFN proposal with good diffusion properties and gives a complete security analysis concerning EGFNs and the particular instantiation. Section V describes the specifications of LILLIPUT, our lightweight block cipher based on the instantiation of our EGFN. Section VI explains our design choices for LILLIPUT especially for its key schedule. In Section VII, we discuss the security of the whole cipher while Section VIII deals with the hardware implementation results. Section IX finally concludes this paper.

This work was partially supported by the French National Agency of Research: ANR-11-INS-011.

Thierry Berger and Gaël Thomas are with XLIM (UMR CNRS 7252), Université de Limoges, 123 avenue A. Thomas, 87060 Limoges Cedex, France, firstname.name@xlim.fr

Julien Francq is with Airbus Defence & Space - CyberSecurity, 1 Bd Jean Moulin, CS 40001, MetaPole, 78996 Elancourt Cedex, France julien.francq@cassidian.com

Marine Minier is with Université de Lyon, INRIA - INSA-Lyon, CITI, F-69621, Villeurbanne, France, marine.minier@insa-lyon.fr

## II. PREVIOUS WORK

While a classical Feistel network, such as DES [2] or Camellia [3], divides a plaintext into 2 $n$-bit-long halves and applies a function $F$ to half of the state before adding the result to the other half, a Generalized Feistel Network (GFN) divides it into $k \geq 2$ $n$-bit-long subblocks and potentially applies a different $F$ function to each subblock. Various GFNs exist in the literature. This includes Type-1 as in CAST-256 [4] where a single $F$ function is used at each round; Type-2 as in HIGHT [5] and CLEFIA [6] and Nyberg's GFNs [7] where $k/2$ $F$ functions are used; Type-3, Source-Heavy (SH) as in SHA-1 [8] and Target-Heavy (TH) as in MARS [9] where more than $k/2$ $F$ functions are used. The pseudorandomness of these constructions is studied in [10], [11], [12] for Type-1, Type-2 and Type-3 and in [11], [13], [12] for SH and TH GFNs. Fig. 1 gives an example of Type-2 GFN. Usually GFNs perform a block-wise cyclic shift in their permutation layer.

In [14] and in [15], two more generic studies are provided: the authors analyzed Type-1, Type-2 and Type-3 GFNs regarding non-cyclic permutations. More precisely, in [14], the concept of maximum diffusion round is introduced. It corresponds to the minimum number of rounds such as every output block depends on every input block. It is a quantifiable measure to maximize the diffusion inside the cipher. The authors of [14] exhaustively searched all the optimum permutations for $k \leq 16$ and found that the diffusion in Type-2 GFNs can be improved. They also showed a lower bound on the maximum diffusion round of Type-2 GFNs and when $k$ is a power of 2, they gave a generic construction based on de Bruijn graphs whose maximum diffusion round is close to the lower bound they found. Besides, they studied the pseudorandomness of these GFNs giving a bound depending on the diffusion delay and their resistance against classical attacks and showed that it is actually improved as well. One of these Type-2 GFNs is used in TWINE [16]. In [15], Yanagihara and Iwata gave the same kind of constructions for Type-1, Type-3, SH and TH GFNs with non-cyclic permutation. For Type-1 and Type-3 GFNs, they showed that the maximum diffusion delay can be improved by changing the permutation while for SH and TH GFNs it cannot. For Type-1 GFNs, they gave an optimum generic construction for any $k$ and identified a necessary and sufficient condition

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



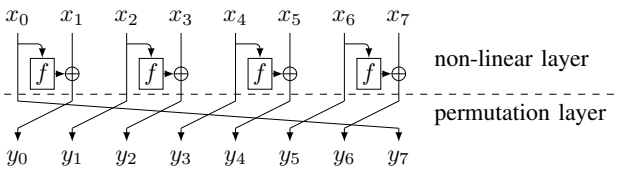


Fig. 1. One round of a Type-2 GFN with $k = 8$ blocks.

for improved Type-3 to have a finite maximum diffusion round. They also evaluated the resistance of all those GFNs against classical attacks and showed that it can be improved in the Type-1 and Type-3 cases.

## III. Matrix Representation of Feistel Networks

Before defining the matrix representation of a GFN, let us introduce a few notations. A GFN divides its input into $k \geq 2$ blocks of $n$ bits each. Let $x_0, \cdots, x_{k-1}$ denote the input blocks of a GFN round and $y_0, \cdots, y_{k-1}$ the corresponding output blocks. A GFN can be separated into two successive layers, as done in [14], [15]: a non-linear layer and a permutation layer, as in Fig. 1. The non-linear layer is made of key-dependent round-functions whose inputs are some of the blocks and whose outputs are added (x-ored) to some other blocks. They are assumed non-zero. The permutation layer is a block-wise permutation of the $k$ blocks. The way the different round-functions are arranged depends on the type of GFN considered, while the permutation is usually the cyclic shift. We further denote by $y_i^r$ the content of the $i$-th block after $r$ rounds.

### A. Diffusion Delay

We say that the input block $x_i$ affects the output block $y_j^r$ if $x_i$ effectively appears in the expression of $y_j^r$ seen as a function of $x_0, \cdots, x_{k-1}$. We say that $x_i$ has diffused at round $r$ if $x_i$ affects every $y_j^r$ for $0 \leq j \leq k-1$. If every input block $x_i$ has diffused at round $r$, we say the GFN has reached full diffusion, that is every output block $y_j^r$ depends on every input block $x_i$. We call full diffusion delay the minimum number of rounds required to reach full diffusion and denote it $d^+$. In fact, the notion of full diffusion delay is a general notion that can be applied to any automaton as done in [17]. In the particular case of GFNs, this is exactly the same notion as the maximum diffusion round introduced in [14].

Another way to see the full diffusion delay is from a graph point of view. For a $k$-block GFN, let us define the associated directed graph as the graph with vertex set $\{0, \cdots, k-1\}$ and such that $(i, j)$ is an edge if the output $y_j$ depends on the input $x_i$ (directly or via a round-function). In other words, this is simply the usual Feistel schemes with outputs folded onto the input with same index. Knowing that, it is easy to see that the notion of *block $x_i$ affecting block $y_j^r$* becomes *there exists a path of length exactly $r$ going from $i$ to $j$*. Thus the full diffusion delay $d^+$ can be alternately defined as the smallest integer $r$ such that for all ordered pair of vertices $(i, j)$ there exists a path of length exactly $r$ going from $i$ to $j$. Two things should now be noticed. First, the full diffusion delay of a GFN depends solely on the structure of this graph and not on the round-functions used in the GFN, provided they are not the zero function. Second, if a GFN is in a full diffusion state at round $r$ then it will remain so at round $r + 1$ because for a vertex $i$, if there is a path of length $r$ to any vertex $j$ then by going through the permutation layer $\mathcal{P}$, there is a path of length $r + 1$ from $i$ to $\mathcal{P}(j)$ for all $j$.

Similarly, we can define full diffusion delay when considering decryption instead of encryption and denote it $d^-$. Following the work of [14], we consider the both-way full diffusion delay $d = \max(d^+, d^-)$. The both-way full diffusion delay $d$ for the different classical GFNs is summed up in Table I. For security reasons, $d$ should be finite.

| GFN Type | SH | TH | Type-1 | Type-2 | Type-3 | Nyberg |
|---|---|---|---|---|---|---|
| $d$ | $k$ | $k$ | $(k-1)^2+1$ | $k$ | $k$ | $k$ |

| GFN Type | [15] Type-1 | [14] Type-2 |
|---|---|---|
| $d$ | $k(k+2)/2-2$ | $2\log_2 k$ |

TABLE I
Both-way full diffusion delay $d$ for various GFNs with $k$ blocks.

### B. Matrix Representation of Feistel Networks

Recall that a GFN is divided into two distinct transformations: first, the non-linear layer and second, the permutation layer, represented by a permutation matrix $\mathcal{P}$. We call matrix representation of the non-linear layer, the matrix denoted $\mathcal{F}$ with an all-one diagonal and with a parameter we call $F$ at position $(i, j)$ if and only if there is a round-function going from $x_j$ to $x_i$. The parameter $F$ is a formal parameter, meaning it merely indicates the presence of a round-function in the GFN, the same $F$ is used for all the different round-functions used throughout the cipher. If one follows the matrix representation idea, one would define the matrix of the whole GFN as $\mathcal{M} = \mathcal{P} \times \mathcal{F}$.

In other words, for a GFN with $k$ blocks, let $\mathcal{M}$ be the $k \times k$ matrix over $\mathbb{Z}[F]$ defined as follows: for indices $0 \leq i, j \leq k-1$, the coefficient at row $i$ and column $j$ of $\mathcal{M}$ is either a 1 if output $y_i$ directly depends on $x_j$, that is without going through a round-function, or a formal parameter $F$, if $y_i$ depends on $x_j$ via a round-function, or 0 otherwise. This corresponds to the definition of Encryption Characteristic Matrix given in [18]. E.g. Fig. 2 gives matrices $\mathcal{M}$, $\mathcal{P}$ and $\mathcal{F}$ of the GFN in Fig. 1.

$$\mathcal{M} = \begin{pmatrix} F & 1 & & & & & & \\ & \frac{1}{F} & 1 & & & & & \\ & & & \frac{1}{F} & 1 & & & \\ & & & & & \frac{1}{F} & 1 & \\ 1 & & & & & & & \end{pmatrix} \quad \mathcal{P} = \begin{pmatrix} & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & & 1 & & \\ & & & & & & 1 & 1 \\ 1 & & & & & & & \end{pmatrix} \quad \mathcal{F} = \begin{pmatrix} \frac{1}{F} & 1 & & & & & & \\ & & & \frac{1}{F} & 1 & & & \\ & & & & & \frac{1}{F} & 1 & \\ & & & & & & & \frac{1}{F} \; 1 \end{pmatrix}$$

Fig. 2. Decomposition of the transition matrix of the GFN given in Fig. 1.

As round-functions in a GFN are unlikely to be linear, such a matrix is not an exact representation. However it still retains enough information to evaluate diffusion; namely which output block $y_i$ is influenced by which input block $x_j$ and whether this is done directly or via a round-function.

An important feature of GFNs is to transform a set of non-invertible round-functions into an invertible permutation. Hence the matrix of the GFN in decryption mode $\mathcal{M}^{-1}$ should not contain inverses of expressions containing a $F$. This translates into $\det(\mathcal{M})$ is independent of $F$, or equivalently $\det(\mathcal{F}) = \pm 1$, as $\mathcal{P}$ is a permutation matrix. This is the case for all of the classical GFNs (SH, TH,...) including those of [14], [15] because the matrix $\mathcal{F}$ is lower triangular with an all-one diagonal.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



Another feature of many GFNs is quasi-involutiveness, that is encryption/decryption is roughly the same process, up to using the direct/inverse permutation layer $\mathcal{P}$. To ensure that, the non-linear layer should be quasi-involutive. Except the Type-3 GFNs where the round-functions must be evaluated sequentially while decrypting, all GFNs non-linear layers are quasi-involutive. We choose to focus on GFNs that satisfy this property:

*Definition 1:* A matrix $\mathcal{M}$ with coefficients in $\{0, 1, F\} \subset \mathbb{Z}[F]$ is a GFN matrix if it can be written as $\mathcal{M} = \mathcal{P}\mathcal{F}$ such that $\mathcal{P}$ is a permutation matrix and the matrix $\mathcal{F}$ satisfies the following conditions:

1) the main diagonal is filled with 1,
2) the off-diagonal coefficients are either 0 or $F$,
3) for each index $i$, row $i$ and column $i$ cannot both have an $F$ coefficient.

In other words, the blocks of the GFN can be partitioned into three categories: blocks that emit (through a round-function), blocks that receive and blocks that do not emit nor receive. This definition encompasses most of the known GFNs, with the exception of the Type-3. The property of quasi-involutiveness comes from the following theorem.

*Theorem 1:* Let $\mathcal{M} = \mathcal{P}\mathcal{F}$ be a GFN according to Definition 1. Then $\mathcal{F}$ is invertible and $\mathcal{F}^{-1} = 2\mathcal{I} - \mathcal{F}$, where $\mathcal{I}$ stands for the identity matrix.

*Proof:* To prove that $\mathcal{F}$ is invertible, we compute $\det(\mathcal{F})$. Because of Condition 3 of Definition 1, for each index $i$ either row $i$ or column $i$ is all-zero except for the diagonal coefficient. Thus by successively expanding the determinant along either row $i$ or column $i$, $\det(\mathcal{F}) = 1$.

