Practical Low Data-Complexity Subspace-Trail Cryptanalysis of Round-Reduced PRINCE

Grassi, Lorenzo; Rechberger, Christian

Published in:
Progress in Cryptology – INDOCRYPT 2016

Link to article, DOI:
10.1007/978-3-319-49890-4_18

Publication date:
2016

Document Version
Peer reviewed version

Practical low data-complexity subspace-trail cryptanalysis of round-reduced PRINCE

Extended Version

Lorenzo Grassi[1] and Christian Rechberger[1,2]

[1] IAIK, Graz University of Technology, Austria
[2] DTU Compute, Technical University of Denmark, Denmark
{firstname.lastname}@iaik.tugraz.at

Abstract. Subspace trail cryptanalysis is a very recent new cryptanalysis technique, and includes differential, truncated differential, impossible differential, and integral attacks as special cases.
In this paper, we consider PRINCE, a widely analyzed block cipher proposed in 2012. After the identification of a 2.5 rounds subspace trail of PRINCE, we present several (truncated differential) attacks up to 6 rounds of PRINCE. This includes a very practical attack with the lowest data complexity of only 8 plaintexts for 4 rounds, which co-won the final round of the PRINCE challenge in the 4-round chosen-plaintext category. The attacks have been verified using a C implementation.
Of independent interest, we consider a variant of PRINCE in which ShiftRows and MixLayer operations are exchanged in position. In particular, our result shows that the position of ShiftRows and MixLayer operations influences the security of PRINCE. The same analysis applies to follow-up designs inspired by PRINCE.

Keywords: PRINCE, Subspace Trails Cryptanalysis, Invariant Subspace Attack, Truncated Differential Attack, Practical Attack, MANTIS

1 Introduction

The area of lightweight cryptography involves ciphers with low implementation costs, adequate for use in smart devices that have very limited resources (regarding memory, computing power, battery supply). Lightweight ciphers are designed in order to ensure a high level of security, even in the presence of tight constraints, that is they should be designed as a trade-off between security, cost of implementation and performance.

One of the most analyzed recent lightweight block ciphers is PRINCE [7]. The structure was designed in order to have efficient instantaneously encryption of a given plaintext, i.e. the entire encryption and decryption process should take place within the shortest possible delay, using little chip area. Follow-up designs (e.g. [2], [3] and [5]) were inspired by PRINCE.

This is the extended version of the article submitted by the authors to Springer-Verlag, which appears in the proceedings of Indocrypt 2016.

PRINCE has already gained a lot of attention from the academic community, and some interesting cryptanalysis have been published. Most of the earlier attacks came with with very high time and data complexity. In order to encourage more practically relevant cryptanalysis, "The PRINCE Challenge"[1] was organized, which started in (middle) 2014 and recently concluded with its third round. The challenge involved two settings: a chosen-plaintext scenario and the known-plaintext one. Since the competition aims at finding practical attacks, submissions must respect some initial restrictions regarding data, time and memory complexity: a particular emphasis is on *restricting the amount of data* (plaintext) that is available to the attacker.

Studying practical attacks on round-reduced versions of ciphers is motivated in many ways, see e.g. [8] for a survey of such reasons. A recent example is a creative attack on a full version of a 2nd-round CAESAR candidate ELmD [4] which in turn relies on an attack on AES reduced to 6 rounds. As use-cases of PRINCE are particularly sensitive to the choice of the number of rounds (due to latency constraints) it is very interesting to understand how much security can be at most hoped for when rounds are reduced.

In this paper, we present truncated differential attacks on reduced PRINCE, derived in a natural way exploiting so-called "subspace trails" of reduced-versions of PRINCE. The subspace trail framework was recently introduced in [12] as generalization of the invariant-subspace attack [16, 17], and found already applications in the cryptanalysis of AES [12] and Simpira [21]. While we describe applications in various settings, we focus on 4-round reduced PRINCE. The result improves upon all earlier results and has the lowest data complexity while still being entirely practical and practically verified. *This attack also co-won the final round of the PRINCE challenge in the 4-round chosen-plaintext category.*

As a second important aspect, we study the security of PRINCE when the rounds are slightly modified. Without going into the details here already, a round of PRINCE is very similar to an AES one, with the main difference that the ShiftRows operation is computed after the MixLayer one (instead of before). We show that the order of these two operations influences the security of PRINCE, and we show a possible way to overcome this problem. The same analysis applies to other encryption schemes that follow the same design of PRINCE, as the low-latency tweakable block cipher MANTIS [5] presented at CRYPTO 2016.

Review of Attacks on PRINCE. Known cryptanalysis of PRINCE includes theoretical attacks and observations on round-reduced and full PRINCE, and also a number of practical attacks on round-reduced versions. Here we review those most relevant to our work.

Derbez and Perrin described in [9] attacks based on a Meet-in-the-Middle approach, applicable (theoretically) up to 10 rounds of the algorithm. In [18], Morawiecki introduced attack relying on Integral and Higher-Order-Differential

[1] `https://www.emsec.rub.de/research/research_startseite/prince-challenge/`

Cryptanalysis, up to 7 rounds[2]. This attack is based on a 3.5 round distinguisher on PRINCE with one active nibble. Starting from this work, Posteuca and Negara [19] found a 4.5 round integral distinguisher for PRINCE which needs three (not arbitrary) active nibbles instead of one.

Due to the involution structure of PRINCE, a modified version of a differential attack was presented in [1]. Instead of choosing pairs of plaintexts with a known difference and studying its propagation through the encryption process (as in a classical differential attack), authors are able to recover the key using the difference among the nibbles of the plaintexts and of the respective ciphertexts (the nibbles are in the same positions). A related work about truncated differentials [15] has been presented in [23], which showed the existence of 5- and 6-round truncated differential distinguishers.

Our Contribution. We describe practical key-recovery attacks based on subspace trails of PRINCE which resemble truncated differentials, and we analyze in details the security of PRINCE-like ciphers focusing on the order of ShiftRows and MixLayer operations.

We base our work on the *Subspace Trail Cryptanalysis*, a technique that was recently introduced in [12]. Starting from [12], in Sect. 3 we investigate the behavior of subspaces in PRINCE. At a high level, we fix a subspace of plaintexts that maintain predictable properties after repeated applications of a key-variant round function. In other words, we identify (constant dimensional) subspace trails, that is a coset of a plaintext subspace that encrypts to proper subspaces of the state space over several rounds.

In Sect. 4 we present an "*equivalent*" version of PRINCE (with respect to the attacks we consider), which allows a better understanding of the design of this encryption scheme. As we have already mentioned, a round of PRINCE is very similar to an AES one, with the main difference that the ShiftRows operation is computed after the MixLayer one. Our analysis shows that if these two operations are exchanged of position (to have something similar to AES), the attacks present in literature can usually cover more rounds with the same (or even less) complexity. As example, for this modified version it is possible to set up a subspace trail that covers one more round. Thus, we present how to modify the middle-rounds of PRINCE in order to obtain a version equivalent to the original one (also from the security point of view) and where the ShiftRows operation is computed before the MixLayer one. Similar analysis applies also to other encryption schemes that follows the (same) design of PRINCE. In particular, a detailed analysis for the MANTIS encryption scheme is presented in App. D. Finally, we highlight that this problem arises only for PRINCE-like ciphers, i.e. the security of AES-like cipher is not influenced by the position of the ShiftRows operation with respect to the MixLayer, while PRINCE-like ciphers are.

In the following sections, we use the found subspace trails as starting points to set up competitive key recovery attacks to round-reduced PRINCE. In particular,

[2] Table 1 of [18] contains an error about the data complexity and the time complexity for the Integral Attack on 4 rounds. The correct values for this attack (as also confirmed by the author of [18]) are those reported in Table 1 of this paper.

Table 1. *Comparison table of attacks on 4-round PRINCE.* These are the four central rounds, that is the middle rounds, one round before and one round after. Data complexity is measured in number of required chosen plaintexts (CP). Time complexity is measured in round-reduced PRINCE encryption equivalents (E). Memory complexity is measured in plaintexts (64 bits).

Technique	Data (CP)	Computation (E)	Memory	Reference
Trunc. Diff. Attack (EE)	$\mathbf{8 = 2^3}$	$\mathbf{2^{18.25}}$	**small**	**Sect. 5**
Bit-pattern Integral	$48 = 2^{5.6}$	2^{22}	small	[18]
(Pre-Computed) Integral	$64 = 2^6$	$2^{7.4}$	small	[20]
Integral	$160 = 2^{7.32}$	$2^{9.32}$	small	[18]
Trunc. Diff. Attack (EB)	$\mathbf{430 = 2^{8.75}}$	$\mathbf{2^{8.15}}$	**small**	**App. G**
Diff. / Logic	2^{10}	5 sec	$\ll 2^{27}$	[9]
Differential	2^{32}	$2^{56.26}$	2^{48}	[1]

(EE: Extension at End - EB: Extension at Beginning)

we present two different truncated differential key-recovery attacks on 3 rounds of PRINCE. The idea - described at the beginning of Sect. 5 and in details in App. E - is simple. Assume to fix a coset of a particular subspace $\mathcal{C}$ of the plaintexts space. After 2.5 rounds, each element of a (fixed) coset of $\mathcal{C}$ belongs to a coset of another particular subspace $\mathcal{M}$, i.e. a coset of $\mathcal{C}$ is mapped into a coset of $\mathcal{M}$ after 2.5 rounds. Equivalently, if two elements belong to the same coset of $\mathcal{C}$, after 2.5 rounds they belong to the same coset of $\mathcal{M}$ independently of the secret key. Thus, the key of the final round must satisfy the condition that, given two ciphertexts (whose plaintexts belong to the same coset of $\mathcal{C}$), they belong to the same coset of $\mathcal{M}$ half round before. As main result, we show that *a truncated differential attack that exploits relationships among the nibbles is (much) more powerful than one that works independently on each nibble.*

In Sect. 5, we show how to extend this attack to 4 rounds by adding one round at the end. This attack needs only 8 chosen plaintexts and it is the best one from the point of view of the data complexity (the computational cost is also very competitive), improving previous results of a factor 6 for the data complexity and of a factor 2^4 for the computational cost. All these attacks have been verified using a C/C++ implementation. A comparison of all known state of art of attacks on PRINCE and our attacks is given in Table[3] 1.

It is also possible to extend the attack on 3 rounds at the beginning (see App. G) which leads to higher data but lower time complexity. Using both the extension at the end and at the beginning, it is possible to attack 5- and 6-rounds of PRINCE (see App. H).

Practical Verification of 3- and 4-Rounds Attacks. We practically verified all the 3 rounds attacks described in this paper and the 4 rounds attack described

[3] This and the following tables have been created using "Cryptanalysis Zoo" by IAIK, TU Graz - see `http://cryptanalysiszoo.iaik.tugraz.at/Mw`

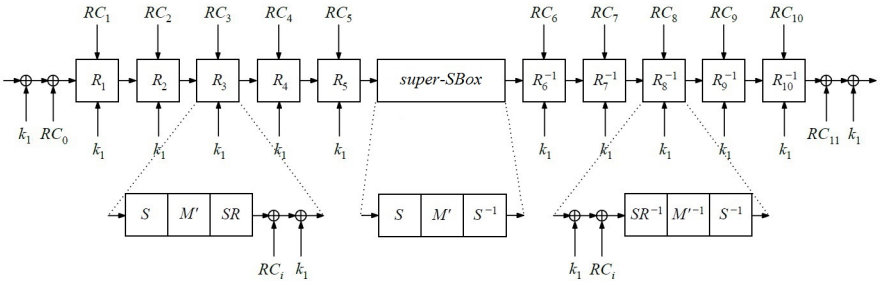

Fig. 1. A scheme of the PRINCEcore cipher.

in Sect. 5, using a C/C++ implementation[4]. For all the attacks, the full key recovery takes a fraction of a second on a desktop PC. We also practical verified some of the attacks (e.g. the square one) present in literature against the modified versions of PRINCE presented in Sect. 4.

2 Description of PRINCE

PRINCE [7] is a lightweight cipher with a state size of 64 bits - the 64-bits state of PRINCE can be visualized as a 4×4-matrix, where every cell represents a nibble - and a key length of 128 bits. It is based on the so-called FX construction [14], where one part of the key is used for a core cipher F, which contains the major encryption process, and the remaining parts are used for whitenings before and after the core: $FX_{k,k_1,k_2} = k_2 \oplus F_k(x \oplus k_1)$. First, the 128-bit key k is split into two 64-bit words (i.e. $k = (k_0||k_1)$), and then it is expanded into 192 with a simple linear transformation: $(k_0||k_0'||k_1) := (k_0||(k_0 \ggg 1) \oplus (k_0 \ll 63)||k_1)$. The 64-bit subkeys k_0 and k_0' are used as whitening keys to the underlying block cipher called PRINCEcore, while the 64-bit key k_1 is used for the core.

The core cipher "PRINCEcore" is a substitution-permutation network composed of 12 rounds (see Fig. 1). Every round in PRINCE consists of an S-Box layer, a Linear layer, a ShiftRows operation, a key addition and the addition of a round constant:

- **S-Box layer**: Every nibble in the internal state is replaced by using a 4×4-bit S-Box, which has algebraic degree 3 and which is differential 4-uniform.
- **Linear layer M′**: In the linear layer, the state is multiplied by an involutive 64×64-matrix, a kind of equivalent of MixColumns in AES. More precisely, two 16×16 submatrices $\hat{M}^{(0)}$ and $\hat{M}^{(1)}$ are arranged on the diagonal of a bigger matrix, where every submatrix affects a 16-bit chunk x_i of the 64-bit state $x = (x_1||x_2||x_3||x_4)$:

$$M' \cdot x = (\hat{M}^{(0)} \cdot x_1||\hat{M}^{(1)} \cdot x_2||\hat{M}^{(1)} \cdot x_3||\hat{M}^{(0)} \cdot x_4).$$

[4] The source code is available at `https://github.com/Krypto-iaik/PRINCE_Attacks`

– **ShiftRows Operation SR**: Equal to the one in the AES cipher.
– **A bit-wise XOR with a round constant** RC_i, for $i = 0, ..., 11$.
– **A bit-wise XOR with the secret key** k_1.

In the last 5 rounds (the backward rounds), the order of operations is inverse with respect to the first 5 rounds (the forward rounds), where only the round constants differ. The middle rounds consist of three key-less operations: an S-Box layer, a matrix multiplication with M' and an inverse S-Box layer. Since the matrix M' is self-inverting (i.e. $M' = M'^{-1}$), the same linear layer M' operation is used in forward and backward rounds. Like AES, the combination of matrix multiplication and shifting provides full diffusion after only two rounds. Moreover, the varying round constants RC_i supplement the round transformation in order to prevent slide attacks. The difference between $RC_i \oplus RC_{11-i}$ is always equal to a constant value α. Since the round constants satisfy $RC_i \oplus RC_{11-i} = \alpha$ and since M' is an involution, the core cipher has the so called *α-reflection property*, i.e. the core cipher is such that the inverse of PRINCEcore parametrized with k is equal to PRINCEcore parametrized with $k \oplus \alpha$: $D_{(k_0||k'_0||k_1)}(\cdot) = E_{(k'_0||k_0||k_1 \oplus \alpha)}(\cdot)$.

For the following, we use the term "PRINCE-like cipher" to denote a cipher with middle rounds, r forward rounds and r backwards rounds, and which has the α-reflection property - examples are MANTIS [5] or QARMA [2].

Notation. In our attack, we suppose that the 64-bit state is organized as a 16×1 array of nibbles, and we use the notation $[z]$ to denote the nibble in position z (the z-th nibble is in row $r = z \bmod 4$ and in column $c = (z-r)/4$). We denote by R one round of PRINCE, while we denote i rounds by $R^{(i)}$ (without distinction between the forward and the backward direction). To simplify the notation we denote by *super-SBox* the middle rounds, by $\hat{k}$ the key of the final round and by $\tilde{k}$ the key of the first round, that is:

$$\textit{super-SBox}(\cdot) = \text{S-Box}^{-1} \circ M' \circ \text{S-Box}(\cdot), \quad \hat{k} := k_1 \oplus k'_0 \oplus \alpha, \quad \tilde{k} := k_1 \oplus k_0.$$

We attack round-reduced variants of PRINCE. In case of an even number of rounds, we keep the symmetry of the cipher.

3 Subspace Trails

Let F denote a round function in a iterative block cipher and let $V \oplus a$ denote a coset of a vector space V. Then if $F(V \oplus a) = V \oplus a$ we say that $V \oplus a$ is an *invariant coset* of the subspace V for the function F. This concept can be generalized to *trails of subspace*.

Definition 1. *Let $(V_1, V_2, ..., V_{r+1})$ a set of $r+1$ subspaces with $\dim(V_i) \leq \dim(V_{i+1})$. If for each $i = 1, ..., r+1$ and for each $a_i \in V_i^{\perp}$, there exists (unique) $a_{i+1} \in V_{i+1}^{\perp}$ such that $F(V_i \oplus a_i) \subseteq V_{i+1} \oplus a_{i+1}$, then $(V_1, V_2, ..., V_{r+1})$ is a subspace trail of length r for the function F. If the previous relation holds with equality, then the trail is called a constant-dimensional subspace trail.*

We refer to [12] for more details about the concept of subspace trails. Our treatment here is however meant to be self-contained.

In the following, we present two subspace trails for 2.5 rounds of PRINCE. The first one is composed of the middle rounds (without the final S-Box^{-1}) and 1 round before it, while the second one is composed of the middle rounds and 0.5 round after it. In particular, this second one is composed of an *invariant subspace* of the middle rounds. All the proofs of the theorems and of the propositions of this section can be found in App. A.

3.1 Subspaces of PRINCE

In this section, we define the subspaces of PRINCE, analogous to those of AES presented in [12]. For the following, let $E = \{e[0], ..., e[15]\}$ denote the unit vectors of $\mathbb{F}_{2^4}^{16}$ (e.g. e_i has a single 1 in position i). Moreover, we recall that given a generic subspace X, two different cosets $X \oplus a$ and $X \oplus b$ (i.e. $a \neq b$) are *equivalent* if and only if $a \oplus b \in X$.

