A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro*

Gregor Leander[1]**, Brice Minaud[2], Sondre Rønjom[3]

[1] Horst Görtz University for IT Security, Ruhr-Universität Bochum, Germany
[2] Agence Nationale de la Sécurité des Systèmes d'Information, France
[3] Nasjonal sikkerhetsmyndighet, Norway
gregor.leander@rub.de, brice.minaud@gmail.com, sondrer@gmail.com

Abstract. Invariant subspace attacks were introduced at CRYPTO 2011 to cryptanalyze PRINTcipher. The invariant subspaces for PRINTcipher were discovered in an ad hoc fashion, leaving a generic technique to discover invariant subspaces in other ciphers as an open problem. Here, based on a rather simple observation, we introduce a generic algorithm to detect invariant subspaces. We apply this algorithm to the CAESAR candidate iSCREAM, the closely related LS-design Robin, as well as the lightweight cipher Zorro. For all three candidates invariant subspaces were detected, and result in practical breaks of the ciphers. A closer analysis of independent interest reveals that these invariant subspaces are underpinned by a new type of self-similarity property. For all ciphers, our strongest attack shows the existence of a weak key set of density 2^{-32}. These weak keys lead to a simple property on the plaintexts going through the whole encryption process with probability one. All our attacks have been practically verified on reference implementations of the ciphers.

Key words: Cryptanalysis, Lightweight Cryptography, Invariant Subspace, Self-Similarity, iSCREAM, LS-Designs, Zorro, CAESAR

Introduction

Block ciphers are one of the most essential cryptographic primitives. Our understanding of how to build secure block ciphers has greatly advanced in the last 20 years. Nowadays, analyzing a given block cipher with respect to a large class of non-trivial attacks, including linear and differential attacks and their variants, is a well-understood process for a large class of block ciphers. However, when it comes to designing block ciphers with strong performance requirements, often less conservative approaches are chosen. Examples of such performance requirements that have recently been studied extensively include low hardware footprint (e.g. PRESENT [6], LED [20], KATAN [10]), low memory consumption on small embedded processors (e.g. ITUBee [21], SPECK [5], PRIDE [2]), low latency (e.g. PRINCE [7]) and ease of side-channel protection (e.g. Zorro [14], LS-Designs [15]).

In order to fit within constrained settings, many of these ciphers feature innovative designs: they may rely on simpler round functions, or minimal key schedules. While in most cases, guarantees against traditional linear or differential attacks are still offered, the simpler structure of many of these ciphers may lend itself to new attacks. Careful cryptanalysis is required in order to assess the security of these new designs; in this process, new techniques have emerged.

One such technique is the invariant subspace attack, introduced in [22] for the cryptanalysis of PRINTcipher. The general idea behind this attack is the following: assume that the round function of a cipher maps a coset A of some vector subspace of the inner state to a coset B of the same space, and a fixed key belonging to $A - B$ is added in every round. Then the set A is preserved by the round function, and hence remains stable through the whole encryption process. This property holds for a large set of keys in PRINTcipher, breaking the cipher in a practical

** The work of Gregor Leander was funded by the BMBF UNIKOPS project.

setting. This type of attack seems particularly well-suited to substitution-permutation networks (SPN) with a minimal key schedule or cryptographic permutations with highly structured round constants.

Invariant subspace attacks are unusual in that they rely on an unexpected form of symmetry in the round function, and yield attacks that are independent of the number of rounds. On the other hand, the same attributes are shared by attacks based on self-similarity properties. These properties were first formally defined in [4] to study alternative descriptions of AES, and later used in [8] to cryptanalyze the SHA-3 candidate *Lesamnta* and the lightweight cipher XTEA.

Interestingly, despite its fundamental nature, the understanding of symmetries and invariant subspaces, in block ciphers or cryptographic permutations, is rather limited. The invariant subspace attack on PRINTCipher was found in an ad hoc fashion and no general approach to detect or avoid such invariant spaces is known. This is even more surprising as for more involved attacks like differential and linear attacks and their variations our general understanding of detection and avoiding those attacks by design is much more evolved.

Our contribution In this paper we aim at increasing the general understanding of invariant subspaces. For this purpose, and as our first main result, we present a generic algorithm that is able to detect invariant subspaces. The running time of this algorithm depends on the block size of the primitive and the density of the existing invariant subspaces. In particular, it is especially efficient if relatively large invariant subspaces exist. As the impact of an invariant subspace increases with its dimension, this can be seen as detecting stronger attacks significantly faster than minor attacks.

We apply this generic algorithm to the lightweight cipher Robin introduced at FSE 2014 as a concrete instance of the LS-design framework [15], the closely related CAESAR [1] candidate iSCREAM [18], as well the lightweight cipher Zorro presented at CHES 2013 [14]. In all cases the algorithm is able to detect invariant subspaces.[4] Attacks resulting from these invariant subspaces break all three ciphers in a practical setting. All attacks have been verified on reference implementations of the ciphers.

As our second main contribution, we show that the invariant subspaces we have discovered are underpinned by a type of self-similarity property of independent interest, stemming from a linear map commuting with the round function. Surprisingly, such a map exists for all three ciphers, despite Robin and Zorro having quite different structures. As a result, we obtain stronger attacks on our target ciphers. We also hope to provide useful insight for the design and analysis of ciphers with minimal key schedules as well as for the choice of round constants in cryptographic permutations.

More specifically, our attacks show the existence of weak keys in Robin, Zorro, as well as iSCREAM in the chosen-tweak scenario. In all cases, the proportion of weak keys is 2^{-32} within the set of all keys. Encryption of a single chosen plaintext is enough to determine whether a key is weak, making our weak key setting very practical. Once a key is recognized as weak, a simple property on plaintexts goes through the whole encryption process with probability one, breaking plaintext confidentiality. In addition, the full 128-bit encryption key can be recovered using one chosen plaintext in time complexity 2^{64}. We also show that all three ciphers are instantly broken in the related-key setting, without any weak key requirement; although only iSCREAM claims security in this model.

In the case of LS-designs, we furthermore present a second attack, based on S-box-dependent invariant subspaces, without an underlying self-similarity. We obtain a new set of weak keys with the same properties as above, including the fact that they can be detected using a single chosen plaintext. However when applying this second attack to Robin and iSCREAM, we obtain a much rarer set of weak keys, with only one key in 2^{80} being weak. We then fine-tune this attack against iSCREAM, and obtain a ratio of weak keys of 2^{-48}, at the cost of requiring 2^{32} chosen-tweak chosen plaintexts in order to detect whether a key is weak; once a weak key is detected, the full 128-bit key can be recovered in time complexity 2^{48} with no additionnal data.

[4] The source code of our tool is available at `http://invariant-space.gforge.inria.fr`.

Regarding LS-designs, it should be pointed out that while our first attack breaks Robin and iSCREAM in a practical setting, Fantomas and SCREAM appear to be safe. Moreover, in the case of Robin and iSCREAM, a careful tweak of the ciphers should be able to prevent our attacks. Thus, the security of the LS-design framework in general is not called into question. On the contrary, in the case of Zorro, our attack adds to other attacks suggesting that partial nonlinear layers should be approached with caution.

Related work Invariant subspace attacks were introduced in [22]. Their application to PRINTCIPHER relies on undesirable properties induced by its 3-bit S-boxes. By contrast, most of our attacks (except the second attack on Robin and iSCREAM) are actually independent of a particular choice of S-boxes.

A thorough analysis of invariant subspaces in PRINTCIPHER was subsequently carried out in [9]. Using a dedicated tool, the authors were able to enumerate all invariant subspaces of PRINTCIPHER, of the type uncovered in the previous article. However their approach is tailored to PRINTCIPHER, and does not extend to other ciphers.

An older line of work has studied "linear factors" of DES [27,11,13], which bear some resemblance to invariant subspaces. The existence of a linear factor is an even stronger property than that of an invariant subspace: essentially, it asks that a (linearly defined) portion of the ciphertext only depend on (linearly defined) portions of the plaintext and key. Nonetheless, it is interesting to note that our attacks do uncover a linear factor in Robin and Zorro (cf. Appendix B), although only in a weak key setting.

Along a similar line, the attack on SAFER in [24] should be mentioned. It exploits the action of the cipher on cosets of a vector space as a whole, rather than isolating a specific trail or characteristic.

Self-similarity properties were used to attack hash functions and block ciphers in [8]. It should be noted that self-similarity is a very wide framework, encompassing attacks ranging from probability one related-key differentials to slide attacks. To the best of our knowledge, the commutation property we consider here is very different from any previous work.

There is no prior cryptanalysis of LS-designs. As for iSCREAM, an issue with the padding in the original CAESAR submission of SCREAM and iSCREAM was pointed out in [28] and subsequently corrected. Our attacks have caused iSCREAM to be temporarily withdrawn from the CAESAR competition for a redesign [17].

By contrast, many attacks have been carried out against Zorro, mostly differential or linear in nature [19,30,26,3,29]. The best attack in [3] is a differential attack requiring 2^{41} data and time complexity 2^{45} to break the full cipher. Our attack is of a different nature: it holds in the weak key setting (with 2^{96} weak keys out of 2^{128}), requires minimal data and time, and is independent of the number of rounds. Similar to [3], our attack can be readily extended to Zorro-like ciphers using the approach in Appendix F.

Structure of the paper

In Section 1, we recall the definitions of invariant subspace and present our generic algorithm for detecting such invariant subspaces. In Section 2, we provide a description of LS-designs, including our targets Robin and iSCREAM. In Sections 3, 4 and 5, we develop our attacks against LS-designs, introduce a particular self-similarity property, the resulting invariant subspaces, and finally describe a different invariant subspace attack not underpinned by self-similarity. In Section 6, we apply our self-similarity and invariant subspace attacks to Zorro. Finally, in Section 7, we conclude with a discussion of our results and outline interesting open problems.

1 A Generic Algorithm to Detect Invariant Subspaces

In this section we first recall the invariant subspace attack and later present our algorithmic approach to detect invariant subspaces in a generic manner.

1.1 Invariant Subspace Attacks

Invariant subspace attacks were introduced and applied to PRINTCIPHER in [22]. We briefly recall the basic principle here.

Consider a n-bit block cipher with round function F_K consisting of a key addition and a SP layer $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$. That is, F_K is defined by $F_K(x) = F(x + K)$. Assume the SP-layer F is such there exists a subspace $A \subseteq \mathbb{F}_2^n$ and two constants $u, v \in \mathbb{F}_2^n$ with the property:

$$F(u + A) = v + A$$

Then, given a (round) key $K \in u - v + A$, i.e. $K = K_A + u - v$ with $K_A \in A$, the following holds:

$$F_K(v + A) = F(v + A + u - v) = F(u + A) = v + A$$

i.e. the round function maps the affine subspace $v + A$ onto itself. If all round keys are in $u - v + A$, in particular if identical round keys are used as in LS-designs and Zorro, then this property is iterative over an arbitrary number of rounds.

In the case where an identical key is added in every round (there is no key schedule), a key is said to be weak iff it belongs to $u - v + A$. Whenever a key is weak, plaintexts in $v + A$ are mapped to ciphertexts in $v + A$, breaking plaintext confidentiality. The number of weak keys is the cardinality of A.

In order to detect whether an unknown key is weak, it is enough to encrypt one plaintext in $v + A$, and test whether the resulting ciphertext is in the same space. Indeed, over the set of all keys, false positives will occur with the same frequency as true positives, and can be discarded with a second chosen plaintext.

1.2 A Generic Algorithm

In this section we present a simple and entirely generic probabilistic algorithm able to discover invariant subspaces for a given round function. The algorithm gives instant results for vector subspaces, and is able to discover affine subspaces in time proportional to their density. Despite its simplicity, this algorithm is enough to automatically discover all invariant subspace attacks to be elaborated upon in the following sections.

The algorithm will identify minimal invariant subspaces and thereby identify invariant subspace attacks automatically. However, further analysis usually allows to significantly improve upon the attacks recovered automatically by the algorithm and gain further insights in the structure of the detected weakness. Furthermore, as the expected running time is determined by the density of invariant subspaces, it might well be that not all possible attacks are detected. Thus, for the moment, this generic algorithm cannot be used to fully exclude the existence of invariant subspaces.

Identifying Minimal Subspaces Assume we are given a permutation $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$. Here F could be a (keyless) round of a block cipher or a cryptographic permutation (like Keccak-f). Our goal is to find affine subspaces $u + A \subset \mathbb{F}_2^n$ such that:

$$F(u + A) = v + A$$

for some $v \in \mathbb{F}_2^n$.

