Improving the Security and Efficiency of Block Ciphers based on LS-Designs

Anthony Journault, François-Xavier Standaert, Kerem Varici

Improving the Security and Efficiency of Block Ciphers based on LS-Designs

- Extended Abstract -

Anthony Journault, François-Xavier Standaert, Kerem Varici.

ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium.

Abstract. LS-designs are a family of bitslice ciphers aiming at efficient masked implementations against side-channel analysis. This paper discusses their security against invariant subspace attacks, and describes an alternative family of eXtended LS-designs (XLS-designs), that enables additional options to prevent such attacks. LS- and XLS-designs provide a large family of ciphers from which efficient implementations can be obtained, possibly enhanced with countermeasures against physical attacks. We argue that they are interesting primitives in order to discuss the general question of "how simple can block ciphers be?".

1 Introduction

LS-designs are a family of block ciphers proposed at FSE 2014, aimed for efficient bitslice implementations [10]. They essentially combine linear diffusion L-boxes with non-linear bitslice S-boxes. The instances proposed so far (namely the involutive cipher Robin and the non-involutive cipher Fantomas) have additionally been selected to minimize the total number of AND gates, in order to allow efficient masked implementations against side-channel attacks [4], which is also beneficial to multiparty computation and fully homomorphic encryption [1]. In a more recent work by Leander et al., it has been shown that the involutive instance Robin was susceptible to an invariant subspace attack, leading to a weak keys set of density 2^{-32} for this cipher [14]. This raised questions regarding the origin of the attack and the possibility to prevent it for involutive LS-designs.

In this paper, we complement these works with two main contributions.

First, we analyze the invariant subspace attack against Robin and show that one can prevent it, e.g. with a better choice of round constants. Of particular interest in this respect is the fact that these constants should have all their bits varying (in bitslice representation), in order to avoid invariant subspaces for the S-boxes or L-boxes to be trivially propagated through the rounds.

Second we question the possibility to improve the efficiency of LS-designs with a better choice (and different sizes) of components. In particular, Robin and Fantomas are based on 8-bit S-boxes and 16-bit L-boxes. While very convenient from an implementation point-of-view, the selection of these components was partially heuristic (since, e.g. an exhaustive analysis of 8-bit S-boxes is computationally out of reach). As a result, we investigate an alternative approach in two steps. First, we design 32-bit "Super S-Boxes" based on optimal components

(i.e. 4-bit S-box and 32-bit L-box based on a MDS code). Second, we combine these Super S-boxes with an additional ShiftColumns operation. Both the use of Super S-boxes and their combination with a ShiftColumns operation are naturally reminiscent from an AES-like cipher [7, 9], but with a bitslice rather than block-oriented structure. Interestingly, we show that the resulting eXtended LS-designs (XLS-designs) can also be implemented very efficiently, e.g. based only on table lookups and word-oriented operations, yet leading to slighly more complex tradeoffs than LS-designs, due to their slightly more involved structure. For concreteness and further investigations, we additionally specify an instance of such XLS-design, denoted as Mysterion, with 128-bit or 256-bit block size.

2 Fixing the invariant subspace attacks against Robin

2.1 LS-designs, Robin and Fantomas

LS-designs are a family of block ciphers that are composed of a combination of lookup table-based L-boxes and bitslice S-boxes. The definition of s-bit S-boxes and l-bit L-boxes directly gives rise to an instance of $n = s \cdot l$-bit cipher. One advantage of LS-designs is their inherent simplicity, as illustrated with the short specifications given in Algorithm 1. The cipher takes n-bit plaintext and key blocks as input, and follows Substitution Permutation Network (SPN) approach. Namely, the inputs and state are represented as $s \cdot l$ arrays of bits, with s the number of rows and l is the number of columns. In each round, the S-box operation acts on the columns, and the L-box operation acts on the rows. These two components combined with constant and key addition define the round function of LS-designs, that is iterated N_r times in order to obtain the ciphertext.

Algorithm 1 LS-design with l-bit L-boxes and s-bit S-boxes

$x \leftarrow P \oplus K$; ▷ x is an $s \cdot l$-bit matrix
for $0 \leq r < N_r$ **do**
  for $0 \leq i < l$ **do** ▷ S-box Layer
    $x[i, \star] = \mathsf{S}[x[\star, i]]$;
  for $0 \leq j < s$ **do** ▷ L-box Layer
    $x[\star, j] = \mathsf{L}[x[j, \star]]$;
  $x \leftarrow x \oplus K \oplus C(r)$; ▷ Key addition and round constant
return x

Concretely, both Robin and Fantomas were based on 8-bit S-boxes and 16-bit L-boxes. For the former one, these components are involutive, in order to improve the performances of the cipher when decryption has to be implemented.