For each $i = 0, ..., 3$, let $\mathcal{C}_i$ the *column subspace* of dimension 16 defined as:

$$\mathcal{C}_i = \langle e[4 \cdot i], e[4 \cdot i + 1], e[4 \cdot i + 2], e[4 \cdot i + 3] \rangle. \tag{1}$$

For instance, $\mathcal{C}_0$ correspond to matrix representation:

$$\mathcal{C}_0 = \left\{ \begin{bmatrix} x & 0 & 0 & 0 \\ z & 0 & 0 & 0 \\ w & 0 & 0 & 0 \\ y & 0 & 0 & 0 \end{bmatrix} \;\middle|\; \forall x, y, z, w \in \mathbb{F}_{2^4} \right\} \equiv \begin{bmatrix} x & 0 & 0 & 0 \\ z & 0 & 0 & 0 \\ w & 0 & 0 & 0 \\ y & 0 & 0 & 0 \end{bmatrix}.$$

For each $i = 0, ..., 3$, let $\mathcal{D}_i$ the *diagonal subspace* and $\mathcal{ID}_i$ the *inverse-diagonal subspace* - both of dimension 16 - defined as:

$$\mathcal{D}_i = SR(\mathcal{C}_i), \qquad \mathcal{ID}_i = SR^{-1}(\mathcal{C}_i) \tag{2}$$

For instance, $\mathcal{D}_0$ and $\mathcal{ID}_0$ correspond to matrix representations:

$$\mathcal{D}_0 \equiv \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & 0 & 0 & y \\ 0 & 0 & w & 0 \\ 0 & z & 0 & 0 \end{bmatrix}, \qquad \mathcal{ID}_0 \equiv \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & z & 0 & 0 \\ 0 & 0 & w & 0 \\ 0 & 0 & 0 & y \end{bmatrix}.$$

Finally, let $\mathcal{M}_i$ the *mixed subspace* and $\mathcal{IM}_i$ the *inverse-mixed subspace* - both of dimension 16 - defined as

$$\mathcal{M}_i := M'(\mathcal{D}_i), \qquad \mathcal{IM}_i := M'(\mathcal{ID}_i) \tag{3}$$

For instance, $\mathcal{M}_0$ and $\mathcal{IM}_0$ correspond to matrix representations:

$$\mathcal{M}_0 \equiv \begin{bmatrix} \alpha_3(x) & \alpha_3(z) & \alpha_0(w) & \alpha_2(y) \\ \alpha_2(x) & \alpha_2(z) & \alpha_3(w) & \alpha_1(y) \\ \alpha_1(x) & \alpha_1(z) & \alpha_2(w) & \alpha_0(y) \\ \alpha_0(x) & \alpha_0(z) & \alpha_1(w) & \alpha_3(y) \end{bmatrix}, \qquad \mathcal{IM}_0 \equiv \begin{bmatrix} \alpha_3(x) & \alpha_1(z) & \alpha_0(w) & \alpha_0(y) \\ \alpha_2(x) & \alpha_0(z) & \alpha_3(w) & \alpha_3(y) \\ \alpha_1(x) & \alpha_3(z) & \alpha_2(w) & \alpha_2(y) \\ \alpha_0(x) & \alpha_2(z) & \alpha_1(w) & \alpha_1(y) \end{bmatrix}$$

where $\alpha_i(\cdot)$ are defined as

$$\alpha_i(x) = x \wedge (0\text{x}2^i \oplus 0\text{x}f), \tag{4}$$

and where $\wedge$ is the *and (logic) operator*.

Let $I \subseteq \{0,1,2,3\}$. Subspaces $\mathcal{C}_I$, $\mathcal{D}_I$, $\mathcal{ID}_I$, $\mathcal{M}_I$ and $\mathcal{IM}_I$ are defined as:

$$\mathcal{C}_I = \bigoplus_{i\in I} \mathcal{C}_i, \quad \mathcal{D}_I = \bigoplus_{i\in I} \mathcal{D}_i, \quad \mathcal{ID}_I = \bigoplus_{i\in I} \mathcal{ID}_i, \quad \mathcal{M}_I = \bigoplus_{i\in I} \mathcal{M}_i, \quad \mathcal{IM}_I = \bigoplus_{i\in I} \mathcal{IM}_i.$$

Note that $\mathcal{C}_I$ is an *invariant subspace* for the middle-rounds of PRINCE, that is for each $a \in \mathcal{C}_I^{\perp}$, there exists unique $b \in \mathcal{C}_I^{\perp}$ such that

$$\text{S-Box}^{-1} \circ M' \circ \text{S-Box}(\mathcal{C}_I \oplus a) = \mathcal{C}_I \oplus b. \tag{5}$$

As noticed in [22] and in [9], the middle rounds of PRINCE have 2^{32} *fixed points* (x is a fixed point of a function f iff $f(x) = x$). In particular, in [9] authors showed that if a plaintext of PRINCE "corresponds" to a fixed point of the middle rounds, then the encryption scheme is much simplified, since the 4 center rounds - minus the first and the last key addition - become a simple S-Box layer. However, no key-recovery attack has been showed: due to the presence of the secret key, it is not possible to choose a priori the plaintexts in order to satisfy the previous requirement. In our case, we show how to exploit the invariant subspace (that is, set of points instead of a single point) in order to mount powerful attacks on reduced-round PRINCE.

3.2 Subspace Trails of PRINCE

In the following, we present several subspace trails for round-reduced PRINCE.

Subspace Trail for 1+1.5 rounds of PRINCE. Let $R^{(1+1.5)}(\cdot)$ defined as:

$$R^{(1+1.5)}(\cdot) := M' \circ \text{S-Box} \circ R \circ ARK(\cdot), \tag{6}$$

i.e. the middle rounds without the final S-Box (denoted by "1.5") and the previous round (denoted by "1").

Theorem 1. *Let $I \subseteq \{0,1,2,3\}$. For each $a \in \mathcal{C}_I^{\perp}$, there exists unique $b \in \mathcal{M}_I^{\perp}$ such that $R^{(1+1.5)}(\mathcal{C}_I \oplus a) = \mathcal{M}_I \oplus b$, where b depends on a and on the secret key. Equivalently:*

$$\text{Prob}(R^{(1+1.5)}(x) \oplus R^{(1+1.5)}(y) \in \mathcal{M}_I \,|\, x \oplus y \in \mathcal{C}_I) = 1. \tag{7}$$

This means that a coset of $\mathcal{C}_I$ is mapped into a coset of $\mathcal{M}_I$ after 2.5 rounds:

$$\mathcal{C}_I \oplus a \xrightarrow{R\circ ARK(\cdot)} \mathcal{D}_I \oplus b \xrightarrow{M'\circ \text{S-Box}(\cdot)} \mathcal{M}_I \oplus c.$$

Thus, a subspace trail for 1+1.5 rounds of PRINCE is composed by the subspaces $\{\mathcal{C}_I, \mathcal{D}_I, \mathcal{M}_I\}$. Since S-Box($\mathcal{M}_I$) is mapped into a subspace of dimension 64 (that is all the space), it is not possible to extend the found subspace trail anymore. Moreover, observe that if X is a generic subspace, $X \oplus a$ is a coset of X and if x and y are two elements of the (same) coset $X \oplus a$, then $x \oplus y \in X$. This justifies the probability (7).

Subspace Trail for 2+0.5 rounds of PRINCE. Let $R^{(2+0.5)}(\cdot)$ defined as:

$$R^{(2+0.5)}(\cdot) := M' \circ SR^{-1} \circ ARK \circ \textit{super-SBox} \circ ARK(\cdot), \tag{8}$$

i.e. the middle rounds ("2") and the linear part of the next round ("0.5").

Theorem 2. *Let $I \subseteq \{0,1,2,3\}$. For each $a \in \mathcal{C}_I^{\perp}$, there exists unique $b \in \mathcal{IM}_I^{\perp}$ such that $R^{(2+0.5)}(\mathcal{C}_I \oplus a) = \mathcal{IM}_I \oplus b$, where b depends on a and on the secret key. Equivalently:*

$$\text{Prob}(R^{(2+0.5)}(x) \oplus R^{(2+0.5)}(y) \in \mathcal{IM}_I \,|\, x \oplus y \in \mathcal{C}_I) = 1. \tag{9}$$

This means that a coset of $\mathcal{C}_I$ is mapped into a coset of $\mathcal{IM}_I$ after 2.5 rounds:

$$\mathcal{C}_I \oplus a \xrightarrow{\textit{super-SBox} \circ ARK(\cdot)} \mathcal{C}_I \oplus b \xrightarrow{M' \circ SR^{-1} \circ ARK(\cdot)} \mathcal{IM}_I \oplus c.$$

Thus, a subspace trail for 2+0.5 rounds of PRINCE is composed by the subspaces $\{\mathcal{C}_I, \mathcal{IM}_I\}$.

Impossible Subspace Trail for PRINCE. The previous subspace trails hold with probability 1. It's also possible to describe a subspace trail with probability 0 (an "impossible subspace trail") for 4.5 rounds of PRINCE, and to set up a known-plaintext impossible differential attack on 6 rounds of PRINCE. More details are given in App. B.

4 An "Equivalent" Representation of PRINCE

In this section, we present an "*equivalent*" representation of PRINCE from the point of view of the security. The PRINCEcore round is very similar to the AES round. The major difference between them is that in a PRINCE round the MixLayer operation is performed before the ShiftRows operation, while in an AES round is the opposite.

In order to better understand the PRINCE algorithm, we evaluate the security of a version of PRINCE - called in the following PRINCE′ - where these two linear operations are exchanged in position, both in the forward and in the backward rounds. First of all, in this case it is possible to set up a subspace trail for 3.5 rounds of PRINCE′ (i.e. one more round than original PRINCE):

$$\mathcal{ID}_I \oplus a \xrightarrow{R \circ ARK(\cdot)} \mathcal{C}_I \oplus b \xrightarrow{\textit{super-SBox}(\cdot)} \mathcal{C}_I \oplus c \xrightarrow{M' \circ SR^{-1} \circ ARK(\cdot)} \mathcal{IM}_I \oplus d, \tag{10}$$

where $I \subseteq \{0,1,2,3\}$, using the property that $\mathcal{C}_I$ is an invariant subspace for the middle rounds. The proof follows immediately by the definition of the subspaces and by the order of the ShiftRows and of the MixLayer operations.

Also due to the following cryptanalysis of PRINCE′ against the most popular attacks present in literature, we can conclude that this version of PRINCE is weaker than the original one (as the designers, we don't consider the related key attacks for this security analysis):

– *Differential/Linear Cryptanalysis:* For the original PRINCE, *"any differential characteristic and any linear-trail over 4 consecutive rounds of PRINCE has at least 16 active S-Boxes"* (see App. C of [7] for more details). For the modified version PRINCE′ and using the same argumentation given in [7], the number of active S-Boxes over 4 consecutive rounds is at least 12 instead of 16.
– *Square Attack:* For the original PRINCE, the balanced property holds for 4.5 rounds (2 forward rounds + middle rounds + 1 backward rounds) starting with three input active nibbles which lie on the same column (see [19] for more details). For PRINCE′, the balanced property holds for 5.5 rounds (2 forward rounds + middle rounds + 2 backward rounds) starting with a single input active nibble (result practical verified).
– *Meet-in-the-Middle Attack:* The Meet-in-the-Middle Attacks presented in [9] are not influenced by the positions of the MixLayer and of the ShiftRows operations, that is there are analogous meet-in-the-middle attacks for this modified version similar to the ones presented for the original PRINCE.

As a consequence, the position of the ShiftRows and of the MixLayer operations influences the security of this encryption scheme.

A version of PRINCE - called in the following PRINCE$^\sharp$ - with the same security of the original one and where ShiftRows and MixLayer operations are ordered as in AES can be obtained by changing the original *middle-rounds* of PRINCE with the following one:

$$\textit{middle-rounds}(x) = \text{S-Box}^{-1} \circ SR^{-1} \circ M' \circ SR \circ \text{S-Box}(x), \tag{11}$$

and with some more slight modifications, as we show in details in the following. By definition of PRINCE:

$$p \xrightarrow{ARK(\cdot)} \xrightarrow{ARK \circ M' \circ SR \circ \text{S-Box}(\cdot)} \ldots \xrightarrow{ARK \circ M' \circ SR \circ \text{S-Box}(\cdot)} \underbrace{\xrightarrow{\text{S-Box}^{-1} \circ M' \circ \text{S-Box}(\cdot)}}_{\textit{middle-rounds}}$$

$$\xrightarrow{\text{S-Box}^{-1} \circ SR^{-1} \circ M' \circ ARK(\cdot)} \ldots \xrightarrow{\text{S-Box}^{-1} \circ SR^{-1} \circ M' \circ ARK(\cdot)} \xrightarrow{ARK(\cdot)} c.$$

Exchanging S-Box, ARK and SR operations, and applying a SR^{-1} operation on the plaintext and a SR on the ciphertext, one obtains:

$$p' \xrightarrow{ARK'(\cdot)} \xrightarrow{ARK' \circ SR \circ M' \circ \text{S-Box}(\cdot)} \ldots \xrightarrow{ARK' \circ SR \circ M' \circ \text{S-Box}(\cdot)} \underbrace{\xrightarrow{\text{S-Box}^{-1} \circ SR \circ M' \circ SR^{-1} \circ \text{S-Box}(\cdot)}}_{\textit{middle-rounds}}$$

$$\xrightarrow{\text{S-Box}^{-1} \circ M' \circ SR^{-1} \circ ARK''(\cdot)} \ldots \xrightarrow{\text{S-Box}^{-1} \circ M' \circ SR^{-1} \circ ARK''(\cdot)} \xrightarrow{ARK''(\cdot)} c',$$

where $ARK'(\cdot) := \cdot \oplus SR^{-1}(k)$, $ARK''(\cdot) := \cdot \oplus SR(k)$, $p' = SR^{-1}(p)$ and $c' = SR(c)$. PRINCE$^\sharp$ is an equivalent representation of PRINCE, where ShiftRows operation is performed before the MixColumns one (as in AES) and where the *super-SBox* operation is a little modified, for the cost of 2 additional ShiftRows operations. With respect to the original PRINCE, a (slight) different key schedule is used and SR^{-1} (respectively SR) is applied on p (respectively on c).

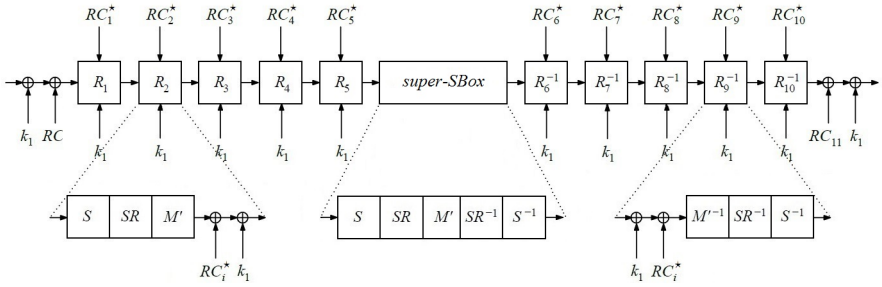

Fig. 2. A scheme of the PRINCEcore of the cipher PRINCE*, analogous (but not completely equivalent - see the text for details) to the original PRINCE.

In this equivalent representation, a ShiftRows operation (respectively Inverse-SR) is applied to the key in the forward rounds (respectively backwards). Thus, we finally define another version of PRINCE - called in the following PRINCE* - depicted in Fig. 2, which is analogous to the original PRINCE but not completely equivalent, since *no* ShiftRows operation (respectively Inverse-SR) is applied to the key in the forward rounds (respectively backward). Thus, we claim that PRINCE* has the same security of the original PRINCE. Note that the order of ShiftRows and MixLayer operations of PRINCE* is the same of AES. In App. C, we present a related key attack that exploits this equivalent version of PRINCE.

Conclusion and AES-like Ciphers. Our analysis can be applied in a natural way to other PRINCE-like ciphers, as for example the MANTIS encryption scheme [5] - see App. D for details and [10], where a similar but independent analysis with analogous results and conclusion has been proposed - or the QARMA block cipher family [2].

By our analysis, *the order of the MixLayer and the ShiftRows operations influences the security of PRINCE-like ciphers.* Thus, it seems advisable to use only one of the two following options for future designs of PRINCE-like schemes:

- the middle-rounds as in the original PRINCE cipher (that is, S-Box^{-1} $\circ$ $M'\circ$ S-Box$(\cdot)$) and the MixLayer computed before (respectively after) the ShiftRows in the forward (respectively backward) rounds;
- the middle rounds as in the PRINCE* cipher (that is, S-Box$^{-1}\circ SR^{-1}\circ M'\circ$ $SR\circ$ S-Box$(\cdot)$) and the ShiftRows computed before (respectively after) the MixLayer in the forward (respectively backward) rounds.

Finally, we emphasize that this analysis holds due to the particular structure of PRINCE-like cipher. In particular, consider key-recovery attacks that are independent of the key-schedule[5], excluding related-key attacks. For an AES-like cipher (with r identical rounds and without middle rounds), the position of the

[5] We observe that attacks that exploit the key-schedule can be affected by the order of the linear operations. To better highlight this fact, we refer to the analysis done in [11]

ShiftRows operation with respect to the MixLayer one does not influence the security. Indeed, consider the AES encryption scheme (where the final MixColumns operation can also be omitted):

$$p \xrightarrow{ARK} \xrightarrow{ARK \circ MC \circ SR \circ \text{S-Box}(\cdot)} \ldots \xrightarrow{ARK \circ MC \circ SR \circ \text{S-Box}(\cdot)} c.$$

Changing the position of SR, ARK and S-Box, and applying a ShiftRows operation to the ciphertexts (note that this is a linear operation, so it doesn't influence the security of the encryption scheme), one obtains:

$$SR(p) \xrightarrow{ARK''} \xrightarrow{ARK'' \circ SR \circ MC \circ \text{S-Box}(\cdot)} \ldots \xrightarrow{ARK'' \circ SR \circ MC \circ \text{S-Box}(\cdot)} c,$$

where $ARK''(\cdot) = \cdot \oplus SR(k)$ and k is the secret key. It follows that an equivalent version of AES - called for consistency AES* - defined as

$$p \xrightarrow{ARK} \xrightarrow{ARK \circ SR \circ MC \circ \text{S-Box}(\cdot)} \ldots \xrightarrow{ARK \circ SR \circ MC \circ \text{S-Box}(\cdot)} c,$$

where ShiftRows operation is computed after the MixColumns one, has the same security of the original version.

5 Truncated Differential Attack on 4 Rounds of PRINCE

Using the first 2.5 rounds subspace trail presented in previous section, it is possible to set up an attack on 3 rounds of PRINCE:

$$p \xrightarrow{R(\cdot)} q \xrightarrow{M' \circ \text{ S-Box}(\cdot)} s \xrightarrow{\text{S-Box}^{-1}(\cdot)} c,$$

where the plaintexts are chosen in the same coset of $\mathcal{C}_I$, and the states s belong in the same coset of $\mathcal{M}_I$. Briefly, given a pair of ciphertexts (c^1, c^2), the idea is to find the final key using the condition S-Box$(c^1 \oplus \hat{k}) \oplus$ S-Box$(c^2 \oplus \hat{k}) \in \mathcal{M}_I$. When the full key $\hat{k}$ has been found, to find k_1 the idea is to use plaintexts in the same coset of $\mathcal{D}_I$, to decrypt the corresponding ciphertexts and to find the key k_1 using the condition that S-Box$(q^1 \oplus k_1) \oplus$ S-Box$(q^2 \oplus k_1) \in SR(\mathcal{M}_I)$, where $q := \textit{super-SBox}^{-1}(c) \oplus R_1$. The attack is presented in details in App. E, and here we focus on the attack on 4 rounds of PRINCE (giving all the details).