Our algorithm is based on the following trivial observation.

Lemma 1. *Assume $u + A$ is an affine subspace such that $F(u + A)$ is also an affine subspace $v + A$. Then for any subset $X \subseteq A$, the linear span of $(F(u + X) - v) \cup X$ is contained in A.*

The idea is to first guess one possible offset u' of the affine space to be found and use $v' = F(u')$. Next, we guess a one-dimensional subspace of A, denote this by A_0. The algorithm will succeed if and only if $u' + A_0$ is contained in $u + A$.

1. We compute A_{i+1} from A_i as:

$$A_{i+1} = \text{span}\{(F(u' + A_i) - v') \cup A_i\}$$

2. If the dimension of A_{i+1} equals the dimension of A_i, we found an invariant subspace and exit.
3. If not, we continue with step 1.

Thus, the idea is to start with what we denote *nucleon* of A and map it using F until it stabilizes. In the case that our initial guess was wrong and $u' + A_0$ is not contained in some non-trivial invariant subspace we will end up with the full space after at most n iterations of the above.

Note that it is not necessary to really map the complete spaces A_i using F but a randomly chosen subset of relatively small size is enough for our purpose and significantly speeds up the process.

If the largest invariant subspace of F has dimension d, the algorithm will detect this space (or any invariant subspaces of this space) after an expected number of $2^{2(n-d)}$ guesses for A_0 and u'. Thus, in this basic form, the algorithm becomes quickly impractical. However, in the case of round functions of a cipher (or a cryptographic permutation) that differ by round constants only, its running time can be greatly improved as described next.

Knowing the Nucleon For block ciphers with identical round keys or cryptographic permutations, we actually have a very good idea about the nucleon we want to be included in the space A, namely the round constants. More precisely, we consider round functions $F_i : \mathbb{F}_2^n \to \mathbb{F}_2^n$ that differ only by the addition of constants, i.e.

$$F_i(x) = F(x) + c_i$$

for $c_i \in \mathbb{F}_2^n$, where for simplicity we assume $c_0 = 0$. We are looking for affine subspaces $u + A$ that are mapped to $v + A$ by all round functions. In particular

$$F_0(u + A) = F(u + A) = v + A$$

and

$$F_i(u + A) = F(u + A) + c_i = v + A$$

which implies

$$v + A = c_i + v + A$$

and thus $c_i \in A$. Thus, given the situation as above, any subspace that is invariant under all round functions must necessarily contain the linear span of all round constants c_i.

For the algorithm outlined above this has significant consequences. Here, the only thing we have to guess is the offset. Therefore, the expected number of iterations of the algorithm is reduced from $2^{2(n-d)}$ to 2^{n-d}.

Moreover, after running the algorithm for m iterations with randomly chosen guesses for the offset, the probability that an invariant subspace of dimension d is not detected by the approach is given by

$$p_{m,n,d} := \left(1 - 2^{n-d}\right)^m$$

which can be approximated by

$$\log p_{m,n,d} \approx -m2^{d-n}.$$

The Algorithm

1: **procedure** CLOSURE(function F, nucleon A, offset u)
2: $v \leftarrow F(u)$
3: StableCount $\leftarrow 0$
4: **while** StableCount $< N$ **do**
5: Pick a random $x \xleftarrow{\$} u + \text{span}\{A\}$

```
        if F(x) − v ∈ span{A} then
            StableCount = StableCount + 1
        else
            Add F(x) − v to A
            StableCount ← 0
        end if
    end while
    return u + span{A}
end procedure
```

For offset u and nucleon A, the above procedure outputs the smallest affine subspace containing $u + \text{span}\{A\}$, that is mapped to a coset of the same space by F (with high probability). The algorithm depends on a global parameter N that controls the risk of error. Namely, when the algorithm exits, elements of $u + \text{span}\{A\}$ are mapped to $v + \text{span}\{A\}$ with probability greater than $1 - 2^{-N}$. This probabilistic result is enough for an invariant subspace attack to go through even for moderate choices of N.

Guessing the Offset If we are actually looking for stable *vector* spaces rather than affine spaces, as will be the case in the S-box independent setting described in Section 3.2, guessing the offset is not needed: we can choose zero as the offset. Then the algorithm above finds the smallest invariant subspace instantly.

In the general case where we are looking for any (affine) invariant subspace, we need to guess one offset u belonging to the affine space we are searching for. Then we can run the procedure above to find the generated invariant subspace, if it exists (otherwise, the algorithm will simply output the full space). If the space we are looking for has dimension d, guessing such an offset u by brute force will require 2^{n-d} tries on average. Of course we just require *one* invariant subspace; so in general 2^{n-d} can be replaced by the density of vectors belonging to (non-trivial) invariant subspaces.

Each iteration of the algorithm requires Gaussian reduction to determine whether a certain n-bit vector belongs to some subspace, amounting to n^2 operations. Hence the overall running time to find an invariant subspace of dimension d is roughly $n^2 \cdot 2^{n-d}$. Thus if n is large, the above approach will only work if $n - d$ is relatively small, or more generally the density of invariant subspaces is large. The case where n is small is also useful in order to find invariant subspaces through a single S-box: this is how we found spaces in Appendix A (after making the algorithm deterministic and exhaustive, which is affordable for small n).

1.3 Applications

We applied the algorithm to the block ciphers Zorro, Robin, Fantomas, LED and NOEKEON, as well as to the CAESAR candidate iSCREAM. We chose $N = 50$ to be very conservative. We ran the algorithm with approximately 2^{34} iterations for each primitive, stopping earlier in the case where an invariant subspace was detected. The results are summarized in the table below.

Table 1. Experimental Results: Here n is the block size and $d_{0.001}$ is the smallest dimension of an invariant subspace that has a probability to exist upper bounded by 0.001

Primitive	n	Dimension found	$d_{0.001}$	Running Time (h)
LED	64	-	34	24
NOEKEON [12]	128	-	98	40
Fantomas	128	-	98	40
Robin	128	96	-	22
iSCREAM	128	96	-	22
Zorro	128	96	-	<1

For LED, NOEKEON and Fantomas, no invariant subspaces were detected given our limited iterations. In that case, Table 1 indicates the dimension $d_{0.001}$ of the largest invariant subspace that has a probability to exist upper bounded by 0.001. More precisely, if x denotes the codimension of the largest invariant subspace, each random guess of an offset has probability 2^{-x} of falling into this subspace. After T tries, the probability of not having found the subspace is thus $(1-2^{-x})^T \approx e^{-T2^{-x}}$. We want this probability to be $1/1000$ within $T = 2^{32}$ tries, which yields $x = 32 - \log(\ln(1000)) \approx 30$, so $d_{0.001} \approx n - 30$. Thus it is unlikely that invariant subspaces of dimension above 98 exist for NOEKEON. However, the existence of smaller subspaces cannot be excluded with high probability by our results.

As we will show below, for Zorro, Robin and the CAESAR candidate iSCREAM the largest invariant subspace has dimension 96 out of 128, i.e. density 2^{-32}. Thus the time complexity is expected to be 2^{32} Gaussian eliminations on 128×128 binary matrices. Our experiments confirm this estimation. Discovering the invariant subspace took 22 hours on a single desktop PC equipped with an Intel Xeon Core i7 with 12 virtual cores used in parallel.

In the case of Zorro, we chose to use a single round as target function, rather than the four rounds separating key addition. It turns out many cosets of the invariant subspace in Appendix E are sent to another coset by a single round (namely, all cosets stemming with offsets where cells 0 and 3 are equal). Our generic approach discovers this fact and the associated subspace instantly, hence the "< 1" time in the previous table.

As mentioned in the introduction, a detailed analysis of the findings of the generic algorithm allows to understand the underlying structure of the invariant subspaces we have found, and improve the attacks. We present those findings in the following sections.

2 Description of LS-Designs, Robin, and iSCREAM

2.1 LS-Designs

LS-designs were introduced by Grosso, Leurent, Standaert and Varici at FSE 2014 [15]. We refer the interested reader to their article for a detailed presentation of LS-designs and their design rationale. For our purpose, a brief technical description suffices.

An LS-design is a block cipher encrypting n-bit plaintext blocks using a n-bit key. The inner state of the cipher, as well as the plaintext, ciphertext, and key, are all represented as an $r \times c$ bit array, with r the number of rows and c the number of columns. A concrete LS-design is parametrized by the following components:

- A choice of r and c. The size of the key and message blocks is $n = r \cdot c$.
- An r-bit S-box s.
- A bijective linear map ℓ on c-bit vectors, called the L-box.
- A number of rounds t.
- A choice of k-bit round constants $C(i)$ for $1 \leq i \leq t$.

In order to encrypt a given n-bit plaintext block, the plaintext is first loaded into the inner state of the cipher, and the master key is added in (all additions are bitwise `XOR`s). Then a round function is applied successively for rounds 1 to t. At that point the cipherext is equal to the inner state. The round function at round i proceeds as follows:

1. The round constant $C(i)$ is added to the inner state.
2. The S-box s is applied to each column of the state.
3. The L-box ℓ is applied to each row of the state.
4. The n-bit master key K is added to the state.

2.2 Notation

When dealing with LS-designs, we will always use the previous notation; that is:

r	the number of rows of the state.
c	the number of columns.
n	the size of the state; that is, $n = r \cdot c$.
s	the r-bit S-box.
S	the S-box step; that is, the application of s on each column of the state.
ℓ	the $c \times c$ binary matrix representing the linear layer, identified with the corresponding linear map on $\mathbb{F}_2^c$.
L	the L-box step; that is, application of ℓ on each row of the state.

2.3 Robin

In [15], two concrete LS-designs are proposed, Robin and Fantomas. The idea behind Robin is that both the S-box and L-box are involutive. This allows the same circuitry to be reused when computing these components and their inverse operation, i.e. when encrypting and decrypting. This saves valuable space on embedded devices when both encryption and decryption capabilities are required. The trade-off is that involutive components have more structure, resulting in a slightly higher number of rounds to reach the same security level as an LS-design based on non-involutive components.

Robin strictly fits within the LS-design framework recalled in the previous section. As such it can be fully described by the following parameters:

- The inner state of Robin has 8 rows and 16 columns, resulting in 128-bit blocks and a 128-bit key.
- The 8-bit involutive S-box is given in [15].
- The 16-bit involutive L-box is depicted as a 16×16 binary matrix on Fig. 1.
- The number of rounds is 16.
- At round i (starting from 1), the round constant $C(i)$ is zero outside of the first row, where it is equal to $\ell(i)$, with ℓ the L-box matrix.

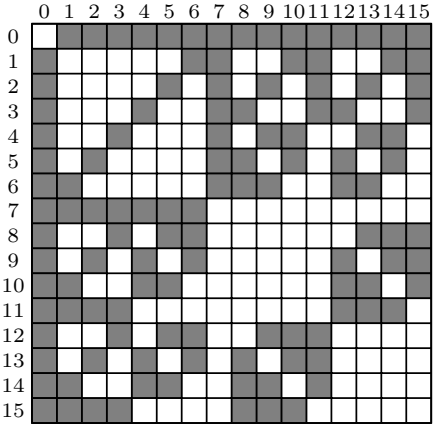

Fig. 1. Matrix representing the L-box of Robin and iSCREAM. Dark cells stand for 1's and white cells for 0's.

2.4 iSCREAM

SCREAM and iSCREAM [18] are two authenticated ciphers closely related to LS-designs. In fact iSCREAM is essentially a tweaked version of Robin, together with a Tweakable Authenticated Encryption (TAE) mode of operation [23]. Meanwhile SCREAM is similar to Fantomas, with a

different linear layer. The TAE mode of operation requires a tweakable block cipher [23]. Accordingly, the difference between the block cipher underlying iSCREAM and Robin stems from the introduction of a 128-bit tweak T into the (previously non-existent) key schedule.

In the remainder of this article we focus on weaknesses of the block cipher on which iSCREAM is built, independently of the mode of operation. We may abuse notations and write iSCREAM to mean its underlying block cipher.

This block cipher can be described as an LS-design, except for the fact that during the key addition phase, instead of adding in K every round: at odd rounds, $K + T$ is added; while at even rounds, $T \lll_c 1$ is added, where $T \lll_c 1$ denotes a circular shift of the columns of T by one column towards the left. The combination of two rounds is called a step. Beside that, iSCREAM can be described by the following parameters:

- The inner state of iSCREAM has 8 rows and 16 columns, resulting in 128-bit blocks and a 128-bit key.
- The S-box and L-box are those of Robin.
- The number of rounds depends on the required security level. The original article lists six variants. However the primary recommendation for iSCREAM as per CAESAR requirements is 12 steps (24 rounds) [16]. A secondary recommendation claiming related-key security has 14 steps (28 rounds). Since our attacks are essentially independent of the number of rounds, we omit other variants.
- At round i (starting from 1), the round constant $C(i)$ is zero outside of the first row, where it is equal to $27 \cdot i$ modulo 256 (affecting only the first 8 bits of the row).

