Fault Analysis of the Prince Family of Lightweight Ciphers

Final Thesis submitted to the

Indian Statistical Institute, Kolkata

For award of the degree

of

Masters of Technology in Cryptology and Security

by

Anup Kumar Kundu

[Roll No: CrS1909]

Under the guidance of

Dr. Dhiman Saha

Department of Electrical Engineering and Computer Science
Indian Institute of Technology, Bhilai

Institute Supervisor

Dr. Goutam Paul

Cryptology & Security Research Unit (CSRU)
Indian Statistical Institute, Kolkata

Cryptology and Security Research Unit

INDIAN STATISTICAL INSTITUTE, KOLKATA

JULY 2021

Certificate

This is to certify that the thesis entitled **"Fault Analysis of the Prince Family of Lightweight Ciphers"** submitted by **Anup Kumar Kundu (CrS1909)** to the Indian Statistical Institute, Kolkata, is a record of bonafide research work carried under my supervision and is worthy of consideration for the degree of Master of Cryptology and Security of the Institute.

Date: 09.07.2021

Place: ISI Kolkata, India

Dr. Dhiman Saha

Acknowledgements

Firstly I want to really thank from the core of my heart to my guide Dr. Dhiman Saha, Assistant Professor, Department of EECS, IIT Bhilai, India. His guidance, enthusiastic encouragement, tolerance of my questions about arbitrary things from various domains and way of giving the answers about all of them patiently really inspire and motivated me a lot. I will be really blessed if I can carry the ways, like how he connects to all his students and other members as friends and remains enthusiastic for all time uniformly from morning to night, through me. I am privileged to have such a support me both academically and personally.

I would also like to thank Banashri Karmakar, Department of EECS, IIT Bhilai, India and Aikata, The Institute of Applied Information Processing and Communications (IAIK), Graz University of Technology, Styria, Austria to support me in my research work all the time and help me to proceed with my work further.

My grateful thanks are also extended to all the lab members of here and Mostafizar Rahman, CSRU, ISI, Kolkata for helping me initially to avoid the initial obstacles and get the momentum towards research. They keep motivated me all the time during the COVID situation also.

Finally, I thank my teachers and friends in ISI and my family members who keep supporting me in all my endeavors and in the toughest period of COVID-19 pandemic situation.

Anup Kumar Kundu

Indian Statistical Institute
Kolkata - 700108 , India.

Abstract

Privacy is one of the most fundamental aspects of the digital age that we live in. With the advent of the Internet and the advances in both nano-scale electronics and communication technologies, data has become the new oil. And wherever there is data there is a notion of its privacy. Whether data is at rest or in motion, privacy and authenticity have always been the hallmark of modern day communication. Cryptography provides us the necessary tools and primitives that help us achieve among others, the goals of privacy, integrity authenticity in isolation and more recently even simultaneously. While conventional crypto tackles most of the problems efficiently, it has been seen to be particularly, *not* suitable for resource constrained environments which are being increasingly prevalent in present-day Internet-of-Thing (IoT) environments, RFID tags so on and so forth. This is primarily attributed to the fact that traditional crypto is "heavy-weight" in terms of the computational resources that it demands, be it in terms of chip-area, power-consumption, throughputs etc and hence become unusable or overwhelming for devices that operated on limited resources. This points us in the direction of a new typo of crypto which is referred to as "Lightweight" crypto. Lightweight Cryptographic algorithms are tailored for resource starved settings and hence perform better in such environments. The importance of lightweight crypto is evidence by the on-going multi-year global competition by NIST for standardizing the next generation lightweight authenticated ciphers and presently in the final round.

This work consists cryptanalysis of two lightweight block ciphers namely `PRINCE`, and `PRINCEv2` which are based on the SPN design philosophy. `PRINCE` has been around for some time and is proposed keeping in mind unrolled implementations. `PRINCEv2` is the new version of `PRINCE` which was reported in SAC 2020. In the current work, we introduce a new fault attack on `PRINCE` based on the random bit-model where faults are injected in the input of 10 th round. The attack is able to uniquely recover the key using 7 faults. It is interesting to see that the random bit-fault model which is a popular fault model has not yet been explored independently on PRINCE. Though Song and Fu [SH13] have explored the random-nibble fault and mentioned the bit-model to be a special case, they actually fail to capture the full scenario. Herein lies the motivation of the current work. We look at the bit-model in isolation and in-depth and conclude that it is more effective both in terms of the point at which the fault is injected as well as the complexity of the resulting DFA. In terms of the point of fault injection it is important to emphasize that in the attack reported in [SH13], the fault is actually injected before/during the Sub-

Byte-Inverse operation in the 10 th round which is the last operation of the 10 th round. Thus it will be more appropriate to state the fault injection point to be 10.5 rounds at best instead of 10 rounds as claimed by the authors in [SH13] (Refer Fig. 3.7). We touch upon this aspect in details in the discussion section later in this work. On the contrary, the random bit-flip DFA proposed here actually induces the fault at the input of 10 th round. The work further gives a classification of fault-invariants that are generated at the end of 11 th round due to a random bit-fault at the beginning of 10 th round. Further, PRINCEv2 was introduced with many modifications primarily in the key-schedule to thwart many classical attacks on PRINCE. We investigated PRINCEv2 in the light of the current work and found that PRINCEv2 is equally vulnerable to all attacks reported here. Finally, we look at PRINCE-like ciphers in general and comment on the impact of the α-reflection property on the amplification of the scope of fault injection.

Contents

CHAPTER 1

Introduction

The word "Cryptology" comes from greek word 'kryptós', which means hidden or secret and the word 'graphein', which means to write or to study. So, Cryptology is the study of techniques secretly. Security and privacy are major issues in the modern digital era. Nowadays almost everything is online. Therefore secure communication is a point of concern. Otherwise, privacy will be lost. Cryptology is such a state-of-art which deals with security and privacy of digital data. That's why we need cryptology so much in today.

Formally, a cryptosystem is a five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where :

- $\mathcal{P}$ is a finite set of possible plaintexts
- $\mathcal{C}$ is a finite set of possible ciphertexts
- $\mathcal{K}$, the key space, is a finite set of possible keys
- For each $K \in \mathcal{K}$, there is an encryption rule $e_K \in \mathcal{E}$ and a corresponding decryption rule $d_K \in \mathcal{D}$. Each $e_K : \mathcal{P} \to \mathcal{C}$ and $d_K : \mathcal{C} \to \mathcal{P}$ are functions such that $d_K(e_K(x)) = x$ for every plaintext element $x \in \mathcal{P}$

Lightweight cryptography is a part of cryptology, wants to secure the devices with minimum resources (memory, power, size). The process of providing privacy in the systems with low cost is the main focus of lightweight cryptography. The word "Cryptanalysis" comes from 'kryptós' which means "hidden", and 'analýein' means "to analyze". So, Cryptanalysis is the analysis of hidden things in the system. It deals with algorithms that try to know the key for a given encrypted message. In general, it can be divided into two, symmetric key cryptosystem, asymmetric key cryptosystem. In the symmetric key, same key is used for encryption and decryption purpose. For asymmetric key, encryption is done by one key, and decryption is by other. Symmetric key cryptosystem can

be divided into two kinds of ciphers, block ciphers and, stream ciphers. Block cipher operates on blocks. It takes an input block of size n as a message, a key of size k, and outputs a block of size n as ciphertext. Stream ciphers encrypt the input stream as one character at a time with the corresponding keystream and output a ciphertext stream.

There are various types of block ciphers. SPN (substitution-permutation networks) and Feistel ciphers are two of them. SPN takes a block of plain text and key as input. It applies substitution and permutation alternatively for a specified number of rounds to make cipher text. In Feistel cipher, the original block of plain text is divided into two parts. The round function is applied to one and its output is XORed with the other part of the plaintext. Then at the end of one round, two parts are swapped. This process is repeated for some number of rounds to make the cipher text. In this work, we focus on PRINCE, PRINCEv2, and SIMON.

PRINCE [BCG+12] was introduced at Asiacrypt 2012 by the Technical University of Denmark (DTU), NXP Semiconductors, and the Ruhr University Bochum. It is an SPN structure. PRINCE became very popular due to its impressive design. AES [DR02] and some other block ciphers take some time to execute their blocks. Due to this reason, the overall computation time increases. If one block executes in one clock cycle, then block ciphers need some clock cycle to compute the ciphertext. PRINCE has unrolled design. Therefore by slowing down the clock cycle, we can execute encryption or decryption of PRINCE within just one clock cycle for every type of machine, whose computation power is slow with whose is fast. The cipher has an α-reflection property, for this encryption with one key, is equivalent to decryption with a related key. PRINCE developers made the usage of the S-Box flexible. They gave a family of S-Boxes(8 many). One can use any of those 8 S-Boxes in PRINCE depending upon the suitability of hardware. As PRINCE has α-reflection property, so implementation of decryption oracle takes a small area in the chip.

PRINCE follows FX construction with a 64-bits block size, which uses a 128-bit key. The key is a composition of two 64-bit keys, k_0 and k_1. It has 12 unrolled rounds, five forward rounds, one middle round, and five backward rounds. Note that, the middle round is of two rounds. After that PRINCE uses a whitening key operation. Each forward round has a S-Box layer, a M-layer, a shift row operation and, XORing of a round constant with 64 bit key. The Middle round has one S-Box layer, M-layer, and one S^{-1}-layer. The Backward round has XORing of a round constant with 64-bit key, one M-layer and one S^{-1}-layer.

The difference between PRINCEv2 [BEK+20] from PRINCE is, PRINCEv2 does not follow the FX construction. For this cipher, two 64-bit keys are used in $\text{PRINCE}_{\text{CORE}}$ and the middle round becomes a keyed function. Along with these, the PRINCEv2 does not have the α-reflection property.

SIMON [BSS+13] is another lightweight cipher, based on the Feistel structure. The SIMON family is denoted by SIMON$2n/mn$, where $2n$ is the block size and mn the master key size. It has three members: SIMON$32/64$, SIMON$48/96$ and SIMON$64/128$.

Suppose the plaintext at the $i-1$-th round is (L^{i-1}, R^{i-1}), where $L^{i-1} = \left\{X_n^{i-1}, X_{n+1}^{i-1}, \cdots X_{2n-1}^{i-1}\right\}$ and $R^{i-1} = \left\{X_0^{i-1}, X_1^{i-1}, \cdots X_{n-1}^{i-1}\right\}$ and the subkey is $K^{i-1} = \left\{K_0^{i-1}, K_1^{i-1}, \cdots K_{n-1}^{i-1}\right\}$. Then round function is

$$(L_i, R_i) = (R^{i-1} \oplus F(L^{i-1}) \oplus K^{i-1}, L^{i-1})$$

where

$$F(x) = ((x \lll a) \wedge (x \lll b)) \oplus (x \lll c)$$

for $i = 1, \cdots n_r$. For SIMON, value of $a = 1, b = 8, c = 2$.

The Prince Challenge

To know how much the cipher PRINCE is secure against the attacks, the Technical University of Denmark (DTU), NXP Semiconductors, and the Ruhr University Bochum proposed PRINCE challenge [pri14]. The academic researchers and the industrialists all were welcomed equally in this competition. They categorized the challenge depending upon the chosen plaintext attack scenario, known-plaintext attack scenario, and how fast the cipher can be attacked in reduced rounds. They have declared winners depending upon the attacks.

NIST LWC Competition

NIST initiated a process to evaluate, and standardize lightweight cryptographic algorithms that are suitable for use in various kinds of aspects [nis] like cybersecurity, health and bioscience etc. They considered 57 submissions and 56 submissions were selected as possible candidates for round 1 and out of them, 32 remained for Round 2. This year, NIST finalized some ciphers. They selected 10 ciphers as finalists of the round. They are ASCON [DEMS16], Elephant [BCDM20], GIFT-COFB [RD19], Grain128-AEAD [HJM+19], ISAP [DEM+20], Photon-Beetle [BCD+19], Romulus [IKMP20], Sparkle [BBdS+19], TinyJambu [WH19],

and Xoodyak [DHP+20]. ACE was selected in the NIST round 1 and round 2 candidate list. But could not be able to clear the next one. We discuss ACE briefly in our literature survey.

Contribution of the Thesis

- New random bit-flip based fault attack
- The attack mounted is based on practical complexities and feasible attack models and lead to unique key-recovery attacks.
- Identification of fault-invariants for random bit-flip model.
- The attack is implemented using C and verified using simulations.
- All attacks also verified to work on PRINCEv2 with no change in complexities
- Discussion on amplification of scope of fault injection attributed to PRINCE-like constructions

Organization of the Thesis

This thesis is organized as follows: Chapter 2 of the thesis mainly contains a brief survey on PRINCE, SIMON, and PRINCEv2. For the ciphers PRINCE and SIMON, descriptions of them and some attacks on them are illustrated in Section 2.1.1, 2.3.1 and 2.1.2, 2.3.2 of Chapters 2.1 and 2.3 respectively. Chapter 3 presents our fault attack on the cipher PRINCE. We found a 2 round bit-based distinguisher and applying that, we attack the blocked cipher PRINCE. It is a random bit fault attack. This attack is described fully in the Chapter. Then after the discussion about future work, we conclude the thesis in Chapter 3.7.

CHAPTER 2

Literature Survey

In this chapter, we study some lightweight ciphers, mainly PRINCE, PRINCEv2, and SIMON. We give a brief description of ACE and SIMECK also. PRINCE and PRINCEv2 are based on the SPN structure. Whereas the other ciphers are based on Feistel structure. After the discussion, we give some attacks on PRINCE and SIMON. As PRINCEv2 is recently introduced, so there is no attack on the new version. But designers gave some security analysis of PRINCEv2. We illustrate some of them also in the chapter. There is a section regarding fault analysis in the chapter where we discuss some of the fault models.

2.1 PRINCE [BCG+12]

PRINCE [BCG+12] is a lightweight cipher, which was designed by keeping in mind the unrolled behavior. Another most interesting part of the cipher is, decryption of one key is equivalent to encryption of a related key. So this decryption oracle can be implemented in hardware from the encryption oracle with a minimum cost. The cipher is described in this section. Then we discuss some attacks on it with a table, which tells the overview of the attacks on PRINCE. Then discussing some interesting properties of PRINCE, we conclude the section.

2.1.1 Description of PRINCE

PRINCE is a lightweight block cipher that follows the FX construction with a block size of 64 bits and a key size of 128 bits. The 128-bit key $(K = K_0||K_1)$ is divided into two keys, where each is of size 64 bits.

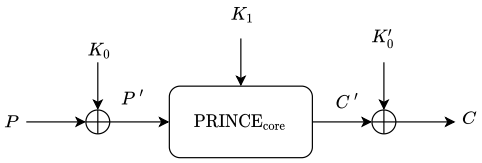


Figure 2.1: FX Construction of PRINCE

The first part of the key is K_0, and the second is k_1. K_1 is used as the round key for the core of PRINCE, which is named as PRINCE$_{\text{core}}$ and K_0 is used as a whitening key to the outside of the core structure.

If we denote the plaintext/ciphertext pair of PRINCE and PRINCE$_{\text{core}}$ by P/C and P'/C' respectively then, $P = P \oplus K_0$, $C = C' \oplus K_0'$, where K_0' from K_0 by the following linear mapping:

$$K_0' = (K_0 \gg 1) \oplus (K_0 \gg 63)$$

The encryption function iterates the round function $\mathcal{R}$ five times, then applies the middle layer $\mathcal{R}'$, which is followed by five applications of the inverse round function $\mathcal{R}^{-1}$.

