Faster Key Recovery Attack on Round-Reduced PRINCE

Shahram Rasoolzadeh and Håvard Raddum

Simula Research Laboratory

Abstract. We introduce a new technique for doing the key recovery part of an integral or higher order differential attack. This technique speeds up the key recovery phase significantly and can be applied to any block cipher with small S-boxes. We show several properties of this technique, then apply it to PRINCE and report on the improvements in complexity from earlier integral and higher order differential attacks on this cipher. Our attacks on 4 and 6 rounds were the fastest and the winner of PRINCE Challenge's last round in the category of chosen plaintext attack.

Keywords: PRINCE, lightweight, block cipher, key recovery attack, integral, higher-order differential.

1 Introduction

PRINCE is a lightweight block cipher that was introduced in [1]. The cipher has received a fair share of attention from cryptanalysts in the last years, with many different attacks on round-reduced versions [3–15]. The Prince Challenge website tracks the best attacks and promises cash prizes for attacks that will have a serious impact in real-world applications [2].

In this paper we will revisit two earlier works [12, 13] on PRINCE and improve the complexities of the attacks described there. Both papers are concerned with integral attacks, with [12] also giving details of a bit-pattern integral attack on 4 rounds and a higher order differential attack on 7 rounds of PRINCE. We re-use the integral distinguishers described in [4, 13].

The improvement we can do lies in the key recovery part. We introduce a new technique to do key recovery, using a binary array. For PRINCE, the size of this array is only 16 bits. Using this technique we can skip the partial trial decryptions to check for balancedness in an integral or higher order differential attack. This technique can be applied to speed up this type of attacks on any block cipher with small S-boxes. In addition to using this technique for key recovery we also apply the accelerated key search technique described in [14, 15] when partial keys need to be guessed.

The improvements and comparisons to previous attacks are summarized in Table 1. All time complexities are given in terms of number of r-round PRINCE encryptions based on counting the number of S-box look-ups needed. The data

complexities are given as the number of chosen plaintext/ciphertext pairs needed. The results from [12, 13] have been transformed into this format, and in some cases slightly corrected, to get a correct comparison.

Table 1. Summary of cryptanalytic results on PRINCE

Rounds	Time	Data	Memory	Technique	Ref.
4	2^{64}	2^4 CP	2^4	Integral	[4]
	$2^{43.4}$	2^5 KP	$2^{26.7}$	Diff./Logic	[11]
	5 sec.	2^{10} CP	$\ll 2^{27}$	MitM	[11]
	$15 \cdot 2^{20}$	$3 \cdot 2^4$ CP	$3 \cdot 2^4$	Integral	[12]
	$2^{9.7}$	$5 \cdot 2^5$ CP	$5 \cdot 2^4$	Integral	[12]
	$2^{7.4}$	2^6 CP	negl.	Integral	Sec. 5.1
5	2^{64}	$5 \cdot 2^4$ CP	2^4	Integral	[4]
	$3 \cdot 2^{23}$	$3 \cdot 2^5$ CP	$3 \cdot 2^5$	Integral	[12]
	$2^{21.4}$	2^5 CP	2^5	Integral	Sec. 5.2
	2^{13}	2^{13} CP	2^5	Integral	Sec. 5.2
6	$2^{101.1}$	2^6 KP	2^{34}	MitM	[11]
	$2^{96.8}$	2 KP	negl.	Acc. Exh.	[15]
	$2^{86} + 2^{86} M.A.$*	2 KP	$2^{24.6}$	Acc. Exh.	[15]
	2^{64}	2^{16} CP	2^{16}	Integral	[4]
	$2^{33.7}$	2^{16} CP	$2^{31.9}$	Mitm	[11]
	$5 \cdot 2^{30}$	$3 \cdot 2^{13}$ CP	$3 \cdot 2^{13}$	Integral	[13]
	$2^{28.9}$	$2^{14.9}$ CP	$\ll 2^{27}$	Diff./Logic	[11]
	$5 \cdot 2^{34}$	$3 \cdot 2^{17}$ CP	$3 \cdot 2^{17}$	Integral	[12]
	$2^{24.6}$	2^{13} CP	2^{13}	Integral	Sec. 5.3
7	$2^{52.1}$	$3 \cdot 2^{33}$ CP	$3 \cdot 2^{33}$	H.-O. Diff.	[12]
	$2^{44.3}$	2^{33} CP	2^{33}	H.-O. Diff.	Sec. 5.4
8	2^{124}	2 KP	2^{20}	SitM	[6]
	$2^{122.7}$	2 KP	negl.	Acc. Exh.	[14]
	$2^{109.3}$	2 KP	2^{65}	MitM	[14]
	$2^{66.3}$	2^{16} CP	$2^{49.9}$	MitM	[11]
	$2^{65.7}$**	2^{16} CP	$2^{68.9}$	MitM	[11]
	2^{60}	2^{53} CP	2^{30}	MitM	[7]
	$2^{50.7}$**	2^{16} CP	$2^{84.9}$	MitM	[11]
9	2^{64}	2^{57} CP	$2^{57.3}$	MitM	[7]
	$2^{51.2}$	$2^{46.9}$ CP	$2^{52.2}$	Multiple Diff.	[8]
10	2^{124}	2 KP	negl.	Acc. Exh.	[14]
	$2^{122.2}$	2 KP	$2^{53.3}$	MitM	[14]
	2^{68}**	2^{57} CP	2^{41}	MitM	[11]
	$2^{60.6}$	$2^{57.9}$ CP	$2^{61.5}$	Multiple Diff.	[8]