3 Invariant Permutation Attack

In the next two sections, we analyze the invariant subspace discovered by the generic algorithm on Robin and iSCREAM. This subspace is actually induced by a particular type of self-similarity of independent interest, as is the invariant subspace of Zorro. Using this self-similarity directly results in an even stronger attack, as will be discussed below.

We start by recalling the concept of self-similarity and explaining a link to commutating linear maps. These concepts will afterwards be applied to LS-designs in general and to Robin and iSCREAM in particular (as well as to Zorro in Section 6).

3.1 Self-Similarity Properties and Linear Commutant

In [4] and [8], self-similarity in general is defined as:

Definition 1 (Self-similarity in a block cipher). *For a fixed block cipher E, let $E_K(x)$ denote the ciphertext block resulting from the encryption of plaintext block x under key K. A self-similarity relation is given by invertible and efficiently computable mappings ϕ, ψ, θ such that:*

$$\forall K, x: \quad \theta(E_K(x)) = E_{\psi(K)}(\phi(x))$$

What we are interested in is the case where $M = \phi = \psi = \theta$ is a linear map. This situation will arise if the cipher follows a generalized Even-Mansour structure where key-independent round functions F_i alternate with the addition of a fixed key K (i.e. no key schedule); and M commutes with the round functions F_i. This last condition is very demanding; but this is precisely what happens in both Robin and Zorro, despite their different structure. We expand on why this might be the case in the discussion (Section 7). The following lemma sums up the attack.

Lemma 2. *Consider a block cipher composed of round functions F_i separated by addition of a fixed key K. Suppose there exists a linear map M such that M commutes with the F_i's. Then:*

$$\forall x: \quad M(E_K(x)) = E_{M(K)}(M(x))$$

In particular, if $K = M(K)$:

$$\forall x: \quad M(E_K(x)) = E_K(M(x))$$

The commutativity of M and the round functions can be interpreted from the invariant subspace perspective. Indeed, if we let $A = \ker(M^i + Id)$ for any i, A is an invariant subspace[5]. Of course self-similarity is a stronger property stemming from a stronger requirement on the cipher.

In our applications, M will be involutive, so we focus on the case $i = 1$. In the remainder, whenever two plaintext blocks (or ciphertext blocks, or inner states, or keys) satisfy $x_2 = M(x_1)$, we say that they are *related*. If a plaintext block (or ciphertext block, or inner state, or key) is related to itself, we say that it is *self-related*. A *weak key* is a self-related key. In short, our attack states that weak keys map self-related plaintexts to self-related ciphertexts; while related keys map pairs of related plaintexts to pairs of related ciphertexts.

3.2 S-box-Independent Setting

We now focus on the case where the cipher is a substitution-permutation network (SPN), whose round function F_i consists of an S-box layer with identical S-boxes, a linear map L, addition of a round constant $C(i)$, and addition of a fixed key K. From the invariant subspace (resp. self-similarity) perspective, we are interested in subspaces (resp. linear maps) that traverse (resp. commute with) each of these components.

It is quite apparent that the main roadblock is the non-linear S-box layer. However even in a generic setting where we do not take into account a particular choice of S-box, any permutation of the S-box inputs will commute with the S-box layer (due to S-boxes being identical). Thus we restrict our attention to permutations of S-box inputs rather than general linear maps.

In terms of invariant subspaces, this corresponds to subspaces containing those vectors whose coordinates belonging to the same cycle in the permutation are equal; that is, subspaces that only require S-box inputs to be equal to some other input, or independent. We call such spaces *equality spaces*. Note that these are vector subspaces and no longer affine subspaces. Our strongest attacks actually occur in this setting.

As for constant and key addition, asking that their addition commutes with M amounts to asking that they belong to $\ker(M + Id)$. Now it remains to find permutations that commute with the linear layer. An efficient algorithm to do so is provided in Appendix G. The invariant subspace variant seems more difficult, as we do not know an algorithm able to efficiently enumerate equality spaces that traverse a linear map.

3.3 Key Recovery

The self-similarity attack above breaks plaintext confidentiality. In addition, if the commuting permutation P is involutive (as will be the case in our applications), efficient key recovery may be possible. In short, the part of the key corresponding to fixed points of the permutation can be guessed independently of the rest.

Intuitively, this is because if two self-related inner states differ only outside the fixed points of the permutation P, this difference will never be propagated to the fixed points of P. This is clear for the S-box layer (because the permutation operates on entire S-box inputs), but also holds for the linear layer. A general statement and proof are provided in Appendix B. In fact the proof also encompasses the case where the S-box layer is partial.

What we show in the appendix is that the cipher contains an embedded subcipher operating on the fixed points F of the permutation: we can project self-related plaintexts and ciphertexts on F and obtain a well-defined map. Note that this embedded subcipher may lend itself to further attacks; this is a direction we have not investigated, as we believe ciphers are sufficiently broken at that point.

[5] It may be that a non-trivial commuting matrix leads only to trivial invariant subspaces, as evidenced by the 2×2 binary matrix with rows [01] and [11]. However if M is involutive, $\ker(M + Id)$ is at least half of the space.

3.4 Invariant Permutation Attack on LS-Designs

Notation. In the S-box-independent setting of Section 3.2, for an LS-design, a permutation of S-box inputs is simply a permutation of the columns of the state. Let us write P for such a permutation. We always denote by the lowercase p its effect on a single row. Thus, P is the application of p on each row of the state. We identify p with the corresponding $c \times c$ permutation matrix. We adopt notations from Section 2.2.

The particular structure of LS-designs means that P commutes with L iff p commutes with ℓ. This is still a strong requirement, but we expect the L-box of an LS-design to have some structure in order to provide a good branch number, especially if it is involutive. In the case of Robin for instance, the linear layer is built from a Reed-Muller code and provides plenty of structure. Applied to LS-designs, Lemma 2 becomes:

Lemma 3. *For an LS-design, assume there exists a permutation P with the following properties:*

- *P commutes with L.*
- *$P(C(i)) = C(i)$ for all round indices i.*

Then for any plaintext message m:

$$\mathrm{Enc}_{P(K)}(P(m)) = P(\mathrm{Enc}_K(m))$$

In particular, if $K = P(K)$:

$$\mathrm{Enc}_K(P(m)) = P(\mathrm{Enc}_K(m))$$

Note that the identity permutation trivially satisfies the above requirements. Hereafter we always assume P is non-trivial. If $\mathrm{ncycles}(p)$ is the number of cycles of p, weak keys form a proportion $2^{-r\cdot(c-\mathrm{ncycles}(p))}$ of all keys (namely, those keys whose columns are equal on each cycle of p).

Key Recovery The previous attack breaks plaintext confidentiality. In addition, when P is involutive, efficient key recovery is possible, as announced in Section 3.3. A general statement and proof are provided in Appendix B.

It may still be worthwhile to provide a simpler statement dedicated to LS-designs. This is what we propose below.

Lemma 4. *Consider an LS-design, and assume there exists a permutation P with the same requirements as in Lemma 3. Also assume that P is an involution. Consider a weak key $K = P(K)$. Denote by F the set of fixed points of P.*

Take any self-related plaintext $m = P(m)$. Then the value of the ciphertext $\mathrm{Enc}_K(m)$ on the columns in F only depends on the value of m and K on the same columns.

Proof. Since P is an involution, all of its cycles have length 1 or 2. Hence we can partition the columns of the state into three subsets F, A, B, such that P is the identity on F, and maps A and B into each other. Take any self-related message m that is zero on F. Then the linear layer maps m to a self-related state $L(m)$ that is also zero on F. To see this, write $m = m_A + m_B$, where m_A is equal to m on A, and zero elsewhere, and likewise m_B is equal to m on B and zero elsewhere. Then $P(m_A) = m_B$, hence $P(L(m_A)) = L(m_B)$ by commutativity of P and L. Since P is the identity on F, this implies that $L(m_A) + L(m_B)$ is zero on F, so $L(m)$ is zero on F.

Thus, if $m = P(m)$ is zero on F, so is $L(m)$. By linearity, this implies that if m_1 and m_2 are self-related and equal on F, then so are $L(m_1)$ and $L(m_2)$. Thus, the property that two self-related states are equal on F goes through the linear layer. This property automatically goes through the S-box layer since it is column-wise. Since the same key and round constants are added to both sides, they have no impact. Hence this property goes through the whole cipher. □

As a direct consequence, the value of the key on the columns corresponding to fixed points of P can be guessed independently of the rest of the key by using any self-related plaintext. In addition, the embedded subcipher is a smaller LS-design, and may lend itself to further attacks. As a side note, both this lemma and the previous one also show that the cipher is malleable in a strong sense.

Permutation Characteristic Instead of considering only permutations P commuting with L, we can naturally look for pairs of permutations (P, Q) such that $L \cdot P = Q \cdot L$. We denote this by $P \to Q$, representing the fact that if two inner states are related by P before the linear layer, then after the linear layer they are related by Q.

From there we can hope to build a form of characteristic $P_0 \to P_1 \to P_2 \to P_3 \to \dots$ The commutative case in the previous section corresponds to $P \to P$. Note that the set of permutations P such that $Q = L \cdot P \cdot L^{-1}$ is a permutation forms a group. Also note that if L is involutive, $P \to Q$ is equivalent to $Q \to P$: indeed $L \cdot P = Q \cdot L$ implies $P^{-1} \cdot L = L \cdot Q^{-1}$, implies $L \cdot Q = P \cdot L$: hence any transition $P \to Q$ yields an iterative characteristic of length at most 2.

A particularly interesting case occurs whenever $P \to P^\alpha$ for some $\alpha \neq 0$. Indeed, in that case we automatically have a cyclic characteristic $P \to P^\alpha \to P^{\alpha^2} \to \dots \to P^{\alpha^i} = P$. Moreover the attack from Lemma 3 goes through with exactly the same requirements on the key and round constants (namely they are self-related by P).

Application to Robin Applying our attack to Robin amounts to finding a permutation p commuting with the matrix ℓ in Fig. 1, such that P leaves all round constants $C(i)$ invariant. More generally, as pointed out just above, we can actually look at transitions $P \to Q$, i.e. permutations p, q such that $\ell \cdot p = q \cdot \ell$. It turns out there are 720 such transitions, and all of them are of the form $P \to P^{-1}$. Moreover 76 of these permutations are involutive, and hence commute with L.

Recall that the round constants of Robin are defined as $C(i) = \ell(i)$ on the first row, and zero on the others, for $1 \leq i \leq t$. Hence we want $p(\ell(i)) = \ell(i)$, which amounts to $p(i) = i$ by commutativity. Since i ranges from 1 to 16, what we are looking for is simply permutations leaving the first 5 columns fixed. It turns out there exists exactly one such permutation, namely the involutive permutation P_0 switching columns 8, 9, 10, 11 respectively with columns 12, 13, 14, 15. Looking at Fig. 1, one can indeed see that permuting the rows and columns of the matrix of ℓ by p_0 leaves the matrix invariant, which is the same as saying p_0 commutes with ℓ.

With P_0, weak keys are simply keys whose last four columns are equal to the previous four. In particular the proportion of weak keys is 2^{-32}. Furthermore P_0 leaves the first 8 columns fixed, so Lemma 4 shows that for self-related plaintexts, the first 8 columns of ciphertexts only depend on the first 8 columns of plaintext and key. This makes it possible to guess the value of the master key on the first 8 columns independently of the rest of the key. This means 64 bits of the key can be guessed separately; then the remaining 64 bits are symmetric through P_0, so only 32 bits remain to be guessed. Thus the full key can be recovered in time complexity 2^{64} by encrypting any self-related message. This may yield a few solutions, which can be checked against any other plaintext/ciphertext pair.

Table 2. Permutations p_0, p_1 and p_2. Fixed points are omitted.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p_0									12	13	14	15	8	9	10	11
p_1					8	9	10	15	4	5	6					7
p_2					12	13	14	11				7	4	5	6	

Beside P_0, two other permutations P_1 and P_2 commuting with L leave the round constants invariant up to the very last round (cf. Table 2). This means related plaintexts are mapped to related inner states after 15 encryption rounds; followed by the final constant addition, S-box layer, L-box layer, and key addition. The final linear layer can be reversed, and the resulting states will agree on pairs of columns transposed by P on which $C(16)$ is equal. In both cases, there is one such pair, so self-related keys with respect to P_1 and P_2 can still be detected easily by encrypting a few self-related plaintexts, reversing the last linear layer, and checking that these two columns agree.