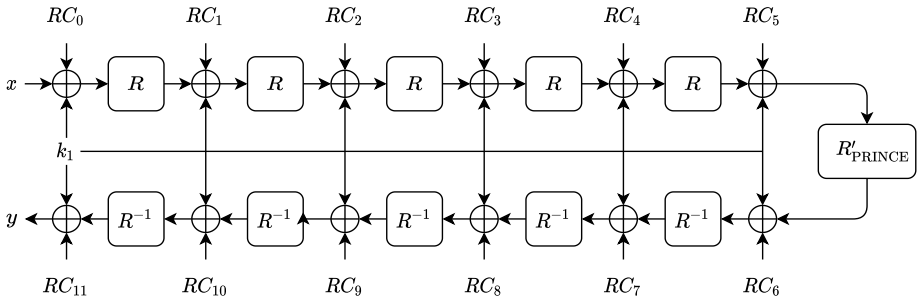


Figure 2.2: Diagram of PRINCE

• **S-Box** : After the linear layer MC and shift row SR, the round function applies an S-box layer SB. The inverse S-Box is applied to the inverse round function. There are 8 Affine equivalent classes for the S-box of PRINCE family. Anyone can be chosen from these classes. The S-box used in the PRINCE proposal :

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
SB[x]	B	F	3	2	A	C	9	1	6	7	8	0	E	5	D	4

• **Shift Row** : The ShiftRows permutation applied in SR is the following:

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

Here i-th row of the state is cyclically rotated to the left of by i positions (similar to AES).

- **Matrix Multiplication (MC)** : The MC operation is built from the following four 4 $\times$ 4 matrices:

$$M_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad M_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$M_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad M_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

or in other words M_i is the 4 $\times$ 4 identity matrix where the i-th row is replaced by the zero vector. From these M_i we build the matrices $\widehat{M}^{(0)}$, $\widehat{M}^{(1)}$ and M'

$$\widehat{M}^{(0)} = \begin{pmatrix} M_1 & M_2 & M_3 & M_4 \\ M_2 & M_3 & M_4 & M_1 \\ M_3 & M_4 & M_1 & M_2 \\ M_4 & M_1 & M_2 & M_3 \end{pmatrix}, \qquad \widehat{M}^{(1)} = \begin{pmatrix} M_2 & M_3 & M_4 & M_1 \\ M_3 & M_4 & M_1 & M_2 \\ M_4 & M_1 & M_2 & M_3 \\ M_1 & M_2 & M_3 & M_4 \end{pmatrix},$$

$$M' = \begin{pmatrix} \widehat{M}^{(0)} & 0 & 0 & 0 \\ 0 & \widehat{M}^{(1)} & 0 & 0 \\ 0 & 0 & \widehat{M}^{(1)} & 0 \\ 0 & 0 & 0 & \widehat{M}^{(0)} \end{pmatrix}$$

i.e., M' is the 64$\times$64 block diagonal matrix with blocks ($\widehat{M}^{(0)}$, $\widehat{M}^{(1)}$, $\widehat{M}^{(1)}$, $\widehat{M}^{(0)}$). Finally, the MC-layer multiplies the state with M' and is an involution.

Therefore for PRINCE we thus have the structure, where $R = SR \circ MC \circ SB$, $R'_{\texttt{PRINCE}} = SB^{-1} \circ MC \circ SB$ and $R'^{-1}_{\texttt{PRINCE}} = SB^{-1} \circ MC \circ SR^{-1}$.

- **Round Constant** : Round constants are RC_i , where $0 \leq i \leq 11$, which are XORed with the states. The round constants used in the backward rounds are related to the round constants of the forward rounds by α XOR difference, where $\alpha =$ 0xc0ac29b7c97c50dd, i.e.

$$RC_i \oplus RC_{11-i} = \alpha, \quad [0 \leq i \leq 11]$$

RC_0	0x0000000000000000
RC_1	0x13198a2e03707344
RC_2	0xa4093822299f31d0
RC_3	0x082efa98ec4e6c89
RC_4	0x452821e638d01377
RC_5	0xbe5466cf34e90c6c
RC_6	0x7ef84f78fd955cb1
RC_7	0x85840851f1ac43aa
RC_8	0xc882d32f25323c54
RC_9	0x64a51195e0e3610d
RC_{10}	0xd3b5a399ca0c2399
RC_{11}	0xc0ac29b7c97c50dd

2.1.2 Summary of Some Attacks on PRINCE

After PRINCE was introduced at ASIACRYPT 2012, many attacks were performed on it. Some authors exploit its keyless middle rounds, almost MDS matrix which is used in its Matrix Multiplication (MC). As the MC is an involutory matrix, some designers take an advantage of this too. Overall, the maximum number of rounds that are attacked in PRINCE is 12, but the complexities are high enough for those attacks.

Biclique Attack [ALL12]: In the paper, due to two applications of the linear layer can active the entire state, designers build a biclique to reduce the data complexity over a single round. They use the matching-with-precomputations approach in the paper. After two rounds we can get full diffusion because of the key differences in the matching part. But for the partial matching, here they consider only three rounds in full.

Differential Attack [ALL12]: To do the differential attack for 2 rounds and 4 rounds authors applied Inside-Out Attack on Two Rounds of $\texttt{PRINCE}_{\text{core}}$. For one forward round, the middle part, and one backward round they proposed a differential trail for a 1-x-1 construction. They break down the attack in some parts like preparation, oracle queries, derivation, discard mismatching pairs, derive solutions for the S-box trails, derive key candidates and eliminate false positives, and derive the key.

Multiple Differential Attack [CFG+14]: Here authors observe some properties of PRINCE. From there they describe the first attack on 10- round PRINCE, which takes $2^{57.94}$ chosen plaintexts, with time complexity less than $2^{60.61}$ encryptions which goes to a product Data $\times$ Time around $2^{118.56}$.

Sieve-in-the-Middle [CNV13]: This paper gives a new progression of Meet-in-the-Middle (MITM) algorithms that allows the attacker to attack in a higher number of rounds. Authors enumerate some input and output bits of a particular middle sbox S without searching the collisions in the middle. If a key does not give a valid transition through S, they discard the corresponding value. Here the technique can be used to attack more rounds than classical MITM as it covers the rounds corresponding to the middle sbox.

Meet-in-the-Middle [LJW13]: In this paper, the authors give a technique through which they can do a 9-round attack on AES-192 by using a 5-round distinguisher. They use this process to do an 8 round attack on PRINCEcore with the help of a 6-round distinguisher. The data, time, and memory complexities of the attack are about 2^{53}, 2^{53}, and 2^{28} 64-bit memories. Then they show that their attack can be extended for an 8-round PRINCE. Along with this, in their paper, they give a bit-based distinguisher to attack 9 rounds of PRINCE.

Boomerang Attack [JNP+13]: Here authors provide the first third-party analysis of the PRINCE cipher. They analyze for related-key attacks the resistance of PRINCE. They show the related-key relations to do a key recovery attack. Their attack requires $(2^{33}, 2^{64}, 2^{33})$ data , time and, memory complexities.

Integral Attack [PN15]: The authors use a 4.5 round distinguisher here. They give 3 round All property (i.e. all 2^{12} many possible plaintexts) and observe Balanced property (i.e. whether the XOR sum of the ciphertexts are 0

Attack	# Rounds	Comlplexity		Reference
		Time	Data	
Differential	2	$2^{32.44}$	2^{32}	[ALL12]
	4	$2^{56.26}$	2^{48}	[ALL12]
Biclique	10	$2^{62.72}$	2^{40}	[ALL12]
Multiple differential	9	$2^{51.21}$	$2^{46.89}$	[CFG+14]
	10	$2^{60.62}$	$2^{57.94}$	[CFG+14]
Sieve-in-the-middle	8	2^{123}	1	[CNV13]
Meet-in-the-middle	8	$2^{66.25}$	2^{16}	[LJW13]
	8	2^{60}	2^{53}	[LJW13]
	9	2^{64}	2^{57}	[LJW13]
Boomerang	12	2^{41}	2^{41}	[JNP+13]
Integral	4	2^{64}	2^{4}	[JNP+15]
	6	$2^{24.6}$	2^{13}	[RR16b]
	6	2^{37}	6×2^{12}	[PN15]
	6	2^{64}	2^{16}	[JNP+15]
Trunc. Diff.	4	$2^{18.25}$	8	[GR16]
Bit-pattern integral	4	2^{28}	6×2^{8}	[Mor17]
Higher-order diff.	7	2^{57}	6×2^{57}	[Mor17]
	7	$2^{44.3}$	2^{33}	[RR16b]
Acc. Exh. Search	7	$2^{96.8}$	2	[RR16a]
Related-Key	12	2^{64}	2^{33}	[JNP+13]
Single-Key	12	$2^{125.47}$	2	[JNP+13]
Diff. / Logic	4	5s	2^{10}	[DP20]

Table 2.1: Overview of Some Attacks on `PRINCE`

or not) after 4.5 rounds. They mount a 6 round attack depending upon the distinguisher with 6×2^{12} chosen plaintexts.

Truncated Differential Attack [GR16]: In this paper, authors show practical key-recovery attacks which are formulated on subspace trails of PRINCE. This is similar to truncated differentials, and they give their focus on the order of ShiftRows and MixLayer operations to give the security of PRINCE-like ciphers. They give constant dimensional subspace trails that are a coset of a plaintext subspace which encrypts to proper subspaces of the state space for many rounds. For setting up the competitive key recovery attacks to round-reduced PRINCE authors start with the found subspace trails. They present two different truncated differential key-recovery attacks on 3 rounds of PRINCE.

Bit-Pattern Integral Attack [Mor17]: The attack uses a collection of plaintexts with some pattern. The basic idea of the attack is to analyze how the set of plaintexts evolve through the encryption algorithm and to use the existence of that property to verify key guesses.

Higher Order Differential Attack [Mor17] [RR16b]: Here authors use the mix column property of PRINCE and check how one nibble all property behaves. They found that the balanced property occurs after 3.5 rounds. They mount an attack depending upon that distinguisher.

2.1.3 Some Properties of PRINCE

This section elaborates on some properties of PRINCE. Section 2.1.3 is discussed about the slow diffusion property. Section 2.1.3 contains integral property, one nibble integral and three nibbles integral are given in two paragraphs. A five round key recovery attack and reflection cryptanalysis is discussed in 2.1.3 and 11 respectively.

Slow Diffusion Property [AKS20]:

For the MC operation, PRINCE uses an almost MDS matrix. For this, it has a slow diffusion property. Single bit difference in the entire state affects three nibbles. The all possible single bit difference transitions are shown in Fig 2.3. Here we start from one single-bit difference at nibble 0 at the beginning and see that it takes 3 rounds for such a difference to diffuse the entire state.

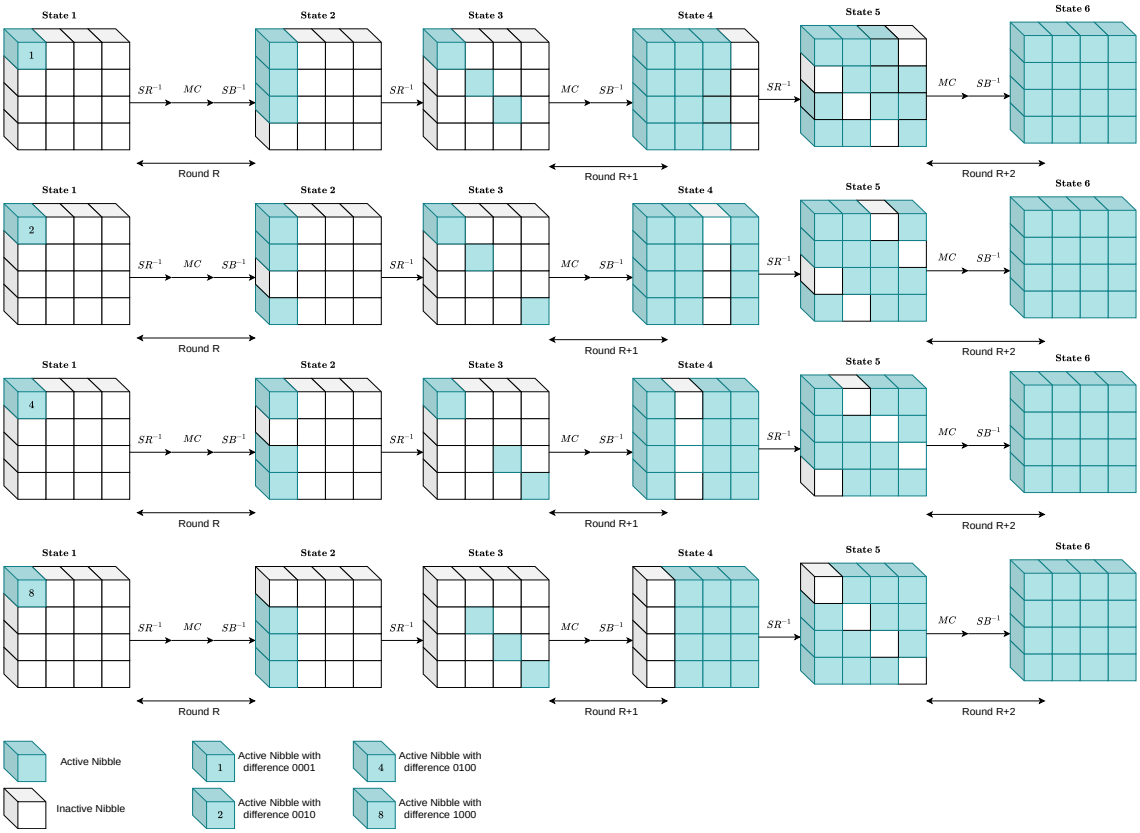


Figure 2.3: Slow Diffusion Property of PRINCE [AKS20]

Integral Property

We discuss two types of integral properties. The first one is about one nibble integral property. The second one is about three nibbles integral property.

One-Nibble Integral Property [AKS20]: A set is called **balanced**, if the XOR-sum of all the encrypted ciphertext of the set is zero. Here, we are giving 1-nibble all property to a nibble and getting balanced property after 3.5 rounds. We are giving 1-nibble all property to state 1. This propagates to state 4 with probability 1. Then after MC, it becomes A^8 along the column.