* Memory Access to a table with 2^{25} indexes.
** Online Time

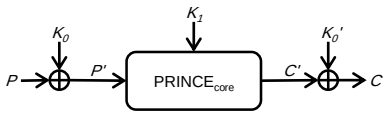

Fig. 1. PRINCE FX Construction

2 PRINCE Block Cipher

PRINCE is an FX-constructed lightweight block cipher with block size of 64 bits [1]. Two keys are used in PRINCE, both of length 64 bits, one for whitening (K_0) and the other as a round key (K_1) for the core of the structure (see Figure 1). The round key is used in every round without any key schedule, and the whitening key before the ciphertext (K_0') is derived by applying a simple linear mapping to K_0.

The PRINCE$_{core}$ is an AES-like block cipher that employs an involutive 12-round structure. PRINCE$_{core}$ starts with two *xor*s with K_1 and a round constant, followed by 5 forward rounds, a middle layer, 5 backward rounds and at the end, two more *xor*s with a round constant and K_1. Figure 2 shows the schematic view of PRINCE$_{core}$.

The state is defined as a 4×4 matrix similar to AES, but in PRINCE, instead of bytes the cells contain nibbles. Each round of PRINCE$_{core}$ consists of 5 operations: S-box, mix column, shift row, round constant addition and key addition. These are described as follows:

- **S-box** (SB): Every nibble in the state is replaced using a 4-bit S-box.
- **Mix Column** (MC): The state is multiplied with an involutive 64×64 binary matrix. More precisely, this large matrix can be expressed as sixteen 4×4 matrices where each of these mixes four bits in one column of the state.
- **Shift Row** (SR): Row i of the state is cyclically rotated by i positions to the left (same as shift row operation in AES).
- **Round Constant Addition** (RC): A bit-wise *xor*ing with a round constant RC_i , $i = 0, ..., 11$.
- **Key Addition** (AK): A bit-wise *xor*ing with the key K_1.

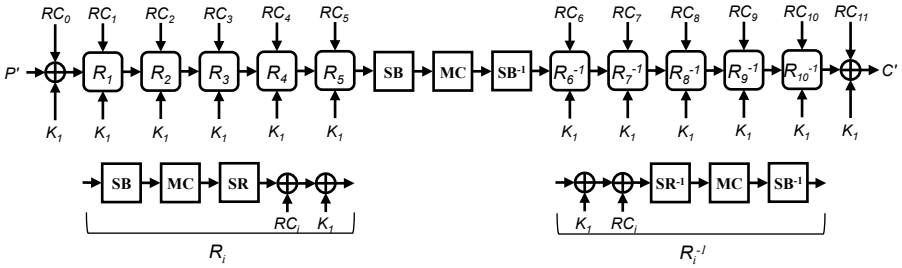

Fig. 2. PRINCE core

The two middle rounds contain only three layers, SB, MC and SB^{-1} which makes it an involutive transformation. This transformation can also be separated into four smaller transformations, one for each column in the state.

In the backward rounds, the operations come in the reverse order of the forward rounds, and SB and SR are replaced with SB^{-1} and SR^{-1}. The round constants are also different, but related to the round constants in the forward rounds. The difference $RC_i \oplus RC_{11-i}, i = 0, ..., 11$ is always equal to the constant value $\alpha =$ 0xc0ac29b7c97c50dd.

3 Integral and Higher-Order Differential Distinguishers for PRINCE

In this section we will briefly introduce integral and higher-order differential attacks. For each of the attacks we will revisit two integral and one higher-order differential distinguisher for PRINCE that are used in previous attacks [4, 12, 13].

3.1 Integral Distinguishers

The integral or square attack was originally designed as a dedicated attack in [16] against the Square block cipher. This cryptanalytic attack is particularly applicable to block ciphers that use S-boxes. Integral cryptanalysis uses sets of chosen plaintexts, where typically most parts of the plaintexts are set to a constant (constant parts) and some parts vary through all possible values (active parts). Then, the cryptanalyst studies how the *xor* sum in the given parts changes through the operations of the cipher. After a few rounds, the cipher states still sum up to zero over one set (balanced state). This property will distinguish a given cipher from a random permutation and can be used for key recovery.