To attack 4 rounds, a possibility is to extend the attack on 3 rounds (the middle rounds and one round before) at the end. Consider four rounds of PRINCE:

$$p \xrightarrow{R(\cdot)} \hat{p} \xrightarrow{M' \circ \text{ S-Box}(\cdot)} q \xrightarrow{\text{S-Box}^{-1}(\cdot)} s \xrightarrow{R^{-1}(\cdot)} c,$$

about the effect of the omission of the final MixColumns operation. While in general key-recovery attacks are not influenced by the presence of the last MixColumns operation, some of the attacks that exploit it (e.g. Meet-in-the-Middle attacks) are affected, since a different key schedule can affect the amount of key material that has to be guessed in key-recovery attacks (also in the standard single-key model). In a similar way, the same analysis holds also when the positions of the MixColumns and ShiftRows operations are exchanged.

where $p \in \mathcal{C}_I \oplus a$ (for $a \in \mathcal{C}_I^{\perp}$ fixed). Given a pair of plaintexts/ciphertexts (where the plaintexts belong to the same coset of $\mathcal{C}_I$), the idea is simply to guess the key of the final round, to decrypt partially one round, and to find the key of the third round such that the two corresponding texts belong to the same coset of $\mathcal{M}_I$. That is, if $\hat{k}$ is a candidate of the final key (as we show in the following, the attacker must test all the possibilities) and if $R_{\hat{k}}(\cdot)$ denotes the final round with key $\hat{k}$, given a pair (c^1, c^2) the right key k_0 must satisfy the condition:

$$\text{S-Box}(R_{\hat{k}}(c^1) \oplus k_1 \oplus RC_1 \oplus \alpha) \oplus \text{S-Box}(R_{\hat{k}}(c^2) \oplus k_1 \oplus RC_1 \oplus \alpha) \in \mathcal{M}_I.$$

Candidates $\hat{k}$ and k_1 of the key must be tested checking if this condition is satisfied for other pairs of ciphertexts. To find them, the idea is to work independently on each column of $\mathcal{M}_I$. For the following, we limit to the case $I = \{0\}$.

First Step of the Attack. Let c^1 and c^2 two ciphertexts such that the corresponding plaintexts belong to the same coset of $\mathcal{C}_0$, that is $p^1 \oplus p^2 \in \mathcal{C}_0$. As for the attack on 3 rounds, the idea is to work independently on each column of the key, due to the fact that the columns of $\mathcal{M}_0$ depend on different and independent variables. Initially the attacker guesses 1 column (that is 4 nibbles) of the final key, as for example $\hat{k}[0], \hat{k}[1], \hat{k}[2]$ and $\hat{k}[3]$, and she uses them to partially decrypt c^1 and c^2, that is she computes 4 nibbles of $s^1 := R_{\hat{k}}(c^1)$ and of $s^2 := R_{\hat{k}}(c^2)$. Note that the attacker cannot guess 4 arbitrary nibbles of the final key but an entire column, since she has to compute the Linear Layer M'. Moreover, since the attacker can not impose any restriction/condition on the final key, she has to repeat the next steps for each values of these four nibbles of the final key, which are $(2^4)^4 = 2^{16}$ possible values in total.

Due to the ShiftRows operation, after one-round of decryption these four nibble belong to different column. To find four nibbles of k_0, the attacker must work independently on each nibble (note that since they lie on different columns, no relationship holds among them, due to the definition of $\mathcal{M}_I$). For example, using the definition of $\mathcal{M}_0$, the nibble $k_1[0]$ has to satisfy the condition:

$$(\text{S-Box}(s^1[0] \oplus k_1[0] \oplus RC_2[0]) \oplus \text{S-Box}(s^2[0] \oplus k_1[0] \oplus RC_2[0])) \wedge 0\text{x}8 = 0, \quad (12)$$

and similar conditions hold for the nibbles $k_1[7], k_1[10]$ and $k_1[13]$. To find these 4 nibbles, the attacker needs at least four different pairs of chosen ciphertexts (each of these conditions involves only one bit - it is satisfied with prob. 2^{-1}).

Since these found 4 nibbles of k_1 (which are $k_1[0], k_1[7], k_1[10], k_1[13]$) depend on the 4 guessed nibbles of $\hat{k}$ (which are $\hat{k}[0], ..., \hat{k}[3]$), for each combination of the first column of $\hat{k}$, the attacker finds on average one combination of the 4 nibbles of k_1, that is 2^{16} in total. To discover the right combination, the attacker has to test these values using other pairs of ciphertexts, that is given other pairs of ciphertexts $(c^\star, c')$ she has to check if the corresponding texts $(q^\star, q')$ - where $q := \text{S-Box}(s \oplus k_1 \oplus RC_2)$ - belong to the same coset of $\mathcal{M}_0$. Since each condition involves one bit and since there are 2^{16} combinations for the 4 nibbles of $\hat{k}$ and k_1, she needs at least other four different pairs to check the found values (since

$2^{16} \times (2^{-4})^4 = 1$). Thus the attacker needs at least eight different pairs for this first step. To save memory, a good idea is to check immediately the found values of $\hat{k}$ and k_1 with other pairs of ciphertexts: in this way, the attacker doesn't need to store anything.

Second Step of the Attack. When the attacker has found one nibble for each column of k_1, the idea is to use the relationships that hold among the nibbles of the same column to discover the other nibbles of the key much faster than working on each nibble independently by the others.

As before, the attacker guesses one column (e.g. the second one) of $\hat{k}$, and decrypts partially the pair of ciphertexts. In order to find other 4 nibbles of k_1 (one per column), the idea is to use the relationships that hold among the nibbles of the same column given in Theorem 7 - App. E, and not to work independently on each nibble. For example, the attacker can find the nibble of the first column $k_1[1]$ using the relationship:

$$\begin{aligned}&(\text{S-Box}(s^1[0] \oplus k_1[0] \oplus RC_2[0]) \oplus \text{S-Box}(s^2[0] \oplus k_1[0] \oplus RC_2[0])) \wedge 0\text{x}b = \\ =&(\text{S-Box}(s^1[1] \oplus k_1[1] \oplus RC_2[1]) \oplus \text{S-Box}(s^2[1] \oplus k_1[1] \oplus RC_2[1])) \wedge 0\text{x}7,\end{aligned}$$

where the left part of the equation is known ($k_1[0]$ is already known). Analogous relationships hold for the other nibbles and for the other columns. Observe that since these relationships involve more than one bit, in this second step the attacker needs a lower number of pairs of ciphertexts to discover the right key.

As before, the attacker has to repeat the step for each possible values of the second column of $\hat{k}$, and to test the found values against other plaintexts/ciphertexts pairs in order to eliminate wrong candidates (remember that she finds on average one candidate of four nibbles of k_1 for each of the 2^{16} guess values of $\hat{k}$). The same procedure is used for the third and for the fourth columns of $\hat{k}$. For example, for each guess value of the third column of $\hat{k}$, the relationships that the nibble $k_1[2]$ have to satisfy are $\tilde{s}[2] \wedge 0\text{x}b = \tilde{s}[1] \wedge 0\text{x}d$ and $\tilde{s}[2] \wedge 0\text{x}4 = \tilde{s}[0] \wedge 0\text{x}4$, where $\tilde{s}[i] := \text{S-Box}(s^1[i] \oplus k_1[i] \oplus RC_2[i]) \oplus \text{S-Box}(s^2[i] \oplus k_1[i] \oplus RC_2[i])$.

For completeness, note that also in this second step the attacker can work independently on each nibble, but the total computational cost would be higher.

Estimation of the Data Complexity. Our implementation shows that if only eight pairs of ciphertexts are used for the first step, then some false positive candidates of the key pass the test. To avoid this problem, one possibility is to use more pairs of ciphertexts, making the filter stronger[6].

In the following, we first try to give a theoretical estimation of the number of pairs necessary to eliminate almost all the false positive candidates of the key. In the first step of the attack, the problem of the false positives arises when $R_{\hat{k}}(c^1)[i] = R_{\hat{k}}(c^2)[i]$ for a certain $i = 0, 7, 10, 13$. For example, note that if $R_{\hat{k}}(c^1)[0] = R_{\hat{k}}(c^2)[0]$ in eq. (12), then each possible value of $k_1[0]$ passes the

[6] We emphasize that the right key is always found. We use more plaintexts only to discard false positives that pass the test.

test. Thus, a first estimation can be done by calculating the minimum number of pairs such that there exist at least eight pairs with 4 different nibbles.

Before to continue, an important observation has to be done. Given n texts, it is possible to construct $n \cdot (n-1)/2$ different pairs, but actually only $n-1$ pairs are useful for the attack. Consider for example three texts t^1, t^2, t^3 and the corresponding pairs (t^1, t^2), (t^1, t^3), (t^2, t^3). If $k_1[0]$ satisfies the condition (12) for the pairs $(t^1[0], t^2[0])$ and $(t^1[0], t^3[0])$, then it automatically satisfies this condition also for the pair $(t^2[0], t^3[0])$[7]. Thus, only two pairs are really useful for the attack. More generally, given n texts, only $n-1$ pairs are useful for the attack - for the following we suppose that one text is in common for all the pairs. As shown in details in App. F, the probability that only the right key is found using 9 chosen plaintexts is about $(0.0604)^4$. By calculation (we refer to App. F for more details), the attacker needs at least 16 plaintexts in order to have a good probability of success, which is approximately 73%.

Actually, this is only a rough approximation, since an important aspect is not taken into account. For each guess of the first column of $\hat{k}$, the attacker is able to find 4 nibbles of k_1. Then, she checks these candidates of the keys using other texts. Note that it is sufficient that one nibble of k_1 doesn't pass this test to conclude that the guess value of $\hat{k}$ is wrong, independently by the other three nibbles of k_1. Moreover, it is also possible that wrong key candidates found at the first step are detected and eliminated in the second step of the attack. Thus, as our implementation shows, a lower number of texts (with respect to that predicted by our theoretical model) turns out to be sufficient for the attack. In particular, we found that 12 chosen plaintexts (instead of 16) - i.e. 11 pairs - are sufficient with (very) high probability. Working in a similar way, it turns out that only 6 chosen plaintexts - i.e. 5 pairs - are sufficient for the second step.

The number of plaintexts chosen for the first step (i.e. 12) allows to eliminate almost all the false candidates of the key. For the second step of the attack, a lower number of plaitexts (i.e. 6) is sufficient to recover the entire key. As a consequence, in the second step the attacker doesn't use and wastes a lot of information about the (chosen) plaintexts/ciphertexts pairs.

To improve the attack, the idea is to reduce the total number of chosen plaintexts used for the attack, and to use all the available plaintexts/ciphertexts pairs also in the second step. That is, the idea is to reduce the number of chosen plaintexts used for the first step and to increase this number for the second step - we assume that these two numbers are equal. As a consequence, in the first step of the attack more false positive candidates pass the test, but they are soon detected in the second step, thanks to the higher number of pairs used. Moreover, note that in the second step the false candidates are detected much faster than in the first one, since the probability that the relationships are satisfied is much lower (remember that they involve an higher number of bits).

Our implementation shows that 8 chosen plaintexts (i.e. 7 pairs) are sufficient for this mode of the attack (note that we use 8 chosen plaintexts both in

[7] Note that: $[\text{S-Box}(t^2[0] \oplus k_1[0]) \oplus \text{S-Box}(t^3[0] \oplus k_1[0])] \wedge 0\text{x}8 = [\text{S-Box}(t^2[0] \oplus k_1[0]) \oplus \oplus \text{S-Box}(t^1[0] \oplus k_1[0]) \oplus \text{S-Box}(t^1[0] \oplus k_1[0]) \oplus \text{S-Box}(t^3[0] \oplus k_1[0])] \wedge 0\text{x}8 = 0$.

Data: 7 ciphertexts pairs (c^1, c^i) where $i = 2, ..., 8$, whose corresponding plaintexts belong in the same coset of $\mathcal{C}_0$
Result: Secret Key $\hat{k}$ and k_1.
for *all* 2^{16} *possible combinations of* $(\hat{k}[0], \hat{k}[1], \hat{k}[2], \hat{k}[3])$ **do**
  decrypt one round: $s^i[j] = \text{S-Box}(c^i \oplus \hat{k}[j])$ $\forall i = 1, ..., 8$ and $\forall j = 0, ..., 3$;
  for $k_1[0]$ from 0 *to* $2^4 - 1$ **do**
    check if for each possible pairs (s^1, s^i) where $i = 2, ..., 8$:
    $[\text{S-Box}(s^1[0] \oplus k_1[0] \oplus RC_2[0]) \oplus \text{S-Box}(s^i[0] \oplus k_1[0] \oplus RC_2[0])] \wedge 0\text{x}8 = 0$;
    If not satisfied, **then** next value (i.e. next $k_1[0]$ or/and $(\hat{k}[0], ..., \hat{k}[3])$);
    else
      identify candidates for $k_1[0]$ and $(\hat{k}[0], ..., \hat{k}[3])$;
      use these candidates of the first column of $\hat{k}$, to find candidates of $k_1[7], k_1[10], k_1[13]$ - use the same algorithm described for $k_1[0]$ and work independently on each nibble;
    end
  end
end
for *all candidates of* $k_1[0], k_1[7], k_1[10], k_1[13]$ *and* $(\hat{k}[0], ..., \hat{k}[3])$ **do**
  for *all* 2^{16} *possible combinations of* $(\hat{k}[4], \hat{k}[5], \hat{k}[6], \hat{k}[7])$ **do**
    decrypt one round: $s^i[j] = \text{S-Box}(c^i \oplus \hat{k}[j])$ $\forall i = 1, ..., 8$ and $\forall j = 4, ..., 7$;
    for $k_1[1]$ from 0 *to* $2^4 - 1$ **do**
      check if for each possible pairs (s^1, s^i) where $i = 2, ..., 8$:
      $[\text{S-Box}(s^1[0] \oplus k_1[0] \oplus RC_2[0]) \oplus \text{S-Box}(s^i[0] \oplus k_1[0] \oplus RC_2[0])) \wedge 0\text{x}b =$
      $[\text{S-Box}(s^1[1] \oplus k_1[1] \oplus RC_2[1]) \oplus \text{S-Box}(s^i[1] \oplus k_1[1] \oplus RC_2[1])] \wedge 0\text{x}7$;
      If not satisfied, **then** next value (i.e. next $k_1[1]$ or/and $(\hat{k}[4], ..., \hat{k}[7])$ or/and $k_1[0]$ or/and $(\hat{k}[0], ..., \hat{k}[3])$);
      else
        identify candidates for $k_1[1]$ and $(\hat{k}[4], ..., \hat{k}[7])$;
        use these candidates of the second column of $\hat{k}$, to find candidates of $k_1[4], k_1[11], k_1[14]$ - use the same algorithm described for $k_1[1]$ and exploit the relationships among the nibbles;
      end
    end
  end
end
Repeat this second step for the third and for the fourth column of $\hat{k}$;
return *Secret Key* $\hat{k}$ *and* k_1.

Algorithm 1: *Truncated differential attack on 4 rounds of PRINCE - extension at the end.* For simplicity, this pseudo-code is not completely optimized as described in the text.

the first step and in the second one), and that the total computational cost is approximately unchanged (with respect to the previous mode).

Estimation of the Computational Cost. The computational cost of the first step can be estimated as follows. Given 5 chosen ciphertexts, the computational

cost to calculate 4 nibbles of $R^{-1}(c)$ is $8 \times 4 = 2^5$ S-Box look-ups. As shown in detail in App. E.1, the cost to find one nibble of k_1 working independently on each nibble is $2^{5.46}$ S-Box look-ups (thus for 4 nibbles, the cost is $4 \times 2^{5.46} = 2^{7.46}$). The cost to check candidates of $\hat{k}$ and of k_1 against other pairs of ciphertexts can be estimated by $4 \times 2 \times 8 = 2^6$ S-Box look-ups. Thus, to find the right combination and to do the requested check, the total computational cost for the first step is well approximated by 2^{16} (possible values) $\times(4 \cdot 2^{5.46} + 8 \times 4 + 2^6) \simeq 2^{24.1}$ S-Box looks ups.

Some optimizations allow to improve the computational cost of the attack. For example, for each guessed value of $\hat{k}$, the attacker should focus on a single nibble of k_1, e.g. $k_1[0]$. In this way, it is possible to eliminate wrong candidates simply checking the found candidates of $k_1[0]$ and of the column of $\hat{k}$ against all the available pairs of texts before to work on the other three nibbles of k_1. It follows that it is sufficient to consider only these survived combinations of the first column of $\hat{k}$ (instead of all the 2^{16} possible values) in order to find the other 3 nibbles of k_1. The computational cost to find the remaining 3 nibbles of k_1 becomes negligible compared to the cost to find $k_1[0]$, and the total computational cost can be approximated by $2^{16} \times (2^{5.46} + 2^5 + 2^6) \simeq 2^{23.1}$ S-Box look ups.

The computational cost for the second step can be computed in a similar way. As shown in an analogous case in App. E.2, the cost to find one nibble of k_1 is of $2^{4.6}$ S-Box look-ups (given another nibble of the same column and exploiting the relationship among the nibbles). Thus, the total cost for this step can be approximated by 3 (columns) $\times 2^{16} \times (2^{4.6}$ (cost of the subspace attack - single equivalence) $+ 4 \times 8$ (check) $+ 2^5$ (partial decryption) $) \simeq 2^{23.9}$ S-Box look-ups, using the same optimizations as before.

The total computational cost can be approximated by $2^{23.9} + 2^{23.1} \simeq 2^{24.25}$ S-Box look-ups, that is $2^{18.25}$ four-rounds encryption, and the attacker needs only 8 different chosen plaintexts (that belong to the same coset of $\mathcal{C}_0$).

Acknowledgements. We thank Navid G. Bardeh for pointing out an error in the C/C++ implementation of PRINCE block cipher. The work in this paper has been partially supported by the Austrian Science Fund (project P26494-N15).