To prove $\mathcal{F}^{-1} = 2\mathcal{I} - \mathcal{F}$, we equivalently prove $(\mathcal{F} - \mathcal{I})^2 = 0$. Let $f_{i,j}$ (resp. $f'_{i,j}$) denote the coefficient of $\mathcal{F} - \mathcal{I}$ (resp. $(\mathcal{F} - \mathcal{I})^2$) at row $i$ and column $j$. By definition of the matrix product, for all $i$ and $j$, we have $f'_{i,j} = f_{i,i}f_{i,j} + f_{i,j}f_{j,j} + \sum_{\ell \neq i \atop \ell \neq j} f_{i,\ell}f_{\ell,j} = \sum_{\ell \neq i \atop \ell \neq j} f_{i,\ell}f_{\ell,j}$. In the sum, consider one term $f_{i,\ell}f_{\ell,j}$. As $\ell \neq i$, $f_{i,\ell}$ can either be zero or $F$. But, if $f_{i,\ell}$ is non-zero then the $\ell$-th column of $\mathcal{F}$ contains an $F$ thus, by Condition 3 the $\ell$-th row must not contain any $F$, implying $f_{\ell,j} = 0$ for all $j \neq \ell$. Thus, each term $f_{i,\ell}f_{\ell,j}$ is zero, so $f'_{i,j} = 0$. ∎

Notice that in the case where the outputs of round-functions are xored with other blocks, then matrix $\mathcal{F}^{-1} = 2\mathcal{I} - \mathcal{F}$ is simply $\mathcal{F}$ itself. Besides, we can characterize the matrices $\mathcal{F}$ for which $\mathcal{F}^{-1} = 2\mathcal{I} - \mathcal{F}$ holds.

*Theorem 2:* Let $\mathcal{F}$ be a matrix that verifies Conditions 1 and 2 of Definition 1. If $(\mathcal{F} - \mathcal{I})^2 = 0$ then $\mathcal{F}$ also verifies Condition 3.

*Proof:* Let $f_{i,j}$ be the coefficient of $\mathcal{F} - \mathcal{I}$ at row $i$ and column $j$. For all $i$ and $j$, we have $0 = \sum_{\ell=0}^{k-1} f_{i,\ell}f_{\ell,j} = \sum_{\ell \neq i,j} f_{i,\ell}f_{\ell,j}$. All the coefficients $f_{i,\ell}$ and $f_{\ell,j}$ in the previous equation are off-diagonal, thus are either $F$ or 0. Hence the sum can be zero only if all its terms are zero. For each index $\ell$, we need to prove that row $\ell$ and column $\ell$ cannot both have an $F$ coefficient. Suppose column $\ell$ has an $F$ coefficient, say $f_{i,\ell}$ with $i \neq \ell$. This implies that for all $j \neq \ell$, $f_{\ell,j} = 0$. Thus row $\ell$ has no $F$ coefficient. By transposing, the same goes when considering rows instead of columns. ∎

In other words, the GFNs non-linear layer matrices $\mathcal{F}$ which are quasi-involutive are exactly those where Condition 3 of Definition 1 holds.

Recall that the full diffusion delay can be expressed in term of distance in a directed graph. If one evaluates the matrix $\mathcal{M}$ of the GFN in $F = 1$, we obtain the adjacency matrix of this graph. The full diffusion delay $d^+$ is then the smallest integer such that $\mathcal{M}^{d^+}$ has no zero coefficient. The same goes for the decryption full diffusion delay $d^-$, using $\mathcal{M}^{-d^-}$.

### C. Matrix Equivalences

Now that we have matrices representing GFNs, we define an equivalence relation on them that will help us to find GFNs.

*Definition 2:* Two GFNs matrices $\mathcal{M}$ and $\mathcal{M}'$ are equivalent if there exists a permutation (matrix) $\pi$ of the $k$ blocks such that $\pi\mathcal{M}\pi^{-1} = \mathcal{M}'$.

In other words, two GFNs are equivalent if they are the same up to block reindexation and thus share the same properties, such as a common full diffusion delay. We then have the property of "equivalent decompositions":

*Theorem 3:* Let $\mathcal{M} = \mathcal{P}\mathcal{F}$ and $\mathcal{M}' = \mathcal{P}'\mathcal{F}'$ be two GFNs according to Definition 1 and equivalent under Definition 2. Let also $\pi$ be such that $\pi\mathcal{M}\pi^{-1} = \mathcal{M}'$. Then $\pi\mathcal{P}\pi^{-1} = \mathcal{P}'$ and $\pi\mathcal{F}\pi^{-1} = \mathcal{F}'$.

*Proof:* By hypothesis, we have $\pi\mathcal{P}\mathcal{F}\pi^{-1} = \mathcal{P}'\mathcal{F}'$. Also by definition, $\mathcal{F}$ and $\mathcal{F}'$ have an all-one diagonal and either $F$ or zero elsewhere. Hence $\mathcal{F}$ and $\mathcal{F}'$ both evaluate to the identity matrix $\mathcal{I}$ in $F = 0$. Thus, specifying the above equation in zero, we obtain $\pi\mathcal{P}\pi^{-1} = \mathcal{P}'$, implying $\pi\mathcal{F}\pi^{-1} = \mathcal{F}'$. ∎

In other words, two GFNs are equivalent if and only if both layers are equivalent with the same conjugating element. For example, if one studies a class of GFNs with a fixed $\mathcal{F}$ matrix, as done in [14], [15], Theorem 3 allows to define an equivalence relation on the permutation layer.

### D. Exhaustive Search of Feistel Networks

We investigated all the GFNs according to Definition 1 with $k = 8$ blocks up to equivalence. We consider three parameters: the full diffusion delay $d$, the number of round-functions $s$, and the cost for full diffusion, i.e the number of round-functions required for full diffusion, $c = d \times s$. We found that there is no GFN with cost $c < 24$. However, there are cases where the number of rounds $d$ is a more important criterion than the total cost $c$. For each possible value of $d \leq 12$, Table II gives the minimum number of round-functions $s$ required for an 8-block GFN to fully diffuse in $d$ rounds.

TABLE II
MINIMUM NUMBER $s$ OF FUNCTIONS PER ROUND REQUIRED TO HAVE A FULL DIFFUSION IN $d$ ROUNDS AND CORRESPONDING TOTAL COST $c = s \times d$. FOR EACH CASE, THE NUMBER OF DIFFERENT $\mathcal{F}$ MATRICES ($\#\mathcal{F}$) AND THE TOTAL NUMBER OF GFNS ($\#\mathcal{M}$) ARE ALSO GIVEN UP TO EQUIVALENCE.

| $d$ | 1, 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $s$ | $\infty$ | 16 | 7 | 6 | 4 | 4 | 4 | 3 | 3 | 3 | 2 |
| $c$ | $\infty$ | 48 | 28 | 30 | 24 | 28 | 32 | 27 | 30 | 33 | 24 |
| $\#\mathcal{F}$ | 0 | 1 | 1 | 8 | 3 | 13 | 13 | 1 | 6 | 6 | 1 |
| $\#\mathcal{M}$ | 0 | 5 | 3 | 26 | 9 | 101 | 652 | 18 | 100 | 56 | 5 |

Note that among the GFNs that fully diffuse in $d = 6$, with $s = 4$ round-functions, are the Type-2 GFNs with non-cyclic

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



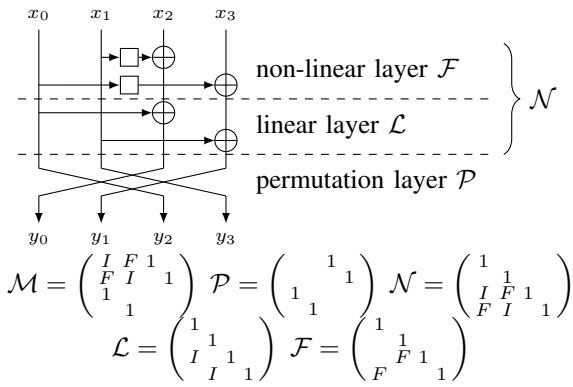


Fig. 3. Overview of an EGFN with three layers and corresponding matrices.

permutation given in [14], which are then diffusion-optimum among the GFNs of Definition 1.

## IV. New Feistel Network Proposals

In this Section, we first introduce Extended Generalized Feistel Networks (EGFNs) as one possible generalization of GFNs. We then introduce a generic EGFN family with good full diffusion delay and cheap cost and we instantiate it for $k = 16$ and a particular permutation layer. Sect. IV-C studies the security of this particular EGFN taking [19] into account.

### A. Extended Generalized Feistel Networks

For a GFN $\mathcal{M} = \mathcal{P}\mathcal{F}$, to achieve quicker diffusion, one can increase the number of round-functions in $\mathcal{F}$. However, this also makes costlier GFNs. The other possibility is to look at the permutation layer $\mathcal{P}$. Definition 1 already allows for block-wise permutations. A possible generalization is to use a linear mapping instead, thus looking for GFNs $\mathcal{M} = \mathcal{G}\mathcal{F}$ with $\mathcal{G}$ an invertible $k \times k$ matrix. This is however much costlier than a simple block-wise permutation and it loses the quasi-involutive property. What we propose is to have a $\mathcal{G}$ which is itself a GFN but with the identity mapping as round-function. In other words, we write $\mathcal{G} = \mathcal{P}\mathcal{L}$ where $\mathcal{P}$ is a permutation matrix and $\mathcal{L}$ is matrix similar to $\mathcal{F}$ but with $I$ off-diagonal non-zero coefficients instead of $F$. We call this matrix $\mathcal{L}$ the linear layer. In that case, the whole Feistel network matrix becomes $\mathcal{M} = \mathcal{P}\mathcal{L}\mathcal{F}$, e.g. Fig. 3. Because matrices $\mathcal{L}$ and $\mathcal{F}$ have common structure, we regroup them into a single matrix $\mathcal{N} = \mathcal{L}\mathcal{F}$, and write $\mathcal{M} = \mathcal{P}\mathcal{N}$. The matrix $\mathcal{N}$ is the new function part of the GFN that contains the non-linear round-functions but now has two formal parameters: $F$ for non-linear round-functions to provide cryptographic security and $I$ for identity round-functions to provide quick diffusion. We call these new schemes Extended Generalized Feistel Networks.

As done in Sect. III-B for GFNs, to be considered an EGFN we require that matrix $\mathcal{M} = \mathcal{P}\mathcal{N}$ is invertible and that $\det(\mathcal{M})$ does not depend on $F$ nor $I$, which translates into $\det(\mathcal{N}) = \pm 1$. Again, we choose to focus on EGFNs that are quasi-involutive. Hence the following definition.

*Definition 3:* A matrix $\mathcal{M}$ with coefficients in $\{0, 1, F, I\} \subset \mathbb{Z}[F, I]$ is an Extended Generalized Feistel Network (EGFN) matrix if it can be written as $\mathcal{M} = \mathcal{P}\mathcal{N}$ such that $\mathcal{P}$ is a permutation matrix and the matrix $\mathcal{N}$ satisfies:

1) the main diagonal is filled with 1,
2) the off-diagonal coefficients are either 0, $F$ or $I$,
3) for each index $i$, row $i$ and column $i$ cannot both contain a non-zero coefficient other than on the diagonal,
4) for each index $i$, if row $i$ contains an $I$ then it also contains an $F$.

As in Sect. III-B, Condition 3 allows to partition the blocks into emitters and receivers. Condition 4 ensures that the pseudorandomness evaluation of EGFNs can be computed (see Sect. IV-C1). Because Definition 3 is essentially the same as Definition 1, the following theorem on quasi-involutiveness is straightforward.

*Theorem 4:* Let $\mathcal{M} = \mathcal{P}\mathcal{N}$ be an EGFN according to Definition 3. Then $\det(\mathcal{N}) = 1$ and $\mathcal{N}^{-1} = 2\mathcal{I} - \mathcal{N}$.

*Proof:* Same as Theorem 1, since Conditions 1, 2 and 3 of Definition 3 are essentially the same as in Definition 1. ■

We can now define matrices $\mathcal{L}$ and $\mathcal{F}$ for the EGFNs of Definition 3.

*Definition 4:* Let $\mathcal{M} = \mathcal{P}\mathcal{N}$ be a EGFN according to Definition 3. Then define matrix $\mathcal{F} \in \mathbb{Z}[F]$ as the evaluation of $\mathcal{N}$ in $I = 0$ and similarly matrix $\mathcal{L} \in \mathbb{Z}[I]$ as the evaluation of $\mathcal{N}$ in $F = 0$.

Theorem 5 proves that this definition is working as intended, i.e. $\mathcal{M} = \mathcal{P}\mathcal{L}\mathcal{F}$.

*Theorem 5:* Let $\mathcal{N}$, $\mathcal{F}$ and $\mathcal{L}$ be defined as in Definition 4, then $\mathcal{N} = \mathcal{L} + \mathcal{F} - \mathcal{I}$ and $\mathcal{N} = \mathcal{L} \times \mathcal{F} = \mathcal{F} \times \mathcal{L}$.