3.5-round integral distinguisher for PRINCE The 3.5-round integral distinguisher for PRINCE first used in [4] covers one forward round, two middle rounds and one backward round except its SB^{-1} operation. In this distinguisher we use 2^4 plaintexts which only differ in one nibble and the other 15 nibbles are constant. When one S-box takes all its 2^4 possible inputs and the inputs for all other S-boxes are constant, the states after the above 3.5 rounds (state right before the last SB^{-1} operation) will be balanced, i.e. the *xor* sum of these states will be equal to zero.

4.5 round integral distinguisher for PRINCE The 4.5-round integral distinguisher for PRINCE introduced by Posteuca and Negara in [13] contains two forward rounds, two middle rounds and one backward round except its SB^{-1} operation. In this distinguisher we use 2^{12} plaintexts which only differ in three nibbles in the same column and the other 13 nibbles are constant. When three

S-boxes in a column take all their 2^{12} possible inputs and the input for all other S-boxes are constant, the state after 4.5 rounds will be balanced.

For explanations for why these sets are balanced after 3.5 and 4.5 rounds we refer to [4, 12, 13].

3.2 Higher-Order Differential Distinguisher

The higher-order differential attack is a generalization of differential cryptanalysis. While in a differential attack the difference between only two plaintexts is used, higher-order differential attack studies the propagation of a set of differences between a larger set of plaintexts. Lai, in 1994, laid the groundwork by showing that differentials are a special case of the more general case of higher order derivatives [17] and Knudsen, in the same year, was able to show how the concept of higher order derivatives can be used to mount attacks on block ciphers [18].

Higher-order differential attacks are applicable to ciphers where the bits of the cipher state at some point can be represented as Boolean polynomials of a low algebraic degree. In PRINCE the only non-linear operation is the SB, so the algebraic degree of the output of one round is three. Using this property of PRINCE, in [12] one 5.5-round higher-order differential for PRINCE is presented. This distinguisher calculates the i-th derivative at some selected state and uses a set of 2^i plaintexts, where i plaintext bits vary over all possible values, while the rest of the state is set to an arbitrary constant.

5.5-round higher-order differential distinguisher for PRINCE The expression of state variables after 3 SB layers have algebraic degree at most 3^3. Therefore, any 28-th or higher order derivative of the state must be zero. The distinguisher uses two more rounds with no cost in algebraic degree to arrive at a 5.5-round distinguisher.

The distinguisher uses 2^{32} chosen plaintexts, where two columns take all possible input values. As the S-box is bijective, the first SB operation preserves the property that the two selected columns take all 2^{32} possible values. In the next step MC works on columns independently, thus still there are 32 state bits taking all possible combinations. The SR operation will move constant value nibbles into columns with all-valued nibbles, so the MC operation in the second round destroys the property.

Therefore, the distinguisher gets the first two SB layers for free and then it covers another three SB and SB^{-1} operations. So, it gives a balanced state after 5.5 rounds (the state right before SB^{-1} in the sixth round).

4 New Technique for Key Recovery

Assume that the S-boxes used in the target block cipher is n bits. Let A be a 2^n-bit binary array, $A = [a_0, a_1, \ldots, a_{2^n-1}]$. For any such array, we define K_A to

be the following set of n-bit values:

$$K_A = \{k \in GF(2)^n | \bigoplus_{i=0}^{2^n-1} a_i \cdot S(k \oplus i) = 0\} \quad (1)$$

For example, for the PRINCE S-box where $n = 4$, if $A = [1, 1, 1, 0, 1, 0, \ldots, 0]$, the only solutions for

$$S(k) \oplus S(k \oplus 1) \oplus S(k \oplus 2) \oplus S(k \oplus 4) = 0$$

is $K_A = \{\mathsf{c}\}$.

The computation cost for finding the corresponding K_A for an array of A is $w_A \times 2^n$ S-box evaluations, where w_A denotes the Hamming weight of array A. This is because for each n-bit value of key we have to compute one S-box look-up for each set bit in A.

Assume that we want to attack an R-round cipher using an $(R - 0.5)$-round distinguisher which says that for a set of 2^d chosen plaintexts, the *xor* sum of the cipher states after $(R-0.5)$ rounds is equal to zero. After these $(R-0.5)$-rounds there is an S-box layer before reaching the ciphertext.

The usual key recovery method would be to guess the 2^n possible values for each of the last round key words in the output of an S-box, and then partially decrypt through the SB operation for every ciphertext. If the xor sum of these 2^d nibbles are equal to zero we accept the guessed value of subkey as a candidate and if not we reject it. The time complexity for finding key candidates for one n-bit word of the last round key using one set of 2^d ciphertexts is equal to $2^n \times 2^d$ S-box evaluations.

In the following we introduce our technique which is faster than the straightforward method. In our technique we will build an A array for each word in the state from the 2^d ciphertexts. Then for each array, we will find key candidates for the corresponding word of last round key.