References

1. F. Abed, E. List, and S. Lucks, "On the Security of the Core of PRINCE Against Biclique and Differential Cryptanalysis," Cryptology ePrint Archive, Report 2016/712, 2016.
2. R. Avanzi, "The QARMA Block Cipher Family – Almost MDS Matrices Over Rings With Zero Divisors, Nearly Symmetric Even-Mansour Constructions With Non-Involutory Central Rounds, and Search Heuristics for Low-Latency S-Boxes," Cryptology ePrint Archive, Report 2016/444, 2016.
3. S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni, "Midori: A Block Cipher for Low Energy," in *Advances in Cryptology - ASIACRYPT 2015: 21st International Conference on the Theory and Application*

of Cryptology and Information Security, New Zealand. Proceedings, Part II, ser. LNCS, vol. 9453, 2015, pp. 411–436.
4. A. Bay, O. Ersoy, and F. Karakoç, "Universal Forgery and Key Recovery Attacks on ELmD Authenticated Encryption Algorithm," Cryptology ePrint Archive, Report 2016/640, 2016, To appear at Asiacrypt 2016.
5. C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. M. Sim, "The SKINNY family of block ciphers and its low-latency variant MANTIS," in *Advances in Cryptology - CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA. Part II*, ser. LNCS, vol. 9815, 2016, pp. 123–153.
6. E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, 1991.
7. J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalçin, "PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract," in *Advances in Cryptology - ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security. Proceedings*, ser. LNCS, 2012, vol. 7658, pp. 208–225.
8. C. Bouillaguet, P. Derbez, O. Dunkelman, P. Fouque, N. Keller, and V. Rijmen, "Low-Data Complexity Attacks on AES," *IEEE Trans. Information Theory*, vol. 58, no. 11, pp. 7002–7017, 2012.
9. P. Derbez and L. Perrin, "Meet-in-the-Middle Attacks and Structural Analysis of Round-Reduced PRINCE," in *Fast Software Encryption - FSE 2015: 22nd International Workshop, Turkey. Revised Selected Papers*, ser. LNCS, 2015, vol. 9054, pp. 190–216.
10. C. Dobraunig, M. Eichlseder, and F. Mendel, "Key recovery for MANTIS-5," Cryptology ePrint Archive, Report 2016/754, 2016.
11. O. Dunkelman and N. Keller, "The Effects of the Omission of Last Round's MixColumns on AES," *Inf. Process. Lett.*, vol. 110, no. 8-9, pp. 304–308, Apr. 2010.
12. L. Grassi, C. Rechberger, and S. Rønjom, "Subspace trail cryptanalysis and its applications to AES," Cryptology ePrint Archive, Report 2016/592, 2016.
13. J. Jean, I. Nikolic, T. Peyrin, L. Wang, and S. Wu, "Security Analysis of PRINCE," in *Fast Software Encryption - FSE 2013: 20th International Workshop, Singapore. Revised Selected Papers*, ser. LNCS, vol. 8424, 2013, pp. 92–111.
14. J. Kilian and P. Rogaway, "How to Protect DES Against Exhaustive Key Search," in *Advances in Cryptology - CRYPTO 1996: 16th Annual Cryptology Conference, Santa Barbara, CA, USA. Proceedings*, ser. LNCS, 1996, vol. 1109, pp. 252–267.
15. L. R. Knudsen, "Truncated and higher order differentials," in *Fast Software Encryption - FSE 1994: 2nd International Workshop, Belgium. Revised Selected Papers*, ser. LNCS, 1995, vol. 1008, pp. 196–211.
16. G. Leander, M. A. Abdelraheem, H. AlKhzaimi, and E. Zenner, "A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack," in *Advances in Cryptology – CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, 2011. Proceedings*, ser. LNCS, vol. 6841, 2011, pp. 206–221.
17. G. Leander, B. Minaud, and S. Rønjom, "A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro," in *Advances in Cryptology - EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Bulgaria. Proceedings, Part I*, ser. LNCS, vol. 9056, 2015, pp. 254–283.
18. P. Morawiecki, "Practical Attacks on the Round-reduced PRINCE," Cryptology ePrint Archive, Report 2016/245, 2016.

19. R. Posteuca and G. Negara, "Integral Cryptanalysis of Round-Reduced PRINCE Cipher," in *Proceedings of the Romanian Academy, Series A, Volume 16*, 2015, pp. 265–270.
20. H. Raddum and S. Rasoolzadeh, "Faster Key Recovery Attack on Round-Reduced PRINCE," Cryptology ePrint Archive, Report 2016/828, 2016, To appear at LightSec 2016.
21. S. Rønjom, "Invariant subspaces in Simpira," Cryptology ePrint Archive, Report 2016/248, 2016.
22. H. Soleimany, C. Blondeau, X. Yu, W. Wu, K. Nyberg, H. Zhang, L. Zhang, and Y. Wang, "Reflection Cryptanalysis of PRINCE-Like Ciphers," *Journal of Cryptology*, vol. 28, no. 3, pp. 718–744, 2013.
23. G. Zhao, B. Sun, C. Li, and J. Su, "Truncated differential cryptanalysis of PRINCE," *Security and Communication Networks*, vol. 8, no. 16, pp. 2875–2887, 2015.

A Proofs of Sect. 3

Proposition 1. *Let* $I \subseteq \{0,1,2,3\}$.

1. *For each* $a \in \mathcal{C}_I^\perp$, *there exists unique* $b \in \mathcal{D}_I^\perp$ *such that*
$$R(\mathcal{C}_I \oplus a) = \mathcal{D}_I \oplus b.$$
2. *For each* $a \in \mathcal{C}_I^\perp$, *there exists unique* $b \in \mathcal{C}_I^\perp$ *such that*
$$\text{S-Box}^{-1} \circ M' \circ \text{S-Box}(\mathcal{C}_I \oplus a) = \mathcal{C}_I \oplus b.$$
3. *For each* $a \in \mathcal{D}_I^\perp$, *there exists unique* $b \in \mathcal{M}_I^\perp$ *such that*
$$M' \circ \text{S-Box}(\mathcal{D}_I \oplus a) = \mathcal{M}_I \oplus b.$$
4. *For each* $a \in \mathcal{C}_I^\perp$, *there exists unique* $b \in \mathcal{IM}_I^\perp$ *such that*
$$M' \circ SR^{-1}(\mathcal{C}_I \oplus a) = \mathcal{IM}_I \oplus b, \qquad b = M' \circ SR^{-1}(a).$$

Proof. 1. In order to prove the statement, we simply compute $R(\mathcal{C}_I \oplus a)$.
First of all, it is easy to observe that the S-Box Layer and that the Linear Layer M' map column space $\mathcal{C}_I$ to column space $\mathcal{C}_I$ (more precisely, the S-Box Layer maps coset $\mathcal{C}_I$ into other coset of $\mathcal{C}_I$, since S-Box$(0) \neq 0$). Indeed, since SubBytes is bijective and operates on each nibble individually, and since the nibbles of the space $\mathcal{C}_I$ are independent, its only effect is to change the coset. Thus, for each $a \in \mathcal{C}_I^\perp$, there exists unique $a' \in \mathcal{C}_I^\perp$ such that S-Box$(\mathcal{C}_I \oplus a) = \mathcal{C}_I \oplus a'$. Moreover, by definition $SR(\mathcal{C}_I) = \mathcal{D}_I$.
Then we prove that b exists and it is unique. For each $a \in \mathcal{C}_I^\perp$, there exists $c \in \mathcal{C}_I^\perp$ such that:
$$\text{S-Box}(\mathcal{C}_I \oplus a) = \mathcal{C}_I \oplus c$$
where $c_i = \text{S-Box}(a_i)$. Since Linear Layer $SR \circ M'(\cdot)$ is a linear operation, then there exists $b \in \mathcal{D}_I^\perp$ such that:
$$SR \circ M'(\mathcal{C}_I \oplus c) = SR \circ M'(\mathcal{C}_I) \oplus SR \circ M'(c) = \mathcal{D}_I \oplus b,$$
where $b = SR \circ M'(c)$.

2. In order to prove the statement, we simply compute *super-SBox*$(C_I \oplus a)$. First of all, it is easy to observe that the S-Box Layer and that the Linear Layer M' map column space $\mathcal{C}_I$ to column space $\mathcal{C}_I$.
   Then we prove that b exists and it is unique. For each $a \in \mathcal{C}_I^{\perp}$, there exists $c \in \mathcal{C}_I^{\perp}$ such that:
   $$\text{S-Box}(\mathcal{C}_I \oplus a) = \mathcal{C}_I \oplus c$$
   where $c_i = \text{S-Box}(a_i)$. Since Linear Layer M' is a linear operation, then there exists $d \in \mathcal{C}_I^{\perp}$
   $$M'(\mathcal{C}_I \oplus c) = \mathcal{C}_I \oplus d$$
   where $d = M'(c)$. Finally, using the previous observation, there exists $b \in \mathcal{C}_I^{\perp}$ such that:
   $$\text{S-Box}^{-1}(\mathcal{C}_I \oplus d) = \mathcal{C}_I \oplus b$$
   where $b_i = \text{S-Box}^{-1}(d_i)$.
3. In order to prove the statement, we simply compute $M' \circ \text{S-Box}(\mathcal{D}_I \oplus a)$. First of all, it is easy to observe that the S-Box Layer maps a diagonal space $\mathcal{D}_I$ to diagonal space $\mathcal{D}_I$ Moreover, by definition $\mathcal{M}_I = M'\mathcal{D}_I$.
   Then we prove that b exists and it is unique. For each $a \in \mathcal{D}_I^{\perp}$, there exists $c \in \mathcal{D}_I^{\perp}$ such that:
   $$\text{S-Box}(\mathcal{D}_I \oplus a) = \mathcal{D}_I \oplus c$$
   where $c_i = \text{S-Box}(a_i)$. Working exactly as before, since M' is a linear operation, it follows that $b = M'(c)$.
4. Since $M' \circ SR^{-1}$ is a linear operation, it follows that:
   $$M' \circ SR^{-1}(\mathcal{C}_I \oplus a) = M' \circ SR^{-1}(\mathcal{C}_I) \oplus M' \circ SR^{-1}(a)\mathcal{I}\mathcal{M}_I \oplus b,$$
   where $b = M' \circ SR^{-1}(a)$ is unique in the class of equivalence of $\mathcal{I}\mathcal{M}_I$.

□

Theorem 3. *Let $I \subseteq \{0,1,2,3\}$. For each $a \in \mathcal{C}_I^{\perp}$, there exists unique $b \in \mathcal{M}_I^{\perp}$ such that*

$$R^{(1+1.5)}(\mathcal{C}_I \oplus a) = \mathcal{M}_I \oplus b,$$

where b depends on a and on the secret key. Equivalently:

$$\text{Prob}(R^{(1+1.5)}(x) \oplus R^{(1+1.5)}(y) \in \mathcal{M}_I \,|\, x \oplus y \in \mathcal{C}_I) = 1.$$

Proof. By the previous proposition (point 1), a coset of $\mathcal{C}_I$ is mapped into a coset of $\mathcal{D}_I$ after one round. That is, there for each $a \in \mathcal{C}_I^{\perp}$, there exists unique $b' \in \mathcal{D}_I^{\perp}$ such that $R(\mathcal{C}_I \oplus a) = \mathcal{D}_I \oplus b'$.

By point 2 of previous proposition, for each $b' \in \mathcal{D}_I^{\perp}$, there exists unique $b \in \mathcal{M}_I^{\perp}$ such that $M' \circ \text{S-Box}(\mathcal{D}_I \oplus b') = \mathcal{M}_I \oplus b$.

Finally, note that the add round key/constant (ARK) simply changes the coset, but not the subspace. That is, if k is a generic key/constant and X is a generic space, then

$$ARK(X \oplus a) = X \oplus (a \oplus k).$$

Thus, the thesis is proved. □

Theorem 4. *Let $I \subseteq \{0,1,2,3\}$. For each $a \in \mathcal{C}_I^{\perp}$, there exists unique $b \in \mathcal{IM}_I^{\perp}$ such that*

$$R^{(2+0.5)}(\mathcal{C}_I \oplus a) = \mathcal{IM}_I \oplus b,$$

where b depends on a and on the secret key. Equivalently:

$$\text{Prob}(R^{(2+0.5)}(x) \oplus R^{(2+0.5)}(y) \in \mathcal{IM}_I \,|\, x \oplus y \in \mathcal{C}_I) = 1.$$

Proof. The proof is analogous to the one of the previous theorem, and it is based on the previous proposition. □

B Impossible Subspace Trail for PRINCE

In Sect. 3, we have presented two subspace trails for 2.5 rounds of PRINCE that hold with probability 1. It is also possible to describe a subspace trails with probability 0 (i.e. an "impossible subspace trail") for 4.5 rounds of PRINCE, and to set up an impossible differential attack on 6 rounds of PRINCE.

Proposition 2. *Let $I, J \subseteq \{0,1,2,3\}$ with $|I| = |J| = 1$. Then $\mathcal{M}_J \cap \mathcal{D}_I = \{0\}$.*

Proof. A basis for $\mathcal{M}_J$ is given by:

$$\mathcal{M}_J = \langle M'(e[4{\cdot}J]), M'(e[4{\cdot}(J{-}1){+}1]), M'(e[4{\cdot}(J{-}2){+}2]), M'(e[4{\cdot}(J{-}3){+}3])\rangle,$$

while a basis for $\mathcal{D}_I$ is given by $\mathcal{D}_I = \langle e[4\cdot I], e[4\cdot(I-1)+1], e[4\cdot(I-2)+2], e[4\cdot(I-3)+3]\rangle$, where in both cases the indexes are taken modulo 16. Moreover, note that in both cases the vectors $e[\cdot]$ lie on different columns.

Suppose by contradiction that $\mathcal{D}_I$ and $\mathcal{M}_J$ have a nonzero intersection. This implies that there exist x_k and y_k for $k = 0, ..., 3$ such that

$$\bigoplus_{k=0}^{3} x_k \cdot \langle e[4\cdot(I-k)+k]\rangle \oplus \bigoplus_{k=0}^{3} y_k \cdot \langle M'(e[4\cdot(J-k)+k])\rangle = 0,$$

that is

$$\bigoplus_{k=0}^{3} \left[x_k \cdot \langle e[4\cdot(I-k)+k]\rangle \oplus y_{\tilde{k}(k)} \cdot \langle M'(e[4\cdot(I-k)+\tilde{k}(k)])\rangle \right] = 0$$

has a nontrivial solution, where $\tilde{k}(k) = (k + J - I) \bmod 4$.

The only possible solution is given by:

$$x_k \cdot \langle e[4\cdot(I-k)+k]\rangle = y_{\tilde{k}(k)} \cdot \langle M'(e[4\cdot(I-k)+\tilde{k}(k)])\rangle \qquad \forall k = 0, ..., 3,$$

which is clearly impossible since $\langle e[4\cdot(I-k)+k)]\rangle$ and $\langle M'(e[4\cdot(I-k)+\tilde{k}(k)])\rangle$ are linearly independent for each $k = 0, ..., 3$ (note that they lie on the same column). Thus, $\mathcal{D}_I$ and $\mathcal{M}_J$ intersect only in zero.

□

Using the previous proposition, it is possible to set up a subspace trail with probability 0.

Theorem 5. *Let $I, J \subseteq \{0,1,2,3\}$ with $|I| = |J| = 1$ and let $R^4(\cdot)$ denote the four middle rounds. Then, for each $x \neq y$:*

$$\begin{aligned}&\text{Prob}(R^{(4)}(x) \oplus R^{(4)}(y) \in \mathcal{C}_I \,|\, x \oplus y \in \mathcal{C}_J) = \\ =&\text{Prob}(M' \circ SR^{-1} \circ R^{(4)}(x) \oplus M' \circ SR^{-1} \circ R^{(4)}(y) \in \mathcal{IM}_I \,|\, x \oplus y \in \mathcal{C}_J) = 0.\end{aligned}$$

Proof. We have previously seen that a coset of $\mathcal{C}_J$ is mapped into a coset of $\mathcal{M}_J$ after 2.5 rounds, that is for each $a \in \mathcal{C}_J^{\perp}$ there exists unique $b \in \mathcal{M}_J^{\perp}$ such that:

$$R^{(1+1.5)}(\mathcal{C}_J \oplus a) = \mathcal{M}_J \oplus b.$$

By previous proposition, $\mathcal{M}_J \cap \mathcal{D}_I = \{0\}$ for each I, J with $|I| = |J| = 1$. That is, if two elements belong to the same coset of $\mathcal{M}_J$, they can not belong to the same coset of $\mathcal{D}_I$, that is for $x \neq y$:

$$\text{Prob}(R^{(1+1.5)}(x) \oplus R^{(1+1.5)}(y) \in \mathcal{D}_I \,|\, x \oplus y \in \mathcal{C}_J) = 0.$$

Since S-Box Layer maps diagonal space $\mathcal{D}_I$ to column space $\mathcal{D}_I$:

$$\text{Prob}(\textit{super-SBox} \circ R(x) \oplus \textit{super-SBox} \circ R(y) \in \mathcal{D}_I \,|\, x \oplus y \in \mathcal{C}_J) = 0.$$

Moreover, since for each $a' \in \mathcal{D}_I^{\perp}$ there exists unique $b' \in \mathcal{C}_I^{\perp}$ such that $R^{-1}(\mathcal{D}_I \oplus a') = \mathcal{C}_I \oplus b'$, it follows that

$$\text{Prob}(R^{(4)}(x) \oplus R^{(4)}(y) \in \mathcal{C}_I \,|\, x \oplus y \in \mathcal{C}_J) = 0,$$

where $R^4(\cdot)$ denote the four middle rounds.

Thus, if two elements belong to the same coset of $\mathcal{C}_J$, they can not belong to the same coset of $\mathcal{C}_I$ after four rounds. The other probability follows immediately by the definition of $\mathcal{IM}_I$, that is $\mathcal{IM}_I = M' \circ SR^{-1}(\mathcal{C}_I)$. □

This means that a coset of $\mathcal{C}_J$ can not be mapped into a coset of $\mathcal{IM}_I$ for $|I| = |J| = 1$ after 4.5 rounds (equivalently into a coset of $\mathcal{C}_I$ after 4 rounds):

$$\mathcal{C}_J \oplus a \xrightarrow{M' \circ \text{S-Box} \circ R(\cdot)} \mathcal{M}_J \oplus b \xrightarrow{\text{S-Box}^{-1}(\cdot)} \mathcal{D}_I \oplus c \xrightarrow{R^{-1}(\cdot)} \mathcal{C}_I \oplus d \xrightarrow{M' \circ SR^{-1}} \mathcal{IM}_I \oplus e.$$

Starting from the previous theorem, it is possible to set up an impossible differential attack for 6-rounds of PRINCE which requires only known-plaintexts. Assume $|I| = |J| = 1$. Since $M' \circ SR^{-1}(\mathcal{C}_J) = \mathcal{IM}_J$, it follows that if $x \oplus y \in \mathcal{C}_J$, then $M' \circ SR^{-1}(x) \oplus M' \circ SR^{-1}(y) \in \mathcal{IM}_J$. Moreover, note that also $M' \circ SR^{-1} \circ R^{(4)}(x) \oplus M' \circ SR^{-1} \circ R^{(4)}(y) \in \mathcal{IM}_I$. Thus, the following property for 6-rounds reduced PRINCE holds.

Proposition 3. *Let $I, J \in \{0,1,2,3\}$ with $|I| = |J| = 1$ and let $\tilde{k}$ and $\hat{k}$ the secret key of 6-rounds reduced PRINCE defined as in the introduction. Given*

two pairs of plaintexts-ciphertexts (p^1, c^1) *and* (p^2, c^2)*, they satisfy the following probability:*

$$\text{Prob}(\text{S-Box}^{-1}(p^0 \oplus \tilde{k}) \oplus \text{S-Box}^{-1}(p^1 \oplus \tilde{k}) \in \mathcal{IM}_J \qquad \textit{and} \qquad \textit{and} \qquad \text{S-Box}(c^0 \oplus \hat{k}) \oplus \text{S-Box}(c^1 \oplus \hat{k}) \in \mathcal{IM}_I) = 0.$$

If $I = J$*:*

$$\text{Prob}(\text{S-Box}^{-1}(p^0 \oplus \tilde{k}) \oplus \text{S-Box}^{-1}(p^1 \oplus \tilde{k}) \oplus \oplus \text{S-Box}(c^0 \oplus \hat{k}) \oplus \text{S-Box}(c^1 \oplus \hat{k}) \in \mathcal{IM}_I) = 0.$$

Using the previous proposition is it possible to set up an impossible differential attack for 6-rounds of PRINCE which requires only known-plaintexts. However, even if this attack is better than a brute force one, it is worse than other know-plaintexts attacks on 6-rounds PRINCE present in literature - e.g. [9].