*Proof:* The first equation is a straightforward consequence of the definition of $\mathcal{N}$, $\mathcal{L}$ and $\mathcal{F}$. As for the second, let $a_{i,j}$ be the coefficient at row $i$ and column $j$ of matrix $\mathcal{L}\mathcal{F}$ and show that $a_{i,i} = 1$ and $a_{i,j} = \mathcal{L}_{i,j} + \mathcal{F}_{i,j}$ otherwise (with obvious notations). Write $a_{i,i} = \mathcal{L}_{i,i}\mathcal{F}_{i,i} + \sum_{\ell \neq i} \mathcal{L}_{i,\ell}\mathcal{F}_{\ell,i}$. Then $a_{i,i} = \mathcal{L}_{i,i}\mathcal{F}_{i,i} = 1$ because all terms in the rightmost sum are 0 as a consequence of Condition 3 of Definition 3. For the same reason, if $i \neq j$, $a_{i,j} = \mathcal{L}_{i,i}\mathcal{F}_{i,j} + \mathcal{L}_{i,j}\mathcal{F}_{j,j} + \sum_{\substack{\ell \neq i \\ \ell \neq j}} \mathcal{L}_{i,\ell}\mathcal{F}_{\ell,j}$ and then $a_{i,j} = \mathcal{L}_{i,j} + \mathcal{F}_{i,j}$. ■

Finally, the last thing to update to EGFNs is the equivalence relation. The definition of two equivalent EGFNs $\mathcal{M}$ and $\mathcal{M}'$ is the same as for GFNs, the only difference being that $\mathcal{M}$ and $\mathcal{M}'$ now also have $I$ coefficients. In other words, a conjugating element $\pi$ of $\mathcal{M}$ and $\mathcal{M}'$ exchanges the positions of $F$'s, as well as the positions of $I$'s but it cannot exchange an $F$ and an $I$. The analogous of Theorem 3 is straightforward.

*Theorem 6:* Let $\mathcal{M} = \mathcal{P}\mathcal{L}\mathcal{F}$ and $\mathcal{M}' = \mathcal{P}'\mathcal{L}'\mathcal{F}'$ be two equivalent EGFNs defined by Definition 3. Let also $\pi$ be such that $\pi\mathcal{M}\pi^{-1} = \mathcal{M}'$. Then $\pi\mathcal{P}\pi^{-1} = \mathcal{P}'$, $\pi\mathcal{L}\pi^{-1} = \mathcal{L}'$ and $\pi\mathcal{F}\pi^{-1} = \mathcal{F}'$.

*Proof:* Same as Thm. 3 by evaluating $I$, $F$ or both in 0. ■

### B. An Interesting Family and An Efficient Instantiation

*1) An Interesting Family:* When looking at optimal EGFNs, we obtained the following result:

*Theorem 7:* Given an EGFN as defined in Def. 3 with $k \geq 4$ blocks, $k$ even, and its associated matrices $\mathcal{M}$, $\mathcal{N}$ and $\mathcal{P}$ with $\mathcal{N}$ the matrix representing the non-linear and linear layers of the EGFN and with $\mathcal{P}$ the matrix of the permutation layer defined by a permutation $\pi$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



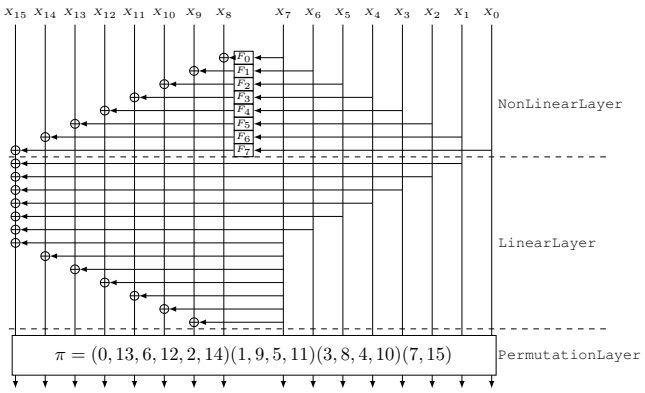


Fig. 4. The EGFN that reaches full diffusion in $d = 4$ rounds. The permutation $\pi$ is given as a product of cycles and can also be found in Table III.

1) If the matrix $\mathcal{N}$ is equal to $\mathcal{N} = \left(\begin{smallmatrix} \mathcal{I} & 0 \\ \mathcal{A} & \mathcal{I} \end{smallmatrix}\right) \in \mathbb{Z}[F, I]$ where $\mathcal{I}$ is the $\frac{k}{2} \times \frac{k}{2}$ identity matrix and where $\mathcal{A}$, the lower left quarter of $\mathcal{N}$, is such that $\mathcal{A} = \begin{pmatrix} (0) & F & I \\ & \ddots & I \\ F & (0) & \vdots \\ F & I & \cdots & I \end{pmatrix}$;
2) And if $\mathcal{P}$ globally exchanges emitters (blocks $x_0$ to $x_{k/2-1}$) with receivers (blocks $x_{k/2}$ to $x_{k-1}$) with $\pi(k/2-1) = k-1$ and $\pi(k-1) = k/2-1$;

Then its diffusion delay $d$ is equal to 4.

*Proof:* Writing $\mathcal{N} = \left(\begin{smallmatrix} \mathcal{I} & 0 \\ \mathcal{A} & \mathcal{I} \end{smallmatrix}\right) \in \mathbb{Z}[F, I]$ and the permutation layer $\mathcal{P}$ as $\mathcal{P} = \left(\begin{smallmatrix} 0 & \mathcal{P}_1 \\ \mathcal{P}_2 & 0 \end{smallmatrix}\right)$ by definition of $\pi$, then the matrix of the EGFN becomes $\mathcal{M} = \mathcal{P}\mathcal{N} = \left(\begin{smallmatrix} \mathcal{P}_1\mathcal{A} & \mathcal{P}_1 \\ \mathcal{P}_2 & 0 \end{smallmatrix}\right)$. Computing $\mathcal{M}^3 = \begin{pmatrix} (\mathcal{P}_1\mathcal{A})^3 + \mathcal{P}_1\mathcal{A}\mathcal{P}_1\mathcal{P}_2 + \mathcal{P}_1\mathcal{P}_2\mathcal{P}_1\mathcal{A} & (\mathcal{P}_1\mathcal{A})^2\mathcal{P}_1 + \mathcal{P}_1\mathcal{P}_2\mathcal{P}_1 \\ \mathcal{P}_2(\mathcal{P}_1\mathcal{A})^2 + \mathcal{P}_2\mathcal{P}_1\mathcal{P}_1\mathcal{P}_2 & \mathcal{P}_2\mathcal{P}_1\mathcal{A}\mathcal{P}_1 \end{pmatrix}$ shows it still has zero coefficients, as $\mathcal{P}_2\mathcal{P}_1\mathcal{A}\mathcal{P}_1$ does. Thus, $d^+ \geq 4$. Compute then $\mathcal{M}^4 = \begin{pmatrix} a & (\mathcal{P}_1\mathcal{A})^3 + \mathcal{P}_1\mathcal{A}\mathcal{P}_1\mathcal{P}_2\mathcal{P}_1 + \mathcal{P}_1\mathcal{P}_2\mathcal{P}_1\mathcal{A}\mathcal{P}_1 \\ b & \mathcal{P}_2(\mathcal{P}_1\mathcal{A})^2\mathcal{P}_1 + (\mathcal{P}_2\mathcal{P}_1)^2 \end{pmatrix}$ with $a = (\mathcal{P}_1\mathcal{A})^4 + (\mathcal{P}_1\mathcal{A})^2\mathcal{P}_1\mathcal{P}_2 + \mathcal{P}_1\mathcal{A}\mathcal{P}_1\mathcal{P}_2\mathcal{P}_1\mathcal{A} + \mathcal{P}_2^2(\mathcal{P}_1\mathcal{A})^2 + \mathcal{P}_2^2\mathcal{P}_1\mathcal{P}_2$ and $b = \mathcal{P}_2(\mathcal{P}_1\mathcal{A})^3 + \mathcal{P}_2\mathcal{P}_1\mathcal{A}\mathcal{P}_1\mathcal{P}_2 + (\mathcal{P}_2\mathcal{P}_1)^2\mathcal{A}$. Thus $\mathcal{M}^4$ has no zero coefficient because by definition $\mathcal{P}_1$ is of the form: $\mathcal{P}_1 = \begin{pmatrix} & & 0 \\ & \mathcal{P}_1' & \vdots \\ & & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$. Thus, as the last row and the last column of $\mathcal{A}$ has no zero, the product $(\mathcal{P}_1 A)^2$ has only non zero coefficients. Thus, the condition $\pi(k-1) = k/2-1$ implies that $d^+ = 4$. The reasoning is the same on $\mathcal{M}^{-1}$ to prove that $d^- = 4$ and finally that $d = 4$. ■

Thanks to Theorem 7, we then have a family of EGFNs with $s = \frac{k}{2}$ round-functions and a diffusion delay of $d = 4$, thus with total cost $c = 2k$. In comparison, [14] gives a family of Type-2 GFNs that diffuse in $d = 2\log_2 k$ rounds. Their total cost is then $c = k\log_2 k$. For $k > 4$, we achieve full diffusion at a cheaper cost than they do.

*2) An Efficient Instantiation:* We give here a particular case of EGFN deduced from Theorem 7. This EGFN with $k = 16$ blocks is depicted in Fig. 4. It is a modified version of the one given in [1]. It was indeed noticed in [19] that the EGFN proposed in [1] had a much lower resistance to differential and linear cryptanalysis than the initial estimation. We then modified the EGFN allowing any permutation layer $\mathcal{P}$.

The permutation $\pi$ given in Table III has been chosen among the 37108 possible permutations (up to block reindexing equivalence) that fulfill the conditions of Theorem 7. It has been chosen to maximize the number of active S-boxes on 18, 19 and 20 rounds as shown in Section IV-C2a.

TABLE III
BLOCK PERMUTATION $\pi$ AND ITS INVERSE.

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\pi(i)$ | 13 | 9 | 14 | 8 | 10 | 11 | 12 | 15 | 4 | 5 | 3 | 1 | 2 | 6 | 0 | 7 |
| $\pi^{-1}(i)$ | 14 | 11 | 12 | 10 | 8 | 9 | 13 | 15 | 3 | 1 | 4 | 5 | 6 | 0 | 2 | 7 |

### C. Security Analysis of our proposed Feistel Scheme

As done in [14], we analyze the security of the proposed EGFN scheme described in Section IV-B2 with essentially $k = 16$ as parameter regarding first the pseudorandomness of the scheme and second its resistance to classical attacks. Note that the pseudorandomness bounds obtained are generic and essentially depend on the $d$ value.

*1) Pseudorandomness:* As we have defined a new block cipher structure, it is legitimate to introduce the pseudo-random-permutation advantage (prp-advantage) and the strong-pseudo-random-permutation advantage (sprp-advantage) of an adversary as done in several works such as [20], [11], [21]. For this purpose, we introduce the two advantage notations as:

$$\mathrm{Adv}_{\mathsf{C}}^{\mathsf{prp}}(q) = \max_{\mathsf{A}:q\text{-}\mathsf{CPA}} \left|\Pr[\mathsf{A}^{\mathsf{C}} = 1] - \Pr[\mathsf{A}^{\mathsf{P}_n} = 1]\right| \quad (1)$$

$$\mathrm{Adv}_{\mathsf{C}}^{\mathsf{sprp}}(q) = \max_{\mathsf{A}:q\text{-}\mathsf{CCA}} \left|\Pr[\mathsf{A}^{\mathsf{C},\mathsf{C}^{-1}} = 1] - \Pr[\mathsf{A}^{\mathsf{P}_n,\mathsf{P}_n^{-1}} = 1]\right| \quad (2)$$

where $\mathsf{C}$ is the encryption function of an $n$-bit block cipher composed of uniform random functions (URFs) as internal modules [20] whereas $\mathsf{C}^{-1}$ is its inverse; $\mathsf{P}_n$ is an $n$-bit uniform random permutation (URP) uniformly distributed among all the $n$-bit permutations; $\mathsf{P}_n^{-1}$ is its inverse. The adversary, $\mathsf{A}$, tries to distinguish $\mathsf{C}$ from $\mathsf{P}_n$ using $q$ queries in a $\mathsf{CPA}$ (Chosen Plaintext Attack) attack and tries to distinguish, always using $q$ queries, $(\mathsf{C}, \mathsf{C}^{-1})$ from $(\mathsf{P}_n, \mathsf{P}_n^{-1})$ in a $\mathsf{CCA}$ (Chosen Ciphertext Attack) attack. The notation means that the final guess of the adversary $\mathsf{A}$ is either 0 if $\mathsf{A}$ thinks that the computations are done using $\mathsf{P}_n$, or 1 if $\mathsf{A}$ thinks that the computations are done using $\mathsf{C}$. The maximums of Equations (1,2) are taken over all possible adversaries $\mathsf{A}$ with $q$ queries and an unbounded computational power. Many results [20], [11], [21] have appeared evaluating the security of Feistel variants in this model. For example, Luby and Rackoff in their seminal work [20] proved the security of a $2n$-bit classical Feistel cipher with 3 rounds in the $\mathsf{prp}$ model and with 4 rounds in the $\mathsf{sprp}$ model considering that the classical Feistel cipher is composed of $n$-bit-to-$n$-bit URFs (the bounds they found are in $\mathcal{O}(q^2/2^n)$ for both cases). Those results have been generalized in [10] and [22].