At the start of the attack, for each word of state we allocate a 2^n-bit array A initialized to all zeroes. Then we look at the corresponding word in each of the 2^d ciphertexts. When the value of this word is equal to x, we will flip the x-th bit in the corresponding A. After doing this for all the 2^d ciphertexts, we can just find the corresponding set K_A for the created array. The values in K_A are the key candidates for this word of last round key. So for a set of 2^d ciphertexts, instead of 2^{d+n} S-box evaluations, our key recovery method will need about $n_{SB} \times 2^n$ S-box evaluations to find the candidates for each word of the last round key, where n_{SB} denotes the average number of S-box evaluations for the created arrays.

Compared with the usual key recovery method, using A arrays is faster, which helps to reduce the complexity of integral or higher-order differential attacks. Specially when size of data sets are big (d is large), the speed-up factor $\frac{2^d}{n_{SB}}$ of our technique gets bigger.

4.1 Some Features of A Arrays

Here we introduce some facts about possible (A, K_A) values which help us to evaluate the average number of S-box operations needed n_{SB} and also make it smaller.

Lemma 1. *In an actual attack, the weight of A is always even.*

Proof: The set of ciphertexts is of size 2^d, and hence even. We flip exactly one bit in A for each ciphertext, so starting from the all-zero array the weight will always be even after processing an even number of ciphertexts.

Lemma 2. *If $A = [0, \ldots, 0]$ or $A = [1, \ldots, 1]$, then $K_A = GF(2^n)$.*

Proof: In the first case, all terms in the xor-sum in (1) are 0 regardless of k, so the statement is trivially true. The second case follows from the fact SB is a bijective operation.

Lemma 3. *Let $\bar{A}$ be the complement of A, that is, $\bar{A} = [1, 1, ..., 1] \oplus A$. Then $K_{\bar{A}} = K_A$*

Proof: Since SB is bijective, we know that $\bigoplus_{i=0}^{2^n-1} S(k \oplus i) = 0$ for any fixed k. If the subset of terms selected by A sum to 0 (so $k \in K_A$), the complementary subset of terms must also sum to 0, hence $k \in K_{\bar{A}}$.

Lemma 4. *If the weight of A is 2 or $2^n - 2$, then $K_A = \emptyset$.*

Proof: SB is bijective, so $S(x) \oplus S(y) \neq 0$ for $x \neq y$ and the case of weight 2 is proven. The case for weight $2^n - 2$ follows from Lemma 3.

Using the properties introduced in the above lemmas, it is sufficient to find the key candidates only for arrays where the weight w_A is even and $4 \leq w_A \leq 2^{n-1}$. This technique is possible to apply to any integral or higher-order differential attacks on block ciphers that use S-boxes, but in the following we will just focus on the PRINCE block cipher.

4.2 Using the A Arrays

Having an $(R - 0.5)$-round distinguisher we can do the key-recovery phase on both R-round and $R + 1$-round PRINCE using the technique introduced above. For attacking R rounds we allocate one array to each of the nibbles in a state. Each array will suggest some candidates for one nibble of $K_1 \oplus K_0'$, including the right value. So the corresponding K_A to these arrays can never be empty. In these attacks we will not save the ciphertexts in the memory, so the memory complexity will just be saving the arrays, which is negligible.

For attacking $R + 1$-round PRINCE, instead of a half-round (one SB^{-1} operation), there is one and a half rounds (one SB^{-1} operation and one complete round) between the ciphertext and the end of the distinguisher. For key recovery

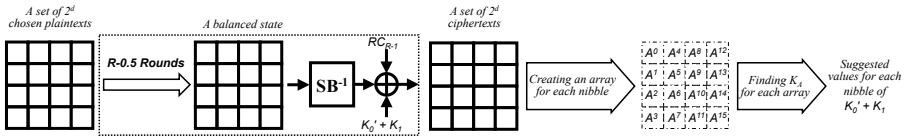

Fig. 3. Attack on R-round PRINCE using $(R-0.5)$-round distinguisher

we then guess one column of $K_0' \oplus K_1$ and partially decrypt the corresponding column in all of the 2^d ciphertexts for one round. Then we build A-arrays for the partially decrypted nibbles and find candidates for 4 corresponding nibbles of K_1. Since we do not know whether the guessed value of $K_0' \oplus K_1$ is the right one or not, the arrays related to these 4 nibbles of K_1 could have a empty set of K_A. In this case, when an array suggests an empty K_A it means the guessed value for $K_0' \oplus K_1$ was wrong. We call this a *false array*.

Both of the R- and $R+1$-round PRINCE attacks are illustrated in Figure 3 and Figure 4, respectively.

4.3 Average Number of S-box evaluations for an Array

For computing the cost of finding key candidates we need to know the average number of S-box evaluations for an array. By using the lemmas, this number will be equal to

$$n_{SB} = 4 \times (P_4 + P_{12}) + 6 \times (P_6 + P_{10}) + 8 \times P_8 \tag{2}$$

where P_w is the probability that the weight of array A is w.