2.2 Invariant subspace attacks and results on Robin

The invariant subspace attack was first introduced at CRYPTO 2011 [13] and applied to the lightweight "PRINTcipher" [12]. We can summarize the attack

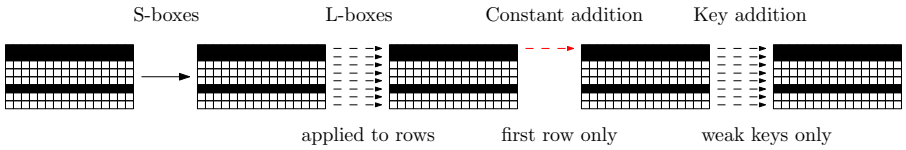

Fig. 1. An example of invariant subspace attack against one round of Robin.

as follows. Let us consider an n-bit iterative block cipher, with round function $\mathsf{R}_k : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$, such that $\mathsf{R}_k(x) = \mathsf{E}(x + k)$, with E an n-bit permutation. If there exists a subspace $\mathcal{S} \subseteq \mathbb{F}_2^n$ and two constants $a, b \in \mathbb{F}_2^n$ such that $\mathsf{E}(\mathcal{S} + a) = \mathcal{S} + b$, then for a round key $k = s + a + b$ with $s \in \mathcal{S}$, the following holds:

$$\mathsf{R}_k(\mathcal{S} + b) = \mathsf{E}((\mathcal{S} + b) + (s + a + b)) = \mathsf{E}(\mathcal{S} + a) = \mathcal{S} + b.$$

That is, the round function maps the affine subspace $\mathcal{S} + a$ onto $\mathcal{S} + b$. Furthermore, if all round keys are in $\mathcal{S} + (a + b)$, then this property is iterative. This is the case for some Even-Mansour block ciphers, where the same master key is used as subkey through the whole cipher, e.g. LED [11], Zorro [8], Noekeon [5], Fantomas and Robin [10], which are therefore natural targets for invariant subspace attacks. The Eurocrypt 2015 paper that exhibits a weak keys set of density 2^{-32} for Robin is based in this property, and essentially takes advantage of the involutive nature of its components together with weak round constants [14].

More precisely, the involutive building blocks of Robin helps finding self-similarities within the cipher – a new type of such self-similarities (for the L-boxes) is actually given in the Eurocrypt paper. Besides, and as mentioned above, LS-designs are such that S-boxes act through columns and the linear layer acts through rows. Hence, if there exists an invariant subspace (e.g.) for the S-box layer and all inputs to the S-boxes are chosen from it, then the linear layer will not change this subspace[1]. That is, if we call the bits which form the invariant subspace active and other bits passive (as in differential cryptanalysis), then the linear layer does not mix active and passive bits. Combined with the fact that the round constants of Robin are sparse, and only apply to one state row, this allowed the propagation of invariant subspaces through the cipher. An illustration of the attack based on an invariant subspace for the S-box layer is given in Figure 1, where black boxes represent bits that form an invariant subspace.

2.3 Fixing the attack

Based on the previous description, a couple of ways to fix the invariant subpace attack could be considered for Robin, e.g. changing its components (linear layer & S-box), applying a key scheduling, or changing the round constants. Among those, changing the round constants is the easiest one, since it implies minimum

[1] As just mentioned, this attack can be applied by finding an invariant subspace for the linear layer as well, in which case the S-box layer will not change the subspace.

changes on the design. Concretely, one suggestion is to use a dense set of round constants, applied to all the rows rather than a single one. For example, a Linear Feedback Shift Register (LFSR) with 16-bit state size (and e.g. primitive polynomial $P(X) = X^{16} + X^5 + X^3 + X^2 + 1$) could be used for this purpose. Eight consecutive states can then be combined together to form each round constant. We verified with the same generic algorithm as described in [14] that this choice was sufficient to remove the invariant subspaces from the Robin rounds (up to the computational limits of the algorithm). We also checked exhaustively that no invariant subspaces can propagate through the rounds of reduced (32-bit) LS-designs using such dense constants. Note that despite no invariant subspace attack has been exhibited against the non-involutive cipher Fantomas, it has a similar structure as Robin, and its round constants are sparse as well. Therefore, tweaking this cipher (e.g. with stronger round constants) could be advisable too. Eventually, we mention that the tweak described in this subsection may not be the most efficient, and is mainly suggested as a proof-of-concept that involutive LS-designs are not inherently susceptible to invariant subspace attacks.