To prove the bounds of our scheme in those models, we follow the methodology of [14] based on the results of [23]. To do so, we introduce the following notations: Let $\Phi_{kn,r}$ denote our $k$-block scheme acting on $n$-bit blocks, using $r$ rounds and with diffusion delay $d$. We first introduce the following definition that will be useful for the next lemma:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



*Definition 5:* Let $H$ be a keyed permutation over $(\{0,1\}^n)^k$ and let $\mathbf{x} = (x_0, \cdots, x_{k-1}) \in (\{0,1\}^n)^k$ with $\mathbf{x}_{[i]} = x_i$. $H$ is said to be an $\epsilon$-AU ($\epsilon$ Almost Universal) function if:

$$\max_{\mathbf{x} \neq \mathbf{x'}} \Pr[H(\mathbf{x})_{[i]} = H(\mathbf{x'})_{[i]}, \text{ for } i \in \{0, \cdots, k-1\}] \leq \epsilon$$

*Lemma 1:* Let $H$ and $H'$ be two keyed permutations over $(\{0,1\}^n)^k$ that are respectively $\epsilon$-AU and $\epsilon'$-AU; Let denote by $\Phi_{kn,r}$ our $r$-round EGFN with $k$ branches acting on $n$-bit blocks with a diffusion delay $d$ where all $n$-bit round-functions are independent URFs. Then we have:

$$\mathrm{Adv}^{\mathsf{prp}}_{\Phi_{kn,2} \circ H}(q) \leq \left(\epsilon + \frac{k}{2^n}\right) \cdot \binom{q}{2} \quad (3)$$

$$\mathrm{Adv}^{\mathsf{sprp}}_{H'^{-1} \circ \Phi_{kn,2} \circ H}(q) \leq \left(\epsilon + \epsilon' + \frac{k}{2^{n-1}}\right) \cdot \binom{q}{2} \quad (4)$$

*Proof:* Intuitively, for equation (3), this lemma uses the fact that after the application of $H$ the inputs of function $\Phi_{kn,2}$ are sufficiently distinct and are random strings. We then have rare collisions at the outputs of $\Phi_{kn,2}$. For equation (4), same arguments hold in both directions. The proof of this lemma is omitted as it is similar to those of Theorem 3.1 and Theorem 3.2 of [13] or is a direct extension of Lemma 9 and Theorem 7 of [22]. ∎

*Theorem 8:* Given the $r$-round EGFN $\Phi_{kn,r}$ with $k$ branches acting on $n$-bit blocks with a diffusion delay $d$ where all $n$-bit round functions are independent URFs. Then we have:

$$\mathrm{Adv}^{\mathsf{prp}}_{\Phi_{kn,d+2}}(q) \leq \frac{kd}{2^n} q^2 \quad (5)$$

$$\mathrm{Adv}^{\mathsf{sprp}}_{\Phi_{kn,2d+2}}(q) \leq \frac{kd}{2^{n-1}} q^2 \quad (6)$$

*Proof:* To demonstrate Theorem 8, we have first to show that $\Phi_{kn,d}$ is an $\epsilon$-AU function and second that $\overline{\Phi_{kn,d}}$ which is $\Phi^{-1}_{kn,d}$ without the final shuffle is also an $\epsilon$-AU function.

Let us first demonstrate (as done in [14]) that $\forall i \in \{0, \cdots, k-1\}$: $\Pr[\Phi_{kn,d}(\mathbf{x})_{[i]} = \Phi_{kn,d}(\mathbf{x'})_{[i]}] \leq \frac{d}{2^n}$.

Without loss of generality, we assume that $(x_{k/2-1}, x_{k/2-2}, x_{k/2+1}) \neq (x'_{k/2-1}, x'_{k/2-2}, x'_{k/2+1})$. We then estimate the probability that $\Phi_{kn,d}(\mathbf{x})_{[0]} = \Phi_{kn,d}(\mathbf{x'})_{[0]}$. By definition of $d$, there is an appropriate path of length $d$ on the graph of $\Phi_{kn,d}$ starting and finishing at vertex 0. For $h = 1, \cdots, d$, we can define a sequence of internal inputs $Y_h = \Phi_{kn,h}(\mathbf{x})_{[s(h)]}$ following the appropriate path. It is straightforward to see that $\Pr[Y_1 = Y'_1] = \Pr[F(x_{k/2-2}) \oplus x_{k/2-1} \oplus x_{k/2+1} = F(x'_{k/2-2}) \oplus x'_{k/2-1} \oplus x_{k/2+1}] \leq 1/2^n$ because the round function $F$ is a URF (using the same reasoning, this result also holds for probabilities of the other branches, even the branch $x_{k-1}$ due to the presence of an $F$ function). Then, $\Pr[Y_d = Y'_d]$ is over bounded by $\sum_{j=2}^{d} \Pr[Y_j = Y'_j | Y_{j-1} \neq Y'_{j-1}] + \Pr[Y_1 = Y'_1] \leq d/2^n$ because all round functions are independent, i.e. $\Pr[Y_j = Y'_j | Y_{j-1} \neq Y'_{j-1}] \leq 1/2^n$. This proves the result. Thus, $\Phi_{kn,h}$ is a $\frac{kd}{2^n}$-AU function. Equation (5) of Theorem 8 is proved using equation (3) of Lemma 1.

To prove the second equation of Theorem 8, we use exactly the same reasoning on $\overline{\Phi_{kn,d}}$ to show that $\Pr[Y_d = Y'_d] \leq d/2^n$ with $Y_h = \overline{\Phi_{kn,h}}(\mathbf{x})_{[s(h)]}$ for $h = 1, \cdots, d$. We then deduce that $\overline{\Phi_{kn,d}}$ is a $\frac{kd}{2^n}$-AU function. Combining the fact that $\Phi_{kn,d}$ is a $\frac{kd}{2^n}$-AU function and that $\overline{\Phi_{kn,d}}$ is a $\frac{kd}{2^n}$-AU function through Equation (4) of Lemma 1, we obtain Equation (6). ∎

*2) Evaluation of Security against classical attacks:* Whereas the bounds obtained in the context of differential and linear cryptanalysis are mainly linked with the studied cipher, the integral and impossible differential attacks are structural attacks and can be studied in a generic way. Thus, we present in this section the bounds for differential/linear cryptanalysis fully instantiated for the EGFN given in Fig. 4. The first bounds given for the integral and impossible differential cases are generic for all possible EGFNs before to be instantiated for the particular example given in Fig. 4.

*a) Differential/Linear Cryptanalysis:* Differential and linear cryptanalysis are the most famous attacks on block ciphers. They have been introduced respectively in [24] and in [25]. Since their discovery, substantial work has been done to first show the links between those two forms of cryptanalysis [26] and to find better ways to prevent those attacks from happening for a given cipher [27]. The usual consensus about this last point is to count the minimal number of active S-boxes crossed all along the cipher by differential and linear characteristics denoted here respectively by $AS_D$ and $AS_L$. From those numbers, we can estimate the induced maximal differential/linear probability depending on the maximal differential/linear probability of the $F_i$ function (for $i$ from 0 to 7 here) denoted by $DP/LP$. Usually, we consider that $DP = LP = 2^{-2}$ supposing that our 16 branches scheme acts at nibble level.

Moreover, the best differential/linear attack against the cipher has a complexity of about $DP^{AS_D}$ (respectively $LP^{AS_L}$) operations. Thus, a cipher is supposed to be secure against differential/linear cryptanalysis as soon as $1/(DP^{AS_D})$ (respectively $1/(LP^{AS_L})$) is greater than the entire codebook.

It was noted in [19] that the EGFN initially proposed in [1] had far fewer minimal number of active S-boxes than expected. Thus we changed the permutation layer to thwart this problem. Among all the permutations allowed by Theorem 7, we chose the one (given in Table III) with the highest number of active S-boxes after 18, 19 and 20 rounds. The minimal number of active S-boxes up to 20 rounds for this permutation is given in Table IV. Those bounds are obtained using a branch and bound algorithm based on binary representation of linear masks or differences. Thus, the algorithm runs through all possible values considering that the linear part could cancel linear masks/differences. The bounds given here take into account all the possible cancellation paths. This last fact explains that the minimal numbers of active S-boxes found for our scheme are lower than the ones given for the best permutation found in [14] (and also used in TWINE [16]). Indeed, crossing the linear layer can cancel some linear masks/differences which is not the case for TWINE.

Moreover, to be sure that there is no exploitable "linear hulls", we have tested for respectively 4, 5 and 6 rounds that the mean of the observed linear bias is equal to the mean of the theoretical linear bias using $2^{22}$ plaintexts repeated on 1000 different keys for 4 and 5 rounds and using $2^{26}$ plaintexts repeated on 100 different keys for 6 rounds (beyond 6 rounds

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



the computations become infeasible). We have tested if there exists a bias on each output nibble using one of the "best" linear mask which is $0x0a \rightarrow 0x0a$. The results of the Z tests performed on each sample are sufficiently good to be convinced that no bias behaves better than expected.

*b) Integral Attack:* In [28] L. Knudsen and D. Wagner analyzed integral cryptanalysis as a dual to differential attacks particularly applicable to block ciphers with bijective components. A first-order integral cryptanalysis considers a particular collection of $m$ words in the plaintexts and ciphertexts that differ on a particular component. The aim of this attack is thus to predict the values in the sums (i.e. the integral) of the chosen words after a certain number of rounds of encryption. The same authors also generalize this approach to higher-order integrals: the original set to consider becomes a set of $ml$ vectors which differ in $l$ components and where the sum of this set is predictable after a certain number of rounds. The sum of this set is called an $l$th-order integral. In [29], the authors improve the already known results in the case of Feistel structure noticing that computations of the XOR sum of the partial decryptions can be divided into two independent parts through a meet-in-the-middle approach. We define the following properties for a set of $2^n$ $n$-bit words:

- '$C$' (for Constant) in the $i$th entry, means that the values of all the $i$th words in the collection of texts are equal.
- '$A$' (for All) means that all words in the collection of texts are different.
- '?' means that the sum of words cannot be predicted.
- '$B$' (for Balanced) means that the sum of all words taken on a particular word is equal to 0.

Integral characteristics are of the form $(\alpha \rightarrow \beta)$ with $\alpha \in \{C, A\}^k$ containing at least one $A$ and $\beta \in \{C, A, ?, B\}^k$ containing at least one $A$ or one $C$ or one $B$. To find integral characteristics, we apply the method and the properties described in [30]. We first look at characteristics $\alpha$ containing exactly one $A$ subblock, the other ones being $C$. By definition of $d$, the state after $d$ rounds does not contain $C$. If we assume that the state after $d$ rounds contains two $A$s for the most favorable $n$-bit blocks, say $i$ and $j$ (for example blocks with indices $k/2-1$ and $k-1$), then by adding one more round, the state at the subblock $s = \mathcal{P}(j)$ becomes a $B = (F(A) \oplus A)$ or a $B = (F(A) \oplus A \oplus A)$ subblock for the simplest transformations, the other transformations straightforwardly give same kind of results. After one more round, the state at indice $t = \mathcal{P}(s)$ is of the same form because no $F$ function has been crossed. Adding another round transforms this state into a state of the form $? = F(B) \oplus ?$ or $? = F(B) \oplus B \oplus ?$ or more complicated expressions for $y_1$. Therefore, an integral characteristic (containing one $A$ and $k-1$ Cs) exists for at most $d+2$ rounds. If we try to extend at the beginning this first order characteristic into an $l$th-order characteristic, we can add at most $d$ rounds at the beginning due to the definition of $d$. Thus, the maximum number of rounds that can be reach by an $l$th order integral characteristic is $d + d + 2 = 2d + 2$.

We confirm this bound by experimental analysis and for the dedicated example of EGFN given in Fig. 4. More precisely, we first construct the following first order integral property on 5 rounds: $(C, C, C, A, C, \cdots, C)$ gives after 6 rounds $(?, B, ?, \cdots, ?)$. There are 7 equivalent first order integral properties for seven different positions of the active nibble.

This first order integral can be extended by 4 rounds at the beginning using a 15th order integral property: $(A^{15}, A^{15}, A^{15}, C, A^{15}, \cdots, A^{15})$ can be turned after four rounds into a property of the form $(C, C, C, A^{15}, C, \cdots, C)$. Thus, we can exhibit a 15-order integral property on 9 rounds.

*c) Impossible Differential Attack:* Impossible differential cryptanalysis [31] is a form of differential cryptanalysis for block ciphers. While ordinary differential cryptanalysis tracks differences that propagate through the cipher with a probability as large as possible, impossible differential cryptanalysis exploits differences with 0 probability in intermediate rounds of the cipher to sieve wrong key candidates.

More formally, impossible differential attacks are represented by a differential transition $\alpha \not\rightarrow \beta$ with $\alpha, \beta \in (\{0,1\}^n)^k$ for a cipher $E$ with $k$ $n$-bit blocks with $Pr[E(x) + E(x+\alpha) = \beta] = 0$ for any $x$. Intuitively, if we want to form an impossible differential transition for our EGFN, we need to first form the first part of the impossible differential on $r_1$ rounds between the input differential $\alpha^0 = (\alpha_0^0, \cdots, \alpha_{k-1}^0)$ and the output differential after $r_1$ rounds $\alpha^{r_1} = (\alpha_0^{r_1}, \cdots, \alpha_{k-1}^{r_1})$. Then, we form the second part of the impossible differential in the decryption direction on $r_2$ rounds between $\beta^0 = (\beta_0^0, \cdots, \beta_{k-1}^0)$ and $\beta^{r_2} = (\beta_0^{r_2}, \cdots, \beta_{k-1}^{r_2})$. Then, the impossible differential on $r_1 + r_2$ rounds is $\alpha^0 \not\rightarrow \beta^0$ if the differences $\alpha^{r_1}$ and $\beta^{r_2}$ are not compatible in the middle of the cipher.

