Cryptography and Communications

Cryptanalysis of 6-round PRINCE using 2 Known Plaintexts

--Manuscript Draft--

Manuscript Number:	
Full Title:	Cryptanalysis of 6-round PRINCE using 2 Known Plaintexts
Article Type:	Special Issue on Arctic Crypt
Corresponding Author:	Håvard Raddum, Ph.D. Simula@UiB NORWAY
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Simula@UiB
Corresponding Author's Secondary Institution:	
First Author:	Shahram Rasoolzadeh
First Author Secondary Information:	
Order of Authors:	Shahram Rasoolzadeh
	Håvard Raddum, Ph.D.
Order of Authors Secondary Information:	
Funding Information:	
Abstract:	In this paper we focus on the PRINCE block cipher reduced to 6 rounds, with two known plaintext/ciphertext pairs. We develop two attacks on 6-round PRINCE based on accelerated exhaustive search, one with negligible memory usage and one having moderate memory requirements. The time complexity for the first attack is 2^{96.78} encryptions. Time complexity for the second attack depends on the implementation, but can be argued to be approximately 2^{89} for a normal PC. The memory consumption of the second attack is less than 200MB and so is not a restricting factor in a real-world setting.

Cryptanalysis of 6-round PRINCE using 2 Known Plaintexts

Shahram Rasoolzadeh and Håvard Raddum

Simula Research Laboratory

Abstract. In this paper we focus on the PRINCE block cipher reduced to 6 rounds, with two known plaintext/ciphertext pairs. We develop two attacks on 6-round PRINCE based on accelerated exhaustive search, one with negligible memory usage and one having moderate memory requirements. The time complexity for the first attack is $2^{96.78}$ encryptions. Time complexity for the second attack depends on the implementation, but can be argued to be approximately 2^{89} for a normal PC. The memory consumption of the second attack is less than 200MB and so is not a restricting factor in a real-world setting.

Keywords: PRINCE, lightweight cipher, exhaustive search

1 Introduction

PRINCE is a lightweight block cipher proposed by Borghoff et. al. at Asiacrypt 2012 [1] and is designed to be efficiently implemented in hardware, with minimal latency and small chip area. It is designed to be a *reflection cipher*, which means that decryption with one key is equal to encryption with another (related) key. For PRINCE the relation between encryption/decryption keys is chosen to be xor with a constant value α.

This novel design and the fact that there are prizes awarded for the best cryptanalysis on PRINCE has attracted quite a bit of attention from cryptanalysts. There is a Prince Challenge website [2] where the best attacks and their complexities are summarized.

In Table 1 we have listed previous published work on 6-round PRINCE. In the known plaintext scenario there was only one result listed on the Prince Challenge site [3]. This paper aims to cast some more light on cryptanalysis of 6-round PRINCE in the known plaintext scenario.

In this work we present two attacks. The first attack is an accelerated exhaustive search attack, where we try to reject wrong guesses of the K_1 key used in PRINCE$_{core}$ as quickly as possible, and to minimize the number of S-box look-ups needed. The main insight here is that assuming a known state in the middle of the cipher, only part of the unknown K_1 needs to be guessed before it is possible to identify a guess as incorrect.

The second attack is similar to the first in the sense that parts of K_1 are guessed. The difference is that we will create tables to store partial guesses, and

use the tables during the attack to quickly identify wrong guesses. We denote this attack as accelerated exhaustive search with memory. However, it should be noted that the memory requirements are less than 200MB, so the memory complexity is not a limiting factor in practice. This attack has the fastest time complexity, equivalent to $2^{88.85}$ 6-round PRINCE encryptions.

The paper is organized as follows. Section 2 presents a brief description of PRINCE. In Sections 3 and 4 we outline the Accelerated Exhaustive Search attacks to 6-round PRINCE with no memory and with memory, respectively. Section 5 concludes the paper.

2 PRINCE Block Cipher

PRINCE [1] is an FX-constructed lightweight block cipher with block size of 64 bits and two keys that both have length 64 bits. One of the keys (K_0) are used for whitening and the other one (K_1) is used as a round key for the core of the structure (see Figure 1). Following [1], we denote the plaintext/ciphertext pair of PRINCE by P/C, and the corresponding input/output of the PRINCE$_{core}$ function by P'/C'. These variables are related through the following equations.

$$P' = P \oplus K_0 \quad , \quad C' = C \oplus K_0', \tag{1}$$

where K_0' is the following linear mapping of K_0

$$K_0' = L(K_0) = (K_0 \ggg 1) \oplus (K_0 \gg 63). \tag{2}$$

In the same way as in [7], we use one property of FX-constructed block ciphers in our Accelerated Exhaustive Search attacks. This property is summarized in the following lemma.

Lemma 1. *[7] For a P/C pair and its corresponding P'/C' in an FX block cipher which uses a linear mapping L between whitening keys, the following equation holds:*

$$L(P') \oplus C' = L(P) \oplus C \tag{3}$$

Table 1. Summary of cryptanalytic results on 6-round PRINCE. Time complexity is given as number of 6-round PRINCE encryptions and memory complexity is given as number of 64-bit blocks.