Permutations P_1 and P_2 both leave 8 columns fixed and hence yield an attack with essentially the same properties as P_0. Actually some key bits can be recovered faster than with P_0 thanks

to the one-round differential at the end, but this involves the symmetric part of the key (that is, outside the fixed points of the permutation) and thus the overall key recovery time is still 2^{64}.

Application to iSCREAM Recall that iSCREAM and Robin share the same linear layer. Round constants only affect the first eight columns of the state, and so we are looking for permutations commuting with L and leaving the first eight columns unchanged. As a matter of fact, there exists exactly one such permutation, namely the same permutation P_0 as above, which switches the last four columns of the state with the previous four.

Another difference between Robin and iSCREAM is the number of rounds, but that is actually irrelevant for our attack. The last difference is the presence of a tweak in the key schedule. Recall that at odd rounds, $T + K$ is added, while at even rounds, $T \lll_c 1$ is added, where T is a 128-bit tweak. In a chosen-tweak scenario, we can simply set T to zero, or any other value such that T and $T \lll_c 1$ are invariant by P. Then the attack against Robin from the previous section applies to iSCREAM essentially unchanged, with the same consequences.

A small variant of our attack is also possible when using P_0 as the commuting permutation. What we truly want is that $K + T$ and $T \lll_c 1$ should be self-related. This amounts to asking that columns 8, 9, 10, 11 should be equal to columns 12, 13, 14, 15. Since $T \lll_c 1$ is a column-wise shift of T by one column towards the left, this means that columns 9, 10, 11, 12 of T should be equal to columns 13, 14, 15, 0. Note that there is no condition on column 8 of T. As a consequence, for $K + T$ to be self-related for some choice of T, it is enough to ask that columns 9, 10, 11 of K should be equal to columns 13, 14, 15. Indeed in that case, we can fix T to be all-zero, except for column 8 which can take any value: exactly one such choice of T will satisfy that $K + T$ is self-related. Thus we obtain a larger set of weak keys (with ratio 2^{-24}), at the cost of requiring 2^8 chosen-tweak messages in order to detect whether a fixed unknown key is weak.

In addition, some variants of iSCREAM claim related-key security. If two keys are related by P_0, then our attack applies immediately without any weak key requirement, following the first consequence of Lemma 3. That is, related plaintexts are mapped by the related keys to related ciphertexts. Thus it is easy to check whether a pair of keys is related, and the cipher is broken in a strong sense.

Generalizations of the Permutation Attack There appears to be a few simple ways in which our attack could be generalized. We discuss them briefly here.

We could consider a probabilistic version of the attack. Instead of requiring $L \cdot P = P \cdot L$, we could consider P's such that the kernel of $L \cdot P - P \cdot L$ is almost the full space. In the case of Robin or iSCREAM, this would incur a cost at least 2^{-8} per round.

Another natural extension is to consider cases where all round constants are P-invariant except for the last few rounds (or first few rounds). Then our attack goes through most of the encryption process, and eventually yields a differential attack on the remaining rounds. When encrypting self-related plaintexts, this differential attack turns into an inner differential.

4 Invariant Equality Space Attack

In this section we study invariant subspaces for LS-designs following the S-box-independent setting of Section 3.2. We begin by defining equality spaces, and then present our results on Robin and iSCREAM. On the way, we will recover the invariant subspace detected by our generic algorithm (Section 1.3), and link it to the commuting permutations from the previous section.

4.1 Equality Spaces

As always, we use notations from Section 2.2. We always view n-bit vectors as an $r \times c$ matrix. In Section 3.2, we defined equality spaces in general terms for an SPN; we now provide a more specific definition suited to LS-designs.

Definition 2. *A subspace E of $\{0,1\}^c$ is an* equality space *iff there exists a partition of $\{0,\ldots,c-1\}$ such that E is the set of vectors whose values on coordinates belonging to the same class in the partition are equal.*

The dimension of E is the number of classes of the partition. By E^r we denote the set of n-bit states whose columns belonging to the same class in the partition underlying E are equal. Equivalently, this means that every row of the state belongs to E, hence the notation E^r. By extension we also call E^r an equality space. The point of this definition is that equality spaces are preserved by the S-box layer. The question is to determine which equality spaces are also preserved by the linear layer. That is, we are looking for equality spaces $E \subset \{0,1\}^c$ such that $\ell(E) = E$.

As pointed out in Section 3.1, when a permutation P commutes with L, the equality space defined by the cycles of P is preserved by the linear layer. The idea is that equality spaces preserved by the linear layer do not necessarily stem from a commuting permutation. Conversely, commuting permutations are an interesting special case, since they lead to a stronger property: indeed, when considering equality spaces rather than permutations, we are looking at a property of a single state, and there is no equivalent to the property that distinct related plaintexts are mapped to related ciphertexts; there is also no equivalent to Lemma 4. Meanwhile, Lemma 3 becomes:

Lemma 5. *For an LS-design, assume there exists an equality space E such that:*

- $\ell(E) = E$.
- $C(i) \in E^r$ *for all round indexes i.*

Then for any key K and plaintext message m:

$$\text{If } K \in E^r \text{ and } m \in E^r \text{ then } \mathrm{Enc}_K(m) \in E^r$$

The lemma trivially holds if E is the full space $\{0,1\}^c$; hereafter we assume this is not the case. Then we have an attack in the weak key setting, where weak keys are keys in E^r. Hence the proportion of weak keys is $2^{-r\cdot(c-\dim(E))}$.

4.2 Variants of the Attack

Essentially the same extensions as in Section 3.4 apply to equality spaces.

Characteristics: if the image $F = L(E)$ of an equality space E is also an equality space, we write $E \to F$. As with permutations, we can aim to build a characteristic $E_0 \to E_1 \to \ldots$ over several rounds. Note that the set of equality spaces is closed under intersection, and as a direct consequence, the set of equality spaces E such that $L(E)$ is an equality space is also closed under intersection. If L is involutive, $E \to F$ is equivalent to $F \to E$, so characteristics are automatically cyclic.

Probabilistic attack: Instead of asking $F = L(E)$, we can require the dimension of the quotient space $F/L(E)$ to be small.

Differential ending: If all round constants are in the required equality spaces except for the last few (or first few) rounds, it may be possible to cover the remaining rounds with an inner differential characteristic. Indeed in the case $E \to E$, the equality space attack may be seen as an all-zero inner differential attack.

Differential attack: the entire attack itself may be transposed into the differential world, at the expense of becoming probabilistic. Consider a state difference living in E^r with $L(E) = E$. Then at each round, require that the S-box layer preserves this equality; that is, the output of some S-boxes which receive equal input, should remain equal. Note that if E stems from an involutive permutation commuting with L, the columns corresponding to fixed points of the permutation can be set to a zero difference: this will be preserved by the linear layer (cf. the proof of Lemma 4). This attack avoids key and round constant requirements, at the cost of much lower probability, and hence high data requirements. In practice this would lead to a weaker attack against Robin than truncated differential product trails in the original article [15], because the branch number is 8 and the non-fixed points of P_0 involve 8 S-boxes.

4.3 Application to Robin and iSCREAM

Since Robin and iSCREAM share the same linear layer L, we consider them together. We enumerated all equality spaces E such that $L(E)$ is an equality space (there are around 2^{33} partitions of 16 elements, so this is feasible), and analyzed the results.

Our first observation is that there are many more *well-behaved* equality spaces E (in the sense that $L(E)$ is also an equality space), than *well-behaved* permutations P (in the sense that $Q = L \cdot P \cdot L^{-1}$ is also a permutation). Namely, there are 720 well-behaved permutations for L, while there are 30162 well-behaved spaces of dimension 8 or more. Even if we remove from this list spaces that are an intersection of larger well-behaved spaces (and thus could have a chance of indirectly resulting from well-behaved permutations), 7746 well-behaved spaces remain.

Recall that L is involutive, so any transition $E \to F$ (i.e. $L(E) = F$ with E and F two equality spaces) yields a cyclic characteristic $E \to F \to E$. Hence all well-behaved spaces belong to cycles of length 1 or 2. The aforementioned 7746 intersection-reduced well-behaved spaces of dimension at least 8 form 2506 cycles of length 1 (that is, $E \to E$) and 2620 cycles of length 2 (that is $E \to F \to E$). Thus equality spaces offer considerably more potential attacks, depending on round constants.

However, all equality spaces compatible with actual round constants for Robin minus the last round, and hence directly usable in an attack, stem from commuting permutations. There exist four such spaces: three of them correspond to permutations P_0, P_1 and P_2 from Table 2, and the last one is a space of dimension 8 resulting from the composition of any two of the previous permutations (any combination yields the same permutation or its inverse). As for iSCREAM, the only well-behaved space compatible with round constants is the one resulting from P_0. Thus, our previous attack is not improved. Moreover, the largest well-behaved spaces have dimension 12 and all stem from involutive permutations (there are 15 of them). The largest well-behaved equality spaces not stemming from a well-behaved permutation have dimension 10. This may be interpreted to mean that the strongest phenomenon is due to commuting permutations.

Thus for both Robin and iSCREAM, the equality space induced by P_0 is the only equality space that goes through the whole cipher, including the last round. This space has dimension 96 over $\mathbb{F}_2$, and it is the invariant subspace automatically discovered by the generic algorithm from Section 1.

4.4 A note on Fantomas and SCREAM

The matrix L of Fantomas is a permutation of the lines and columns of the matrix of Robin. As a consequence, they have the same number of well-behaved permutations and spaces. However we found no cycle among well-behaved spaces of Fantomas of dimension 6 or more (lower dimensions would yield very weak attacks); and no characteristic of length more than 2. Hence Fantomas seems safe from this attack.

The same is true for SCREAM. However, it is worth noting that there exists no well-behaved permutation for the matrix of SCREAM, while we found 5404 well-behaved spaces of dimension 8 or more.

5 A Second Invariant Subspace Attack on LS-Designs

In this section we present a different invariant subspace attack on LS-designs, which may be regarded as a form of dual of the previous attack. This attack does not stem from an underlying permutation; nor does it have an equivalent for Zorro. Thus, this section is specific to LS-designs, and takes advantage of their particular structure: namely, the fact that LS-designs not only rely on a layer of identical S-boxes, but also on a layer of identical L-boxes.

Now that we have understood the invariant subspace discovered by our generic algorithm as being an equality space, i.e. a space that is automatically preserved by the S-box layer, it is natural to ask if something similar can be done with the L-box layer. That is, we are now going to look for a property that is automatically preserved by the L-box layer.

This gives us more freedom, since we can leverage linearity. Essentially, if all columns of the state live in the same linear subspace, this will remain true after the linear layer (in Appendix D, we prove that this is in fact the most general property generically preserved by the linear layer); whereas in the previous case, we were limited to equality spaces. Beside this difference, the attack is essentially a dual version of the previous one, reversing the roles of the L-box and S-box layers.

5.1 Description of the Attack

In the previous attack, we searched for equality spaces $E \subset \{0,1\}^c$ on the rows of the state such that $\ell(E) = E$. Instead, we are now interested in general linear subspaces $A \subset \{0,1\}^r$ on the columns of the state such that $s(A) = A$. Once again, if A is a linear space on the columns (or one of its cosets), we denote by A^c the set of states whose columns all belong to A.

The core of the attack is the following: assume $s(A) = A$ for some linear space A. If the inner state lies in A^c, this will remain true after the S-box layer. Moreover, this property is automatically preserved by the linear layer. Indeed, the linear layer of an LS-design is not truly "line-wise": precisely because the same linear map is applied to each row, the linear layer may be seen as directly adding together column vectors. From this point of view, it becomes clear that if all columns lie in the same linear space A, this remains true after the linear layer.

Thus we are still within the invariant subspace framework, and follow the corresponding strategy: we choose A such that all round constants belong to A^c, and we consider a weak key scenario by requiring that the key also lie in A^c. If these requirements are fulfilled, plaintexts in A^c are mapped to ciphertexts in A^c.

More generally, we can consider cosets of linear spaces (i.e. affine spaces) rather than just linear spaces: indeed, as long as each coordinate at the output of ℓ is the sum of an odd number of coordinates at the input, the linear layer still preserves the property that all columns belong to a fixed coset. The following lemma sums up the attack.