From the $\mathcal{U}$-method of [18] or the UID-method of [32], the differences $\alpha^{r_1}$ and $\beta^{r_2}$ can be of the types: zero difference (denoted 0), nonzero unfixed difference (denoted $\delta$), non zero fixed difference (denoted $\gamma$), exclusive-or of nonzero fixed and nonzero unfixed difference (denoted by $\delta + \gamma$), and unfixed difference (denoted $t$). As done in [14], we can determine the maximal number of rounds for an impossible differential attack using the $\mathcal{U}$-method described in [18]. This number of rounds mainly depends on $d$ as shown below:

• If $\alpha_i^d$ for $i$ in $\{k/2, \cdots, k-1\}$ has type $\gamma$, there exists a data path, $P$ that does not pass through any $F$ (i.e. the equation corresponding to this path does not contain $\alpha_i^0$ as a part of arguments of $F$). If $\alpha_j^d$ for $j$ in $\{0, \cdots, k/2-1\}$ has type $\delta$ then $\alpha_l^{d+1}$ with $l = \mathcal{P}(i)$ has type $\delta + \gamma$. If $\beta_k^d$ has type $\gamma$, we are able to construct an impossible differential attack on $2d+1$ rounds.

• If all the data paths pass through at least one $F$ function, then both $\alpha^d$ and $\beta^d$ do not contain differences of type neither $\gamma$ nor 0. Thus, we can only mount differences on $d-1$ rounds for the direct sens (i.e. $\alpha$ difference) and on $d$ rounds for the decryption sens (i.e. $\beta$ difference). The maximal number of rounds for this type of impossible differential attack is $2d-1$.

• By definition of $d$, there exists $\alpha^0$ such that $\alpha_i^{d-1}$ has type $\gamma$ for some $i$. Similarly, there exists $\beta^0$ with $\beta_j^{d-1}$ has type $\gamma'$ for some $j$. If $i = j$ and $\gamma \neq \gamma'$, we can construct an impossible differential attack on $2d-2$ rounds.

Finally, for the dedicated example of EGFN given in Fig. 4, implementing the $\mathcal{U}$-method described in [33], we found 2 impossible differential characteristics on 8 rounds of the form:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



TABLE IV
MINIMAL NUMBER OF ACTIVE S-BOXES FOR EVERY ROUND.

| Round | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $AS_D$ | 0 | 1 | 2 | 3 | 5 | 9 | 12 | 14 | 15 | 17 | 21 | 24 | 26 | 28 | 29 | 31 | 36 | 38 | 40 | 42 |
| $AS_L$ | 0 | 1 | 2 | 3 | 5 | 9 | 13 | 14 | 15 | 17 | 20 | 23 | 26 | 28 | 30 | 31 | 34 | 38 | 40 | 42 |

$(0,0,0,0,0,0,0,\alpha,0,0,0,0,0,0,0,0)$ that gives after 4 rounds $(*,*,*,*,*,*,*,*,0,*,*,*,*,*,*,*)$ with $\alpha$ a non-zero difference and $*$ any difference. This difference is incompatible with $(0,0,0,\beta,0,0,0,0,0,0,0,0,0,0,0,0)$ that gives after 4 backward rounds $(*,*,*,*,*,*,*,*,\neq 0,*,\ *,*,*,*,*,*,*,*)$ with $\beta$ a non-zero difference.

## V. SPECIFICATIONS OF THE LIGHTWEIGHT BLOCK CIPHER LILLIPUT

In this section we present our dedicated lightweight block cipher LILLIPUT that is based on the EGFN described in Fig. 4. Our goal is for LILLIPUT to fit in harsh constraints environments where *both* encryption and decryption are mandatory, like some protocols described in ISO 9798-2 [34]. The targetted environment of LILLIPUT is hardware and the primary metric we will minimize is chip area. However, we also chose not to deter software performances too much, thus avoiding bitwise permutations or easying bitslice implementations; and we also tried to reduce latency.

But first, let us introduce the following notation: For binary strings, denote concatenation by $||$; To emphasize a string $x$ is of length $n$, we may write it $x_{(n)}$; We denote by $\gg i$ and $\ll i$ respectively the right and left shifts, and by $\ggg i$ and $\lll i$ the right and left rotations of $i$ bits.

### A. Encryption Process

LILLIPUT is a 64-bit block cipher with an 80-bit key. The whole encryption process is depicted in Fig. 5. As previously explained, LILLIPUT uses an Extended Generalized Feistel Network (EGFN) with 64-bit state and a round function acting at nibble level. The state $X$ is seen as 16 nibbles. These nibbles are denoted $X_{15},\cdots,X_0$. It is composed of 30 rounds, *i.e.* 30 repetitions of a single EGFN called `OneRoundEGFN`, depicted in Fig. 4 where each $F_i$ for $i$ from 0 to 7 is defined as $F_i = S(X_{7-i} \oplus RK_i)$ where $S$ is an S-box that acts at nibble level and where $RK_i$ is the nibble of position $i$ of the 32-bit subkey $RK^j$ of the round $j$. The 30 32-bit subkeys $RK^j$ are generated from the master key using the key schedule.

Note that the last round misses a `PermutationLayer` for involution reasons.

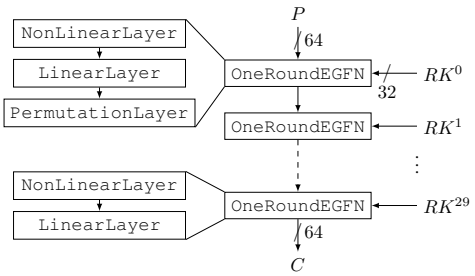


Fig. 5. LILLIPUT Encryption

The S-box $S$ used here is the 4-bit S-box given in Table V.

TABLE V
S-BOX IN HEXADECIMAL NOTATION.

| $x_{(4)}$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S(x_{(4)})$ | 4 | 8 | 7 | 1 | 9 | 3 | 2 | E | 0 | B | 6 | F | A | 5 | D | C |

### B. Key Schedule

The key schedule depicted in Fig. 6 produces the 30 subkeys $RK^0$ to $RK^{29}$ from the master key $K$ and is designed to allow on-the-fly computations. It uses an 80-bit Linear Finite State Machine (LFSM). Its inner state is denoted $Y$ and is initialized with the master key $K$. The subkeys are extracted from the LFSM state $Y$ using `ExtractRoundKey`, a layer of parallel S-boxes. Subkey $RK^0$ is extracted from the LFSM initial state (*i.e.* the master key $K$). The LFSM state $Y$ is then updated using `RoundFnLFSM` and the next subkey $RK^1$ is extracted. And so on until $RK^{29}$.

The state $Y$ is made of 20 nibbles $Y_{19},\cdots,Y_0$ split among 4 LFSRs, $\mathcal{L}_0$ to $\mathcal{L}_3$, acting on 5 nibbles each: $\mathcal{L}_0$ acts on $Y_0$ to $Y_4$, $\mathcal{L}_1$ on $Y_5$ to $Y_9$ and so on.

We use here LFSRs inspired by the results of [35] and [17] on LFSRs. As described in [35], our 4 LFSRs are nibble oriented and seen as a Feistel network. More precisely, using the matrix representation introduced in [35], it is possible to construct word-oriented LFSRs with word feedbacks well chosen to fit with a Feistel representation generalizing the classical Fibonacci and Galois representations. Moreover, the operations used to compute the feedbacks are simple word-oriented operations such as shifts or rotations. The two main advantages of those constructions are a clear speed-up of diffusion and -due to the Feistel construction- a simple reverse. The 4 LFSRs used in the LILLIPUT key schedule are Feistel-like word-oriented LFSRs acting on 5 nibbles. They are completely described in the following (see also Fig. 7 in Appendix A for a pictural view). Thus, the `RoundFnLFSM` function seen as 4 Feistel-like word-oriented LFSRs can be divided into two transformations: `MixingLFSM` which holds the feedbacks, followed by `PermutationLFSM` which is the word-wise cyclic shift, as depicted in Fig. 6. The precise description of both transformations is given below.

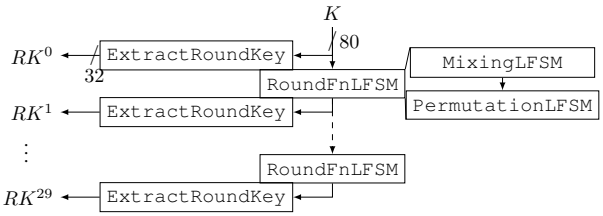


Fig. 6. LILLIPUT Key Schedule

`MixingLFSM`: For each of the 4 parallel LFSRs $\mathcal{L}_0$ to $\mathcal{L}_3$ (see also Appendix A), it consists in xoring some nibbles to some others nibbles in a Feistel-like manner:

- for $\mathcal{L}_0$: $Y_0 \leftarrow Y_0 \oplus (Y_4 \ggg 1)$ and $Y_1 \leftarrow Y_1 \oplus (Y_2 \gg 3)$,
- for $\mathcal{L}_1$: $Y_6 \leftarrow Y_6 \oplus (Y_7 \ll 3)$ and $Y_9 \leftarrow Y_9 \oplus (Y_8 \lll 1)$,
- for $\mathcal{L}_2$: $Y_{11} \leftarrow Y_{11} \oplus (Y_{12} \ggg 1)$ and $Y_{13} \leftarrow Y_{13} \oplus (Y_{12} \gg 3)$,
- for $\mathcal{L}_3$: $Y_{16} \leftarrow Y_{16} \oplus (Y_{15} \ll 3) \oplus (Y_{17} \lll 1)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



`PermutationLFSM`: For each of the 4 parallel LFSRs $\mathcal{L}_0$ to $\mathcal{L}_3$, it consists in a left cyclic shift of its 5 nibbles, i.e. $Y_i \leftarrow Y_{i-1 \bmod 5}$, as depicted in Appendix A.

`ExtractRoundKey`: For each round of encryption, the subkey $RK^i$ is extracted from the LFSM inner state using a non linear extracting function in a bitsliced way. First, some nibbles of the state are extracted:

$Z_{(32)} \leftarrow Y_{18}||Y_{16}||Y_{13}||Y_{10}||Y_9||Y_6||Y_3||Y_1$ and let $Z_{31}, \cdots, Z_0$ be the bits of $Z$ then $RK^i_j = S(Z_j||Z_{8+j}||Z_{16+j}||Z_{24+j})$ where $S$ is the same S-box as in the datapath. Finally, the current round number $i \in \{0, \cdots, 29\}$ is xored to the last 5 bits of the subkey: $RK^i \leftarrow RK^i \oplus i_{(5)}||0_{(27)}$.

### C. Decryption Process

As LILLIPUT is a Feistel network, decryption is quite analogous to encryption but uses the inverse block permutation $\pi^{-1}$ given in Table III and the subkeys in the reverse order. Note that `NonLinearLayer` and `LinearLayer` commute, hence can be computed in any order. The whole decryption process is depicted in Fig. 8 in Appendix B. Besides, as the LFSM of the key schedule is made of Feistel-like LFSRs, the same argument leads to the ability to compute the different subkeys in the reverse order starting from the LFSM inner state $Y^{29}$, as depicted in Fig. 9 in Appendix C.

## VI. Design Rationale

We justify here our design choices in a diffusion purpose perspective for the encryption process and the key schedule.

### A. Structure

When designing LILLIPUT, our overall goal was to maximize diffusion between blocks while keeping reasonable implementation performances. We decide to use an EGFN to reach this purpose because for the EGFN used in LILLIPUT the full diffusion delay can be reduced to $d = 4$ which is the best diffusion delay obtained for a Feistel-like cipher.

We chose a 64-bit state as it is consistent with our goal: to limit the memory footprint. We split that state into 16 nibbles rather than 8 bytes so that the block size matches the S-box size, *i.e.* the non linear layer is made only of 8 parallel calls to a 4-bit S-box and 8 subkey additions. As said before, the $\pi$ permutation has been chosen to maximize the number of active S-boxes on 18, 19 and 20 rounds (see Section IV-C2a for more details).

### B. S-box

LILLIPUT uses a single S-box as its single non linear component. Hence its choice is of crucial importance for the LILLIPUT security and efficiency. For hardware efficiency, we chose a 4-bit S-box rather than an 8-bit one, as they are much cheaper to implement. With a future hardware optimized serial implementation in mind, we have decided to use the same S-box 8 times, rather than having 8 different S-boxes. It will help to implement only one S-box and reuse it intensively.