C A Related Key Attack

In Sect. 4, we have presented an equivalent version of PRINCE, illustrated in Fig. 2. This version - called PRINCE* - is analogous to the original PRINCE, but it is not equivalent, since *no* (respectively Inverse) ShiftRows operation is applied to the key in the forward rounds (respectively backward). This version has the advantage that the order of ShiftRows and MixLayer operations are the same of AES, and we claim that it has the same security of the original version. For the following, we define the constant $RC_i^\star$ of this version of PRINCE as $RC_i^\star = SR^{-1}(RC_i)$ for the forward rounds and $RC_i^\star = SR(RC_i)$ for the backward rounds, where RC_i are the constants of original PRINCE.

We stress that if an InverseShiftRows operation (respectively ShiftRows) is applied to the key in the backward rounds (respectively forward), then this version is completely equivalent to the original one.

Obviously, if $k_0 = SR(k_0) = SR^{-1}(k_0)$ and $k_1 = SR(k_1) = SR^{-1}(k_1)$, then the encryption process of this second version doesn't change. Note that $x = SR(x)$ if and only if $x = SR^{-1}(x)$. Moreover, $x = SR(x)$ if and only if $x[1] = x[5] = x[9] = x[13]$, $x[2] = x[10]$, $x[6] = x[14]$ and $x[3] = x[7] = x[11] = x[15]$, which happens with probability 2^{-32}.

Assume that an attacker has access to the original version of PRINCE and to this second one PRINCE* (where we suppose that an InverseShiftRows is applied to the plaintext, and a ShiftRows is applied to the ciphertext) instantiated with the same secret key. Given a plaintext p, assume she can obtain the corresponding ciphertext for the original version of PRINCE (denoted by c) and for the second version (denote by $c^\star$). If $c = c^\star$, then she can deduce that $k_0 = SR(k_0)$ and $k_1 = SR(k_1)$ with probability $1 - 2^{-64} \approx 1$. If this happens, she has to test $2^{32} \cdot 2^{32} = 2^{64}$ keys instead of 2^{128} in order to find the right key.

D MANTIS Encryption Scheme: Subspace Trail Cryptanalysis

MANTIS encryption scheme [5] is a low-latency tweakable block cipher proposed at CRYPTO 2016. The starting point used by the designer for this encryption scheme is a PRINCE-like encryption scheme, keeping the entire design symmetric around the middle (to have the α-reflection property). In order to improve the security, the PRINCE-round has been replaced by the MIDORI-round function, that is the S-Box, the MixLayer and the ShiftRows of PRINCE have been replaced with the S-Box, the MixLayer and the PermuteCells (analogous to ShiftRows) of MIDORI. This simple change results in a cipher with improved latency and improved security compared to PRINCE. Note that in contrast to PRINCE, the PermuteCells operation is performed before the MixLayer one.

MANTIS$_r$ has a 64-bit block length and works with a 128-bit key k and 64-bit tweak T. The parameter r specifies the number of rounds of one half of the cipher (for example, MANTIS$_6$ has the same number of rounds of PRINCE). As PRINCE, MANTIS is based on the FX-construction and thus applies whitening keys before and after applying its core components (the whitening keys are generated in the same way as for PRINCE). That is, the 128-bit key is first split into $k = k_0||k_1$ with 64-bit subkeys k_0, k_1. Then, $(k_0||k_1)$ is extended to the 192 bit key $(k_0||k_0'||k_1) := (k_0||(k_0 \gg 1) \oplus (k_0 \lll 63)||k_1)$, and k_0, k_0' are used as whitening keys in an FX-construction. The subkey k_1 is used as the round key for all of the $2r$ rounds of MANTIS$_r$. Every round in MANTIS consists of an S-Box layer, a round constant addition, a tweak addition, a PermuteCells operation, a Linear layer and a final key addition

$$R^i(\cdot) = M \circ P(h^i(T) \oplus k_1 \oplus RC_i \oplus \text{S-Box}(\cdot)),$$

for $i = 0, ..., r$, where[8]:

- **S-Box layer**: Every byte in the internal state is replaced by using the involutory 4×4-bit MIDORI S-Box;
- **A bit-wise XOR with a round constant** RC_i, for $i = 0, ..., r$;
- **A bit-wise XOR with the the (full) round tweakey state** $h^i(T) \oplus k_1$, for $i = 0, ..., r$, where T is the tweak and h^i is the tweak permutation;
- **PermuteCells Operation P**: The cells of the internal state are permuted according to the MIDORI permutation:

$$P = [0, 11, 6, 13, 10, 1, 12, 7, 5, 14, 3, 8, 15, 4, 9, 2];$$

- **MixColumns M**: Each column of the cipher internal state array is multiplied by the MixColumns binary matrix of MIDORI M:

$$M = \begin{bmatrix} 0\,1\,1\,1 \\ 1\,0\,1\,1 \\ 1\,1\,0\,1 \\ 1\,1\,1\,0 \end{bmatrix},$$

[8] We refer to [5] and [3] for a complete description of the S-Box, the PermuteCells and the MixColumns operations.

where $M = M^{-1}$;
- **A bit-wise XOR with the key** k_1.

As for PRINCE, in the last r rounds the order of operations is inverse with respect to the first r rounds, where only the round constants differ. Moreover, the middle rounds consist of three key-less operations: an S-Box layer, a matrix multiplication with M and an inverse S-Box layer. Finally, as PRINCE, MANTIS has the *α-reflection property*, that is $D_{(k_0||k_0'||k_1)}(\cdot, T) = E_{(k_0'||k_0||k_1 \oplus \alpha)}(\cdot, T)$. Thus, our results presented in Sect. 4 can be applied on MANTIS.

Subspace Trail of MANTIS. Proceeding as for PRINCE, we first identify analogous subspace trails for MANTIS. The column, diagonal and mixed subspaces are defined exactly as the ones defined for PRINCE in Sect. 3.1, but their representations are a little different (expect for the column space).

For instance, $\mathcal{D}_0 = P(\mathcal{C}_0)$, $\mathcal{ID}_0 = P^{-1}(\mathcal{C}_0)$, $\mathcal{M}_0 = M(\mathcal{D}_0)$ and $\mathcal{IM}_0 = M(\mathcal{ID}_0)$ correspond to matrix representations:

$$\mathcal{D}_0 \equiv \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & 0 & y & 0 \\ 0 & 0 & 0 & z \\ 0 & w & 0 & 0 \end{bmatrix} \quad \mathcal{ID}_0 \equiv \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & w \end{bmatrix} \quad \mathcal{M}_0 \equiv \begin{bmatrix} 0 & w & y & z \\ x & w & 0 & z \\ x & w & y & 0 \\ x & 0 & y & z \end{bmatrix} \quad \mathcal{IM}_0 \equiv \begin{bmatrix} 0 & w & y & z \\ x & 0 & y & z \\ x & w & 0 & z \\ x & w & y & 0 \end{bmatrix}.$$

Note that MANTIS as some similarities to the version of PRINCE called PRINCE$'$ presented in Sect. 4, in particular for the definition of the middle-rounds and for the ordered of the operations in each round. Thus, let $I \subseteq \{0, 1, 2, 3\}$. Since $\mathcal{C}_I$ is an invariant subspace for the middle rounds, note that it is possible to set up a subspace trail for 3.5 rounds of MANTIS:

$$\mathcal{ID}_I \oplus a \xrightarrow{R \circ ARK(\cdot)} \mathcal{C}_I \oplus b \xrightarrow{super\text{-}SBox(\cdot)} \mathcal{C}_I \oplus c \xrightarrow{M' \circ SR^{-1}(\cdot)} \mathcal{IM}_I \oplus d.$$

A More Secure Version of MANTIS. As for PRINCE, we consider a version of MANTIS where the MixColumns and the PermuteCells operations are exchanged in positions - called for the following MANTIS*. In this version, the rounds of MANTIS* are defined similar of the PRINCE ones, where the MixColumns operation is performed before (resp. after) the PermuteCells one in the forward (resp. backwards) rounds.

As first consequence, in this case it is only possible to set up a subspace trail for 2.5 rounds (similar to PRINCE), that is:

$$\mathcal{C}_I \oplus a \xrightarrow{super\text{-}SBox(\cdot)} \mathcal{C}_I \oplus b \xrightarrow{M \circ SR^{-1}(\cdot)} \mathcal{IM}_I \oplus c$$

or

$$\mathcal{C}_I \oplus a \xrightarrow{R(\cdot)} \mathcal{D}_I \oplus b \xrightarrow{M \circ \text{S-Box}(\cdot)} \mathcal{M}_I \oplus c.$$

Moreover, "*as one round of MANTIS is almost identical to one round in MIDORI, most of the security analysis can simply be copied from the latter*"

(see Sect. 6.3 of [5]). By our analysis of Sect. 4 and since MIDORI [3] is an AES-like cipher, its security is not influenced by the positions of the MixColumns and of the PermuteCells operations. Thus, the version of MIDORI - called for consistency MIDORI* - in which the MixColumns operation is performed before the PermuteCells operation has the same security of the original one.

Due to previous considerations and since the analysis done for PRINCE in Sect. 4 also applies on MANTIS as well, we can claim that *MANTIS** (i.e. the version of MANTIS in which MixColumns and PermuteCells are exchanged in positions) *is more secure than the original version proposed by [5] with respect to the attack vectors considered in this paper*. Note that this claim is also justified by the fact that authors didn't consider related-key attacks in order to evaluate the security of MANTIS, and that its key schedule is linear (in particular, there is no key-schedule since all the subkeys are equal to the whitening key).

For completeness and following our analysis of Sect. 4, we defined another version of MANTIS - called in the following MANTIS$'$, such that MANTIS$'$ is identical to the original MANTIS excepted for the middle rounds, defined as

$$\textit{middle-rounds}(\cdot) = \text{S-Box}^{-1} \circ P^{-1} \circ M \circ P \circ \text{S-Box}(\cdot).$$

As for MANTIS*, we can claim that MANTIS$'$ is more secure than the original version proposed by [5], and that it has the same security of MANTIS*.

For completeness, a similar but independent analysis is proposed in [10], which leads to analogous results and conclusions.

E Truncated Differential Attack on 3 Rounds of PRINCE

Using the first 2.5 rounds subspace trail presented in the previous section, we present a truncated differential attack on 3 rounds of PRINCE. Consider 3 rounds of PRINCE:

$$p \xrightarrow{R(\cdot)} q \xrightarrow{M' \circ \text{S-Box}(\cdot)} s \xrightarrow{\text{S-Box}^{-1}(\cdot)} c,$$

where the plaintexts must be chosen in the same coset of $\mathcal{C}_I$, and the states s belong in the same coset of $\mathcal{M}_I$.

The following attack on 3-rounds of PRINCE is based on the properties of the subspace trail presented in Sect. 3.1. As we have seen, if p^1 and p^2 are a pair of plaintexts that belongs to the same coset of $\mathcal{C}_I$ (that is $p^1 \oplus p^2 \in \mathcal{C}_I$), then $s^1 \oplus s^2 := R^{(1+1.5)}(p^1) \oplus R^{(1+1.5)}(p^2)$ belongs to the subspace $\mathcal{M}_I$ with probability 1. Thus, consider a pair of plaintexts that belong to the same coset of $\mathcal{C}_I$, i.e. such that the differences of the nibbles in $4 - |I|$ columns are equal to 0. After 2.5 rounds, the corresponding texts belong to the same coset of $\mathcal{M}_I$, i.e. their sum belongs to the subspace $\mathcal{M}_I$.

If we denote by $\hat{k} := k'_0 \oplus k_1 \oplus \alpha$ the key of the final round, it has to satisfy the condition that $s^1 = \text{S-Box}(c^1 \oplus \hat{k})$ and $s^2 = \text{S-Box}(c^2 \oplus \hat{k})$ belong to the same coset of $\mathcal{M}$ with probability 1, that is:

$$\begin{aligned} &R^{(1+1.5)}(p^1) \oplus R^{(1+1.5)}(p^2) \in \mathcal{M}_I, \qquad \text{or equivalently} \\ &\text{S-Box}(c^1 \oplus k'_0 \oplus k_1 \oplus \alpha) \oplus \text{S-Box}(c^2 \oplus k'_0 \oplus k_1 \oplus \alpha) \in \mathcal{M}_I. \end{aligned} \tag{13}$$

If the previous condition is not satisfied, then the guessed key is certainly wrong.

Theorem 6. *Let p^1 and p^2 be two plaintexts of the same coset of $\mathcal{C}_I$ and let $\hat{k}$ be the secret round-key of the final round.*

Let $R^{(1+1.5)}$ be defined as in (6). If there exists a pair of plaintexts such that $(R^{(1+1.5)}(p^1), R^{(1+1.5)}(p^2))$ don't satisfy the conditions (13) for a certain key $\hat{k}$, then $\hat{k}$ is certainly wrong.

Proof. Suppose by contradiction that $\hat{k}$ is the right key.

If there exists a pair $(R^{(1+1.5)}(p^1), R^{(1+1.5)}(p^2))$ such that $\hat{k}$ doesn't satisfy (13), then $R^{(1+1.5)}(p^1) \oplus R^{(1+1.5)}(p^2) \notin \mathcal{M}_I$ (i.e. $R^{(1+1.5)}(p^1)$ and $R^{(1+1.5)}(p^2)$ belong to two different cosets of $\mathcal{M}_I$), that is $\text{Prob}(R^{(1+1.5)}(p^1) \oplus R^{(1+1.5)}(p^2) \in \mathcal{M}_I \,|\, p^1 \oplus p^2 \in \mathcal{C}_I) \neq 1$.

Since $\hat{k}$ is the right key, then $\text{Prob}(R^{(1+1.5)}(p^1) \oplus R^{(1+1.5)}(p^2) \in \mathcal{M}_I \,|\, p^1 \oplus p^2 \in \mathcal{C}_I) = 1$ (see (7)), which is a contradiction. □

When the attacker has found all the nibbles of $k'_0 \oplus k_1$, she can compute $q^1 = \textit{super-SBox}(c^1 \oplus \hat{k})$ and $q^2 = \textit{super-SBox}(c^2 \oplus \hat{k})$ (note that $\textit{super-SBox}(\cdot) \equiv \textit{super-SBox}^{-1}(\cdot)$). In order to find k_1, the idea is to repeat the same attack of before. For this second step, it is better to use (chosen) plaintexts that belong to the same coset of $\mathcal{D}_I$. Indeed, for each $a \in \mathcal{D}_I^{\perp}$ there exists unique $b \in SR(\mathcal{M}_I)^{\perp}$ such that:

$$R(\mathcal{D}_I \oplus a) = SR(\mathcal{M}_I) \oplus b.$$

Thus, the attacker has to guess k_1 and to check that the following condition is satisfied:

$$\text{S-Box}(q^1 \oplus k_1 \oplus RC_1) \oplus \text{S-Box}(q^2 \oplus k_1 \oplus RC_1) \in SR(\mathcal{M}_I).$$

If this condition is not satisfied, then the guessed key is certainly wrong.

In order to reduce the number of plaintexts and the time complexity of the attack, the idea is to work independently on each column of $\mathcal{M}_I$ (equivalent of $SR(\mathcal{M}_I)$), since they depend on different and independent variables. Moreover, intuitively the attack is more competitive when the dimension of the subspace $\mathcal{M}_I$ is smallest as possible, that is $|I| = 1$. The details of the attacks are given in the following, where we limit to consider the case $I = \{0\}$. Here, we limit to highlight that given a key $\hat{k}$ (similar for k_1), there are two possibilities to check if two texts s^1 and s^2 (where $s^i := \text{S-Box}(c^i \oplus \hat{k})$ for $i = 1, 2$) belong or not to the same coset of $\mathcal{M}_I$, that is:

1. work independently on each nibble;
2. exploit the relationships that hold among the nibbles.

To show briefly these two methods, consider the first column (analogous for the others) and let $s := s^1 \oplus s^2$. In the first case, the conditions that one can use are

$$s[0] \wedge \text{0x8} = 0, \quad s[1] \wedge \text{0x4} = 0, \quad s[2] \wedge \text{0x2} = 0, \quad s[3] \wedge \text{0x1} = 0, \qquad (14)$$

where each one of these conditions involves only 1 bit. In the second case, the relationships that one can use are

$$s[1] \wedge 0\text{x}4 = 0, \qquad s[0] \wedge 0\text{x}b = s[1] \wedge 0\text{x}7, \qquad s[2] \wedge 0\text{x}b = s[1] \wedge 0\text{x}d,$$
$$s[3] \wedge 0\text{x}b = s[1] \wedge 0\text{x}e, \qquad s[0] \wedge 0\text{x}4 = s[2] \wedge 0\text{x}4 = s[3] \wedge 0\text{x}4,$$

which involve more bits. Intuitively, *an attack that exploits the relationship among the nibbles is (much) more competitive than one which works independently on each nibble.* In particular, as shown in the following, the first attack requires 18 chosen plaintexts and the computational cost is approximately 2^5 three-rounds PRINCE, while the second one requires 10 chosen plaintexts and the computational cost is approximately $2^{4.8}$ three-rounds PRINCE. Extending these attacks at the end, we set up a very competitive attacks on 4 rounds of PRINCE - presented in Sect. 5 - which won the PRINCE challenge in the 4-round chosen-plaintext category.

For completeness, all the relationships that can be exploited to set up an attack are given in the following theorem.