Lemma 6. *Let u, v, w be r-bit vectors, and A be a linear subspace of r-bit vectors. Assume the following conditions hold:*

- *The S-box s maps all vectors in $u + A$ to vectors in $v + A$.*
- *Either $v = 0$ or all rows of the matrix of ℓ have an odd number of 1's.*
- *The columns of all round constants are in $w + A$.*
- *The columns of the key are in $(u + v + w) + A$.*

Then any plaintext in $(u + w) + A$ is encrypted into a ciphertext in $(u + w) + A$ (and conversely).

Weak keys are keys in $(u + v + w) + E$. This means a proportion $2^{-c\cdot(r-\dim(A))}$ of keys is weak.

5.2 Application to Robin and iSCREAM

In the case of Robin, the second condition in Lemma 6 is automatically true. In order to satisfy the third condition (round constants), since round constants only affect the first row of the state, we require that the r-bit vector denoted by 1, with 1 on the first row and 0 elsewhere, belongs to E. To instantiate the attack, it remains to look for affine spaces whose direction contains the vector 1, that are mapped by the S-box to affine spaces with the same direction.

It turns out the largest such spaces have dimension 3, and are mapped into themselves. We list all six choices in Table 3. Since these spaces have dimension 3, and the state has 8 rows and 16 columns, a proportion $2^{-16\cdot 5} = 2^{-80}$ of keys are weak. This means our attack is considerably weaker than the first one against Robin. By comparison, a generic multi-target time-memory trade-off with 2^{48} memory would lead to key recovery for the same proportion of keys. Of course our attack requires no memory or table lookup.

We now turn to iSCREAM. Recall that its S-box is the same as that of Robin, and round constants still only affect the first row of the state. We want both $K + T$ and T to live in the same coset, so we require T to lie in $(u + v + w + A)^c$, and K to lie in A^c. In our actual attack we have $u = v$ and $w = 0$ so in the end, we can set the tweak to zero (or any value in A^c), and the attack goes through with the same parameters as before.

Table 3. Six affine spaces of dimension 3 invariant through s.

Values in A								Dir(A)		
00	01	26	27	84	85	a2	a3	01	26	84
18	19	7c	7d	9e	9f	fa	fb	01	64	86
28	29	32	33	8a	8b	90	91	01	1a	a2
3c	3d	5e	5f	b2	b3	d0	d1	01	62	8e
44	45	66	67	c8	c9	ea	eb	01	22	8c
4e	4f	54	55	6c	6d	76	77	01	1a	22

5.3 Taking Advantage of the iSCREAM Tweak Schedule

In the case of iSCREAM, it is possible to leverage the tweak schedule to create a trade-off between the ratio of weak keys and the number of chosen-tweak messages required to detect a weak key. To simplify notations, we explain this technique using vector spaces; it extends to their cosets in a straightforward manner. Assume we have two vector spaces A and B with $S(A) = B$. As before, we assume $1 \in A$ and $1 \in B$ so that round constants belong to A^c and B^c. Since S is involutive, we have $S(B) = A$, so $A \to B \to A$ is a characteristic for the the S-box.

In order for this characteristic to traverse encryption, we need $K + T \in A^c$, and $T \lll_c 1 \in B^c$, which is equivalent to $T \in B^c$. For this it is enough to ask $K \in A^c + B^c = (A + B)^c$. Indeed in that case, write $K = K_A + K_B$ with $K_A \in A^c$ and $K_B \in B^c$. Then for $T = K_B$, we have $K + T \in A^c$ and $T \in B^c$, which is precisely what we want. Of course the key is unknown to the attacker, so she cannot compute T in this way. Instead, she can try every value in the supplementary space of A^c in $(A + B)^c$ (which is smaller than B^c, if only because $1 \in A \cap B$). For exactly one such value of the tweak, every plaintext in A^c will be encrypted to a ciphertext in B^c.

Now the question is to find two spaces A and B as above. Actually we look for cosets of linear spaces with the same properties, since the linear layer of iSCREAM also preserves these cosets. In summary, we look for affine spaces $u + A \neq v + B$ such that $S(u + A) = v + B$, and 1 belongs to $A \cap B$.

It turns out the largest such spaces have dimension 3. There are 11 such spaces (counting only 1 for $u + A \to v + B$ and $v + B \to u + A$), listed in Appendix A. Furthermore, 8 of these spaces satisfy $\dim(A + B) = 5$, which is the maximal possible value since 1 belongs to $A \cap B$. Thus $K \in (A + B)^c$ yields a ratio of weak keys of $2^{-c \cdot (r - \dim(A+B))} = 2^{-48}$.

In order to detect whether a key is weak, one needs to encrypt a message for each tweak in the supplementary of A^c in $(A + B)^c$, which is of dimension $2 \cdot c$, hence 2^{32} chosen-tweak messages are required (for a random key and a given choice of the tweak, a false positive has probability only 2^{-80}, and can be discarded by one additionnal chosen-tweak message). Finally, once a weak key is detected in this way, we know $K + T \in A^c$ for one specific T, hence $K = T + A^c$, so only $2^{c \cdot \dim(A)} = 2^{48}$ possibilities remain for the value of the key.

5.4 Variants of the Attack

It seems natural to consider a probabilistic version of the attack, where instead of requiring that every vector in $u + A$ be mapped by the S-box to a vector in $v + A$, we only require *most* of them to comply. If only x elements in $u + A$ are not mapped to $v + A$, the probability to pass an S-box is $1 - x/2^r$. The cost for each round is then $(1 - x/2^r)^c$. In the case of Robin, there is no A of dimension 4 with $x < 3$, so there does not appear to be an obvious interesting probabilistic version of the attack.

6 Commuting Permutation and Invariant Subspace for Zorro

6.1 Description of Zorro

The block cipher Zorro was introduced at CHES 2013 [14]. Like LS-designs, the design goal is to offer a cipher that can efficiently be made resistant to side-channel attacks through masking [25].

This is achieved by two main techniques: first, a carefully constructed 8-bit S-box; and second, an AES-like structure where S-boxes are only applied on the first row of the state.

The 128-bit state is represented as a 4×4 array of 8-bit cells. The round function applies the following transformations:

- `SubBytes`: A fixed 8-bit S-box is applied to the first row of the state.
- `AddConstant`: At round i, the constants i, i, i and $i << 3$ are added to the four cells of the first row (from left to right).
- `ShiftRow`: This step is identical to AES. Row i, counting from zero, is shifted by i cells to the left.
- `MixColumns`: This step is again identical to AES. A fixed 4×4 circulant matrix on $\mathbb{F}_{2^8}$ is applied to each column of the state. The matrix is the same as that of AES.

Four consecutive rounds are called a step. After each step, the 128-bit master key is simply added to the inner state: there is no key schedule. Encryption consists in key addition, followed by 6 steps (24 rounds), each followed by key addition.

6.2 Self-Similarity and Invariant Subspace

We are interested in an S-box-independent commuting linear map, as in Section 3.1. To simplify, we focus on a single round: commuting with every round is a sufficient condition to commute with every step. Thus we are looking for a linear map M acting as a permutation on the S-boxes, and commuting with the linear layer.

Since there are only four S-boxes, there are only 24 choices for the permutation. In fact, because the constant added to the fourth S-box is different from the others, we impose that this S-box should remain fixed by the permutation, leaving only 6 possibilities. In this way, our linear map will automatically commute with both the S-box and constant addition layers.

For each of the 6 permutation choices on the first 3 S-boxes, the set of linear maps behaving as this particular permutation on the first 4 cells, and independently on the other cells, is itself a vector space. Furthermore the commutant of the linear layer is naturally a vector space. Thus, it suffices to intersect these two spaces to find a solution, if it exists.

It turns out there exists exactly one solution, for the permutation swapping the first and third S-boxes, and leaving the other two fixed. This solution is given in Appendix E, together with the resulting invariant subspace. This subspace has dimension 12 over $\mathbb{F}_{2^8}$, that is, 96 over $\mathbb{F}_2$. Hence the proportion of weak keys is 2^{-32}.

In Appendix F, we show how to enumerate all invariant subspaces for Zorro, and deduce that the previous space is in fact the only invariant subspace (in the S-box-independent setting). The strategy used to enumerate spaces extends naturally to any SPN with a partial S-box layer of only a few S-boxes per round.

6.3 Key Recovery

The key recovery strategy from Section 3.4 extends to partial S-box layers such as Zorro. In brief, if an involutive linear map commutes with the components of an SPN, and acts as a permutation on the S-box inputs, part of the key may be recovered independently of the rest. When the S-box layer is full, i.e. the commuting map is simply a permutation, this part of the key corresponds to the fixed poins of the permutation. When the S-box layer is partial, and hence the commuting map M is not fully a permutation, the role of the non-fixed points is essentially played by $I = \mathrm{Im}(M + Id)$. A formal statement and proof are provided in Appendix B.

The consequence for Zorro is that once a key is recognized as weak, 64 bits of the key can be guessed independently of the rest using one chosen plaintext (any self-related plaintext). Indeed, the part of the key in I only influences the part of the ciphertext in I. After these 64 bits have been recovered by brute force, only 32 bits remain to be guessed, due to the key being weak. Thus key recovery requires only one chosen plaintext and a time complexity of 2^{64} offline encryptions.

7 Conclusion

In this article, we present a unified cryptanalysis of several ciphers based on invariant properties traversing the cipher under certain conditions, while providing generic tools for this type of attack. Our attacks are able to break lightweight ciphers Robin, iSCREAM and Zorro in a practical setting.

Our attacks from sections 4 and 5 are quite similar in principle. The state of an LS-design is a rectangular array. A fixed line-wise operation is performed in each direction. Each attack looks for properties of the inner state that would be *structurally* preserved in one direction (in the sense that this does not depend on the specificities of the S-box or linear layer), that would happen to also be preserved in the other (this time due to the particular choices of S or L).

In the case where the generic direction is linear, any linear space is preserved, and under some conditions any coset; if it is nonlinear, only equality spaces are preserved. In appendix C and D, we prove that these are in fact the most general properties structurally preserved in each direction, so our attacks fully realize the program outlined in the previous paragraph. It remains an open question whether a similar attack could in some way combine information from both directions; that is, neither direction would preserve the invariant property in a fully generic way.

Concerning our first attack on LS-designs from sections 3 and 4 (encompassing both invariant permutations and invariant equality spaces), the structure of the linear map is a key component. It seems unlikely that the attack could succeed in cases where the linear layer is not involutive. Indeed, as shown by the matrices of SCREAM and Fantomas, even in the presence of a large number of well-behaved equality spaces, it appears that iterative characteristics do not occur by accident. By contrast, if the linear layer is involutive, any well-behaved equality space (or permutation) yields a cyclic characteristic of length at most 2; and indeed, in the case of Robin and iSCREAM, thousands of iterative characteristics exist. Of course, the matrix of Robin and iSCREAM has much more structure than a generic involutive matrix.

It is quite striking that exactly the same attacks exist on Zorro, despite its quite different structure (byte-oriented vs. bit-oriented, partial S-box layer vs. full, AES-like vs. somewhat SERPENT-like). It is worth noting however that both ciphers attempt precisely the same goal, namely to offer efficient masked implementations. As a result both reduce non-linear operations to a minimum per round, while giving more weight to the linear layer; LS-designs achieve this by parallelizing the S-box through bit slicing; Zorro by resorting to a partial S-box layer. In both cases the contribution of the non-linear layer is very structured with respect to the linear layer; this, together with the minimal key schedule and simple round constants leads to our attacks.

We note that all our attacks can be prevented by a careful choice of round constants. One needs only ensure that no weaker (such as probabilistic or differential) version of the attack is left behind. This is particularly true when claiming related-key security (as in iSCREAM), since in this setting our attacks do not require weak keys, and hence weaker probabilistic versions are quite relevant.

Going back to the generic algorithm used to find the attacks, an interesting open problem is to specialize it to SPN structures, hoping to achieve better time complexity. In particular, it may be worthwhile to find an algorithm that is able to enumerate all invariant subspaces through a layer of n S-boxes, given n and the S-box. With improvements in time complexity, it may become possible to entirely disprove the existence of invariant subspaces for some SPNs.

Finally, we hope our analysis contributes some insight for the design of future ciphers with minimal key schedules and the choice of round constants in cryptographic permutations.

Acknowledgments

The authors would like to thank Henri Gilbert for many fruitful discussions related to the attacks presented in this article.

References

1. CAESAR– Competition for Authenticated Encryption: Security, Applicability, and Robustness. General secretary Daniel J. Bernstein, information available at `http://competitions.cr.yp.to/caesar.html`, 2013.