We can verify this from the bit equations. The bit equations for MC of Prince are:

$\hat{M}^{(0)}$:

$$y_0^0 = x_1^0 \oplus x_2^0 \oplus x_3^0 \quad y_1^0 = x_0^0 \oplus x_1^0 \oplus x_2^0 \quad y_2^0 = x_0^0 \oplus x_1^0 \oplus x_3^0 \quad y_3^0 = x_0^0 \oplus x_2^0 \oplus x_3^0$$

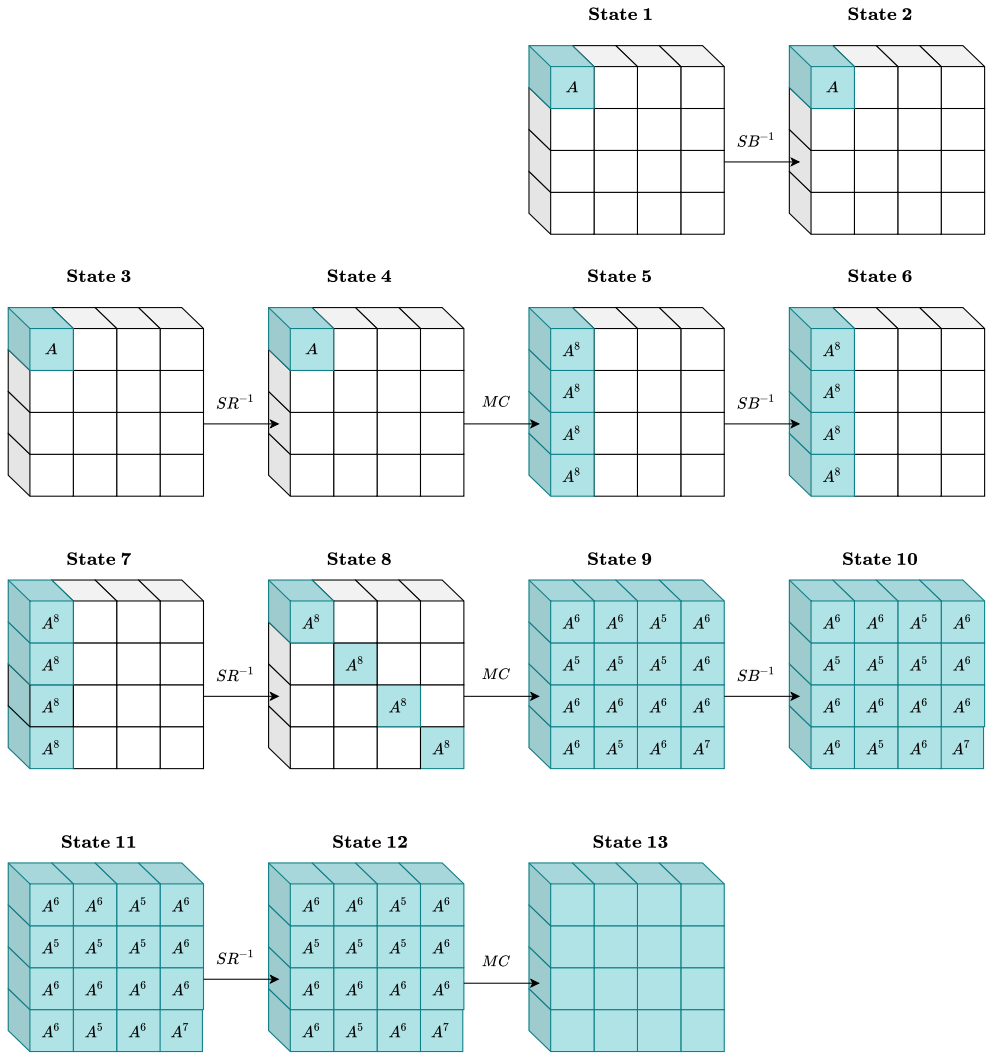


Figure 2.4: 1-Nibble Integral Property of PRINCE [AKS20]

$$y_0^1 = x_0^1 \oplus x_2^1 \oplus x_3^1 \quad y_1^1 = x_1^1 \oplus x_2^1 \oplus x_3^1 \quad y_2^1 = x_0^1 \oplus x_1^1 \oplus x_2^1 \quad y_3^1 = x_0^1 \oplus x_1^1 \oplus x_3^1$$
$$y_0^2 = x_0^2 \oplus x_1^2 \oplus x_3^2 \quad y_1^2 = x_0^2 \oplus x_2^2 \oplus x_3^2 \quad y_2^2 = x_1^2 \oplus x_2^2 \oplus x_3^2 \quad y_3^2 = x_0^2 \oplus x_1^2 \oplus x_2^2$$
$$y_0^3 = x_0^3 \oplus x_1^3 \oplus x_2^3 \quad y_1^3 = x_0^3 \oplus x_1^3 \oplus x_3^3 \quad y_2^3 = x_0^3 \oplus x_2^3 \oplus x_3^3 \quad y_3^3 = x_1^3 \oplus x_2^3 \oplus x_3^3$$

In-state 4 of Fig 2.4, we have all property in nibble 0. So, only those bits will be affected, for which at MC the bits at nibble 0 are involved (we can get the involved bits from the above equations). From the state 6 of Figure 2.4 we can see that at each non-zero difference nibble, there are 3-bit positions that can take 0 and 1. So, the total distinct numbers will be, $2^3 = 8$. So, we will get A^8 after one round.

If we look at the 0-th nibble at state 5 of Fig 2.4, then after 1st MC, it becomes CAAA (bitwise). After SB^{-1}, it will be either {0xb, 0x7, 0x3, 0x2, 0xf, 0xd, 0x8, 0x9} or {0xa, 0x6, 0x4, 0x0, 0x5, 0xe, 0xc, 0x1}. If we take either side, from bit equations after MC, 0-th nibble will be affected by 1st, 2nd, 3rd bit of 0-th nibble. As there are 6 distinct values for those 3 bits, so, there will be 6 distinct values after MC for nibble 0. The explanation will be the same for or side and for other nibbles also. That's why, after another MC, we will get the pattern like state 9 of Fig 2.4. By A^i, we mean, there are i many distinct elements in the nibble.

Three-Nibble Integral Property [PN15]: Here like the same as previous, we are giving All property in three nibbles i.e. 2^{12} many distinct collections of plaintexts and getting balanced after 4.5 rounds (two forward rounds, the middle rounds, and one incomplete backward round — without the S-Layer).

The choice of the three nibbles is not random, it has to be from the set of the type $\{4i, 4i+1, 4i+2, 4i+3\}$, where i ranging from 0 to 3. There are $4 \times \binom{4}{3} = 16$ possible combinations for this. The property will not hold if an additional S-box layer is applied. From this distinguisher, we can implement a 5-round attack on PRINCE, and then we extended this attack to 6-rounds, by adding one final round.

Five-Round Key Recovery Attack [PN15]:

From the above distinguisher, we can mount a 5 round integral attack on PRINCE. We will choose 2^{12} many distinct plaintexts for those 3 nibbles and encrypt them. Then after decrypting one SB layer we will check whether the XOR-sum of those are '0' or not. Depending upon that we will reduce the

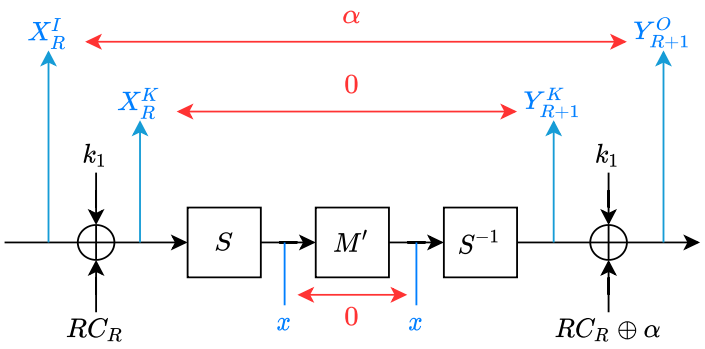
keyspace size and finally reveal the original key.

Algorithm 1: RECOVER_K_0_XOR_K_1

```
Generate a collection of 2^12 plaintexts in which the first three nibbles
  (indexes 0, 1, 2) are active
Encrypt each plaintext from the set using 5-round Prince cipher and
  obtain the collection of 2^12 corresponding ciphertexts
foreach nibble position POS (from 0 to 15) of the key k_0 ⊕ k_1 do
    foreach possible value k_v (from 0 to 15) of the POS nibble
      from k_0 ⊕ k_1 do
        Partially decrypt each of the ciphertexts, by applying the S-layer
        Compute the XOR-sum of the 2^12 obtained nibble values
        if the XOR-sum is 0 then
            keep k_v as probable value of the nibble from position POS
              of the key k_0 ⊕ k_1
        end
    end
end
```

Reflection Cryptanalysis [SBY+13]

Let $f : A \to A$ be a function on a set A. A point $x \in A$ is called a **fixed point** of the function f if and only if $f(x) = x$. The number of fixed points of an involution $f : \mathbb{F}_n^2 \to \mathbb{F}_n^2$ is on the average equal to $2^{n/2}$. We will use the advantage of this property as for PRINCE, the MC matrix is an involution.



$\Pr_X[M'(X) = X]$

Figure 2.5: Characteristics I for Reflection Property

Here as M' is an involution, so if we give input x, then after M' output will also be x. So, the difference will be 0 for this. If we give some input x' in place of X_R^K such that $S(x) = x'$, then after S^{-1} also it will be x'. So, the difference between X_R^K and Y_{R+1}^K is 0, as in Fig 2.5. Then the difference between X_R^I and Y_{R+1}^O is α, as the difference between round constants is α.

Therefore, Characteristics I_1 will hold with probability $\frac{|F_{M'}|}{2^n}$. So, PRINCE has exactly $2^{32} = 2^{n/2}$ fixed points i.e. probability of the characteristic I_1 is then $\frac{2^{32}}{2^{64}} = 2^{-32}$.

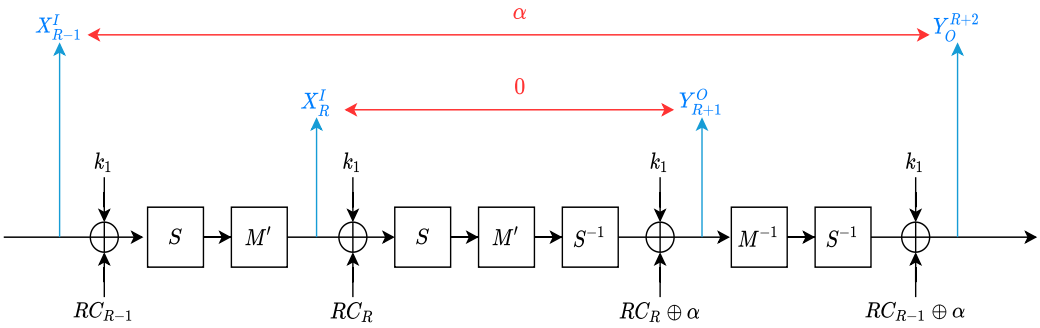


Figure 2.6: Characteristics II for Reflection Property

Similarly, from Fig 2.6 for Characteristics II, we get,

$$P_{I_2} = 2^{-n} \times \#\left\{x \in \mathbb{F}_2^n \mid S^{-1}(M'(S(x))) \oplus x = \alpha\right\}$$

as , if $S^{-1}(M'(S(x))) \oplus x = \alpha$ happens, then we will get difference between X_{R-1}^I and Y_O^{R+2} as α.

2.2 PRINCEv2 [BEK+20]

To give more security to the PRINCE cipher, PRINCE developers came up with a new design of PRINCE. This Section describes the cipher, PRINCEv2. We discuss some differences between PRINCE and PRINCEv2 also. The security analysis of the new version and the discussion about which attacks on PRINCE can be applied on PRNCEv2 is given in the next subsection.

2.2.1 Description of PRINCEv2

PRINCEv2 [BEK+20] also belongs to the same family of block ciphers as PRINCE. In order to achieve higher security level, PRINCEv2 was proposed by making

some small tweaks to the PRINCE block cipher, without changing the number of rounds or round operations. Basically, the tweaks were made on one of the sensitive parts of block cipher design, namely the design of the key-scheduling, as shown in Fig. 2.7. The basic structural differences between PRINCE and PRINCEv2 are highlighted as below:

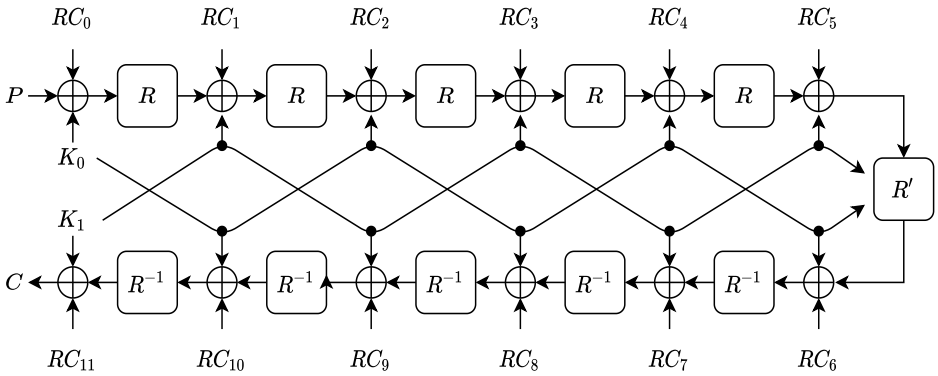


Figure 2.7: Schematic diagram of PRINCEv2

1. Unlike PRINCE, PRINCEv2 does not follow the FX-construction.

2. The 128-bit master key K is split into two parts, $K = (K_0 \| K_1)$ and the i^{th} round key is defined as:

$$K_i = \begin{cases} K_0, & \text{if } i \bmod 2 = 0 \\ K_1, & \text{if } i \bmod 2 \neq 0 \end{cases}$$

3. The middle layer of PRINCEv2 is now a keyed operation, which is shown as below:

$$R' = \mathrm{SB}^{-1} \circ \oplus_{RC_{11}} \circ \oplus_{K_1} \circ \mathrm{MC} \circ \oplus_{K_0} \circ \mathrm{SB}$$

4. The round constant derivation part is almost same like PRINCE, but instead of XORing the same α, we alternate the constants α and β to derive the round constants in the second half. where

$$\begin{cases} \alpha = c0ac29b7c97c50dd \\ \beta = 3f84d5b5b5470917 \end{cases}$$

The round constants are given in Table:2.2.

Table 2.2: Round Constants (RC) of PRINCEv2

RC_0	0000000000000000
RC_1	13198a2e03707344
RC_2	a4093822299f31d0
RC_3	082efa98ec4e6c89
RC_4	452821e638d01377
RC_5	be5466cf34e90c6c
RC_6	7ef84f78fd955cb1
RC_7	7aacf4538d971a60
RC_8	c882d32f25323c54
RC_9	9b8ded979cd838c7
RC_{10}	d3b5a399ca0c2399
RC_{11}	3f84d5b5b5470917

5. Here α-reflection property is slightly weakened and the decryption is not anymore simply encryption with a modified key. Instead a *Swap* function is used to utilize the same circuit for both encryption and decryption (see Fig. 2.8).

$$\text{Swap}(K_0, K_1, dec) = \begin{cases} K_0, K_1; & \text{if } dec = 0 \\ K_1 \oplus \beta, K_0 \oplus \alpha; & \text{if } dec = 1 \end{cases}$$

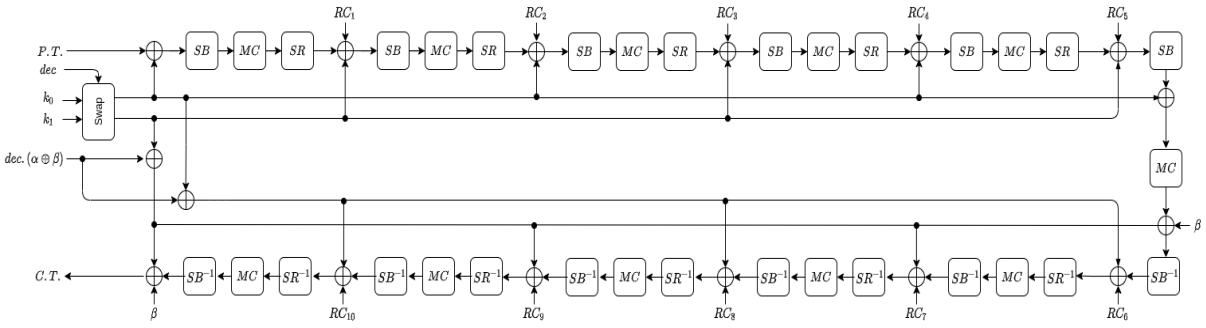


Figure 2.8: PRINCEv2 structure for encryption and decryption

2.2.2 Designers' Perspective on PRINCEv2

The new version of PRINCE, PRINCEv2 is designed to modify some security aspects of PRINCE. PRINCE developers conducted a PRINCE challenge to encourage all the cryptographers so that, they come up with some distinguisher

on PRINCE. This helped the designers to construct a better version of the cipher for the security aspects. In the paper of PRINCEv2 authors gave some security analysis on PRINCE, which will not be applicable or be less effective on this new version.

- **Differential Attack:** One of the most effective differential attacks on PRINCE was introduced by Canteaut *et al.* in [CFG+14]. By this attack, the authors can do a 10 round attack on PRINCE. The data, time, and memory complexities of this attack are $(2^{57.94}, 2^{60.62}, 2^{61.52})$ respectively. Here authors used a 6 round distinguisher. Then prefix 2 rounds at the beginning and postfix 2 rounds at the end. PRINCEv2 is also equally vulnerable to the attack as PRINCE is.
- **Impossible Differential Attack:** One of the impossible differential attacks on PRINCE was proposed by Ding *et al.* in [DZLY17]. Here they used a 4 round distinguisher and enhance it to 7 rounds. The complexities of this attack are $(2^{56}, 2^{53.8}, 2^{43})$. In PRINCEv2, the attack is also applicable with complexities $(\alpha.2^{65}, 2^{128}.e^{-\alpha}, \alpha.2^{65})$, where α is balancing between data or memory, and time complexity.
- **Accelerated Exhaustive Search:** In this attack [JNP+13], authors used a keyless middle round to find out a distinguisher. But designers of PRINCE made the middle round of the new version as keyed. So, the attack will not be applicable in PRINCEv2.
- **Meet-in-the-middle:** The attack described in [CNV13] uses the keyless property of the PRINCE cipher. So, in the new cipher, PRINCEv2 it will not be that much useful.
- **Biclique attack:** The Biclique attack in the paper [ALL12], take an advantage of the FX construction of PRINCE. As the new version of PRINCE is not an FX construction, so this attack will not be applicable.