Mode	Time	Data	Memory	Technique	Ref.
KP	2^{101}	2^6	2^{34}	MITM	[3]
	$2^{96.8}$	2	negl.	Acc. Exh. Search	Sec. 3
	$2^{86.04} + 2^{94.05}$ MA*	2	$2^{24.6}$	Acc. Exh. Search	Sec. 4
CP	$2^{33.7}$	2^{16}	$2^{31.9}$	MITM	[3]
	2^{64}	2^{16}	2^{16}	Integral	[4]
	2^{41}	$2^{18.58}$	2^{16}	Integral	[6]
	$2^{32.9}$	$2^{14.9}$	$\ll 2^{27}$	Differential/Logic	[5]
	$2^{33.7}$	2^{16}	$2^{31.9}$	MITM	[5]

* MA = memory access in table of size 2^{25}.

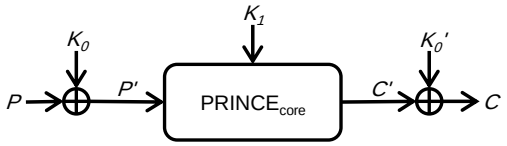

Fig. 1. PRINCE FX Construction

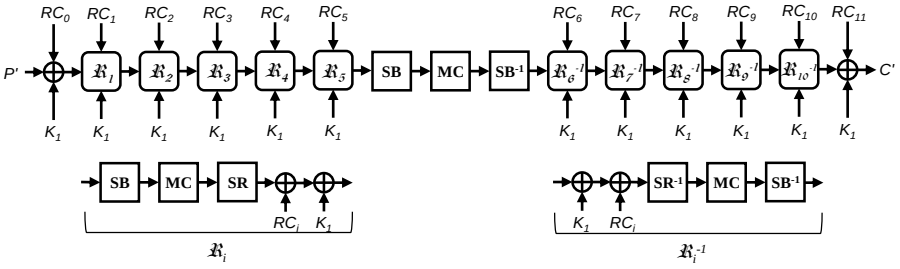

Fig. 2. PRINCE core

The PRINCE$_{core}$ is an AES-like block cipher that employs an involutive 12 rounds structure. PRINCE$_{core}$ starts with two *xor*s with the key 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 the key. 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, matrix multiplication, 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.
- **Matrix Multiplication** (MC): The state is multiplied with an involutive 64×64 binary matrix. More precisely, this large matrix can be expressed as four 16×16 matrices where each of these mixes four nibbles 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 the 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.

The middle two rounds contain only three layers, SB, MC, SB^{-1} which makes it an involutive keyless 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.

As a result of this involutive structure of PRINCE$_{core}$, in implementations decryption can use the same circuit as encryption. In decryption mode the key only needs to be *xor*ed with α, i.e.

$$C' = PRINCE_{core}(P', K_1) \Longleftrightarrow P' = PRINCE_{core}(C', K_1 \oplus \alpha). \quad (4)$$

This property is called α-reflection.

3 Accelerated Exhaustive Key Search

In this section we will present an accelerated exhaustive key search on 6-round PRINCE. Our way of doing this is faster than a simple exhaustive key search that guesses a key, fully encrypts a known plaintext, and checks if it matches the given ciphertext.

In the attack, for a known plaintext/ciphertext pair we guess one state in the middle of the cipher and then by using (3) we find one candidate for K_1. Knowing K_1 and one inner state for this pair of data allows us to find K_0. We can then check this candidate key of (K_0, K_1) on a second pair of known plaintext/ciphertext. If (K_0, K_1) matches both plaintext/ciphertext pairs it should be the correct key.

For simplifying our accelerated exhaustive search analysis, we define two equivalent keys for PRINCE$_{core}$. These are

$$\begin{aligned} K_1' &= SR^{-1}(MC(K_1)), \\ K_1'' &= L(K_1) \oplus K_1. \end{aligned} \quad (5)$$

When we use K_1', we must position the AK layer between the SB and MC layers of the round to get an equivalent description of PRINCE$_{core}$ (see Figure 3). Clearly, by recovering K_1' we can recover K_1.

By using K_1' instead of K_1, we can also expand the keyless middle rounds by two SR and two MC operations. As shown in Figure 3, we denote the states right before and after these keyless functions by X and X'.

Like the accelerated exhaustive search attack in [7], for a given P and its C we will guess the value of X and calculate the value of the corresponding X'. For each of the 2^{64} X/X'-values, we will guess some nibbles of K_1' and then partially decrypt/encrypt the X/X' to find some nibbles in $P'' = P' \oplus K_1$ and $C'' = C' \oplus K_1$. The position of the found nibbles will be equal in P'' and C''. Then we evaluate the value of the corresponding nibble in

$$F(P'', C'', P, C) = L(P'') \oplus C'' \oplus (L(P) \oplus C)$$

which gives us the value of the same nibble in K_1'', because

$$\begin{aligned} & L(P'') \oplus C'' \oplus (L(P) \oplus C) \\ &= (L(P') \oplus C') \oplus (L(P) \oplus C) \oplus L(K_1) \oplus K_1 \\ &= K_1''. \end{aligned} \quad (6)$$

