University of Wollongong
Research Online

University of Wollongong Thesis Collection

University of Wollongong Thesis Collections

2012

Algebraic and side-channel analysis of lightweight block ciphers

Shekh Faisal Abdul Latip
University of Wollongong

Recommended Citation

Abdul Latip, Shekh Faisal, Algebraic and side-channel analysis of lightweight block ciphers, Doctor of Philosophy thesis, School of Computer Science and Software Engineering, University of Wollongong, 2012. http://ro.uow.edu.au/theses/3463

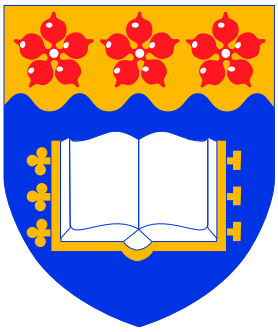
Research Online is the open access institutional repository for the University of Wollongong. For further information contact Manager Repository Services: morgan@uow.edu.au.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.



Algebraic and Side-Channel Analysis of Lightweight Block Ciphers

A thesis submitted in fulfillment of the
requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Shekh Faisal Abdul Latip

School of Computer Science and Software Engineering
Faculty of Informatics
February 2012

© Copyright 2012

by

Shekh Faisal Abdul Latip

All Rights Reserved

Dedicated to

My mother, my wife and my children.

Certification

I, Shekh Faisal Abdul Latip, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science and Software Engineering, Faculty of Informatics, University of Wollongong, is wholly my own work unless referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Shekh Faisal Abdul Latip
February 20, 2012

Abstract

The design and analysis of lightweight block ciphers is gaining increasing popularity due to the general assumption that in the future extensive use will be made of block ciphers in ubiquitous devices. In this PhD thesis we address cryptanalysis of several lightweight block ciphers using algebraic and side channel attacks.

In the first part of the thesis, we investigate the security of the NOEKEON block cipher. We provide the first result of side channel attack on NOEKEON using side channel cube attack.

In the second part of this thesis, we improve the original cube attack by Dinur and Shamir in EUROCRYPT 2009 by introducing an efficient method called *extended cube* for extracting low-degree nonlinear equations. We apply our extended cube method on PRESENT-80 and PRESENT-128. We show that using our extended cube method, we have been able to improve the previous side channel cube attack on PRESENT-80 from CANS 2009. However our attack on PRESENT-128 was the first attack in the side channel model.

Finally, in the final part of this thesis we investigate the security of the three variants of the KATAN block cipher, namely, KATAN32, KATAN48 and KATAN64 against fault attacks.

Acknowledgements

I would like to thank several individuals who have helped me directly or indirectly throughout the achievement of this PhD thesis.

First I would like to thank my principal supervisor Professor Jennifer Seberry for giving her trust and accepting me to be under her supervision. I feel very fortunate and proud for having her as my supervisor for her willingness and patience in guiding and supporting me in various aspects until the completion of this thesis. I also would like to thank my co-supervisor Professor Willy Susilo, who has also shown his willingness, support and interest to supervise me and providing me with access to various resources. I would also like to express my gratitude to my co-supervisor Dr. Mohammad Reza Reyhanitabar who has continuously helped me in evaluating my works and ideas, and sharing his knowledge and experience.

I would also like to thank Associate Professor Yi Mu, Head of School of Computer Science and Software Engineering (SCSSE), Associate Professor Minjie Zhang, Head of Postgraduate Studies (HPS), Professor Philip Ogunbona, Dean of Faculty of Informatics, IT Support staffs, all members of Center for Computer and Information Security Research (CCISR), School of Computer Science and Software Engineering (SCSSE) and Faculty of Informatics who have directly or indirectly helped me in various stages.

Besides, I would also like to thank my friends Dr. Xinyi Huang and Dr. Wei Wu who have helped me in using LATEX for preparing research papers, presentations and my PhD thesis. I am also grateful to Dr. Thomas Plantard for helping me in using MAGMA; Dr. Angela Piper who has helped to improve the readibility of our research paper, to my other friends, Dr. Pairat Thorncharoensri, Bill Tourloupis, Dr. Rungrat Wiangsripanawan, Dr. Siamak Fayyaz Shahandashti, Dr. Man Ho Allen, Tsz Hon Yuen, Dr. Shuhong Wang for making a very nice working atmosphere in our lab.

I would also like to express my gratitude to Dr. Chris Charnes from Macquarie

University for helping me on MiniSAT, Dr. Orr Dunkelman, who visited CCISR in 2009, for giving advice on my work, also to Dr. Abdul Samad bin Hasan Basari from Universiti Teknikal Malaysia Melaka for his invaluable advice and motivating discussions.

Special thanks to Professor Shahrin bin Sahib @ Sahibuddin, Dean of Faculty of Information and Communication Technology of the Universiti Teknikal Malaysia Melaka, where I worked prior to my study leave. He has encouraged me to pursue a PhD in Cryptography. My PhD was fully sponsored by the Universiti Teknikal Malaysia Melaka and the Ministry of Higher Education of Malaysia.

I would also like to extend my gratitude to my thesis examiners, Professor Josef Pieprzyk, director of the Centre for Advanced Computing - Algorithms and Cryptography (ACAC) at the Macquarie University, and Professor Bimal K. Roy of the Indian Statistical Institute, for spending their time reading, evaluating and giving their constructive comments on my thesis.

Last but definitely not least, I would like to thanks my mother Chek Nin binti Md. Ali and my wife Azreen binti Omar for continuously giving moral supports and praying for my success; my son Sheikh Aiman Hadi bin Shekh Faisal, and my two daughters Rose Aliyah binti Shekh Faisal and Alisha Safiyah binti Shekh Faisal for accompanying me in Wollongong and helping to make my life cheerful. My special thanks go to the soul of my father, Abdul Latip bin Abdul Hadi for the motivation and encouragement he gave me during his life.

Thank you.

Shekh Faisal

Publications

The following papers have been published based on the contributions of this thesis.

- Shekh Faisal Abdul-Latip, Mohammad Reza Reyhanitabar, Willy Susilo, and Jennifer Seberry: On the Security of NOEKEON Against Side Channel Cube Attacks. In: Kwak, J. et al. (Eds.) ISPEC 2010. *Lecture Notes in Computer Science*, vol. 6047, pp. 45–55. Springer, Heidelberg (2010)

- Shekh Faisal Abdul-Latip, Mohammad Reza Reyhanitabar, Willy Susilo, and Jennifer Seberry: Extended Cubes: Enhancing the Cube Attack by Extracting Low-Degree Non-Linear Equations. In: Cheung, B. et al. (Eds.) ASIACCS 2011, pp. 296–305. ACM (2011)

- Shekh Faisal Abdul-Latip, Mohammad Reza Reyhanitabar, Willy Susilo, and Jennifer Seberry: Fault Analysis on the KATAN Family of Block Ciphers. In: Ryan, M.D. et al. (Eds.) ISPEC 2012. *Lecture Notes in Computer Science*, vol. 7232, pp. 319-336. Springer, Heidelberg (2012)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Cryptology

Cryptology is the art and science of secret writing. It consists of two main area of studies namely *cryptography*, which concern designing secure cipher systems, and *cryptanalysis* which study how the cipher systems can be broken. It has been generally agreed that the era of modern cryptology has started as early as 1949 when Shannon publish a paper entitled "Communication Theory of Secrecy Systems" [126].

However the major innovations on the field only began in mid-1970s which was initiated by the two important developments. First was the work by Diffie and Hellman [53] in 1976, which proposed some ways to assure the privacy of the data transmitted over insecure channels without the need a separate secure channel for the key exchange. They called this new branch of idea as *public key cryptography*. Second is the work which started a bit earlier by the U.S. National Bureau of Standards (NBS) - now named National Institute of Standards and Technology (NIST) - that called for encryption primitives. The call for primitives had consequently recognized the LUCIFER block cipher [62, 63] designed by IBM in 1971 as a successful candidate in 1974. The LUCIFER block cipher was then turned into the *Data Encryption Standard* (DES) [99] after a year of collaboration between IBM and National Security Agency (NSA).

The transformation of LUCIFER into DES became controversial [130] just after its specification was made public, as the length of the secret key had been reduced from 128 bits as originally used in LUCIFER, to only 56 bits in DES. Furthermore there was also controversy due to its classified design principles of its S-boxes and suspicions about an NSA backdoor. Since the standard had been widely used in governmental and private organizations, and the number of applications that use

DES had also increased, DES consequently came under intense academic scrutiny which motivated the understanding of the design and cryptanalysis of modern cipher systems.

For 15 years exhaustive key search was expected to be the only threat against DES, until the first attack on DES called *differential cryptanalysis* [29] which was faster than exhaustive key search was introduced by Biham and Shamir in 1991. The other attacks on DES which were faster than exhaustive key search were *linear cryptanalysis* [93] introduced by Matsui, and *Davies' attack* [48] initially introduced by Donald Davies and Sean Murphy in 1987, and later improved by Biham and Biryukov [27] in 1994. However all these attacks were only considered theoretical rather than practical and sometimes termed *certificational weaknesses*. Note that, it was the innovations that occurred in the early 1990's that contributed to the foundation of the design and cryptanalysis of today's cipher systems and played an important role in the development of today's new block cipher standard namely the *Advanced Encryption Standard* (AES) [47] which superseded DES.

The search for AES as a new standard to replace DES was conducted by the NIST from 1997 until 2000. The new search received much attention from the cryptographic community since the previous block cipher standard, DES, was believed to be susceptible to brute force attacks due to its relatively small key size, as well as suspicion about the NSA backdoor. Furthermore, DES was designed specifically for hardware and this resulted in slower implementation in software. Although the use of Triple-DES[1] [16] (i.e. applying DES three times) as an immediate alternative to DES can solve the problem of small key size in DES, its implementation is three times slower than DES. In addition, Triple-DES was also not suitable for implementation in an environment with limited resources such as the hardware for which DES was aimed. Thus the effort initiated by the NIST to establish a new block cipher standard to replace DES was essential. At the beginning of the selection process, fifteen different block ciphers were submitted, but only five were selected as the AES finalists in August 1999 which were MARS [39], RC6 [117], Rijndael [47], Serpent [8] and Twofish [124]. The selection of the AES finalists was based on their resistance to cryptanalysis, performance in a variety of settings, feasibility to operate in limited environment and the advantages that they offered. Finally, on October 2, 2000, the Rijndael block cipher designed by Daemen and Rijmen was

[1]Note that double-DES (i.e. applying DES two times) is susceptible to meet-in-the-middle attack.

announced the new block cipher standard, AES, to replace DES.

The development in the field of modern cryptology did not cease with the advent of AES, but was continued by another four main efforts. First was a European research project called NESSIE (New European Schemes for Signatures, Integrity and Encryption) funded from 2000 until 2003 aimed at contributing to the final phase of the AES standardization process and identifying secure cryptographic primitives in several different categories. In this project, forty-two algorithms had been received, but only twelve were successfully selected for the portfolio in February 2003. Of the twelve algorithms, three of them were selected as the portfolio in the block cipher category, two in the category of public-key encryption, three in the category of MAC algorithms and cryptographic hash functions, three in the category of digital signature algorithms and only one in the category of identification schemes. However, none of the six stream cipher candidates was selected for the portfolio.

Second was a project known as CRYPTREC (Cryptography Research and Evaluation Committees) initiated by the Japanese Government to assess and recommend cryptographic techniques for government and industrial use. The project was started in May 2000 and at first glance can be seen as an overlapping and conflicting project with NESSIE. The difference between the two projects can be seen in terms of the selection and recommendations made.

The failure of all the six stream cipher candidates to be selected as the portfolio in the previous NESSIE project and the decline of stream ciphers popularity in real world applications, due to systematic replacement by block ciphers, had resulted in a new effort to identify a portfolio for promising stream ciphers. The new effort was initiated by the EU ECRYPT network through a project known as eSTREAM Stream Cipher project, funded from 2004 until 2008. The project aimed at identifying new stream ciphers that meet at least one of the two profiles namely PROFILE 1 that was dedicated to software with high throughput requirements, and PROFILE 2 that was dedicated to hardware which can operate in limited environment such as limited storage, gate count and power consumption such as smart cards. The call for primitives had resulted in thirty-four stream cipher candidates to the eSTREAM project. Of the thirty-four candidates only eight were successfully selected to be in the portfolio in May 2008, of which four of them were in PROFILE 1 which were HC-128 [134], Rabbit [33], Salsa20/12 [21] and SOSEMANUK [20]; and the remaining four were in PROFILE 2 which were Grain v1 [74], MICKEY v2 [14], Trivium [52] and F-FCSR-H v2 [3]. However four months later, one of the stream ciphers

in PROFILE 2 namely F-FCSR-H v2 was dropped from the portfolio after the first revision due its susceptibility to cryptanalysis [73]. Consequently, the total number of stream ciphers in the PROFILE 2 portfolio became only three as this thesis was being written.

During this time also[2], NIST was organizing the SHA-3 competition to identify new standards for hash functions. The project can be seen as a response to the advances in the cryptanalysis of hash functions especially the collision attack against SHA-1 in [133]. As a result, NIST conducted a workshop to review the implications of this attack between October 31 until November 1, 2005. The attack was believed to affect some of digital signature applications which include time-stamping and certificate signing operations. The highlight of the NIST response to the new attacks was the SHA-3 competition, which had witnessed sixty-four hash function algorithms being received, but only fifty-one of them had fulfilled the minimum submission requirements, and thus were accepted for the first round. However, of the sixty-four entries, only fourteen were selected for the second round before ultimately only five of them were selected as the finalists, which were Blake [13], Grøstl [70], JH [135], Keccak [22] and Skein [64]. The result of the new hash function standard, SHA-3, however was still unknown as this thesis was being written.

1.2 Cryptanalysis of Symmetric Key Primitives

The advances in the field of modern cryptology (as can be seen in Section 1.1) which focus on symmetric key primitives since 1970's were heavily influenced by the efforts put forward by the cryptanalysts to find potential weaknesses of the existing ciphers. When significantly new attacks emerge, new cipher designs that supposedly resist the new attacks (and all other significantly known attacks) are proposed based on new design strategies. Thus, we find that if "carefully designed", a symmetric key primitive can only provide *computationally security* against known attacks but is not guaranteed to be resistance against attacks which are still unknown during the design process. Consequently in practice, the security of many real-world symmetric key primitives rarely depends on the difficulty of finding the secret key through exhaustive key search, instead it usually depends on the effort and time consumed by the cryptanalysts to invent new attacks against them.

[2] i.e. when this thesis was being written

The development in the field of cryptanalysis of symmetric key primitives is very well known in the academic world. However, not much is known about the development in the real world as it is classified or proprietary, especially when considering the development in military, intelligence and national security. While the real world cryptanalysis aims at revealing the exact meaning of an encrypted message (i.e. finding the real attack), the academic cryptanalysis differs slightly in which it uses mathematical techniques to discover at least the certificational weaknesses of the cipher and not necessarily to discover the real attacks against the cipher. More generally, it seeks to obtain more appropriate methods to recover the secret key with the computational complexity faster than exhaustive key search or previously best known attack against the cipher. Hence, most of the time the academic cryptanalysis of a symmetric key primitive begins with a reduced variant (i.e. either reduced round or simplified variant) of the cipher. The reduced variant will evolutionary be upgraded each time a new improved attack against the cipher being discovered, and this process continues probably until the full cipher can be practically broken.

Note that, the evaluation of the security of a symmetric key primitive must follow one of the *standard attack models* such as *ciphertext-only attack*, *known-plaintext attack*, *chosen-plaintext attack*, *adaptive chosen-plaintext attack*, *chosen-ciphertext attack* and *adaptive chosen-ciphertext attack* (cf. Chapter 2 for detailed explanation about these attack models). However there is an attack which is not based on the standard attack models called *related-key attacks* [25]. In theory, this type of attacks is considered unrealistic as the secret keys are supposed to be random. However in the real world the attack might be possible if the secret keys were not random which may due to the weakness of the secret key generation process.

In addition, we have also another attack model called *side-channel attacks* which are also not based on the standard attack models. This model is based on information leaked from physical implementation of cryptographic algorithms rather than theoretical weaknesses of cryptographic algorithms. The information obtained from side channel can be either, timing information, power consumption, electromagnetic leaks, temperature, light, sound etc. The leakage of these types of information may be used by an attacker to examine the internal state of cryptographic devices and subsequently recovering the secret key that reside therein. The most powerful side channel attack was known as *differential power analysis* [85, 86] introduced by Kocher. Besides side channel attacks, there is another type of attack called *fault attacks* [36] which also exploit the implementation of cryptographic algorithms. The

idea behind this attack is to inject errors to the internal state during the execution of cryptographic algorithms with the aim of exploiting the faulty results which enables information about the secret key to be extracted. Since most of the standard attacks are more theoretical rather than practical, and rarely cause a complete break of a cryptographic algorithm, the side-channel attacks are believed to be capable of attacking the cryptographic algorithm with a complexity much lower than standard attacks, which can often lead to a complete breakage.

Due to the importance of cryptanalysis in helping assess the security of a cryptographic algorithm and its practical implementation, and in turn helping in the development of a more secure cryptographic algorithm, this thesis is intended to take similar measures to study the security of some existing cryptographic algorithms through cryptanalysis.

1.3 Thesis Outline and Contributions

In this thesis, we present our research findings on several symmetric key primitives from a cryptanalyst's viewpoint. The general outline and the main contributions of this thesis are summarized, highlighted and presented in different chapters as below:

- In Chapter 1 we provide the introduction and motivation of the research in symmetric key cryptography, and also the outline and the main contributions of this thesis.

- In Chapter 2 we present a theoretical background of our work explaining about boolean functions, symmetric key encryptions, attack models, and algebraic and side-channel cryptanalysis.

- In Chapter 3 we present the side-channel cube attack on the NOEKEON block cipher. First we show how to find the suitable round for side channel attack. Then by applying one-bit leakage side channel attack model which can be obtained after each round, we show that using cube attacks we are able to extract independent linear equations over the secret key variables and hence recover the secret key.

- In Chapter 4 we propose an efficient method to obtain low degree nonlinear equations in cube attacks by using our extended cube method. We combine this method with a side-channel attack based on the Hamming-weight leakage

model and apply it to the PRESENT block cipher. We show that our approach has successfully improved previous side-channel cube attacks on PRESENT-80 and was the first known side-channel attack on PRESENT-128.

- In Chapter 5 we analyze the security of the KATAN family of block ciphers against fault attacks. First, we determine the efficient rounds for applying fault attacks on KATAN32, KATAN48 and KATAN64. Then, we construct a differential characteristic for each faulty and non-faulty ciphertext bit differential. Using the idea of cube and extended cube methods, we construct polynomial equations (i.e. linear and quadratic) representing each ciphertext bit differential. To find the value of the right hand side of each equation, faults are induced to obtain the differential values of the corresponding ciphertext bits. Then the polynomial equations are solved to recover the secret key.

- In Chapter 6 we conclude the thesis.

Chapter 2

Preliminaries

In this chapter we provide a general introduction and background relating to our work. First we introduce a mathematical background focusing on boolean functions. Then we provide an introduction to symmetric-key cryptosystems. Finally we review two different cryptanalysis models namely algebraic and side-channel analyses which are the focus of this thesis.

2.1 Mathematical Background: Boolean Functions

This section focuses on boolean functions, i.e. $\{0,1\}$-valued functions of a finite number of $\{0,1\}$-valued variables. We only provide the information which is related to our work, and for interested readers we refer to [45, 119] for a more detailed and extensive explanation on the subject matter.

Definition 2.1 *A boolean function of n variables is a function $f : \mathcal{B}^n \rightarrow \mathcal{B}$, where $\mathcal{B} = \{0,1\}$ is the boolean domain, n is a positive integer and $\mathcal{B}^n$ is the cartesian product of n elements of set $\mathcal{B}$ with itself.*

The most fundamental way to define a boolean function, f, is through a *truth table* in which each value of all possible points in $\mathcal{B}^n$ gives one value to the function. For a boolean function with n number of arguments (or variables), the number of possible points in the truth table would be 2^n. Thus, the truth table will be too cumbersome if n is large. To avoid this, most of the time boolean functions are defined implicitly rather than explicitly in that they are defined using certain procedures to compute the value of the function for a certain domain of interest. These procedures are normally seen as a *black box oracle* in which the effect of certain input to the function output can be seen but the detailed algorithm to compute such an output can be either unknown or inefficient to give a closed mathematical formula.

2.1.1 Boolean Functions over $\mathbb{GF}(2)$

Boolean functions can be represented over $\mathbb{GF}(2)$. In this field operation, two operators are used in boolean expressions, i.e. $\oplus$ (i.e. exclusive-or or XOR) and $\wedge$. The exclusive-or operator can be defined as a boolean function $\oplus : \mathcal{B}^2 \rightarrow \mathcal{B}$ with the following definition:

$$\oplus(x_1, x_2) = x_1\bar{x_2} \vee \bar{x_1}x_2$$

for $x_1, x_2 \in \mathcal{B}$. Thus considering $\oplus$ as a binary operator it can also be defined as giving the following results

$$0 \oplus 0 = 0; 0 \oplus 1 = 1; 1 \oplus 0 = 1; 1 \oplus 1 = 0;$$

The $\oplus$ operator also satisfies both commutative and distributive laws, such that, $x_1 \oplus x_2 = x_2 \oplus x_1$ and $x_1 \oplus (x_2 \oplus x_3) = (x_1 \oplus x_2) \oplus x_3$, for all $x_1, x_2, x_3 \in \mathcal{B}$.

2.1.2 Normal Forms

Boolean expressions can be represented in different *normal forms*. One of those which are normally used is *Algebraic Normal Form* (ANF) for standardizing and normalizing logical formulas. It is commonly used in the design of *pseudo random number generators* (PRNG) such as *linear feedback shift registers* (LFSRs) and also can be used in *automated theorem proving* (ATP). A boolean expression is considered to be in ANF if the expression consists XORs of one or more conjunctions of function variables x_i (for $1 \leq i \leq n$), and only a constant a_0. Generally, boolean expression in ANF can be represented in the following form:

$$f(x_1, x_2, x_3, \ldots, x_n) = a_0 \oplus a_1x_1 \oplus \ldots \oplus a_nx_n \oplus a_{1,2}x_1x_2 \oplus \ldots \oplus a_{n-1,n}x_{n-1}x_n \oplus \ldots \oplus a_{1,2,\ldots,n}x_1x_2\ldots x_n$$

where $a_0, a_1, \ldots, a_{1,2,\ldots,n} \in \{0, 1\}$. Representing boolean expressions in ANF enables linear functions to be easily identified as a linear function in ANF is a sum of literals. A literal is a boolean variable of the form x or $\bar{x}$.

Another normal form in which a boolean expression can be represented is called *Disjunctive Normal Form* (DNF), i.e. one in which is a disjunctive of conjunctive clauses. More precisely, a boolean expression is in DNF if and only if it is a disjunction of one or more conjunctions of one or more literals. The only operators used

in DNF are negation, conjunction and disjunction. The negation operator can only only applied on literals.

Definition 2.2 (DNF) *A conjunction expression (sometimes also called term, monomial or cube) is the expression of the form* $C = \bigwedge_{i \in A} x_i \bigwedge_{j \in B} \bar{x_j}$*, where* $A \cap B = \phi$*. Hence, a disjunctive normal form (DNF) is a boolean expression of the form* $\bigvee_{k=1}^{m} C_k = \bigvee_{k=1}^{m} (\bigwedge_{i \in A} x_i \bigwedge_{j \in B} \bar{x_j})$*, where each* C_k *is a conjunction of literals.*

The following example shows boolean expressions in DNF:

$$x_1$$

$$x_1 \wedge x_2$$

$$(x_1 \wedge x_2) \vee x_3$$

$$(x_1 \wedge x_2 \wedge \bar{x_3}) \vee (x_4 \wedge \bar{x_5} \wedge \bar{x_6})$$

We can also represent a boolean expression in *Conjunctive Normal Form* (CNF), i.e. one which is the conjunction of clauses. Each clause is a disjunction of literals, and each literal and its complement cannot appear within the same clause. Similarly to DNF, the only operators used in the expression of this normal form are negation, conjunction and disjunction operators. The negation operator can only be applied on literals.

Definition 2.3 (CNF) *The basis of disjunction expression is the expression of the form* $D = \bigvee_{i \in A} x_i \bigvee_{j \in B} \bar{x_j}$*, where* $A \cap B = \phi$*. Hence, a conjunctive normal form (CNF) is a boolean expression of the form* $\bigwedge_{k=1}^{m} C_k = \bigwedge_{k=1}^{m} (\bigvee_{i \in A} x_i \bigvee_{j \in B} \bar{x_j})$*, where each* C_k *is a disjunction of literals.*

The following example shows boolean expression in CNF:

$$x_1 \wedge x_2$$

$$x_1 \vee x_2$$

$$\bar{x_1} \wedge (x_2 \vee x_3)$$

$$(x_1 \vee x_2) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_4}) \wedge (x_4 \vee \bar{x_5})$$

Converting a boolean expression into DNF or CNF requires one to use *logical equivalences* such as *double negative elimination*, *De Morgan's law* and the *distributive law*. Conversion may lead to an exponential explosion to the length of the new expression.

2.2 Symmetric-key Encryption

The main idea of a cipher system is to conceal sensitive information by scrambling the characters therein so that its meaning is unintelligible by an unauthorized user. Nowadays such systems are widely used as back-end applications in email systems, mobile phone, computer systems and networks, ATM machine, etc. The information to be encrypted (or to be concealed) is called a *plaintext* and the encrypted plaintext which cannot be understood by the unauthorized user is called a *ciphertext*. In order to convert a plaintext into a ciphertext, we require a set of rules called an *encryption algorithm* which depends on an *encryption key*. However, to obtain the original message from the ciphertext, there has to be a *decryption algorithm* which, when used with the appropriate *decryption key*, reproduces the plaintext from the ciphertext. A cipher system can be classified through the way the encryption and decryption keys being used. If the knowledge of one of the keys allows one to derive the other key, we call such systems as *symmetric-key encryption*. However, if the knowledge of one of the key does not allow one to derive the other key, we call such system as *asymmetric-key encryption*. Since this thesis is focusing on symmetric-key encryption, a detailed explanation about the asymmetric-key encryptions is not relevant here.