Various classifications of 4-bit Sboxes [36], [37], [38] exist in the literature. For LILLIPUT, we wanted the S-box to satisfy the following security criteria: the maximum differential probability and the maximum (absolute) linear bias are $2^{-2}$, the algebraic degree is 3, and it has no fixed-point. We select four such S-boxes with simple algebraic expressions (i.e. a minimal number of non-linear transformations), each one corresponding with one of the four classes proposed in [38]. To choose the S-box that will be finally implemented, we have first implemented these 4 S-boxes separately with the goal to choose those with the smallest gate count. It appeared that all the 4 S-boxes have around the same area (around 23 GEs[1]). So, we have implemented LILLIPUT with each S-box to see which one combines the best with the rest of the design. It finally appears that, due to the compiler internal optimizations, the S-box depicted in Table V induces the smallest gate count for LILLIPUT, and so this is the one we have definitely chosen.

### C. Key Schedule

As done in other lightweight block ciphers, such as PRESENT [39], TWINE [16], LBlock [40] or SIMON [41], an 80-bit register initially loaded with the master key is sequentially updated and the subkeys are extracted from that register. However, we chose to split the key into 4 smaller LFSRs that are updated in parallel because small LFSRs mix their content faster and increase performance. The downside is that each LFSR could be attacked independently if their contents were not combined back during the subkey extraction which is not the case here. Yet this design is not entirely new as a similar situation happens in the DES [2] key schedule.

We use LFSRs inspired by the results of [35] and [17] on LFSRs. In [35], the authors generalize LFSR beyond Fibonacci/Galois representation by allowing any cell to be used as feedback in any other cell. They call these new LFSRs "Ring-LFSRs" because of the rotation occuring at each update.

As the LFSRs in [17], the LFSRs chosen here have also a word-oriented structure: instead of performing bit-wise shift at each iteration and having binary feedbacks, they are shifted by one word at each update. As for the feedbacks, they are also word-oriented: one whole word is xored to another after possibly being transformed by a software-friendly operation such as shift or rotation. Those LFSRs are called Word-LFSRs by their authors [17]. When a LFSR is both a Word and Ring LFSR, we call it a Word-Ring-LFSR. Word-Ring LFSRs have thus a smaller diffusion delay than classical Fibonacci or Galois LFSRs. Moreover, and to simplify the reverse process, those Word-Ring-LFSRs can be represented in a Feistel-like manner (see in Fig. 7 in Appendix A for practical examples).

We therefore investigated all possible Word-Ring-LFSRs acting on 5 nibbles with the following conditions: the feedbacks are made in a Feistel-like manner, a feedback nibble can be rotated or shifted before being xored, the connection polynomial is primitive of degree 20 (to produce $m$-sequences) and the number of XOR gates used is minimal. This way, the LFSRs will be well-suited for both hardware and software implementations and can be easily reversed. It turned out that, up to block reindexing equivalence, there are exactly 16 such LFSRs, each with a cost of 5 XOR gates. On those 16 LFSRs,

[1]To see the meaning of GE metric, please refer to Section VIII.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



we observed the following properties/symmetries: half of them are the mapping-composition inverse of the other half; half of them can be deduced from the other half by inverting the role of left and right inside the nibbles, *e.g.* $\ll i$ becomes $\gg i$. Finally, the 16 LFSRs we observed can be deduced from just two of them. So, we took two LFSRs from the first class and two LFSRs from the second class. This choice ensures that different parts of the key are handled differently by different LFSRs, *i.e.* to break symmetries in the key schedule.

To summarize, a Feistel-like LFSRs structure has been chosen to perform on-the-fly computations for both encryption and decryption: the subkeys can be computed in the reverse order up to the equivalent decryption key (see Fig. 9 in Appendix C).

For the `ExtractRoundKey` operation, the extraction points have been chosen such that three consecutive subkeys deplete the whole key space entropy and to allow a bitslice software implementation.

## VII. Security Analysis

We analyze the security of LILLIPUT regarding classical attacks in the unknown key model and also in the related and chosen key models. First, using the instantiated bounds and the results already provided in Section IV-C2, we present the best attacks we can mount regarding impossible differential attack, integral attack and differential/linear cryptanalysis and compare them with the results obtained for TWINE [16] due to the similarities existing between both designs: indeed the EGFN we use can be seen as a TWINE-like GFN with an additional linear layer. In the same way, we give the best results we obtain for related key attacks and chosen key attacks.

### A. Classical Attacks

*1) Impossible Differential Attack:* Based on the 8-round impossible differential distinguisher described in Section IV-C2c, we are able to attack 14 rounds of LILLIPUT adding 3 rounds at the beginning and 3 rounds at the end. To do so, we need to guess 72 bits of the subkeys (40 at the beginning and 32 at the end), based upon plaintexts of the form $(\delta_1, 0, \delta_2, 0, 0, 0, 0, 0, \delta_3, \delta_4, \delta_5, 0, \delta_6, \delta_7, \delta_8, \delta_9)$ and ciphertexts of the form $(\gamma_1, 0, 0, \gamma_2, 0, 0, 0, 0, \gamma_3, \gamma_4, \gamma_5, \gamma_6, \gamma_7, \gamma_8, \gamma_9, 0)$ where $\delta_i$ and $\gamma_i$ are differences. In this case, the complexity of this attack is about $2^{77}$ encryptions using $2^{70}$ pairs of plaintexts/ciphertexts that can be generated using $2^{63}$ well chosen plaintexts.

*2) Integral Attack:* Based on the 15-order integral property on 9 rounds described in Section IV-C2b, we can turn this distinguisher into a 13-round attack against LILLIPUT adding 4 rounds at the end. In this case, we need to guess 76 subkey bits. First, we cipher a structure of $2^{60}$ elements with 15 active nibbles (*i.e.* 15 nibbles with the $P$ property) and obtain the corresponding ciphertexts. From those texts, we decipher the 4 added rounds and test if the corresponding values have a sum equal to 0 using the partial sum method [42] and its improvement in the case of a Feistel structure [29]. The cost for testing if the property occurs by deciphering 4 rounds and by testing 76 subkey bits is about $2^{77.8}$ S-box evaluations corresponding to $2^{71}$ 14-round encryptions. If yes, maybe the good key has been found. However, the false alarm probability is equal to $1 - 1/2^4$. So we need to repeat the process on several structures, say 4, to decrease the false alarm probability. Thus, the complexity of the attack becomes $2^{73}$ encryptions using $2^{62}$ plaintexts/ciphertexts.

*3) Differential / Linear Cryptanalysis:* Based on the analysis presented in Section IV-C2a, we provide here the best truncated differential and linear masks we found for 16 rounds of LILLIPUT with 31 active S-boxes leading to over-approximate the differential/linear characteristic probability by $2^{-62}$. The best truncated differential path is given by an input of the form $(\alpha_0, 0, \alpha_0, \alpha_0, \alpha_0, \alpha_0, \alpha_0, \alpha_1, \alpha_0, 0, 0, 0, 0, 0, 0, 0)$ that gives after 16 rounds an output of the form $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \beta, 0)$ where $\alpha_0$, $\alpha_1$, and $\beta$ are non-zero 4-bit differences such that $\alpha_0 \rightarrow \alpha_1$ is a valid differential for the S-box. In the same way, the best truncated linear input mask is given by $(\gamma_0, \gamma_0, 0, \gamma_0, \gamma_0, \gamma_0, \gamma_0, \gamma_1, \gamma_0, 0, 0, 0, 0, 0, 0, 0)$ that gives after 16 rounds an output truncated linear mask of the form $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \delta, 0)$ where $\gamma_0$,$\gamma_1$, and $\delta$ are any non-zero 4-bit linear masks such that $\gamma_0 \rightarrow \gamma_1$ is a valid linear transition of the S-box.

Moreover, compared to the impossible differential attack, we naturally expect that the number of rounds that can be added at the beginning and at the end for the key guessing step is lower because the number of non-zero variables at the input is larger in both cases. So, we except to be able to mount differential/linear attacks on at most 21 rounds.

### B. Related Key and Chosen Key Attacks

The related key attacks introduced by E. Biham in [43] in 1993 allow an attacker to know some relations between different keys without knowing the keys themselves and to encrypt under those keys some plaintexts. From those pairs of plaintexts/ciphertexts, the aim of the attacker is to recover the key. In the related key settings, we first evaluate the related key pairs that activate the lowest number of S-boxes in the key schedule. Due to the bit oriented nature of our key schedule, we have only evaluated this number for at most 6 bits with differences. We obtain that the best related key path is obtained for two master keys having differences in bits 28 and 38 leading to activate 32 S-boxes in all the subkeys (this value does not take into account the S-boxes crossed during the encryption process itself). Moreover, trying to launch a related key impossible differential attack seems to lead to an attack on 17 rounds of LILLIPUT considering two master keys with a single bit difference at position 43.

Chosen key attacks have been introduced by Biryukov *et al.* in [44]. Usually, to test if there exists such differential paths in byte or nibble oriented ciphers, the algorithm to implement is the one described in [45]. However, in our case, and due to the bit oriented structure of our key schedule, such an algorithm has an infeasible complexity. So, we limit our study to the case of one up to five active bits injected through the master key difference and thus the key schedule. We thus obtain a differential path with 40 active S-boxes on 23 rounds.

Finally, to prevent slide attacks [46] and as usually done in other block cipher proposals, different round constants are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



added to each subkey during the key schedule process. So, we consider LILLIPUT immune to slide attacks. In the same way, Meet-In-The-Middle (MITM) attacks are impracticable against LILLIPUT because the key schedule has been designed such that every 3 consecutive subkeys use all bits of the master key.

### C. Security Evaluation Summary

Due to the better diffusion of our scheme compared to TWINE (in our case, $d = 4$ whereas for TWINE, $d = 8$), structural attacks such as impossible differential attack or integral attack reach fewer rounds than for the TWINE case. However, as the improved diffusion delay comes from the additional linear layer composed of XORs, our results concerning classical differential and linear cryptanalysis are worse than the TWINE ones even if in our case, we only obtain an upper bound. Moreover, as TWINE has a nibble oriented key-schedule whereas our own key schedule is bit oriented, we cannot compare the results obtained in the related key and chosen key models.

Finally, as our best attack against LILLIPUT reaches, in the differential/linear context, at most 21 rounds, we conjecture that there is no attack against 22 rounds of LILLIPUT in the single key settings more efficient than the exhaustive key search leaving 8 rounds of security margin. As our best attack in the related/chosen models reaches 23 rounds, we also conjecture that there is no attack against 25 rounds of LILLIPUT in the related, known and chosen key settings more efficient than the exhaustive key search leaving 5 rounds of security margin.

## VIII. IMPLEMENTATION ASPECTS

As stated at the beginning of Sect. V, our goal is to check if LILLIPUT can fit in harsh constraints environments where 3-pass mutual authentication between the reader and the tag is required. In this scenario, we have to implement a challenge-response protocol where *both* encryption and decryption are mandatory like some protocols described in ISO 9798-2 [34]. Such protocols are implemented nowadays in industrial products like *e.g.* NXP's DESFire family.

Usually, in lightweight cryptography, one of the most dominant metric to optimize is chip area. This metric can be expressed in $\mu m^2$ but this value is dependent of the standard cells technology. To ease comparisons between implementations, the circuit area is measured in Gate Equivalences (GEs). A GE is the area of a 2-input NAND gate in the used standard cell technology. So the area of the circuit expressed in GEs is the surface of the circuit in $\mu m^2$ divided by the surface of a NAND gate. Thanks to this GE metric, we can more easily compare the area cost of two proposals implemented in different technologies. In addition to the area circuit, we have also to take care of the power consumption of our LILLIPUT implementation and its latency.

### A. Advantages of the LILLIPUT Structure

*1) SPN vs. Feistel-like networks:* Usually, if a designer wants to create an *ad hoc* algorithm (*i.e.* from scratch), he has two possibilities: implement a Feistel or an SPN network.

Implementing Feistel networks brings some advantages compared to SPNs regarding circuit area. First, the round function $f$ is identical for the encryption and the decryption processes. So, since we want to implement a protocol where both encryption and decryption are mandatory, the same circuit can be used for both procedures. Compared to an SPN, it is a real gain because the encryption and decryption functions that have to be implemented are different, and so no hardware resource can be shared between encryption and decryption. Another comparative advantage is that only one half of the cipher state is processed at each round, so hardware resources for the other half are saved.

Usually, two commonly believed drawbacks reduce these advantages. The first is that Feistel networks need some XORs to recombine the modified half part of the cipher state with the other. In our case, it only consists of $21 \times 4$ XORs, which are equivalent to only $21 \times 4 \times 2.25 = 189$ GEs in our used ASIC technology. Another argument is that Feistel networks need more rounds than an SPN for a comparable diffusion effect, which induces throughput, power consumption and energy overheads. This is not the case for LILLIPUT because of its better diffusion: we only have to compute 30 rounds, which is comparable to some other lightweight SPNs like PRESENT (31 rounds) or other GFNs like *Piccolo* (25 rounds for a 80-bit key). We have only to take care of the power consumption/energy overhead brought by the added 189 GEs; We can guess that the contribution of these 189 GEs on the power consumption is negligible compared to the global circuit.