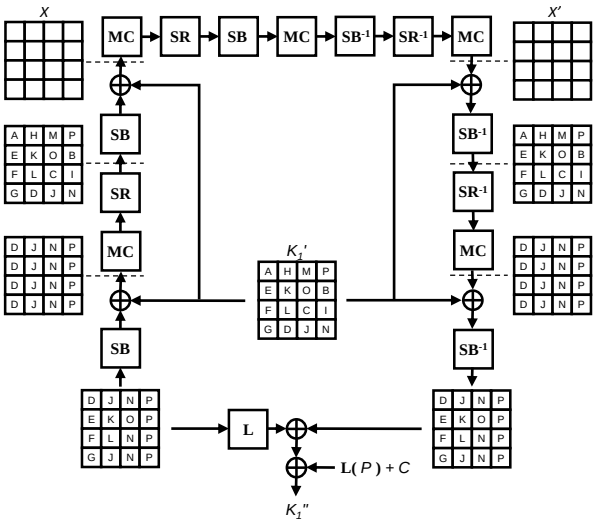

Fig. 3. Accelerated Exhaustive Key Search for 6 round PRINCE

As K_1' and K_1'' has a linear relation, finding n bits of K_1'' gives us n linear equations in the K_1' bits. All 64 linear relations between K_1' and K_1'' are listed in Appendix A. Using the relation between K_1' and K_1'' means we do not need to guess all bits of K_1'; some of them can be deduced from already guessed K_1'-bits and known K_1'' bits. In [7] all 64 bits of K_1' must be guessed before it can be verified or rejected, but with only six rounds we can reject partial guesses as incorrect at an earlier stage and cut down on the search space.

There will be one value of K_1' in average for each X/X' that will produce P' and C' which will match the given right-hand side in (3). The value for P' computed for this K_1' and X/X' is then used to deduce K_0. So for each X/X' guess we can expect one (K_0, K_1) candidate. This candidate for the full key can be tried on one other plaintext/ciphertext pair, and if it matches it should be the correct key.

In our analysis we try to minimise the number of S-box look-ups needed, and also try to find the most bits of K_1'' as quickly as possible. The results show that 6-round PRINCE can be attacked with complexity equal to $2^{96.78}$ encryptions using only 2 known plaintexts. This is lower than the previous best attack on 6-round PRINCE in the known plaintext mode [5].

3.1 Attack Procedure

The strategy of the attack is to minimize the number of total S-box look-ups needed when we guess values for the K_1' nibbles, and to reject wrong guesses as soon as possible. Figure 3 shows the order for guessing the nibbles of K_1' and in the following we explain what happens in Figure 3, focusing on the forward

rounds. Because of the reflective property of PRINCE, the exact same computations done in these rounds can be done in the backward rounds.

Let Z be a binary matrix with 64 columns. Z is empty at first, but as we start to guess values for the K_1' nibbles we will fill in the rows of Z to store the linear constraints we get. We thus build a system of linear equations

$$ZK_1' = V,$$

where V is the value given by the current guess. Whenever we find bits of K_1'', we will add the linear relations between K_1' and K_1'' as rows to Z as well.

The nibbles of K_1' will be guessed in alphabetical order, starting with A. The letters in the other states of Figure 3 indicate which nibbles can be computed after which guess. After D has been guessed, we have enough known nibbles to go backwards through SR and MC in round 1 and find the input. As we have already guessed the A-value of K_1', we can add this to the top left nibble and compute the input to the top left S-box in round 1. This is indicated with the state at the bottom with a single D in this position.

At this point we have computed the value of nibble D in both P'' and C'', so we can compute the part of $F(P'', C'', P, C)$ that affects this nibble only, and find a value for 3 bits of $K_1'' = (k_{63}'', \ldots, k_0'')$. These are the 3 least significant bits in this nibble, i.e. k_{62}'', k_{61}'' and k_{60}''. These three k'' bits are linearly related to the k' bits with

$$\begin{aligned} k_{62}'' &= k_{62}' \oplus k_{59}' \oplus k_{55}' \oplus k_{54}' \oplus k_{51}' \oplus k_{50}', \\ k_{61}'' &= k_{62}' \oplus k_{61}' \oplus k_{57}' \oplus k_{54}' \oplus k_{50}' \oplus k_{49}', \\ k_{60}'' &= k_{61}' \oplus k_{60}' \oplus k_{57}' \oplus k_{56}' \oplus k_{52}' \oplus k_{49}'. \end{aligned}$$

So with the current guess of k' bits the $F(P'', C'', P, C)$ function will give us three extra linear constraints in addition to the 16 guessed ones, for a total of 19 independent linear equations in the 64 K_1' variables. All of these are added to Z, so after guessing 16 bits we have 19 linear constraints on K_1'.

Next, we guess the four bits of E, which allows us to compute another 4 bits of K_1''. Eight new linear equations get added to Z. After guessing F we find another 4 bits of K_1'' and can add another 8 linear equations to Z. Now it's time to guess G, but three of the bits in G are actually determined by the system $ZK_1' = V$ that we have built so far. Hence there is only one bit left to guess in G, but we still earn four new linear equations from the K_1'' bits that become known after fixing the value for G.

All of this is summarized in the first rows of Table 2, where we list the number of bits to guess in each nibble, the indices of K_1''-bits that become known and the rank of Z after each guess.

We now need to guess all of H, I and J before being able to produce more K_1'' bits. After fixing J it turns out that some of the K_1''-bits that gives extra equations are linearly dependent with the equations currently in Z. The dependencies function as a filter, allowing us to reject the current guess as wrong if we get an inconsistent system when adding the equations to Z. The k'''s involved in the dependent linear equations are listed in the column labelled "matching bits"