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 *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 57–76, 2014.
3. Achiya Bar-On, Itai Dinur, Orr Dunkelman, Virginie Lallemand, and Boaz Tsaban. Improved analysis of Zorro-like ciphers. Cryptology ePrint Archive, Report 2014/228, 2014. `http://eprint.iacr.org/2014/228`.
4. Elad Barkan and Eli Biham. In how many ways can you write rijndael? In Yuliang Zheng, editor, *Advances in Cryptology ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 160–175. Springer Berlin Heidelberg, 2002.
5. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404. `http://eprint.iacr.org/2013/404`.
6. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and Charlotte Vikkelsø. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer Berlin Heidelberg, 2007.
7. Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knežević, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçın. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In *ASIACRYPT*, volume 7658 of *LNCS*, pages 208–225. Springer, 2012.
8. Charles Bouillaguet, Orr Dunkelman, Gaëtan Leurent, and Pierre-Alain Fouque. Another look at complementation properties. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption*, volume 6147 of *Lecture Notes in Computer Science*, pages 347–364. Springer Berlin Heidelberg, 2010.
9. Stanislav Bulygin, Michael Walter, and Johannes Buchmann. Many weak keys for printcipher: Fast key recovery and countermeasures. In Ed Dawson, editor, *Topics in Cryptology CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 189–206. Springer Berlin Heidelberg, 2013.
10. Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 272–288, 2009.
11. David Chaum and Jan-Hendrik Evertse. Crytanalysis of des with a reduced number of rounds: Sequences of linear factors in block ciphers. In *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 192–211. Springer, 1985.
12. Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. NESSIE proposal: NOEKEON. Homepage `http://gro.noekeon.org/`, 2000.
13. Jan-Hendrik Evertse. Linear structures in blockciphers. In *Advances in Cryptology - EUROCRYPT '87, Workshop on the Theory and Application of of Cryptographic Techniques, Amsterdam, The Netherlands, April 13-15, 1987, Proceedings*, volume 304 of *Lecture Notes in Computer Science*, pages 249–266. Springer, 1987.
14. 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*, volume 8086 of *Lecture Notes in Computer Science*, pages 383–399. Springer Berlin Heidelberg, 2013.
15. 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, available at `http://www.uclouvain.be/crypto/people/show/382`, 2014.
16. Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici, François Durvaux, Lubos Gaspar, and Stéphanie Kerckhof. Addendum to the CAESAR submission for SCREAM and iSCREAM. Posted on the official CAESAR submission list, available at `http://competitions.cr.yp.to/round1/scream-ordering.txt`, 2014.
17. Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici, François Durvaux, Lubos Gaspar, and Stéphanie Kerckhof. CAESAR candidate SCREAM. Presentation by Gaëtan Leurent at DIAC 2014, available at `http://2014.diac.cr.yp.to/slides/leurent-scream.pdf`, 2014.
18. Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici, François Durvaux, Lubos Gaspar, and Stéphanie Kerckhof. SCREAM & iSCREAM. Entry in the CAESAR competition [1], available at `http://competitions.cr.yp.to/round1/screamv1.pdf`, 2014.

19. Jian Guo, Ivica Nikolić, Thomas Peyrin, and Lei Wang. Cryptanalysis of Zorro. Cryptology ePrint Archive, Report 2013/713, 2013. http://eprint.iacr.org/2013/713.
20. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matt Robshaw. The LED block cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer Berlin Heidelberg, 2011.
21. Ferhat Karakoç, Hüseyin Demirci, and Emre Harmancı. ITUbee: A Software Oriented Lightweight Block Cipher. In *Second International Workshop on Lightweight Cryptography for Security and Privacy (LightSec)*, 2013.
22. 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*, volume 6841 of *Lecture Notes in Computer Science*, pages 206–221. Springer Berlin Heidelberg, 2011.
23. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *Advances in Cryptology CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer Berlin Heidelberg, 2002.
24. Sean Murphy. An analysis of SAFER. *Journal of Cryptology*, 11(4):235–251, 1998.
25. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer Berlin Heidelberg, 2013.
26. Shahram Rasoolzadeh, Zahra Ahmadian, Mahmood Salmasizadeh, and Mohammad Reza Aref. Total break of Zorro using linear and differential attacks. *The ISC International Journal of Information Security*, 6(1), 2014. Available at http://isecure-journal.com/index.php/isecure/article/view/14-215/104.
27. J.A. Reeds and J.L. Manferdelli. Des has no per round linear factors. In GeorgeRobert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 377–389. Springer Berlin Heidelberg, 1985.
28. Siang Meng Sim and Lei Wang. Practical forgery attacks on SCREAM and iSCREAM. Posted on the crypto competitions mailing list at https://groups.google.com/d/forum/crypto-competitions; report available at http://www1.spms.ntu.edu.sg/~syllab/m/images/b/b3/ForgeryAttackOnSCREAM.pdf, 2014.
29. Hadi Soleimany. Probabilistic slide cryptanalysis and its applications to LED-64 and Zorro. To appear in the proceedings of FSE 2014, available at http://research.ics.aalto.fi/publications/bibdb2014/pdf/fse2014.pdf, 2014.
30. Yanfeng Wang, Wenling Wu, Zhiyuan Guo, and Xiaoli Yu. Differential cryptanalysis and linear distinguisher of full-round Zorro. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security*, volume 8479 of *Lecture Notes in Computer Science*, pages 308–323. Springer International Publishing, 2014.

A Well-Behaved Affine Spaces for the Robin and iSCREAM S-Box

Only spaces whose direction contains 1 are listed.

Values in $u + A$	Basis of A	Values in $v + B = S(u + A)$	Basis of B	$\dim(A + B)$
00 01 26 27 84 85 a2 a3	01 26 84	00 01 26 27 84 85 a2 a3	01 26 84	3
18 19 7c 7d 9e 9f fa fb	01 64 86	18 19 7c 7d 9e 9f fa fb	01 64 86	3
28 29 32 33 8a 8b 90 91	01 1a a2	90 91 8a 8b 32 33 28 29	01 1a a2	3
3c 3d 5e 5f b2 b3 d0 d1	01 62 8e	b2 b3 d0 d1 3c 3d 5e 5f	01 62 8e	3
44 45 66 67 c8 c9 ea eb	01 22 8c	c8 c9 ea eb 44 45 66 67	01 22 8c	3
4e 4f 54 55 6c 6d 76 77	01 1a 22	77 76 6d 6c 55 54 4f 4e	01 1a 22	3
28 29 32 33 6c 6d 76 77	01 1a 44	90 91 8a 8b 54 55 4e 4f	01 1a c4	4
28 29 32 33 4e 4f 54 55	01 1a 66	90 91 8a 8b 76 77 6c 6d	01 1a e6	4
2e 2f 38 39 8c 8d 9a 9b	01 16 a2	6f 6e 63 62 cd cc c1 c0	01 0c a2	4
08 09 2e 2f 8c 8d aa ab	01 26 84	4d 4c 6f 6e c1 c0 e3 e2	01 22 8c	5
08 09 38 39 9a 9b aa ab	01 30 92	4d 4c 63 62 cd cc e3 e2	01 2e 80	5
0a 0b 12 13 c6 c7 de df	01 18 cc	2c 2d 36 37 68 69 72 73	01 1a 44	5
0e 0f 16 17 c2 c3 da db	01 18 cc	bd bc ad ac f1 f0 e1 e0	01 10 4c	5
20 21 3e 3f 86 87 98 99	01 1e a6	59 58 5d 5c e9 e8 ed ec	01 04 b0	5
22 23 34 35 80 81 96 97	01 16 a2	78 79 74 75 d8 d9 d4 d5	01 0c a0	5
24 25 3a 3b 82 83 9c 9d	01 1e a6	47 46 43 42 fd fc f9 f8	01 04 ba	5
4a 4b 50 51 8e 8f 94 95	01 1a c4	e4 e5 f4 f5 a8 a9 b8 b9	01 10 4c	5

B Key Recovery for SPN with Commuting Permutation

We show that the key recovery strategy from Section 3.4 extends to Zorro. In fact we are going to prove the more general result announced in Section 3.3.

We want to encompass the case where the S-box layer is only partial. To do so, we relax the requirement that the commuting linear map M be a permutation, as follows. We ask that the linear map commuting with the round function present itself as a block matrix, where the first block covers all S-box inputs, and is only allowed to permute them as before; while the second block is a general bijective linear map with no further restriction (if the S-box layer is not partial, this second block is simply nonexistent).

In this more general case, the role of the points that were *not* fixed by the permutation will be played by $\mathrm{Im}(M + Id)$ (which indeed cannot contain any non-zero fixed point of M–note that we still require M to be involutive). Let us then denote $S = \ker(M + Id)$ (the self-related space we have considered so far), and define $I = \mathrm{Im}(M + Id)$. Observe that because M is involutive, I is a subspace of S (indeed, $(M + Id)(M(x) + x) = M(M(x) + x) + M(x) + x = M(M(x)) + x = 0$). Our claim is the following.

Lemma 7. *Consider an SPN whose round function is composed of:*

- *A potentially partial S-box layer;*
- *A linear layer L;*
- *Addition of a fixed key K;*
- *Addition of rounds constants.*

We assume the existence of an involutive *linear map M commuting with all of these components. Furthermore M is a block matrix, where the first block covers all S-box inputs, and only permutes them; while the second block is a general bijective linear map with no further restriction.*

Let $S = \ker(M + Id)$ and $I = \mathrm{Im}(M + Id)$. We already know that $x \in S$ and $K \in S$ implies $E_K(x) \in S$. But in addition:

$$\forall K_1, K_2, x_1, x_2 \in S : \quad \left.\begin{array}{r} x_1 + x_2 \in I \\ K_1 + K_2 \in I \end{array}\right\} \Rightarrow E_{K_1}(x_1) + E_{K_2}(x_2) \in I$$

Proof. We show that if two self-related inner states s_1 and s_2 satisfy $s_1 + s_2 \in I$, this remains true after every component of the round function:

- S-box layer: let us restrict our attention to the part E of the state affected by S-boxes. We view E as a subspace of the states, and we group bits corresponding to the same S-box input together to view E as a space on $\mathbb{F}_2^b$, where b is the bit size of an S-box. We know that M acts as a permutation P on E. Since P is involutive, E can be decomposed into 3 disjoint subspaces $E = F + A + B$, where F are the fixed points of the permutation, and A and B are mapped to each other by P. Let Z be the space of vectors whose value is zero on F in the previous decomposition.
  We claim that $I = S \cap Z$. Indeed $I \subseteq S$ because P is involutive; and $I \subseteq Z$ because $P(x) + x$ equals zero on fixed points of P; so $I \subseteq S \cap Z$. Conversely choose $x \in S \cap Z$; then let x' be the projection of x on A (parallel to $B+Z$); then $x = x' + P(x')$. To see this, use the decomposition along $Z + A + B$: x and $x' + P(x')$ are both zero on Z; they are equal on A, as x and x' are equal on A by definition and $P(x')$ is zero; and finally they are equal on B because they are equal on the rest and both sides are self-related. Thus we have shown $I \supseteq S \cap Z$; so $I = S \cap Z$.
  Now observe that if s_1 and s_2 satisfy $s_1 + s_2 \in Z$, this remains true after the S-box layer, since belonging to Z for the sum of two states only means that the columns of these two states in F are equal. We already know that belonging to S is preserved by the S-box layer; so in the end $s_1 + s_2 \in S \cap Z = I$ implies that this is still true after the S-box layer.
- Linear layer: assume $s_1 + s_2 \in I$; then $L(s_1) + L(s_2) \in L(I)$. However $L(I) = I$, because L commutes with $M + Id$, so the property of belonging to the image of $M + Id$ is stable by L. Indeed, $\forall x : \quad x \in I \Leftrightarrow x = P(y) + y \Leftrightarrow L(x) = P(L(y)) + L(y) \Leftrightarrow L(x) \in I$.
- Constant addition: this step does not affect the value of $s_1 + s_2$.
- Key addition: it is assumed that $K_1 + K_2 \in I$, so $s_1 + s_2 \in I$ implies $(s_1 + K_1) + (s_2 + K_2) \in I$. □

As a consequence, provided the key is in S (i.e. weak), and we encrypt self-related plaintexts, the value of the ciphertext on any supplementary space F of I in S only depends on the value of the plaintext and key on the same space (we denote this supplementary space by F, as in the case where M is a permutation on the full state, the fixed points of M is a valid choice). This allows us to try and guess the value of the key on F independently of the rest of the key, by encrypting any self-related plaintext.