Theorem 7. *Let $a, b \in \mathcal{M}_0^{\perp}$, and let $s^1 \in \mathcal{M}_0 \oplus a$ and $s^2 \in \mathcal{M}_0 \oplus b$. Denote s as the sum of s^1 and s^2, i.e. $s = s^1 \oplus s^2$. If $a = b$ (that is, if s^1 and s^2 belong to the same coset of $\mathcal{M}_0$), then the following equivalences are satisfied:*

$$s[1] \wedge 0\text{x}4 = 0, \qquad s[0] \wedge 0\text{x}b = s[1] \wedge 0\text{x}7, \qquad s[2] \wedge 0\text{x}b = s[1] \wedge 0\text{x}d,$$
$$s[3] \wedge 0\text{x}b = s[1] \wedge 0\text{x}e, \qquad s[0] \wedge 0\text{x}4 = s[2] \wedge 0\text{x}4 = s[3] \wedge 0\text{x}4;$$
$$s[5] \wedge 0\text{x}4 = 0, \qquad s[4] \wedge 0\text{x}b = s[5] \wedge 0\text{x}7, \qquad s[6] \wedge 0\text{x}b = s[5] \wedge 0\text{x}d,$$
$$s[7] \wedge 0\text{x}b = s[5] \wedge 0\text{x}e, \qquad s[4] \wedge 0\text{x}4 = s[6] \wedge 0\text{x}4 = s[7] \wedge 0\text{x}4;$$
$$s[10] \wedge 0\text{x}4 = 0, \qquad s[8] \wedge 0\text{x}b = s[10] \wedge 0\text{x}e, \qquad s[9] \wedge 0\text{x}b = s[10] \wedge 0\text{x}7,$$
$$s[11] \wedge 0\text{x}b = s[10] \wedge 0\text{x}d, \qquad s[8] \wedge 0\text{x}4 = s[9] \wedge 0\text{x}4 = s[11] \wedge 0\text{x}4;$$
$$s[12] \wedge 0\text{x}4 = 0, \qquad s[13] \wedge 0\text{x}b = s[12] \wedge 0\text{x}d, \qquad s[14] \wedge 0\text{x}b = s[12] \wedge 0\text{x}e,$$
$$s[15] \wedge 0\text{x}b = s[12] \wedge 0\text{x}7, \qquad s[13] \wedge 0\text{x}4 = s[14] \wedge 0\text{x}4 = s[15] \wedge 0\text{x}4.$$

Proof. We prove the statement only for the first column (it is analogous for the others). Let $a = b$. Since s^1 and s^2 belong to the same coset $\mathcal{M}$, then there exist x_1 and x_2 in $GF(2^4)$ such that $s^i[j] = \alpha_j(x_i) \oplus a_j$ for $i = 1, 2$ and $j = 0, ..., 3$. If we consider the first two nibbles:

$$s[0] \wedge 0\text{x}b = [(x_1 \oplus x_2) \wedge 0\text{x}7] \wedge 0\text{x}b = (x_1 \oplus x_2) \wedge 0\text{x}3,$$
$$s[1] \wedge 0\text{x}7 = [(x_1 \oplus x_2) \wedge 0\text{x}b] \wedge 0\text{x}7 = (x_1 \oplus x_2) \wedge 0\text{x}3.$$

That is, $s[0] \wedge 0\text{x}b = s[1] \wedge 0\text{x}7$.

Note that the second, the third and the fourth equivalence don't involve the bit $s[i] \wedge 0\text{x}4$ for $i = 0, 2, 3$. This justify the last equivalence.

In the same way, it is possible to prove that the other equivalences are satisfied. □

Note that *both these attacks are truncated differential attacks*, where in the second case one exploits the relationships among the nibbles in order to find the secret key. *Truncated Differential attack* was introduced by Knudsen in [15] and it is a generalization of *Classical Differential Attack* introduced by Shamir and Biham in [6]. The differential attacks exploit that pairs of plaintexts with certain differences yield other certain differences in the corresponding ciphertexts with a non-uniform probability distribution. Statistical key information is deduced from ciphertext blocks obtained by encrypting pairs of plaintext block with a specific (bitwise) difference under the target key. In truncated differential, the attacker considers only part of the difference between pairs of texts, i.e. it is a differential attack where only a part of the difference in the ciphertexts can be predicted. In the second case, the relationships among the nibbles can be derived by the equivalent requirement

$$M' \circ \text{S-Box}(c^1 \oplus k'_0 \oplus k_1 \oplus \alpha) \oplus M' \circ \text{S-Box}(c^2 \oplus k'_0 \oplus k_1 \oplus \alpha) \in M' \circ \mathcal{M}_I = \mathcal{D}_I,$$

that is the attacker is looking for a key $k'_0 \oplus k_1$ for which the difference in $16 - 4 \cdot |I|$ nibbles is equal to 0 (by definition of $\mathcal{D}_I$), while no condition is imposed on the other nibbles, which is exactly a truncated differential attack.

Finally, we would like to highlight that the same attack on 3 rounds work in the same way for the other 2.5 rounds subspace trails given before.

E.1 Details of the Attack - Working Independently on Each Nibble

Suppose that the attacker knows two ciphertexts c^1 and c^2 such that the respective plaintexts p^1 and p^2 belong to the same coset of $\mathcal{C}_0$ (that is $p^1 \oplus p^2 \in \mathcal{C}_0$). We first consider the case in which the attacker works independently on each nibble - that is, she doesn't exploit the relationships among the nibbles - in order to recover the secret key. We focus on the condition that the first nibble (i.e. the nibble in position 0) of the key of the final round $\hat{k} := k_1 \oplus k'_0 \oplus \alpha$ has to satisfy in order to guarantee that $s^1 \oplus s^2 = \text{S-Box}\ (c^1 \oplus \hat{k}) \oplus \text{S-Box}\ (c^2 \oplus \hat{k}) \in \mathcal{M}_0$, working independently on each nibble of the states: the conditions for the other ones are completely equivalent. The condition for the first nibble $\hat{k}[0]$ is[9]

$$(\text{S-Box}(c^1[0] \oplus \hat{k}[0]) \oplus \text{S-Box}(c^2[0] \oplus \hat{k}[0])) \wedge \text{0x8} = 0. \tag{15}$$

Indeed, if s^1 and s^2 belong to the same coset $\mathcal{M} \oplus a$ (where $a \in \mathcal{M}_0^{\perp}$ fixed), then by definition of $\mathcal{M}_0$, there exists x_1 and x_2 such that $s^i[0] = (x_i \wedge \text{0x7}) \oplus a$ for $i = 1, 2$. By simple computation, $s[0] = s^1[0] \oplus s^2[0] = (x_1 \oplus x_2) \wedge \text{0x7}$, that is

$$s[0] \wedge \text{0x8} = ((x_1 \oplus x_2) \wedge \text{0x7}) \wedge \text{0x8} = (x_1 \oplus x_2) \wedge (\text{0x7} \wedge \text{0x8}) = 0,$$

since $\text{0x7} \wedge \text{0x8} = \text{0x0}$. Similar conditions can be obtained for the other nibbles. If $\hat{k}$ doesn't satisfy the previous condition, it is certainly wrong.

[9] Note that the presence of the *and* logic operator $\wedge$ is very important. Indeed, suppose to consider the following condition $\text{S-Box}(x \oplus \hat{k}) \oplus \text{S-Box}(y \oplus \hat{k}) = 0$ without $\wedge$. Since the S-Box is bijective, the previous equivalence is satisfied iff $x = y$.

Data: 8 ciphertexts pairs (c^1, c^i) where $i = 2, ..., 9$, whose corresponding plaintexts belong in the same coset of $\mathcal{C}_0$
Result: $\hat{k}[0]$
for $\hat{k}[0]$ from 0 *to* $2^4 - 1$ **do**
  flag = 0;
  for i from 2 *to* 9 **do**
    if ([S-Box$(c^1[0] \oplus \hat{k}[0]) \oplus$ S-Box$(c^i[0] \oplus \hat{k}[0])] \wedge$ 0x8 ! = 0) **then**
      flag = 1;
      next value (i.e. next $\hat{k}[0]$);
    end
  end
  if flag = *0* **then**
    return $\hat{k}[0]$.
  end
end

Algorithm 2: *Truncated differential attack on 3 rounds of PRINCE - working independently on each nibble.* For simplicity, we show how to recover only one nibble of $\hat{k}$, that is $\hat{k}[0]$. The other nibbles can be recovered in the same way using the same pairs. When $\hat{k}$ has been discovered, the attacker takes 9 chosen plaintexts in the same coset of $\mathcal{D}_0$, decrypts one round the corresponding ciphertexts and finds k_1 using the same algorithm. For simplicity, in this pseudo-code, we don't use the following optimization: if $\hat{k}[0]$ satisfies (15) then also $\hat{k}[0] \oplus c^1[0] \oplus c^2[0]$ satisfies it.

When the attacker finds all the nibbles of $k_1 \oplus k'_0$, the idea is to compute $q := \textit{super-SBox}(c \oplus \hat{k})$ and to repeat the previous attack on 2 rounds in order to find k_1, choosing plaintexts in the same coset of $\mathcal{D}_0$ and working with $SR(\mathcal{M}_0)$ instead of $\mathcal{M}_0$ (as described previously).

Estimation of the Computational Cost and of the Data Complexity. Given $c^1[0] \neq c^2[0]$, a guessed value of $\hat{k}[0]$ satisfies (12) only with probability 2^{-1}, that is on average only $2^4 \times 2^{-1} = 8$ values of $\hat{k}[0]$ pass the first step. To find the right key, the attacker has to test these 8 values with other pairs of ciphertexts (which plaintexts belong to the same coset of $\mathcal{C}_0$). Thus she needs (at least) 4 steps and 4 different pairs of ciphertexts to find $\hat{k}[0]$. The computational cost in order to find the right values of a single nibble of the secret key is $2 \times (16 \times 2^{-1} + 8 + 4 + 2) \simeq 2^{5.46}$ S-Box look-ups, since on average 8 values pass the first step, 4 pass the second one and so on. Observe that at the first step, the attacker can take advantage of the fact that if $\hat{k}[0]$ satisfies (or not) (12), then also $(\hat{k} \oplus c^1 \oplus c^2)[0]$ satisfies (or not) (12), that is the attacker has to test only 8 values of $\hat{k}[0]$ instead of 16 (this explains the term 16×2^{-1}). Furthermore, suppose that the attacker finds a value of $\hat{k}[0]$ that satisfies (12). A good idea is to check immediately this value with other pairs of ciphertexts: in this way, the attacker doesn't need to store anything.

Observe that if $c^1[0] = c^2[0]$, then the sum (12) is equal to 0 for each values of $\hat{k}[0]$, and the attacker cannot recover any information. That is, a pair of ciphertexts is useful only if $c^1[0] \neq c^2[0]$. Thus, we compute the minimum number of plaintexts that the attacker needs in order to eliminate false candidates and to find (only) the right key with high probability (or, at most, a low number of candidates for the right key). First of all, suppose that 3 ciphertexts are given. Then it is possible to build 3 different pairs, that is $(c^1[0], c^2[0])$, $(c^1[0], c^3[0])$, and $(c^2[0], c^3[0])$. However, only 2 of these pairs are useful for the attack. Indeed, if $\hat{k}[0]$ satisfies the condition (12) for the pairs $(c^1[0], c^2[0])$ and $(c^1[0], c^3[0])$, then it automatically satisfies this condition also for the pair $(c^2[0], c^3[0])$[10], so one of these pair is useless for the attack. Thus, if the attacker needs r different pairs, then she needs $r + 1$ different chosen plaintexts to construct them.

Given 5 chosen plaintexts (that is 4 different pairs), the probability that $c^i[h] \neq c^j[h]$ for each $i \neq j$ and for each nibble h (that is the probability that only the right key passes the test) is only $(0.4999)^{16} = 0.0016\%$. By calculation, the attacker needs at least 9 different chosen plaintexts in order to have a good probability that almost all false positive candidates of the keys are eliminated (which is about 92.2%). We refer to Appendix F for more details about this computation.

For the second step (that is, to find k_1), the attacker has to use other 9 chosen plaintexts that belong to the subspace $\mathcal{D}$. As we have already seen, after one round, the intermediate states belong to $SR(\mathcal{M})$, and the attacker can finds all the nibbles of k_1.

This attack requires $9 \times 2 = 2^{4.17}$ chosen plaintexts and the computational cost is of 2^4 (nibbles) $\times 2^{5.46} \times 2$ (attacks) $+9 \times 16$ (one rounds decryption) $\simeq 2^{10.6}$, S-Box look-ups, that is about 2^5 three-rounds of PRINCE.

E.2 Details of the Attack - Exploiting the Relationship among the Nibbles

Secondly, we present the attack in the case in which the attacker exploits the relationships among the nibbles. Since the columns of $\mathcal{M}_0$ depend on different and independent variables, the attacker can work independently on each columns. For this reason, we show the attack only for the first column (it is completely equivalent for the others).

By Theorem 6, the right key has to satisfy conditions given in Theorem 7. Indeed, as we have seen, key candidates that don't satisfy these conditions are certainly wrong. Consider a pair of ciphertexts c^1 and c^2 whose plaintexts belong to the same coset of $\mathcal{C}_0$. In order to guarantee that $s^1 \oplus s^2 = \text{S-Box}\ (c^1 \oplus \hat{k}) \oplus \text{S-Box}\ (c^2 \oplus \hat{k}) \in \mathcal{M}_0$, the first column of $\hat{k}$ has to satisfy the following conditions (equivalent for the others):

$$(\text{S-Box}(c^1[0] \oplus \hat{k}[0]) \oplus \text{S-Box}(c^2[0] \oplus \hat{k}[0])) \wedge 0\text{x}b =$$
$$= (\text{S-Box}(c^1[1] \oplus \hat{k}[1]) \oplus \text{S-Box}(c^2[1] \oplus \hat{k}[1])) \wedge 0\text{x}7;$$

[10] Indeed: $[\text{S-Box}(c^2[0] \oplus \hat{k}[0]) \oplus \text{S-Box}(c^3[0] \oplus \hat{k}[0])] \wedge 0\text{x}8 = [\text{S-Box}(c^1[0] \oplus \hat{k}[0]) \oplus \text{S-Box}(c^2[0] \oplus \hat{k}[0])] \wedge 0\text{x}8 \oplus [\text{S-Box}(c^1[0] \oplus \hat{k}[0]) \oplus \text{S-Box}(c^3[0] \oplus \hat{k}[0])] \wedge 0\text{x}8 = 0$.

$$
\begin{aligned}
&(\text{S-Box}(c^1[0] \oplus \hat{k}[0]) \oplus \text{S-Box}(c^2[0] \oplus \hat{k}[0])) \wedge 0\text{x}b = \\
=&(\text{S-Box}(c^1[2] \oplus \hat{k}[2]) \oplus \text{S-Box}(c^2[2] \oplus \hat{k}[2])) \wedge 0\text{x}d; \\
&(\text{S-Box}(c^1[0] \oplus \hat{k}[0]) \oplus \text{S-Box}(c^2[0] \oplus \hat{k}[0])) \wedge 0\text{x}b = \\
=&(\text{S-Box}(c^1[3] \oplus \hat{k}[3]) \oplus \text{S-Box}(c^2[3] \oplus \hat{k}[3])) \wedge 0\text{x}e.
\end{aligned} \tag{16}
$$

and

$$
\begin{aligned}
&(\text{S-Box}(c^1[0] \oplus \hat{k}[0]) \oplus \text{S-Box}(c^2[0] \oplus \hat{k}[0])) \wedge 0\text{x}4 = \\
=&(\text{S-Box}(c^1[2] \oplus \hat{k}[2]) \oplus \text{S-Box}(c^2[2] \oplus \hat{k}[2])) \wedge 0\text{x}4 = \\
=&(\text{S-Box}(c^1[3] \oplus \hat{k}[2]) \oplus \text{S-Box}(c^2[3] \oplus \hat{k}[2])) \wedge 0\text{x}4.
\end{aligned} \tag{17}
$$

The idea of the attack is to guess the value $\hat{k}[1]$ and to find the values of $\hat{k}[0], \hat{k}[2]$ and $\hat{k}[3]$ that satisfy the previous conditions. Since the attacker can not impose any restriction/condition on $\hat{k}[1]$, she has to repeat this step for each possible values of $\hat{k}[0]$. Remember that the combinations of keys that don't satisfy the previous conditions are certainly wrong.

Using a pair of ciphertexts c^1 and c^2, for each column on average $2^{16} \cdot 2^{-12} = 2^4$ possible combinations of the key satisfy the previous conditions. Thus, the attacker has to use a second pair of ciphertexts (c^1, c^3) - such that all the plaintexts p^1, p^2 and p^3 belong to the same coset of $\mathcal{C}_0$ - to test the combinations that satisfy the conditions (16)-(17) for the first pair (c^1, c^2). Since conditions (16)-(17) are satisfied with probability 2^{-12}, she certainly finds the right combination for each column of the key. In order to save memory, a good idea is to test immediately the values that pass the first test with the second pair of ciphertexts.

When the attacker has found $k'_0 \oplus k_1$, she can discover k_1 computing $q := \textit{super-SBox}(c \oplus \hat{k})$ and repeating the previous attack on 2 rounds. For this step, she has to choose plaintexts in the same coset of $\mathcal{D}_0$ and to work with $SR(\mathcal{M}_0)$ instead of $\mathcal{M}_0$ (as described previously).

Estimation of the Computational Cost and of the Data Complexity. About the computational cost, to improve the final computational cost of the attack, the attacker should focus on a single equivalence of (16), e.g. the first one. Working only on the first equivalence, the attacker uses the three texts (the two pairs) to find the exact values of $\hat{k}[0]$ and $\hat{k}[1]$. Indeed, observe that with the first pair, on average only $2^8 \cdot 2^{-4} = 2^4$ pairs survived, and using the second pair the attacker can find the right one. When the attacker has found $\hat{k}[1]$, she knows the exact value of the right part of equations (16). Thus, using the first pair of ciphertexts and working on the second equation (similar for the third one), on average only 2 possible values of $\hat{k}[2]$ (similar for $\hat{k}[3]$) survived, and she can finally discover the right one using eq. (17).

For each column, the computational cost can be approximated by 2^4 (values of $\hat{k}[1]$)$\times(2 + 2^4$ (values of $\hat{k}[0]$)$\times 2 + 2 \cdot 2) = 2^{9.25}$ S-Box look-ups for the first

Data: 4 ciphertexts pairs (c^1, c^i) where $i = 2, ..., 5$, whose corresponding plaintexts belong in the same coset of $\mathcal{C}_0$
Result: First column of $\hat{k}$: $\hat{k}[0], ..., \hat{[3]}$
for $\hat{k}[0]$ from 0 *to* $2^4 - 1$ **do**
  for $\hat{k}[1]$ from 0 *to* $2^4 - 1$ **do**
    flag = 0;
    for i from 2 *to* 5 **do**
      if ([S-Box($c^1[0] \oplus \hat{k}[0]$) $\oplus$ S-Box($c^i[0] \oplus \hat{k}[0]$)] $\wedge$ 0xb ! = [S-Box($c^1[1] \oplus \hat{k}[1]$) $\oplus$ S-Box($c^i[1] \oplus \hat{k}[1]$)] $\wedge$ 0x7) **then**
        flag = 1;
        next value (i.e. next $\hat{k}[0]$ or/and $\hat{k}[1]$);
      end
    end
    if flag = *0* **then**
      identify candidates for $\hat{k}[0]$ and $\hat{k}[1]$;
    end
  end
end
for ***all*** *candidates* $\hat{k}[0]$ *and* $\hat{k}[1]$ **do**
  for $\hat{k}[2]$ from 0 *to* $2^4 - 1$ **do**
    as before, check if $\hat{k}[2]$ satisfies the relationships given in 16 and in 17 for all the possible pairs of ciphertexts;
    If satisfied, **then** identify candidate for $\hat{k}[2]$;
    else reject $\hat{k}[2]$ as candidate;
  end
  for $\hat{k}[3]$ from 0 *to* $2^4 - 1$ **do**
    as before, check if $\hat{k}[3]$ satisfies the relationships given in 16 and in 17 for all the possible pairs of ciphertexts;
    If satisfied, **then** identify candidate for $\hat{k}[3]$;
    else reject $\hat{k}[3]$ as candidate;
  end
end
return *First column of* $\hat{k}$*:* $\hat{k}[0], ..., \hat{[3]}$.