Table 2. Details of Attack Procedure

Nibble of K_1'	Number of guessed bits	Number of SB	Index of found bits from K_1''	matching bits	p	rank(Z)
A	4	1	–	–	1	4
B	4	1	–	–	1	8
C	4	1	–	–	1	12
D	4	2	62, 61 ,60	–	1	19
E	4	2	59, 58, 57, 56	–	1	27
F	4	2	55, 54, 53, 52	–	1	35
G	1	2	51, 50, 49, 48	–	1	40
H	4	1	–	–	1	44
I	4	1	–	–	1	48
J	4	2	34, 33, 32	$k''_{34}, k''_{33}, k''_{32}$	2^{-3}	52
		1	47, 46, 45, 44	$k''_{47} \oplus k''_{46}$	2^{-1}	55
K	1	1	43, 42, 41, 40	$k''_{43} \oplus k''_{42}$	2^{-1}	59
L	0	1	39, 38, 37, 36, 35	$k''_{39}, k''_{38} \oplus \ldots \oplus k''_{35}$	2^{-2}	62
		2			1	
M	1	1	–	–	1	63
N	1	2	31, 30, 29, 28	$k''_{31}, k''_{30}, k''_{29}, k''_{28}$	2^{-4}	64
		1	22, 21, 20,	$k''_{22}, k''_{21}, k''_{20}$	2^{-3}	64
		1	19, 18, 17, 16	$k''_{19}, k''_{18}, k''_{17}, k''_{16}$	2^{-4}	64
O	0	1	27, 26, 25, 24, 23	$k''_{27}, k''_{26}, k''_{25}, k''_{24}, k''_{23}$	2^{-5}	64
		1			1	
P	0	2	15, 14, 13, 12	$k''_{15}, k''_{14}, k''_{13}, k''_{12}$	2^{-4}	64
		1	11, 10, 9, 8	$k''_{11}, k''_{10}, k''_{9}, k''_{8}$	2^{-4}	64
		1	7, 6, 5, 4	$k''_{7}, k''_{6}, k''_{5}, k''_{4}$	2^{-4}	64
		1	3, 2, 1, 0, 63	$k''_{3}, k''_{2}, k''_{1}, k''_{0}, k''_{63}$	2^{-5}	64

in Table 2, and the probability that the current guess is not rejected is listed in the column named p.

Table 2 shows the details of what happens when the remaining nibbles are guessed. Note that when the final undetermined bit in nibble N is guessed, Z gets full rank and all of K_1' is determined. Remaining nibbles will only be used to verify/reject the current guess. If a guess gets to the end of Table 2 without being rejected, we will calculate P' and C' from X/X' and this guess and check for a match in (3). Only a single K_1 is expected to remain after this matching.

3.2 Complexity

Similarly to [7] we will focus on the number of S-box look-ups to estimate the complexity of the attack, where we equate $16 \times 6 = 2^{6.58}$ S-box look-ups with one 6-round PRINCE encryption.

Let s_Y be the number of S-box look-ups we can do after guessing nibble Y. These numbers are listed in the third column of Table 2 for all nibbles $A, \ldots, P$. Note that after some of the nibbles we do not execute all possible S-box look-ups right away. Using nibble J as an example, we can evaluate 3 S-boxes after

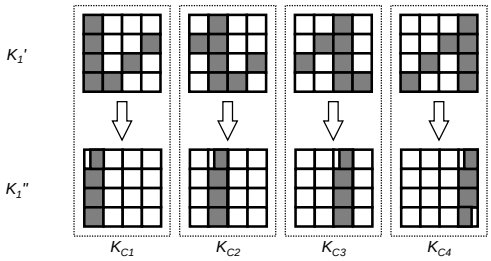

Fig. 4. 4 subsets of K_1' and their corresponding key bits from K_1''.

guessing this nibble, but we only do two of them first. The reason for this is that after these two S-boxes are executed we get some K_1'' bits giving dependent equations to be used as a filter for the current guess. In most cases the last S-box look-up does not need to be done because the guess can already be rejected as wrong, hence we save in the complexity. For nibbles L, N, O and P we do the same, and only execute the minimal number needed to filter out wrong guesses.

Let p_Y be the probability that the current guess is not rejected after guessing nibble Y (second to last column in Table 2) and g_Y be the number of bits to guess in nibble Y (second column in Table 2). We will store the outputs of evaluated S-boxes during the attack, and only recompute them when a new guess is made that affects them. The expression for the total number of S-box look-ups (both forward and backward rounds) needed to do in the attack is then

$$2 \times 2^{g_A}(s_A + p_A 2^{g_B}(s_B + \ldots + p_I 2^{g_J}(s_{J_1} + p_{J_1}(s_{J_2} + p_{J_2}(\ldots (s_{P_3} + p_{P_3}(s_{P_4}))\ldots))))). \tag{7}$$

Plugging in the values in Table 2 in the expression, it evaluates to $2^{39.36}$. This guessing needs to be done for each of the 2^{64} values for X/X'. Trading $2^{6.58}$ S-box look-ups for one encryption we get the final time complexity for the attack to be $2^{96.78}$.

4 Accelerated Exhaustive Key Search Using Memory

In this section we will present an attack similar to the one in the previous section. The attack in this section introduces a time/memory trade-off, and makes the attack faster by using tables of precomputed data. Our technique for this is to do 4 separate phases of key guessing for each X/X', and save the results in separate tables. The tables have partially overlapping information, and we match data from the tables to find unique candidates for K_1'.

The 4 subsets of K_1' and their corresponding found key bits from K_1'' are shown in Figure 4. We denote them by K_{C1}, K_{C2}, K_{C3}, and K_{C4}.