Though PRINCEv2 designers discussed the security improvement of PRINCE in the new version, they did not claim anything regarding the fault attacks on PRINCE. This motivated us to look into the fault model and apply that in the PRINCE family.

2.3 SIMON [BSS+13]

This section contains the description and some attack techniques related SIMON [BSS+13] cipher. The description of the cipher and an attack is given in the next. The attack uses a dynamic key guessing technique to recover the key.

2.3.1 Description of SIMON

This section contains a description of the lightweight cipher SIMON 2.9. In the first part, we give some notation related to SIMON which will be used throughout the thesis. In the next part, we give the structure of the SIMON family.

Notations

We will use the following notations in our thesis for SIMON:

X^{r-1} : input of the r-th round
L^{r-1} : left half of r-th round input
R^{r-1} : right half of r-th round input
K^{r-1} : subkey used in r-th round
X_i : $i-$ th bit of X from left to right
$X \lll r$: left rotation of X by r bits
$X \ggg r$: right rotation of X by r bits
$\oplus$: bitwise exclusive OR (XOR)
$\cap$: bitwise AND
ΔX : the XOR difference of X and X'
$+$: addition operation
$\%$: modular operation

SIMON is an application of Feistel structure. The SIMON family is denoted by SIMON$2n/l_k$, where $2n$ is the block length and l_k is the key length respectively.

The round function of SIMON has bitwise AND, XOR and rotation. It can be expressed as follows:

$$R_{i+1} = L_i$$
$$L_{i+1} = f(L_i) \oplus R_i \oplus rk_i$$

where,

$$f(x) = ((x \lll a) \wedge (x \lll b)) \oplus (x \lll c)$$

. For SIMON family, the values of a, b, c are $1, 8, 2$ respectively.

2.3.2 Differential Attack on SIMON32 [QHS16]

In the following attack on SIMON, we use an existing 13-round differential distinguisher,

$$D_1 : (0000, 0040) \rightarrow (4000, 0000)$$

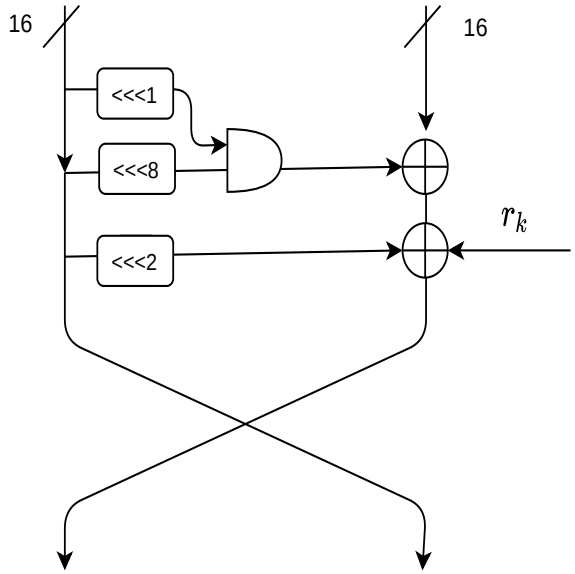


Figure 2.9: Structure of SIMON

, which holds with probability $2^{-28.56}$. We use that distinguisher to mount a key recovery attack on 21 round SIMON32 cipher. We use the distinguisher in the middle of the cipher i.e. for $4 \rightarrow 17$ rounds, then initially prefix some rounds at the beginning and after applying the distinguisher we postfix some rounds at last, like in Fig. 2.10.

Finding Right Plaintext Pairs for the Trail

- Here in the plaintext differences, there are 10 conditions, and in the input of the 2nd round, there are 8 conditions respectively. Here we have 18 fixed differences, so 2^{18} structures are there with 2^{14} many plaintexts. This is shown in Fig. 2.11 Now, in each structure, we have to take those plaintext pairs whose difference in the 3 bits $(X^0_{19}, X^0_1, X^1_{21})$ are 1. So we divide a structure into pair of substructures (A_1, A'_1),(A_2, A'_2),(A_3, A'_3),(A_4, A'_4), where for

A_1 and $A'_1, (X^0_{19}, X^0_1, X^1_{21})$ are 000 and 111 resp.

A_2 and $A'_2, (X^0_{19}, X^0_1, X^1_{21})$ are 001 and 110 resp.

A_3 and $A'_3, (X^0_{19}, X^0_1, X^1_{21})$ are 010 and 101 resp.

Rounds	Input Differences of Each Round
0	$0,*,0,1,*,*,*,*,*,0,0,0,0,*,*,*\quad *,1,*,*,*,*,*,*,0,*,0,*,*,*,*,*$
1	$*,0,\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{1},*,*,0,*,0,\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},*\quad 0,*,0,1,*,*,*,*,*,0,0,0,0,*,*,*$
2	$0,*,0,0,\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{1},*,0,0,0,0,\mathbf{0},\mathbf{0},\mathbf{0}\quad 0,*,0,0,0,0,0,1,*,0,0,0,0,0,0,0$
3	$\mathbf{0},0,0,0,0,0,\mathbf{0},\mathbf{0},0,\mathbf{1},0,0,0,0,0,\mathbf{0}\quad 0,*,0,0,0,0,0,1,*,0,0,0,0,0,0,0$
4	$0,\mathbf{0},0,0,0,0,0,0,\mathbf{0},0,0,0,0,0,0,0\quad 0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0$
$4 \rightarrow 17$	13-round differential D_1
17	$0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0\quad \mathbf{0},0,0,0,0,0,0,0,0,\ \mathbf{0},0,0,0,0,0,0$
18	$*,0,0,0,0,0,0,0,0,*,0,0,0,0,0,1\quad 0,\mathbf{1},0,0,0,0,0,\mathbf{0},\mathbf{0},0,0,0,0,0,\mathbf{0},\mathbf{0}$
19	$0,*,0,0,0,0,0,*,*,0,0,0,0,1,*,*\quad *,0,0,0,0,\mathbf{0},\mathbf{0},\mathbf{0},0,*,0,0,\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{1}$
20	$*,0,0,0,0,*,*,*,0,*,0,1,*,*,*,*\quad 0,*,0,\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},*,*,0,\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{1},*,*$
21	$0,*,0,*,*,*,*,*,*,1,*,*,*,*,*,*\quad *,\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},*,*,*,\mathbf{0},*,\mathbf{0},\mathbf{1},*,*,*,*$

Figure 2.10: Extended Differential Path of 21-round SIMON32

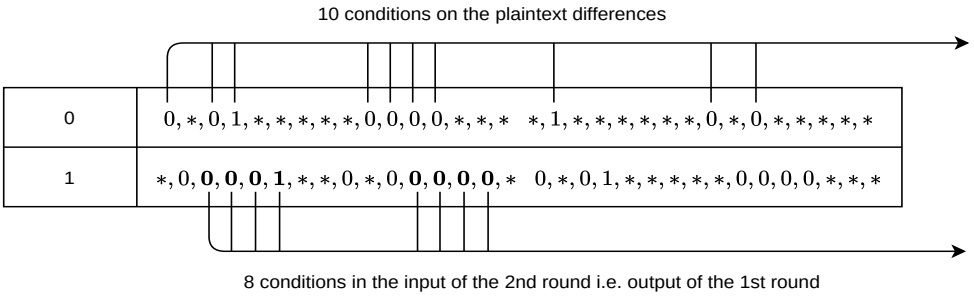


Figure 2.11: Position of 18 Conditions in the Input Difference

$$A_4 \text{ and } A'_4, (X^0_{19}, X^0_1, X^1_{21}) \text{ are 011 and 100 resp.}$$

and all the other 15 positions can be chosen from $\{0,1\}^{15}$. So, for A_1, there are 2^{15} elements, and for A'_1 also. We take those two elements from A and A', whose other 15 positions are equal and make them pair. So, there are $2^{15} \times 4 = 2^{17}$ many reduced structures, shown in Fig. 2.12,

each contains 2^{14} many plaintexts and satisfies the input difference of 0 and 1 round.

- We consider the structures A and A' with 3 different bits (X_{19}, X_{10}, X_{21}) compute the ciphertexts, and save them into a table indexed by $X_t^{21}(t = 1, 2, 3, 4, 8, 10, 16, 18)$ with $\Delta X_t^{21} = 0$. Then for each structure, there are about $2^{14\times 2-8} = 2^{20}$ remaining pairs.

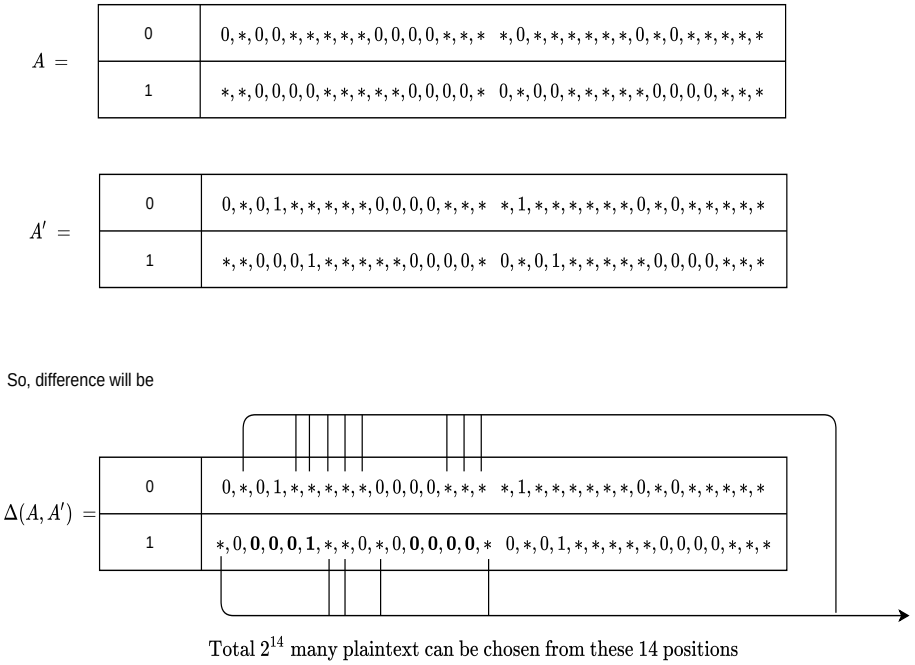


Figure 2.12: Visualisation of How the Structures can be Made with ParticularPositions 1

As there are 2^{14} many plaintexts in A, and 2^{14} many in A'. So, we can make $\binom{2^{14}}{2}$ many pairs from there, which is about $2^{14\times 2}$ many and that should satisfy the 8 conditions of the 21st round. So, we have about $2^{14\times 2-8} = 2^{20}$ remaining pairs for each structure.

- For the attack, here we take 2^{17} structures, decrypt one round for the ciphertexts by applying conditions like in Fig 2.10 and take only those pairs that satisfies the conditions. Then there are $2^{17-1+20-10} = 2^{26}$ pairs are left. The reason is given in the following. Here 2^{17} many structures are there, and we are taking the structures A and A' to make the table. Now for each of these like structures, there are 2^{20} many pairs

and for decrypting 1 round we need extra 10 conditions. Store the pairs in table T.

Here we need about $2^{17+14} = 2^{31}$ (as 2^{17} structures are there and for each structure, there are 2^{14} many plaintexts) encryptions for data collection. We get about $2^{17-1+28-14} = 2^{30}$ pairs which satisfy the input difference of the 13-round differential D_1 for the 2^{31} collected plaintexts. So there are on average $2^{30-28.56} = 2.7$ right pairs.

Key Recovery Technique

- Take a pair from T
- There are 7 conditions in the 2nd round of Fig. 2.10 (i.e. 7 dark bits). We get some equations from there and solving those we can recover the key. Like $(\Delta X^2_{30} = 0, \Delta X^2_{21} = 0, \Delta X^2_{23} = 1)$, for these conditions we get the corresponding equations by partially decrypting the 2nd round

$$\Delta(X^1_{31} \cap X^1_{22}) \oplus \Delta X^1_{16} \oplus \Delta X^0_{30} = 0 \tag{2.1}$$
$$\Delta(X^1_{22} \cap X^1_{29}) \oplus \Delta X^1_{23} \oplus \Delta X^0_{21} = 0 \tag{2.2}$$
$$\Delta(X^1_{31} \cap X^1_{24}) \oplus \Delta X^1_{25} \oplus \Delta X^0_{23} = 1 \tag{2.3}$$

Now,

$$X^1_{31} = (X^0_{16} \cap X^0_{23}) \oplus X^0_{17} \oplus X^0_{15} \oplus K^0_{15} \tag{2.4}$$
$$X^1_{22} = (X^0_{23} \cap X^0_{30}) \oplus X^0_{24} \oplus X^0_{6} \oplus K^0_{6} \tag{2.5}$$
$$X^1_{29} = (X^0_{30} \cap X^0_{21}) \oplus X^0_{31} \oplus X^0_{13} \oplus K^0_{13} \tag{2.6}$$
$$X^1_{24} = (X^0_{25} \cap X^0_{16}) \oplus X^0_{26} \oplus X^0_{8} \oplus K^0_{8} \tag{2.7}$$

From the last 4 equations we can get the solutions about subkey $(K^0_{15}, K^0_6, K^0_{13}, K^0_8)$ which will depend upon values of $(\Delta X^1_{22}, \Delta X^1_{31}, \Delta X^1_{16} \oplus \Delta X^0_{30}, \Delta X^1_{23} \oplus \Delta X^0_{21}, \Delta X^1_{25} \oplus \Delta X^0_{23} \oplus 1)$

  - $(0, 0, 0, 0, 0)$: In this case, there are 16 solutions of the equations 2.1, 2.2, 2.3.
  - $(0, 1, *, 0, *)$: In this case, there are 4 solutions of the equations 2.1, 2.2, 2.3 for each of the 4 cases.
  - $(1, 0, *, *, 0)$: In this case, there are 4 solutions of the equations 2.1, 2.2, 2.3 for each of the 4 cases.

– $(1, 1, *, *, *)$: In this case, there are 2 solutions of the equations 2.1, 2.2, 2.3 for each of the 2 cases.

So here we get 64 solutions for 17 cases i.e. $\frac{64}{17}$ values of subkeys $(K_{15}^0, K_6^0, K_{13}^0, K_8^0)$ on average for each pair in T.

Similarly, we solve the equations corresponding to the subkeys and obtain the guess value. We do this until T is empty. By this procedure, we deduce the keyspace and get the unique key.

2.4 Some Other Lightweight Ciphers

In this section, we describe two other lightweight ciphers, ACE and SIMECK. Interestingly, ACE permutation uses SIMECK for each ACE step.

2.4.1 SIMECK [YZS+15]

The structure of SIMECK is much similar with the structure of SIMON. Like SIMON, SIMECK [YZS+15] also applies Feistel structure. It is denoted by SIMECK$2n/mn$, where the block length is $2n$ and master key length is mn. SIMECK family has three members: SIMECK$32/64$, SIMECK$48/96$ and SIMECK$64/128$, where number of rounds varries with $n_r = 32, 36$ and 44 respectively.

The round function of SIMECK has bitwise AND, XOR and rotation. It can be expressed as follows:

$$R_{i+1} = L_i$$

$$L_{i+1} = f(L_i) \oplus R_i \oplus rk_i$$

where,

$$f(x) = ((x \lll a) \wedge (x \lll b)) \oplus (x \lll c)$$

. For SIMECK family, the values of a, b, c are $5, 0, 1$ respectively.