Algorithm 3: *Truncated differential attack on 3 rounds of PRINCE - exploiting relationships among nibbles.* For simplicity, we show how to recover only one column of $\hat{k}$, that is $\hat{k}[0], ..., \hat{k}[3]$. The other column can be recovered in the same way using the same pairs. When $\hat{k}$ has been discovered, the attacker takes 5 chosen plaintexts in the same coset of $\mathcal{D}_0$, decrypts one round the corresponding ciphertexts and finds k_1 using the same algorithm.

step (the first equality) and $2 \times (2^4 \cdot 2 + 2 \cdot 2) = 2^{6.2}$ for the second step (the second and the third equalities). Since the attacker has to repeat this procedure for each column of $\mathcal{M}_0$ (remember that the columns of $\mathcal{M}_0$ are independent), the total computational cost is of $4 \cdot (2^{4.5} + 2^{9.25} + 2^{6.2}) = 2^{11.3}$ S-Box look-ups for the four columns. Actually, note that the attacker can test only 2^3 values of $\hat{k}[\cdot]$ instead of 2^4. Indeed, given $\hat{k}[1]$, note that if $\hat{k}[0]$ satisfies (or not) condition

(16), then also $\hat{k}[0] \oplus c^1[1] \oplus c^2[1]$ satisfies (or not) it (similar for $\hat{k}[2]$ and $\hat{k}[3]$). Similar consideration holds also for $\hat{k}[1]$ and $\hat{k}[1] \oplus c^1[1] \oplus c^2[1]$. Thus, the total computational cost of this step is well approximated by $2^{9.3}$ S-Box look-ups.

Observe that if $c^1[i] = c^2[i]$ and $c^1[j] = c^2[j]$ for each $i, j = 0, ..., 3$ and $i \neq j$, then one of the equations (16) is always satisfied for each possible values of the guess key. Thus, the attacker needs more texts to eliminate false candidates and to discover the secret key. For this reason, we are interested to understand how many plaintexts the attacker needs in order to recover (only) the right key with high probability[11]. Using the technique described in Appendix F, it is possible to prove that using only 3 different chosen plaintexts this probability is about 4.2%, using 4 different chosen plaintexts this probability is about 65.7%, while using 5 chosen plaintexts, it is about 94.6%. Thus, the total number of chosen plaintexts needed for this attack is about $2 \times 5 = 10$. The total computational cost is 2 (attacks) $\times 2^{9.3} + 5 \times 16$ (decryption) $\simeq 2^{10.4}$ S-Box look-ups, that is about $2^{4.8}$ three-round of PRINCE.

Finally, we would like to compute the computational cost when the right part of equation (16) is known - this value is used to compute the cost of the attack on 4 rounds (with the extension at the end) described in Sect. 5. Assume that the attacker already knows the value of S-Box$(c^1[0] \oplus \hat{k}[0]) \oplus$ S-Box$(c^2[0] \oplus \hat{k}[0])$. Thus, given $\hat{k}[0]$ and using a single pair of ciphertexts, on average only $(2^4)^3 \times (2^{-3})^3 = 2^3$ possible combinations satisfy the condition (16) for each column of the key. This means that the attacker needs only 2 pairs in order to recover $\hat{k}[1], \hat{k}[2], \hat{k}[3]$. In this case, the computational cost to find $(\hat{k}[1], \hat{k}[2], \hat{k}[3])$ given $\hat{k}[0]$ is well approximated by $3 \times (2^4 \cdot 2^{-1} + 2) \times 2 + 4 = 2^6$ S-Box look-ups, that is approximately $2^{4.6}$ for each nibble $\hat{k}[i]$ for $i = 1, 2, 3$.

F How to Compute the Theoretical "Probability of Success" of the Attacks.

As we have seen, for all the attacks presented in the paper, the attacker needs more texts in order to eliminate false candidates (that is, to make the filter stronger) and to discover (only) the secret key (or, at most, a low number of candidates for the right key). In this section, we discuss how to compute the probability that almost all the false positives candidate for the key are detected and eliminated given n pairs of texts (that is $n+1$ plaintexts) for all the attacks presented in the paper. In other words, we are interested to understand how many plaintexts the attacker needs in order to recover (only) the right key with high probability. For simplicity, we take as example the truncated differential attack on 3 rounds that works independently on each nibble described in Appendix

[11] We would like to emphasize that the right key is always found. However, for all the attacks described in this paper, our implementations show that there are some false positives that pass the test. Thus, it is better to use more plaintexts and to repeat the attack, making the filters stronger.

E.1, but using analogous calculation it is possible to compute the probabilities of success given in the other sections.

Remember that for this attack, one plaintext is in common for all the pairs, and that the attacker needs 4 pairs with the following properties:

1. given a pair (x, y) (where $x, y \in GF(2^4)$), then $x \neq y$, that is $x \oplus y \neq 0$;
2. the pair (x, y) is equivalent to the pair (y, x);
3. given two pairs (x, y) and (x, z), then they have to be different, that is $y \neq z$.

The goal is to compute the probability that given n pairs, there are 4 pairs with different nibbles, i.e. for each $i = 0, ..., 16$ there are 4 different pairs such that $c^j[i] \neq c^k[i]$ for $j \neq k$ and $1 \leq j, k \leq n$.

First of all, remember that $\forall x, y, z \in \mathbb{N}$ such that $x \neq 0$, $y < x$ and $z < x$:

$$\binom{x}{0} = 1 \qquad \binom{x}{y} = \binom{x}{x-y} = \frac{x!}{(x-y)! \cdot y!} \qquad \binom{x}{y} \cdot \binom{x-y}{z} = \binom{x}{z} \cdot \binom{x-z}{y}.$$

Given 4 pairs (i.e. 5 chosen plaintexts), the probability that $c^i[h] \neq c^j[h]$ for each $i \neq j$ and for each nibble h (that is the probability that the attack works) is:

$$p = \left(\frac{15}{16} \cdot \frac{14}{16} \cdot \frac{13}{16} \cdot \frac{12}{16}\right)^{16} = (0.4999)^{16} = 0.0016\%.$$

Given 5 pairs (that is 6 chosen plaintexts), the probability of success is:

$$p = \left[\frac{16!}{11!}\frac{1}{16^5} + \frac{15!}{11!}\binom{5}{2}\frac{1}{16^5}\right]^{16} = (0.812302)^{16} = 3.594\%,$$

since the attacker needs at least 4 pairs with the requested properties.

In particular, the first term of p is for the case in which all pairs are different each others. Observe that in this case the attack works even in the case one pair doesn't satisfy property (1) (this explain the factor 16! instead of 15!).

The second term is for the case in which 2 pairs are equal (that is 2 pairs don't satisfy property (3)) and 4 are different. The binomial coefficient is due to the fact that it is not important which pairs are equal. However, in this case it is very important that all the pairs satisfy property (1).

Given 6 pairs (that is 7 chosen plaintexts), the probability of success is:

$$\begin{aligned} p =& \left[\frac{16!}{10!}\frac{1}{16^6} + \frac{16!}{11!}\binom{6}{2}\frac{1}{16^6} + \frac{15!}{11!}\left(\binom{6}{3} + \frac{1}{2!}\binom{6}{2}\binom{4}{2}\right)\frac{1}{16^6}\right]^{16} = \\ =& (0.939224)^{16} = 36.670\%. \end{aligned}$$

The explanation for the first and for the second term is (essentially) the same as before. About the third term, it is for the case in which 3 pairs are equal (that is the binomial coefficient $\binom{6}{3}$) and for the case in which there are 2 couples of 2 equal pairs, that is 4 pairs such that 2 pairs are equal and the other 2 pairs are equal, but these 2 couples are different each others. The number of possible

combinations for this case is given by $\frac{1}{2!}\binom{6}{2}\binom{6-2}{2}$ (in general, if we consider c couples each one of p pairs, we have to divide by $c!$ and not by 2!).

Proceeding in this way, it is possible to compute the probability for the other cases and for the generic case. Thus, given 8 chosen plaintexts, the probability is

$$p = \Bigg[\frac{16!}{9!}\frac{1}{16^7} + \frac{16!}{10!}\binom{7}{2}\frac{1}{16^7} + \frac{16!}{11!}\left(\binom{7}{3} + \frac{1}{2}\binom{7}{2}\binom{5}{2}\right)\frac{1}{16^7} + \\ + \frac{15!}{11!}\left(\binom{7}{4} + \frac{1}{3!}\binom{7}{2}\binom{5}{2}\binom{3}{2} + \binom{7}{3}\binom{4}{2}\right)\frac{1}{16^7}\Bigg]^{16} = (0.981933)^{16} = 74.698\%,$$

while, given 9 chosen plaintexts, the probability is equal to $(0.994912)^{16} = 92.163\%$.

Attack on 4 Rounds (EE) - Probability of success. In this subsection, we give the probability of success of the attack described in Sect. 5, and which are obtained using the same technique described above (note that in this case the attacker works independently only on 4 nibbles and not 16):

- for 9 chosen plaintexts, the probability is $(0.0604)^4$;
- for 10 chosen plaintexts, the probability is $(0.196)^4$;
- for 11 chosen plaintexts, the probability is $(0.373)^4$;
- for 12 chosen plaintexts, the probability is $(0.5485)^4 \simeq 9.05\%$;
- for 13 chosen plaintexts, the probability is $(0.692)^{16} \simeq 22.93\%$;
- for 14 chosen plaintexts, the probability is $(0.801)^4 \simeq 41.18\%$;
- for 15 chosen plaintexts, the probability is $(0.871)^4 \simeq 57.55\%$;
- for 16 chosen plaintexts, the probability is $(0.923758)^4 \simeq 72.82\%$;
- for 17 chosen plaintexts, the probability is $(0.989733)^4 \simeq 95.96\%$.

G Attack on 4 Rounds - Extension at the Beginning

Another way to attack 4 rounds of PRINCE is to extend the 3-rounds (the middle rounds and one round after) attack described in App. E at the beginning. The idea is to find pairs of plaintexts p^1 and p^2 that belong to the same coset of $\mathcal{C}_0$ after one round, that is:

$$R(p^1) \oplus R(p^2) \in \mathcal{C}_0, \tag{18}$$

in order to repeat the attack on 3 rounds.Consider 4 rounds of PRINCE:

$$p \xrightarrow{R(\cdot)} \hat{q} \xrightarrow{M' \circ \text{S-Box}(\cdot)} q \xrightarrow{\text{S-Box}^{-1}(\cdot)} s \xrightarrow{R^{-1}(\cdot)} c.$$

Given p^1 and p^2, the attacker finds the values of $k_0 \oplus k_1$ for which the condition (18) is satisfied, that is the values of $k_0 \oplus k_1$ such that $R(p^1)$ and $R(p^2)$ belong to the same coset of $\mathcal{C}$. Then, the idea is to repeat the combination of the truncated differential attack and of the subspace attack on 3 rounds described in App. E. However, in this case, when the attacker has found $k_0' \oplus k_1$ and k_1, she has to

check these values with the values of $k_0 \oplus k_1$ that satisfy condition (18). If one combinations is compatible, then the attacker has found the right key, otherwise she has to repeat this procedure.

Since she can not impose any restriction/condition on $k_0 \oplus k_1$, she has to repeat this procedure for each value of $k_0 \oplus k_1$. Thus, for a given pairs of plaintexts p^1 and p^2, the idea is to look for all the keys that satisfy (18). In more details, the idea is to look for p^1 and p^2 in order to minimize the number of nibbles of $k_0 \oplus k_1$ that the attacker has to guess in order to guarantee that $R(p^1) \oplus R(p^2) \in \mathcal{C}_0$, and at the same time to maximize the number of keys such that $R(p^1) \oplus R(p^2) \in \mathcal{C}_0$ for each pair of plaintexts.

Details of the Attack - Estimation of the Data Complexity. The goal to minimize the data complexity becomes easier if the attacker works with a subspace of $\mathcal{C}_0$. In particular, the idea is to consider the subspace $\mathcal{C}_0 \cap \mathcal{ID}_i = \langle e_i \rangle$ of dimension 4 for $i = 0, 1, 2, 3$, where e_i denotes the unit vector of $\mathbb{F}_{2^8}^{16}$ which has a single 1 in the i-th nibble. In the same way as before, it is not difficult to prove that for each $a \in (\mathcal{C}_0 \cap \mathcal{ID}_i)^{\perp}$, there exists unique $b \in (\mathcal{C}_0 \cap \mathcal{IM}_i)^{\perp}$ such that for each $i = 0, ..., 3$

$$M' \circ \text{S-Box}(\mathcal{C}_0 \cap \mathcal{ID}_i \oplus a) = C_0 \cap \mathcal{IM}_i \oplus b,$$

and, since $C_0 \cap \mathcal{IM}_i \subseteq C_0$, that there exists unique $c \in {\mathcal{C}_0}^{\perp}$ such that

$$\text{S-Box}^{-1} \circ M' \circ \text{S-Box}(\mathcal{C}_0 \cap \mathcal{ID}_i \oplus a) \subseteq \mathcal{C}_0 \oplus c.$$

For instance, $\mathcal{C}_0 \cap \mathcal{ID}_0$ and $\mathcal{IM}_0 \cap \mathcal{C}_0$ are defined as:

$$\mathcal{C}_0 \cap \mathcal{ID}_0 \equiv \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathcal{IM}_0 \cap \mathcal{C}_0 \equiv \begin{bmatrix} \alpha_3(x) & 0 & 0 & 0 \\ \alpha_2(x) & 0 & 0 & 0 \\ \alpha_1(x) & 0 & 0 & 0 \\ \alpha_0(x) & 0 & 0 & 0 \end{bmatrix},$$

for each $x \in GF(2^4)$.

In conclusion, for each $a \in (\mathcal{C}_0 \cap \mathcal{ID}_i)^{\perp}$ and for each $i = 0, ..., 3$, there exists unique $d \in \mathcal{IM}_0^{\perp}$ such that

$$R^{(2+0.5)}(\mathcal{C}_0 \cap \mathcal{ID}_i \oplus a) \subseteq \mathcal{IM}_0 \oplus d,$$

where d depends on a and on the secret key. Thus, the attack on 3 rounds works exactly as before using $\mathcal{C}_0 \cap \mathcal{ID}_i$ instead of $\mathcal{C}_0$. The idea is to choose p^1 and p^2 and to ask that that $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_i$ for a certain $i = 0, ..., 3$.

A good choice for p^1 and p^2 is the following:

$$p^1[i] = p^2[i] \qquad \forall i = 3, 4, 5, ..., 15, \tag{19}$$

since in this case the attacker has to guess only 3 nibbles of the key of the first round (that is $(k_1 \oplus k_0)[0], (k_1 \oplus k_0)[1]$ and $(k_1 \oplus k_0)[2]$) in order to guarantee that the condition $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_i$ is satisfied for a certain $i = 0, ..., 3$.

For example, for this choice of pair of plaintexts and for the case of $\mathcal{C}_0 \cap \mathcal{ID}_0$, the condition $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_0$ is equivalent to the following ones:

$$A \wedge 0\text{x}b \oplus B \wedge 0\text{x}d \oplus C \wedge 0\text{x}e = 0,$$
$$A \wedge 0\text{x}d \oplus B \wedge 0\text{x}e \oplus C \wedge 0\text{x}7 = 0,$$
$$A \wedge 0\text{x}e \oplus B \wedge 0\text{x}7 \oplus C \wedge 0\text{x}b = 0,$$

where

$$A = \text{S-Box}(p^1[0] \oplus \tilde{k}[0]) \oplus \text{S-Box}(p^2[0] \oplus \tilde{k}[0]),$$
$$B = \text{S-Box}(p^1[1] \oplus \tilde{k}[1]) \oplus \text{S-Box}(p^2[1] \oplus \tilde{k}[1]),$$
$$C = \text{S-Box}(p^1[2] \oplus \tilde{k}[2]) \oplus \text{S-Box}(p^2[2] \oplus \tilde{k}[2]),$$

and where $\tilde{k}[i]$ is defined as $\tilde{k}[i] := k_1[i] \oplus k_0[i]$.

A possible solution of the previous equivalences is $A = B = C = 0\text{x}1$. Observe that for $A = B = C = 0\text{x}1$ the previous equivalences are completely equivalent to S-Box $(x) \oplus$ S-Box $(x \oplus \alpha) = 0\text{x}1$, where $p^1[\cdot] \oplus p^2[\cdot] = \alpha$ and $x = p^1[\cdot] \oplus \tilde{k}[\cdot]$. By computer test, we have found that if $\alpha = 0\text{x}1$ or $\alpha = 0\text{x}c$ there exists 4 possible solutions of x for the previous equations, which are $x = 0\text{x}2, 0\text{x}3, 0\text{x}8, 0\text{x}9$ for $\alpha = 0\text{x}1$, and $x = 0\text{x}6, 0\text{x}7, 0\text{x}a, 0\text{x}b$ for $\alpha = 0\text{x}c$.

Using these values to choose p^1 and p^2, for each combination $(p^1[0], p^2[0], ..., p^2[2])$ there are 2^6 possible combinations of $(\tilde{k}[0], \tilde{k}[1], \tilde{k}[2])$ such that $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_0$. Consequently, $(2^4)^3 \times 2^{-6} = 2^6$ possible combinations of $(p^1[0], p^2[0], ..., p^2[2])$ are enough for this attack.

However, a lower number of combinations is sufficient. Indeed, given a pair of plaintexts p^1 and p^2, the idea is to look for all the keys such that $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_i$ for each $i = 0, ..., 3$ and not only for $\mathcal{C}_0 \cap \mathcal{ID}_0$. By computer test, we have found that that the best situation occurs when $p^1[j] \oplus p^2[j] = 0\text{x}1$ for $j = 0, 1, 2$. In particular, we have found that:

- $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_1$ if $A = B = C = 0\text{x}8$ and there exists 4 possible solutions of x if $\alpha = 0\text{x}1$;
- $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_2$ if $A = B = C = 0\text{x}4$ and there exists 4 possible solutions of x if $\alpha = 0\text{x}8, 0\text{x}b$, while for $\alpha = 0\text{x}1$ there are only 2 solutions;
- $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_3$ if $A = B = C = 0\text{x}2$; in this case there isn't a value of α for which there are 4 solutions, and for $\alpha = 0\text{x}1$ there isn't any solution.

Thus, given $p^1[j] \oplus p^2[j] = 0\text{x}1$ for each $j = 0, 1, 2$, there are $2 \cdot 4^3 + 2^3 = 136 = 2^{7.09}$ different keys such that $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_i$ for $i = 0, ..., 3$ That is, on average $2^{12} \cdot 2^{-7.09} = 2^{4.91}$ possible combinations of $(p^1[0], p^2[0], ..., p^2[2])$ are enough for this attack. In particular, for each pair p^1 and p^2 there are on average $2^{7.09}$ keys such that $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_i$ for a certain $i = 0, ..., 3$.

For each of these combinations of $p^1[i]$ and $p^2[i]$ for $i = 0, 1, 2$, the attacker chooses 7 pairs of plaintexts that satisfy (19), and the attack is equivalent to the one on 3 rounds described in App. E.