The number of key bits we are thus allowed to guess independently of the rest is the dimension of F, which is $\dim(S) - \dim(I) = 2 \cdot \dim(S) - n$ (this number is positive because M is an involution). Another viewpoint is that the cipher contains an embedded subcipher operating on F with a $\dim(F)$-bit key: we can project self-related plaintexts, ciphertexts and keys on F parallel to I and obtain a well-defined new cipher.

C Properties Structurally Preserved by the S-Box Layer

In the following, we show that belonging to an equality space is essentially the only property that is preserved by the S-Box layer of an LS-design, under the hypothesis that we do not use any specific property of the S-box. While this is fairly intuitive, a formal proof may still be meaningful, if only as a complement to the proof in Appendix D.

Note that there is a small subtlety: the property that is being preserved is not only that all rows lie in some fixed equality space, but also that they are not included in any equality space stemming from a coarser partition. That is, not only columns that are equal remain equal, but so do columns that are distinct. This is mostly irrelevant for our attack; nonetheless, it has to be taken into account in a formal proof.

Earlier we wrote E^r to denote the set of states whose rows all lie in some equality space E. Accordingly, we now write E_r to denote the subset of E^r containing the states that not only satisfy that columns belonging to the same class in the partition underlying E are equal, but also that columns belonging to distinct classes are distinct.

Consider an LS-design with r rows and c columns, L-box L and S-box S. Assume we have two properties P and Q over the set of states $\mathcal{S}$ such that for all states x, $P(x) \Leftrightarrow Q(S(x))$, where S abusively denotes the column-wise application of S on the columns of x.

Let us write $\mathcal{P} = \{x \in \mathcal{S} : P(x)\}$ and $\mathcal{Q} = \{x \in \mathcal{S} : Q(x)\}$. Our goal is to show that $\mathcal{P} = \mathcal{Q}$, and that $\mathcal{P}$ is a union of sets of the form E_r. For any state x, define $E(x) \subseteq \{0,1\}^c$ as the equality space defined by the partition p of $\{0, \ldots, c-1\}$ such that i and j are in the same class iff columns i and j of x are equal.

Lemma 8. *For any two states x and y, there exists an S-box S such that $S(x) = y$ iff $E(x) = E(y)$.*

Proof. If $S(x) = y$, clearly columns that are equal in x are still equal in y, and distinct columns are still distinct by bijectivity of S. As a consequence $E(x) = E(y)$. Conversely, if $E(x) = E(y)$, the image of S on each column of x can be defined as the value of y on the same column. The condition $E(x) = E(y)$ guarantees that there is no contradiction. □

Corollary 1. *If $x \in \mathcal{P}$ then $E(x)_r \subseteq \mathcal{Q}$.*

Proof. Since our hypothesis is that we use no specific property of S, any possible image of $x \in \mathcal{P}$ through an S-box must belong to $\mathcal{Q}$. By the previous lemma, the possible images of x through an S-box are all states y such that $E(x) = E(y)$, which is precisely the set $E(x)_r$. Hence $E(x)_r \subseteq \mathcal{Q}$. □

As a result, $\bigcup_{x \in \mathcal{P}} E(x)_r \subseteq \mathcal{Q}$. Since $x \in E(x)_r$, this implies $\mathcal{P} \subseteq \mathcal{Q}$. By symmetry $\mathcal{P} = \mathcal{Q}$, and $\mathcal{P} \subseteq \bigcup_{x \in \mathcal{P}} E(x)_r \subseteq \mathcal{Q} = \mathcal{P}$, so $\mathcal{P}$ is a union of sets of the form E_r.

D Properties Structurally Preserved by the Linear Layer

In the following, we show that the only property structurally preserved by the linear layer of an LS-design is that the columns of the state generate a fixed linear space. By "structurally preserved", we mean preserved under the hypothesis that we do not use any specific property of the linear map L. The general outline of the proof is the same as that of Appendix C.

Note that there is a small subtlety: the property that is being preserved is not only that all columns of the state lie in some fixed linear space, but that they actually generate a fixed space. This is mostly irrelevant for our attack; nonetheless, it has to be taken into account in a formal proof. Earlier we wrote A^c to denote the set of states whose columns all lie in some subspace $A \subseteq \{0,1\}^r$. Accordingly, we now write A_c to denote the subset of A^c containing the states whose columns not only lie in A, but also generate A.

Consider an LS-design with r rows and c columns, L-box L and S-box S. Assume we have two properties P and Q over the set of states $\mathcal{S}$ such that for all states x, $P(x) \Leftrightarrow Q(L(x))$, where L abusively denotes the row-wise application of L on the columns of x. Let us write $\mathcal{P} = \{x \in \mathcal{S} : P(x)\}$ and $\mathcal{Q} = \{x \in \mathcal{S} : Q(x)\}$. Our goal is to show that $\mathcal{P} = \mathcal{Q}$, and that $\mathcal{P}$ is a union of sets of the form A_c. For any state x, define $A(x)$ as the subspace of $\{0,1\}^r$ generated by the columns of x.

Lemma 9. *For any two states x and y, there exists an L-box L such that $L(x) = y$ iff $A(x) = A(y)$.*

Proof. If $L(x) = y$, clearly the columns of y are linear combinations of the columns of x, so $A(y) \subseteq A(x)$. Since L is invertible, by symmetry $A(y) = A(x)$.

Conversely, assume $A(x) = A(y)$. For $0 \leq i < r$ and $0 \leq j < c$, denote by $x_{i,j}$ the bit at the intersection of row i and column j. Furthermore $x_{i,*}$ denotes the i-th row of x, and $x_{*,j}$ denotes the j-th column. Now define:

$$D(x) = \{F \subseteq \{0,1\}^r : \sum_{i \in F} x_{i,*} = 0\}$$

The core observation is that this is essentially the annihilator of $A(x)$ in the dual space $(\{0,1\}^r)^*$. Indeed, note that for any $F \subseteq \{0,1\}^r$, $\sum_{i \in F} x_{i,*} = 0$ iff this holds on each column of x, i.e. $\forall 0 \leq j < c, \sum_{i \in F} x_{i,j} = 0$. Now observe that $x_{*,j} \mapsto \sum_{i \in F} x_{i,j}$ is a linear functional on $\{0,1\}^r$, so $D(x)$ is the set of F's such that the corresponding linear functionals cancel all columns of x. Since the columns of x generate $A(x)$, this means exactly that these F's correspond to the linear functionals in the annihilator of $A(x)$.

Thus $D(x)$ is (in a natural bijection with) the annihilator of $A(x)$. The point of this observation is that our hypothesis $A(x) = A(y)$ is equivalent to $D(x) = D(y)$. This will prove useful shortly. Choose a maximal subset $I \subseteq \{0, \ldots, r-1\}$ such that the rows of x with indexes in I are independent. Since $D(x) = D(y)$, the same subset of rows of y is also independent. Thus we can choose a bijective map L such that $L(x_{i,*}) = y_{i,*}$ for each $i \in I$.

Now we claim that $L(x_{i,*}) = y_{i,*}$ for all i. Indeed, all indexes i not in I correspond to rows of x that are linear combinations of rows with indexes in I. Furthermore because $D(x) = D(y)$, the corresponding rows in y result from the same linear combinations of rows of y. Hence by linearity $L(x) = y$. □

Corollary 2. *If $x \in \mathcal{P}$ then $A(x)_c \subseteq \mathcal{Q}$.*

Proof. Since our hypothesis is that we use no specific property of L, any possible image of $x \in \mathcal{P}$ through an L-box must belong to $\mathcal{Q}$. By the previous lemma, the possible images of x through an L-box are all states y such that $A(x) = A(y)$, which is precisely the set $A(x)_c$. Hence $A(x)_c \subseteq \mathcal{Q}$. □

As a result, $\bigcup_{x \in \mathcal{P}} A(x)_c \subseteq \mathcal{Q}$. Since $x \in A(x)_c$, this implies $\mathcal{P} \subseteq \mathcal{Q}$. By symmetry $\mathcal{P} = \mathcal{Q}$, and $\mathcal{P} \subseteq \bigcup_{x \in \mathcal{P}} A(x)_c \subseteq \mathcal{Q} = \mathcal{P}$, so $\mathcal{P}$ is a union of sets of the form A_c.

E Commuting Linear Map and Invariant Subspace for Zorro

The commuting linear map M is represented as a 16×16 matrix over $\mathbb{F}_{2^8}$, using the AES representation of $\mathbb{F}_{2^8}$ as $\mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$.

$$\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 34 & 101 & 35 & 101 & 50 & 249 & 50 & 249 & 249 & 116 & 249 & 116 \\
0 & 0 & 0 & 0 & 101 & 35 & 101 & 34 & 249 & 50 & 249 & 50 & 116 & 249 & 116 & 249 \\
0 & 0 & 0 & 0 & 35 & 101 & 34 & 101 & 50 & 249 & 50 & 249 & 249 & 116 & 249 & 116 \\
0 & 0 & 0 & 0 & 101 & 34 & 101 & 35 & 249 & 50 & 249 & 50 & 116 & 249 & 116 & 249 \\
0 & 0 & 0 & 0 & 17 & 86 & 17 & 86 & 1 & 0 & 0 & 0 & 249 & 50 & 249 & 50 \\
0 & 0 & 0 & 0 & 86 & 17 & 86 & 17 & 0 & 0 & 0 & 1 & 50 & 249 & 50 & 249 \\
0 & 0 & 0 & 0 & 17 & 86 & 17 & 86 & 0 & 0 & 1 & 0 & 249 & 50 & 249 & 50 \\
0 & 0 & 0 & 0 & 86 & 17 & 86 & 17 & 0 & 1 & 0 & 0 & 50 & 249 & 50 & 249 \\
0 & 0 & 0 & 0 & 51 & 190 & 51 & 190 & 86 & 17 & 86 & 17 & 35 & 101 & 34 & 101 \\
0 & 0 & 0 & 0 & 190 & 51 & 190 & 51 & 17 & 86 & 17 & 86 & 101 & 34 & 101 & 35 \\
0 & 0 & 0 & 0 & 51 & 190 & 51 & 190 & 86 & 17 & 86 & 17 & 34 & 101 & 35 & 101 \\
0 & 0 & 0 & 0 & 190 & 51 & 190 & 51 & 17 & 86 & 17 & 86 & 101 & 35 & 101 & 34
\end{bmatrix}$$

The invariant subspace $\ker(M + Id)$ is generated by the following 12 row vectors, in the same representation.

```
(1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0)
(0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0)
(0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0)
(0  0  0  0  1  0  0  0  0  0  0  38 0  0  159 0)
(0  0  0  0  0  1  0  0  0  0  0  1  0  0  0   3)
(0  0  0  0  0  0  1  0  0  0  0  38 0  0  159 0)
(0  0  0  0  0  0  0  1  0  0  0  1  0  0  0   3)
(0  0  0  0  0  0  0  0  1  0  0  79 0  0  38  1)
(0  0  0  0  0  0  0  0  0  1  0  1  0  0  0   0)
(0  0  0  0  0  0  0  0  0  0  1  79 0  0  38  1)
(0  0  0  0  0  0  0  0  0  0  0  0  1  0  1   0)
(0  0  0  0  0  0  0  0  0  0  0  0  0  1  0   1)
```

F Enumerating Invariant Subspaces for Zorro

We view the inner state of Zorro as the 16-dimensional vector space E over $\mathbb{F}_{2^8}$. We have a basis $e_0, \ldots, e_{15}$ corresponding to the cells of the state. The linear layer of Zorro is viewed as a bijective linear map $\mathcal{L} : E \to E$.

Define $E_1 = \text{span}\{e_0, \ldots, e_3\}$, the linear space over the first row of the inner state, where S-boxes are applied. Similarly define $E_2 = \text{span}\{e_4, \ldots, e_{15}\}$. We are interested in stable subspaces $S \subset E$, i.e. $\mathcal{L}(S) = S$. Since we also want these spaces to be stable through the S-box and constant addition layers, we have some additional constraints.

Namely, we want our stable subspace to be the direct sum of an equality space on the S-box inputs (in the sense of Section 3.2), and a general vector subspace on the rest of the state; so that the S-box layer is traversed automatically. That is, $S = S_1 \oplus S_2$ with $S_1 = S \cap E_1$ and $S_2 = S \cap E_2$; and in addition, each cell $e_0, \ldots, e_3$ should be either independent, or equal to one of the other three cells. Furthermore, in order to traverse the constant addition layer, where at round i, the constant $i, i, i, i << 3$ is added respectively to e_0, e_1, e_2, e_3, we require that the fourth cell e_3 be independent of the others.