The master key K for SIMECK$2n(4n)$ has four n-bit words (K_3, K_2, K_1, K_0). Simeck uses a feedback shift register to generate round keys from the given master key K.

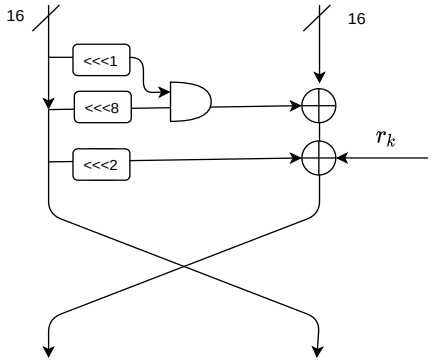


Figure 2.13: Structure of SIMON

2.4.2 ACE [MAR]

ACE is a lightweight cipher which was designed to minimum the cost in hardware without compromising the efficiency. It has a $320-$bit lightweight permutation. It divides the 320-bit permutation in five 64-bit structure, namely A, B, C, D, E. In the middle of ACE-step, SIMECK is applied on A, C and E, like in Fig. 2.14. For this, it is called ACE. There are 16 ACE steps in each ACE permutation. ACE gives Authenticated Encryption with Associated Data (AEAD) and hashing functionalities both.

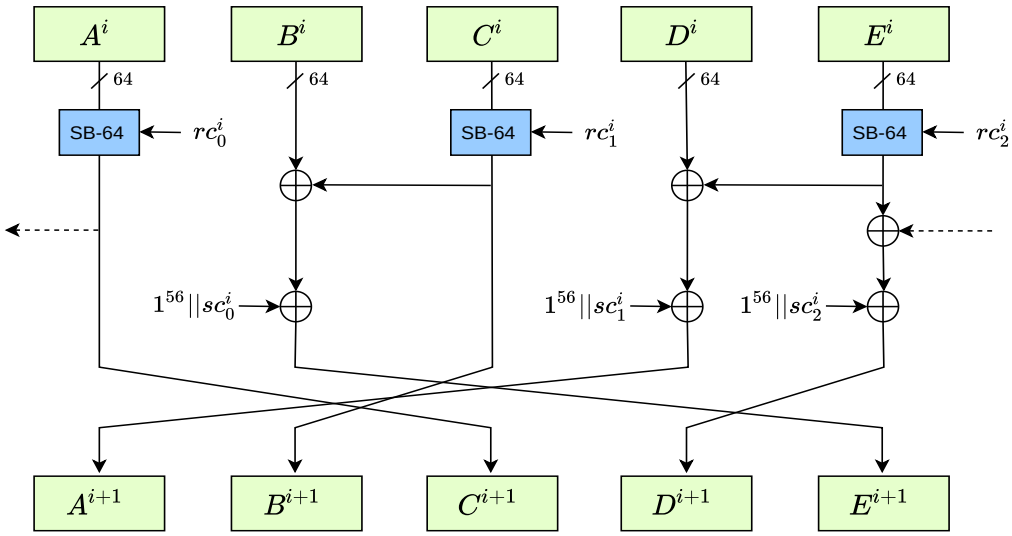


Figure 2.14: ACE-step

- **AEAD algorithm:** ACE AEAD algorithm consists of two things, an authenticated encryption algorithm and a decryption algorithm.

The encryption algorithm takes a key of length k, a message number of length n, associated data AD and a message m. It outputs a corresponding ciphertext C and a tag T.

$$\text{ACE-}\mathcal{E} : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^t$$

with

$$\text{ACE-}\mathcal{E}(K, N, AD, M) = (C, T)$$

The decryption algorithm takes the key K, message number N, associated data AD, ciphertext C and tag T and outputs the plaintext M only is the tag is correct else return $\perp$.

$$\text{ACE-}\mathcal{D}(K, N, AD, C, T) \in \{M, \perp\}$$

- ACE **hash algorithm:** This algorithm takes the message M as input and initial vector IV and outputs a message digest of length h.

$$\text{ACE-}\mathcal{H}\text{-}h : \{0,1\}^* \times \{0,1\}^{iv} \rightarrow \{0,1\}^h$$

with $H = \text{ACE-}\mathcal{H}\text{-}h(M, IV)$.

2.5 Fault Analysis

For any Cryptosystem, to attack the whole cipher directly is a big problem. When a cipher is introduced then it is designed by keeping its security in mind. But there are some other ways to do so. There are some distinguishers which are applicable for some round reduced version of a cipher. We can use those to break the full cipher. We can do so by exploitng the hardware circuit. There are some attack models which helps an attacker to get the information about the cipher.

- **Transient faults:** Suppose a person A sends some messages continuously to some other person B through public channel. Due to some hardware related issues, some of the messages that B gets, become faulty. Then by using the messages B can get some information about the cipher that A uses.

- **Induced faults:** This happens when the cipher is physically presents to the attacker. Then he can do anything with the hardware. The attacker can change the voltage, give high energy radiation, fluctuate the temparature to induce some fault. Then by seeing the output of the ciphertext, he can get some information about the cipher.

2.5.1 Some Papers Related to Fault Attack

In this subsection, we discuss some initial papers related to fault attacks. Boneh *et al.* introduced the fault attack at first. After that Biham and Shamir introduced the Differential fault Attack (DFA). These papers and some others described briefly in the following:

- **Checking Cryptographic Protocols for Faults:** In this paper [BDL97], the authors used hardware faults to break some ciphers. Their work in this paper was based on some authentication protocols and some public key-related ciphers. Their attack was useful against RSA schemes and some signature schemes. Using the Chinese remainder theorem they showed that if a faulty version of RSA signature was given then with high probability an attacker can get the RSA modulus factor. They discuss how some ciphers can be vulnerable using this fault technique. In the paper, they tell the impact of faults so that security measurements can be taken for the ciphers.

- **Differential Fault Attack Technique against SPN Structures:** In this paper [BS97], authors introduce the concept of Differential Fault Analysis (DFA), which can be applied in almost any secret key encryption scheme. In this type of attack, one has to inject a fault in some internal state of the cipher. Suppose initially he gives a message m. The ciphertext of the given message is c. Now, because of the fault injection, the message modifies to m' and the ciphertext that is generated because of the fault is called faulty ciphertext and suppose that is denoted by c'. Then the person observes how the ciphertext changes because of the fault. From there he tries to recover the key. The main idea of this type of attack is, by giving a fault, we inject a difference in the internal state. Then we check how the difference propagates through the cipher. As we can give fault at any intermediate round, we can observe the propagation of difference at the output of the cipher. Along with this in the paper, they show if a cipher has permanent hardware faults, how it can be used to break the cipher.

- **DFA of Secret Key Cryptosystems:** Here in the paper [PQ03], the authors illustrate a technique which can be used to break Substitution-Permutation Networks using very less number of faults. In this paper, they show that, they can break AES [DR98] using only 2 faults with some assumption that, the fault will occur at some particular point. They have used diffusion property to do their attack. Their model is realistic as they use random byte faults.

  The attacks that are stated briefly in the upper section have their historical significance in the fault related area. There are many other attacks in this area till now and some of them are on `PRINCE` also. We discuss about those atttacks in Section 3.1 of Chapter 3.

2.5.2 Different Fault Models

The fault model consists of two things the behavior of the fault and its impact on the system. The fault model can be characterized based on three things: impact, distribution and, location. Fig. 2.15 shows different types of fault models. At bit-level fault, we give a fault in a bit and check the impact of the fault in the system. Similarly, the faults can be done at a byte level, nibble level. The distribution of the fault depends upon whether the fault is uniform or random and its impact can be viewed in bit level, or byte level, and so on.

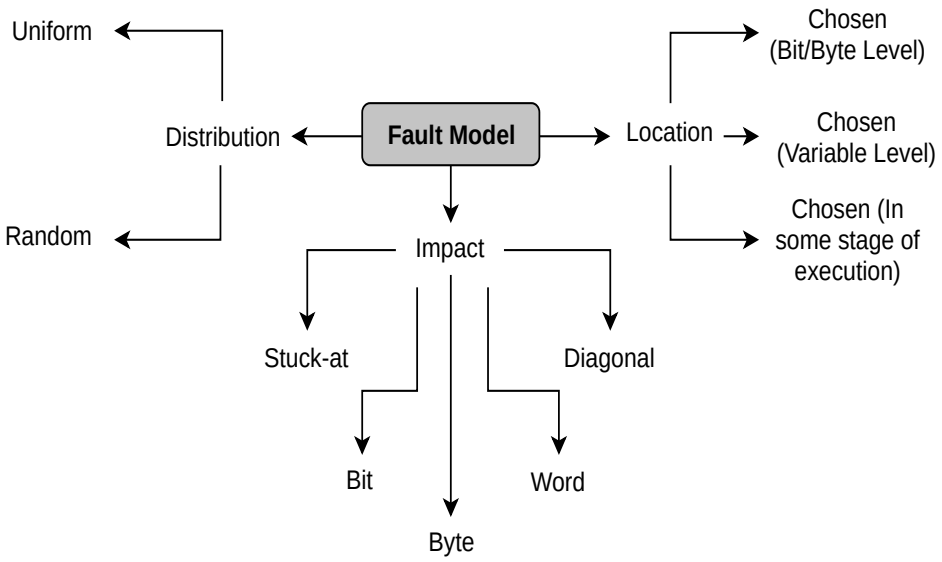


Figure 2.15: Fault Model [Sah17]

- **Nibble-Based Fault Model:** This model is same like Differential fault ones. But in this case we give the faults in some nibbles. That is, we induce some difference at the nibbles. From there we check how the difference diffuses to the entire state. Random nibble-based fault is also one type of nibble-based fault attack. In this case, initially we do not have to know the nibbles, where we induce the faults.
- **Bit-Based Fault Model:** The Bit-based fault attack is done in bit-level. Here we flip a bit and check the diffusion. Our attack, described in the Section 3.2.4 of Chapter 3 is based upon random bit fault attack.
- **Random Fault Model:** In this type of attack, one can inject the fault with random value at a random position depending upon the structure. This type of attack can be random bit-based, random nibble-based, random byte-based, etc. In random bit-based, the attacker gives fault at any bit and follows the pattern. In the case of a random nibble-based fault model, the attacker can give fault at any nibble and so on.

2.6 Conclusion

We discuss mainly `PRINCE`, `PRINCEv2` and `SIMON` in this survey. We illustrate the attacks regarding `PRINCE` and discuss briefly the security analysis of `PRINCEv2`. We give fault attack models and some works related to that. Along with these things, we describe dynamic key guessing and key recovery technique on `SIMON`. Then some other lightweight ciphers `SIMECK` and `ACE`.

CHAPTER 3

Fault Analysis of PRINCE family

Fault attack is a widely used technique nowadays to break a cryptosystem. We can inject the faults at nibble or bit or byte position depending upon the model. Our work here is based upon a random bit flip fault attack on the PRINCE family. We can do the random bit fault at any arbitrary position in our attack. For this, we describe some fault models, diffusion of faults, classification of faults on the cipher PRINCE. Then we illustrate some of the attacks for which PRINCEv2 is equally vulnerable as PRINCE. We give some points on PRINCE regarding fault injection and reflection point

3.1 Related Work

There are two papers of fault attacks on PRINCE till the date. The first one is by Song and Hu [SH13], and the second one is by Aikata *et al.* [AKS20]. In the first one, Song and Hu proposed a random nibble-based fault attack and, then the second paper describes integral fault attack and slow diffusion attack. We discuss those attacks in the following section.

3.1.1 Random Nibble Based Fault Attack [SH13]

This is the first fault attack technique, that is used for PRINCE cipher by Song and Hu [SH13]. Their attack is random nibble-based. They inject faults during the subbyte of 10^{th} round. Their attack model is depicted in Fig. 3.1. Then they checked how the difference propagates through the cipher. Their attack takes less than 7 fault injections to recover the whole key. Along with this, they check the case for giving the fault during the subbyte of 11^{th} round also.

But for the random bit fault case, i.e. for the faults with hamming weight 1, they have stated that less information can be obtained from there. Here in our work, we have improved their statement about random bit fault and show that 7 faults are sufficient to recover the key uniquely. In our work, we give the fault at the beginning of the round 10, whereas they give during the subbyte operation. This proves that our fault covers more rounds than theirs.

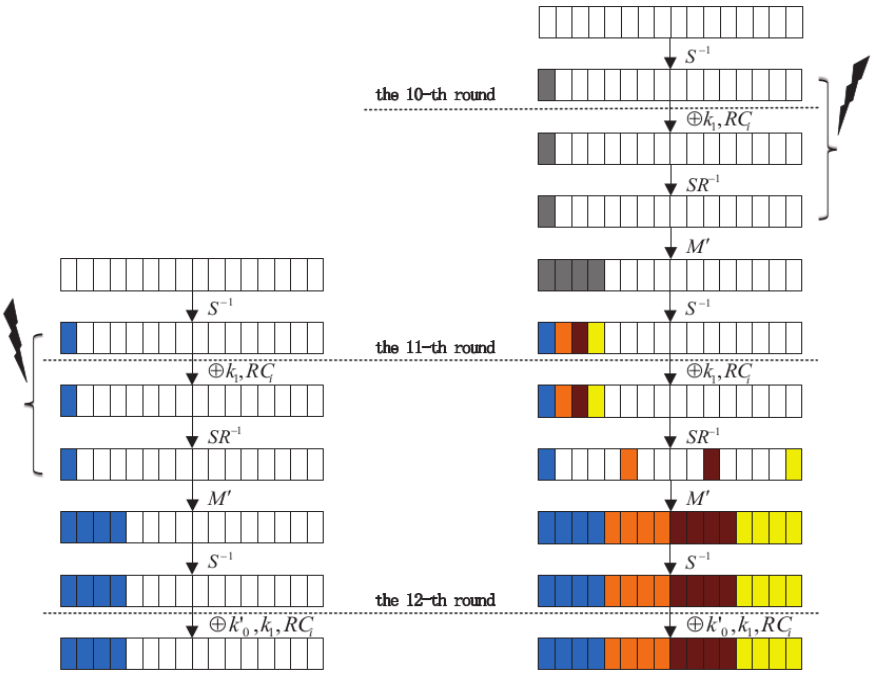


Figure 3.1: Attack of PRINCE at round 10 and 11 by Song and Hu [SH13]

3.1.2 Integral Fault Attack [AKS20]

This is the second fault attack on PRINCE. Their attack covers more round than the attack proposed by Song and Hu, which is described in Section 3.1.1. Authors use integral fault attack to recover the whole key. They give 15 faults after the MC of 9-th round and collect all the ciphertexts. By 15 faults, they give All property (i.e. in one nibble we give all possible 16 differences) in that nibble. As PRINCE SB^{-1} is a nibble-based operation, so All property remains after the operation also. In the next round, the All property spreads through the column. In all columns, it becomes A^8, i.e. there are 8 distinct values in each nibble. After SB^{-1}, the A^8 property remains as it is. Then in the next round, it spread to fixed A^i, where i is fixed for individual nibbles. After the MC of the next round, the Balanced property remains (i.e. XOR-sum of all the

ciphertexts is 0) and after the SB^{-1} operation, the property goes away. Here the authors give the faults at 8.5 rounds and 9.5 rounds, i.e. after the MC operation of 8^{th} and 9^{th} round. For 8.5 round, by guessing the keys, they invert the columns and check whether the pattern like state 10 of Fig. 3.2 appears or not. If so, they consider the guessed value of the key as a possible one, otherwise, they reject that. They have done the same attack one round before i.e. at 8.5 round. In this case, they choose all the possible values of $K_1 \oplus K_0'$ for one column (see Fig. 3.2) and one diagonal for K_1 and check whether the pattern comes after SB^{-1} of 10^{th} round or not. For these two attacks, their time complexity is 2^{20} and 2^{36} respectively.