In fact, the value for P_w depends on the number of texts used to produce the array A. For example, when $d = 4$ then $P_{16} < P_0$ even though there is only one array of weight 0 and one of weight 16. We can evaluate P_w, by using a recursive formula that we explain in the following. Let A_i be the array for one nibble, after processing i ciphertexts in one set. Assume we have processed $i-1$

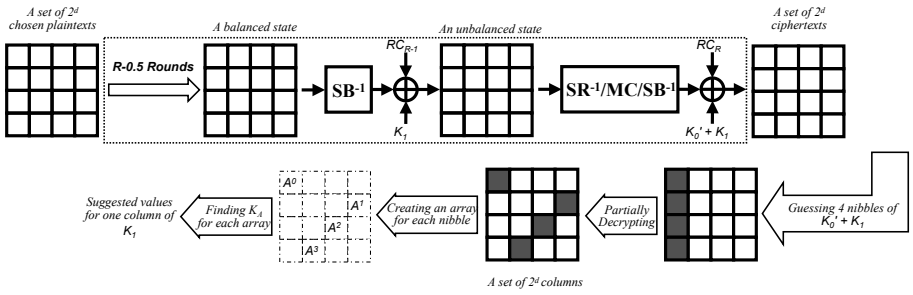

Fig. 4. Attack on $R+1$-round PRINCE using $(R-0.5)$-round distinguisher

nibble values and found that the weight of A_{i-1} is w. By processing one more nibble in the set, we have the relations below for the weight w' of the array A_i:

$$\begin{array}{rcll} Pr(w'=1) & = & 1 & w = 0 \\ Pr(w'=15) & = & 1 & w = 16 \\ Pr(w'=w-1) & = & \frac{w}{16} & w \neq 0,16 \\ Pr(w'=w+1) & = & 1-\frac{w}{16} & w \neq 0,16 \end{array} \tag{3}$$

Let $f_i = [f_i(0), f_i(1), \ldots, f_i(16)]$ be the probability distribution for the weight of the array A_i. In an actual attack, the array A is initialized to all-zeroes, so $f_0 = [1, 0, \ldots, 0]$. Given f_{i-1}, the probability distribution f_i is given as

$$\begin{array}{l} f_i(0) = \frac{1}{16} \cdot f_{i-1}(1) \\ f_i(j) = (1-\frac{j-1}{16}) \cdot f_{i-1}(j-1) + \frac{j+1}{16} \cdot f_{i-1}(j+1) \quad 0 < j < 16 \\ f_i(16) = \frac{1}{16} \cdot f_{i-1}(15) \end{array} \tag{4}$$

The values of P_w to be used in (2) are the values in f_{2^d} when using a set of 2^d chosen plaintexts, and these values can be computed by using (4) recursively. When d is equal to 4, n_{SB} will be 6.37419 and it converges quickly to 6.51904 for larger d's.

4.4 Average Number of Key Candidates for an Array

By running through the all 16-bit arrays with weight of 4, 6 and 8, we can count the number of arrays giving K_A's of the same size. These numbers are summarized in Table 2. For each possible weight w we find the average size of K_A, denoted $\bar{n}_w$, both with and without false arrays.

In a set of 2^d ciphertexts produced according to the distinguisher, the average number for $|K_A|$ is equal to

$$|\bar{K}| = \sum_{w=0}^{15} P_w \cdot \bar{n}_w \tag{5}$$

Table 2. Average Number for Suggested Key Candidates

	number of arrays with $\|K_A\| =$						average value for $\|K_A\|, \bar{n}_w$	
w	0	1	2	3	4	16	with F.A.	without F.A.
0	0	0	0	0	0	1	16	16
2	120	0	0	0	0	0	0	-
4	392	816	432	160	20	0	1.2308	1.5686
6	3120	3040	1488	288	72	0	0.8951	1.4664
8	4502	4320	2976	640	432	0	1.0816	1.6635
10	3120	3040	1488	288	72	0	0.8951	1.4664
12	392	816	432	160	20	0	1.2308	1.5686
14	120	0	0	0	0	0	0	-
16	0	0	0	0	0	1	16	16

The values of P_w to be used in (5) are the values in f_{2^d} when using a set of 2^d chosen plaintexts. We can now compute the exact value of the expected number of suggested keys from one set of plaintexts in an attack, given by (5). In our attacks with false arrays $|\bar{K}|$ is always 1. In attacks without false arrays $\bar{n}_K$ is about 1.5360 using 2^4 ciphertexts, and $|\bar{K}|$ converges quickly to 1.5453 for larger sets.

5 Cryptanalysis of Round-Reduced PRINCE

In this section we will use the distinguishers from Section 3 and the key recovery method introduced in Section 4 to cryptanalyze round-reduced PRINCE.