To summarize:

Our goal. We want to find stable subspaces $S \subset E$, i.e. $\mathcal{L}(S) = S$ satisfying the following two conditions:

(1) $S = S_1 \oplus S_2$, with $S_1 = S \cap E_1$ and $S_2 = S \cap E_2$.
$S_1 = \text{span}\{e_0, e_1, e_2, e_3\} = E_1$
or $S_1 = \text{span}\{e_0 \oplus e_1, e_2, e_3\}$
(2) **or** $S_1 = \text{span}\{e_0, e_1 \oplus e_2, e_3\}$
or $S_1 = \text{span}\{e_0 \oplus e_2, e_1, e_3\}$
or $S_1 = \text{span}\{e_0 \oplus e_1 \oplus e_2, e_3\}$

What we will do is find the smallest and largest solutions (for the inclusion order) to this problem, which are the most interesting. We will also have access to the other solutions if required. To reach this goal, we proceed in three steps.

Step 1. Choose S_1 among the five possibilities above. In the remainder, S_1 is fixed, and we always impose $S \cap E_1 = S_1$.

Since $S_1 \subseteq S$ and S is stable by $\mathcal{L}$, clearly $\mathcal{L}^i(S_1) \subseteq S$ for all i, where $\mathcal{L}^i$ denotes the i-th iteration of $\mathcal{L}$. Denote by $\overline{S_1} = \bigoplus_{i=0}^{\infty} \mathcal{L}^i(S)$ the closure of S_1 by $\mathcal{L}$. We have $\overline{S_1} \subseteq S$ and $\overline{S_1}$ is stable by $\mathcal{L}$.

Observe that $\mathcal{L}$ is idempotent (because it can be viewed as a permutation of a finite number of elements), so $\mathcal{L}^{-1} = \mathcal{L}^k$ for some $k > 0$. Hence the closure by $\mathcal{L}$ and $\mathcal{L}^{-1}$ is the same: $\overline{S_1}$ is closed by both. Also note that the closure can be computed effectively, as the sequence $A_n = \bigoplus_{i=0}^{n} \mathcal{L}^i(S)$ stabilizes as soon as $A_{n+1} = A_n$, and it can only increase less than 16 times (the dimension of E) until then, so at worst $\overline{S_1} = A_{16}$.

Once we have computed $\overline{S_1}$, two things can happen. Either $\overline{S_1}$ does not satisfy conditions (1) and (2) above. Since $\overline{S_1} \subset S$ is a necessity, this implies that S also fails these conditions, so there is no solution for our choice of S_1: we need to start over with a different choice. On the other hand, if $\overline{S_1}$ does satisfy both conditions, we have found a solution. In fact we have found the smallest solution (for the inclusion order) to our problem. Our goal now is to find the largest solution (and enumerate intermediate solutions if we want).

In the remainder, we assume that $\overline{S_1}$ satisfies conditions (1) and (2).

Step 2. Quotient E by $\overline{S_1}$, and denote by $\pi : E \to E/\overline{S_1}$ the corresponding projection. Now define the quotient map:

$$\begin{aligned} \mathcal{L}' : E/\overline{S_1} &\to E/\overline{S_1} \\ [x] &\mapsto [\mathcal{L}(x)] \end{aligned}$$

where $[x]$ denotes the class of an element in $E' = E/\overline{S_1}$. This map is well-defined because for all $x, y \in E$, $[x] = [y]$ iff $x \oplus y \in \overline{S_1}$ iff $\mathcal{L}(x \oplus y) \in \overline{S_1}$ (because $\overline{S_1}$ is stable by $\mathcal{L}$ and $\mathcal{L}^{-1}$) iff $[\mathcal{L}(x)] = [\mathcal{L}(y)]$.

Define $E_2' = \pi(E_2)$. The point of quotienting is the following lemma:

Lemma 10. *A subspace $S \subseteq E$ is stable by $\mathcal{L}$ and satisfies (1) and (2) iff $S' = \pi(S)$ is stable by $\mathcal{L}'$ and satisfies $S' \subseteq E_2'$.*

Thus we reduce our original problem to a simpler equivalent problem (which we will solve in step 3) with a single condition $S' \subseteq E_2'$. The trivial solution $\pi(S) = \{0\}$ corresponds to the minimal solution $S = \overline{S_1}$ we have already found; in general, π preserves inclusion, so the smallest and largest solutions are the same on both sides.

Proof of the lemma. We prove the implication in each direction. First, assume S is stable and satisfies (1) and (2). Then $\pi(S)$ is stable: for $[x] \in \pi(S)$, $\mathcal{L}'([x]) = [\mathcal{L}(x)] \in \pi(S)$ by stability of S. Moreover $S = S_1 \oplus S_2$, so $\pi(S) = \pi(S_1) \oplus \pi(S_2)$, and $\pi(S_1) \subseteq \pi(\overline{S_1}) = \{0\}$, so $\pi(S) \subseteq \pi(S_2) \subseteq \pi(E_2) = E_2'$.

Conversely, assume $\pi(S)$ is stable by $\mathcal{L}'$ and satisfies $\pi(S) \subseteq E_2'$. Recall that we have fixed $S_1 \subseteq S$, so $\overline{S_1} \subseteq S$. Let us show that S is stable: pick $x \in S$, then $[\mathcal{L}(x)] \in \pi(S)$ by stability of S', so $\mathcal{L}(x) = s + \overline{s_1}$ with $s \in S$ and $\overline{s_1} \in \overline{S_1} \subseteq S$, so $\mathcal{L}(x) \in S$.

Now we show that condition (1) holds. Observe that for any subspace $A \subseteq E$, $\pi^{-1}(\pi(A)) = A \oplus \overline{S_1}$. As a consequence, since $\overline{S_1} \subseteq S$, we have $S = \pi^{-1}(\pi(S))$. On the other hand, $\pi(S \cap E_2) = \pi(S) \cap \pi(E_2) = \pi(S)$ since $\pi(S) \subseteq \pi(E_2)$. Hence $S = \pi^{-1}(\pi(S \cap E_2))$, so $S = \overline{S_1} \oplus (S \cap E_2)$. We have already assumed $\overline{S_1} = S_1 \oplus (\overline{S_1} \cap E_2)$, so we get $S = S_1 \oplus (S \cap E_2)$ and we are done. As for condition (2), it holds iff it holds for $\overline{S_1}$, which is already assumed. □

Step 3. At Step 2, we have reduced our problem to finding $S' \subseteq E_2'$ stable by $\mathcal{L}' : E' \to E'$. Note that E', E_2' and $\mathcal{L}'$ are all effectively computable. Also note that any solution S' to this problem yields a solution of the original problem defined by $S = \pi^{-1}(S')$; and conversely. Now we separate two cases, depending on whether we chose $S_1 = E_1$ or otherwise.

Case 1: $S_1 = E_1$. In this case, $E_2' = E'$, so the condition $S' \subseteq E_2'$ is empty, and we need only find subspaces S' stable for $\mathcal{L}'$. In fact E' itself is stable by $\mathcal{L}'$, but this corresponds to the trivial solution $S = E$, which we do not care about.

However we only have to find stable subspaces for $\mathcal{L}'$ with no extra condition. We can write the matrix of $\mathcal{L}'$ in (block) Jordan form. Finding (non-trival) maximal spaces amounts to choosing all columns except the leftmost block in one Jordan block. The highest dimensional spaces correspond to excluding the smallest such block.

Case 2: $S_1 \subsetneq E_1$. In this case, $E_2' \subsetneq E'$ so E_2' is not a trivial solution. Moreover E_2' may not be stable by $\mathcal{L}'$. However, since $S' \subseteq E_2'$ is stable by $\mathcal{L}'$, we have $\mathcal{L}'(S') \subseteq E_2'$ hence $S' \subseteq \mathcal{L}'^{-1}(E_2')$. We can iterate this reasoning to deduce $S' \subseteq \mathcal{L}'^{-i}(E_2')$ for all $i \geq 0$. This means $S' \subseteq \bigcap_{i=0}^{\infty} \mathcal{L}'^{-i}(E_2')$.

As we have already observed earlier, $\mathcal{L}'$ is idempotent, so we can define $E_\cap = \bigcap_{i=0}^{\infty} \mathcal{L}'^{i}(E_2')$ $= \bigcap_{i=0}^{\infty} \mathcal{L}'^{-i}(E_2')$; and furthermore, $n \mapsto \bigcap_{i=0}^{n} \mathcal{L}'^{i}(E_2')$ stabilizes as soon as two consecutive terms are equal, so we need only iterate less than 16 times to compute $E_\cap$.

Thus we have $S' \subseteq E_\cap$. Now observe that $E_\cap$ is in fact stable by $\mathcal{L}'$ (again, because $\mathcal{L}'$ is idempotent). As a result, $S' = E_\cap$ is the (unique, as it turns out) largest solution to our problem, and we are done. If we want smaller intermediate solutions (between $S' = \{0\}$ and $S' = E_\cap$, if they are different) we can write the matrix of $\mathcal{L}'$ on $E_\cap$ (which is well-defined because $E_\cap$ is stable by $\mathcal{L}'$) in Jordan form, and proceed as in the previous step.

(Note: if we only want to find the largest space, in the case $S_1 \neq E_1$, it is enough to iterate $S_1 \oplus E_2$ through L, take the intersection of all resulting spaces, and check that S_1 is still included.)

Application to Zorro. The process described above could be easily adapted to any Zorro-like cipher, in the sense of any SPN with a partial non-linear layer of a few S-boxes per round. In the case of Zorro, we find that for $S_1 = \mathrm{span}\{e_0, e_1, e_2, e_3\}$, $S_1 = \mathrm{span}\{e_0 \oplus e_1, e_2, e_3\}$ and $S_1 = \mathrm{span}\{e_0, e_1 \oplus e_2, e_3\}$, $\overline{S_1}$ is the whole space; while for $S_1 = \mathrm{span}\{e_0 \oplus e_1 \oplus e_2, e_3\}$, $e_1 \in \overline{S_1}$, which contradicts the choice of S_1. Hence the only remaining choice is $S_1 = \mathrm{span}\{e_0 \oplus e_2, e_1, e_3\}$. For this choice, $\overline{S_1}$ is the invariant subspace in Appendix E. Moreover $E_\cap = \{0\}$, so there is no other solution. Thus, it turns out that the subspace in Appendix E is the only invariant subspace for Zorro in the S-box-independent setting.

G Enumerating Permutations that Commute with a Linear Map

Our target linear map is regarded as a binary matrix M. We will not only enumerate permutations that commute with M, but also all couples of permutations (P, Q) such that $P \cdot M = M \cdot Q$. Indeed these couples can be used in characteristics as in Section 3.4. In this appendix we describe a heuristic algorithm that gives instant results for all cases we have encountered.

Denote by n the size of M, i.e. M is a $n \times n$ binary matrix. Note that $P \cdot M$ is a permutation of the rows of M, while $M \cdot Q$ is a permutation of its columns. The core of our algorithm lies in the following simple observation. For any subset $R \subseteq \{1, \ldots, n\}$ of rows of size r, if we denote by M_R the $r \times n$ submatrix of M restricted to rows in R, then the columns of M_R are a permutation of the columns of $(M \cdot Q)_R$. As a consequence, the number of columns containing only 1's is the same on both sides. If $P \cdot M = M \cdot Q$, we can check this property for $(P \cdot M)_R$ and M_R, independently of Q.

This allows us to build P step by step: as soon as we have guessed the image of at least two elements by P, we can check that the previous criterion holds. That is, say we have guessed $P(1)$ and $P(2)$; then we can check that rows 1 and 2 of M contain the same number of columns with two 1's as its rows $P(1)$ and $P(2)$. If not our guess of P is invalid.

This yields the following algorithm: we guess $P(1)$, then $P(2)$, and so forth, scanning the tree of all permutations P. However, each time we venture a new guess, we check the previous criterion. If it is not satisfied, we prune the branch. Despite the simplicity of the criterion (which could easily be strengthened), in practice wrong choices are pruned very quickly, and we only scan a very small portion of the tree of permutations.

Once valid choices of full P are determined as outlined above, since M is expected to be invertible, all of its columns are distinct; hence there is only one possible choice of Q for a given choice of P. In all practical cases we have encountered (for the matrices of Robin, SCREAM and Fantomas), results are instant. In fact, we implemented a simpler algorithm where we only checked the previous criterion for *pairs* of rows as opposed to any subset: this proved to be enough in practice.