The security of a symmetric-key encryption should only depend on the secrecy of the key and not the secrecy of the encryption algorithm (*Kerckhoff's assumption*). Since both the encryption and decryption keys in the symmetric-key encryption are identical, in practice, the keys represent a shared secret between two or more parties that are involved in a private communication. The main advantage of symmetric-key encryptions is that, they generally consume much less computationally intensive than asymmetric-key encryptions. Thus, they are more suitable to encrypt plaintext messages compared with the asymmetric-key encryption (which are normally used to encrypt a shared secret key in a communication protocol). Two types of symmetric-key encryptions that are widely used are block ciphers and stream ciphers. In the subsequent two sections we will explain the basic constructions and properties of the two types of symmetric-key encryptions.

2.2.1 Block Ciphers

A block cipher is a mapping function that transforms one at a time (i.e. block by block) an n-bit block of plaintext, M, to an n-bit block of ciphertext, C, parameterized by k-bit secret key K which takes the values of the key space $\mathcal{K}$. Each of the values in $\mathcal{K}$ is a k-bit vector V_k. More formally, this function can be defined by a bijective mapping $E : V_n \times \mathcal{K} \rightarrow V_n$ for each $K \in \mathcal{K}$. Thus we call $E_K(M)$ the *encryption function* for K from V_n to V_n, and $D_K(C)$ the *decryption function*, which is the inverse of $E_K(M)$. Consequently, the ciphertext C can be derived by encrypting the plaintext M under the key K, i.e. $C = E_K(M)$, and the original plaintext M can be recovered from the corresponding ciphertext C by decrypting the ciphertext C under the same key K, i.e. $M = D_K(C)$.

To obtain a unique decryption, the mapping function must be one-to-one which requires it to be bijection given n-bit plaintext and ciphertext blocks, and a fixed key. Thus each key $K \in \mathcal{K}$ should be able to define a different bijection. An ideal n-bit block cipher (i.e. a truly random block cipher) should be able to implement all $2^n!$ bijections on 2^n elements for each of the $2^n!$ keys. However it is not practical to implement a truly random block cipher as it requires the size of the key to be $\lg(2^n!) \approx (n - 1.44)2^n$ bits length which is about 2^n times the length of the message block. Since having a truly random block cipher is impractical, it is accepted, in the design principle, if the block cipher only appears as a randomly chosen invertible function corresponding to the randomly selected key.

A notable property about block ciphers is that, the block size should not be too small to avoid statistical analysis such as frequency analysis of ciphertext blocks. However, having too large a block size will increase the complexity for implementation as the complexity grows rapidly with block size. Thus to preserve the random-looked property of the cipher, while being able to minimize the increased complexity of implementation, the cipher function should be chosen in a way that it is easily implementable. Having smaller block size or a plaintext message longer than the block size, requires the block cipher to operate in a certain *mode of operation* to iteratively encrypt the plaintext message using the same secret key securely, and thus avoids vulnerabilities due to the statistical analysis on the ciphertext block.

Iterated Block Ciphers

Most of block ciphers operate by iteratively executing a simpler function many times before outputting the ciphertext block. Each iteration is called a *round* and the simpler function which is executed many times is called the *round function*. The construction of the round function, f, is based on the principles of *confusion* and *diffusion* as proposed by Shannon in his seminal work [126]. Confusion is the act of making the relationship between the key and ciphertext as complex as possible; while diffusion is the act of making each bit of a ciphertext depends on every bit from the plaintext. Each round function, f, will usually take a *round key*, K_i, as the second input (for i is the round number) which is produced by a *key schedule algorithm* from the original key, K. The block cipher which operates in this way is called an *iterated block cipher*. Each round of an iterated block cipher can be depicted as

$$S_i = f_{K_i}(S_{i-1}) \tag{2.1}$$

in which S_0 is the plaintext block and S_r is the ciphertext block, for r is the total number of rounds. To increase the security of an iterated block cipher, normally *key whitening* is applied at the beginning and the end of the cipher operation by adding extra key materials using a simple arithmetic operation such as XOR, addition or subtraction. This can be shown as below:

$$\begin{aligned} S_0 &= M \oplus K_0 \\ S_i &= f_{K_i}(S_{i-1}); \qquad i = 1 \ldots r \\ C &= S_r \oplus K_{r+1} \end{aligned} \tag{2.2}$$

Block Cipher Constructions

The design of block ciphers is normally based on one of the two main constructions namely *Feistel networks* and *substitution-permutation networks* (SPNs). A Feistel network is a construction type due to H. Feistel [63]. In this construction a plaintext block M is divided into two halves, such that $M = \langle L_0, R_0 \rangle$. Then both halves enter the round iteratively in which each round is defined by

$$\langle L_i, R_i \rangle = \langle R_{i-1}, L_{i-1} \oplus f(R_{i-1}) \rangle \tag{2.3}$$

for $i = 1 \ldots r$ and f is the round function made out of a diffusion and confusion layer, and a round key mixing layer. This construction type provides flexibility in

designing strategies as the round function, f, does not need to be bijective. Hence, the decryption can be distinguished from encryption only by the order of the round keys, and the initial and the final swaps of the two halves, which makes it really suitable to be implemented in hardware and any other constrained environments.

In contrast, SPNs consist of round functions constructed from layers of simple invertible functions namely substitution layers (S-boxes), γ, which provide nonlinear transformation for the input, and permutation layers (P-boxes), π, which are linear. Besides, there is also a key addition layer, $\sigma[K_i]$, in each round function. The substitution layers, γ, acts on small data size, usually not more than 8 bits of input data. However the permutation layers, π, perform linear transformations across the complete block. Thus, each round, ρ, in SPNs constructions can be depicted as

$$\rho = \sigma[K_i] \circ \pi \circ \gamma \tag{2.4}$$

for $i = 1 \ldots r-1$ ($\circ$ denotes function composition). Since $\sigma[K_i] \circ \pi = \pi \circ \sigma[\pi^{-1}(K_i)]$, the last π can be simply undone. Implementing key whitening before and after the first and the last round respectively, the whole SPNs can be represented by

$$\sigma[K_r] \circ \gamma \circ (\bigcirc_{i=1}^{r-1} \sigma[K_i] \circ \pi \circ \gamma) \circ \sigma[K_0] \tag{2.5}$$

The main advantage of SPNs construction is that it is easy to analyze and hence can minimize the probability of design flaws occurring.

2.2.2 Stream Ciphers

Stream ciphers encrypt plaintext messages in blocks as small as a single bit using a transformation which varies with time. The idea of stream ciphers is motivated by the one-time-pad (OTP), i.e. a proven unbreakable cipher [126], in which the encryption and decryption transformations are defined by $c_i = m_i \oplus z_i$ and $m_i = c_i \oplus z_i$ respectively (for $i = 1, 2, 3, \ldots$; c_i and m_i is the ciphertext and plaintext digit respectively; and z_i is the keystream digit), where the keystream digits, z_is, are independently generated at random. Instead of using a completely random keystream a stream cipher requires the key length to be at least the same length as the plaintext as in the OTP: stream ciphers use shorter and more convenient keys to generate a pseudorandom keystream. This pseudorandom keystream is combined with the plaintext digits (bits or bytes) one at a time in the similar fashion to the OTP.

Security Consideration for Stream Ciphers

Shannon in [126] proved that for a symmetric-key ciphers to be unconditionally secure (i.e. can provide a perfect secrecy), it is necessary that $H(K) \geq H(M)$, where $H(\cdot)$ denotes the entropy function, while K and M are random variables denoting the secret key and the plaintext respectively. If the secret key is k bits length and is chosen independently at random, then $H(K) = k$. Consequently, the necessary condition for a symmetric-key cipher to be unconditionally secure becomes $k \geq H(M)$. To be more precise, the length of the secret key that is chosen randomly and independently must be at least the same as the length of the plaintext. This explains why the OTP is a secure cipher. However, the requirement of the length of the secret key to be at least the same as the length of the plaintext will introduce the problem of key management and key distribution. Thus, this motivates the design of stream ciphers which generate a *pseudorandom* keystream from a smaller seed (such as a secret key) with the aim that the keystream looks random to a computationally bounded adversary.

Thus, for a stream cipher to be secure, the keystream should only be used once. Having two different ciphertexts, c_1 and c_2, encrypted under the same keystream z, such that $c_1 = m_1 \oplus z$ and $c_2 = m_2 \oplus z$, enables the adversary to XOR both ciphertext to obtain two plaintext messages m_1 and m_2 being XORed together, i.e. $c_1 \oplus c_2 = m_1 \oplus z \oplus m_2 \oplus z = m_1 \oplus m_2$. Knowing that both m_1 and m_2 are plaintext messages in the English language, the adversary can apply dictionary attacks to guess (let's say) m_1 and for each guess to XOR the guessed m_1 with the string $m_1 \oplus m_2$ to deduce the plaintext m_2. The correct guessing of m_1 will result in m_2 being the plaintext message in the English language. To avoid such problems, the keystream must have a large period, which avoids the same keystream being repeatedly used too soon.

To ensure the security of a stream cipher also, the adversary should not be able to recover either the secret key or the cipher's internal state by observing the keystream. The keystream should also "*look random*" in the sense that the adversary should not be able to distinguish the keystream from a random noise. There also should be no weak keys and no detectable relationship between the keystream and the corresponding related keys or related initialization vectors.

2.3 Attack Models

2.3.1 Standard Models

The objective of cryptanalysis of block ciphers is to recover either the plaintext of a given ciphertext or the secret key i.e. the decryption key. Being able to recover the secret key is considered the most powerful attack as the secret key recovered can be used to deduce any other plaintext being encrypted under the same secret key. There are six standard cryptanalytic attack models in which each one of them assumes the adversary have complete knowledge about the cryptographic algorithm being used for key recovery attacks. These are *ciphertext-only attacks*, *known-plaintext attacks*, *chosen-plaintext attacks*, *adaptive-chosen-plaintext attacks*, *chosen-ciphertext attacks*, *adaptive-chosen-ciphertext attacks* and *distinguishing attacks*.

A *ciphertext-only attack* or sometimes also called *known-ciphertext attack* is where the adversary is able to deduce either the secret key or the plaintext by only observing the ciphertext produced by the target cryptographic algorithm. As the adversary only requires the ciphertext of the target cryptographic algorithm, this attack model assumes the weakest adversary. Thus any cryptographic algorithms that are susceptible to this type of attack is considered to be completely insecure. Classical ciphers are known to be susceptible to the ciphertext-only attacks, such as statistical techniques using, for example, frequency analysis. There are also some modern proprietary ciphers which are also susceptible to ciphertext-only attacks such as the RC4 stream cipher [92] and the Akelarre block cipher [83]. Besides, modern ciphers with small key space are also susceptible to ciphertext-only attack through exhaustive key search, such as Content Scramble System (CSS) cipher [59] with 40-bit key length used to encrypt DVD video discs.

A *known-plaintext attack* is where the adversary recovers the secret key using a set of plaintexts and the corresponding ciphertexts encrypted under the same secret key. The adversary does not have the ability to select the plaintext of his choice but he can only use the plaintexts which are currently available to him. Classical ciphers such as Caesar Cipher are susceptible to known-plaintext attack. Knowing a single letter of plaintext and the corresponding ciphertext letter, enables recovering the plaintext message. In modern ciphers, ZIP encrypted file archives are known to be susceptible to known-plaintext attack as shown in [28, 128]. For example, an attack on ZIP requires only one unencrypted file from the archives to form a

known-plaintext attack. Then using software, one can decrypt the whole archive by deducing the secret key.

A *chosen-plaintext attack* is where the adversary is given a power to choose arbitrary plaintexts and then to feed them into the encryption algorithm to deduce the corresponding ciphertexts. The information deduced is used to recover the plaintext from the previously unseen ciphertext. In the worst case, this attack model can also be used to recover the secret key. Any ciphers that are protected against chosen-plaintext attack are also protected against known-plaintext and ciphertext-only attacks. One example of chosen-plaintext attack on block ciphers is the well-known *Differential cryptanalysis* [25] invented by Biham and Shamir in 1993 which was applied on DES [99].

An *adaptive-chosen-plaintext attack* is a type of attack where the adversary not only can choose the plaintext, but he can also modify his choice based on the information obtained from the previous encryption. For example, the adversary can choose a large plaintext block in chosen-plaintext attack, however in adaptive chosen-plaintext attack the adversary can choose a smaller plaintext block and then choose another based on the result of the encryption of the first plaintext block and so forth.

A *chosen-ciphertext attack* is where the adversary chooses arbitrary ciphertexts and then is given the corresponding plaintexts. This attack model requires the adversary to have access to the decryption device but have no access to the decryption key as the key is embedded securely in the device. The aim is to deduce a plaintext from a different ciphertext later on, without having access to such devices.

An *adaptive-chosen-ciphertext attack* is an interactive form of chosen-ciphertext attack in which the adversary chooses the ciphertext based on the information obtained from the previous decryption (i.e. previous plaintext). The aim of the attack is to gradually recover the plaintext message or the decryption key. This attack model remained theoretical until 1998 when a practical attack against systems using RSA algorithm was found [31].

In addition to standard attack models that aim at key recovery attacks, there is also a standard attack model which aims at determining whether the encrypted data is generated by a cipher or by a truly random function. This attack model is known as *distinguishing attacks*. In this attack model a function is said to be a random oracle (i.e. a truly random function) if the adversary cannot predict any of its output. However if it is distinguishable from a random oracle, it is said to have

non-randomness properties such that the adversary might be able to show either a relation between different outputs or between the function input and output.

2.3.2 Other Models

Besides the standard attack models, we have also other attack models (i.e. non-standard) which are also can be found in the literature. In this section we provide two of the most popular type non-standard attack model namely *related-key attacks* and *side-channel attacks*.

A *related-key attack* is a type of attack where the adversary exploits the operation of a cipher under several different unknown keys with some known mathematical relationship connecting the keys. Even this model seems to be unrealistic (since in principle the key should be randomly generated), in reality this might be possible as modern cryptographic algorithms are implemented in some sorts of computers (and not vetted by cryptographers). Thus related key attacks might be possible if there exist some weaknesses in key generation which gives certain relationships between different keys. Wired Equivalent Privacy (WEP) used in WiFi wireless network is known to be susceptible to related-key attack [129]. Each client and access point in the WEP-protected wireless network share the same secret key. However, this secret key needs to be changed manually which results in the key being changed infrequently. Since WEP uses the RC4 stream cipher as the underlying encryption mechanism, the keystream should not be used more than once (cf. Section 2.2.2 for more detailed). Thus to avoid repeated use of the same keystream, WEP uses a 24-bit initialization vector (IV) in each message packet in which the RC4 key is the IV concatenated with the secret key. This results in two packets which will share the same RC4 key in every 4096 packets sent allowing the packets to be attacked.

A *side-channel attack* is a type of attack which exploits the information gained from the implementation of a cryptographic algorithm rather than the weaknesses of the algorithm itself. The information gained from the implementation could be the leakage in the form of power consumptions, electromagnetic fields, radiation, timing etc. Section 2.5 in this chapter provides further explanation on this attack model.

2.4 Algebraic Attacks

Any cryptosystems can be described as a system of multivariate polynomial equations, usually over $\mathbb{GF}(2)$, or over other rings. Having this system of polynomial equations, the adversary can try to deduce the secret key of the cipher by solving the equations. To recover the secret key by solving the polynomial equations describing a cipher is known as *algebraic attack*.

Algebraic attacks began to receive attention in the open literature only in the late 20'th century. Initially, algebraic attacks were applied on the DES algorithm in mid 70s [75], but these efforts failed and were ignored. Then, the emergence of differential [29] and linear [93] cryptanalysis in the early 1990s motivated the efforts to introduce the algebraic structure into cipher designs such as in [101, 23, 100] to prevent both forms of attack. Since then, the efforts for exploiting the algebraic structure within block ciphers had begun, with the first attack was the Jakobsen and Knudsen's *interpolation attack* [81] against SHARK [116]. After Rijndael [47] was chosen as the AES, there was concern about its algebraic structure [65]. Later in 2002, an algorithm to solve system of polynomial equations called XSL (Extended Sparse Linearization) was introduced by Courtois and Pieprzyk to attack block ciphers. This algorithm became controversial as it claimed to have the potential to break the AES. However, Cid and Leurent in [40] provided evidence that the XSL algorithm is not efficient enough to solve the system of equations describing the AES. Although the XSL algorithm has failed, the effort to find new advances in algebraic attacks continues. In this section, we review some algorithms used for solving polynomial equations.

2.4.1 Gröbner Basis Algorithms

It is well-known that while Gaussian elimination has been widely used to solve systems of linear equations, *Gröbner bases* [38] through the *Buchberger Algorithm* is the corresponding algorithm to solve polynomial equations. In this section we briefly review Gröbner bases and the Buchberger algorithm. For a more detailed explanation we recommend two excellence references on the topic, i.e. the book by Cox, Little and O'Shea [44] and the tutorial by Ajwa, Liu and Wang [6].

Definition 2.4 *Let $x_1, \ldots, x_n$ be n variables, $\alpha = (\alpha_1, \ldots, \alpha_n)$ be a power vector in $\mathbb{Z}^n_{\geq 0}$ and x^α as a monomial in $x_1, \ldots, x_n$ such that $x^\alpha = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot \ldots \cdot x_n^{\alpha_n}$. Thus, the*

total degree of x^α *is* $|\alpha| = \alpha_1 + \alpha_2 + \ldots + \alpha_n$. *However, for a multivariate polynomial* f *in variables* $x_1, \ldots, x_n$ *with coefficients in field* k *denoted by* $k[x_1, \ldots, x_n]$, *the total degree of polynomial* $f(x_1, \ldots, x_n) = \sum_\alpha a_\alpha x^\alpha$ *is the highest degree* $|\alpha|$ *such that* $a_\alpha \neq 0$.

Monomial ordering is crucial for the computation of Gröbner bases since the result varies according to the monomial ordering. There are several monomial orderings that can be adopted, such as the *graded reverse lex order* (grevlex) as defined below:

Definition 2.5 *Let* $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{Z}^n_{\geq 0}$. *Then* $\alpha >_{grevlex} \beta$ *iff either* $|\alpha| > |\beta|$, *or (*$|\alpha| = |\beta|$ *and the right-most non-zero entry in* $\alpha - \beta$ *is negative).*

Given a nonzero polynomial $f \in k[x_1, x_2, \ldots, x_n]$ we denote *multideg*(f), $LM(f)$, $LC(f)$ and $LT(f)$ as *multidegree*, *leading monomial*, *leading coefficient* and *leading term* of f respectively (cf. [6] for the definition). Given also a multivariate polynomials F, we denote $I = \langle F \rangle$ as the ideal I generated by F. Thus, under the above definition and notation, we define a Gröbner basis as follows.

Definition 2.6 *We call a finite subset* $G = \{g_1, \ldots, g_m\} \subset I$ *as a Gröbner basis if*

$$\langle LT(g_1), \ldots, LT(g_m) \rangle = \langle LT(I) \rangle$$

Note that a unique Gröbner basis of an ideal I can be determined from the *reduced Gröbner basis* which is defined as below.

Definition 2.7 (cf. [4]) *A Gröbner basis* $G = \{f_1, \ldots, f_2\}$ *of an ideal* I *is called the reduced Gröbner basis if for all* i, $LC(f_i) = 1$ *and any monomial of* f_i *is not divisible by any element of* $LM(G \setminus \{f_i\})$.

The Buchberger Algorithm

In this section we focus on the Buchberger algorithm to compute a Gröbner basis.

Definition 2.8 *Given two nonzero polynomials* $f, g \in k[x_1, \ldots, x_n]$ *and let* $J = l.c.m(LM(f), LM(g))$. *The S-polynomial of* f *and* g *is defined as a linear combination*

$$S(f, g) := LC(g) \cdot \frac{J}{LM(f)} \cdot f - LC(f) \cdot \frac{J}{LM(G)} \cdot g$$

Algorithm (cf. [4]) The Buchberger Algorithm

INPUT : An ordered set $F = (f_1, \ldots, f_m)$ in $k[x_1, \ldots, x_n]$;
OUTPUT : A Gröbner basis $G = (g_1, \ldots, g_s)$ for $I = \langle f_1, \ldots, f_m \rangle$ with $F \subset G$;
$G := F$;
repeat
  $H := G$;
  for each pair (p, q), $p \neq q$ in H
    if $S := \overline{S(p,q)}^{H} \neq 0$ **then**
      $G := G \cup \{S\}$;
    endif
until $H = G$;

Figure 2.1: The Buchberger algorithm to compute Gröbner basis

Given a polynomial $f \in k[x_1, \ldots, x_n]$ and a finite set of polynomials $G \in k[x_1, \ldots, x_n]$ we denote $\overline{f}^{G}$ as a remainder resulting from the division of f by G.

Theorem 2.1 (cf. [4]) *A basis $G = \{g_1, \ldots, g_m\}$ of an ideal $I \in k[x_1, \ldots, x_n]$ is a Gröbner basis if and only if for all pairs $i \neq j$, $\overline{S(g_i, g_j)}^{G} = 0$.*

Thus, the Buchberger algorithm to compute the Gröbner basis is shown in Figure 2.1.

2.4.2 Linearization

Giving a system of polynomial equations of degree d with n variables over $\mathbb{GF}(2)$, the maximum number of possible monomials of degree between 1 and d within the system is bounded by $\sum_{i=1}^{d} \binom{n}{i}$. For example, having a system of multivariate quadratic equations of n number of variables, will give us $(n^2+n)/2$ possible number of monomials within the equation.

To solve a system of polynomial equations using *linearization*, first each monomial within the system is defined as a new variable to form a new system of linear equations. Then a method such as Gaussian elimination is applied on the linear system to find the solution for the new variables. The number of solutions that can be found depends on the number of independent linear equations (or the rank of the set of equations), M, and the number of new variables, N. If $M = N$, Gaussian elimination will give a unique solution for each variable. However if the number of

independently linear equations is less than the number of variables, i.e. $M < N$, then the number of possible solutions given by Gaussian elimination is 2^{N-M}.

If the original system of polynomial equations has a solution, then not all the solutions provided by Gaussian elimination (i.e. when $M < N$) on the linearized system can lead to the solution for the original system of polynomial equations. To illustrate this situation, we use the example provided by [15] as follows.

$$\begin{aligned} x_1 + x_2x_3 &= 1 \\ x_1x_2 + x_1x_3 + x_1 &= 0 \\ x_2x_3 + x_2 &= 0 \\ x_1x_2 + x_1 + x_3 + x_2 &= 0 \\ x_1 + x_1x_2 + x_3 &= 0 \\ x_2x_3 + x_1 + x_2 &= 1 \end{aligned}$$

Considering the above polynomial equations, linearization is performed by replacing each monomial within the equation by a new variable, such that, $x_1 = y_1$, $x_2 = y_2$, $x_3 = y_3$, $x_1x_2 = y_4$, $x_1x_3 = y_5$, $x_2x_3 = y_6$. This results a new system of linear equations as follows.

$$\begin{aligned} y_1 + y_6 &= 1 \\ y_4 + y_5 + y_1 &= 0 \\ y_6 + y_2 &= 0 \\ y_4 + y_1 + y_3 + y_2 &= 0 \\ y_1 + y_4 + y_3 &= 0 \\ y_6 + y_1 + y_2 &= 1 \end{aligned}$$

Note that, the resulting linearized system of equations is of rank $M = 5$, which is less than the number of variables $N = 6$. As a result, the number of possible solutions resulting from Gaussian elimination is $2^{6-5} = 2^1 = 2$. More precisely, the result of applying Gaussian elimination gives $y_1 = 1$, $y_2 = 0$, $y_3 = y_5$, $y_4 = y_5 + 1$, $y_5 = free$ and $y_6 = 0$, which can be either (1,0,0,1,0,0) or (1,0,1,0,1,0) depending on the value of y_5. Denote the vector in the parenthesis as representing $(y_1, y_2, y_3, y_4, y_5, y_6)$. To choose the first vector as the solution is absurd, since it gives $x_1 = 1$, $x_2 = 0$ and $x_3 = 0$, while $x_1x_2 = 1$ contradicts with $x_2 = 0$. Both x_1 and x_2 must be equal to 1 in order for the monomial x_1x_2 to be equal to 1. Thus, the acceptable solution is the second vector which reasonably solves the original polynomial equations.

From this example, it is obvious that if there were a solution in the original polynomial equations, it will definitely provide the solution for the linearized system of equations. However the converse is not necessarily true. The reason for this is that linearization destroys information. Although this is the case, the use of linearization is still useful as it does not destroy solutions. Thus, one can try all the possible solutions resulting from linearization in order to find the correct solution for the original polynomial equations. However this requires the rank of the linearized equations to be as high as possible, such that $M \approx N$, to avoid guessing a huge number of possible solutions which is inefficient.

2.4.3 The XL Algorithm

The standard linearization technique can be extended into a more advanced technique called the *eXtended Linearization* (XL) algorithm to solve multivariate polynomial equations. This algorithm was introduced by Courtois et al. in [43]. Given a system of polynomial equations of degree d in n variables with the number of equations m less then the number of variables, one can solve such system of equations using the XL algorithm by deriving equations of degree D which is a bit higher than the degree d of the original polynomial equations. Typically, the degree D can be either equal to $d+1$ or in some cases to be equal to $d+2$. To derive more equations of degree D, one should multiply each original equation by all possible monomials of degree $D-d$. As a result, the number of equations in the new derived system of equations becomes $m(1+\binom{n}{D-d})$. Having a new and larger set of derived polynomial equations, one applies the standard linearization technique (as discussed in Section 2.4.2) and solves the resulting linearized system using the Gaussian elimination method.

2.4.4 SAT-Solvers

The boolean satisfiability problem (or SAT problem) is a problem to determine if there exists at least an assignment to variables in a given Boolean formula such that the formula evaluates to TRUE. If there exists an assignment which evaluates the formula to TRUE we call the formula is SATISFIABLE, otherwise it as UNSATISFIABLE. The SAT problem is known as an NP-complete problem in that no efficient algorithm can solve all instances of SAT. The past few years have witnessed the progress in the development and performance of the boolean satisfiability solvers,

also known as *SAT-solvers* [122]. Even all known SAT solvers require worse-case exponential run time, they are increasingly being adopted as a general-purpose tool in many different areas and disciplines.