5.1 Attack on 4-round PRINCE

For 4-round PRINCE, we will use the 3.5-round integral distinguisher. One set consists of 16 chosen plaintexts that gets encrypted through 4-round PRINCE. One bit-array A is initialized to all zero for each of the 16 nibbles in a state. For the value x in a nibble, we flip the bit a_x in the correpsonding A for each ciphertext as they are produced. Finally we use the arrays to find the key candidates, and repeat with one more set to get unique values. The exact procedure for recovering a unique value for $K'_0 \oplus K_1$ using s sets of data is summarized in Algorithm 1. In the algorithm we use $C_i^{j,t}$ to denote i-th nibble in the j-th ciphertext of the t-th set. K^i denotes candidate subkeys for the i-th nibble.

After finding $K'_0 \oplus K_1$, we follow the attack in [12] and use a 2.5-round distinguisher starting from the second round of the 3.5-round distinguisher to

Algorithm 1 Key Recovery Attack without False Arrays

for $i = 0 : 15$ **do**
  $K^i = \mathbb{F}_2^4$;
end for
for $t = 1 : s$ **do**
  for $i = 0 : 15$ **do**
    $A^i = [0, \ldots, 0]$;
  end for
  for $j = 0 : 2^d - 1$ **do**
    for $i = 0 : 15$ **do**
      Put $x = C_i^{j,t}$ and flip the bit a_x^i;
    end for
  end for
  for $i = 0 : 15$ **do**
    Find k^i, the key candidates for A^i;
    $K^i = K^i \cap k^i$;
  end for
end for
Return $K = [K^0, \ldots, K^{15}]$;

find the exact value of K_1. The internal states will be balanced after 2.5 rounds, just before the S-box layer in the third round. We use the recovered $K'_0 \oplus K_1$ to decrypt the 4-round ciphertexts one round, and follow Algorithm 1 on the cipher states after three rounds to recover K_1. When both $K'_0 \oplus K_1$ and K_1 are known it is trivial to find the user-selected key.

Complexity: This attack is without false arrays and each set of data has 2^4 pairs. Each array will suggest on the average $m = 1.5360$ keys. After the first set of data is processed we expect $m^{16} = 2^{9.91}$ key candidates remaining for the whole $K'_0 \oplus K_1$. Using another set of data we will only have an expected $(2^{9.91})^2 \times 2^{-64} = 2^{-44.19}$ key candidates remaining for the value of $K'_0 \oplus K_1$. With very high probability, only the correct value for $K'_0 \oplus K_1$ will remain. So we need only two sets of data for finding $K'_0 \oplus K_1$ and similarly another two sets of data for finding K_1.

The data complexity of the attack is $2^4 \times (2 + 2) = 2^6$ chosen plaintexts and its memory complexity is just saving $16 \times m$ nibbles for storing the initial key candidates and 16 arrays of 16-bits each which is negligible.

The time complexity of this attack will be producing the chosen data (2^6 4-round encryptions), 16 times finding the key candidates for each nibble of $K'_0 \oplus K_1$ for each set (on average $2 \times 16 \times 2^4 \times 6.37419$ SB operations), one round partial decryption of second data sets (2^5 one round encryptions) and 16 times finding the key candidates for each nibble of K_1 for each set. In total this is equal to about $2^{7.44}$ 4-round PRINCE encryptions.

5.2 Attack on 5-round PRINCE

For cryptanalyzing 5-round PRINCE, we can use either the 3.5-round or 4.5-round integral distinguishers. Using the 3.5-round distinguisher will lead to lower data complexity, but a higher time complexity than using the 4.5-round distinguisher. We present both attacks below.

Attack with 4.5-round distinguisher: The attack has a similar procedure to the 4-round one, except that in each set there are 2^{12} ciphertexts instead of 2^4. First we find a unique value for $K'_0 \oplus K_1$ using Algorithm 1. Then we can decrypt one round and use the 3.5-round distinguisher to find the value of K_1. For the 3.5-round distinguisher it is not necessary to ask for more data, we can use subsets of size 2^4 that exist in the sets of 2^{12} pairs we already have. Using the recovered value for $K'_0 \oplus K_1$ we will partially decrypt only two subsets of 2^4 ciphertexts for one round to reach the internal state after 4 rounds and use them to find the exact value of K_1 with Algorithm 1.

Complexity: This attack is without false arrays and each set of data has 2^{12} pairs. So each array will suggest $m = 1.5453$ keys. After the first set of data is processed we can expect to have $m^{16} = 2^{10.05}$ key candidates for the whole $K'_0 \oplus K_1$. Using another set of data we expect to have only $(2^{10.05})^2 \times 2^{-64} = 2^{-43.91}$ key

candidates for the value of $K'_0 \oplus K_1$. So we need only two sets of data for finding $K'_0 \oplus K_1$, and reuse subsets within these to find K_1.