3 The Mysterion instance

The previous section highlighted that invariant subspace attacks against (non-involutive) LS-designs exploit the structural simplicity of these ciphers. While this simplicity is highly beneficial to implementation efficiency, it also leads to the question whether a slightly more involved structure could provide better security margins. In this section, we consequently investigate this option and, motivated by the efficient masking goal of LS-designs, combine it with a further improvement of the balance between linear and non-linear operations within the cipher. The rationale behind this tweaked approach is twofold. First, for the linear part, we observe that from the security point-of-view it would be interesting to take advantage of a (non-binary) MDS code to build the diffusion layer. Second, for the non-linear part, it would be interesting to take advantage of optimal S-boxes, that we only know for smaller bit sizes. For example, Ullrich et al. found an optimal (from the linear and differential cryptanalysis points-of-view) 4-bit S-box requiring only 4 AND gates (next denoted as Class 13) [18].

Based on these observations, we propose new instances of ciphers where we first combine an optimal 4-bit S-box with an MDS diffusion matrix, which results in 32-bit Super S-boxes, and then combine these Super S-boxes with a ShiftColumns operation to obtain 128- and 256-bit ciphers. Admittedly, this approach does not strictly follow the LS-design specifications, since (i) its diffusion layer is not based on binary matrices anymore, and (ii) it requires an additional ShiftColumns operations. So it primarily aims at improving the security margins of LS-designs, e.g. against linear and differential cryptanalysis and invariant subspace attacks (see Section 3.2). Yet, and quite interestingly, we will show in Section 3.3 that the resulting XLS-designs can still be implemented very efficiently, taking advantage of the linearity of the MDS diffusion and ShiftColumns operations. So intuitively, the main price to pay for the latter approach is slightly

more complex specifications (although they can be viewed as a bitslice counterpart to AES-like ciphers and therefore have a concise description), which are interesting to compare with the extreme simplicity of LS-designs, both from the implementation efficiency and the physical security points-of-view.

3.1 Specifications

XLS-designs can be described as the gathering of b LS-designs of $s \cdot l$ bits, where s is the size of the S-box (in bits, as in LS-designs), and l is the size of the underlying MDS matrix of the L-box (and no longer the bit size of the L-box as in LS-designs), resulting in a $n = b \cdot s \cdot l$-bit cipher. Note that this slight change of meaning for the l notation is mandatory to keep notations consistent with LS-designs, since a binary matrix cannot be MDS. Concretely, the internal state of an XLS-design can be written as $X[\star, \star, \star]$, such that $X[i, \star, \star]$ is an $s \cdot l$-bit block (with $1 \leq i \leq b$), $X[i, j, \star]$ is block i's jth l-bit row (with $1 \leq j \leq s$) and $X[i, \star, j]$ is block i's jth s-bit column (with $1 \leq j \leq l$). As illustrated in Figure 2, the S-box layer of XLS-designs is strictly the same as in Algorithm 1. Their L-box layer slightly changes compared to LS-designs, since it is applied to all the rows of each block at once (rather than to row by row in LS-designs). And the main difference is the additional ShiftColumns layer, that can be viewed as the bitslice dual to the ShiftRows operation in the AES Rijndael, and will be defined next. XLS-designs are succintly described in Algorithm 2.

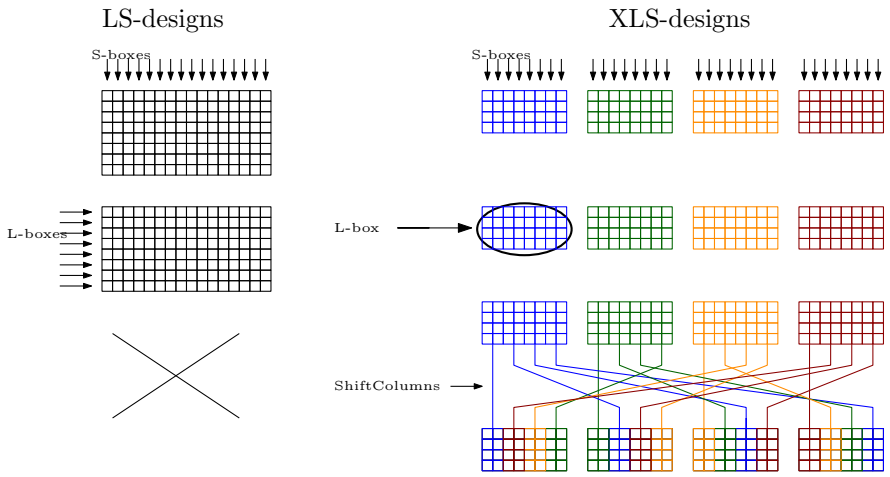

Fig. 2. 128-bit LS-design vs. 128-bit XLS-design.

We now describe the different components that give rise to the Mysterion-128 instance (with a state made of four 32-bit blocks), and the Mysterion-256 instance (with a state made of eight 32-bit blocks), that both exploit 4-bit S-boxes.