The idea of using SAT-Solvers in algebraic cryptanalysis to solve multivariate polynomial equations was introduced by Bard, Courtois and Jefferson in [17]. Solving multivariate polynomial equations over $\mathbb{GF}(2)$ using SAT-solvers requires the conversion of the polynomial in ANF to SAT (CNF) in several steps.

The first step is to convert the polynomial system into a linear system. This conversion requires one dummy variable to be introduced for each monomial of degree $d > 1$. For example, given a monomial $wxyz$, one should define a new dummy variable, let say a, such that $a = pqrs$. This can then be written in CNF as

$$(p \vee \bar{a})(q \vee \bar{a})(r \vee \bar{a})(s \vee \bar{a})(a \vee \bar{p} \vee \bar{q} \vee \bar{r} \vee \bar{s})$$

In the second step, the derived linear system is converted into CNF. Suppose the linear system is $w + x + y + z = 0$. Thus, the conversion of this linear system into CNF becomes

$$(\bar{w} \vee x \vee y \vee z)(w \vee \bar{x} \vee y \vee z)(w \vee x \vee \bar{y} \vee z)(w \vee x \vee y \vee \bar{z})$$
$$(\bar{w} \vee \bar{x} \vee \bar{y} \vee z)(\bar{w} \vee \bar{x} \vee y \vee \bar{z})(\bar{w} \vee x \vee \bar{y} \vee \bar{z})(w \vee \bar{x} \vee \bar{y} \vee \bar{z})$$

Note that the length of a CNF expression increases exponentially with the length of the linear equation in ANF. To avoid this problem, long linear equations in ANF are cut and separated into shorter equations prior to the conversion. Finally, having a set of equations in CNF, to feed these equations into a SAT-solver requires the equation in CNF to be translated into the format which is understandable by the SAT-solver such as DIMACS CNF file format [121].

2.4.5 The Raddum-Semaev Algorithms

Raddum and Semaev in [113, 112] introduced a new method for solving a sparse polynomial system of equations over $\mathbb{GF}(2)$. The main idea of the method is to build a graph from the system of equations. Each equation, E, within the system corresponds to a vertex in the graph.

Suppose S and T are two different vertices, and $X(S)$ and $X(T)$ denote variables associated with vertices S and T respectively. Then if there exist variables $X(U) = X(S) \cap X(T)$, we create a vertex U (if it does not already exist) and connect both

U to S and T with an edge. We check each possible pair of vertices and create a new vertex and edge where possible to form a complete graph.

Having the completed graph that corresponds to the system of polynomial equations, the next step is to carry the information about all possible solutions the system of equations could give into the graph. More precisely, we assign all possible values to variables in the equations which will give the corresponding value of right-hand side of each equation. With regard to the graph point of view, this corresponds to the list of bit strings, called the *configuration list*, for each vertex in which each bit in the string represents the value of a particular variable within a particular equation. We denote the configuration list for vertices S, T and U as $L(S), L(T)$ and $L(U)$ respectively.

Having the configuration lists of all vertices in the graph, the next step is to find the solution by deleting those configurations which are not part of a solution. Each pair of vertices that are connected with an edge can send the message about their configurations to each other. The vertex that is receiving the message can determine and filter the configuration leaving only those which are part of a solution. This message passing is performed for any pair of vertices connected by an edge and is continued until the graph "*agree*", i.e. no more useful messages can be passed. The process of sending messages to delete configurations which are not part of a solution is called the *Agreeing algorithm*. If the system has a unique solution, after the graph has agreed each vertex will only left with one configuration.

However, there might be a case where no solution can be found using the Agreeing algorithm although the system of equations can provide a unique solution. When this happens, a technique calls *Splitting* is used in which the focus is on a single vertex (let's say) S and splits its configuration list, $L(S)$, into L_1 and L_2. The correct configuration can be found in one of these two new configuration lists. Then to find the correct configuration, one can use a guessing strategy, i.e. to replace $L(S)$ by either L_1 or L_2 one at a time and restart the Agreeing algorithm.

Another way that one can use to restart the Agreeing algorithm is through a method called *Gluing*. In this method two different vertices are combined to form a bigger vertex. Suppose two vertices S and T are to be considered. By gluing S and T together, results in a new vertex Z, such that $X(Z) = X(S) \cup X(T)$ and $L(Z)$ consists of all configurations in $L(S)$ and $L(T)$. Note that, gluing S and T together requires both of them to be discarded as they are replaced by the new vertex Z. With the new vertex Z is in place, the graph need to be reconstructed before restarting the

Agreeing algorithm. However the price that one need to pay in using this method is that, it requires a larger configuration list $L(S)$ such that $max\{|L(S)|, |L(T)|\} \leq |L(Z)| \leq |L(S)| \cdot |L(T)|$. To avoid having too large a configuration list, certain thresholds should be determined. Gluing should be performed only if the resulting configuration list is below the threshold.

2.5 Side-Channel Attacks

In this section, we describe the state-of-the-art of side-channel attacks. A side-channel attack is a type of attack which exploits the information gained from the implementation of a cryptosystem rather than the weaknesses of the algorithm. Information such as power consumption, electromagnetic leaks, timing, temperature etc., can be used as the required information to break the cryptosystem. Side-channel attacks are known to be much more powerful than the standard attacks against the algorithm. They are also considered as a serious threat by cryptographic devices manufacturers. There are two main classification of side-channel attacks

- *Invasive* vs *non-invasive* attacks: An invasive attack is one in which the adversary obtains the side channel information through an intrusive activity such as de-packaging the chip to get access to its internal components. On the other hand, a non-invasive attack is one in which the adversary obtains the side channel information by exploiting externally available information such as electromagnetic fields, radiation and power consumption.

- *Active* vs *passive*. An active attack is where the adversary mount the attack by tampering the cryptographic device functionality. One example of such attack is fault attack in which the adversary induces faults to cause errors to the computation of a cryptographic module in order to to recover secret information. On the other hand, passive attacks only require the adversary to observe the behaviour of a cryptographic device during its execution without disturbing it.

Note that both classes of side-channel attacks are orthogonal in which both invasive and non-invasive attacks can either be active or passive.

2.5.1 Probing Attacks

Probing attacks are an invasive type of attack in which the adversary de-packages the chip to observe its behaviour. Probing is tedious as it requires penetration of the target device to get access to its internal state. Electronic devices such as smart cards are protected by a *passivation* layer, i.e. a shield covering the chip. To apply a probing attack on a smart card, this passivation layer need to be removed. In [88] Kömmerling and Kuhn show ways of de-packaging and probing smart cards.

To mount a probing attack a *probe station* is required which consists of microscopes with micro-manipulators attached. Such equipment is widely used in semiconductor industries for testing purposes. The main target of probing in key recovery attacks is the part of memory that stores the secret key.

2.5.2 Timing Attacks

Timing attacks [84] are a type of attack where the adversary exploits the time taken to execute a cryptographic algorithm in order to recover the secret information. Every cryptographic algorithm takes slightly different amount of time to execute depending on the input (i.e. data or key) provided. This could be due to performance optimization to bypass unnecessary operations depending on the input data, branching etc. Hence, with precise measurement of the time for each execution, secret information can be recovered from the measurement of the time of certain queries.

To eliminate the data-dependent timing information of a cryptographic algorithm, the implementation of the algorithm should hide such information by always executing the algorithm within a fixed range of time. This requires the algorithm to be executed until the maximum possible time it can take to process an input. The drawback of this approach is that it will decrease the average performance of the algorithm and slow down the encryption process especially in block ciphers where the encryption is done block by block. Each block of different input will take the same time to produce an output even if it should be faster without the protection mechanism.

Timing attacks are more dominant in public key cryptosystems compared to symmetric key cryptosystems as the timing characteristics in the symmetric key cryptosystems are not as key-dependent as the public key cryptosystems. As an example of timing attack on public key cryptosystem is the attack by Kocher in

[84] which shows the adversary can find the Diffie-Hellman exponents and factor the RSA key by measuring the private key operations.

2.5.3 Power Analysis Attacks

Generally, *power analysis attacks* [86] are a type of side channel attack which exploit the power consumption of cryptographic devices with the aim to recover the secret key. The motivation for power analysis attack is that all electronic devices consume different power variations when performing different operations and processing different data. As a result, power traces from electronic devices, such as in smart cards executing a cryptographic algorithm, can be easily identified.

Simple power analysis (SPA) involves interpreting power traces over time. In order to apply SPA, the adversary needs to have access to the target cryptographic device in some way. He can have the ability to know the input and output of the device such as the plaintext and the corresponding ciphertext; he can have the ability to invoke the cryptographic device to execute using the (unknown) secret key; or he can also have access to the model of the cryptographic device to enable predicting the intermediate values in the computation of the cryptographic algorithm which depends on the known input and output, and on the secret key. To recover the secret key the adversary needs to perform several steps such as guessing part of the secret key, mapping the intermediate values to hypothetical power consumption values, and comparing the hypothetical power consumption values to the actual device's power consumption values which were obtained before-hand using known input and output values. If the hypothetical power consumption values correlate with the actual device's power consumption values, then all the assumptions and the key guesses are correct. However the correlation should be really high before accepting the guessed key as the correct key.

SPA can be enhanced into a more advanced model called *differential power analysis* (DPA) [86]. In DPA the adversary computes the intermediate values of the cryptographic algorithm computation by statistically analyzing a large number of power traces to reveal the secret key. With a large number of traces the required information can be extracted even from extremely noisy channel. In contrast to an SPA, a DPA analyzes the power consumption at any specific time which depends on the input data. DPA is also much more difficult to prevent than the SPA and requires

no detailed knowledge about the cryptographic device. Furthermore, if the implementation of a cryptographic algorithm is secure against SPA, it is not necessarily secure against DPA. In order to apply DPA, first the power traces that are collected from several cryptographic operations are partitioned into two groups according to the intermediate bit value, b, calculated in each operation. By manipulating this bit during each operation, the observed power consumption may be affected such that different power biases will emerge between the two group of traces at locations when b is manipulated. Finding the average of the traces in each group can help to reduce the noise. To find the locations where the biases occur within the traces can be determined by plotting the difference of the two average traces.

2.5.4 Electromagnetic Analysis Attacks

Every electronic device emits electromagnetic fields during its operation due to the movement of electrical charges. The presence of electromagnetic fields in devices that implement cryptographic algorithms are also no exception. The first side channel attack that exploits the leakage of electromagnetic fields in cryptographic device was introduced by Quisquater and Samyde in [110] and then further developed to involve the use of coils in the attack as described in [69, 111]. This type of attack which exploits the leakage of electromagnetic fields in cryptographic devices to extract the secret information therein is called *electromagnetic analysis* (EMA). The information measured from this attack can be analyzed in a similar fashion as in power analysis. Thus similar to power analysis, EMA can also be implemented as *simple electromagnetic analysis* (SEMA) and *differential electromagnetic analysis* (DEMA) or more advanced correlation attacks.

Placing the coils in three dimensional positions allows more information to be collected from the device. Different information about the underlying computation can be collected from the electromagnetic emanations as shown by Agrawal et al. in [5]. Two main categories of emanations were introduced by Agrawal et al. namely *direct emanations* which results from intentional current flow, and *unintentional emanations* which results from neighboring components in between in close proximity. The unintentional emanations can provide more useful information than the direct emanations and some of them can also have better propagation compared with direct emanations. Compared to power analysis attacks, EMA is a more flexible type of attack and can provide a wider range of information. It can be implemented either

as invasive or non-invasive attacks in order to obtain the required information.

2.5.5 Fault Attacks

The last type of side channel attacks which we want to discuss in this chapter is *fault attacks*. It is considered as an active attack as it requires the adversary to induce faults into a target device that runs a cryptographic algorithm. For more detailed description of this type of attack please refer to Chapter 5 in this thesis which discusses this attack model.

Chapter 3

Side Channel Cube Attacks on NOEKEON

3.1 Introduction

Almost any cryptosystem can be represented by a system of multivariate polynomial equations over a finite field, e.g. $\mathbb{GF}(2)$. The cube attack, formalized by Dinur and Shamir at EUROCRYPT 2009 [55], is a generic type of algebraic attacks. The attack aims to derive low-degree equations that can be exploited for constructing distinguishers, e.g. [12], and/or key recovery attacks, e.g. [55, 12]. An interesting feature of the cube attack is that it only requires a black-box access to a target cryptosystem and may be applied even if only a few output bits can be accessed by an adversary.

For a properly designed ("good") cipher, whose algebraic representation over $\mathbb{GF}(2)$ is of degree d (i.e. the maximum degree of the output bits represented as boolean functions is d), the cube attack will require about 2^d computations. Therefore, *in general*, the attack's success depends on whether this computational complexity is feasible or not; however we note that this "generic complexity" does not imply that the attack may not be successful for specific cases. The cube attack has been successfully applied (for key recovery) against a reduced-round variant of the Trivium [52] stream cipher in [55] and (as a distinguisher) against a reduced-round variant of the MD6 hash function [115] in [12].

In trying to apply cube attacks to block ciphers, the main problem is that the degree of the polynomial representing a ciphertext bit grows exponentially with the number of rounds in the cipher. Hence, the cube attack usually becomes ineffective after a few rounds if one considers only the standard attack model that is used in the well-known statistical attacks such as the differential and linear attacks. Nevertheless, considering the practical implementations of the block cipher, especially

in resource limited systems such as smart cards, there is a stronger attack model, namely the side channel attack model, where the adversary is given more power by having access to some "*limited* " information leaked about the internal state of the cipher. This information leakage can be via physical side channels, such as timing, electrical power consumption, electromagnetic radiation, probing, etc.

Dinur and Shamir in [56] proposed a side channel attack model, where the adversary is assumed to have access to "*only one bit of information*" about the internal state of the block cipher after each round. This one bit of information can be, for example, a single bit of the internal state itself or a bit of information about the Hamming weight of the internal state. They showed that the cube attack in this single-bit-leakage side channel model can recover the secret key of the Serpent [8] and AES [47] block ciphers much easier than the previously known side channel attacks. Yang et al. at CANS 2009 [136] investigated the side channel cube attack on the PRESENT block cipher [35]. It is worth noticing that the single-bit-leakage side channel cube attack against a block cipher is different from a cube attack against a reduced-round variant of the cipher; while in the former the adversary has access to only one bit of information about the internal state, in the latter the adversary has access to the whole internal state after a reduced number of rounds.

In this chapter, we investigate the security of the NOEKEON block cipher [46] against the cube attack in the single-bit-leakage side channel attack model. NOEKEON is designed to be efficient and secure on a wide range of platforms including the environment with limited resources such as smart cards. The designers of NOEKEON have shown how the cipher can be adapted to provide an anti-DPA variant of NOEKEON which is secure against differential power analysis (DPA) attacks. However, the implementation cost for the anti-DPA version is almost twice of that for a normal NOEKEON implementation. They also noticed that to prevent higher-order DPA attacks one will have to use even less efficient variants. If the cipher were to be implemented in a time-critical system, it is likely that this kind of anti-DPA variants not to be implemented due to the efficiency/security trade-off.

Many papers on side channel attacks such as in [90, 91] also concentrate on countermeasures which minimize the leakage from the implementation. We note that these kind of countermeasures against specific types of side channel attacks (e.g. timing attack or DPA) do not remove the possibility of all side channel attacks such as electromagnetic radiations, probing, etc. In our work we do not consider these specific issues and countermeasures about the actual physical aspects of the

implementation attacks and how information leakage can be measured. Rather, we assume the single-bit-leakage side channel model as an abstract attack model, and concentrate on investigating the security of ciphers against cube attacks in this attack model.

3.1.1 Our Contribution

We investigate the security of the NOEKEON block cipher against cube attacks in the single-bit-leakage side channel attack model. First, we show how to determine the appropriate round to find most of the key bits. Using a single bit of the internal state after the second round, we extract 60 *independent* linear equations over 99 key variables. To recover the whole 128-bit key, the attack requires about 2^{10} chosen plaintext and 2^{68} time complexity.

3.2 Cube Attacks: A Theoretical Background

In algebraic attacks, one aims at recovering the secret key in cryptosystems by manipulating and solving the underlying algebraic equations. Solving a system of multivariate equations over a finite field $\mathbb{F}$, in general, is known to be an NP-hard problem [68]. However, it has been demonstrated that finding solutions faster than by the exhaustive search may be possible if the algebraic equations have a relatively low algebraic degree when considered as multivariate polynomial equations. An ideal situation is when one can derive sufficient independent linear equations which are easily solvable (e.g. by Gaussian elimination); the cube attack aims at doing this.

3.2.1 Terminology and Notation

In this section we describe the formal terminology and notation used in the rest of this chapter. The main point of the cube attack is that, the multivariate "master" polynomial $p(v_1, \ldots, v_m, k_1, \ldots, k_n)$, representing an output bit of a cryptosystem over $\mathbb{GF}(2)$ of secret variables k_i (key bits) and public variables v_i (i.e. plaintext or initial values), may induce algebraic equations of lower degrees, in particular *linear* equations. The cube attack provides a method to derive such lower degree (especially linear) equations, given the master polynomial only as a black-box which can be evaluated on the secret and public variables.

Let's ignore the distinction between the secret and public variables' notations and denote all of them by $x_i, \ldots, x_\ell$, where $\ell = m+n$. Let $I \subseteq \{1, ..., \ell\}$ be a subset of the variable indexes, and t_I denote a monomial term containing multiplication of all variables in I. Factoring the master polynomial p by the monomial t_I results in:

$$p(x_1, \ldots, x_\ell) = t_I \cdot p_{S(I)} + q(x_1, \ldots, x_\ell) \tag{3.1}$$

where $p_{S(I)}$, which is called the *superpoly* of t_I in p, does not have any common variable with t_I, and each monomial term t_J in the residue polynomial q misses at least one variable from t_I. A term t_I is called a "*maxterm*" if its superpoly in p is linear polynomial which is not a constant, *i.e.* $deg(p_{S(I)}) = 1$. As an example, suppose a master polynomial p is given as follows

$$p(x_1, x_2, x_3, x_4, x_5) = x_1x_2x_3x_4 + x_1x_2x_3x_5 + x_1x_2x_3 + x_3 + x_4x_5 + x_1x_3 + x_2x_3x_5 \tag{3.1}$$

Choosing a subset $I = \{1, 2, 3\}$ (i.e. $t_I = x_1x_2x_3$), result in a *tweakable polynomial* such that

$$p(x_1, x_2, x_3, x_4, x_5) = x_1x_2x_3(x_4 + x_5 + 1) + (x_3 + x_4x_5 + x_1x_3 + x_2x_3x_5) \tag{3.2}$$

Then $x_4 + x_5 + 1$ becomes the superpoly, $p_{S(I)}$, of the monomial t_I. In this case t_I is called a maxterm as $p_{S(I)}$ is linear. Each monomial within the master polynomial p that misses at least one variable from t_I is assigned to be a term in the polynomial q; i.e., $q = x_3 + x_4x_5 + x_1x_3 + x_2x_3x_5$.

3.2.2 Observation

The main observation of the cube attack is that, if we sum p over t_I, i.e. by assigning all the possible combinations of 0/1 values to variables in I and fixing the value of all the remaining variables not in I, the resultant polynomial equals to $p_{S(I)} \pmod 2$. Summing the polynomial p over all possible values of variables in t_I, requires the polynomial p to be summed over an even number of times. If $|t_I| = L$, then the summation requires 2^L number of derived polynomials to be summed over. During the summation, since the monomial t_I will only equal to 1 when all variables in t_I are set to 1, then the superpoly $p_{S(I)}$ is summed only once. This is because other combination of values for variables in t_I result in $t_I = 0$ and consequently $t_I \cdot p_{S(I)} = 0$. As a result, the summation of $t_I \cdot p_{S(I)}$ over t_I is always equal to $p_{S(I)}$. However, since each term in the polynomial q misses at least one variable from the

Table 3.1: The summation of the master polynomial p in Equation 3.1 over vectors in boolean cube C_I

x_1	x_2	x_3	Derived Polynomials
0	0	0	x_4x_5
0	0	1	$1 + x_4x_5$
0	1	0	x_4x_5
0	1	1	$1 + x_4x_5 + x_5$
1	0	0	x_4x_5
1	0	1	x_4x_5
1	1	0	x_4x_5
1	1	1	$x_4 + x_4x_5 + 1$
	$\sum$		$x_4 + x_5 + 1$

variables in the monomial t_I, each term in q is summed over an even number of time in $\mathbb{GF}(2)$, and hence is canceled out. This explains the result of summing the polynomial p over the monomial t_I.

The above observation can also be explained more formally in boolean cube point of view. A subset I of size L (where $L \leq \ell$) defines a boolean cube C_I containing 2^L boolean vectors which are formed by assigning all 2^L values to the variables in I, and leaving all the remaining variables undetermined. For example, if $I = \{1, 2\}$, then $C_I = \{(0, 0, x_3, \ldots, x_\ell),\ (0, 1, x_3, \ldots, x_\ell),\ (1, 0, x_3, \ldots, x_\ell),\ (1, 1, x_3, \ldots, x_\ell)\}$. Any vector $\mathbf{w} \in C_I$ defines a derived polynomial $p_{|\mathbf{w}}$ with $\ell - L$ variables whose degree may be the same or lower than the degree of the master polynomial p. Summing the 2^L derived polynomials over $\mathbb{GF}(2)$ defined by the vectors in the cube C_I, we get a new polynomial p_I defined by $p_I \triangleq \sum_{\mathbf{w} \in C_I} p_{|\mathbf{w}}$.

Considering the master polynomial, p, in Equation 3.1 and choosing $I = \{1, 2, 3\}$, the summation of p over all boolean vectors $\mathbf{w} \in C_I$ results in the derived polynomials as in Table 3.1. The following theorem states the main observation used by the cube attack.

Theorem 3.1 (The Main Observation [55]) *Given a polynomial* p over $\mathbb{GF}(2)$ *with* ℓ *variables, and any index subset* $I \subseteq \{1, \ldots, \ell\}$*, we have* $p_I = p_{S(I)}$*.*

Given access to a cryptographic function with public and secret variables, this observation enables an adversary to recover the value of the secret variables (k_is) in two steps, namely *preprocessing* and *online phase*.

3.2.3 Preprocessing Phase

During the preprocessing phase, adversary first finds sufficiently many maxterms, i.e. t_Is, such that each t_I consists of a subset of public variables $v_1, \ldots, v_m$. To find the maxterms, the adversary performs a probabilistic linearity test on $p_{S(I)}$ over the secret variables $k_i \in \{k_1, \ldots, k_n\}$ while the value of the public variables not in t_I are fixed (to 0 or 1). For example, the BLR test of [32] can be used for this purpose. This test requires the adversary to choose a sufficient number of vectors $\mathbf{x}$, $\mathbf{y} \in \{0,1\}^n$ independently and uniformly at random representing samples of n-bit key, and then for each pair of vectors $\mathbf{x}$ and $\mathbf{y}$, the adversary sums the polynomial p over t_I to verify whether or not each one of them satisfies the relation:

$$p_{S(I)}[\mathbf{0}] + p_{S(I)}[\mathbf{x}] + p_{S(I)}[\mathbf{y}] = p_{S(I)}[\mathbf{x} + \mathbf{y}] \tag{3.3}$$

If all the vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy the relation, with high probability $p_{S(I)}$ is either linear over the secret variables (i.e. t_I is a maxterm) or a constant superpoly. Thus the adversary needs to distinguish between linear and constant superpolys as both of them satisfy the Relation (3.3) for any given vectors $\mathbf{x}$ and $\mathbf{y}$.

One way to distinguish between the two types of superpolys is to count the number of 0/1 values from N computations of $p_I[\cdot] = p_{S(I)}[\cdot]$. Suppose all N vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy Relation (3.3) and M is the number of $p_I[\cdot] = p_{S(I)}[\cdot] = 1$. The adversary is assured that the corresponding superpoly is linear iff $0 < M < N$. However, for $M = N$ or $M = 0$ the corresponding superpoly is *constant 1* or *constant 0* respectively. Note that, a constant 0 superpoly indicates a *nonexist* superpoly, or in other words the selected monomial t_I does not exist within the master polynomial p. If all the test on superpoly $p_{S(I)}$s results in constant 0 superpoly, it gives an indication that the size L of the monomial t_I is larger than the degree d of the master polynomial p. Thus the adversary needs to reduce the size L by omitting one variable at a time from the monomial t_I and restarts the linearity test. This *random walk* is continued until maxterms are found in which $L = d - 1$. However if none of the monomial t_Is that are selected satisfies Relation (3.3), it shows that the monomials' size L is much smaller than the degree d of the master polynomial p. Hence the adversary needs to increase the size of t_Is in their selection by repeatedly adding one more variable at a time until maxterms are found.

Each time a maxterm is found, the adversary derives the corresponding linear equation over the secret variables k_is from $p_{S(I)}$. For a particular maxterm t_I, to

derive the corresponding linear superpoly $p_{S(I)}$, first the adversary sets all variables (i.e. secret and public variables) not in the monomial t_I to zero. Then, the adversary chooses one secret variable k_i at a time and computes $p_I[k_i = 1]$ and $p_I[k_i = 0]$. If the results of these computations are different, it shows that k_i exists as a term in the superpoly $p_{S(I)}$, otherwise it does not exist in $p_{S(I)}$. The adversary needs to repeat this for all other k_is. To complete the equation, the adversary also needs to determine the existence of a constant term within $p_{S(I)}$. To do this, all variables not in the monomial t_I are set to zeros values and then compute p_I. If the result is equal to 1, then it shows that there exists a constant term within the superpoly $p_{S(I)}$.

3.2.4 Online Phase

Once sufficiently many linearly independent equations in the secret variables are found, the preprocessing phase is completed. In the online phase, the adversary's aim is to find the value of the right-hand side of each linear equation by summing the black box polynomial p over the same set of maxterms t_Is which are obtained during the preprocessing phase. Now, the adversary can easily solve the resultant system of the linear equations, e.g. by using the Gaussian elimination method, to determine the values of the secret (key) variables.

3.3 A Brief Description of the NOEKEON Block Cipher

NOEKEON [46] is a block cipher with a block and key length of 128 bits. It produces a ciphertext after iterating a round function 16 times, followed by a final output function. The specification of NOEKEON [46], provides a key schedule which converts the 128-bit "Cipher Key" (i.e. the original key) into a 128-bit "Working Key", which is used in the round function. However, the use of the key schedule is optional. If related-key attack scenarios [25] are not of a concern, then the key schedule is not applied (i.e. the Cipher Key is used directly as the Working Key), and the cipher is called to be used in the "direct-key mode". Otherwise, it operates in the "indirect-key mode", where the Cipher Key is processed by the key schedule algorithm to produce the Working Key.