The data complexity of the attack is $2 \times 2^{12} = 2^{13}$ chosen plaintexts and its memory complexity is saving two set of 2^4 data for recovering K_1. The time for producing the chosen data sets is dominating the time complexity in the attack. Hence the time complexity is approximately 2^{13} 5-round PRINCE encryptions.

Attack with 3.5-round distinguisher: In this attack, we will guess one column of $K'_0 \oplus K_1$ and partially decrypt the ciphertexts for one round. The values for four nibbles at the end of the fourth round can be computed for each guessed column. We build A-arrays from these values. Then we will find the corresponding candidates for 4 nibbles of K_1. When the guess for a column of $K'_0 \oplus K_1$ is wrong, we may end up with an A array such that K_A is the empty set, that is, we have a false array. In this case we can reject the guessed value for $K'_0 \oplus K_1$ as wrong and go to next value. The exact procedure for recovering a unique value for the c-th column of $K'_0 \oplus K_1$ and its corresponding 4 nibbles in K_1 is summarized in Algorithm 2.

Complexity: In this attack we may get false arrays and each set of data has 2^4 pairs. Then each array will suggest one key on the average. So after processing

Algorithm 2 Key Recovery Attack with False Arrays

for $K \in \mathbb{F}_2^{16}$ **do**
  for $i = 0 : 3$ **do**
    $K_1^i = \mathbb{F}_2^4$;
  end for
  for $t = 1 : s$ **do**
    for $i = 0 : 3$ **do**
      $A^i = [0, \ldots, 0]$;
    end for
    for $j = 0 : 2^d - 1$ **do**
      Using K partially decrypt $C^{j,t}_{4c:4c+3}$ to reach $[x^0, x^1, x^2, x^3]$;
      for $i = 0 : 3$ **do**
        Flip the bit $a^i_{x^i}$;
      end for
    end for
    for $i = 0 : 3$ **do**
      Find k^i, the key candidates for A^i;
      $K_1^i = K_1^i \cap k^i$;
      if K_1^i is empty **then**
        Reject the current value of K and go to next value;
      end if
    end for
  end for
  Return K and $[K_1^0, K_1^1, K_1^2, K_1^3]$;
end for

the first set of data we will have one candidate for 4 nibbles of K_1 related to the guessed value for 4 nibbles of $K_0' \oplus K_1$. Using another set of data, there will remain only $2^{16} \times 2^{-16} = 1$ key candidate for 4 nibbles of $K_0' \oplus K_1$ and the related 4 nibbles in K_1. For finding the other subkeys related to the other columns we will use these sets of data again, so we need only these two sets.

The data complexity of the attack is $2^4 \times 2 = 2^5$ chosen plaintexts. The memory complexity is saving the 2^5 ciphertexts.

The time complexity is in the worst case (when we never exit early due to empty K_1^i)

$$4 \times 2^{16} \times 2 \times (2^4 \times 4 + 4 \times 2^4 \times 6.519) = 2^{27.91}$$

SB operations which is about $2^{21.59}$ 5-round PRINCE encryptions. Using the accelerating techniques from [14] (which stores S-box evaluations that are not affected by new guesses of $K_1 \oplus K_0'$), the time complexity can be further reduced to $2^{27.76}$ SB operations or $2^{21.44}$ 5-round PRINCE encryptions.

5.3 Attack on 6-round PRINCE

For attacking 6-round PRINCE, we will use the 4.5-round integral distinguisher with partial key guessing following Algorithm 2. With two sets of 2^{12} chosen plaintext/ciphertetx pairs, we will guess one column of $K_0' \oplus K_1$, partially decrypt the ciphertexts for one round and build A-arrays, and then find the corresponding values for 4 nibbles of K_1.

This attack will need only two sets of data, so the data complexity of the attack is $2^{12} \times 2 = 2^{13}$ chosen plaintexts. The memory complexity is again saving the ciphertexts. The time complexity will be

$$4 \times 2^{16} \times 2 \times (2^{12} \times 4 + 4 \times 2^4 \times 6.519) = 2^{33.04}$$

SB operations, and by using the accelerating techniques from [14] when guessing the 4 nibbles of $K_0' \oplus K_1$, the time complexity can be reduced to $2^{31.22}$ SB operations or approximately $2^{24.64}$ 6-round PRINCE encryptions.

5.4 attack to 7-round PRINCE

For the attack on 7-round PRINCE, we use the 5.5-round higher-order differential distinguisher and Algorithm 2 with two sets of 2^{32} pairs of data. Each set of data will be balanced right before the S-box layer in the sixth round, so the key recovery procedure in Algorithm 2 can be applied.