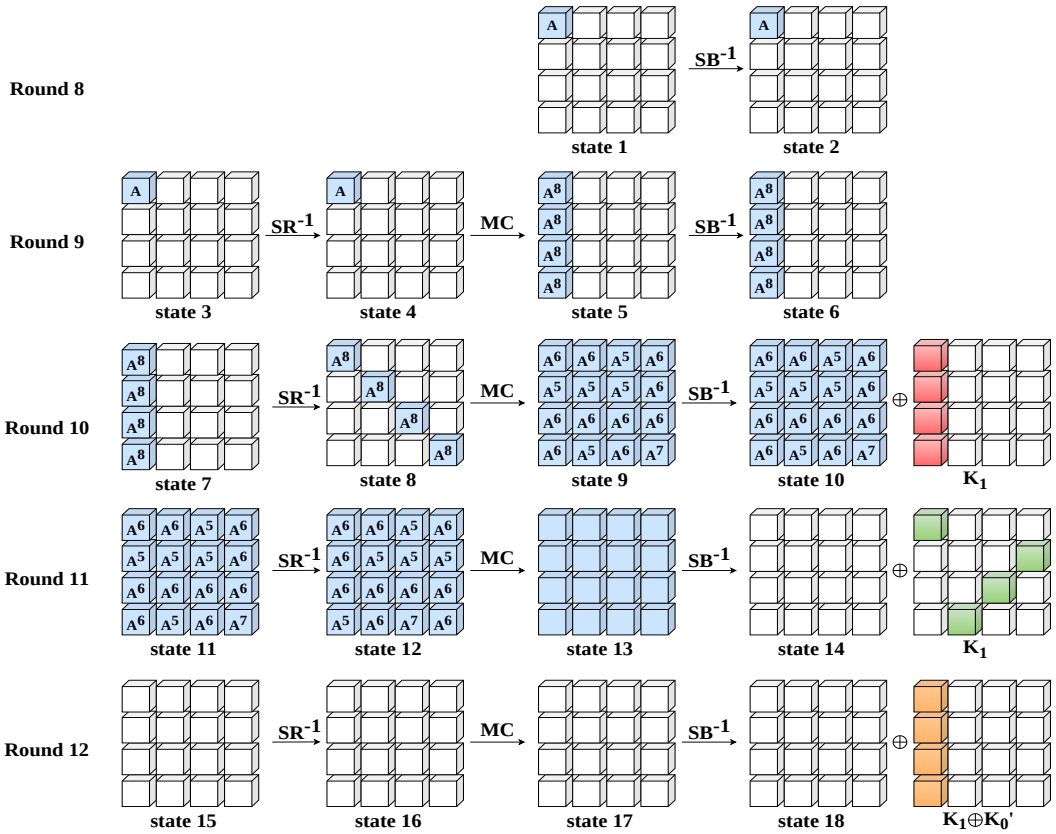


Figure 3.2: Integral Fault Attack on PRINCE [AKS20]

3.1.3 Slow Diffusion Fault Attack [AKS20]

In the paper [AKS20], authors use the slow diffusion property, which is described in the section 2.1.3 of Chapter 2. Here they give the single bit faults at a particular nibble in the input of 10^{th} round. Suppose the fault differences that

are given at nibble 0 are {0001, 0010, 0100, 1000}. Here we illustrate the case when the fault is given at bit 0, as depicted in 3.3. Then from the ciphertexts they retrieve the whole key. At first by all the possible values of keys in the column, they decrypt one round and check whether the pattern at state 5 of Fig. 3.3 occurs or not. Depending upon that they recover the whole key. The (Data, Time, Memory) complexity of their attack is $(2^2, 2^{22}, \text{neg.})$ respectively.

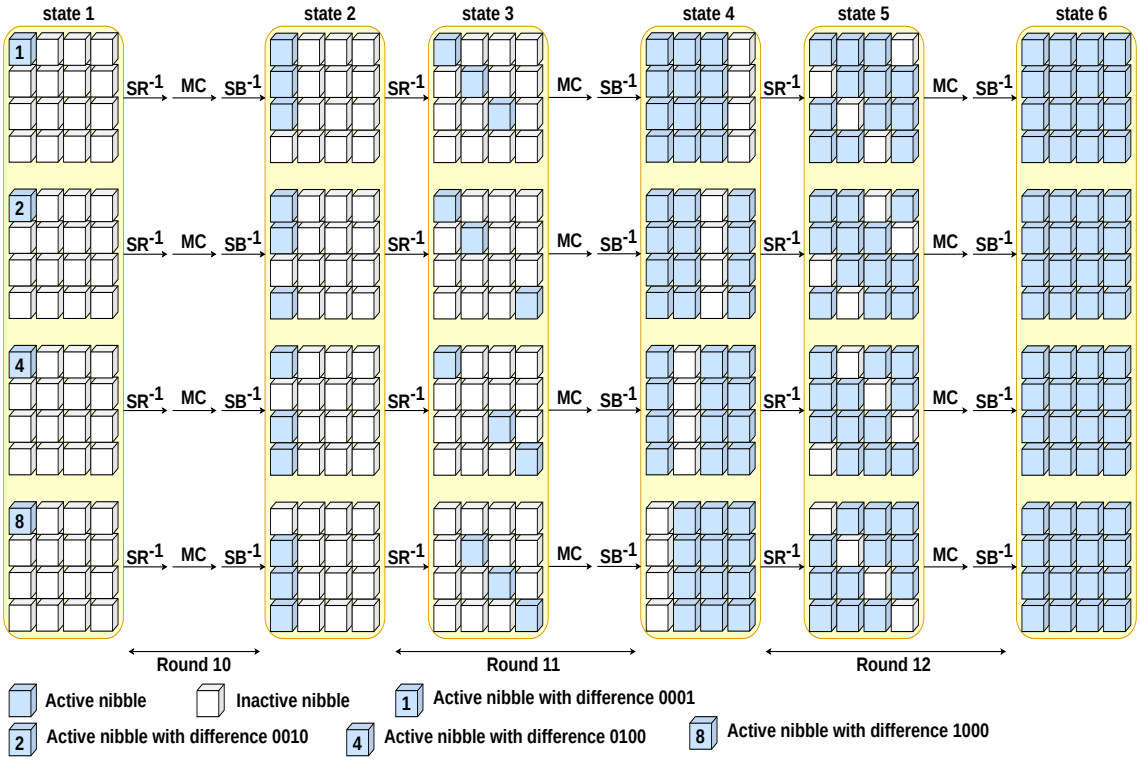


Figure 3.3: Slow Diffusion Property of PRINCE [AKS20]

3.2 Proposed Differential Fault Attack on PRINCE

This is the main contributory section of our current work. Our work is based on a random bit-flip attack. We check our attack for both of the versions of PRINCE. Starting with the bit-based fault attack, we show how a given fault is diffused. Depending upon how fault is propagated, we classify the affected bits.

3.2.1 The Fault Model

A Bit-based fault attack can be viewed as a special case of a nibble-based fault attack. Giving some difference at a bit is the same as giving a particular difference at a particular nibble. There are some differences as giving fault at bit level is a bit easier than giving the fault in a nibble. Here our attack is on a random bit fault model. In this case, we can give the fault in a bit and check the propagation of the difference. The fault can be at any random bit position. From there also we are able to recover the whole key. The attack is described in the following section.

3.2.2 Fault Diffusion in Internal State of PRINCE

As PRINCE uses almost Maximum Distance Separable matrix For MC, it has slow diffusion property. The difference at any state can take at most three rounds to spread all the states. Here we use this property of MC to find a distinguisher of PRINCE. We inject a bit of fault at the input of 10^{th} round and observe the propagation of the difference. Depending upon this, we recover the whole key uniquely by giving some faults. The most interesting part of this attack is that we can give the fault at any single bit position and we don't have to know the position beforehand. In the previous works in this area, Song and Hu [SH13] give the faults during the subbyte operation and they did nibble-based fault in their research. In our work, our fault injection point is before their ones and they discussed random nibble fault and told the bit fault is a special case. But in our work, we discuss bit fault explicitly. In Fig. 3.4, every bit difference from the left shaded region goes to the bit difference in the right side, by 2 forward rounds. Depending upon the structure of each round, PRINCE can be broken into three parts, forward rounds, middle rounds, and backward rounds. In Fig. 3.4, we show how for each part the difference will propagate.

3.2.3 Classification of Fault Invariants

As PRINCE has slow diffusion property, the bit difference takes 3 rounds to spread in the whole state. Depending upon the similarity, the rounds of PRINCE can be classified into 3 classes, forward, middle and backward. Forward rounds contain subbyte, MC and shiftrow. Middle round contains Subbyte, MC and inverse subbyte. The backward round contains inverse subbyte, MC, inverse shiftrow. Because of these three classes if we give fault at some position, then it will affect the bits depending upon in which class it belongs. If we inject the fault at 0-th position in forward round, then it will not effect {0, 5, 9, 12, 16, 20, 27, 31, 35, 38, 42, 47, 49, 53, 58, 62} bits by 2 rounds. The same bits will

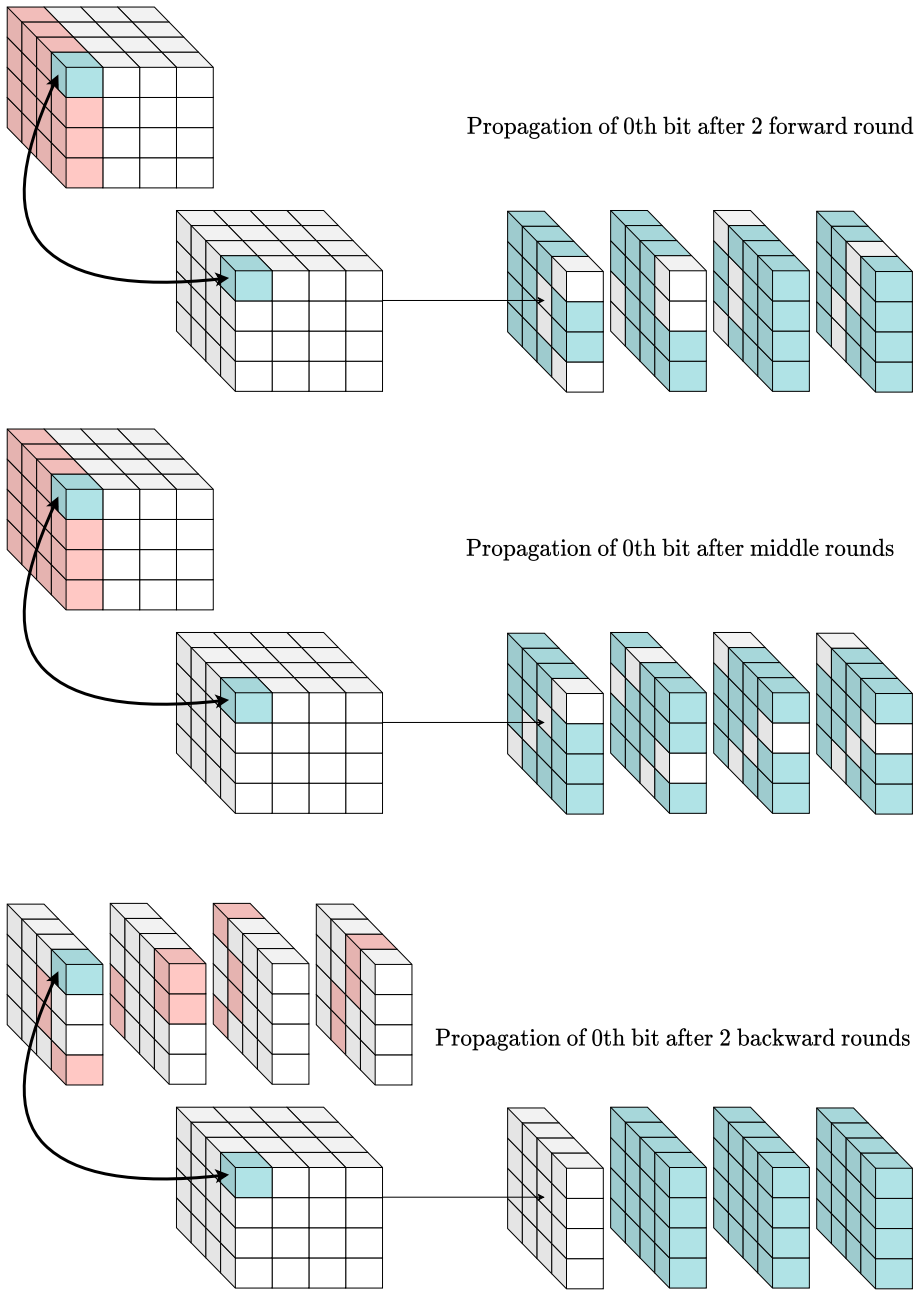


Figure 3.4: Bit Difference Propagation in PRINCE. Any bit belonging to the red region would leads to the same output difference as shown in the image

not be affected if we give the fault at 1-th bit in the forward round. In the same way, we can make a collection of bits, which is $0 - 15$ bit position, for which the bit differences of ciphertext and faulty ciphertext at those places will be 0.

Similarly, we can get a collection of bits, for which if we give the fault at those positions then the same bit position will be constant. The full classification is shown in the table 3.1.

3.2.4 Description of the Attack

For this attack, we inject the fault at the input of 10^{th} round. As it is a bit fault so after 2 rounds also there will remain some bits whose difference will be 0. That difference occurs at the output of 11^{th} round. If a bit is flipped at any position, then after two rounds difference of individual four nibbles will be zero. Depending upon the position of the flipped bit, we will get four classes. This is depicted in Fig. 3.5. Then we decrypt one round after guessing the keys and check whether the pattern appears or not. Depending upon that we reduce the keyspace. We give 7 faults at random bits in the whole state and save the ciphertext and faulty ciphertext pair. As we consider random bit flip i.e. the bit can be flipped at any position from these four classes, so we take the union of the patterns from these classes. If for a chosen value of the key, any of those patterns comes, we keep the value. Else reject.

Algorithm 2: RANDOM_BIT_FAULT_ATTACK_3 $(C_1, \cdots, C_7, C'_1, \cdots, C'_7)$

1 $K'_0\_xor\_K_1 \leftarrow$ RECOVER_K'_0_XOR_K_1_RBFA $(C_1, \cdots, C_7, C'_1, \cdots, C'_7)$
2 $K_1 \leftarrow$ RECOVER_K_1_RBFA$(C_1, \cdots, C_7, C'_1, \cdots, C'_7, K'_0_xor_K_1)$
3 $K'_0 \leftarrow$ RECOVER_K'_0_RBFA$(K'_0_xor_K_1, K_1)$
4 **return** K'_0 and K_1

The attack is described as follow:

- Initially we guess all the possible values of each column of $K'_0 \oplus K_1$, then go back to state 8 of Fig. 3.6 for each pair of original and faulty ciphertexts.
- We check the inactive nibble position to reduce the keyspace for $K'_0 \oplus K_1$. We accept the guessed value if any single nibble in that column is inactive at state 8. If we take the intersection of all 7 reduced key-spaces for each pair of original and faulty ciphertexts, we observe that the key-space gets reduced to 2^2 only (refer to Alg. 3).