Table 3.2: Specification of the S-box in the function Γ

Input:	$x_3x_2x_1x_0$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output:	$y_3y_2y_1y_0$	7	A	2	C	4	8	F	0	5	9	1	E	3	D	B	6

A graphical representation of the round function of NOEKEON is shown in Fig. 3.1. It consists of two linear functions, Θ and Π (as illustrated in the figure), and a nonlinear function Γ which is described shortly. We describe the encryption mode of the cipher. The description of the cipher in the decryption (i.e. inverse) mode is also quite straightforward, where the inverse of the component functions are used (we refer to [46] for a complete description of the cipher). The constants C_1 and C_2 are two round constants. C_2 is set to zero during the encryption process and C_1 is set to zero during the decryption process. The constant C_1 that is used during the encryption process can be computed in recursive way as follows:

C_1^0 `= 0x80;`
`if (`C_1^r `& 0x80 != 0) then` C_1^{r+1} `=` $C_1^r \ll 1$ `^ 0x1B`
`else` C_1^{r+1} `=` $C_1^r \ll 1$`;`

Let $A_r = A_r^0 A_r^1 A_r^2 A_r^3$ denote the 128-bit internal state after round r; where A_r^is are 32-bit words, and A_0 contains the input plaintext P to the cipher. Then NOEKEON encryption algorithm can be described as follows:

$$\begin{aligned} &\text{For } r = 0;\ r < 16;\ r{+}{+} \\ &\quad A_{r+1} = \Pi^{-1}(\Gamma(\Pi(\Theta(A_r, K)))); \\ &A_{17} = \Theta(A_{16}, K) \end{aligned}$$

where A_{17} denotes the 128-bit output ciphertext, and $K = K_0K_1K_2K_3$ is the 128-bit working key (K_is are 32-bit words).

The nonlinear transformation Γ operates on 32 4-tuples of bits (4-bit boxes) using a S-box which is shown in Table 3.2 (4-bit values are represented by a hexadecimal number). The 4-bit boxes which are input to the S-Boxes, at the round r, are formed by selecting four bits from the words $A_r^0, A_r^1, A_r^2, A_r^3$, that are in the same position. For example, box 1 consists of the first bit from each word, box 2 contains the second bit from each word, and so on.

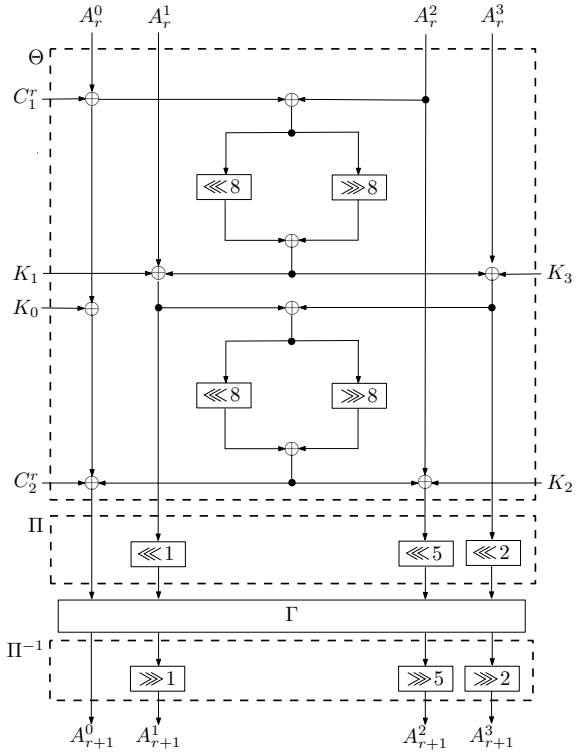


Figure 3.1: The round function of NOEKEON

3.4 The Side Channel Cube Attack on NOEKEON

3.4.1 Finding the Efficient Round for Side Channel Attacks

In order to apply the single-bit-leakage side channel cube attack on the NOEKEON block cipher, we need to find out the round in which the cipher begins achieving complete diffusion. This enables us to find most of the key variables from low degree master polynomials in early rounds of the encryption process.

To do this, we determine the number of key variables that can be found within the master polynomials representing each bit of the internal state after each round; i.e. the master polynomials representing $A_r^j[i]$ for $1 \leq r \leq 16, 0 \leq j \leq 3$, and $0 \leq i \leq 31$.

The following lemma is a corollary of the main observations supporting the cube attack as described in Section 3.2.

Lemma 1. *Let p be a given (black-box) master polynomial over $\mathbb{GF}(2)$ in ℓ variables $x_1, \ldots, x_\ell$; $I \subseteq \{1, \ldots, \ell\}$; $L = |I|$, and t_I denote the multiplication of variables in I. Let $p_I \triangleq \sum_{\mathbf{w} \in C_I} p_{|\mathbf{w}}$ be the derived polynomial obtained by summing p over the cube C_I (see Section 3.2). t_I appears in p, either as a monomial or as a common*

subterm of some monomials, if and only if there exist at least a vector $\mathbf{x} \in \{0,1\}^{\ell-L}$ *which satisfies* $p_I[\mathbf{x}] = \sum_{\mathbf{w}\in C_I} p_{|\mathbf{w}}[\mathbf{x}] = 1$.

Proof: If t_I is neither a monomial nor a common subterm of some monomials in the master polynomial p, then each monomial term in p must miss at least one variable from t_I. Therefore, in the summation of p over the cube C_I each monomial will be summed an even number of times, and hence will be canceled out as the coefficients are from $\mathbb{GF}(2)$; that is, $p_I \equiv 0$ (and hence for all $\mathbf{x} \in \{0,1\}^{\ell-L}$ we have $p_I[\mathbf{x}] = 0$). To complete the proof, simply note that if t_I appears as a monomial or a common subterm of some monomials in p, then p can be factored as $p(x_1, \ldots, x_\ell) = t_I \cdot p_{S(I)} + q(x_1, \ldots, x_\ell)$, where $p_{S(I)}$ is either the constant 1 (when t_I is an independent monomial in p) or a polynomial over at most $\ell - L$ variables (when t_I is a common subterm of some monomials). From Theorem 1 we know that $p_I = p_{S(I)}$, and hence $p_I[\mathbf{x}] = p_{S(I)}[\mathbf{x}] = 1$ for *at least* one vector $\mathbf{x} \in \{0,1\}^{\ell-L}$. □

This lemma essentially provides the underlying idea for the following probabilistic test, proposed originally in [12], to detect whether a given t_I appears either as a monomial or as a common subterm of some monomials within a master polynomial p:

Step 1: select a value of vector $\mathbf{x} = (x_{L+1}, \ldots, x_\ell) \in \{0,1\}^{\ell-L}$.

Step 2: sum the polynomial $p(x_1, \ldots, x_\ell)$ over the cube C_I (i.e. over all values of $(x_1, \ldots, x_L)$, where $L = |I|$), to get $p_I[\mathbf{x}]$ $(= p_{S(I)}[\mathbf{x}])$.

Step 3: repeat the previous two steps N times and record the values of $p_I[\mathbf{x}]$.

A random master polynomial p is expected to contain at least a monomial $x_1 \ldots x_L \ldots x_j$ $(j \geq L)$ in which $t_I = (x_1, \ldots, x_L)$ is either the monomial itself (for $j = L$) or a subterm of the monomial (for $j > L$), with high probability. Hence, after a sufficiently large number of repetitions N, one would find at least one nonzero superpoly $p_{S(I)}$ with high probability. However if t_I is neither a term nor a common subterm in p, the superpoly $p_{S(I)}$ will always evaluate to zero.

We apply this test to *estimate* the the number of key variables within the master polynomials representing each bit of the internal state; i.e. the master polynomials representing $A_r^j[i]$ for $1 \leq r \leq 16, 0 \leq j \leq 3$, and $0 \leq i \leq 31$. To determine whether a key variable k_i, $0 \leq i \leq 127$, exists in a master polynomial p, the following process is repeated for each i $(0 \leq i \leq 127)$. Let $t_I = k_i$, randomly choose "sufficiently many" vectors $\mathbf{x} \in \{0,1\}^{128+127}$ (i.e. the space for 128 bits of the plaintext and the

Table 3.3: The number of key variables within the black box polynomials.

	$A_r^0[i]$ $0 \leq i \leq 31$	$A_r^1[i]$ $0 \leq i \leq 31$	$A_r^2[i]$ $0 \leq i \leq 31$	$A_r^3[i]$ $0 \leq i \leq 31$
1^{st} round $(r = 1)$	28	28	28	21
2^{nd} round $(r = 2)$	127	127	127	123
$3^{rd} - 16^{th}$ round $(r \geq 3)$	128	128	128	128

secret variables excluding k_i), and sum the polynomial p over t_I for each sample vector $\mathbf{x}$. If at least one sample vector $\mathbf{x}$ results in $p_I = 1$, then from Lemma 1, we can conclude that the key variable k_i exists within the master polynomial p. In our experiment, only after testing about 300 random sample vectors, we have been able to successfully determine that a high portion of the key variables exist in the master polynomials after only two rounds, as shown in Table 3.3.

We also need to estimate the degree d of each (black-box) master polynomial in order to verify whether the cube attack can be efficiently implemented. For this purpose, we construct the boolean functions representing output bits of the NOEKEON's S-boxes. Let x_i and y_i be, respectively, the input and output bits of the S-box, for $i = 0, 1, 2, 3$. Each output bit y_i can be represented as a boolean function, as follows.

$$
\begin{aligned}
y_0 &= x_0 + x_1 + x_2 + x_0x_1 + x_0x_2 + x_0x_3 + x_1x_3 + x_2x_3 + x_0x_2x_3 + 1 \\
y_1 &= x_2 + x_3 + x_0x_1 + x_1x_2 + x_0x_2x_3 + x_1x_2x_3 + 1 \\
y_2 &= x_0 + x_1 + x_1x_2 + x_2x_3 + 1 \\
y_3 &= x_0 + x_1x_2
\end{aligned}
$$

It is easy to see that the highest degree of these four boolean functions is 3, corresponding to y_0 and y_1. Since the only nonlinear transformations in NOEKEON are the S-boxes (used by the function Γ), we can estimate the degree d of the master polynomial for any round. If we consider a single bit leakage after third round, the degree of the master polynomials would be approximately (at most) 27, which is still considerably practical for cube attacks. However if we consider the degree after the second round, it turns out that the degrees of the master polynomials are at most 9. Considering that diffusion process is almost complete after the second round

Table 3.4: Cube indexes for maxterms and the linear superpoly for each maxterm.

Cube Indexes	Maxterm Equation	Cube Indexes	Maxterm Equation
{11,75,82}	k_{39}	{0,53,60,97}	$k_{113}+k_{89}+k_{57}+k_{49}+k_{33}$
{27,82,91}	k_{55}	{0,4,39,62}	$k_{98}+k_{34}+k_{26}+k_{10}+k_2+1$
{1,34,42,127}	k_{100}	{1,4,34,58}	$k_{119}+k_{111}+k_{95}+k_{71}+k_{55}+k_{47}$
{0,4,49,126}	k_{102}	{1,3,25,66}	$k_{118}+k_{110}+k_{94}+k_{70}+k_{54}+k_{46}$
{2,7,36,107}	k_{105}	{0,1,24,99}	$k_{116}+k_{108}+k_{92}+k_{68}+k_{52}+k_{44}$
{1,34,42,79}	k_{108}	{0,20,31,99}	$k_{112}+k_{96}+k_{80}+k_{64}+k_{48}+k_{32}$
{105,118,126}	$k_{107}+1$	{0,28,39,62}	$k_{114}+k_{106}+k_{50}+k_{42}+k_{26}+k_2$
{2,23,36,123}	k_{121}	{0,5,22,86}	$k_{113}+k_{97}+k_{81}+k_{65}+k_{49}+k_{33}$
{0,12,49,126}	k_{110}	{82,86,123}	$k_{114}+k_{106}+k_{90}+k_{66}+k_{50}+k_{42}$
{118,121,126}	k_{123}	{0,2,75,87}	$k_{115}+k_{107}+k_{91}+k_{67}+k_{51}+k_{43}$
{1,34,58,119}	k_{124}	{3,4,62,76}	$k_{116}+k_{61}+k_{60}+k_{52}+k_{36}+k_{28}$
{5,33,62,85}	$k_{126}+1$	{70,82,123}	$k_{120}+k_{96}+k_{80}+k_{72}+k_{56}+k_{32}$
{4,13,37,126}	k_{33}	{20,84,119}	$k_{122}+k_{114}+k_{74}+k_{66}+k_{58}+k_{50}$
{104,120,126}	k_{36}	{0,2,15,67}	$k_{122}+k_{106}+k_{58}+k_{42}+k_{26}+k_{10}$
{96,104,126}	k_{44}	{1,3,9,66}	$k_{123}+k_{107}+k_{91}+k_{75}+k_{59}+k_{43}$
{0,7,8,64}	k_{60}	{1,4,34,42}	$k_{126}+k_{118}+k_{78}+k_{70}+k_{62}+k_{54}$
{0,6,50,67}	k_{63}	{2,3,101,125}	$k_{116}+k_{60}+k_{52}+k_{45}+k_{37}+k_{12}+k_4$
{2,31,36,107}	k_{97}	{0,7,9,107}	$k_{120}+k_{104}+k_{96}+k_{56}+k_{40}+k_{32}+k_0+1$
{0,3,101,125}	$k_{98}+1$	{1,4,13,126}	$k_{118}+k_{110}+k_{102}+k_{78}+k_{54}+k_{46}+k_{38}+1$
{46,121,126}	$k_{99}+1$	{1,5,92,99}	$k_{125}+k_{109}+k_{101}+k_{69}+k_{61}+k_{45}+k_{37}$
{0,8,47,74}	$k_{103}+k_{38}$	{1,2,44,107}	$k_{127}+k_{119}+k_{111}+k_{63}+k_{56}+k_{55}+k_{47}+k_{23}+1$
{2,89,96,101}	$k_{118}+k_{53}$	{1,2,60,89}	$k_{119}+k_{111}+k_{55}+k_{47}+k_{40}+k_{32}+k_{31}+k_7+1$
{0,7,8,90}	$k_{119}+k_{54}$	{1,36,99,126,127}	$k_{111}+k_{103}+k_{56}+k_{47}+k_{39}+k_{32}+k_{31}+k_{23}$
{0,8,47,66}	$k_{127}+k_{62}$	{3,38,114,124,127}	$k_{57}+k_{41}+k_{33}+k_1$
{30,82,123}	$k_{98}+k_{90}+k_{74}+k_{66}+k_{34}$	{27,54,122,124,127}	$k_{121}+k_{33}+k_{25}+k_9+k_1+1$
{0,10,19,95}	$k_{99}+k_{91}+k_{75}+k_{67}+k_{35}$	{105,108,120,124,126,127}	$k_{92}+k_{61}+k_{28}+1$
{0,1,64,99}	$k_{100}+k_{92}+k_{76}+k_{68}+k_{36}$	{57,120,121,124,126,127}	$k_{113}+k_{57}+k_{49}+k_{33}+k_{25}+1$
{4,6,11,34}	$k_{103}+k_{95}+k_{79}+k_{71}+k_{39}+1$	{102,113,122,124,126,127}	$k_{121}+k_{65}+k_{57}+k_{41}+k_{33}+1$
{6,62,68,74}	$k_{104}+k_{80}+k_{72}+k_{64}+k_{40}$	{94,121,122,124,126,127}	$k_{108}+k_{60}+k_{44}+k_{37}+k_{36}+k_4$
{0,5,6,70}	$k_{105}+k_{81}+k_{73}+k_{65}+k_{41}$	{78,113,114,124,126,127}	$k_{127}+k_{119}+k_{79}+k_{71}+k_{63}+k_{55}$

(refer to Table 3.3) and fewer chosen plaintexts are required, it turns out that the second round is a more appropriate round for single-bit-leakage side channel attack purposes.

3.4.2 Finding Maxterm Equations

As an example of the single-bit-leakage attack, in this section we provide the results of the analysis to find maxterm equations from a master polynomial associated with the first bit of the internal state after the second round, i.e. $A_2^0[0]$. (We note that this process can also be straightforwardly applied to any other single bit position in the internal state after the second round, i.e. considering the master polynomial of any $A_2^j[i]$ for $0 \leq j \leq 3$, and $0 \leq i \leq 31$.) By running the preprocessing phase of the attack (on a single PC) for several weeks, we have been able to find collectively thousands of maxterm equations using different cubes sizes, where most of them were found to be redundant and linearly dependent equations. To filter the equations and obtain only linearly independent equations among them, we used the well-known Gaussian elimination. The elimination gives us only 60 linearly independent equations over 99 key variables. Table 3.4 shows the indexes of variables

in the maxterms and the corresponding linearly independent equations that we have obtained. (The indexes for both the plaintext and the key variables start from index 0, namely the MSB, until 127.)

As shown in the table, the maxterms start to appear within t_Is of size 3; we have 2 maxterms of size 3, 50 maxterms of size 4, 3 maxterms of size 5, and 5 maxterms of size 6. Hence, the total number of the chosen plaintexts for the online phase of the cube attack is $2 \times 2^3 + 50 \times 2^4 + 3 \times 2^5 + 5 \times 2^6 \approx 2^{10.27}$. Considering that we have 60 linearly independent equations over the key variables, the total time complexity to find the correct 128-bit key reduces to 2^{68} (compared to the 2^{128} for an exhaustive key search attack).

3.5 Summary

We investigated the security of the direct-key mode of the NOEKEON block cipher against cube attacks in the (single-bit-leakage) side channel attack model. Our analysis shows that one can recover the 128-bit key of the cipher, by considering a one-bit information leakage from the internal state after the second round, with time complexity of 2^{68} evaluations of the cipher, and data complexity of about 2^{10} chosen plaintexts. At this step, we have been able to find 60 linearly independent equations over 99 key variables, but from an initial observation, it seems that some nonlinear equations of low degree especially of degree 2 can also be easily found, which may further reduce the complexity of the attack. We leave extending the attack and exploiting such quadratic equations for a future research.

Chapter 4

Extended Cubes and the Attack on PRESENT

4.1 Introduction

The cube attack, put forth by Dinur and Shamir at EUROCRYPT 2009 [55], is a generic type of algebraic attacks that may be applied against any cryptosytem, provided that the attacker has access to a bit of information that can be represented by a "low-degree" multivariate polynomial over $\mathbb{GF}(2)$ of the secret and public variables of the target cryptosytem. Dinur and Shamir in [55] compared the cube attack to some of the previously known similar techniques and stated that the attack generalizes and improves some of those methods. As for some of the previously known similar attacks, which exploit the vulnerability of ciphers with low-degree polynomials, we refer to [89, 131].

The cube attack aims to derive low-degree (especially linear) implicit equations that can be exploited for constructing distinguishers, e.g. [12], and/or key recovery attacks, e.g. [55, 12]. An interesting feature of the cube attack is that it only requires a black-box access to a target cryptosystem and may be applied even if only a few output bits can be accessed by an adversary. When using the original cube attack [55, 136], one tries to derive independent *linear* equations over secret variables of the cryptosystem. This system of linear equations can be easily solved to recover the value of the secret variables by using the well-known Gaussian elimination method.

Our work is motivated by the observation that in most cases, for properly designed cryptographic algorithms, it may not be possible to extract a sufficient number of independent 'linear' equations using the (preprocessing phase of the) original cube attack. In fact various potential extensions of the cube attack were suggested to be considered for future research in [55]. One of the methods to generalize the original cube attack, left in [55] for future work without further elaboration, is that

one should try to find and employ some additional low degree nonlinear equations, e.g. equations of degree 2 or 3, provided that the system of equations is simple (sparse and low degree) and solvable using existing methods. In this paper we elaborate this idea and develop an extension of the cube attack to extract such (low degree) nonlinear equations. To demonstrate the application of our extended cube method, we provide a side channel cube attack against the PRESENT block cipher [35], which improves upon the previous work of Yang, Wang and Qiao at CANS 2009 [136].

In attempting to apply cube attacks to block ciphers, the main problem is that the degree of the polynomial representing a ciphertext bit grows with the number of rounds in the cipher. Hence, the cube attack usually becomes ineffective after a few rounds if one considers only the standard attack model that is used in the well-known statistical attacks, such as the Differential and Linear attacks. Nevertheless, considering the practical implementations of the block cipher, especially in resource limited systems such as smart cards, there is a stronger attack model, namely the side channel attack model, where the adversary is given more power by having access to some "*limited*" information leaked about the internal state of the cipher. This information leakage can be via physical side channels, such as timing, electrical power consumption, electromagnetic radiation, probing, etc.

We note that the idea of combining algebraic cryptanalysis with side channel attacks was already introduced by Bogdanov, Kizhvatov and Pyshkin at INDOCRYPT 2008 [34], and also recently investigated in several other works such as [114, 56, 136]. Compared to the recent side channel cube attack of Yang et al. [136], our attack in this chapter offers two improvements: it is based on a more relaxed leakage model; namely, the Hamming weight leakage model, and it has a better (i.e. reduced) complexity as well. The improved complexity is due to applying the extension of the cube attack, to derive simple low degree nonlinear (especially quadratic) equations, which itself is of an independent interest, as the primary contribution in this chapter.

4.1.1 Previous Side Channel Cube Attack on PRESENT

The leakage model used by Yang et al. [136] assumes that adversary has access to the exact value of some of the internal state bits after each round. We note that obtaining the exact value of the internal state bits in practice will require a probe station that allows the attacker to monitor the value of a specific bit position in the

internal state during the encryption or decryption process. This implies an intrusive physical measurement and is known to involve a wide range of difficulties such as penetrating the device to access its internals and guessing which bit position is being recorded. To relax the leakage model, in contrast, we assume the Hamming weight leakage as a more common side channel leakage model, e.g. see [7, 24, 41, 42, 95].

From time and data complexity viewpoints, we show that, for PRESENT-80 (80-bit key variant of PRESENT), our attack has time complexity of 2^{16} and data complexity of about 2^{13} chosen plaintexts; whereas, the attack of Yang et al. has time complexity of 2^{32} and needs about 2^{15} chosen plaintexts. Also our method directly applies to PRESENT-128 (i.e. 128-bit key variant) with time complexity of 2^{64} and the same data complexity of 2^{13} chosen plaintexts, and is the only attack in this model considered against PRESENT-128.

We should stress that both of these side channel cube attacks against PRESENT, provided by Yang et al. in [136] and our attack in this thesis, need clean leaked data values (i.e. the exact value of some internal state bits in the case of [136] and the Hamming weight of the internal state in our case). Hence, to the best of our knowledge these are only of a theoretical interest, at the moment, and do not directly impose any real threat to the security of PRESENT implementations in practice, where the side channel information measurements (e.g. power traces, EM radiations, or timing) are almost always noisy. We refer to [56, 114] for some related discussions on the possibility of handling the noisy data obtained from side channels when combined with the algebraic attacks. We note that, the current issue is that these methods are very sensitive to measurement noise levels and can only handle very low error rates than what may happen in practice.

4.2 Extensions and Generalizations of Cube Attacks

After introduction of cube attacks by Dinur and Shamir [54, 55], several generalizations and extensions have been put forth. In this section we review the generalizations and extensions of cube attacks which we were aware of while writing this thesis.

4.2.1 Side Channel Cube Attacks

Side channel cube attack, as discussed in Chapter 3 and this chapter, has been proposed by Dinur and Shamir in [54]. Dinur and Shamir also elaborated on some further generalizations of the cube attack in [56, 57]. The side channel cube attack is more suitable to be applied to iterated block ciphers where the same physical hardware is executed iteratively during the encryption process. Since the information gained from side channels is error prone, due to noise and quantization problems, a method for dealing with such problems (i.e. an error correction method) was also proposed by Dinur and Shamir in [54]. However we note that their error correction method increases the degree of the master polynomials (i.e. d) and hence the attack complexity (2^d) also increases.

4.2.2 Cube Attacks based on Low Degree Annihilators

Zhang, Lim, Khoo, Lei and Pieprzyk, [138] introduced three variants of cube attacks. Zhang et al. assumed that the structure of the target cipher is known to the adversary as opposed to the original cube attack which consider it as a black box. The main idea of the extension by Zhang et al. was to extend the cube attack to polynomials f for which a low degree polynomial g can be found such that computation of fg also results in a low degree polynomial. This idea was then generalized and called *sliding window cube attack* with annihilators which was dedicated to stream ciphers whose filter function is smaller than the LFSR. Besides, the idea was also generalized to include vectorial filter functions (i.e. stream ciphers where the state function is filtered by a vectorial boolean function) such that $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^r, r > 1$, and called *vectorial cube attack*.

4.2.3 Cube Testers

Cube testers, introduced by Aumasson, Dinur, Meier and Shamir [12], is a generalization of cube attacks that was built as a "*distinguisher*". The aim is to detect non-randomness based on algebraic property-testing algorithms. Compared to the original cube attack which requires the polynomial describing the cipher to be low degree in order to be successful, some cube testers do not need the polynomial to be low degree. Suppose $\mathcal{F}_n$ is a set of all functions mapping $f : \{0,1\}^n \rightarrow \{0,1\}, n > 0$ and a family $\mathcal{F} \subsetneq \mathcal{F}_n$. Thus, the general idea of a cube tester is that, given a

randomly sampled function $f \subsetneq \mathcal{F}^*$ for $\mathcal{F}^* \in \{\mathcal{F}, \mathcal{F}_n\}$, it determines whether $\mathcal{F}^*$ is chosen from $\mathcal{F}$ or $\mathcal{F}_n$. Cube testers apply distinguishing attacks on the superpoly $p_{S(I)}$ of the tweakable master polynomial p.

Several examples of testable properties have been shown by Aumasson et al. in [12] as follows:

- *Balance.* A function f is balanced if the numbers of zeros and ones it outputs over its input set are the same. Hence a superpoly $p_{S(I)}$ which does not possess this property is said to be non-random.