Algorithm 2 XLS-design with $l \cdot s$-bit L-boxes, s-bit S-boxes and b blocks

$x \leftarrow P \oplus K$; ▷ x is a $s \cdot (l.b)$ bits matrix
for $0 \leq r < N_r$ **do**
  for $0 \leq j < b$ **do**
    for $0 \leq i < l$ **do**
      $x[j, \star, i] = \mathsf{S}[x[j, \star, i]]$; ▷ S-box layer
  for $0 \leq j < b$ **do**
    $x[j, \star, \star] = \mathsf{L}[x[j, \star, \star]]$; ▷ L-box layer
  for $0 \leq k < s$ **do**
    $x[\star, k, \star] = \mathsf{ShiftColumns}[x[\star, k, \star]]$; ▷ ShiftColumns layer
  $x \leftarrow x \oplus K \oplus C(r)$; ▷ Key and round constant addition
return x

The S-box. Mysterion uses the Class13 S-box [18], that is non-involutive, has a bitslice representation with 4 AND[2] and 4 XOR gates (see Appendix A), algebraic degree 3, differential probability of 2^{-2}, and linear probability of 2^{-1}.

The L-box. Mysterion uses a linear transform derived from the recent paper by Augot and Finiasz [3], in which an algorithm allowing to find recursive MDS diffusion layers using shortened BCH codes is described. A recursive MDS matrix is a matrix that can be expressed as a power of the companion matrix of a polynomial. This algorithm uses the degree of the polynomial k (hence the size of the companion matrices), and the field size $q = 2^s$ as parameters, and provides all the polynomials of degree k over $\mathbb{F}_{2^s}$ such as their companion matrices raised to the power k gives MDS diffusion layers. We ran it with parameters $k = 8$ and $s = 4$ using Magma, in order to obtain an 8×8 MDS matrix over $\mathbb{F}_{2^4}$. The selected degree-8 polynomial with coefficients in $\mathbb{F}_{2^4} \cong \mathbb{F}_2[\alpha]/(\alpha^4 + \alpha + 1)$, is $P(X) = X^8 + \alpha^3 \cdot X^7 + \alpha^4 \cdot X^6 + \alpha^{12} \cdot X^5 + \alpha^8 \cdot X^4 + \alpha^{12} \cdot X^3 + \alpha^4 \cdot X^2 + \alpha^3 \cdot X + 1$. The resulting diffusion layer is coming from an MDS code $[16, 8, 9]_{\mathbb{F}_{2^4}}$ and therefore has both its differential and linear branch number equal to 9.

ShiftColumns. For Mysterion-128, ShiftColumns acts on columns two by two. The two first columns of each block are not moved, the two second columns are moved by one block, the two third columns are moved by two blocks, and the two fourth columns are moved by three blocks. This operation can also be described as a bit permutation of a 32-bit word, with logic operations: $X = (X$ & 0xC0C0C0C0) $\vee$ ROL(X & 0x03030303,8) $\vee$ ROL(X & 0x0C0C0C0C,16) $\vee$ ROL(X & 0x30303030,24), where $\vee$ and & stand for logic OR and AND, and ROL(X,n) stands for the left rotation of X by n bits. For Mysterion-256, ShiftColumns acts on columns one by one. The first columns of each block are not moved, the second columns are moved by one block, ..., and the eighth columns are moved by seven blocks. See Appendix A for the alternative description.

These components directly define our two instances Mysterion-128, with parameters $b = 4$, $s = 4$, $l = 8$, and Mysterion-256, with parameters $b = 8$, $s = 4$, $l = 8$.

[2] More precisely, 3 ANDs and one OR, which can be masked at the same cost.

As for round constants, we suggest to use simpler ones as in the original Robin and Fantomas ciphers. This will be further justified in the next section.

3.2 Security analysis

A complete security analysis of the Mysterion instances is out of the scope of this extended abstract because of place constraints (more details will be given in the full version of the paper). Hence, our main goal in this section is twofold. On the one hand, we want to show that simple 4-round bounds against linear and differential cryptanalyses can be obtained for XLS-designs, inheriting from their AES-like structure. On the other hand, we want to argue why its more complex structure also improves resistance against invariant subspace attacks. Note that as for LS-designs, no related-key security is claimed for Mysterion.

Security against linear and differential cryptanalyses. A straightforward application of the wide-trail strategy [6] leads to the following theorems.