*2) Basic Components of* LILLIPUT*:* To implement the confusion effect, LILLIPUT uses 4×4 bit S-boxes which are by far smaller than 8×8 S-boxes and 6×4 ones: the LILLIPUT S-box takes around 23 GEs to implement, compared to some dozens for 6×4 ones, and some hundreds for 8×8 ones. Moreover, to limit the number of rounds, we have chosen an S-box which gives a good trade off between security and hardware cost. The diffusion effect is only computed thanks to 189 GEs and a bit permutation which is very efficient to implement in hardware.

The memory elements (flip-flops) used to store the subkeys and the cipher state are the costliest hardware elements to implement, and are those which consume the most energy. Just to store the 64-bit cipher and 80-bit key state, we need $(80 + 64) \times 5.75 = 828$ GEs. Typically, this storage is responsible for at least 50% of the total power consumption/energy and the area of the circuit.

For the key schedule, we have chosen to make it simple, but not as simple as KTANTAN or PRINTCIPHER to protect it against related key and slide attacks. The same S-box is used for the datapath part and the key schedule part so a resource sharing is possible between the two parts. Unfortunately, doing so, an overhead is brought by adding 2-to-1 multiplexors (which cost 2 GEs per bit in our ASIC technology). Moreover, it will prevent from generating subkeys in parallel of cipher state processing, so the number of needed round cycles will increase. So, we decided in our benchmark implementation to implement two sets of S-boxes, one for the cipher state and the other for the subkeys generation. The differences between encryption and decryption are minor, only `PermutationLayer` and `PermutationLFSM` need to be reversed, so the overhead of implementing decryption over encryption is negligible.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



### B. Round-Wise Implementation of LILLIPUT

We give hereafter the implementation results of a straightforward round-wise implementation of LILLIPUT. It processes 64 bits of plaintext with an 80-bit key on 30 clock cycles. Subkeys are computed on-the-fly, in parallel of the cipher state processing. There is no resource sharing between the cipher state and the key schedule processes. The S-boxes are implemented in a Look-Up-Table way, so we let the compiler do its own optimizations. Only the encryption process is implemented.

We implemented LILLIPUT in VHDL and synthesized it using a low-power High Vt 65 nm standard-cell library. We used Synopsis Design Vision D-2010.03-SP5-2 for synthesis and power simulation. The foundry typical values (of 1.2 V for the core voltage and 25° for the temperature) were used. In this straightforward implementation, non-scan flip-flops are used. We applied priority optimizations on area.

Our straightforward round-wise low-power LILLIPUT occupies 1832 GEs and has a simulated power of 0.9 $\mu$W. This practical result can be compared with theory. Theoretically, our LILLIPUT implementation needs 828 GEs to store both the subkeys and the cipher state, $16 \times 23 = 368$ GEs for the S-boxes, $((21 + 8) \times 4 + 20 + 5) \times 2.25 = 317.25$ GEs for all the XORs, and $(80 + 64) \times 2 = 288$ for all the 2-to-1 multiplexors (which selects between the encryption key (resp. the plaintext) or the subkey (the cipher state)). So, we can estimate (neglecting the cost of the finite state machine) that our round-wise implementation of LILLIPUT needs at least $828 + 368 + 317.25 + 288 = 1801.25$ GEs in total [2].

### C. Round-Wise Implementations and Comparisons (Encryption Only)

For doing a fair comparison, we need to implement the same kind of hardware optimizations than LILLIPUT competitors.

A first possible optimization concerns scan flip-flops (instead of standard ones) to save GEs. In fact, if we consider and implement separately one 2-to-1 multiplexor followed by one flip-flop, we have a hardware cost in our ASIC library of $2 + 5.75$ GEs $= 7.75$ GEs. However, if we implement one scan flip-flop (without "enable" signal) instead, which contains intrinsically one 2-to-1 multiplexor, then it costs only 6 GEs, which allows to save 1.75 GEs per 1-bit storage.

A second possible optimization has been proposed by the designers of Piccolo [47]: they infer AND-NOR gates to optimize XOR/XNOR gate count, which allows to save 0.25 GE per XOR.

In the following, we give implementation results of all the LILLIPUT competitors taken into account these optimizations[3].

A comparison with other ciphers which only implements encryption follows in Table VI. We have only listed in this table block ciphers with 80-bit key and 64-bit plaintext with hardware implementations compliant with HF RFID tags constraints, a reasonable throughput and no known attack on the key schedule. The throughputs are given with a 100kHz clock frequency.

TABLE VI
COMPARISON WITH OTHER ROUND-WISE IMPLEMENTATIONS OF LIGHTWEIGHT BLOCK-CIPHERS (ENCRYPTION ONLY).

| | Key Size | Block Size | Lat. (cycles) | Throughput (kbit/s) | Area (GEs) | Logic Process |
|---|---|---|---|---|---|---|
| **PRESENT-80** | 80 | 64 | 31 | 200 | 1553 | 0.18$\mu$m |
| **TWINE** | 80 | 64 | 36 | 178 | 1464 | 90nm |
| ***Piccolo*-80** | 80 | 64 | 25 | 237 | 1496 | 0.13$\mu$m |
| **LILLIPUT** | 80 | 64 | 30 | 213 | 1545 | LP 65nm |

Looking at the throughput, due to its smaller number of rounds, LILLIPUT is faster than PRESENT-80 [39] and TWINE [16]. It is only a little bit slower than *Piccolo*-80 [47].

Concerning the power consumption, comparisons are very difficult first because we use low-power ASIC standard-cell libraries and the others do not, second because the CMOS technologies are not the same in the papers and third because in most cases no average nor peak power consumption result is given by the authors. We can only deduce that the power consumption of our implementation of LILLIPUT is smaller than PRESENT-80 (0.9 $\mu$W *vs.* 5). This result is mainly due to the advanced CMOS library.

The following discussions will be only focused on area comparisons. In short, the differences between LILLIPUT and the other ciphers come first from the added cost of decryption on top of encryption, and second our desire to build a strong key schedule.

The only competitor of LILLIPUT which is bigger for straightforward, round-wise encryption-only implementations is PRESENT. However, LILLIPUT is more competitive when both encryption and decryption modes are necessary.

### D. Round-Wise Implementations and Comparisons (Encryption + Decryption)

A comparison with other ciphers which implements both encryption and decryption modes follows in Table VII.

TABLE VII
COMPARISON WITH OTHER ROUND-WISE IMPLEMENTATIONS OF LIGHTWEIGHT BLOCK-CIPHERS (ENCRYPTION + DECRYPTION).

| | Key Size | Block Size | Lat. (cycles) | Throughput (kbit/s) | Area (GEs) | Logic Process |
|---|---|---|---|---|---|---|
| **PRESENT-80** | 80 | 64 | 31 | 200 | 2018 | 0.18$\mu$m |
| **TWINE** | 80 | 64 | 36 | 178 | 1799 | 90nm |
| ***Piccolo*-80** | 80 | 64 | 25 | 237 | 1634 | 0.13$\mu$m |
| **LILLIPUT** | 80 | 64 | 30 | 213 | 1581 | LP 65nm |

LILLIPUT is then smaller than all its competitors where both encryption and decryption are required in a mutual authentication scenario. Since LILLIPUT has an involutive structure, the addition of decryption mode over an encryption one is almost free: only the inverses of `PermutationLayer` and `PermutationLFSM` must be implemented in addition (only cross wiring, so very little overhead) with scan flip-flops which integrates 2-to-1 multiplexors. These latter will only induce an overhead of 0.25 GE per stored bit, so $(64 + 80) \times 0.25 = 36$ GEs in total.

### E. Serial Implementations and Comparisons (Encryption + Decryption)

To decrease power consumption and area requirements, implementations can also be serialized. Contrary to round-wise implementations, serial implementations compute only a

[2]We know that the interface can have a huge impact on the size and the performance of our implementation, but it must be noticed that all the state-of-the-art lightweight block ciphers we cite in this article do not include the area and latency overheads brought by the interface.

[3]We have decided to exclude LBlock from our comparisons since its hardware advantage is due to a simpler architecture of its key schedule that has some undesirable security properties [48].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



fraction of one round in a clock cycle. This can allow lowering the number of implemented arithmetic instances (*e.g.*, SBoxes) at the cost of additional control logic (multiplexers).

We have implemented a serialized version of LILLIPUT which works as follows. It consists of 2 8×4-bit shift registers: one containing the nibbles $x_{15} \parallel x_{14} \parallel ... \parallel x_8$ (right shift) and one containing $x_7 \parallel x_6 \parallel ... \parallel x_0$ (left shift). In a first step, the `NonLinearLayer` and `LinearLayer` are computed at the same time to get the new values of $x_8$, $x_9$, ..., $x_{14}$ before the `PermutationLayer` computation (this latter is computed in one clock cycle afterwards). In fact, it is computed at each clock cycle, for $i$ from 9 to 14: $x_i \leftarrow F_{i-8}(x_{15-i}) \oplus x_i \oplus x_7$ and $x_8 \leftarrow F_0(x_7) \oplus x_8$. An additional 4-bit register is then needed to duplicate the storage of $x_7$ value. In a second step, it is needed to compute $x_{15}$ value. It is first initialized from the first step $x_{15} \leftarrow F_7(x_0) \oplus x_{15} \oplus x_7$. Then, the shift register containing $x_7 \parallel x_6 \parallel ... \parallel x_0$ initiates a new rotation phase to accumulate at each clock cycle, for $i$ from 1 to 6: $x_{15} \leftarrow x_{15} \oplus x_i$.

In summary, only 2 8×4-bit shift registers for the storage of $x_i$ nibbles, one 4-bit register for $x_7$ storage, one instance of the S-box and 2 4-bit XORs have to be implemented in our serialized design. One round is then computed in 17 clock cycles and so an encryption takes $30 \times 17 = 510$ clock cycles.

A comparison with *Piccolo-80* (which is the only competitor of LILLIPUT that provides results for both encryption and decryption modes in a serialized way) follows in Table VIII.

TABLE VIII
COMPARISON WITH OTHER SERIALIZED IMPLEMENTATIONS OF LIGHTWEIGHT BLOCK-CIPHERS (ENCRYPTION + DECRYPTION).

| | Key Size | Block Size | Lat. (cycles) | Throughput (kbit/s) | Area (GEs) | Logic Process |
|---|---|---|---|---|---|---|
| ***Piccolo*-80** | 80 | 64 | 432 | 14.8 | 1103 | 0.13$\mu$m |
| **LILLIPUT** | 80 | 64 | 510 | 12.5 | 1055 | LP 65nm |

The throughput of *Piccolo-80* is better than LILLIPUT, but since the decryption can be implemented almost for free, LILLIPUT is smaller when implemented in a serialized way.

## IX. CONCLUSION

In this article, we have introduced a generic matrix representation that captures most existing Generalized Feistel Networks. We explained diffusion properties of those schemes through this representation. We then introduced a new kind of schemes called Extended Generalized Feistel Networks that adds a diffusion layer to the classical GFNs. We finally instantiated this class of schemes into a new lightweight block cipher proposal called LILLIPUT and analyzed its security in generic and instantiated models.

Concerning practical implementations of the lightweight design, we have shown that LILLIPUT compares well to other lightweight ciphers when implemented in a round-based fashion and in a serialized way on a low-power 65 nm standard-cell library. Thanks to its involutive nature, the overhead of implementing both encryption and decryption on the same circuit is small. LILLIPUT is adapted to the harsh constraints of RFID systems.

Concerning future works related to hardware implementations, we will first evaluate the impact of using methods which reduce glitches in the circuit (*i.e.* clock-gating). Concerning software implementations, we want also to benchmark the performances of LILLIPUT on 4-bit, 8-bit and 16-bit low-power microcontrollers used in smart cards and wireless sensor networks.

We finally encourage all the cryptographic community to perform independent security analysis on LILLIPUT. Test vectors are given in Appendix D.

## ACKNOWLEDGEMENTS

The authors would like to thank Guillaume Reymond (CEA-LETI, Gardanne, France) for providing his ASIC implementation results and Cédric Marchand (Université Jean-Monnet, Saint-Étienne, France) for our fruitful discussions.

## REFERENCES

[1] T. Berger, M. Minier, and G. Thomas, "Extended Generalized Feistel Networks using Matrix Representation," in *Selected Areas in Cryptography - SAC 2013*, ser. LNCS, vol. 8282. Springer, 2013, pp. 289–305.

[2] *Data Encryption Standard*, National Bureau of Standards, U. S. Department of Commerce, 1977.

[3] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis," in *Selected Areas in Cryptography - SAC 2000*, ser. LNCS, vol. 2012. Springer, 2000, pp. 39–56.

[4] C. Adams and J. Gilchrist, "The CAST-256 Encryption Algorithm," Network Working Group, RFC 2612, june 1999, http://tools.ietf.org/html/rfc2612.

[5] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee, "HIGHT: A New Block Cipher Suitable for Low-Resource Device," in *Cryptographic Hardware and Embedded Systems - CHES 2006*, ser. LNCS, vol. 4249. Springer, 2006, pp. 46–59.

[6] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, "The 128-Bit Blockcipher CLEFIA (Extended Abstract)," in *Fast Software Encryption - FSE 2007*, ser. LNCS, vol. 4593. Springer, 2007, pp. 181–195.