- *Constantness.* A superpoly $p_{S(I)}$ can be distinguished from a random function if it defines a constant function. This happens either when the cube size $|t_I|$ is slightly greater than the degree of the master polynomial p, i.e. $deg(p)$, which yields the constant 0 superpoly (i.e. non-exist superpoly), or the cube has the same size as the degree of p which yields the constant 1 superpoly. Besides, given a monomial $t_I = x_1x_2x_3$ and $|t_I| < deg(p)$, the corresponding $p_{S(I)}$ determines a constant function, if and only if there exists no term in the master polynomial p whose sub-monomial is in the form of $x_1x_2x_3\ldots$, or $x_1x_2x_3$ is the only term in p having such a sub-monomial.

- *Low degree.* A superpoly can also be distinguished from a random function if it defines a low degree polynomial. One particular case of low degree test is to test for the presence of *linear variables*. Given $x_0, x_1, \ldots, x_n$ as the parameters of the master polynomial p, in order to test for the linearity of (for example) x_0, one perform the algorithm shown in [12] as follows.

  1. pick random $(x_2, \ldots, x_n)$
  2. **if** $p(0, x_2, \ldots, x_n) = p(1, x_2, \ldots, x_n)$
  3. **return** NONLINEAR
  4. **repeat** steps 1 to 3 N times
  5. **return** LINEAR

- *Neutral variables.* The existence of certain variables within the superpoly $p_{S(I)}$ can also be used as a distinguisher. This lead to neutrality test of the variables. The following algorithm [12] can be used as a neutrality test.

  1. pick random $(x_2, \ldots, x_n)$

2. **if** $p(0, x_2, \ldots, x_n) \neq p(1, x_2, \ldots, x_n)$
3. **return** NOT NEUTRAL
4. **repeat** steps 1 to 3 N times
5. **return** NEUTRAL

4.2.4 FPGA Implementation of Cube Testers

Aumasson et al. in [11] generalized cube testers to run 256 instances of GRAIN-128 [74] in parallel in FPGA implementation. The aim is to speed up the cube tester. This method can also be applied in the preprocessing phase of the original cube attacks. Besides FPGA, other tools are also used, i.e. a *bitsliced* implementation in software [26] (e.g. a C program) and a simple evolutionary algorithm to find *good cubes*.

Aumasson et al. show that the bitsliced C implementation on a cube of size 30 requires 45 minutes to run 64 instances of GRAIN-128 compared with more than one day using the original C code implementation as provided by the designers of GRAIN-128. The bitsliced software implementation is used to find good parameters for the experiment in hardware. However for hardware implementation of cube testers, 256 instances of GRAIN-128 were run in parallel. This method enables 2^{54} clockings of GRAIN-128 which is the heaviest run to be performed with 256×32 parallelization. An extrapolation of this result is that the full GRAIN-128 algorithm can be distinguished in time 2^{83} compared with 2^{128} for exhaustive search.

4.2.5 Meet-in-the-Middle Cube Attack

One of the generalizations of cube attacks suggested by Dinur and Shamir in [54] was to combine a cube attack with a meet-in-the-middle attack. Mroczkowski and Szmidt [98] show how to apply such attack on Courtois Toy Cipher (CTC). As shown by Mroczkowski and Szmidt, in applying this generalization on CTC, the adversary first finds many independently linear equations during the preprocessing phase from the reduced round cipher (i.e 4 rounds of CTC) in a similar fashion as in the original cube attack.

After obtaining sufficiently independent linear equations from the reduced round cipher, the preprocessing phase finishes and the adversary moves to the online phase. In the online phase the adversary adds one more round and encrypts the plaintext

of the cubes obtained from the preprocessing phase (that provide the independent linear equations) and collects the ciphertext after the five rounds. Then for each cube, the adversary decrypts for one round all the corresponding ciphertexts resulting from the same cube and sums the output of the decryptions over the ciphertext. One round decryption yields boolean functions in ciphertext and key bits. Then the equation obtained from 4 rounds of encryption and one round of decryption of the same cube are equated.

4.2.6 Dynamic Cube Attacks

Dinur and Shamir in [58] further generalized cube attacks and call the new variant as *dynamic cube attacks*. The idea of the attack is to recover the secret key by exploiting distinguishers obtained from cube testers. The main problem of the original cube attack is that, it is inefficient to perform 2^d computations if the degree d of the master polynomial p describing a cipher is high. Dynamic cube attacks aim at handling the problem by providing lower degree representations of the cipher.

The main difference between the original and the dynamic cube attacks lies in the value of variables $\forall x_i \notin t_I$. Unlike the original cube attack, some of the variables in the dynamic cube attack (called dynamic variables), that are not a subset of the variables of the monomial t_I, are not set to a fixed value. Instead, each of the chosen variables is a function in some variables within the monomial t_I (which are public) and some expressions in secret variables. The aim is to eliminate some of the state bits or expressions by changing the value of the dynamic variables according to the assigned functions and hence amplifying the bias or non-randomness for the cube tester.

More precisely, suppose P_1, P_2 and P_3 are polynomial equations such that $P = P_1P_2+P_3$. Suppose also P_1 and P_3 are low degree polynomials and P_2 is a polynomial of high degree which appears random. Then, it is very likely that the polynomial P is not susceptible to the original cube testers. However by setting the polynomial $P_1 = 0$, the adversary can eliminate P_2 and thus making $P = P_3$. To set $P_1 = 0$ one way is to find a linear variable x_i within P_1 and set $x_i = P_1 - x_i$. The summation of the polynomial over the cube results in the value of variable x_i being changed accordingly to match the value of the assigned function, making $P_1 = 0$.

4.3 Extended Cubes: Deriving Low Degree Equations

The idea of extending cube attacks to obtain low degree equations in addition to the linear ones have been suggested by Dinur and Shamir in [54] without further elaboration. In this section, we provide the answer how such low degree equations can be obtained efficiently using a technique we call as *extended cube* method as follows.

4.3.1 The Main Observation

Based on the the output of the BLR linearity test [32], which is used in [55] during the process of finding a maxterm; i.e. t_I with associated linear superpoly $p_{S(I)}$, we can distinguish the following cases:

1. If the test output is "success", i.e., all (tested) vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy Relation (3.3) (cf. Chapter 3) then we have either:
   - *a superpoly that is constant 1*; i.e. $p_{S(I)} = 1$, which trivially satisfies Relation (3.3) for any pair of vectors $\mathbf{x}$ and $\mathbf{y}$.
   - *a non-exist superpoly*, which may happen if t_I is neither a monomial nor a common subterm of some monomials in the master polynomial p, and hence cannot be factored out. This makes $p_I = 0$ for any pair of vectors $\mathbf{x}$ and $\mathbf{y}$, and hence both sides of Equation 3.3 evaluate to 0.
   - *a linear superpoly*, which is the case if neither of the previous two cases (i.e. constant 1 or non-exist case) happens.
2. If the test output is "fail", i.e., at least one pair of vectors $\mathbf{x}$ and $\mathbf{y}$ is found not satisfying Relation (3.3) then the superpoly $p_{S(I)}$ is *nonlinear*.

Note that the above observation is the basis for constructing a distinguisher as shown in [12], which we then formalized as Lemma 1 as in Chapter 3 of this thesis. Next, we propose an efficient method for deriving nonlinear equations of low degree. We introduce the notion of *extended cube* to efficiently extract nonlinear equations of degree D as follows.

Definition 1. *Adopting the notion of a boolean cube C_I (cf. Section 3.2 of Chapter 3), where $I \subseteq \{1, \cdots, \ell\}$ is the index subset and $L = |I|$, an extending cube C_K*

indexed by a subset $K \subseteq \{1, \cdots, \ell\}$ *of size* r *(i.e.* $r = |K|$*) such that* $I \cap K = \phi$*, can be combined with the cube* C_I *to construct a larger extended cube as* $C_{I \cup K}$ *consisting of* 2^{r+L} *boolean vectors formed by assigning* 2^{r+L} *values to all variables in* $I \cup K$*, and leaving all the remaining variables (i.e. all variables not in* $I \cup K$*) undetermined.*

To illustrate Definition 1, let $I = \{1, 2\}$ and $K = \{3\}$, then

$C_{I \cup K} = \{(0, 0, 0, x_4, \cdots, x_\ell), (0, 0, 1, x_4, \cdots, x_\ell), \cdots, (1, 1, 1, x_4, \cdots, x_\ell)\}$

Any vector $\mathbf{w} \in C_{I \cup K}$ defines a derived polynomial $p_{|\mathbf{w}}$ with $\ell - (r + L)$ variables whose degree may be the same or lower than the degree of the master polynomial p. Summing the 2^{r+L} derived polynomials over $\mathbb{GF}(2)$ defined by the vectors in the extended cube $C_{I \cup K}$, we get a new polynomial $p_{(I \cup K)}$ defined by $p_{(I \cup K)} \triangleq \sum_{\mathbf{w} \in C_{I \cup K}} p_{|\mathbf{w}}$. Thus, we can revise the notion of tweakable master polynomial p in Equation 3.0 (cf. Chapter 3) as

$$p(x_1, ..., x_\ell) = t_I \cdot X_K \cdot p_{S(I \cup K)} + q(x_1, ..., x_\ell) \tag{4.0}$$

where t_I is a subterm of size L over variables in I; X_K is a subterm of size r over variables in K, and $p_{S(I \cup K)}$ is the superpoly of $t_I \cdot X_K$ in p. Note that since we factored out both subterms t_I and X_K from p, the superpoly $p_{S(I \cup K)}$ does not contain any common variable with t_I and X_K, and each term t_J in the residue polynomial q misses at least one variable from $t_I \cdot X_K$. Now using the main theorem of the cube attack (namely, Theorem 1), if we sum p over '$t_I \cdot X_K$', by assigning all the possible combinations of 0/1 values to the variables in $I \cup K$ and fixing the value of all the remaining variables not in $I \cup K$, the resultant polynomial equals to $p_{S(I \cup K)} \pmod 2$; i.e. $p_{(I \cup K)} = p_{S(I \cup K)}$.

This observation enables the attacker to derive nonlinear superpoly equations of degree D over the secret variables k_is in two steps; namely the preprocessing and online phases, as described in the following.

4.3.2 Attack Phases

Preprocessing Phase. During the preprocessing phase to derive polynomial equations $p_{S(I)}$s of degree D, the degree d of the master polynomial p should be estimated in some way such as through the known structure of the cipher or using a variant of the random walk as proposed in [67]. Knowing the degree d of the master polynomial enables the attacker to know the size of t_I (i.e. $L = d - D$) in

order to find the nonlinear superpoly of degree D. Next, the attacker finds many monomials t_Is, such that each t_I consists of a subset of public variables $v_1, \cdots, v_m$, and the corresponding superpoly $p_{S(I)}$ is a polynomial of degree D. To find those t_Is, the attacker chooses a monomial t_I of size L one at a time and performs the generalized version of the BLR test as proposed by Dinur and Shamir in [55] on $p_{S(I)}$ over the secret variables $k_1, \cdots, k_n$, while the value of the public variables v_is with $i \notin I$ are fixed (to 0 or 1). For example, if one uses this generalized version of the BLR test to capture the superpoly $p_{S(I)}$ of degree 2, κ sets of vectors $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \{0,1\}^n$, representing samples of n-bit secret keys, are chosen independently and uniformly at random, and then for each pair of vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ the attacker sums the polynomial p over t_I to verify whether or not each one of them satisfies the Relation:

$$p_{S(I)}[\mathbf{0}] + p_{S(I)}[\mathbf{x}] + p_{S(I)}[\mathbf{y}] + p_{S(I)}[\mathbf{z}] + p_{S(I)}[\mathbf{x}+\mathbf{y}] + p_{S(I)}[\mathbf{x}+\mathbf{z}] + p_{S(I)}[\mathbf{y}+\mathbf{z}]$$
$$= p_{S(I)}[\mathbf{x}+\mathbf{y}+\mathbf{z}] \tag{4.1}$$

Note that, the relation will capture all polynomials of degree $D \leq 2$. Hence, to obtain only the polynomials of degree 2, one should filter the linear, constant 1 and non-exist superpolys during the test. Having all κ sets of vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ satisfying the Relation (4.1), one can tell that the superpoly $p_{S(I)}$ (of the monomial t_I) is of degree at most 2, after repeating the test sufficiently many times, that is using a sufficient number of samples κ.

To derive efficiently a nonlinear equation $p_{S(I)}$ of degree D over secret variables k_is, one should identify the subset $S \subseteq \{1, \cdots, n\}$ that consists of the secret variable indexes within $p_{S(I)}$, in which each k_i with $i \in S$ is either a term or a subterm of $p_{S(I)}$. To do this, the subterm X_K (cf. Equation 4.0) is assigned with each secret variable $k_i \in \{k_1, \cdots, k_n\}$ one at a time while the subterm t_I is fixed to the monomial in which its superpoly $p_{S(I)}$ is of degree D, and all public variables v_is with $i \notin I$ are fixed to 0 or 1. For each assignment of X_K, we choose κ sets of vector $\mathbf{x} \in \{0,1\}^{n-1}$ representing samples of $n-1$ secret variables k_is with $i \notin K$ independently and uniformly at random, and verify that X_K (or similarly the secret variable k_i that is assigned to X_K) exists as a variable in the superpoly $p_{S(I)}$ if $p_{(I \cup K)}[\mathbf{x}] = \sum_{\mathbf{w} \in C_{I \cup K}} p_{|\mathbf{w}}[\mathbf{x}] = 1$ for at least an instance vector $\mathbf{x}$ due to Lemma 1 of Chapter 3. This procedure is shown by Algorithm 1 in Figure 4.1.

Having the set of secret variables k_is with $i \in S$ of the nonlinear superpoly $p_{S(I)}$

Algorithm 1.

INPUT : n; // the total number of secret key variables
t_I; // the monomial in which $deg(p_{S(I)}) = D$
OUTPUT : S; // the set of secret variable indexes within $p_{S(I)}$
repeat
assign X_K (cf. Equation 4.0) with a secret variable $k_i \in \{k_1, \cdots, k_n\}$ which has not been considered;
choose κ vectors $\mathbf{x} \in \{0,1\}^{n-1}$ independently and uniformly at random;
repeat
choose one of κ vectors $\mathbf{x}$ which has not been used to represent $n-1$ secret variables $k_i \notin X_K$;
compute $p_{(I \cup K)}[\mathbf{x}] = \sum_{\mathbf{w} \in C_{I \cup K}} p_{|\mathbf{w}}[\mathbf{x}]$;
until $p_{(I \cup K)}[\mathbf{x}] = 1$ **or** all κ vectors $\mathbf{x}$ have been used;
if $p_{(I \cup K)}[\mathbf{x}] = 1$ is the case **then**
consider X_K as a variable within the superpoly $p_{S(I)}$ of degree D;
endif
until all n secret variables k_is have been considered;

Figure 4.1: Finding secret variables within a superpoly equation

of degree D enables the attacker to derive the nonlinear equation over the secret variables by finding all terms of degrees $0, 1, \cdots, D$ within the superpoly equation.

Lemma 2. *The monomial t_I (cf. Equation 3.0 of Chapter 3) is a term in polynomial p if and only if fixing all variables not in I to zeros results in $p_I = \sum_{\mathbf{w} \in C_I} p_{|\mathbf{w}} = 1$. On the other hand, the monomial X_K (cf. Equation 4.0) is a term in the superpoly $p_{S(I)}$ if and only if fixing all variables not in $I \cup K$ to zeros results in $p_{(I \cup K)} = \sum_{\mathbf{w} \in C_{I \cup K}} p_{|\mathbf{w}} = 1$.*

Proof. If t_I is a term in the polynomial p, assigning all variables not in I to zeros will result t_I becomes the only remaining term in p; that is, the value of the superpoly $p_{S(I)}$ is evaluated to 1. From the fact that $p_I = p_{S(I)}$ due to Theorem 1, we have $p_I = 1$ which shows the existence of t_I as a term in p. On the other hand, if X_K is a term in the superpoly $p_{S(I)}$, assigning all variables not in $I \cup K$ to zeros results in $t_I \cdot X_K$ becomes the only remaining term in the polynomial p. Hence, the value of the superpoly $p_{S(I \cup K)}$ of $t_I \cdot X_K$ is evaluated to 1. Since $p_{(I \cup K)} = p_{S(I \cup K)}$ (cf. Section 4.3.1) then $p_{(I \cup K)} = 1$, which shows the existence of $t_I \cdot X_K$ as a term in p. Hence, the existence of $t_I \cdot X_K$ as a term in p implies that X_K exists as a term in the superpoly $p_{S(I)}$ because X_K (factored out from the superpoly $p_{S(I)}$) is a subterm

of the term $t_I \cdot X_K$. □

Algorithm 2.

INPUT : S; // a set of indexes for secret variables k_is in a superpoly $p_{S(I)}$
D; // the degree of the superpoly $p_{S(I)}$
T; // a set of cube indexes for monomials of degree 1 until D over k_is with $i \in S$
$N = |S|$; // the number of secret variables k_is in $p_{S(I)}$
t_I; // in which $deg(p_{S(I)}) = D$
$r = 0$; // the initial size for terms in $p_{S(I)}$
OUTPUT: $p_{S(I)}$s; // the derived low degree nonlinear equations
repeat
  increase the size r by 1;
  repeat
    assign X_K with a monomial from T of size r which has not been considered;
    assign all variables not in $I \cup K$ to 0s;
    if $p_{(I \cup K)} = \sum_{\mathbf{w} \in C_{I \cup K}} p_{|\mathbf{w}} = 1$ **then**
      write X_K as a term in superpoly $p_{S(I)}$;
    endif
  until all $\binom{N}{r}$ terms have been considered;
until $r = D$;
assign public variables v_is with $i \notin I$ and secret variables $k_1, \cdots, k_n$ to 0s;
if $p_I = \sum_{\mathbf{w} \in C_I} p_{|\mathbf{w}} = 1$ **then**
  write '1' as a constant in superpoly $p_{S(I)}$;
endif

Figure 4.2: Deriving a nonlinear superpoly equation of degree D

Suppose $N = |S|$ is the number of secret variables k_is with $i \in S$ of the superpoly $p_{S(I)}$ of degree D. To derive $p_{S(I)}$, firstly we assign the subterm X_K one at a time with a monomial indexed by a subset $K \in T$ where T is a set of cube indexes of monomials constructed from all combinations of k_is from degree 1 until degree D with $i \in S$. For example, if $S = \{1, 2, 3\}$ and the degree of the superpoly $p_{S(I)}$ is 2, then $T = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}\}$.

In each assignment, all variables v_is and k_is not in $I \cup K$ are set to zeros. Then to verify the existence of the monomial $X_K \in T$ as a term in $p_{S(I)}$, we compute $p_{(I \cup K)} = \sum_{\mathbf{w} \in C_{I \cup K}} p_{|\mathbf{w}}$. If the value of $p_{(I \cup K)}$ is equal to 1, then with probability 1, X_K is a term in the superpoly $p_{S(I)}$ due to Lemma 2. Finally, we determine whether there also exists a constant term (i.e. a term of degree 0) in the superpoly $p_{S(I)}$ by

setting all public variables v_is not in I and all secret variables $k_1, \cdots, k_n$ to zeros, and computing $p_I = \sum_{w \in C_I} p_{|\mathbf{w}}$. Similarly, if the value of p_I is equal to 1, then with probability 1, a constant term exists within the superpoly $p_{S(I)}$ due to Lemma 2. Thus, the procedure for finding all terms in the nonlinear superpoly $p_{S(I)}$ of degree D requires $\binom{N}{1} + \binom{N}{2} + \ldots + \binom{N}{D} + 1$ number of computations. The procedure is shown by Algorithm 2 in Figure 4.2. The total complexity to deduce a nonlinear equation of degree D using Algorithm 1 and Algorithm 2 is bounded by

$$\kappa n 2^{d-D+1} + \sum_{i=0}^{D} 2^{d-D+i} \binom{N}{i}$$

Online Phase. Once sufficiently many equations in the secret variables k_is have been found, the preprocessing phase is complete. In the online phase, the attacker's aim is to find the value of the right-hand side of each of the equations (both linear and nonlinear ones) by summing the black box polynomial p over the same set of t_Is obtained during the preprocessing phase. Now, the attacker can solve the resultant system of equations, e.g. by using the standard linearization method, a SAT solver [17], etc., to determine the values of the secret variables.

4.4 A Brief Description of the PRESENT Block Cipher

PRESENT [35] is a block cipher with a 64-bit block. The recommended key length is 80 bits, but a 128-bit key variant is also proposed. PRESENT produces a ciphertext after iterating a Substitution-Permutation Network (SP-Network) 31 times. In each round, a 64-bit round key $k_i = \kappa_{63}\kappa_{62}\cdots\kappa_0$, for $1 \leq i \leq 32$ is introduced to the current STATE $b_{63}\cdots b_0$ using addRoundKey as follows:

$$b_j \rightarrow b_j \oplus \kappa_j^i$$

for $0 \leq j \leq 63$. The round key K_{32} is used for post-whitening after the final round. The addition of round key k_i using the addRoundKey in each round always follows by sBoxLayer (i.e. a nonlinear substitution layer) and pLayer (i.e. a linear bitwise permutation). The graphical representation of PRESENT is shown in Figure 4.3.

PRESENT uses a single 4-bit S-box S as shown in Table 4.1, which is applied 16 times in parallel in each round. The 4-bit nibble i at the input of an S-box is substituted by the 4-bit $\mathsf{S}[i]$ in output, i.e. $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$.

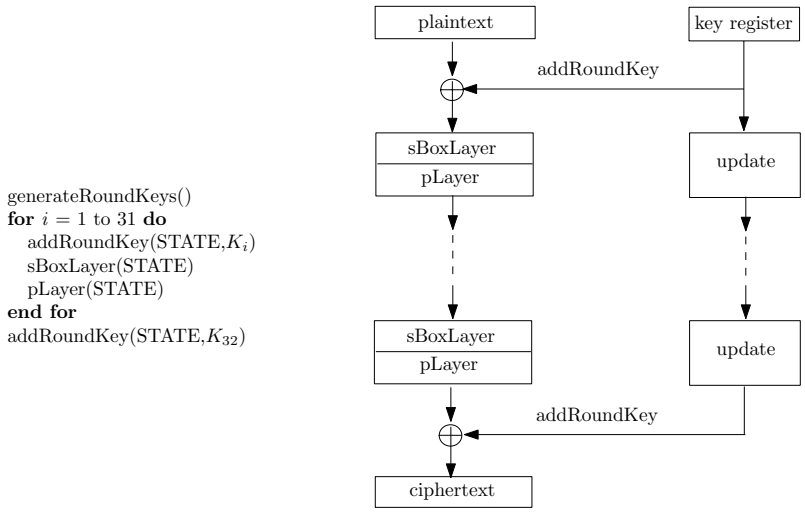


Figure 4.3: A top-level algorithmic description of 31-round PRESENT encryption.

Table 4.1: Specification of PRESENT S-box.

i		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\mathsf{S}[i]$		C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

The pLayer which provide linear bitwise permutation is shown in Table 4.2. During the permutation, each bit i of STATE is moved to bit position $P(i)$.

The difference between the 80-bit key and the 128-bit key variants of PRESENT is on the key schedule.

For the 80-bit key variant, the user-supplied key that is stored in key register K is represented as $k_{79}k_{78}\cdots k_0$. The 64-bit round key $K_i = \kappa_{63}\kappa_{62}\cdots\kappa_0$ consists of the leftmost bits of the current contents (i.e. at round i) of register K. Thus the round key at round i can be depicted as:

$$K_i = \kappa_{63}\kappa_{62}\cdots\kappa_0 = k_{79}k_{78}\cdots k_{16}$$

For each round i, after applying addRoundKey, the key register K is updated as in the following steps:

1. $[k_{79}k_{78}\cdots k_1k_0] = [k_{18}k_{17}\cdots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \texttt{round\_counter}$

Table 4.2: Specification of the PRESENT permutation layer.

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

For the 128-bit key variant, the user supplied key that stored in a key register K is represented as $k_{127}k_{126}\cdots k_0$. Similarly, the 64-bit round key $K_i = \kappa_{63}\kappa_{62}\cdots\kappa_0$ consists of the leftmost bits of the current contents (i.e. at round i) of register K. Thus the round key at round i for 128-bit key variant can be depicted as:

$$K_i = \kappa_{63}\kappa_{62}\cdots\kappa_0 = k_{127}k_{126}\cdots k_{64}$$

After applying addRoundKey in each round, the key register K is updated as in the following steps:

1. $[k_{127}k_{126}\cdots k_1k_0] = [k_{66}k_{65}\cdots k_{68}k_{67}]$
2. $[k_{127}k_{126}k_{125}k_{124}] = S[k_{127}k_{126}k_{125}k_{124}]$
3. $[k_{123}k_{122}k_{121}k_{120}] = S[k_{123}k_{122}k_{121}k_{120}]$
4. $[k_{66}k_{65}k_{64}k_{63}k_{62}] = [k_{66}k_{65}k_{64}k_{63}k_{62}] \oplus \texttt{round\_counter}$

4.5 Side Channel Cube Attack on PRESENT

In this section, we explain our side channel cube attack against PRESENT using the Hamming weight leakage model and our extended cube method.

4.5.1 Hamming Weight

Let $B = b_{\beta-1}\cdots b_0$ be the binary string of length β bits, representing the internal state of the cipher. The Hamming weight of B is the number of bits with value 1 in the binary representation of B, which can be computed as $HW(B) = \sum_{j=0}^{\beta-1} b_j$

and has a value between 0 and β. Clearly, the Hamming weight can also be viewed as a boolean vector mapping $HW : \{0,1\}^b \rightarrow \{0,1\}^{\lfloor \log_2 b \rfloor + 1}$, where the first bit of $HW(B)$, i.e. the least significant bit (LSB), is the XOR of all bits from B; the MSB of $HW(B)$ is the AND of all bits from B; and each bit in between is a boolean function whose degree increases as the bit position gets closer to the MSB.

4.5.2 Application to the PRESENT Block Cipher

The difference between the two variants of PRESENT is in the key schedule algorithm as described in Section 4.4. Note that, at each round i the addition of the round key k_i to the internal state using addRoundKey is performed before updating the key register K. Therefore, the first 64 bits of the secret key for both variants (i.e. $k_{79} \cdots k_{16}$ for the 80-bit key variant PRESENT-80 and $k_{127} \cdots k_{64}$ for the 128-bit key variant PRESENT-128) are applied directly to the internal state at the first round. Hence, both variants are equally susceptible to any attack that can recover the first 64 bits of the secret key, e.g. by considering the Hamming weight leakage after the first round as is the case for our attack.