Guessing the values of the K_1' nibbles affecting the i-th column of K_1'' is similar to what we did in the previous section for guessing the $A, \ldots, G$ nibbles

and finding teh 15 corresponding bits of K_1''. The only difference is that when we guess E, F and G we only do one S-box look-up for the related nibbles in round 1. There are 28 bits of K_1' in each K_{Ci}, but remember that 3 of the bits in K_1' can be determined by the other 25 and the bits we find in K_1''. So the time complexity for finding the K_1'' bits for all values of the 25 independent bits in the K_1' subset in K_{Ci} is equal to

$$2 \times 2^4(1 + 2^4(1 + 2^4(1 + 2^4(2 + 2^4(1 + 2^4(1 + 2^1(1))))))) = 2^{26.62} \quad (8)$$

S-box look-ups. It should be mentioned that in K_{C4}, the 25 bits of K_1' only gives 14 bits from K_1'' due to the $L(.)$ function used in the FX-construction of PRINCE.

The attack procedure is similar to the attack in previous section. The difference is the way we find K_1' candidates related to a guessed value of X/X'. Assuming known X/X', we will separately create 3 tables $\mathcal{T}_2$, $\mathcal{T}_3$ and $\mathcal{T}_4$ where $\mathcal{T}_i$ contains all information about 2^{25} values of (K_{Ci}', K_{Ci}'') pairs in K_{Ci}.

4.1 Constructing Tables

Every K_{Ci} has 28 bits from K_1' and 15 bits from K_1'' (14 bits in K_{C4}). However, 3 bits in the last guessed nibble from K_1' get determined from the 11 first found bits of K_1''. It means that for every guessed value of 25 bits of K_1', we find 15 or 14 bits of K_1'' which gives a total of 40 or 39 independent linear equations in the K_1' bits.

Both K_{C1} and K_{C3} has 40 information bits about K_1, but the rank of $K_{C1} \cup K_{C3}$ is 60. It means K_{C1} and K_{C3} have $40 + 40 - 60 = 20$ common information bits, which we denote by I_{C_1,C_3} (8 of the common information bits are from overlapping guesses in K_1').

When we create $\mathcal{T}_3$, we will calculate $v_1 = I_{C_1,C_3}$ for each of the (K_{C3}', K_{C3}'') pairs in K_{C3} and just put this pair in the index of v_1. So on average, each index of $\mathcal{T}_3$ will have $2^{25-20} = 32$ different values of (K_{C3}', K_{C3}'') pairs. For reducing the amount of used memory, it is not necessary to save all of the 40 bits in K_{C3}. We only need to save the other $40 - 20 = 20$ bits not in common with K_{C1}. In this way, the amount of used memory for storing $\mathcal{T}_3$ is 20×2^{25} bits.

The rank of $K_{C1} \cup K_{C3}$ is 60, there are 40 bits in K_{C2} and $K_{C1} \cup K_{C2} \cup K_{C3}$ has full rank (64). Then $K_{C1} \cup K_{C3}$ and K_{C2} have $60 + 40 - 64 = 36$ common information bits which we denote by $I_{(C_1,C3),C2}$.

For the second table $\mathcal{T}_2$, we will calculate $(v_2, v_3) = I_{(C_1,C3),C2}$ for each of the (K_{C2}', K_{C2}'') pairs, where $|v_2| = 25$ and $|v_3| = 11$, and put this pair in the index of v_2. Again it is not necessary to save all of the 40 bits in K_{C2}. We only need to save v_3 and the other $40 - 36 = 4$ bits, v_4, not in common with (K_{C1}, K_{C3}). So the amount of memory used for storing $\mathcal{T}_2$ is 15×2^{25} bits.

Creating $\mathcal{T}_4$ is easier. We will just save the 14 bits of K_{C4}'' (v_6) in the index of the related K_{C4}' (v_5). So the memory needed for storing $\mathcal{T}_4$ is 14×2^{25} bits.

Using the three precomputed tables we will match values from the K_{Ci}s for each of the 2^{25} K_{C1}-values. In the following we will explain how to use the tables

and do the matching in detail. The procedure will be precisely summarized in Algorithm 1.

4.2 Attack Procedure

After creating the three tables, for every guess of K'_{C1} we will find its corresponding K''_{C1} and then compute the related 20 common information bits with K_{C3}, $v'_1 = I_{C_1,C_3}$. By retrieving $\mathcal{T}_3[v'_1]$, we will get 32 candidates for K_{C1} and K_{C3}. For each of these candidates we will compute their 36 common information bits with K_{C2}, $(v'_2, v'_3) = I_{(C_1,C_3),C_2}$. Then we look up $\mathcal{T}_2[v'_2]$ which has two elements, 11 bits of v_3 and 4 bits of v_4. The v'_3 value must be equal to the v_3 found in $\mathcal{T}_2$. If this matching happens, we will use v_4 to learn all 64 bits of K_1.

Having a candidate for (K_{C1}, K_{C3}, K_{C2}), we will compute 25 bits $v'_5 = K'_{C4}$ and 14 bits $v'_6 = K''_{C4}$ for this candidate. Finally we check if $\mathcal{T}_4[v'_5] = v'_6$. If the values do not match the candidate (K_{C1}, K_{C3}, K_{C2}) can not give the right K_1.

As there were 32 candidates after the $\mathcal{T}_3$ look-up, 11 matching bits in the $\mathcal{T}_2$ look-up and 14 matching bits for $\mathcal{T}_4$, there will remain only $2^{25} \times 2^5 \times 2^{-11} \times 2^{-14} = 2^5$ candidates for K_{C1} that will match the other key subsets K_{Ci}. This gives 32 candidates for K_1. We will decrypt/encrypt X/X' using each of these K_1 candidates to reach P'/C'. Checking for equality of (3) will give one candidate for K_1. We can then find the corresponding K_0 for this value of K_1, and by checking (K_0, K_1) on another plaintext/ciphertext pair we will find the correct key.