Data: 430 chosen plaintexts - the choice of this plaintexts is described in details in the text ciphertexts (in particular, for each pair p^1 and p^2 such that $p^1[j] \oplus p^2[j] = 0\text{x}1$ for each $j = 0, 1, 2$, there are on average $2^{7.09}$ combinations of keys $(\tilde{k}[0], \tilde{k}[1], \tilde{k}[2])$ such that $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_i$ for a certain $i = 0, ..., 3$.).
Result: Secret key k_0 and k_1.
for all 2^{12} *combinations of keys* $(\tilde{k}[0], \tilde{k}[1], \tilde{k}[2])$ **do**
  for the guessed key, pick up the relative different pairs of chosen plaintexts stored in memory (and corresponding i) - as described in details in the text
  3-*Rounds Truncated Differential Attack* that exploits relationships among nibbles (see Algorithm 3): identify candidates for $k_1 \oplus k'_0$ and k_0
  check *key schedule* conditions using the three nibbles of the guessed key $k_1 \oplus k_0$- see details at the end of this section;
end
return Secret key k_0 and k_1 that satisfy key schedule.

Algorithm 4: *Truncated differential attack on 4 rounds of PRINCE - extension at the beginning.*

Details of the Attack - Estimation of the Computational Cost. Using these 7 pairs of plaintexts, the attacker finds the first and the second columns of $k_1 \oplus k'_0$ (cost of $2 \times (2^{5.5} + 2^6)$ S-Box look-ups - see combination of truncated differential attack and subspace one). Then, using the particular shape of $\mathcal{C}_0 \cap \mathcal{IM}_i$, the attacker can also discover 2 nibbles of k_1, that is $k_1[0]$ and $k_1[1]$ (cost of $2^{5.5} + 2^{4.5}$ S-Box look-ups) using the truncated differential attack in a similar way as before - note that $\mathcal{C}_0 \cap \mathcal{IM}_i$ is a subspace of $\mathcal{IM}_i$ for $i = 0, ..., 3$, i.e. the first column of $\mathcal{C}_0 \cap \mathcal{IM}_i$ is equal to the first column of $\mathcal{IM}_i$. The computational cost of this step is of $(2^{5.5}+2^6)\times 2+2^{5.5}+2^{4.5}+14\times 8 = 2^{8.55}$ S-Box look-ups. Remember that the attacker has to repeat this step for each possible combinations of $\tilde{k}[0], \tilde{k}[1]$ and $\tilde{k}[2]$ (equivalently for each of the $2^{4.91}$ possible combinations of $(p^1[0], ..., p^2[2])$), since she cannot impose any restrictions to the secret key.

In order to improve the total computational cost of the attack, before to find the other columns of $k_1 \oplus k'_0$, the idea is to check if these found values are compatible with those of $k_1 \oplus k_0$. For each combination of $(p^1[0], ..., p^2[2])$, the attacker knows 8 nibbles of $k_1 \oplus k'_0$, $2^{7.09}$ possible values of 3 nibbles of $k_1 \oplus k_0$ and 2 nibbles of k_1. As we show in the following, the probability that these values agree is 2^{-7} (the attacker can works with two nibbles, but she cannot use one of the bit, so 2^{-7}). Thus, since there are $2^{7.09}$ possible values of $k_1 \oplus k_0$ for each combinations of (p^1, p^2), on average only $2^{7.09} \times 2^{4.91} \times 2^{-7} = 2^5$ combinations of keys survived this first check. (indeed note that if they are not compatible, then they are certainly wrong). Observe that the cost of this last check is negligible (since no S-Box is involved, only Shift and XoR).

For the survived combinations, the attacker finds the third column of $k_1 \oplus k'_0$ (cost of $2^{5.5} + 2^6$ S-Box look-ups) and the nibble $k_1[3]$ (cost of $2^{4.5}$ S-Box look-ups). In this way, the probability that all the values of the keys are compatible becomes 2^{-11}, that is on average only 2 combinations of keys survived this second

check. For this second step, the computational cost is $(2^{5.5}+2^6)+2^{4.5}+14\times 4 = 2^{7.5}$ S-Box look-ups for each combinations.

Finally, for both these two combinations, the attacker finds the last column of $k_1 \oplus k'_0$ and the nibble $k_1[4]$ with a computational cost of $2^{7.5}$ S-Box look-ups for each combinations. However, in this case, the attacker cannot discover which of these two combinations is the correct one. Indeed, she needs to know all the other nibbles of k_1 to discover the right combination (working only 3 nibbles of $k_1 \oplus k_0$, the attacker can use only 11 bits to do the check).

Thus, the attacker has to find the values of the other 3 columns of k_1. Observe that she cannot use the previous plaintexts, since the corresponding $q := \text{S-Box}(R(c))$ belongs to $\mathcal{C}_0 \cap \mathcal{IM}_i$. Thus the simplest idea is to choose other 8 plaintexts p that belong to the same coset of $\mathcal{C}_0$. Using the knowledge of the key of the final round, the attacker can easily compute $s = R(p)$ and then she can easily perform the attack on 3 rounds described in App. E and discover k_1 (observe that q belongs to $\mathcal{IM}_0$ since $p \in \mathcal{C}_0$). In this way, she can finally discover the right key among the two final survived combinations. The total computational cost for this step is approximately $8 \times 2^4 + 3 \times (2^5 + 2^6) = 2^{8.8}$ S-Box look-ups for each combinations.

In conclusion, for this attack, the total number of plaintexts that the attacker needs is $14 \times 2^{4.91} + 8 = 430 = 2^{8.75}$, and the total computational cost is $2^{4.91} \times 2^{8.55} + (2^5 + 1) \times 2^{7.5} + 2 \times 2^{8.8} \simeq 2^{14.15}$ S-Box look-ups, that is about $2^{8.15}$ four-rounds Encryption.

How to do the check? Suppose that the attacker knows 8 nibbles of $k_1 \oplus k'_0$ (which are $(k_1 \oplus k'_0)[i]$ for $i = 0, ..., 7$), $2^{7.09}$ possible values of 3 nibbles of $k_1 \oplus k_0$ (which are $(k_1 \oplus k_0)[i]$ for $i = 0, 1, 2$) and 2 nibbles of k_1 (which are $k_1[0]$ and $k_1[0]$). How can she check that all these nibbles are compatible (that is, they satisfy the key schedule)?

We denote the ith bit of k_0 by k_0^i. By definition:

$$k'_0 \equiv (k_0 \ggg 1) \oplus (k_0 \gg 63) = k_0^1 \oplus k_0^{63}, k_0^2, k_0^3, k_0^4, ..., k_0^{63}, k_0^0$$

that is:

$$k_0 \oplus k'_0 = k_0^0 \oplus k_0^1 \oplus k_0^{63}, k_0^1 \oplus k_0^2, k_0^2 \oplus k_0^3, k_0^2 \oplus k_0^4, ..., k_0^{62} \oplus k_0^{63}, k_0^0 \oplus k_0^{63}.$$

Since the attacker knows $(k_0 \oplus k'_0)[i]$ for $i = 0$ and 1. In particular, 7 bits (i.e. $k_0^0 \oplus k_0^{63}$ and k_0^i for $i = 2, ..., 7$) depend of k_0^1, that is $k_0^0 \oplus k_0^{63} = k_0^0 \oplus k_0^{63}(k_0^1)$ and $k_0^i = k_0^i(k_0^1)$, where $i = 2, 3, ..., 7$. Using the knowledge of the nibbles of k_1, $k_0 \oplus k_1$ and $k_0(k_0^1)$, the attacker checks if $k_1^i = (k_0^i \oplus k_1^i) \oplus k_0^i(k_0^1)$, for $i = 0, 2, ..., 7$. The probability that this condition is verified is 2^{-7} (remember that k_0^1 is unknown).

H Attacks on 5 and 6 Rounds of PRINCE

Using both the extensions at the end and the beginning, it is possible to attack 5 rounds of PRINCE. Moreover, it is also possible to attack 6 rounds of PRINCE using two extensions at the beginning and one extension at the end. However, these two attacks are not better than others present in the literature.

Table 2. *Comparison table of attacks on round-reduced PRINCE.* Data complexity is measured in number of required chosen plaintexts (CP). Time complexity is measured in round-reduced PRINCE encryption equivalents (E). Memory complexity is measured in plaintexts (64 bits). The attacks of this paper are in bold.

Technique	Rounds	Data (CP)	Computation (E)	Memory	Reference
Integral	5	$80 = 2^{6.32}$	2^{64}	2^8	[13]
Integral	5	$96 = 2^{6.6}$	$2^{22.7}$	small	[18]
Our attack	**5**	$\mathbf{2^{8.7}}$	$\mathbf{2^{25}}$	**small**	**App. H.1**
Integral	5	2^{13}	$2^{14.7}$	small	[19]
Integral	6	$2^{14.6}$	$2^{30.42}$	small	[19]
Diff. / Logic	6	$2^{14.9}$	$2^{32.9}$	$\ll 2^{27}$	[9]
MitM	6	2^{16}	$2^{33.7}$	$2^{31.9}$	[9]
Integral	6	2^{16}	2^{64}	2^{16}	[13]
Integral	6	$2^{18.6}$	$2^{34.4}$	small	[18]
Our attack	**6**	$\mathbf{2^{27.8}}$	$\mathbf{2^{43.9}}$	**small**	**App. H.2**

(EE: Extension at End - EB: Extension at Beginning - MitM: Meet-in-the-Middle)

H.1 Attack on 5 Rounds

We have seen how to extend the 3-rounds attack at the beginning and at the end. In order to attack 5 rounds, the idea is to use both the extensions. In particular, we consider the 2+0.5 rounds subspace trails, that is we extend at the end the attack on 4 rounds described in Sect. G. That is, this 5 rounds attack involves 2 rounds after the middle rounds and 1 before them. The attack on 6 rounds is then obtained by extending this attack on 5 rounds at the beginning.

Consider the following situation:

$$p \xrightarrow{R} \tilde{p} \xrightarrow{super\text{-}SBox} s \xrightarrow{R^{-1}} \tilde{c} \xrightarrow{R^{-1}} c.$$

As before, the idea is to consider a pair of plaintexts p^1 and p^2, and to find for which key of the first round the condition $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_i$ is satisfied. The pairs of plaintexts have to be chosen as described in Sect. G in order to minimize the number of nibbles that the attacker has to guess in order to guarantee that $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_i$. Given these pairs of plaintexts, the idea is to repeat the attack on 4-rounds (with the extension at the end) described in Sect. 5.

In particular, using 7 pairs of chosen plaintexts such that $R(p^i) \oplus R(p^j) \in \mathcal{C}_0 \cap \mathcal{ID}_i$ for the same set of keys, the attacker guesses 4 nibbles (that is 1 column) of the key of the final rounds, finds 4 nibbles of k_1 and checks if they agree with $k_1 \oplus k'_0$. She repeats this step for all the 3 columns, that is the attacker finds all the nibbles of $k_1 \oplus k'_0$ and of k_1.

When the attacker finds the key of the final rounds ($k_1 \oplus k'_0$ and k'_1), she has to check if they are compatible with the keys of the first round $k_1 \oplus k_0$ that satisfy the condition $R(p^1) \oplus R(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_i$ for certain i. If they are compatible, then

the attacker has found the right key, otherwise she has to repeat the previous procedure.

Observe that the attacker certainly discovers the secret key, since the number of possible combinations of $k_1 \oplus k_0$ is 2^{12}, while the probability that the keys of the final rounds and the key of the first round agree is $(2^{-4})^3 = 2^{-12}$.

For this attack, the number of plaintexts that the attacker needs is $2^{4.91} \times 2 \times 7 = 2^{8.72}$, and the total computational cost is approximately $2^{31.3}$ S-Box look-ups, that is about 2^{25} five-rounds Encryption.

H.2 Attack on 6 Rounds

In order to attack 6 rounds, the idea is to start from the previous attack on 5 rounds and to extend it again at the beginning, that is, starting from the attack on 3 rounds (which exploits the 2+0.5 rounds subspace trails), the idea is to extend it two times at the beginning and one at the end. For simplicity, suppose to work only with $\mathcal{C}_0 \cap \mathcal{ID}_0$.

In particular, consider the following situation:

$$p \xrightarrow{R} \hat{p} \xrightarrow{R} \tilde{p} \xrightarrow{super\text{-}SBox} s \xrightarrow{R^{-1}} \tilde{c} \xrightarrow{R^{-1}} c,$$

where $\tilde{p} \in \mathcal{C}_0 \cap \mathcal{ID}_0 \oplus a$ (and where $a \in (\mathcal{C}_0 \cap \mathcal{ID}_0)^{\perp}$). In order to guarantee that $R^{(2)}(p^1) \oplus R^{(2)}(p^1) \in \mathcal{C}_0 \cap \mathcal{ID}_0$ (so that $R(p^1)$ and $R(p^2)$ satisfy the condition (19)) and with the goal to minimize the number of nibbles that the attacker has to guess, we consider pairs of plaintexts p^1 and p^2 such that

$$p^1[i] = p^2[i] \qquad \forall i = \{3, 5, 8, 12, 13, 14, 15\}.$$

With this choice of pairs of plaintexts, the attacker has to guess 9 nibbles of $k_0 \oplus k_1$ and 3 nibbles of k_1.

In particular, given $k_0 \oplus k_1$ and k_1, we show how to choose the first column (it is completely equivalent for the second and for the third one) of the pairs of plaintexts p^1 and p^2, in order to minimize the total number of chosen plaintexts. As before, remember that the attacker has to repeat this procedure for each possible values of the 9 nibbles of $k_0 \oplus k_1$ and of the 3 nibbles of k_1, since she can not impose any restriction on the secret key.

About the the first column (analogous for the others), the 3 nibbles of $\tilde{k} = k_0 \oplus k_1$ and the pair (p^1, p^2) have to satisfy the same conditions given in App. G:

$$\begin{aligned} A \wedge \text{0x}b \oplus B \wedge \text{0x}d \oplus C \wedge \text{0x}e &= \text{0x0}, \\ A \wedge \text{0x}d \oplus B \wedge \text{0x}e \oplus C \wedge \text{0x7} &= \text{0x0}, \\ A \wedge \text{0x}e \oplus B \wedge \text{0x7} \oplus C \wedge \text{0x}b &= \text{0x0}, \end{aligned}$$

where $A = \text{S-Box}\ (p^1[0] \oplus \tilde{k}[0]) \oplus \text{S-Box}(p^2[0] \oplus \tilde{k}[0])$, $B = \text{S-Box}\ (p^1[1] \oplus \tilde{k}[1]) \oplus$ S-Box $(p^2[1] \oplus \tilde{k}[1])$, and $C = \text{S-Box}\ (p^1[2] \oplus \tilde{k}[2]) \oplus \text{S-Box}(p^2[2] \oplus \tilde{k}[2])$.

As we have seen, a solution for the previous equivalences is given by $A = B = C = 0\text{x}1$. For this solution, the difference in the first nibble (that is the nibble in position 0) after one round, is 0x1:

$$A \wedge 0\text{x}7 \oplus B \wedge 0\text{x}b \oplus C \wedge 0\text{x}d = 0\text{x}1,$$

that is after one round:

$$\hat{p}^1[0] \oplus \hat{p}^2[0] = 0\text{x}1, \qquad \hat{p}^1[7] = \hat{p}^2[7], \qquad \hat{p}^1[10] = \hat{p}^2[10], \qquad \hat{p}^1[12] = \hat{p}^2[12],$$

where $\hat{p}^h = R(p^h)$ for $h = 1, 2$. Observe that the attacker doesn't know $\hat{p}^1[0]$ and $\hat{p}^2[0]$, since she doesn't know $\tilde{k}[3]$ (note that she hasn't guessed it).

Then, the attacker needs that $R(\hat{p}^1) \oplus R(\hat{p}^2) \in \mathcal{C}_0 \cap \mathcal{ID}_0$ (more generally, in $\mathcal{C}_0 \cap \mathcal{ID}_i$ for some i). The conditions that the 3 nibbles of k_1 have to satisfy are exactly the same of those given in Sect. G.

Note that since she doesn't know $\hat{p}^h[i]$ for $h = 1, 2$ and $i = 0, 1, 2$, she cannot guess the three nibbles of k_1. Anyway, since $\hat{p}^1[i] \oplus \hat{p}^2[i] = 0\text{x}1$ for each $\hat{p}^h[i]$ for $h = 1, 2$ and $i = 0, 1, 2$, we can guarantee that there are 4 solutions for each of the previous equivalence (as we have seen in details before).

Since the attacker can not impose any restriction/condition on $k_1[0], k_1[1], k_1[2]$, the idea is to take for each combination of $p^1[0], p^2[0], ..., p^2[2]$ at least 4 chosen values of $p^1[3] = p^2[3]$ (and analogous for the other columns). In this way, the values of $\hat{k}$ that satisfy the first equivalences don't change, while the values $k_1[0], k_1[1], k_1[2]$ that satisfy the previous equivalences are different. Thus all the values of $k_1[0], k_1[1], k_1[2]$ are taken.

With these previous observations, the attack is completely equivalent to the previous one. In particular, suppose to consider a combination of $p^1[0], p^2[0], ..., p^2[2]$ and a given $p^1[3] = p^2[3]$ (analogous for the other columns). In this case, there are $(2^4)^9 = 2^{36}$ possibilities for the nine nibbles of $\tilde{k}$ and $(2^4)^3 = 2^{12}$ possibilities for the three nibbles of k_1 (that the attacker doesn't know) such that $R^{(2)}(p^1) \oplus R^{(2)}(p^2) \in \mathcal{C}_0 \cap \mathcal{ID}_0$.

As before, the attacker takes 16 pairs of chosen plaintexts for this particular combination. Note that the attacker has to work on $p^1[12] = p^2[12], ..., p^1[15] = p^2[15]$ and not on $p^1[3] = p^2[3], p^1[5] = p^2[5], p^1[8] = p^2[8]$. Using these 16 pairs of chosen plaintexts, the attacker can discover $k_1 \oplus k_0'$ and k_1. Then she checks if k_1 satisfies the condition $R(\hat{p}^1) \oplus R(\hat{p}^2) \in \mathcal{C}_0 \cap \mathcal{ID}_0$ and checks if k_1 and $k_1 \oplus k_0'$ are compatible with $k_1 \oplus k_0$. If they agree, she has found the right key, otherwise she has to repeat this step.

It is important that the attacker repeats this step using the same combination of $p^1[0], p^2[0], ..., p^2[2]$, but using the other 3 possible values of $p^1[3] = p^2[3]$ (and of $p^1[5] = p^2[5]$, $p^1[8] = p^2[8]$ for the other columns). In this way, all the values of $k_1[1], k_1[1], k_1[2]$ are taken. As for the previous attacks, only one key satisfies all the checks. Indeed, observe that the probability that all the conditions are satisfied is $(2^4)^{-12}$, while the number of possible keys of $k_1 \oplus k_0$ and k_1 is $(2^4)^{12}$.

For this attack, the number of plaintexts that the attacker needs is $2^{24} \times 2 \times 7 = 2^{27.8}$, and the total computational cost is approximately $2^{50.4}$ S-Box look-ups, that is about $2^{43.9}$ six-rounds Encryption.