Assuming access to the value of the Hamming weight of the internal state after the first round, which can be represented as a byte, we consider all bits starting from the LSB position towards the MSB position (of the 8-bit binary representation of the Hamming weight). To search for maxterms, we use various cube sizes which increase as we consider more bits towards the MSB. We follow the technique which is a variant of the random walk [67] as used also in [55] to find the implicit degree d of the master polynomial p. Knowing the degree d of the master polynomial enables us to find the maxterms easily, as the maxterms are expected to be found when we choose the cube of size $L = d-1$. To know whether a particular selected monomial t_I is a maxterm, we apply the BLR test by choosing 100 pairs of vectors $\mathbf{x}$, $\mathbf{y} \in \{0,1\}^n$ (where n is the length of the secret key which is either 80 for PRESENT-80 or 128 for PRESENT-128), and verify t_I as a maxterm only if all the 100 pair of vectors ($\mathbf{x}$, $\mathbf{y}$) satisfy Relation (3.3). The linear equation is then derived by finding all terms within the equation. To find whether a secret variable k_i is a term of the linear equation, we check whether the value of $p_I = \sum_{\mathbf{w} \in C_I} p_{|\mathbf{w}}$ will change if we change the value of k_i to either 0 or 1, while setting all other variables (i.e. secret and public variables) which are not in the monomial t_I to fixed values. If changing the value of the secret variable k_i changes the value of p_I, one can tell that k_i is a term

in the linear equation. To check whether a constant term exists in the equation, we set all the variables not in the monomial t_I to zeros and compute p_I and check if it is equal to 1 that indicates the existence of the constant term in the equation.

We ran our simulation for several weeks to find maxterms; i.e., t_Is with "linear" superpoly. However, it was not possible to find enough linearly independent equations. Then, using the extended cube method, we continued finding some additional quadratic equations. To do this we chose the cube of size $s = d - 2$. This time, to know whether a particular monomial t_I which we select has a superpoly $p_{S(I)}$ of degree 2, we apply the generalized BLR test (cf. Section 4.3.1) by selecting 100 samples of vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z} \in \{0,1\}^n$ and verify that $p_{S(I)}$ is of degree at most 2 only if all the 100 samples of vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ satisfy Relation (4.1). The quadratic equation is then derived using two steps. First, we find the secret variables k_is that exist within the quadratic equation using Algorithm 1 where we choose 300 (i.e. $\kappa = 300$) samples of vector $\mathbf{x}$ independently and uniformly at random (cf. Lemma 1 of Chapter 3 and Algorithm 1 for more detail). Secondly, we derive all terms of degree 0 (constant terms), degree 1 and degree 2 within the quadratic equation using Algorithm 2.

After searching for nonlinear superpoly equations of degree 2, we have been able to find 62 simple quadratic equations, which combined with 32 independent linear equations (among the previously obtained set of linear equations), can provide a unique solution for all the 64 secret key variables. The system of equations can then be easily solved by linearizing the system and solving it using the well-known Gaussian elimination method. Table A.1 and Table A.2, which are provided in the Appendix A, show the sets of equations which can provide the unique solution for the 64 secret key variables, respectively, for PRESENT-80 and PRESENT-128.

Having this set of equations, the preprocessing phase is completed and the attacker during the online phase should find the value of the right-hand side of each equation by summing the master polynomials (corresponding the second and third bits of the Haming weight) over each cube which is listed in Table A.1 and Table A.2. Since we have 2 cubes of size 2, 62 cubes of size 4 and 30 cubes of size 8, then we require $2 \times 2^2 + 62 \times 2^4 + 30 \times 2^8 \approx 2^{13}$ chosen plaintexts to find the value of right-hand side of all 94 equations. Considering that we have been able to find a unique solution for all the 64 secret key variables, the total time complexity to find the correct 80-bit key and 128-bit key reduces to 2^{16} and 2^{64} respectively.

4.6 Summary

We showed an extension to the cube attack to efficiently derive low degree nonlinear superpoly equations to help improve the success level in a key recovery attack, especially in the cases that only limited number of maxterms can be found by the original cube attack. We also investigated the security of the PRESENT block cipher against side channel cube attack, assuming the Hamming weight leakage model and applying our extended cube attack. Our attack improves upon the previous attack of Yang et al. on PRESENT-80, from both leakage model and complexity viewpoints. Improving the attack against PRESENT-128, e.g. by combining more leakage information from several other rounds of the cipher, and investigating the applicability of the proposed extended cube attack to other ciphers, especially in the traditional cryptanalytical framework (without side channel leakage assumptions), are proposed as interesting works for future research.

Chapter 5

Fault Attacks on the KATAN Family of Block Ciphers

5.1 Introduction

Fault analysis as a type of side channel attack (or implementation attack) was originally introduced by Boneh et al. [36] by an attack against implementations of public key algorithms. The method was then adapted and extended by Biham and Shamir [30] to differential fault analysis, making it applicable to implementations of symmetric key algorithms as well [76, 77]. Several models for fault attacks have been introduced in the literature, among which we adopt a popular model, called transient single-bit fault model, as used for example in [27, 77, 76]. In this model it is assumed that adversary can induce one bit of error into the internal state of a cipher during its execution (e.g. using a laser beam) without damaging the bit position permanently; that is, the cipher can be reset to resume its normal (unfaulty) operation and this fault induction can be repeated as many times as required. For some interesting practical settings for carrying out these attacks we refer to [127].

In this chapter we present fault attacks on the KATAN family of block ciphers [51]. KATAN consists of three variants with 32, 48 and 64-bit block sizes, named KATAN32, KATAN48 and KATAN64, respectively. All three variants have the same key length of 80 bits. KATAN aims at meeting the needs of an extremely resource-limited environment such as RFID tags. Assuming the transient single-bit fault attack model, we present a differential fault attack empowered by the algebraic techniques of the cube attack [55] and its extended variants [2] (cf. Chapter 4 of this thesis).

The cube attack, put forth by Dinur and Shamir at EUROCRYPT 2009 [55], is a generic type of algebraic attack that may be applied against any cryptosytem, provided that the attacker has access to a bit of information that can be represented

by a low-degree multivariate polynomial over GF(2) of the secret and public variables of the target cryptosytem. Dinur and Shamir in [55] compared the cube attack to some of the previously known similar techniques [89, 131]. In Chapter 4 of this thesis, we have shown an extended variant of the cube attack to extract low-degree (mainly quadratic) sparse system of equations in addition to the linear equations obtainable from the original cube attack. We use these techniques together with fault analysis to build a hybrid attack against KATAN.

First, we show how to apply the cube and extended cube methods to extract a system of low-degree polynomial multivariate equations in the key and plaintext variables, using differences between faulty and non-faulty ciphertext bits in a chosen plaintext attack scenario. Next, we show how to determine the faulty bit positions within the internal state using bit strings called difference characteristics, and how to construct such a difference characteristic for a particular faulty bit of a certain round. Using the cube method, we also determine the most effective rounds in which faults should be induced for all three variants of KATAN. Our fault attack on KATAN32 requires 2^{59} computations and turns out to need about 2^8 more (off-line) operations than the previous side channel attack by Bard et al. [18] which requires 2^{51} computations; nevertheless, our attack model (namely, assuming that an adversary can induce a fault at a random bit position in the internal state and then observes the associated ciphertext) is essentially *different* from (and arguably sounds more practical than) the attack model used by Bard et al. [18] in which they assumed that adversary can obtain (read) the "exact value" of a bit in the internal state. Furthermore, our attack is directly adapted to the cases of KATAN48 and KATAN64 (both requiring 2^{55} computations) and, so far, is the only attack against the latter variants of KATAN in the side channel attack model.

5.2 A Background on Fault Analysis

The study of faults in electronic systems began in 1970s when radioactive particles were found to cause errors in electronic chips. As a typical example, [94] showed that presence of radioactive particles in packaging material had caused faults in electronic chips due to flipped bits. This discovery subsequently paved the way for studying the effect of cosmic rays on semiconductors. Since cosmic rays are very weak at ground level and becoming more intense at higher levels (where the atmosphere layer becomes thinner or at the beginning of outer space), there was

a concern about the effect of cosmic rays on semiconductors in airborne electronic systems particularly in the airplane and aerospace industries [139].

Many efforts have been carried out to ensure protecting electronic systems from errors caused by environmental and implementation factors. Among the methods that have been widely used is through simulation for modeling electronic circuits and testing, which involve random fault induction to the electronic systems and investigating their effects. A testing method based on fault induction is largely used by IC manufacturers, Quality Assurance (QA) laboratories and the military as a part of normal manufacturing and testing procedure. The use of fault induction by these entities is for verifying the ability of certain functions to operate within a wide range of operational parameters such as temperature, voltage, shock, vibration, clock speed, electromagnetic fields, etc.

The fault induction can also be misused to compromise the security of the cryptographic modules. This is based on the fact that in the real world, most cryptographic modules have to be implemented either in software or hardware. Therefore, the security of a cryptographic module does not only depend on mathematical properties of its underlying algorithms but also depends on the physical environment in which it is being implemented. Since many cryptographic algorithms aim at being implemented in a wide range of hardware/software platforms, more opportunities are provided for an adversary to break them in real-world conditions beyond purely mathematical cryptanalytic methods. The exploitation of faults to compromise the implementation of a cryptographic module is known as *fault analysis attacks* or simply *fault attacks*.

5.2.1 Fault Models

Fault attacks have been successfully used in practice to attack systems such as pay TV cards [9]. A theoretical analysis of fault attacks against an specific cryptosystem requires a *fault model* to be first formalized to describe how faults are assumed to be induced in practice. For an attack to be finally implementable in the real world, several engineering and practical aspects should also be considered [137] to avoid unrealistic attack models and assumptions.

A fault model should specify how an adversary can choose the (exact) time and location for fault induction during the execution of a cryptographic module in the target device. For example, to be able to a induce fault at an exact time, side-channel

information such as power consumption traces or number of instances of clocking can be used to monitor the progress of the cryptographic algorithm execution in the device. However, determining and imposing the exact location in which a fault should be induced sounds a more challenging task in practice. The fault model also should specify the number of bits that are affected by a single fault induction. This can be either a single bit, a bounded number of bits such as a byte or an arbitrary number of bits. The single-bit fault model assumes the most powerful adversary in this category; hence, is the hardest one to be realized in practice [137]. Nevertheless, it is a very popular and interesting model for theoretical analysis of security level of an algorithm against fault attacks, that may clarify, at least, some certificational weaknesses. The byte fault model assumes a moderately strong adversary and is more realistic compared to the single-bit fault model. However, considering modern high-end implementations such as the new smart cards, this model can also be unrealistic for many devices in practice. The arbitrary fault model assumes the weakest adversary and hence is the most realistic fault model. If there exists an attack which is successful in this fault model it will also be successful in any other fault models, but unfortunately it would need a very complicated analysis and to the best of our knowledge there is no work in the literature on how this could be done.

Finally, with regard to a fault model, the physical impact of fault induction on the target cryptographic module should also be considered, i.e. whether the fault is transient or permanent. A transient fault means that the cryptographic module can resume its normal operation after being reset. However, a permanent fault cannot be undone: that is, it may damage the cryptographic module, e.g., by cutting a wire or destroying a memory cell permanently in the device.

5.2.2 Fault Induction Methods

There are numerous ways for inducing faults in a target cryptographic module. Since this chapter only considers the theoretical analysis of fault attacks, we only provide a brief description about some of these methods in the following.

Electromagnetic Fields. The use of electromagnetic fields as a fault induction method has been well-known and established for a long time, for example see [10, 106, 88]. To induce a fault using this method, the target device is placed

close to the electromagnetic field in order to influence transistors and memory cells. However the main problem of this technique is that it is hard to target specific bit locations in the devices.

Power Spikes. Devices such as Smart Cards do not have a built-in power supply. Power is supplied by an external device (machine) called an Smart Card reader. By replacing a Smart Card reader with a laboratory equipment, tampering the power supply becomes possible. Smart Card manufacturers specify that the voltage threshold for Smart Cards can tolerate ±10% variations of the standard 5V Vcc level. Hence, if the power supplied to the card exceeds this threshold, the card may not work properly. By exploiting this threshold, short massive variations of the power, which are called 'power spikes'(or glitches), can be supplied to the card to cause errors during the computation. Errors can either be in the memory cell or in the execution of a program and can effect an arbitrary number of bits or just a single bit [88].

External Clock Variations. Besides power supply, Smart Cards also do not produce their own clock signal. The clock signal in also provided by the card readers. Hence, controlling the externally provided clock signal (either by increasing or reducing the frequency) is also possible. Similar to the power supply, most Smart Card manufacturers specify the allowed clock signal features to ensure that the card operates properly. Typically, the accepted range of the high clock signal is between 0.7 Vcc to Vcc, while the low clock signal is between 0 to 0.5 Vcc. Hence, causing variations in the external clock exceeding the accepted threshold may result in data loss or instruction misreading, as the card's microprocessor may not be given enough time to execute the current instruction before executing the next instruction.

Temperature. The manufacturers of electronic devices define the upper and lower temperature thresholds in which the device can operate properly. By tuning the devices's temperature to exceed the thresholds, faults may occur causing the device to lose its normal functions. For example, an adversary can tune the chip temperature to a value where the write operations work but the read operations don't work, or vice versa, to attack the device.

White Light. Electronic circuits are sensitive to light. Using a focused flash light

(e.g. with the help of a microscope and aluminum foil) from an ordinary camera, an adversary can set or unset a specific bit of an SRAM memory cell. This requires the chip to be unpacked to reveal the memory cells. Hence, fault induction using this method yields a very strong attack as it allows any single bit position to be targeted.

Laser. The effect of using laser beam as a fault induction method is similar to white light. A laser can be used to simulate fault induction using particle accelerators. The main advantage of laser over white light is that we can influence the directionality of the laser beam into the specific target of a small circuit area.

X-rays and Ion Beams. Many fault induction methods require unpackaging the electronic chip. Using x-rays and ion beams one can gain the advantage of avoiding unpackaging the chip.

5.3 Description of the KATAN Block Ciphers

KATAN is a family of block ciphers [51] consisting of three variants, namely: KATAN32, KATAN48 and KATAN64. Each variant accepts an 80-bit secret key and performs 254 rounds to produce a ciphertext. All variants also share the same key schedule as well as the same nonlinear functions. KATAN ciphers aim at constrained environments such as hardware implementations with limited resources (power consumption, clock frequency and gate counts). KATAN32 with block size of 32 bits is the lightest variant in the family. A 32-bit plaintext block is loaded into two registers L_1 and L_2, respectively, of length 13 and 19 bits. The bits are indexed in the right-to-left order, from 0 to 12 for L_1 (i.e. $L_1 = (L_1[12], \ldots, L_1[0])$) and from 0 to 18 for L_2 (i.e. $L_2 = (L_2[18], \ldots L_2[0])$). The least significant bit (LSB) of the plaintext block is loaded to bit 0 of register L_2 followed by the other bits until the 18-th bit, and then remaining bits are loaded into register L_1 until the most significant bit (MSB) of the plaintext is loaded into bit 12 of register L_1. One round of KATAN32 consists of shifting the register L_1 and L_2 one bit to the left, and computing two new bit values using nonlinear functions f_a and f_b, respectively. These new bits are then loaded into the LSB bits of registers L_2 and L_1, respectively. The nonlinear

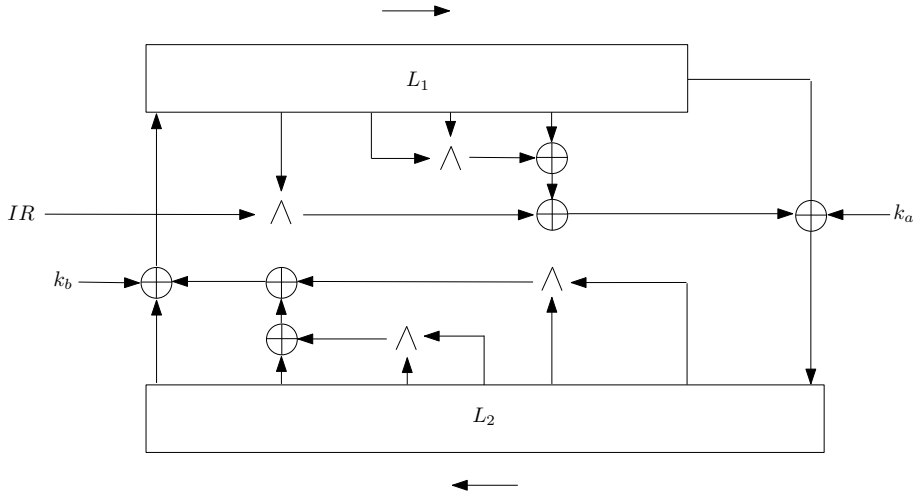


Figure 5.1: The Outline of the KATAN Family of Block Ciphers

Table 5.1: Irregular update rule used in the KATAN ciphers.

Rounds	0-9	10-19	20-29	30-39	40-49	50-59
Irregular	1111111000	1101010101	1110110011	0010100100	0100011000	1111000010
Rounds	60-69	70-79	80-89	90-99	100-109	110-119
Irregular	0001010000	0111110011	1111010100	0101010011	0000110011	1011111011
Rounds	120-129	130-139	140-149	150-159	160-169	170-179
Irregular	1010010101	1010011100	1101100010	1110110111	1001011011	0101110010
Rounds	180-189	190-199	200-209	210-219	220-229	230-239
Irregular	0100110100	0111000100	1111010000	1110101100	0001011001	0000001101
Rounds	240-249	250-253				
Irregular	1100000001	0010				

functions f_a and f_b are defined as follows:

$$f_a(L_1) = L_1[x_1] \oplus L_1[x_2] \oplus (L_1[x_3] \cdot L_1[x_4]) \oplus (L_1[x_5] \cdot IR) \oplus k_a \quad (5.1)$$

$$f_b(L_2) = L_2[y_1] \oplus L_2[y_2] \oplus (L_2[y_3] \cdot L_2[y_4]) \oplus (L_2[y_5] \cdot L_2[y_6]) \oplus k_b \quad (5.2)$$

where IR specifies an irregular update rule (i.e. $L_1[x_5]$ is used only when $IR = 1$), and k_a and k_b are two subkey bits. The details on the irregular update rules (IRs) for each round as specified in [51] are shown in Table 5.1. The key schedule for all variants of KATAN expands an 80-bit secret key K to 508 subkey bits using the following linear mapping

$$k_i = \begin{cases} K_i, & \text{for } 0 \le i \le 79, \quad (5.3) \\ k_{i-80} + k_{i-61} + k_{i-50} + k_{i-13}, & \text{Otherwise} \quad (5.3') \end{cases}$$

Given the precomputed subkey values, the values of k_a and k_b for a particular round t are defined as k_{2t} and k_{2t+1}, respectively. Thus the subkey for round t is defined as $k_a||k_b = k_{2t}||k_{2t+1}$. The selection for tap positions, x_is $(1 \leq i \leq 5)$ and y_js $(1 \leq j \leq 6)$, and the length of registers L_1 and L_2 are defined independently for each variant as shown in Table 5.2.

Table 5.2: Parameters for the KATAN Family of Block Ciphers

Cipher	$\|L_1\|$	$\|L_2\|$	x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3	y_4	y_5	y_6
KATAN32	13	19	12	7	8	5	3	18	7	12	10	8	3
KATAN48	19	29	18	12	15	7	6	28	19	21	13	15	6
KATAN64	25	39	24	15	20	11	9	38	25	33	21	14	9

Besides the tap positions and the length of the registers, the difference between all the three variants is the number of times the nonlinear functions f_a and f_b are applied in each round using the same subkey. One round of KATAN48 is shifting the registers L_1 and L_2 two bits to the left (i.e. requires two clock cycles). In each shift within the same round, the function f_a and f_b are applied using the same subkey $k_a||k_b$. Hence, full round KATAN48 requires 508 clock cycles (i.e. 254 rounds $\times$ 2 clocks per round) to produce the ciphertext.

In contrast, one round of KATAN64 requires the registers L_1 and L_2 to be shifted three bits to the left (i.e. requires three clock cycles). Similarly, in each shift within the same round, the function f_a and f_b are applied using the same subkey $k_a||k_b$. As a result, the full round KATAN64 requires 672 clock cycles to produce the ciphertext. Figure 5.1 shows the generic structure of the KATAN family of block ciphers.

5.4 Fault Analysis of KATAN

We simulate a fault attack assuming that the adversary can cause one transient single-bit error at a time in the internal state during the encryption/decryption process. It is assumed that the adversary can choose the target round(s) in which faults should be induced, for example, based on the side channel information inferred from power consumption traces and/or the clocking sequence (e.g., this can be done by triggering a laser beam with the target number of clocks of the cryptographic module). However, it is assumed that adversary cannot influence the exact position

of the faulty bit within the internal state; he can only induce the fault randomly with the hope that it will hit the target bit positions by chance.

Using this fault model, our aim is to recover the 80-bit secret key used in KATAN. Firstly, we demonstrate a method to determine the position of the faulty bit within the internal state using *difference characteristics*. Secondly, we show how to recover a *low-degree* system of multivariate polynomial equations which are obtainable within certain rounds of the enciphering process using the difference between faulty and non-faulty ciphertexts. More precisely, we only concentrate on extracting linear and quadratic equations from the internal state and subkey bits that are easily solvable. Having a sufficient number of independent equations that are solvable, we exploit the key schedule algorithm to recover the 80-bit secret key. Finally, we identify the faulty rounds of the enciphering process that should be considered in order to efficiently implement a successful fault attack on KATAN.

Our attack on the KATAN ciphers exploits the observation that after recovering n neighboring "subkey bits", we can recover the 80-bit "secret key" with time complexity of 2^{80-n} computations. This is because the 80-bit secret key is directly loaded into an 80-bit LFSR (the key register) and the subkey bits for round $t > 79$ are computed using a linear update function and shift operations (cf. Equation 5.3 and Equation 5.3′). Therefore, at any round $t > 79$, if we can recover the value of some of the LFSR bits (or similarly the value of the subkey bits), we can guess the remaining $80 - n$ values of the LFSR internal state bits and iteratively clock the LFSR backward until round $t = 0$ to recover the secret key. Suppose the highest and the lowest index values of the subkey bits to be recovered are H and L respectively. Hence, our aim is to recover the subkey bits such that $H - L \leq 79$, as all subkey bits between these index range will be the content of the 80-bit LFSR at a particular round t (Section 5.4.1 provides more details).

5.4.1 Extracting Low Degree Polynomial Equations

The main idea behind algebraic attacks is to recover the secret key of a cipher by solving a system of multivariate polynomial equations (over the plaintext and secret key variables) that describe the cipher. Since the system of equations representing a cipher is usually of very high degree, directly solving such equations, in general, to recover the secret key, is a well-known hard problem. A well-known method to try to solve such a system is linearization, i.e. replacing a high degree monomial or

equation with a new variable. In our work we take a similar approach in which, to induce faults in a targeted round on KATAN, we redefine each bit of registers L_1 and L_2 as a new variable instead of defining each one of them as a boolean function over the plaintext and secret key variables. Similarly, for the key schedule algorithm, we also redefine each subkey bit that is generated by the key register update function as a new variable instead of considering each one of them as a boolean function over the 80 secret key variables. Consequently, the subkey bits are indexed from 0 to 507. Thus, the system of equations arising from the faulty and non-faulty ciphertext differential is in the linearized parameters.

Although the linearization method is used, considering fault induction at an early round of the enciphering process will cause the polynomials representing the ciphertext bits in the linearized parameters to become too complex to be analyzed explicitly. Hence, to avoid dealing with such a complex problem, we only represent them as black-box polynomials and extract them using the recently proposed cube [55] and extended cube methods [2] (as discussed in Chapter 3 and Chapter 4 of this thesis). This removes the need to know the explicit (enormous) symbolic representations of the related polynomials. Application of these cube based methods in our fault attack is inspired by the observation that computing ciphertext differentials (obtained by XORing non-faulty and faulty ciphertexts) in the single-bit fault model is similar to summing over the black-box master polynomial over cubes of size 1.

From the cube and extended cube methods (i.e. using cubes of size 1), we found that the subkey variables only begin to appear in quadratic polynomial equations. We utilize both linear and quadratic equations to recover n subkey bits which enable us to recover the secret key by guessing the remaining $80 - n$ bits of the key register and clocking backward until round $t = 0$, where the secret key can be found.

5.4.2 Fault Position Determination

Since faults are randomly introduced into the internal state of registers L_1 and L_2 after a certain "known" number of rounds, say t, identifying the exact faulty bit position is necessary to ensure the success of the attack. To find this exact position, we construct a *difference characteristic* for each internal state bit generalizing the idea of [77] to locate the faulty bit position in the internal state of the Trivium stream cipher. A difference characteristic corresponding to any bit position of the internal state is a string obtained by XORing the non-faulty ciphertext and the

faulty ciphertext (resulting from fault induction at the bit position). We use right-to-left ordering of bit numbers, denoting index 0 of the difference characteristic bit as the LSB of the characteristic and the highest index as the MSB. Values '0' and '1' represent difference values 0 and 1 respectively for the corresponding characteristic bits with probability 1, while the '-' sign represents unknown values (i.e. can be either '0' or '1'). The difference characteristic for each faulty bit position of a certain round can be represented by a lookup table.

Given a faulty ciphertext resulting from a random fault induction into an "unknown" internal state bit s_{j+t} after t-th round, to determine the position j, first we compute the ciphertext differential, Δc, by XORing (summing modulo 2) the non-faulty ciphertext c with the faulty ciphertext c' such that $\Delta c_j = c_j \oplus c'_j$, for $0 \leq j < |L_1| + |L_2|$. Then, guided by the lookup table, we refer to positions with values '0' and '1' (and ignore those with a '-' sign) within each characteristic and compare them with bits in the same positions in Δc. If all the corresponding bits in Δc match the bits in the characteristic of the faulty bit s_{j+t} then we can ensure that a fault has been induced into the bit at position j. However, there might be a case where there are no characteristics uniquely distinguishing two or more faulty bit positions. This may occur as the result of inducing faults in the early rounds of the enciphering (or deciphering) process. If there were only very few faulty bit positions sharing the same characteristic, we can just guess them in order to find the correct one. Having too many faulty bit positions sharing the same characteristic will cause extra overhead in the attack complexity. Thus, one should try to consider later rounds to avoid such an overhead.