Theorem 1. *Four rounds of* `Mysterion`*-128 activate at least 45 S-boxes.*

Theorem 2. *Four rounds of* `Mysterion`*-256 activate at least 81 S-boxes.*

A sketch of the proofs is in Appendix B. As a result, we directly have the following bounds for the probabilities of linear and differential characteristics for 4 Mysterion-128 rounds (where we use the same definitions as in [10]):

$$\mathrm{Pr}_{lin}^{char}(4\mathrm{R}) \leq \mathrm{Pr}_{lin}^{max}(\text{S-box})^{45} = 2^{-45},\ \mathrm{Pr}_{diff}^{char}(4\mathrm{R}) \leq \mathrm{Pr}_{diff}^{max}(\text{S-box})^{45} = 2^{-90}.$$

And similarly, for the Mysterion-256 instance, we have:

$$\mathrm{Pr}_{lin}^{char}(4\mathrm{R}) \leq \mathrm{Pr}_{lin}^{max}(\text{S-box})^{81} = 2^{-81},\ \mathrm{Pr}_{diff}^{char}(4\mathrm{R}) \leq \mathrm{Pr}_{diff}^{max}(\text{S-box})^{81} = 2^{-162}.$$

For illustration, Table 1 compares the upper bounds for the maximum probabilities of differential characteristics for Robin, Fantomas and Mysterion. Setting the number of rounds to 12 for Mysterion-128 and 16 for Mysterion-256 leads to very comfortable security margins, and better bounds than for Robin and Fantomas. The table for linear characteristics leads to similar recommendations.

Number of rounds	8	12	16
Prob. diff char. for Robin	2^{-128}	2^{-192}	$\mathbf{2^{-256}}$
Prob. diff char. for Fantomas	2^{-160}	$\mathbf{2^{-256}}$	2^{-344}
Prob. diff char. for Mysterion-128	2^{-180}	$\mathbf{2^{-270}}$	2^{-360}
Prob. diff char. for Mysterion-256	2^{-324}	2^{-486}	$\mathbf{2^{-648}}$

Table 1. Max. prob. of differential characteristics for LS- and XLS-design instances.

Security against invariant subspace attacks. As briefly discussed in Section 2.2, invariant subspace attacks can be of two kinds. A (simpler) one taking advantage of invariant subspaces in the S-box and (a more intricate) one using equality spaces in the L-box (that is highly structured in the case of Robin). The first one is easy to bypass with a good choice of S-box, e.g. the Class13 S-box has no invariant subspaces. The second one is more difficult to analyze. So far, the Eurocrypt 2015 results only describe a heuristic tool allowing to look for such invariant subspaces. Hence, running this tool (with the available computational resources) on full cipher instances, and exhaustively on reduced cipher instances, is the best that we can currently do. For example, Leander et al. could not spot invariant subspaces against Fantomas using this appraoch. In the case of Mysterion, we first note that the use of a 32-bit L-box is not sufficient to prevent the existence of invariant subspaces within the rounds (as revealed by an exhaustive analysis performed on a 32-bit block). However, the addition of a ShiftColumns operation will break the propagation of any subspace found for the L-box with high probability. This was confirmed by a computationally-bounded analysis performed on Mysterion-128. We therefore conclude that XLS-designs can withstand invariant subspace attacks even with sparse round constants (as usually used in block cipher designs, in order to limit their memory requirements).

3.3 Performances

One of the goals of LS-designs (hence, by extesion, XLS-designs) is to allow efficient masked implementations. In this respect, a natural problem is to find out whether the slightly more complex structure of XLS-designs, using (non-binary) MDS matrices and an additional ShiftColumns transform, leads to a loss of efficiency. In this section, we briefly discuss this issue and detail how efficient table lookup-based implementations of Mysterion-128 can be obtained.

In general, the implementation of a 32-bit Super S-box can be directly implemented with logic operations (which is more time consuming), or with table lookups as in the case of the AES Rijndael (which is faster, but requires 16 tables of 256 bytes, rather than 4 such tables for a 128-bit LS-design). Next, the ShiftColumns operation mixes bits of different blocks, which can exploit the logic representation given in Section 3.1, or be implemented with table lookups. This leads to interesting tradeoffs from the physical security point-of-view. On the one hand, the logic representation of ShiftColumns requires less memory than its table-based execution, and acts at the row level. On the other hand, performing ANDs with constants including some bits set to zero can be viewed as a bit manipulation that may lead to harder to prevent leakages (as argued in [10]).

For illustration, we implemented Mysterion-128 on a 8-bit microcontroller (Atmel AtMega644p), based on a mixed approach, namely table lookups for the L-boxes and logic operations for ShiftColumns. We wrote our reference code in *C* and used the *avr-libc* library with headers *#include <avr/pgmspace.h>* and *#include <avr/io.h>*. We also used the *PROGMEM* attribute to save RAM. Results were obtained with the *avr-gcc* compiler and optimization option *-O2*.

We simulated the execution time of the implementations using the *Atmel AVR Studio 6* software. Performances are reported for an unrolled version of the code.