4.3 Complexity

The memory complexity of this attack is saving $\mathcal{T}_2$, $\mathcal{T}_3$ and $\mathcal{T}_4$ which needs

$$15 \times 2^{25} + 5 \times 2^{27} + 14 \times 2^{25} = 49 \times 2^{25} = 2^{30.61} \tag{9}$$

bits which is equal to $2^{24.61}$ PRINCE blocks.

Regarding the time complexity, for each guess of X/X' pair, we create 3 tables and also find K''_{C1} for each of the 2^{25} K'_{C1} candidates. These calculations need $4 \times 2^{26.62}$ S-box look-ups (t_{SB}). For each guess of K'_{C1} we do a look-up in $\mathcal{T}_3$ (t_{T3}), which gives us 2^5 candidates. For each candidate we do a call for $\mathcal{T}_2$ (t_{T2}). After matching in $\mathcal{T}_2$, the number of candidates gets reduced by fraction of 2^{-11} which means time for the remaining part of the attack is negligible. So the total time complexity of attack is approximately

$$2^{64} \times (2^{28.62} t_{SB} + 2^{25}(t_{T3} + 2^5 t_{T2})). \tag{10}$$

The final time complexity depends on how the time to do one S-box look-up compares to the time to access $\mathcal{T}_2$ and $\mathcal{T}_3$. These ratios will vary depending on the implementation. We have implemented the tables and measured the times to do look-ups in them as well as the time to do one S-box look-up. On the computer we used we found that $t_{T2} \approx 1.1626 \times t_{SB}$ and $t_{T3} \approx 32 \times t_{T2}$. Using these values in (10) we get the total time complexity of our attack to be about $2^{95.44}$ S-box look-ups which can be translated to $2^{88.85}$ 6-round PRINCE encryptions.

Algorithm 1 Accelerated exhaustive search attack using memory

for $X \in \mathbb{F}_2^{64}$ **do**
  Calculate value of X' related to X;
  for $K'_{C3} \in \mathbb{F}_2^{25}$ **do**
    Find the 15 bits K''_{C3};
    Calculate the 20 information bits v_1 of K_{C3} shared by K_{C1};
    Store the other 20 information bits of K_{C3} in $\mathcal{T}_3[v_1]$;
  end for
  for $K'_{C2} \in \mathbb{F}_2^{25}$ **do**
    Find the 15 bits of K''_{C2};
    Calculate the 36 information bits (v_2, v_3) of K_{C2} shared by (K_{C1}, K_{C3}), $|v_2| = 25, |v_3| = 11$;
    Let v_4 be the 4 information bits of K_{C2} not in common with (K_{C1}, K_{C3});
    Store (v_3, v_4) in $\mathcal{T}_2[v_2]$;
  end for
  for $K'_{C4} \in \mathbb{F}_2^{25}$ **do**
    Find the 14 bits of K''_{C4};
    Store K''_{C4} in $\mathcal{T}_4[K'_{C4}]$;
  end for
  for $K'_{C1} \in \mathbb{F}_2^{25}$ **do**
    Find K''_{C1};
    Calculate v'_1, the 20 information bits of K_{C1} shared with K_{C3};
    Find matching candidates (K_{C1}, K_{C3}) from $\mathcal{T}_3[v'_1]$;
    for every candidate (K_{C1}, K_{C3}) **do**
      Calculate (v'_2, v'_3), the 36 information bits of (K_{C1}, K_{C3}) shared with K_{C2};
      Find $\mathcal{T}_2[v'_2] = (v_3, v_4)$;
      if $v'_3 = v_3$ **then**
        Calculate $v'_5 = K'_{C4}$ and $v'_6 = K''_{C4}$ from (K_{C1}, K_{C2}, K_{C3});
        Find $\mathcal{T}_4[v'_5] = v_6$;
        if $v'_6 = v_6$ **then**
          Calculate value P' and C' using X/X' and K_1;
          if (3) holds **then**
            Calculate value of K_0 from K_1 and P';
            Check (K_0, K_1) on second P/C pair;
            if (K_0, K_1) matches second P/C pair **then**
              (K_0, K_1) is the secret key;
            end if
          end if
        end if
      end if
    end for
  end for
end for

5 Conclusions

In this paper we have analysed PRINCE in the limited setting of a 6-round version using two known plaintext/ciphertext pairs. We have shown that in this

scenario it is possible to reject a wrong guess of K_1' after guessing 35 of the 64 unknown bits when we know the middle states X and X'. A basic guess-and-determine attack where we also try to minimize the number of S-box look-ups needed will then succeed with time complexity equivalent to $2^{96.78}$ encryptions.

Trading some of the time with memory, we have shown how to store all possible values for parts of K_1' in tables that will speed up the identification of wrong guesses. Contrary to many other time/memory trade-off attacks, the memory needed is quite modest and not a limiting factor in practice. The exact complexity for this attack is implementation dependent, but has been shown to be $2^{88.85}$ encryptions on a normal PC.

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. P. Derbez, L. Perrin. *Meet-in-the-Middle Attacks and Structural Analysis of Round-Reduced PRINCE*, IACR Cryptology ePrint Archive, Report 2015/239, 2015.
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. 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.
6. P. Morawiecki. *Practical Attacks on the Round-reduced PRINCE*, IACR Cryptology ePrint Archive, Report 2015/245, 2015.
7. Sh. Rasoolzadeh and H. Raddum. *Cryptanalysis of PRINCE with Minimal Data*, AfricaCrypt 2016, LNCS, vol. 9646, pp. 109 – 126, Springer 2016.