Again this attack will need only two sets of data, so its data complexity is 2^{33} chosen plaintext/ciphertext pairs. Saving the ciphertexts is the substantial memory complexity. The time complexity will be

$$4 \times 2^{16} \times 2 \times (2^{32} \times 4 + 4 \times 2^4 \times 6.519) = 2^{53}$$

SB operations which by applying the accelerating technique the time complexity can be reduced to $2^{51.09}$ SB operations or $2^{44.29}$ 7-round PRINCE encryptions.

6 Conclusion

In this paper we have introduced a technique of using arrays for the key recovery part of an integral or higher order differential attack. In particular, integral and higher order differential attacks on block ciphers with S-boxes will benefit from this technique.

We have applied the faster key recovery to the same integral or higher order differential distinguishers used in earlier attacks on PRINCE. The improvements in complexity, as measured by the number of S-box evaluations, gains a significant factor from the earlier attacks. Our attacks on 4 and 6 rounds were the fastest and the winner of PRINCE Challenge's last round in the category of chosen plaintext attack.

References

1. J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knežević, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalçın, "PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications", *ASIACRYPT 2012*, LNCS, vol. 7658, pp. 208 – 225, Springer 2012.
2. The PRINCE Team, "PRINCE Challenge", `https://www.emsec.rub.de/research/research_startseite/prince-challenge/`.
3. F. Abed, E. List, and S. Lucks, "On the Security of the Core of PRINCE Against Biclique and Differential Cryptanalysis", *IACR Cryptology ePrint Archive*, Report 2012/712, 2012.
4. J. Jean, I. Nikolić, T. Peyrin, L. Wang, and S. Wu, "Security Analysis of PRINCE", *Fast Software Encryption 2013*, LNCS, vol. 8424, pp. 92 – 111, Springer 2013.
5. H. Soleimany, C. Blondeau, X. Yu, W. Wu, K. Nyberg, H. Zhang, L. Zhang, and Y. Wang, "Refection Cryptanalysis of PRINCE-like Ciphers", *Fast Software Encryption 2013*, LNCS, vol. 8424, pp. 71 – 91, Springer 2013.
6. A. Canteaut, M. Naya-Plasencia, and B. Vayssiére, "Sieve-in-the-Middle Improved MITM Attacks", *CRYPTO 2013*, LNCS, vol. 8042, pp. 222 – 240, Springer 2013.
7. L. Li, K. Jia, and X. Wang, "Improved Meet-in-the-Middle Attacks on AES-192 and PRINCE", *IACR Cryptology ePrint Archive*, Report 2013/573, 2013.
8. A. Canteaut, T. Fuhr, H. Gilbert, M. Naya-Plasencia, and J.-R. Reinhard, "Multiple Differential Cryptanalysis of Round-Reduced PRINCE", *Fast Software Encryption 2014*, LNCS, vol. 8540, pp. 591 – 610, Springer 2014.
9. P.-A. Fouque, A. Joux, and C. Mavromati, "Multi-user collisions: Applications to Discrete Logarithm, Even-Mansour and PRINCE", *ASIACRYPT 2014*, LNCS, vol. 8873, pp. 420 – 438, Springer 2014.
10. Itai Dinur, "Cryptanalytic Time-Memory-Data Tradeoffs for FX-Constructions with Applications to PRINCE and PRIDE", *EUROCRYPT 2015*, LNCS, vol. 9056, pp. 231 – 253, Springer 2015.
11. P. Derbez and L. Perrin, "Meet-in-the-Middle Attacks and Structural Analysis of Round-Reduced PRINCE", *Fast Software Encryption 2015*, LNCS, vol. 9054, pp. 190 – 216, Springer 2015.
12. P. Morawiecki, "Practical Attacks on the Round-reduced PRINCE", *IACR Cryptology ePrint Archive*, Report 2015/245, 2015.

13. R. Posteuca and G. Negara, "Integral Cryptanalysis of Round-reduced PRINCE Cipher", *Proc. Rom. Acad. Ser. A Math. Phys. Tech. Sci. Inf. Sci.*, vol. 16 (2015), Special issue, pp. 265–269, 2015.
14. Sh. Rasoolzadeh and H. Raddum, "Cryptanalysis of PRINCE with Minimal Data", *AfricaCrypt 2016*, LNCS, vol. 9646, pp. 109 – 126, Springer 2016.
15. Sh. Rasoolzadeh and H. Raddum, "Cryptanalysis of 6-round PRINCE using 2 Known Plaintexts", to be presented at *ArcticCrypt 2016*.
16. J. Daemen, L. Knudsen, and V. Rijmen, "The Block Cipher Square", *Fast Software Encryption 1997*, LNCS, vol. 1267, pp. 149 – 165, Springer 1997.
17. X. Lai, "Higher Order Derivatives and Differential Cryptanalysis", *Communications and Cryptography*, vol 276, pp. 227 – 233, Springer 1994.
18. L. R. Knudsen, "Truncated and Higher Order Differentials", *Fast Software Encryption 1994*, LNCS, vol. 1008, pp. 196 – 211, Springer 1994.