Figure 3 summarizes our results in terms of number of cycles for Mysterion-128, together with natural competitors, i.e. Robin and Fantomas [10], Zorro [8], Noekeon [5], PICARO [16] and the AES [17]. Security order 0 means unprotected implementation *i.e.* one share or no mask, security order 1 means two shares or one mask, and so on. Excepted for unprotected implementions (for which Mysterion-128 is slightly less efficient than its competitors), the main conclusion of these evaluations is that such an XLS-design has excellent performances. More precisely, the reduced amount of non-linear operations in Mysterion-128 allows its implementations to compare favorably with its competitors already for first-order security. As previously mentioned, the price to pay for these excellent performances are potentially more leaky operations, which can be avoided using table lookups, but would then lead to larger memory requirements.

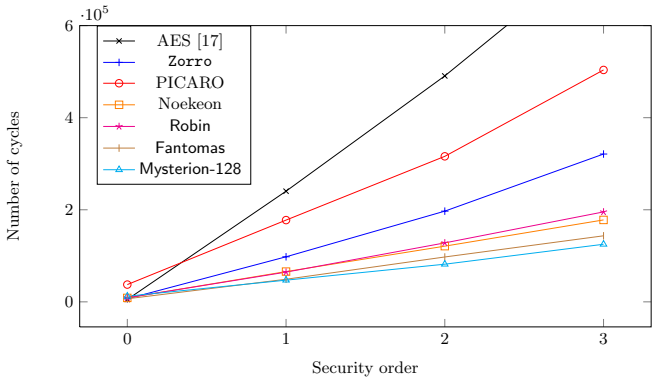

Fig. 3. Encryption times for different 128-bit block ciphers in an Atmel AtMega644p.

4 Conclusions

This work enlarges the block cipher design space from LS-designs to XLS-designs. We believe this is an interesting step forward, since it is in line with the general question of "how simple can block ciphers be?", in a context – i.e. considering the risk of side-channel analysis – where simplicity is usually correlated with security. Indeed, simple and very structured ciphers are generally easier to protect against physical attacks. In this respect, our first contribution is to show that LS-designs are not inherently susceptible to invariant subspace attacks, but that their instantiation should carefully consider them. And our second contribution is to show that XLS-designs can indeed be implemented efficiently (and lead to better security bounds against linear and differential cryptanalysis), but that their best implementation requires informed decisions (e.g. on whether the use of bit manipulations can be critical). These questions lead to many open

problems regarding the best cipher instances for different block/key sizes. For example, instances with 3-bit S-boxes could be considered to minimize the AND depth as in [1]; instances with 5-bit S-boxes could lead to even reduced round requirements (for linear and differential attacks); even for the 4-bit instances, it could be interesting to investigate the use of L-boxes based circulant matrices such as advertized in [2], which would allow alternative implementations to prevent cache-based timing attacks (although SSSE3 instructions can also be used for this), ... And should we use LS- or XLS-designs for any of those instances? Whether a key scheduling has to be included and how, especially for cipher instances claiming some related key security (contrary to this work), but also to prevent invariant subspace attacks, is another interesting question.

Because of place constraints, this extended abstract focused on the main concepts that make LS- and XLS-designs intersting families of ciphers. More detailed specifications and security analysis will be provided in the full paper.

Acknowledgements. F.-X. Standaert is a research associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in parts by the European Commission through the ERC project 280141 (CRASH).

References

1. Martin Albrecht, Christian Rechberger, Thomas Schneider, and Tyge Tiessen. Ciphers for MPC and FHE. *To appear in the proceedings of EUROCRYPT 2015.*
2. Martin R. Albrecht, Benedikt Driessen, Elif Bilge Kavun, Gregor Leander, Christof Paar, and Tolga Yalçin. Block ciphers - focus on the linear layer (feat. PRIDE). In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2014.
3. Daniel Augot and Matthieu Finiasz. Direct construction of recursive mds diffusion layers using shortened bch codes. Cryptology ePrint Archive, Report 2014/566, 2014. `http://eprint.iacr.org/`.
4. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
5. Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. The block cipher Noekeon (NESSIE project submission). *http://gro.noekeon.org/.*
6. Joan Daemen and Vincent Rijmen. The wide trail design strategy. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001.
7. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard.* Information Security and Cryptography. Springer, 2002.
8. Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert. Block ciphers that are easier to mask: How far can we go? In Guido Bertoni

and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2013.