A Linear Relations Between K'' and K'

$$
\begin{aligned}
k''_{63} &= k'_{59} \oplus k'_{55} \oplus k'_{51} \oplus k'_{28} \oplus k'_{24} \oplus k'_{20} & k''_{62} &= k'_{62} \oplus k'_{59} \oplus k'_{55} \oplus k'_{54} \oplus k'_{51} \oplus k'_{50} \\
k''_{61} &= k'_{62} \oplus k'_{61} \oplus k'_{57} \oplus k'_{54} \oplus k'_{50} \oplus k'_{49} & k''_{60} &= k'_{61} \oplus k'_{60} \oplus k'_{57} \oplus k'_{56} \oplus k'_{52} \oplus k'_{49} \\
k''_{59} &= k'_{60} \oplus k'_{56} \oplus k'_{52} \oplus k'_{47} \oplus k'_{43} \oplus k'_{35} & k''_{58} &= k'_{47} \oplus k'_{46} \oplus k'_{43} \oplus k'_{42} \oplus k'_{38} \oplus k'_{35} \\
k''_{57} &= k'_{46} \oplus k'_{42} \oplus k'_{41} \oplus k'_{38} \oplus k'_{37} \oplus k'_{33} & k''_{56} &= k'_{44} \oplus k'_{41} \oplus k'_{37} \oplus k'_{36} \oplus k'_{33} \oplus k'_{32} \\
k''_{55} &= k'_{44} \oplus k'_{36} \oplus k'_{32} \oplus k'_{31} \oplus k'_{23} \oplus k'_{19} & k''_{54} &= k'_{31} \oplus k'_{30} \oplus k'_{26} \oplus k'_{23} \oplus k'_{19} \oplus k'_{18} \\
k''_{53} &= k'_{30} \oplus k'_{29} \oplus k'_{26} \oplus k'_{25} \oplus k'_{21} \oplus k'_{18} & k''_{52} &= k'_{29} \oplus k'_{25} \oplus k'_{24} \oplus k'_{21} \oplus k'_{20} \oplus k'_{16} \\
k''_{51} &= k'_{24} \oplus k'_{20} \oplus k'_{16} \oplus k'_{15} \oplus k'_{7} \oplus k'_{3} & k''_{50} &= k'_{15} \oplus k'_{14} \oplus k'_{10} \oplus k'_{7} \oplus k'_{3} \oplus k'_{2} \\
k''_{49} &= k'_{14} \oplus k'_{13} \oplus k'_{10} \oplus k'_{9} \oplus k'_{5} \oplus k'_{2} & k''_{48} &= k'_{13} \oplus k'_{9} \oplus k'_{8} \oplus k'_{5} \oplus k'_{4} \oplus k'_{0} \\
k''_{47} &= k'_{47} \oplus k'_{43} \oplus k'_{39} \oplus k'_{8} \oplus k'_{4} \oplus k'_{0} & k''_{46} &= k'_{47} \oplus k'_{43} \oplus k'_{42} \oplus k'_{39} \oplus k'_{38} \oplus k'_{34} \\
k''_{45} &= k'_{45} \oplus k'_{42} \oplus k'_{38} \oplus k'_{37} \oplus k'_{34} \oplus k'_{33} & k''_{44} &= k'_{45} \oplus k'_{44} \oplus k'_{40} \oplus k'_{37} \oplus k'_{33} \oplus k'_{32} \\
k''_{43} &= k'_{44} \oplus k'_{40} \oplus k'_{32} \oplus k'_{31} \oplus k'_{27} \oplus k'_{19} & k''_{42} &= k'_{31} \oplus k'_{30} \oplus k'_{27} \oplus k'_{26} \oplus k'_{22} \oplus k'_{19} \\
k''_{41} &= k'_{30} \oplus k'_{26} \oplus k'_{25} \oplus k'_{22} \oplus k'_{21} \oplus k'_{17} & k''_{40} &= k'_{28} \oplus k'_{25} \oplus k'_{21} \oplus k'_{20} \oplus k'_{17} \oplus k'_{16} \\
k''_{39} &= k'_{28} \oplus k'_{20} \oplus k'_{16} \oplus k'_{15} \oplus k'_{11} \oplus k'_{3} & k''_{38} &= k'_{15} \oplus k'_{14} \oplus k'_{11} \oplus k'_{10} \oplus k'_{6} \oplus k'_{3} \\
k''_{37} &= k'_{14} \oplus k'_{10} \oplus k'_{9} \oplus k'_{6} \oplus k'_{5} \oplus k'_{1} & k''_{36} &= k'_{12} \oplus k'_{9} \oplus k'_{5} \oplus k'_{4} \oplus k'_{1} \oplus k'_{0} \\
k''_{35} &= k'_{63} \oplus k'_{55} \oplus k'_{51} \oplus k'_{12} \oplus k'_{4} \oplus k'_{0} & k''_{34} &= k'_{63} \oplus k'_{62} \oplus k'_{58} \oplus k'_{55} \oplus k'_{51} \oplus k'_{50} \\
k''_{33} &= k'_{62} \oplus k'_{61} \oplus k'_{58} \oplus k'_{57} \oplus k'_{53} \oplus k'_{50} & k''_{32} &= k'_{61} \oplus k'_{57} \oplus k'_{56} \oplus k'_{53} \oplus k'_{52} \oplus k'_{48} \\