Fault Bit Positions	Constant Bit Positions for each Bit Difference
Forward Rounds	
0 - 15	0, 5, 9, 12, 16, 20, 27, 31, 35, 38, 42, 47, 49, 53, 58, 62
16 - 31	1, 6, 10, 13, 17, 21, 24, 28, 32, 39, 43, 44, 50, 54, 59, 63
32 - 47	2, 7, 11, 14, 18, 22, 25, 29, 33, 36, 40, 45, 51, 55, 56, 60
48 - 63	3, 4, 8, 15, 19, 23, 26, 30, 34, 37, 41, 46, 48, 52, 57, 61
Middle Rounds	
0 - 15	0, 5, 10, 15, 18, 23, 24, 29, 35, 36, 41, 46, 51, 52, 57, 62
16 - 31	3, 4, 9, 14, 17, 22, 27, 28, 34, 39, 40, 45, 50, 55, 56, 61
32 - 47	2, 7, 8, 13, 16, 21, 26, 31, 33, 38, 43, 44, 49, 54, 59, 60
48 - 63	1, 6, 11, 12, 19, 20, 25, 30, 32, 37, 42, 47, 48, 53, 58, 63
Backward Rounds	
0, 5, 9, 12, 16, 20, 27, 31, 35, 38, 42, 47, 49, 53, 58, 62	0 - 15
1, 6, 10, 13, 17, 21, 24, 28, 32, 39, 43, 44, 50, 54, 59, 63	16 - 31
2, 7, 11, 14, 18, 22, 25, 29, 33, 36, 40, 45, 51, 55, 56, 60	32 - 47
3, 4, 8, 15, 19, 23, 26, 30, 34, 37, 41, 46, 48, 52, 57, 61	48 - 63

Table 3.1: Bit Difference Propagation for 2 Rounds of PRINCE

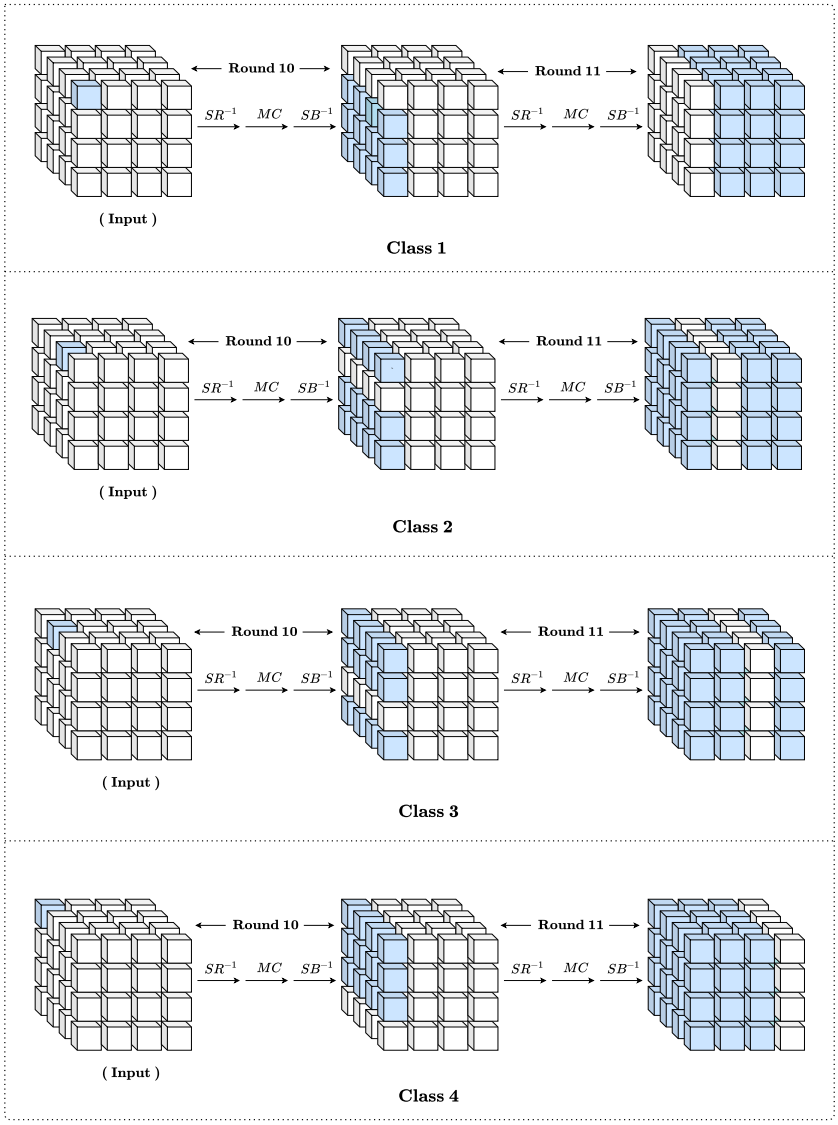


Figure 3.5: Four classes of fault-invariants at the output of Round-11 due to a random bit-fault induced at the beginning of Round-10. Each class contains 16 bit-positions (Refer Table 3.1) that all lead the same invariant at the shown in the state in the right.

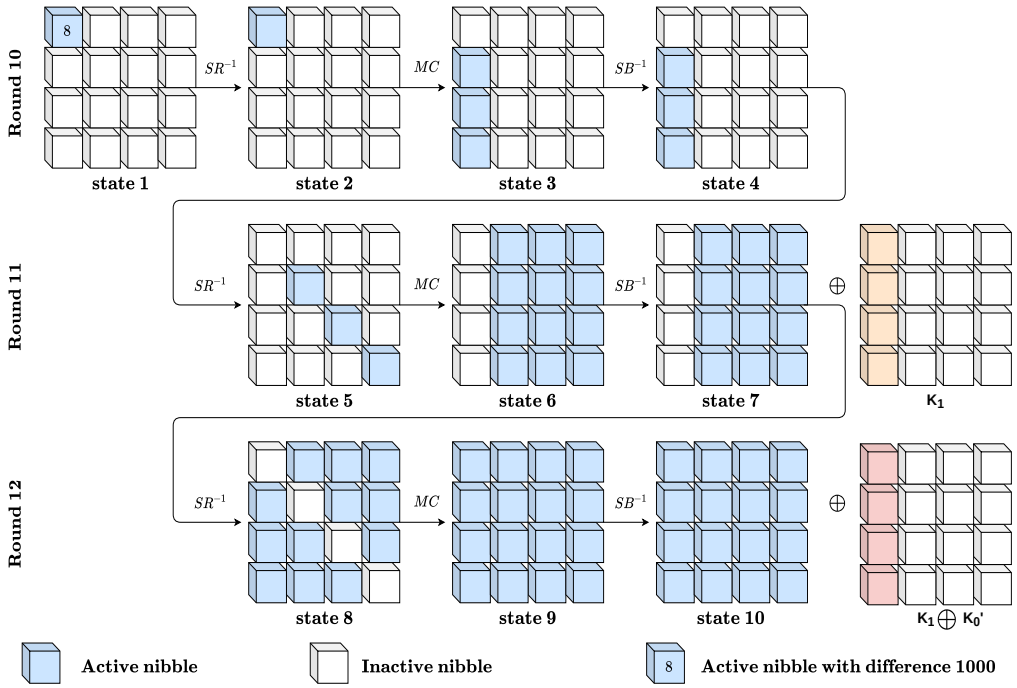


Figure 3.6: 3 round random-bit fault attack on PRINCE

- We use this reduced key-space of $K_0' \oplus K_1$ to go back one more round by guessing the possible values of each column of K_1 and see if those satisfy the pattern of state 5 of Fig. 3.6.
- We accept the guessed value if at least 3 nibbles of that column are inactive at state 5. We again take the intersection of different K_1 guesses for all 7 pairs of original and faulty ciphertexts (refer to Alg. 4). In this step, we get the unique value of K_1 and $K_0' \oplus K_1$.
- We compute the unique value of K_0' from the values of K_1 and $K_0' \oplus K_1$ (refer to Alg. 5).
- We can check whether our retrieved key value is right by going back one more round (till state 2 of Fig. 3.6) easily. As only one bit is flipped, we can find out the right value from there.

Here the data complexity is 2^3 as initially we give 7 random bit faults and store the plaintext ciphertext pairs. We use these only to recover the whole key. The time complexity is 2^{21}. After 7 faults the size of the keyspace of $K_0' \oplus K_1$ is reduced to around 2^2. Then for each value of $K_0' \oplus K_1$, we use the same procedure to reduce the keyspace of K_1 (refer to line 8 of Alg. 4). This takes

Algorithm 3: RECOVER_K'_0_XOR_K_1_RBFA
$(C_1, \cdots, C_7, C'_1, \cdots, C'_7)$

1 $K'_0\_xor\_K_1 \leftarrow \{\{\}, \{\}, \{\}, \{\}\}$
2 $Key_{i,j} \leftarrow \{\}\ \forall\ i \in \{1, \cdots, 7\},\ j \in \{1, 2, 3, 4\}$
3 **for** *each column* j **do**
4 **for** k *in range* $\{0$ - $(2^{16} - 1)\}$ **do**
5 **for** *each* C_i-C'_i *pair* **do**
6 $col \leftarrow MC(SB(C[j] \oplus k \oplus RC_{11}[j])) \oplus MC(SB(C'_i[j] \oplus k \oplus RC_{11}[j]))$
7 **if** *If any of the 4 nibbles of col is 0 (inactive) (state 8 of Fig. 3.6)* **then**
8 append k to $Key_{i,j}$
9 **end**
10 **endfor**
11 **endfor**
12 $K'_0\_xor\_K_1[j] \leftarrow \bigcap_{i \in (1, \cdots, 7)} Key_{i,j}$
13 **endfor**
14 **return** $K'_0\_xor\_K_1$

an effective time complexity of 2^{18}. As we have 7 original and faulty ciphertext pairs, we continue this process until no pairs are left (refer to line 4 of Alg. 4). So, the time complexity becomes approximately 2^{21}.

As the size of the reduced key-space for $K'_0 \oplus K_1$ is around 2^2. For each of the keys in this reduced key-space, we invert the ciphertext up to the output of Round 11. Next, for each of 2^{16} possible values (refer to line 8 of Alg. 4) of each column of K_1, we go back one more round to verify the pattern of state 5 of Fig. 3.6. This translates to effective time complexity of 2^{18}. As we have to repeat this process for the 7 original and faulty ciphertext pairs (refer to line 4 of Alg. 4), the time complexity stands at approximately 2^{21}. The memory complexity is almost negligible since most of the operations can be done on-the-fly.

3.3 Fault Attack Vulnerability Assessment of PRINCEv2

The cipher PRINCEv2 was introduced to overcome some of the drawbacks that PRINCE has. But some attacks are still applicable for PRINCEv2. Our attack is

Algorithm 4: RECOVER_K_1_RBFA
$(C_1, \cdots, C_7, C'_1, \cdots, C'_7, K'_0\_xor\_K_1)$

$K_1 \leftarrow \{\{\}, \{\}, \{\}, \{\}\}$
$Key_{i,j} \leftarrow \{\}\ \forall\ i \in \{1, \cdots, 7\},\ j \in \{1, 2, 3, 4\}$
for p *in* $K'_0\_xor\_K_1$ **do**
  for *each* C_i-C'_i *pair* **do**
    $S \leftarrow SR(MC(SB(C_i \oplus p \oplus RC_{11})))$
    $S' \leftarrow SR(MC(SB(C'_i \oplus p \oplus RC_{11})))$
    for *each column* j **do**
      for k *in range* $\{0$ - $(2^{16} - 1)\}$ **do**
        $col \leftarrow MC(SB(S[j] \oplus k \oplus RC_{10}[j])) \oplus MC(SB(S'[j] \oplus k \oplus RC_{10}[j]))$
        if *all 4 nibbles of col are 0 (inactive) or any 3 nibbles of col are 0 (state 5 of Fig. 3.6)* **then**
          append k to $Key_{i,j}$
        end
      endfor
    endfor
    if $\exists j$ *such that* $Key_{i,j}$ *is empty* **then**
      remove p from $K'_0\_xor\_K_1$ and remove all the $Key_{i,j}$ we got for this p
    end
  endfor
endfor
for *each column* j **do**
  $K_1[j] \leftarrow \bigcap_{i \in (1, \cdots, 7)} Key_{i,j}$
endfor
return K_1

applicable for the new version of PRINCE. For the updated version of PRINCE, the authors originally changed the position of XOR-ing of the keys. PRINCEv2 is equally vulnerable for integral attacks and differential attacks like PRINCE. We give some attacks for which the second version of PRINCE is also equally vulnerable in table 3.2. We verify all the attacks for PRINCEv2 by C language.

- **Random Bit fault Attack:** The random bit fault attack described in Section 3.2.4 is also applicable for PRINCEv2. In the attack, as we induce a single bit difference and the difference spreads through the cipher in the whole state after some rounds and there is no involvement of keys in the

Algorithm 5: RECOVER_K'_0_RBFA(K'_0_xor_K_1, K_1)

```
K'_0 ← {{}, {}, {}, {}}
for each column j do
    K'_0[j] ← K'_0_xor_K_1[j] ⊕ K_1[j]
endfor
return K'_0
```

distinguisher, so, the attack works for the new version of PRINCE also.

- **Random Nibble Fault Attack:** This attack is also a differential fault attack. So, the distinguisher does not depend upon the key. As in PRINCEv2, only XOR-ed position of keys is changed, so it works for PRINCEv2 also.
- **Integral Fault Attack:** Due to the same reason as the previous ones, the integral fault attack also works for PRINCEv2.
- **Slow Diffusion Fault Attack:** This attack also works for the new version of PRINCE and the reason is the same as the previous ones.

Attack type	version 1	version 2
Random Bit Fault	yes	yes
Random Nibble Fault	yes	yes
Integral Fault	yes	yes
Slow Diffusion Fault Attack	yes	yes

Table 3.2: Fault Attacks on PRINCE family

3.4 Experimental Results

We have implemented our work on the random bit fault attack using C but for the outer structure, Python has been used. We ran our implementations on Intel(R) Xeon(R) CPU - E5-2623 v3 @3.00GHz, with 32 GB RAM and it was able to complete within practical time (1 hour). We repeated the experiment over 100 times with random keys and plaintexts and were able to uniquely recover the key every time. We also tried for different fault positions and we were able to recover the key uniquely in those cases as well. Our code is available at https://github.com/de-ci-phe-red-LABS/

PRINCE-under-Differential-Fault-Attack-Now-in-3D. The code works for the new version of PRINCE also.

3.5 Discussion

In some of the fault attacks, a question may arise as that how one can inject a fault at a specific position. But here as we have random bit fault, so in this case, we don't have to worry much about that. We can inject the fault at any position at the unrolled version of PRINCE and PRINCEv2 also.

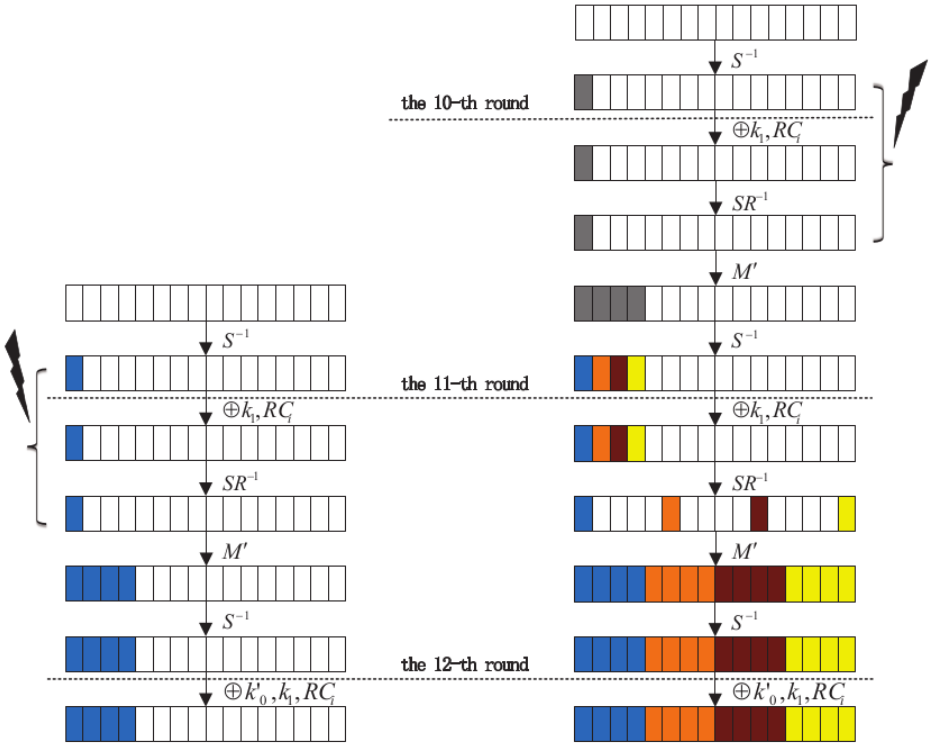


Figure 3.7: Fault-injection (FI) point as given in [SH13] where authors claim that fault is injected in Round-11 (left) and Round-10 (right). It can easily seen that actual FI point is Round-11.5 (left) and Round-10.5 (right).