9. Henri Gilbert and Thomas Peyrin. Super-sbox cryptanalysis: Improved attacks for AES-like permutations. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers*, volume 6147 of *Lecture Notes in Computer Science*, pages 365–383. Springer, 2010.
10. Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-designs: Bitslice encryption for efficient masked software implementations. *To appear in the proceedings of FSE 2014.*
11. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.
12. Lars R. Knudsen, Gregor Leander, Axel Poschmann, and Matthew J. B. Robshaw. Printcipher: A block cipher for ic-printing. In Mangard and Standaert [15], pages 16–32.
13. Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A cryptanalysis of PRINTcipher: The invariant subspace attack. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 206–221. Springer, 2011.
14. Gregor Leander, Brice Minaud, and Rønjom. A generic approach to invariant subspace attacks: Cryptanalysis of Robin, iSCREAM and Zorro. *To appear in the proceedings of EUROCRYPT 2015.*
15. Stefan Mangard and François-Xavier Standaert, editors. *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*. Springer, 2010.
16. Gilles Piret, Thomas Roche, and Claude Carlet. PICARO - A block cipher allowing efficient higher-order side-channel resistance - extended version -. *IACR Cryptology ePrint Archive*, 2012:358, 2012.
17. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Mangard and Standaert [15], pages 413–427.
18. Markus Ullrich, Christophe De Canniere, Sebastiaan Indesteege, Özgül Küçük, Nicky Mouha, and Bart Preneel. Finding optimal bitsliced implementations of 4x4-bit s-boxes. In *SKEW 2011 Symmetric Key Encryption Workshop, Copenhagen, Denmark*, pages 16–17, 2011.

A Specifications of Mysterion's components

A.1 Mysterion S-box

Algorithm 3 Class13 S-box, bitslice representation

Require: 4 input bits (A, B, C, D)
Ensure: 4 output bits such as $(a, b, c, d) = \mathsf{S}(A, B, C, D)$
1: $a = A\&B$;
2: $a = a \oplus C$;
3: $c = B \mid C$;
4: $c = c \oplus D$;
5: $d = a\&D$;
6: $d = d \oplus A$;
7: $b = c\&A$;
8: $b = b \oplus B$;
9: **return** (a, b, c, d)

A.2 Mysterion L-box

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & \alpha^3 & \alpha^4 & \alpha^{12} & \alpha^8 & \alpha^{12} & \alpha^4 & \alpha^3 \end{pmatrix}, C^8 = \begin{pmatrix} 1 & \alpha^3 & \alpha^4 & \alpha^{12} & \alpha^8 & \alpha^{12} & \alpha^4 & \alpha^3 \\ \alpha^3 & \alpha^{13} & \alpha^4 & \alpha & 1 & \alpha^2 & \alpha^2 & \alpha^{12} \\ \alpha^{12} & \alpha^{14} & \alpha^{12} & \alpha^{14} & \alpha^2 & \alpha^7 & \alpha^5 & \alpha^8 \\ \alpha^8 & 1 & \alpha^5 & \alpha^{14} & \alpha^7 & \alpha & \alpha^2 & \alpha^3 \\ \alpha^3 & \alpha^{14} & \alpha^9 & \alpha^{10} & \alpha^{10} & \alpha^9 & \alpha^{14} & \alpha^3 \\ \alpha^3 & \alpha^2 & \alpha & \alpha^7 & \alpha^{14} & \alpha^5 & 1 & \alpha^8 \\ \alpha^8 & \alpha^5 & \alpha^7 & \alpha^2 & \alpha^{14} & \alpha^{12} & \alpha^{14} & \alpha^{12} \\ \alpha^{12} & \alpha^2 & \alpha^2 & 1 & \alpha & \alpha^4 & \alpha^{13} & \alpha^3 \end{pmatrix}.$$

C is the companion matrix of the polynomial defined in Section 3.1.
C^8 is the underlying matrix of the Mysterion L-box.

A.3 ShiftColumns of Mysterion-256

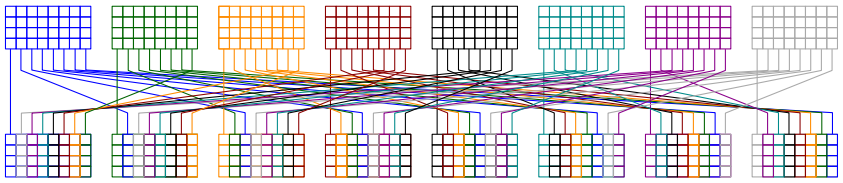

Fig. 4. ShiftColumns of Mysterion-256.

B Proofs for the bound of the number of active S-boxes

Proof of Theorem 2. The internal state of Mysterion-256 can be seen as a square, since the number of blocks is equal to the number of columns in a block in this case. Therefore, the proof directly results from the Four-Round Propagation Theorem of the AES Rijndael given in [6]. That is, the number of active S-boxes over four rounds of Mysterion-256 is lower bounded by the square of the branch number of the L-box, which corresponds to $9^2 = 81$ active S-boxes.