Constructing Difference Characteristics.

To construct a differential characteristic, we study the error propagation in the ciphertext bits due to one faulty bit at a certain position in the internal state. Based on this we can determine which bits within the ciphertext are

- affected by the faulty bit with *probability 1*,
- not affected by the faulty bit with *probability 1*, or
- affected by the faulty bit with some *probability less than 1*.

The ciphertext bits that are certainly affected (affected with probability 1) will always give value '1' in the differential, while those that are certainly not affected

will always be '0' in the differential. However, the bits that are affected with some non-zero probability less than 1 can hold either value '0' or '1' in the differential (i.e. an unknown value) and are denoted by '-' in the difference characteristic.

To know how a faulty bit after round t affects bits in the ciphertext, we consider each ciphertext bit as a boolean function in the linearized variables after round t. Then we analyze how the faulty bit can appear as a parameter in the boolean function. There are three ways that a faulty bit can appear as a parameter within the polynomial describing the the boolean function, namely it either

- exists as a term but not as a subterm of some monomials in the polynomial (i.e. appears as a linear variable),
- does not exist within the polynomial, or
- exists as a subterm of some monomials (and probably also as a term) within the polynomial.

To identify which of the above three cases occurs, we utilize the method used in the cube attack as described in [55]. We select the faulty bit as the monomial, t_I, and apply the linearity test on the corresponding superpoly, $p_{S(I)}$, to determine whether the test will result constant 0, constant 1, linear or higher degree superpoly. Constant 0 and constant 1 superpolys indicate values '0' and '1' in the difference characteristic bits, respectively. However linear and higher degree superpolys indicate unknown values in the characteristic bits, i.e. the '-' sign.

5.4.3 Finding Effective Rounds for Fault Induction

To recover most of the subkey bits, the rounds which contain a high number of quadratic equations (resulting from non-faulty and faulty ciphertext differential) need to be determined as the subkey bits only begin to appear within these equations. We analyze the distribution of the linear and quadratic equations after each round, obtainable from non-faulty and faulty ciphertext differentials, by considering every bit (one bit at a time) of the internal state (i.e. from register L_1 and L_2) being induced by a fault value. We apply the cube and extended cube methods considering cubes of size 1 to simulate the non-faulty and faulty ciphertext differentials due to one faulty bit for each of the internal state bits and after each round. For each linear and quadratic equations found, we accumulate the total number of such equations

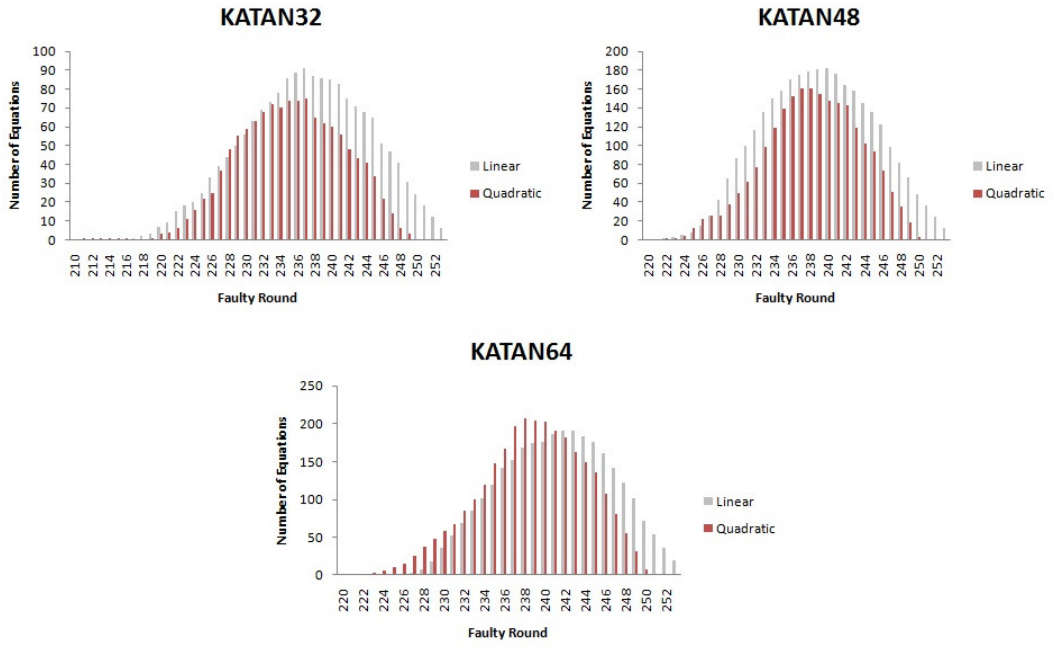


Figure 5.2: Distribution of Linear and Quadratic Equations in KATAN

for each round. Fig. 5.2 shows the result of our analysis. In the figure, "Faulty Round" denotes the number of rounds that the cipher has run before inducing a fault into the internal state. It is obvious that the ranges of faulty rounds which can provide a high number of quadratic equations for all variants of KATAN are about the same. We call such Faulty Round numbers *effective rounds* for a particular variant. Thus, the fault attack on KATAN will be possible if faults are induced into the internal state within these specific effective rounds.

5.4.4 Practicality of Fault Attacks on KATAN

Recall that in our fault model the adversary has the ability to influence an exact faulty round, for example, by triggering a laser beam from the number of clocks the cipher has executed. The adversary however cannot influence the exact faulty bit from the fault induction. All he can do is to induce a single-bit fault as many times as he wants and each time only at a "*random position*".

Considering a real world implementation of a single-bit fault model, the adversary will never induce fault randomly into the internal state each time. The reason for

this is that, although at the beginning the position of each internal state bit is unknown, via fault induction some clue can be obtained about the position of the target bits. For example, suppose that the index of one of the target bits is j. Knowing that a random fault induction has affected the bit with index $j-\alpha$ (which can be verified from its difference characteristic) gives us a clue that by pointing the fault induction device (e.g. the laser beam) slightly to the left or to the right we may also hit the target bit with index j. In this way, the adversary may increase the success probability in targeting specific bits. This is because the internal state bits are not scattered in random positions in a real-world device, instead they are often ordered in a sequence, such as, bit indexed j is adjacent to bit indexed $j+1$, and so on.

Also, suppose that the same bits are targeted in more than one faulty round. If we are able to hit a bit by a fault induction at one faulty round, without changing the position of the induction point, we can further clock the cipher to reach another target round that requires the same target bit to be faulty. Reaching another target round, the same bit position can be hit successfully by a fault induction with probability 1. In this way, we can reduce the hassle and increase the success probability for fault inductions. Generalizing this idea, since we are targeting four different faulty rounds for each cipher (more explanation about this is given later in Section 5.4.5, 5.4.6 and 5.4.7), we can target the target internal state bits of the four different rounds at the same time instead of considering them separately. This can significantly reduce the number of fault inductions required for the whole attack to be successful. More precisely, each time a random induction hits an internal state bit, the adversary will identify all the target faulty rounds which require the bit to be faulty. Then without moving the induction point the adversary clocks the cipher to reach those target rounds and for each round a fault is induced in the bit position. Of course this requires the adversary to reset the cipher each time he aims for different rounds. We call this as *overlapping* method.

Note that, the structure of KATAN ciphers allows the adversary to induce fault at some of the target bits "*quite easily*". The use of FSR as the underlying structure in KATAN allows a random fault induction to hit the target bits successfully using a *sliding fault induction* method. Suppose the adversary aims at the bit indexed j after round t of KATAN32 to be induced by a faulty value. If by random induction the adversary is only able to hit, for example, the bit indexed $j-\alpha$, instead of trying to move the induction point towards the target bit (as mentioned previously), without

moving the induction point the adversary can reset the cipher and consider round $t-\alpha$ and induce the fault at the same bit indexed $j-\alpha$ and continue the execution of the cipher to completion. The result of fault induction at bit $j-\alpha$ after round $t-\alpha$ will be the same as the result of fault induction at bit j after round t. However this requires two conditions to be satisfied as follows:

1. There should be no taps between the bit indexed j and the bit indexed $j-\alpha$. However, having a tap on the bit indexed j is acceptable but there should be no tap on the bit indexed $j-\alpha$.

2. Random fault induction should only hit bits with index $\leq j$ for encryption, as encryption in KATAN requires internal state bits to be shifted towards the MSB of the corresponding registers (i.e. L_1 and L_2).

The sliding method can be directly applied on KATAN48 and KATAN64 variants which differ in the number of clocks for each round. We expect that this sliding fault induction can be generalized to consider the existence of taps in between the two bits j and $j-\alpha$. We leave this generalization on KATAN as an open problem for future research.

5.4.5 Attack on KATAN32

The fault attack can be efficiently applied if the rounds that have high number of quadratic equations (resulting from non-faulty and faulty ciphertext differentials) are considered. As for KATAN32, this refers to the fault inductions after $t = 237$ rounds as shown in Fig. 5.2. Considering this faulty round, we provide a sample set of linear and quadratic equations that can help in recovering the target subkey bits as shown in Table B.2 in Appendix B.

In the table, $L_2 = (s_{18+t}, \ldots, s_{0+t})$ and $L_1 = (s_{31+t}, \ldots, s_{19+t})$. Δc_j denotes a ciphertext bit difference where the difference is obtained by XORing the non-faulty ciphertext bit c_j with the faulty ciphertext bit c'_j, i.e. $\Delta c_j = c_j \oplus c'_j$, for $0 \leq j \leq 31$. For subkey bits we use a slightly different notation to facilitate our analysis, in which we denote the k_is as subkey bits whose indexes ranging from $0 \leq i \leq 507$ (in which bits indexed 0 until 79 are from the original secret key bits). We do not consider each subkey bit indexed $i > 79$ as a boolean function over the 80 secret key bits. Instead, to facilitate our analysis we only consider each one of them as an independent new variable.

Considering fault induction after $t = 237$ rounds, 10 subkey bits can be found within the quadratic equations, i.e. $k_{474}, \ldots, k_{482}$ and k_{484} (cf. Table B.2 for the polynomial equations and Table B.6 for the difference characteristics in Appendix B). Recovering these subkey bits, requires solving the corresponding quadratic equations in which some of the linear equations listed in the table should also be involved, as they can provide the solution for the internal state bits of registers L_1 and L_2 within the quadratic equations. For example, to find the solution for k_{474}, we consider s_{1+t} as the faulty bit after $t = 237$ rounds. Considering the difference between non-faulty and faulty ciphertext bit c_{24}, i.e. Δc_{24}, the symbolic representation of the differential is

$$s_{22+t} + s_{26+t} + s_{31+t} + k_{474} + s_{24+t}s_{27+t} = \Delta c_{24}.$$

The value of the right hand side (RHS) of this equation (either 0 or 1) can be determined by numerically computing Δc_{24}, such that $\Delta c_{24} = c_{24} \oplus c'_{24}$. Then recovering k_{474}, requires the bits $s_{22+t}, s_{26+t}, s_{31+t}, s_{24+t}$ and s_{27+t} to be known. With the exception of bit s_{31+t}, all these bits can be recovered directly by utilizing the linear equations available within the faulty round. However, observation of Table B.2 shows that no solution can be found for s_{31+t} which prevents us from finding the solution for k_{474}. Applying the same approach to recover $k_{475}, k_{476}, k_{477}, k_{478}, k_{479}, k_{480}, k_{481}, k_{482}$ and k_{484}, we encounter the same problem, in which we are unable to recover those subkey bits, except for subkey bit k_{484}. This is because no solutions can be found for $s_{18+t}, s_{30+t}, s_{17+t}, s_{29+t}, s_{16+t}, s_{28+t}$ and s_{15+t} to make the system of equations in the subkey and internal state bits solvable. However, knowing k_{484} enables us to reduce the key space by 50 percent (more explanation about this in Section 5.4.8).

Next, we further reduce the complexity of our attack by recovering the unknown bits s_{31+t}, s_{18+t}, s_{30+t}, s_{17+t}, s_{29+t}, s_{16+t}, s_{28+t} and s_{15+t} to find the solutions for the remaining 9 subkey bits. We exploit the properties of registers L_1 and L_2 to recover those bits. It is obvious that all bits in L_1 and L_2 except s_{0+t} and s_{19+t}, are updated each round only by a shift operation (i.e. bit indexed j is moved to bit indexed $j+1$). Thus we can search for equivalent bits in any other rounds (i.e. other than $t = 237$) to recover the unknown bits from the original faulty round. Now, we consider round $t = 231$ as another faulty round (cf. Table B.1 for the polynomial equations and Table B.5 for the difference characteristics in Appendix B). Since $t = 231$ is 6 rounds earlier than $t = 237$, bit s_{j+t} at faulty round $t = 237$ can be considered similarly to bit s_{j-6+t} at faulty round $t = 231$. This implies bits s_{31+t},

s_{18+t}, s_{30+t}, s_{17+t}, s_{29+t}, s_{16+t}, s_{28+t} and s_{15+t} at round $t = 237$ are similar to bits s_{25+t}, s_{12+t}, s_{24+t}, s_{11+t}, s_{23+t}, s_{10+t}, s_{22+t} and s_{9+t} at round $t = 231$ respectively being shifted 6 clocks towards the MSB. Hence, considering faulty round $t = 231$, the bits s_{25+t}, s_{12+t}, s_{24+t}, s_{11+t}, s_{23+t}, s_{10+t}, s_{22+t} and s_{9+t} can be recovered directly which enables the remaining 9 subkey bits at faulty round $t = 237$ to be recovered too.

Note that choosing the faulty round $t = 231$ enables another 10 subkey bits which do not exist in faulty round $t = 237$ to be found, i.e. bits $k_{462}, \ldots, k_{470}$ and bit k_{472}. Again, we cannot recover all the 10 subkey bits directly as the internal state bits s_{26+t}, s_{31+t}, s_{27+t}, s_{18+t}, s_{30+t}, s_{17+t}, s_{29+t}, s_{16+t}, s_{28+t} and s_{15+t} that can help in finding the solutions for the subkey bits are unknown. Thus we repeat the process as described before to recover all the 10 subkey bits by further considering earlier rounds. However considering another earlier round will result in having difficulty in determining the faulty bit position within registers L_1 and L_2. This is because the difference characteristics that can uniquely define the faulty bit position are reducing as we consider more earlier rounds. Consequently, there exist many possible bit positions that have the same difference characteristic. Hence, to locate the exact faulty bit position, guessing the bit positions among those with similar difference characteristic is required. However, this will introduce an overhead in the attack, and hence we don't consider recovering the 10 subkey bits within this faulty round, $t = 231$ (i.e. bits $k_{462}, \ldots, k_{470}$ and bit k_{472}).

To avoid too much guessing, one should try inducing faults at round $t > 237$ as this can provide a more defined difference characteristic. We consider faulty round $t = 243$ as our next target round (cf. Table B.3 for the polynomial equations and Table B.7 for the difference characteristics in Appendix B). Considering this round, we find another 10 new subkey bits within the quadratic equations in which 2 of them can be directly recovered, i.e. k_{494} and k_{496}, while the other 8 subkey bits, i.e. $k_{486}, \ldots, k_{493}$, cannot be recovered unless the internal state bits s_{31+t}, s_{18+t}, s_{30+t}, s_{17+t}, s_{29+t}, s_{16+t}, s_{28+t} and s_{15+t} are known. However, these bits can be recovered by referring directly to bits s_{25+t}, s_{12+t}, s_{24+t}, s_{11+t}, s_{23+t}, s_{10+t}, s_{22+t} and s_{9+t} respectively which can be directly solved at faulty round $t = 237$. Thus, all the 10 subkey bits can also be recovered in this round.

Tips for optimization: Note that the value of right-hand side of bits s_{9+t}, s_{10+t}, s_{7+t}, s_{12+t} and s_{8+t} of round $t = 243$ can be recovered from the equations which are

available in this round (but are not shown in the table). To avoid additional fault inductions to be performed, the adversary can obtain the value of the right-hand side from faulty bits s_{3+t}, s_{10+t}, s_{7+t}, s_{12+t} and s_{8+t} respectively of round $t = 237$. Recovering the bits from those at faulty round $t = 237$ however does not require additional fault induction as they are also used by other equations in that round.

Finally, we consider faulty round $t = 249$ as our last target round, as no other rounds can provide subkey bits within the differential equations. In this final round, there is only one subkey bit which can be obtained from the equations, i.e. k_{498}. The equations and difference characteristics are shown in Table B.4 and Table B.8 respectively in Appendix B. It is obvious s_{31+t} is unknown. However it can be recovered by referring to s_{25+t} at $t = 243$.

5.4.6 Attack on KATAN48

Following the method used on KATAN32, we consider the KATAN48 block cipher as our next target. Recall that the main differences between KATAN32 and KATAN48 are the block size, the tap positions in the internal state L_1 and L_2, and the number of clocks for each round. For KATAN48 we have $L_2 = (s_{28+t}, \ldots, s_{0+t})$ and $L_1 = (s_{47+t}, \ldots, s_{29+t})$. Since KATAN48 requires two clocks for each round, if a certain internal state bit s_{j+t} cannot be solved directly in certain faulty round t, then its solution may be found by referring to bit s_{j-2n+t} in an earlier faulty round $t - n$, for $j - 2n \geq 29$ and $31 \leq j \leq 47$, or $j - 2n \geq 0$ and $2 \leq j \leq 28$.

Our attack on KATAN48 considers faulty rounds $t = 234, 238, 242, 246, 250$ as the target rounds. Similarly to KATAN32, the selection of these rounds is based on the number of quadratic equations that can be found within the effective rounds. Fig. 5.2 shows that the highest number of quadratic equations for KATAN48 can be found at faulty rounds $t = 237$ and $t = 238$. Since the difference characteristics are more clearly defined when we consider later rounds: thus we choose $t = 238$ rather than $t = 237$ as our first target round.

Considering faulty round $t = 238$, we have been able to find 8 subkey bits within the quadratic equations, i.e. $k_{476}, \ldots, k_{483}$ (cf. Table C.2 for the polynomial equations and Table C.7 for the difference characteristics in Appendix C). The same problem as in KATAN32 occurs where some of the internal state bits cannot be solved directly from this round which prevents the 8 subkey bits being solved. These bits are s_{22+t}, s_{23+t}, s_{25+t}, s_{27+t}, s_{45+t} and s_{46+t}. Finding the solution for each of these bits requires

us to consider 4 earlier rounds, i.e. faulty round $t = 234$. At $t = 234$ (cf. Table C.1 for the polynomial equations and Table C.6 for the difference characteristics in Appendix C), these bits are equal to bits s_{14+t}, s_{15+t}, s_{17+t}, s_{19+t}, s_{37+t} and s_{38+t} respectively.

Note that, we don't consider recovering the subkey bits at faulty round $t = 234$, to avoid the possibility of needing to perform too much guessing for the exact faulty bit positions in the earlier faulty round $t = 230$ (for recovering the unknown internal state bits) resulting from lack of clearly defined difference characteristics. Hence, we consider faulty round $t = 242$ instead as our next target round. Considering this faulty round gives us another 8 subkey bits within the quadratic equations, i.e. $k_{484}, \ldots, k_{491}$ (cf. Table C.3 for the polynomial equations and Table C.8 for the difference characteristics in Appendix C). There are only 5 internal state bits which cannot be solved when considering the system of equations within this faulty round. These are bits s_{22+t}, s_{23+t}, s_{25+t}, s_{27+t} and s_{46+t}. However, each of these bits holds the same value as bits s_{14+t}, s_{15+t}, s_{17+t}, s_{19+t} and s_{38+t} respectively at faulty round $t = 238$. Since the relevant bits can be solved, thus their values obtained enables the 8 subkey bits to be solved.

Tips for optimization: To reduce the number of fault inductions, one can recover the value of right-hand side of bits s_{12+t}, s_{13+t}, s_{16+t}, s_{17+t}, s_{18+t}, s_{37+t} and s_{40+t} of faulty found $t = 242$ by referring to bits s_{4+t}, s_{5+t}, s_{8+t}, s_{9+t}, s_{10+t},s_{29+t} and s_{32+t} respectively of round $t = 238$.

Another 8 subkey bits can also be found within quadratic equations in faulty round $t = 246$, i.e. $k_{492}, \ldots, k_{499}$ (cf. Table C.4 for the polynomial equations and Table C.8 for the difference characteristics in Appendix C). To recover the value of these subkey bits requires 5 internal state bits, i.e. s_{22+t}, s_{23+t}, s_{25+t}, s_{28+t} and s_{47+t}, to be solved by referring to the equivalent bits at faulty round $t = 242$ which are s_{14+t}, s_{15+t}, s_{17+t}, s_{19+t} and s_{38+t} respectively. This in turn can help to provide the solution for the 8 subkey bits.

Tips for optimization: Considering faulty round $t = 246$, bits s_{9+t}, s_{8+t}, s_{15+t} and s_{42+t} can be recovered by referring to bits s_{1+t}, s_{0+t}, s_{7+t} and s_{34+t} respectively of faulty round $t = 242$. However bit s_{19+t} of faulty round $t = 246$ can be recovered by referring to bit s_{3+t} of faulty round $t = 238$.

Finally we consider faulty round $t = 250$ as the last target round as no more later rounds can supply subkey bits within quadratic equations. Considering this round,

only one subkey bit can be found within the quadratic equations, i.e. k_{500} (cf. Table C.5 for the polynomial equations and Table C.10 for the difference characteristics in Appendix C). There is only one internal state bit, i.e. bit s_{47+t}, that is required to find the solution for the subkey bit which cannot be solved directly in this round. However this bit is equal to bit s_{39+t} in faulty round $t = 246$. Knowing the value of this bit enables the subkey bit to be recovered successfully.

TIPS FOR OPTIMIZATION: For faulty round $t = 250$, bits s_{41+t} and s_{44+t} can be recovered by referring to bits s_{33+t} and s_{36+t} respectively of faulty round $t = 246$.

5.4.7 Attack on KATAN64

Now we consider a fault attack on the third variant of the KATAN block cipher, namely, KATAN64. In KATAN64 we have $L_2 = (s_{38+t}, \ldots, s_{0+t})$ and $L_1 = (s_{63+t}, \ldots, s_{39+t})$. Since each round in KATAN64 requires 3 clocks, if certain internal state bits s_{j+t} cannot be recovered at faulty round t, we can try to recover their values from bit s_{j-3n+t} of faulty round $t - n$, for $j - 3n \geq 39$ and $42 \leq j \leq 63$, or $j - 3n \geq 0$ and $3 \leq j \leq 38$.

As in the attack on the previous two variants, we concentrate on the round which can provide the highest number of quadratic equations from the differential. According to Fig. 5.2, this refers to faulty round $t = 238$. There are 8 subkey bits can be found within the quadratic equations at this round, i.e. $k_{476}, \ldots, k_{483}$ (cf. Table D.2 for the polynomial equations and Table D.7 for the difference characteristics in Appendix D). However 7 internal state bits are unknown to help recover those subkey bits, i.e. s_{61+t}, s_{51+t}, s_{60+t}, s_{25+t}, s_{38+t}, s_{34+t} and s_{24+t}, but these bits are equal to bits s_{55+t}, s_{45+t}, s_{54}, s_{19+t}, s_{42+t}, s_{28+t} and s_{18+t} at faulty round $t = 236$ (cf. Table D.1 for the polynomial equations and Table D.6 for the difference characteristics in Appendix D). However s_{55+t} can be recovered from bit s_{3+t} and equation $s_{3+t} + s_{55+t}$ within the same faulty round $t = 236$; the bit s_{45+t} can be recovered from bit s_{2+t} and equation $s_{2+t} + s_{45+t}$; the bit s_{54+t} from s_{2+t} and $s_{2+t} + s_{54+t}$; and bit s_{32+t} can be recovered from bit s_{42+t} and equation $s_{32+t} + s_{42+t}$. However bits s_{25+t}, s_{34+t} and s_{24+t} (in faulty round $t = 238$) can be recovered by directly referring to the corresponding bits s_{19+t}, s_{28+t} and s_{18+t} in faulty round $t = 236$ respectively.

Considering faulty round $t = 242$, another 8 subkey bits can be found (cf. Table D.3 for the polynomial equations and Table D.8 for the difference characteristics in Appendix D), i.e. bits $k_{484}, \ldots, k_{491}$ with 6 unknown internal state bits, i.e. s_{62+t},

s_{23+t}, s_{29+t}, s_{32+t}, s_{27+t} and s_{24+t}, which need to be solved by referring to the equivalent bits in faulty round $t = 238$, i.e. bits are s_{50+t}, s_{11+t}, s_{17+t}, s_{20+t}, s_{15+t} and s_{12+t} respectively; while 2 unknown internal state bits, i.e. s_{36+t} and s_{34+t} need to be solved by referring to the equivalent bits in faulty round $t = 236$, i.e. s_{18+t} and s_{16+t} respectively.

Tips for optimization: Bits s_{21+t}, s_{52+t}, s_{53+t} and s_{58+t} of faulty round $t = 242$ can be recovered by referring to bits s_{9+t}, s_{40+t}, s_{41+t} and s_{46+t} of round $t = 238$ to reduce slightly reduce the number of fault inductions.

Next, we consider faulty round $t = 246$ to find 8 more subkey bits, namely, $k_{492}, \ldots, k_{499}$ (cf. Table D.4 for the polynomial equations and Table D.9 for the difference characteristics in Appendix D). However bits s_{63+t} and s_{36+t} are unknown, preventing the recovery of k_{492} and k_{493} respectively, if only $t = 246$ is considered. These unknown bits can be recovered by referring to the equivalent bits, i.e. s_{39+t} and s_{12+t} respectively in faulty round $t = 238$.

Tips for optimization: Considering faulty round $t = 246$, some improvement in the reduction of the number of fault inductions can be performed by recovering bits s_{12+t}, s_{19+t}, s_{20+t}, s_{24+t}, s_{29+t} and s_{55+t} from bits s_{0+t}, s_{7+t}, s_{8+t}, s_{12+t}, s_{17+t} and s_{43+t} of faulty round $t = 242$.

Finally, we consider faulty round $t = 250$ as the last target round as no more subkeys can be recovered from later rounds. In this round, only one subkey bit (i.e. $t = 250$) can be found (cf. Table D.5 for the polynomial equations and Table D.10 for the difference characteristics in Appendix D). However the internal state bit, s_{62} needs to be recovered by referring to bit s_{50+t} in faulty round $t = 246$ before k_{500} can be solved.