In the paper [SH13], Song and Hu proposed a fault attack on PRINCE. This was the first attack on this lightweight cipher. Their attack was nibble-based. There they showed, they can get the whole key uniquely by giving 6.61 fault injections. Together with this, using only 4 faulty ciphertexts, they reduced the keyspace to $2^{17.18}$. They stated in their paper that they were able to do fault attacks for 2 and 3 rounds. So, they claimed, if we can

secure the last 2 and 3 rounds from faults, then the cipher will be secured from fault attacks. This seems to be untrue as is evident from Fig. 3.7. It can be easily appreciated from the figure taken from [SH13], faults are injected at the last operation of 10^{th} and 11^{th} round namely, the SB^{-1} operation. Thus the fault-injection point can at best be attributed to 10.5 and 11.5 rounds respectively. Till now, the attack proposed by Aikata *et al.* is the fault attack for the highest number of rounds. Their attack can be done from the last 3.5 rounds and 4.5 rounds also. So, to secure the `PRINCE` family against fault attacks first and last 4.5 needed to be secured. The reason for the security of the first rounds is also required, as given in Section 3.6. Finally, a point worth noting is that `PRINCEv2` offers no additional protection against fault attack which seems to be also stated by the authors. This is apparent from the fact that the changes made are primarily to the key-schedule and round constants which do not meddle with attacks philosophy devised here.

3.6 A Note on Fault Analysis of PRINCE-like Designs

It is worth noting that `PRINCE` offers an interesting opportunity in terms of DFA which is otherwise not possible for conventional block ciphers like AES [DR02], PRESENT [BKL+07]. The opportunity comes from one of the most advertised features of `PRINCE` the $\alpha-$reflection property which ensures that: *"Decryption is equivalent to encryption with a related key"*. This formed one of the major aspects of `PRINCE` design rationale since decryption had to be achievable with minimal overhead over encryption. This seemingly useful property actually opens us new avenues in terms of fault injection and leads to what we like to refer to as *fault-reflection*. The idea stems from the fact that for every fault injection point in any of the attacks reported on `PRINCE` till date (including the current work), there is an equivalent point which too will lead to the the same fault attack preserving the complexities. For instance, in this work we have the slow diffusion attack that works for a fault injected in the input of 10^{th} round. The claim is that the same attack would work even if the fault is injected at 3^{rd} round, as shown in Fig. 3.8.

The only difference is that we inject the fault during decryption. This is also true of other ciphers, however, the difference is that to inject a fault in decryption we need access to the decryption module which may not be available in the real-world scenario. This is where `PRINCE`-like designs make

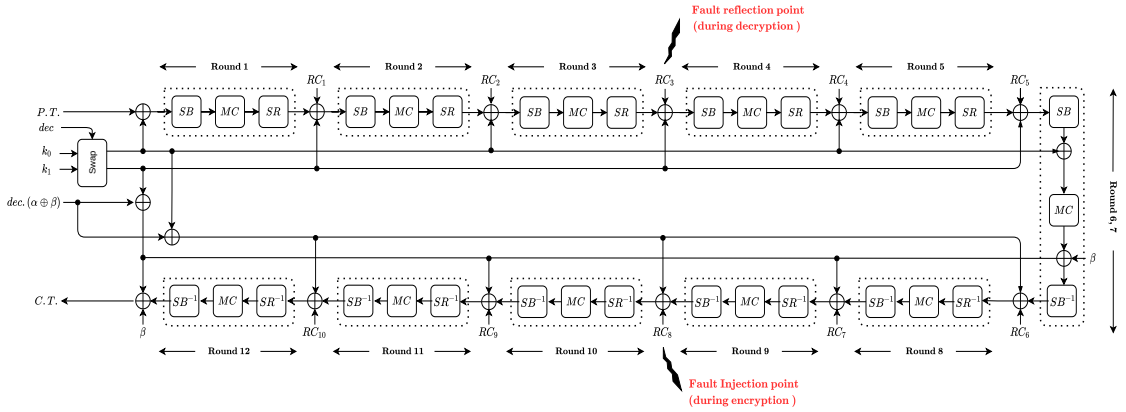


Figure 3.8: fault injection reflection point

an *exception* as they allow us to have access to the decryption oracle on the encryption module itself thereby amplifying the scope of fault injection. The second implication comes in the form of fault-counter measures. The general approach in deploying counter-measures is to protect the lower rounds in the cipher implementation up till the point where a fault-injection can lead to an attack. This actually saves costly hardware real-estate which in general is on the higher side specially when we have an unrolled design like PRINCE. However, for PRINCE-like constructions, this overhead seems to double-up due to the fault-reflection as now one has also protect the upper rounds corresponding to lower rounds as well. In Table 3.3, we showcase the fault-injection matrix taking into account the fault-reflection for all fault attacks reported on PRINCE. Another important aspect that needs to be discussed is that the new version of PRINCE i.e., PRINCEv2 also fits the above definition and hence is come under the purview of the fault-reflection.

3.7 Conclusion

We discussed the related work of Song and Hu on the random nibble-based fault attack and how our attack is different from that. We argue their claim about the fault injection point in the cipher. It is worth mentioning that our attack is based on the random-bit fault model and we can uniquely recover the whole key by injecting only 7 faults at the input of the 10-th round, which makes our attack as one of the best fault attacks on PRINCE and PRINCEv2, known so far, considering both number of faults and the fault injection point together. We also gave an interesting discussion about fault

Attack type	FI point	FR point	Reference
DFA (random nibble)	10.5^a	2.5^a	[SH13]
DFA (random nibble)	11.5^a	1.5^a	[SH13]
IFA	8.5	4.5	Sec. 3.1.2
IFA	9.5	3.5	Sec. 3.1.2
Slow Diffusion FA	10	3	Sec. 3.1.3
DFA (random bit-flip)	10	3	Sec. 3.2.4

Table 3.3: Capturing the idea of fault-reflection (FR). Here FI represents the fault-injection point. Any attack using an FI point during encryption will still work with the fault injected at the FR point during the decryption (which by the $\alpha-$reflection property can be mounted using the encryption circuit itself).

Here a is because the fault is induced at the SB opertion of 10^{th} round

injection and relection point and how `PRINCE`-like designs can be vulnerable using this.

Conclusion and Future Works

Conclusion

This work provides a comprehensive account of fault attacks on PRINCE and PRINCEv2. We did random bit fault in both of the versions of PRINCE. The work started with the literature survey on some lightweight ciphers. This work proposes a random bit-flip fault attack where using only 7 faults we can recover the key uniquely with a practical time complexity of 2^{21}. All attacks reported on PRINCE and PRINCEv2 were verified using simulations. It is worth noting that PRINCEv2 is equally vulnerable to all attacks reported on initial version highlighting the fact that that design changes made do not interfere with the attack methodologies. Finally, we discuss the interesting notion of fault-reflection which sheds lights on the fault-injection vulnerability of PRINCE-like constructions.

Future Works

- We will try to use the random bit fault technique for SIMON and find a better distinguisher than the existing ones.
- The discussion on fault injection-reflection point is an important part of our work, which was not considered previously. We will check, whether it can be exploited to mount a different type of attack in this kind of ciphers.
- We will try to use the bit propagation and classification technique in some other ciphers and then try to build some distinguisher depending upon that.

Dissemination of The Work

Under Review

- **Anup Kumar Kundu**, Aikata, Banashri Karmakar, Dhiman Saha: Fault Analysis of the PRINCE Family of Lightweight Ciphers, Journal of Cryptographic Engineering (JCEN)

Bibliography

[AKS20] Aikata, Banashri Karmakar, and Dhiman Saha. PRINCE under differential fault attack: Now in 3d. In Chip-Hong Chang, Ulrich Rührmair, Stefan Katzenbeisser, and Patrick Schaumont, editors, *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security Workshop, ASHES@CCS 2020, Virtual Event, USA, November 13, 2020*, pages 81–91. ACM, 2020. x, 11, 12, 13, 31, 32, 33, 34

[ALL12] Farzaneh Abed, Eik List, and Stefan Lucks. On the security of the core of PRINCE against biclique and differential cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2012:712, 2012. 8, 9, 10, 19

[BBdS+19] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Qingju Wang, and Alex Biryukov. Schwaemm and esch: lightweight authenticated encryption and hashing using the sparkle permutation family. *NIST round*, 2, 2019. 3

[BCD+19] Zhenzhen Bao, Avik Chakraborti, Nilanjan Datta, Jian Guo, Mridul Nandi, Thomas Peyrin, and Kan Yasuda. Photon-beetle authenticated encryption and hash family. *NIST lightweight competition round*, 1:115, 2019. 3

[BCDM20] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Status update on elephant. *NIST lightweight competition (Sep 2020)*, 2020. 3

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications -

extended abstract. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012. ix, 2, 5, 7, 9, 11, 13, 15

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997. 28

[BEK+20] Dusan Bozilov, Maria Eichlseder, Miroslav Knezevic, Baptiste Lambin, Gregor Leander, Thorben Moos, Ventzislav Nikov, Shahram Rasoolzadeh, Yosuke Todo, and Friedrich Wiemer. Princev2 - more security for (almost) no overhead. *IACR Cryptol. ePrint Arch.*, 2020:1269, 2020. ix, 3, 16, 17

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007. 45

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997. 28

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptol. ePrint Arch.*, 2013:404, 2013. ix, 3, 19, 21, 23

[CFG+14] Anne Canteaut, Thomas Fuhr, Henri Gilbert, María Naya-Plasencia, and Jean-René Reinhard. Multiple differential cryptanalysis of round-reduced PRINCE. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 591–610. Springer, 2014. 9, 10, 19

[CNV13] Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-middle: Improved MITM attacks. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 222–240. Springer, 2013. 9, 10, 19

[DEM+20] CE Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2. 0. 2020. 3

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1. 2. *Submission to the CAESAR Competition*, 2016. 3

[DHP+20] Joan Daemen, Seth Hoffert, Michaël Peeters, G Van Assche, and R Van Keer. Xoodyak, a lightweight cryptographic scheme. 2020. 4

[DP20] Patrick Derbez and Léo Perrin. Meet-in-the-middle attacks and structural analysis of round-reduced PRINCE. *J. Cryptol.*, 33(3):1184–1215, 2020. 10

[DR98] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In Jean-Jacques Quisquater and Bruce Schneier, editors, *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings*, volume 1820 of *Lecture Notes in Computer Science*, pages 277–284. Springer, 1998. 29

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. 2, 45

[DZLY17] Yao-Ling Ding, Jing-Yuan Zhao, Lei-Bo Li, and Hong-Bo Yu. Impossible differential analysis on round-reduced PRINCE. *J. Inf. Sci. Eng.*, 33(4):1041–1053, 2017. 19

[GR16] Lorenzo Grassi and Christian Rechberger. Practical low data-complexity subspace-trail cryptanalysis of round-reduced PRINCE. In Orr Dunkelman and Somitra Kumar Sanadhya, editors, *Progress in Cryptology - INDOCRYPT 2016 - 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings*, volume 10095 of *Lecture Notes in Computer Science*, pages 322–342, 2016. 10, 11

[HJM+19] Martin Hell, Thomas Johansson, Alexander Maximov, FHNW Willi Meier, Switzerland Jonathan Sönnerup, and Hirotaka Yoshida. Grain-128aeadv2-a lightweight aead stream cipher. 2019. 3

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. New results on romulus. In *NIST Lightweight Cryptography Workshop*, 2020. 3

[JNP+13] Jérémy Jean, Ivica Nikolic, Thomas Peyrin, Lei Wang, and Shuang Wu. Security analysis of PRINCE. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 2013. 9, 10, 19

[JNP+15] Jérémy Jean, Ivica Nikolic, Thomas Peyrin, Lei Wang, and Shuang Wu. Security analysis of PRINCE. *IACR Cryptol. ePrint Arch.*, 2015:372, 2015. 10

[LJW13] Leibo Li, Keting Jia, and Xiaoyun Wang. Improved meet-in-the-middle attacks on AES-192 and PRINCE. *IACR Cryptol. ePrint Arch.*, 2013:573, 2013. 9, 10

[MAR] Guang Gong Kalikinkar Mandal Mark Aagaard, Riham AlTawy and Raghvendra Rohit. Ace: An authenticated encryption and hash algorithm. ix, 26

[Mor17] Pawel Morawiecki. Practical attacks on the round-reduced PRINCE. *IET Inf. Secur.*, 11(3):146–151, 2017. 10, 11

[nis] NIST Lightweight Competition. https://csrc.nist.gov/projects/lightweight-cryptography. 3

[PN15] Raluca Posteuca and Gabriel Negara. Integral cryptanalysis of round-reduced prince cipher. *Proceedings of the Romanian Academy - Series A: Mathematics, Physics, Technical Sciences, Information Science*, 16:265–269, 01 2015. 9, 10, 14

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003. 29

[pri14] THE PRINCE CHALLENGE. https://www.emsec.ruhr-uni-bochum.de/research/research_startseite/prince-challenge/, 2014. 3

[QHS16] Kexin Qiao, Lei Hu, and Siwei Sun. Differential analysis on simeck and SIMON with dynamic key-guessing techniques. In Olivier Camp, Steven Furnell, and Paolo Mori, editors, *Information Systems Security and Privacy - Second International Conference, ICISSP 2016, Rome, Italy, February 19-21, 2016, Revised Selected Papers*, volume 691 of *Communications in Computer and Information Science*, pages 64–85. Springer, 2016. ix, 20

[RD19] Behnaz Rezvani and William Diehl. Hardware implementations of nist lightweight cryptographic candidates: A first look. *IACR Cryptol. ePrint Arch.*, 2019:824, 2019. 3

[RR16a] Shahram Rasoolzadeh and Håvard Raddum. Cryptanalysis of 6-round PRINCE using 2 known plaintexts. *IACR Cryptol. ePrint Arch.*, 2016:132, 2016. 10

[RR16b] Shahram Rasoolzadeh and Håvard Raddum. Faster key recovery attack on round-reduced PRINCE. In Andrey Bogdanov, editor, *Lightweight Cryptography for Security and Privacy - 5th International Workshop, LightSec 2016, Aksaray, Turkey, September 21-22, 2016, Revised Selected Papers*, volume 10098

of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2016. 10, 11

[Sah17] Dhiman Saha. *CRYPTANALYSIS OF HASH FUNCTIONS AND AUTHENTICATED ENCRYPTION SCHEMES*. PhD thesis, INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR, 2017. 29

[SBY+13] Hadi Soleimany, Céline Blondeau, Xiaoli Yu, Wenling Wu, Kaisa Nyberg, Huiling Zhang, Lei Zhang, and Yanfeng Wang. Reflection cryptanalysis of prince-like ciphers. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 71–91. Springer, 2013. 15

[SH13] Ling Song and Lei Hu. Differential fault attack on the PRINCE block cipher. In *Lightweight Cryptography for Security and Privacy - Second International Workshop, LightSec 2013, Gebze, Turkey, May 6-7, 2013, Revised Selected Papers*, pages 43–54, 2013. vii, viii, ix, 31, 32, 35, 44, 45, 47

[WH19] Hongjun Wu and Tao Huang. Tinyjambu: A family of lightweight authenticated encryption algorithms. *Submission to the NIST Lightweight Cryptography Competition, available online at https://csrc. nist. gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/TinyJAMBU-spec. pdf*, 2019. 3

[YZS+15] Gangqiang Yang, Bo Zhu, Valentin Suder, Mark D. Aagaard, and Guang Gong. The simeck family of lightweight block ciphers. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 307–329. Springer, 2015. ix, 25