k''_{31} &= k'_{56} \oplus k'_{52} \oplus k'_{48} \oplus k'_{31} \oplus k'_{27} \oplus k'_{23} & k''_{30} &= k'_{31} \oplus k'_{27} \oplus k'_{26} \oplus k'_{23} \oplus k'_{22} \oplus k'_{18} \\
k''_{29} &= k'_{29} \oplus k'_{26} \oplus k'_{22} \oplus k'_{21} \oplus k'_{18} \oplus k'_{17} & k''_{28} &= k'_{29} \oplus k'_{28} \oplus k'_{24} \oplus k'_{21} \oplus k'_{17} \oplus k'_{16} \\
k''_{27} &= k'_{28} \oplus k'_{24} \oplus k'_{16} \oplus k'_{15} \oplus k'_{11} \oplus k'_{7} & k''_{26} &= k'_{15} \oplus k'_{11} \oplus k'_{10} \oplus k'_{7} \oplus k'_{6} \oplus k'_{2} \\
k''_{25} &= k'_{13} \oplus k'_{10} \oplus k'_{6} \oplus k'_{5} \oplus k'_{2} \oplus k'_{1} & k''_{24} &= k'_{13} \oplus k'_{12} \oplus k'_{8} \oplus k'_{5} \oplus k'_{1} \oplus k'_{0} \\
k''_{23} &= k'_{63} \oplus k'_{59} \oplus k'_{51} \oplus k'_{12} \oplus k'_{8} \oplus k'_{0} & k''_{22} &= k'_{63} \oplus k'_{62} \oplus k'_{59} \oplus k'_{58} \oplus k'_{54} \oplus k'_{51} \\
k''_{21} &= k'_{62} \oplus k'_{58} \oplus k'_{57} \oplus k'_{54} \oplus k'_{53} \oplus k'_{49} & k''_{20} &= k'_{60} \oplus k'_{57} \oplus k'_{53} \oplus k'_{52} \oplus k'_{49} \oplus k'_{48} \\
k''_{19} &= k'_{60} \oplus k'_{52} \oplus k'_{48} \oplus k'_{43} \oplus k'_{39} \oplus k'_{35} & k''_{18} &= k'_{46} \oplus k'_{43} \oplus k'_{39} \oplus k'_{38} \oplus k'_{35} \oplus k'_{34} \\
k''_{17} &= k'_{46} \oplus k'_{45} \oplus k'_{41} \oplus k'_{38} \oplus k'_{34} \oplus k'_{33} & k''_{16} &= k'_{45} \oplus k'_{44} \oplus k'_{41} \oplus k'_{40} \oplus k'_{36} \oplus k'_{33} \\
k''_{15} &= k'_{44} \oplus k'_{40} \oplus k'_{36} \oplus k'_{11} \oplus k'_{7} \oplus k'_{3} & k''_{14} &= k'_{14} \oplus k'_{11} \oplus k'_{7} \oplus k'_{6} \oplus k'_{3} \oplus k'_{2} \\
k''_{13} &= k'_{14} \oplus k'_{13} \oplus k'_{9} \oplus k'_{6} \oplus k'_{2} \oplus k'_{1} & k''_{12} &= k'_{13} \oplus k'_{12} \oplus k'_{9} \oplus k'_{8} \oplus k'_{4} \oplus k'_{1} \\
k''_{11} &= k'_{63} \oplus k'_{59} \oplus k'_{55} \oplus k'_{12} \oplus k'_{8} \oplus k'_{4} & k''_{10} &= k'_{63} \oplus k'_{59} \oplus k'_{58} \oplus k'_{55} \oplus k'_{54} \oplus k'_{50} \\
k''_{9} &= k'_{61} \oplus k'_{58} \oplus k'_{54} \oplus k'_{53} \oplus k'_{50} \oplus k'_{49} & k''_{8} &= k'_{61} \oplus k'_{60} \oplus k'_{56} \oplus k'_{53} \oplus k'_{49} \oplus k'_{48} \\
k''_{7} &= k'_{60} \oplus k'_{56} \oplus k'_{48} \oplus k'_{47} \oplus k'_{39} \oplus k'_{35} & k''_{6} &= k'_{47} \oplus k'_{46} \oplus k'_{42} \oplus k'_{39} \oplus k'_{35} \oplus k'_{34} \\
k''_{5} &= k'_{46} \oplus k'_{45} \oplus k'_{42} \oplus k'_{41} \oplus k'_{37} \oplus k'_{34} & k''_{4} &= k'_{45} \oplus k'_{41} \oplus k'_{40} \oplus k'_{37} \oplus k'_{36} \oplus k'_{32} \\
k''_{3} &= k'_{40} \oplus k'_{36} \oplus k'_{32} \oplus k'_{27} \oplus k'_{23} \oplus k'_{19} & k''_{2} &= k'_{30} \oplus k'_{27} \oplus k'_{23} \oplus k'_{22} \oplus k'_{19} \oplus k'_{18} \\
k''_{1} &= k'_{30} \oplus k'_{29} \oplus k'_{25} \oplus k'_{22} \oplus k'_{18} \oplus k'_{17} & k''_{0} &= k'_{59} \oplus k'_{55} \oplus k'_{51} \oplus k'_{29} \oplus k'_{28} \oplus k'_{25} \oplus k'_{24} \oplus k'_{20} \oplus k'_{17}
\end{aligned}
$$