[7] K. Nyberg, "Generalized Feistel Networks," in *Advances in Cryptology - ASIACRYPT '96*, ser. LNCS, vol. 1163. Springer, 1996, pp. 91–104.

[8] SHS, "Secure Hash Standard," in *FIPS PUB 180-4, Federal Information Processing Standards Publication*, 2012.

[9] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. M. Jr, L. O'Connor, M. Peyravian, D. Stafford, and N. Zunic, "MARS - an AES candidate," *NIST AES Proposal*, 1999.

[10] Y. Zheng, T. Matsumoto, and H. Imai, "On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses," in *Advances in Cryptology - CRYPTO '89*, ser. LNCS, vol. 435. Springer, 1989, pp. 461–480.

[11] S. Moriai and S. Vaudenay, "On the Pseudorandomness of Top-Level Schemes of Block Ciphers," in *Advances in Cryptology - ASIACRYPT 2000*, ser. LNCS, vol. 1976. Springer, 2000, pp. 289–302.

[12] V. T. Hoang and P. Rogaway, "On Generalized Feistel Networks," in *Advances in Cryptology - CRYPTO 2010*, ser. LNCS, vol. 6223. Springer, 2010, pp. 613–630.

[13] M. Naor and O. Reingold, "On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited," *J. Cryptology*, vol. 12, no. 1, pp. 29–66, 1999.

[14] T. Suzaki and K. Minematsu, "Improving the Generalized Feistel," in *Fast Software Encryption - FSE 2010*, ser. LNCS, vol. 6147. Springer, 2010, pp. 19–39.

[15] S. Yanagihara and T. Iwata, "Improving the Permutation Layer of Type 1, Type 3, Source-Heavy, and Target-Heavy Generalized Feistel Structures," *IEICE Trans.*, vol. 96-A, no. 1, pp. 2–14, 2013.

[16] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, "TWINE : A Lightweight Block Cipher for Multiple Platforms," in *Selected Areas in Cryptography - SAC 2012*, ser. LNCS, vol. 7707. Springer, 2012, pp. 339–354.

[17] F. Arnault, T. P. Berger, M. Minier, and B. Pousse, "Revisiting LFSRs for Cryptographic Applications," *IEEE Trans. on Info. Theory*, vol. 57, no. 12, pp. 8095–8113, 2011.

[18] J. Kim, S. Hong, and J. Lim, "Impossible differential cryptanalysis using matrix method," *Discrete Mathematics*, vol. 310, no. 5, pp. 988–1002, 2010.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers




[19] L. Zhang and W. Wu, "Differential analysis of the Extended Generalized Feistel Networks," *Inf. Process. Lett.*, vol. 114, no. 12, pp. 723–727, 2014.

[20] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM J. Comput.*, vol. 17, no. 2, pp. 373–386, 1988.

[21] H. Gilbert and M. Minier, "New Results on the Pseudorandomness of Some Blockcipher Constructions," in *Fast Software Encryption - FSE 2001*, ser. LNCS, vol. 2355. Springer, 2001, pp. 248–266.

[22] U. M. Maurer, "Indistinguishability of Random Systems," in *Advances in Cryptology - EUROCRYPT 2002*, ser. LNCS, vol. 2332. Springer, 2002, pp. 110–132.

[23] A. Mitsuda and T. Iwata, "Tweakable Pseudorandom Permutation from Generalized Feistel Structure," in *Provable Security, Second International Conference - ProvSec 2008*, ser. LNCS, vol. 5324. Springer, 2008, pp. 22–37.

[24] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," in *Advances in Cryptology - CRYPTO '90*, ser. LNCS, vol. 537. Springer, 1990, pp. 2–21.

[25] M. Matsui, "Linear Cryptoanalysis Method for DES Cipher," in *Advances in Cryptology - EUROCRYPT '93*, ser. LNCS, vol. 765. Springer, 1993, pp. 386–397.

[26] F. Chabaud and S. Vaudenay, "Links Between Differential and Linear Cryptoanalysis," in *Advances in Cryptology - EUROCRYPT '94*, ser. LNCS, vol. 950. Springer, 1994, pp. 356–365.

[27] FIPS 197, "Advanced Encryption Standard," Federal Information Processing Standards Publication 197, 2001, u.S. Department of Commerce/N.I.S.T.

[28] L. R. Knudsen and D. Wagner, "Integral Cryptanalysis," in *Fast Software Encryption - FSE 2002*, ser. LNCS, vol. 2365. Springer, 2002, pp. 112–127.

[29] Y. Sasaki and L. Wang, "Meet-in-the-middle technique for integral attacks against feistel ciphers," in *Selected Areas in Cryptography - SAC 2012*, ser. LNCS, vol. 7707. Springer, 2012, pp. 234–251.

[30] A. Biryukov and A. Shamir, "Structural Cryptanalysis of SASAS," in *Advances in Cryptology - EUROCRYPT '01*, ser. LNCS, vol. 2045. Springer, 2001, pp. 394–405.

[31] E. Biham, A. Biryukov, and A. Shamir, "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials," in *Advances in Cryptology - EUROCRYPT '99*, ser. LNCS, vol. 1592. Springer, 1999, pp. 12–23.

[32] Y. Luo, Z. Wu, X. Lai, and G. Gong, "A Unified Method for Finding Impossible Differentials of Block Cipher Structures," *IACR Cryptology ePrint Archive*, vol. 2009, p. 627, 2009.

[33] J. Kim, S. Hong, J. Sung, C. Lee, and S. Lee, "Impossible Differential Cryptanalysis for Block Cipher Structures," in *Progress in Cryptology - INDOCRYPT 2003*, ser. LNCS, vol. 2904. Springer, 2003, pp. 82–96.

[34] I. 9798-2, *Information technology – Security techniques – Entity authentication – Part 2: Mechanisms using symmetric encipherment algorithms*. ISO/IEC, 2008.

[35] T. P. Berger, M. Minier, and B. Pousse, "Software Oriented Stream Ciphers Based upon FCSRs in Diversified Mode," in *Progress in Cryptology - INDOCRYPT 2009*, ser. LNCS, vol. 5922. Springer, 2009, pp. 119–135.

[36] C. D. Cannière, "Analysis and Design of Symmetric Encryption Algorithms," Ph.D. dissertation, Katholieke Universiteit Leuven, 2007.

[37] G. Leander and A. Poschmann, "On the Classification of 4 Bit S-Boxes," in *WAIFI*, ser. LNCS, vol. 4547. Springer, 2007, pp. 159–176.

[38] M.-J. O. Saarinen, "Cryptographic Analysis of All 4 x 4 - Bit S-Boxes," Cryptology ePrint Archive, Report 2011/218.

[39] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, ser. LNCS, vol. 4727. Springer, 2007, pp. 450–466.

[40] W. Wu and L. Zhang, "LBlock: A Lightweight Block Cipher," in *Applied Cryptography and Network Security - ACNS 2011*, ser. LNCS, vol. 6715, 2011, pp. 327–344.

[41] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK Families of Lightweight Block Ciphers," Cryptology ePrint Archive, Report 2013/404, 2013.

[42] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved Cryptanalysis of Rijndael," in *Fast Software Encryption - FSE 2000*, ser. LNCS, vol. 1978. Springer, 2001, pp. 213–230.

[43] E. Biham, "New Types of Cryptoanalytic Attacks Using related Keys (Extended Abstract)," in *Advances in Cryptology - EUROCRYPT '93*, ser. Lecture Notes in Computer Science, vol. 765. Springer, 1993, pp. 398–409.

[44] A. Biryukov, D. Khovratovich, and I. Nikolic, "Distinguisher and Related-Key Attack on the Full AES-256," in *Advances in Cryptology - CRYPTO 2009*, ser. LNCS, vol. 5677. Springer, 2009, pp. 231–249.

[45] A. Biryukov and I. Nikolic, "Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others," in *Advances in Cryptology - EUROCRYPT 2010*, ser. LNCS, vol. 6110. Springer, 2010, pp. 322–344.

[46] A. Biryukov and D. Wagner, "Slide Attacks," in *Fast Software Encryption - FSE '99*, ser. LNCS, vol. 1636. Springer, 1999, pp. 245–259.

[47] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: An Ultra-Lightweight Blockcipher," in *Cryptographic Hardware and Embedded Systems - CHES 2011*, ser. LNCS, vol. 6917. Springer, 2011, pp. 342–357.

[48] C. Boura, M. Naya-Plasencia, and V. Suder, "Scrutinizing and Improving Impossible Differential Attacks: Applications to CLEFIA, Camellia, LBlock and Simon," in *Advances in Cryptology - ASIACRYPT 2014*, ser. LNCS, vol. 8873. Springer, 2014, pp. 179–199.


**Thierry P. Berger** received the Ph.D. degree and the French Habilitation (Mathematics) from the University of Limoges, France.

From 1992, he has been with the University of Limoges. He is currently Professor in the Department of Mathematics and Computer and the scientific head of the Cryptology and Information Security group. His research interests include finite algebra, automorphism group of codes, links between coding and cryptography, stream ciphers, pseudorandom generators and dedicated block ciphers.

**Julien Francq** is born in 1982 in Lille (France). After preliminary studies in Physics, Chemistry, Microelectronics and Control Systems, and a PhD thesis in Computer Science domain obtained in 2009 in Montpellier University (France), he is now working in Airbus Defence & Space - CyberSecurity company as a cryptography and key management expert. His main research interest is the security and the efficiency of (hardware/software) implementations of cryptography against (mathematical/physical) attacks.

**Marine Minier** received the Ph.D. degree in 2002, from the University of Limoges and the French Habilitation (Computer Sciences) from the University of Lyon, France in 2012. In 2005, she joined the INSA de Lyon (Institut National des Sciences Appliquées), as an Assistant Professor, in the CITI laboratory, a team working in telecommunications and the INRIA Team Privatics in 2012. Her research interests include Symetric Key Cryptography, Security in Sensor Networks and Privacy.

**Gaël Thomas** defended his Ph.D. degree in June 2015, at the University of Limoges. His research interests include Symetric Key Cryptography and pseudo-random generators.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2468218, IEEE Transactions on Computers



## APPENDIX A
## UPDATING THE FOUR LFSRS USED IN THE KEY-SCHEDULE

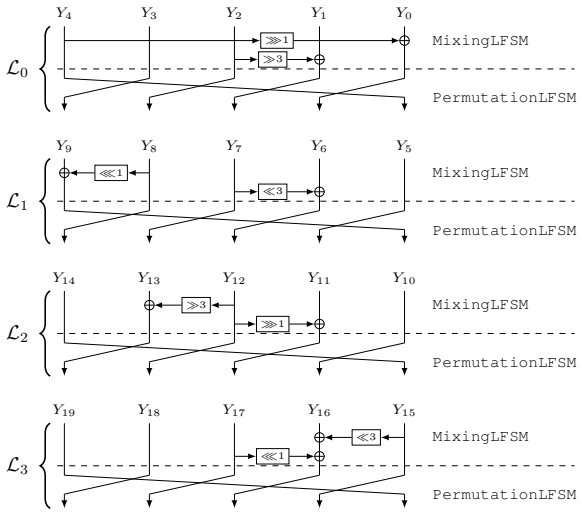


Fig. 7. LFSRs $\mathcal{L}_0$ to $\mathcal{L}_3$ used in the key-schedule.

## APPENDIX B
## THE LILLIPUT DECRYPTION PROCESS

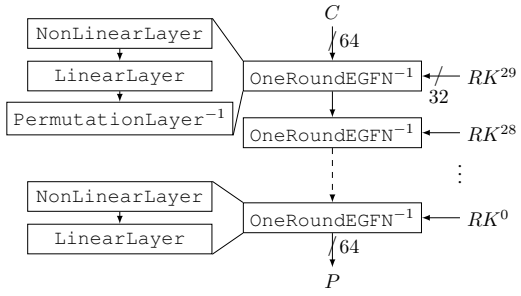


Fig. 8. LILLIPUT Decryption

## APPENDIX C
## THE DECRYPTION KEY-SCHEDULE PROCESS

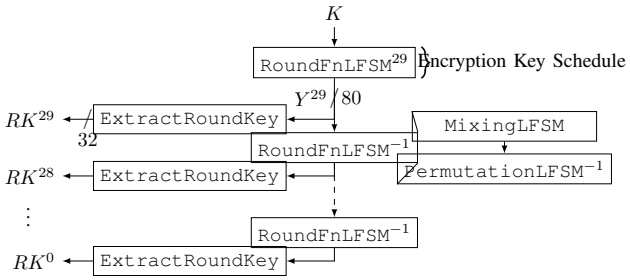


Fig. 9. LILLIPUT Decryption Key Schedule

## APPENDIX D
## TEST VESTORS

We provide the following test vectors given in little endian and in hexadeximal for LILLIPUT:

```
input_message = 0x0000000000000000
KEY = 0x00000000000000000000
ciphertext = 0x5041b83331b27668

input_message = 0x0123456789abcdef
KEY = 0x0123456789abcdef0123
ciphertext = 0x9d60ea93c2c5a914
```