Proof of Theorem 1. Contrary to Mysterion-256, we cannot directly use the Four-Round Propagation Theorem of the AES to lower bound the number of active S-boxes in Mysterion-128, since its number of columns is larger than its number of blocks (*i.e.* the state is no longer a square). However, similar bounds can be deduced from a modified version of the theorem proven in [6]. We show in the following how Mysterion-128 can fulfill the hypotheses of this theorem with a simple rearrangement of its operations. For this purpose, we first need to set some definitions and notations. First, a bundle is a 4-bit word and corresponds to a column in the representation of the internal state of Mysterion-128. We denote by $\mathcal{L}$ the application of the L-box on each block of the state, which are divided into four independent parts of eight bundles each. We call this partition of the bundles Ξ. Mysterion-128 is a key-alternating block cipher and iteratively applies the same round function, composed of an S-box layer, an L-box layer, a ShiftColumns layer, and key and round constant additions. As the latter do not influence the number of active S-boxes, we will omit them in the following. Based on this, four rounds of Mysterion-128 can then be written as:

$$\mathsf{ShiftColumns} \circ \mathcal{L} \circ \mathsf{S} \circ \mathsf{ShiftColumns} \circ \mathcal{L} \circ \mathsf{S} \circ \mathsf{ShiftColumns} \circ \mathcal{L} \circ \mathsf{S} \circ \mathsf{ShiftColumns} \circ \mathcal{L} \circ \mathsf{S}.$$

We next reorganise these operations in order to highlight a particular structure of the linear transformation for 4 rounds, which allows a simpler analysis. More precisely, since ShiftColumns commutes with S, we have the following equivalent definition of four rounds of Mysterion-128:

$$\mathsf{ShiftColumns} \circ \mathcal{L} \circ \mathsf{ShiftColumns} \circ \mathsf{S} \circ \mathcal{L} \circ \mathsf{S} \circ \mathsf{ShiftColumns} \circ \mathcal{L} \circ \mathsf{ShiftColumns} \circ \mathsf{S} \circ \mathcal{L} \circ \mathsf{S}.$$

Thanks to this representation, we easily identify two different transformations $\tau^a = \mathcal{L} \circ S$ and $\tau^b = \mathbf{L} \circ S$, where $\mathbf{L} = \mathsf{ShiftColumns} \circ \mathcal{L} \circ \mathsf{ShiftColumns}$. Then four rounds of $\mathsf{Mysterion} - 128$ are the alternation of τ^a and τ^b:

$$\tau^a \circ \tau^b \circ \tau^a \circ \tau^b.$$

Figure 5 summarizes our notations and modified representation of Mysterion-128.

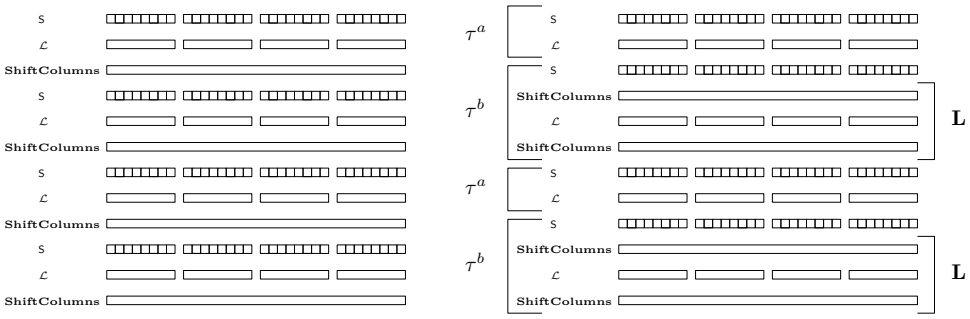

Fig. 5. Two equivalent representations of four rounds of Mysterion − 128.

We finally exploit the following theorem from [6]:

Theorem 3. *For a key alternating block cipher with round tranformations τ^a and τ^b, the number of active S-boxes of any trail over*

$$\tau^b \circ \tau^a \circ \tau^b \circ \tau^a$$

is lower bounded by $\mathcal{B}(\mathcal{L}) \times \mathcal{B}(\mathbf{L}, \Xi)$, where $\mathcal{B}(\mathcal{L})$ is the branch number of the linear transformation $\mathcal{L}$ and $\mathcal{B}(\mathbf{L}, \Xi)$ is the branch number of the linear transformation $\mathbf{L}$ with respect to the partition of the bundles Ξ.

The branch number of $\mathcal{L}$ is 9 (the L-box of Mysterion is an MDS code $[16, 8, 9]_{\mathbb{F}_{2^4}}$). The partition Ξ divide the state into the 4 blocks. We say a block is active when it has at least one active (*.i.e* non zero) bundle. As the ShiftColumns operation spreads two bundles of each block into other blocks, and as $\mathcal{L}$ is MDS, we have that the minimum number of input/output active blocks, therefore the branch number of $\mathbf{L}$ with respect to the partition Ξ, is 5. As a result, the number of active S-boxes over four rounds is lower bounded by $9 \times 5 = 45$.