Tips for optimization: One can recover bits s_{53+t} and s_{58+t} of faulty round $t = 250$ by referring to bits s_{41+t} and s_{46+t} of faulty round $t = 246$ to reduce the number of inductions.

5.4.8 Analysis of the Attack Complexity

First we would like to emphasize that the linear and quadratic equations shown in the tables in appendices B, C and D are only a sample set of equations that can provide a solution to recover the subkey bits. In fact, not only they can be extracted

from the faulty bits shown in the tables, but also some of them can be extracted from other faulty bits (within the same faulty round) not listed in the tables.

Instead of using the set of equations shown in the appendices to recover the subkey bits, the adversary can also use different set of equations which are also solvable. Hence, recovering the subkey bits is not restricted to using only the sample set of equations shown in the appendices. That is, recovering the subkey bits may still be possible even if the adversary cannot use exactly the same set of equations as shown in the appendices.

Note also that, the adversary can also recover the target subkey bits by using equations available from different faulty rounds other than the one described earlier and shown in the appendices. Considering other faulty rounds, the adversary may find equations in which the same subkey variables appear.

Suppose the adversary aims at inducing fault at the bit positions listed in the appendices B, C and D. Thus the attack success depends on the adversary's ability to induce faults into the target faulty bits. Having n target internal state bits for fault induction, the probability of being able to cause one bit error to any one of these n target bits resulting from a random fault induction is $\frac{n}{|L_1|+|L_2|}$. If the adversary had to hit all the remaining target bits at the same round then he would have to continue injecting faults to aim any one of the remaining $n-1$ target bits with probability $\frac{n-1}{|L_1|+|L_2|}$, any one of the remaining $n-2$ target bits with probability $\frac{n-2}{|L_1|+|L_2|}$, and so forth until hitting the last remaining target bit with probability $\frac{1}{|L_1|+|L_2|}$. Thus, in this worst case scenario (i.e., being restricted to only one faulty round) the success probability of causing errors into n target internal state bits would be $(\frac{n}{|L_1|+|L_2|})(\frac{n-1}{|L_1|+|L_2|})\ldots(\frac{1}{|L_1|+|L_2|}) = \prod_{i=1}^{n} \frac{i}{|L_1|+|L_2|} = \frac{n!}{(|L_1|+|L_2|)^n}$ with $n \leq |L_1|+|L_2|$.

Result on KATAN32

Our experimental simulation of the attack on KATAN32 shows that 21 subkey bits from faulty rounds $t = 231, 237, 243, 249$ can be recovered, requiring collectively 20 specific internal state bit positions (regardless of the round number) to be considered as target faulty bits, as shown in Table B.1, B.2, B.3 and B.4 in Appendix B. The average number of fault injections needed to successfully hit these 20 target faulty bits is 115 (where the average is taken over 10,000 trials).

Since the highest index of the subkey bits is $H = 498$ and the lowest index is $L = 474$ (hence $H - L = 24 < 80$) the target subkey bits can be found in the 80-bit

key register within rounds $209 \leq t \leq 236$. Therefore, to recover the secret key, we need to guess the remaining 59 bits of the key register and then to clock the key register backward until round $t = 0$. This reduces the complexity of the attack to 2^{59} computations compared to 2^{80} by exhaustive key search.

Result on KATAN48

The attack on KATAN48 results in recovering 25 subkey bits considering faulty rounds $t = 234, 238, 242, 246, 250$ which requires collectively 27 specific internal state bits positions to be considered as target faulty bits, as shown in Table C.1, C.2, C.3, C.4, C.5 in Appendix C. The average number of required fault injections to successfully hit these 27 target faulty bits is 211 (where the average is taken over 10,000 trials).

The highest and the lowest subkey bit indexes are $H = 500$ and $L = 476$, respectively (hence $H - L = 24 < 80$), so all the subkey bits can be found within the content of the 80-bit key register at rounds $210 \leq t \leq 237$. Therefore, to recover the secret key we need to guess the remaining 55 bits of the key register and then to clock backward until round $t = 0$ to recover the secret key. Thus, finding the correct key requires 2^{55} computations in this attack.

Result on KATAN64

In attacking KATAN64 we consider faulty rounds $t = 236, 238, 242, 246, 250$ to recover (at least) the same 25 subkey bits as in the attack on KATAN48 which requires collectively 44 specific internal state bit positions to be faulty as shown in Table D.1, D.2, D.3, D.4, D.5 in Appendix D. The average number of required fault injections to successfully hit these 44 target faulty bits is 278 (where the average is taken over 10,000 trials). This results in an attack with complexity 2^{55} (Noticing that the highest index of the subkey bits is $H = 491$ and the lowest index is $L = 476$ (i.e. $H - L = 15 < 80$); hence, these 25 target subkey bits can be found in the 80-bit secret key register and we only need to guess the remaining 55 bits of the key register).

5.5 Summary

In this chapter, we showed fault attacks using a single-bit transient fault model against all the three variants of KATAN; namely, KATAN32, KATAN48 and KATAN64. Our fault attacks on the KATAN ciphers assume the adversary can influence the target rounds to perform fault induction, but cannot influence the exact location of the target bits within the internal state. Thus, fault induction can only be performed within the target rounds at random positions with the hope that it will hit the target bits.

To identify the bit position which has been affected by a fault induction, we constructed a lookup table in which the entries are the difference characteristics for all internal state bit of a given faulty round. Since the aim of our fault attack is to recover as many subkey bits as possible, which subsequently helps recover the secret key, we constructed the symbolic representation (in the form of linear and quadratic equations) of the faulty and non-faulty ciphertext differentials using the cube and extended cube methods. Having a sufficient number of linear and quadratic equations that are solvable, we can find the value of each subkey bit using the well-known Gaussian elimination method. Using the cube method, we also determined the rounds in which most of the linear and quadratic equations can be found most efficiently.

Our fault attack on KATAN32 requires 2^{59} computations and turns out to be less efficient than the previous side channel attack of [18], that needs 2^{51} computations. Nevertheless, the physical model underlying our attack (that is, assuming that adversary can inject a fault at a random bit position in the internal state and then only needs to get the ciphertexts) is essentially different from that of [18], in which one assumes that adversary can obtain (read) the "exact value" of a bit in the internal state. Furthermore, our attack is directly adapted to the cases of KATAN48 and KATAN64 and both require 2^{55} computations, and so far, is the only attack against the later variants of KATAN in the side channel attack model.

Chapter 6

Conclusion and Scope of Further Research

In this thesis we have presented several contributions to algebraic and side-channel cryptanalysis of lightweight block ciphers. The presented attacks are chosen plaintext attacks and adversary does not need to consider plaintexts longer than one block (i.e., the attacks do not depend on any mode of operation for the target block ciphers).

In Chapter 3, as our main contribution, we analysed the security of a hardware oriented block cipher called NOEKEON. Our analysis on NOEKEON adopts the method introduced by Dinur and Shamir in [54, 56], which combines a cube attack with a side channel attack. Assuming a single-bit leakage of the internal state as the side channel information after each round, we applied the cube attack to obtain sufficient number of independent linear equations over the secret key variables. In order to efficiently apply the side channel cube attack on NOEKEON, the round in which the single-bit leakage model can be efficiently applied is determined. Such efficient round is required as the degree of the polynomial describing the block cipher increases with the number of rounds. However, considering rounds too early, one may find only a few secret key bits as the result of the incomplete diffusion process. Therefore, to find the efficient round, first we examined the evolution of the degree of the polynomials by studying the nonlinear function of the cipher. Second, we studied the round in which the cipher begins achieving complete diffusion, by counting the number of secret variables which exist in each of the internal state bits after each round. Assuming a single-bit leakage on the first bit (i.e. the MSB) of the internal state, we have been able to show that NOEKEON is susceptible to side channel cube attack, such that, one can recover about half of the secret key directly by solving the system of equations. To the best of our knowledge, when this thesis was being written, this was the only known attack on NOEKEON in the side channel model.

As an interesting question for future research in this line, we suggest studying the attack complexity by considering other internal state bits used as the targets in the single-bit leakage model. Also, considering other leakage models such as the Hamming weight leakage model could be an interesting direction to recover more secret key bits.

In Chapter 4, our main contribution is twofold. First, we introduced an efficient method, which we call "extended cube method", to derive nonlinear equations of low degree. The main idea of the extended cube method is to consider secret variables as part of the boolean cube during the preprocessing phase of the cube attack. The sub-monomial of the cube which only consists of secret variables is called the extending cube of the sub-monomial of the cube consisting only of public variables. In other words, an extended cube is the whole boolean cube which consists of both public and secret variables. To derive low degree nonlinear equations efficiently using this extended cube method, we introduced two algorithms to be used together. The first algorithm (Algorithm 1) used to find secret variables within the superpoly equations and the second algorithm (Algorithm 2) used to derive low degree nonlinear equations over the secret variables found using Algorithm 1.

As our second contribution in this chapter, we applied our extended cube method on two variants of the PRESENT block cipher, namely, PRESENT-80 and PRESENT-128. PRESENT is a block cipher aimed at environments with limited resources (such as RFIDs). Our analysis of PRESENT combines our extended cube method with a side channel attack and considers the Hamming weight of the internal state as the leakage model. Considering the Hamming weight leakage model helps to avoid considering more rounds (which in turn implies dealing with higher degree polynomials) to obtain the rounds in which complete diffusion is achieved. As a result, only after the first round we have been able to recover all the 64 secret key variables being introduced as the round key for both variants of PRESENT by solving the derived polynomial equations (i.e. linear and quadratic equations). Our attack on PRESENT-80 has improved the previous attack by Yang et al. [136]. However, to the best of our knowledge our attack on the PRESENT-128 variant is currently the only known attack against the cipher in the side channel attack model.

As an interesting problem for future research, we suggest studying application of our extended cube method to other ciphers. For example, one should try to apply our extended cube method in the standard attack model on the Trivium stream cipher improving the previous cube attack (which is known to be the best

attack on Trivium) by Dinur and Shamir. Another possible direction is to combine more leakage information from several other rounds of PRESENT using the same Hamming weight leakage side channel attack model.

In Chapter 5, as our main contribution, we applied fault attack method on the KATAN block cipher. In our analysis we assume that the adversary has the power to cause errors to exactly one bit of the internal state each time during the cipher execution. However this one bit error is transient rather than permanent, such that, the adversary is able to undo the error by resetting the cipher and induce faults into the internal state as many times as he wants. There are two assumptions that we adopt in our fault model; first we assume the adversary can influence the exact rounds in which the fault is induced; and second the adversary can only induce the fault at a random position with the hope that it will hit the target bit by chance. Adopting this fault model, we first identified the efficient rounds in which the fault attack should be performed on KATAN. To do this we determined the number of linear and quadratic equations that can be found from faulty and non-faulty ciphertext differentials. To derive the equations we used the cube attack method and our extended cube method. To improve the effectiveness of fault induction and consequently improving the success probability, we considered overlapping and sliding fault induction methods to significantly reduce the number of inductions needed. A notable feature of our attack is that, the reduction in the number of faulty bits can improve the success probability if one faces difficulties in performing the fault inductions to affect all the target bits. However, there is a tradeoff; as this results in an increase in the complexity for recovering the remaining bits of the subkey (i.e. those unknown bits that should be guessed). The results of our analysis show that all the three variants of KATAN are susceptible to fault attacks (with an attack complexity which is better than the exhaustive key search and small number of fault inductions is required).

There are several envisaged possibilities for improvements that we leave as interesting open questions for future research. First, we suggest that one may generalize the idea of sliding fault induction method to consider the existence of taps between the actual and the target fault positions. We think that this technique can enhance the success probability of the attack while maintaining the same complexity. Second, we see the potential of recovering more subkey bits if one further considers earlier rounds (within the efficient rounds) as additional target faulty rounds. We noticed the need to involve guessing if one considers earlier rounds due to the lack

of differential characteristics that can uniquely indicate the exact faulty bit positions. The open question to considered is whether considering earlier rounds, which may require some guessing about the faulty bit position, by generalizing the idea of sliding fault induction allows us to obtain a better result.

Although our work in this thesis mainly focused on lightweight block ciphers, we expect that the idea of combining algebraic attack and side-channel attacks could also be applied to standard block ciphers. With regard to our works on lightweight block ciphers (e.g. the attack on the KATAN family of block ciphers) using fault attacks, we envisage that the methods we introduced (combining cube and extended cube methods with fault attacks) could also be applied to other block ciphers if the boolean functions defining the round function and key schedule algorithm are of low degrees. We leave applying these techniques for analysis of other standard block ciphers as an open problem for future research.

Bibliography

[1] Abdul-Latip, S.F., Reyhanitabar, M.R., Susilo, W., Seberry, J.: *On the Security of NOEKEON against Side Channel Cube Attacks*. In: Kwak, J., Deng, R., Won, Y. (Eds.) ISPEC 2010. LNCS, vol. 6047, pp. 45–55. Springer, Heidelberg (2010)

[2] Abdul-Latip, S.F., Reyhanitabar, M.R., Susilo, W., Seberry, J.: *Extended Cubes: Enhancing the Cube Attack by Extracting Low-Degree Non-Linear Equations*. In: Cheung, B. et al. (Eds.) ASIACCS 2011. ACM, pp. 296–305 (2011)

[3] Arnault, F., Berger, T., Lauradoux, C.: *F-FCSR Stream Cipher*. In: Robshaw, M.J.B., Billet, O. (Eds.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 170–178. Springer, Heidelberg (2008)

[4] Ars, G., Faugère, J.-C., Imai, H., Kawazoe, M., Sugita, M.: *Comparison Between XL and Gröbner Basis Algorithms*. In: Lee, P.J. (Ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 157–167. Springer, Heidelberg (2004)

[5] Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: *The EM Side Channel*. In: Kaliski, B.S., Koç, Ç.K., Paar, C. (Eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2002)

[6] Ajwa, I.A., Liu, Z., Wang, P.S.: *Gröbner Bases Algorithm*. Technical report, ICM Technical Reports Series (1995) ICM-199502-00

[7] Akkar, M.L., Bévan, R., Dischamp, P., Moyart, D.: *Power analysis, what is now possible....* In: Okamoto, T. (Ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 489-502. Springer, Heidelberg (2000)

[8] Anderson, R., Biham, B., Knudsen, L.: *Serpent: A Proposal for the Advanced Encryption Standard*. In: Vaudenay, S. (Ed.) FSE 1998. LNCS, vol. 1372 ,pp. 222-238. Springer, Heidelberg (1998)

[9] Anderson, R.J., Kuhn, M.G.: *Tamper Resistance - A Cautionary Note*. In: Proceedings of the Second USENIX Workshop on Electronic Commerce, vol. 2, pp. 1–11. USENIX Association (1996)

[10] Anderson, R.J., Kuhn, M.G.: *Low Cost Attacks on Tamper Resistance Devices*. In: Lomas, M. (Ed.) Proceedings of the 5th International Workshop on Security Protocols. LNCS, vol. 1361, pp. 125–136. Springer, Heidelberg (1997)

[11] Aumasson, J.P., Dinur, I., Henzen, L., Meier, W., Shamir, A.: *Efficient FPGA Implementations of High-Dimensional Cube Testers on the Stream Cipher Grain-128*. Cryptology ePrint Archive, Report 2009/218 (2009)

[12] Aumasson, J.P., Dinur, I., Meier, M., Shamir, A.: *Cube Testers and Key Recovery Attacks On Reduced-Round MD6 and Trivium*. In: Dunkelman, O. (Ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009)

[13] Aumasson, J.P., Henzen L.: *SHA-3 proposal BLAKE*. Available at http://csrc.nist.gov.ezproxy.uow.edu.au/groups/ST/hash/sha-3/index.html (2009)

[14] Babbage, S., Dodd, M.: *The MICKEY Stream Cipher*. In: Robshaw, M.J.B., Billet, O. (Eds.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 191–209. Springer, Heidelberg (2008)

[15] Bard, G.V.: *Algebraic Cryptanalysis*. Springer (2009)

[16] Barker, W.C.: *Recommendation for the Triple Data Encryption Standard (TDEA) Block Cipher*. NIST pub 800-67, U.S Department of Commerce (2004)

[17] Bard, G.V., Courtois, N.T., Jefferson, C.: *Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers*. Cryptology ePrint Archive, Report 2007/024 (2007)

[18] Bard, G.V., Courtois, N.T., Jr, J.N., Sepehrdad, P., Zhang, B.: *Algebraic, AIDA/Cube and Side Channel Analysis of KATAN Family of Block Ciphers*. In: Gong, G., Gupta, K.C. (Eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 176–196. Springer, Heidelberg (2010)

[19] Berbain, C., Billet, O., Canteaut, A., Courtois, N., Debraize, B., Gilbert, H., Goubin, L., Gouget, A., Granboulan, L., Lauradoux, C., Minier, M., Pornin,

T., Sibert, H.: *DECIM v2*. In: Robshaw, M.J.B., Billet, O. (Eds.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 140-151. Springer, Heidelberg (2008)

[20] Berbain, C., Billet, O., Canteaut, A., Courtois, N., Gilbert, H., Goubin, L., Gouget, A., Granboulan, L., Lauradoux, C., Minier, M., Pornin, T., Sibert, H.: *SOSEMANUK, a Fast Software-Oriented Stream Cipher*. In: Robshaw, M.J.B., Billet, O. (Eds.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 98-118. Springer, Heidelberg (2008)

[21] Bernstein, D.J.: *The Salsa20 Family of Stream Ciphers*. In: Robshaw, M., Billet, O. (Eds.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 84–97. Springer, Heidelberg (2008)

[22] Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: *The KECCAK Sponge Function Family*. Available at http://keccak.noekeon.org/ (2008)

[23] Beth, T., Ding, C.: *On Almost Perfect Nonlinear Permutations*. In: Helleseth, T. (Ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 65–76. Springer, Heidelberg (1994)

[24] Bévan, R., Knudsen, R.: *Ways to Enhance Differential Power Analysis*. In: Lee, P.J, Lim, C.H. (Eds.) ICISC 2002. LNCS, vol. 2587, pp. 327-342. Springer, Heidelberg (2003)

[25] Biham, E.: *New Types of Cryptanalytic Attacks Using Related Keys*. In: Tor Helleseth (Ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 229–246. Springer, Heidelberg (1994)

[26] Biham, E.: *A Fast New DES Implementation in Software*. In: Biham, E. (Ed.) FSE 1997. LNCS, vol. 1267, pp. 260–272. Springer, Heidelberg (1997)

[27] Biham, E., Biryukov, A.: *An Improvement of Davies' Attack on DES*. In: Santis, A. D. (Ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 461–467. Springer, Heidelberg (1994)

[28] Biham, E., Kocher, P.C.: *A Known Plaintext Attack on the PKZIP Stream Cipher*. In: Preneel, B. (Ed.) FSE 1994. LNCS, vol. 1008, pp. 144–153. Springer, Heidelberg (1994)

[29] Biham, E., Shamir, A.: *Differential Cryptanalysis of the full 16-round DES*. In: Brickell, E.F. (Ed.) CRYPTO 1992. LNCS, vol. 740, pp. 487–496. Springer, Heidelberg (1993)

[30] Biham, E., Shamir, A.: *Differential Fault Analysis of Secret Key Cryptosystems*. In: Kaliski, B.S (Ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)

[31] Bleichenbacher, D.: *Chosen Ciphertext Attacks against Protocols Based on the RSA Encryption Standard PKCS #1*. In: Krawczky, H. (Ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)

[32] Blum, M., Luby, M., Rubinfield, R.: *Self-Testing/Correcting with Application to Numerical Problems*. In: Ortiz, H. (Ed.) STOC 1990. ACM, pp. 73–83 (1990)

[33] Boesgaard, M., Vesterager, M., Zenner, E.: *The Rabbit Stream Cipher*. In: Robshaw, M., Billet, O. (Eds.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 69–83. Springer, Heidelberg (2008)

[34] Bogdanov, A., Kizhvatov, I., Pyshkin, A.: *Algebraic Methods in Side-Channel Collision Attcks and Practical Collision Detection*. In: Chowdhury, D.R., Rijmen, V., Das, A. (Eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 251–265. Springer, Heidelberg (2008)

[35] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.,Vikkelsoe, C.: *PRESENT: An Ultra-Lightweight Block Cipher*. In: Paillier, P., Verbauwhede, I. (Eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)

[36] Boneh, D., DeMillo, R., Lipton, R.: *On the Importance of Checking Cryptographic Protocols for Faults*. In: Fumy, W. (Ed.) EUROCRYPT 1997. LNCS, vol. 1223, pp. 37–51. Springer, Heidelberg (1997)

[37] Brier, E., Clavier, C., Olivier, F.: *Correlation Power Analysis with a Leakage Model*. In: Joye, M., Quisquater, J.-J. (Eds.) CHES 2004, LNCS, vol. 3156, pp. 16-29. Springer, Heidelberg (2004)

[38] Buchberger, B.: *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD Thesis, University of Innsbruck (1965)

[39] Burwick, C., Coppersmith, D., D'Avignon, E., Gennaro, R., Halevi, S., Jutla, C., Matyas Jr, S.M., O'Connor, L., Peyravian, M., Safford, D., Zunic, N.: *MARS - A Candidate Cipher for AES*. Available at http://www.research.ibm.com/security/mars.pdf (2010)

[40] Cid, C., Leurent, G.: *An Analysis of the XSL Algorithm*. In: Roy, B.K. (Ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 333-352. Springer, Heidelberg (2005)

[41] Clavier, C., Coron, J.S., Dabbous, N.: *Differential Power Analysis in the Presence of Hardware Countermeasures*. In: Koç, Ç.K., Paar, C. (Eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)

[42] Coron, J.S., Kocher, P., Naccache, D.: *Statistics and Secret Leakage*. In: Frankel, Y. (Ed.) FC 2000. LNCS, vol. 1962, pp. 157–173. Springer, Heidelberg (2001)

[43] Courtois, N., Klimov, A., Patarin, J., Shamir, A.: *Effcient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*. In: Preneel, B. (Ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)

[44] Cox, D., Little, J., OShea, D.: *Ideals, Varieties, and Algorithms - An Introduction to Computational Algebraic Geometry and Commutative Algebra, second edn*. Undergraduate Texts in Mathematics. Springer (2006)

[45] Crama, Y., Hammer, P.L.: *Boolean Functions - Theory, Algorithms and Applications*. Encyclopedia of Mathematics and its Application. Cambridge University Press (2011)

[46] Daemen, J., Peeters, M., Van Assche, G.,Rijmen, V.: *Nessie Proposal: NOEKEON*. Available at http://gro.noekeon.org/ (2010)

[47] Daemen, J., Rijmen, V.: *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer (2002)

[48] Davies, D., Murphy, S.: *Pairs and Triplets of DES S-Boxes*. In: Journal of Cryptology, vol. 8(1), pp. 1–25 (1995)

[49] de Cannière, C.: *Analysis and Design of Symmetric Encryption Algorithms*. PhD Thesis, Katholieke Universiteit Leuven (2007)

[50] de Cannière, C., Biryukov, A., Preneel, B.: *An Introduction to Block Cipher Cryptanalysis*. In: Proceedings of the IEEE, vol. 94(2), pp. 346–356 (2006)

[51] de Cannière, C., Dunkelman, O., Knezevic, M.: *KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers*. In: Clavier, C., Gaj, K. (Eds.) CHES 2009. LNCS, vol. 5754, pp. 272–288. Springer, Heidelberg (2009)

[52] de Canniere, C., Preneel, B.: *TRIVIUM*. In: Robshaw, M.J.B., Billet, O. (Ed.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008)

[53] Diffie, W., Hellman, M.E.: *New Directions in Cryptography*. In: IEEE Transaction on Information Theory, vol. 22(6), pp. 644–654 (1976)

[54] Dinur, I., Shamir, A.: *Cube Attacks on Tweakable Black Box Polynomials*. Cryptology ePrint Archive, Report 2008/385 (2008)

[55] Dinur, I., Shamir, A.: *Cube Attacks on Tweakable Black Box Polynomials*. In: Joux, A. (Ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)

[56] Dinur, I., Shamir, A.: *Side Channel Cube Attacks on Block Ciphers*. Cryptology ePrint Archive, Report 2009/127 (2009)

[57] Dinur, I., Shamir, A.: *Generic Analysis of Small Cryptographic Leaks*. In: Breveglieri, L., Joye, M., Koren, I., Naccache, D., Verbauwhede, I. (Eds.) FDTC 2010, pp. 39–48. IEEE Computer Society (2010)

[58] Dinur, I., Shamir, A.: *Breaking GRAIN-128 with Dynamic Cube Attacks*. In: Joux, A. (Ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2009)

[59] DVDCCA. *Content Scrambling System* (CSS). Available at http://www.dvdcca.org/css.aspx, (2010)

[60] Eibach, T., Pilz, E., Völkel, G.: *Attacking Bivium Using SAT Solvers*. In: Büning, H.K., Zhao, X. (Eds.) SAT'08. LNCS, vol. 4996, pp. 63–76. Springer, Heidelberg (2008)

[61] Englund, H., Johansson, T., Turan, M.S.: *A Framework for Chosen IV Statistical Analysis of Stream Ciphers*. In: Srinathan, K., Rangan, C.P., Yung, M. (Eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268-281. Springer, Heidelberg (2007)

[62] Feistel, H.: *Block Cipher Cryptographic System*. US Patent 3,798,359. Filed June 30, (1971)

[63] Feistel, H.: *Cryptography and Computer Privacy*. In: Scientifc American, vol. 228(5), pp. 15–23 (1973)

[64] Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein Hash Function Family. Available at http://www.skein-hash.info/ (2008)

[65] Ferguson, N., Schroeppel, R., Whiting, D.: *A Simple Algebraic Representation of Rijndael*. In: Vaudenay, S., Youssef, A.M. (Eds.) SAC 2001. LNCS, vol. 2259, pp. 103-111. Springer, Heidelberg (2001)

[66] Filiol, E.: *A New Statistical Testing for Symmetric Ciphers and Hash Functions*. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (Eds.) ICICS 2002. LNCS, vol. 2513, pp. 342-353. Springer, Heidelberg (2002)

[67] Fischer, S., Khazaei, S., Meier, W.: *Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers*. In: Vaudenay, S. (Ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008)

[68] Fraenkel, A.S., Yesha, Y.: *Complexity of Problems in Games, Graphs, and Algebraic Equations*. In: Discrete Applied Mathematics, vol. 1, pp. 15-30 (1979)

[69] Gandolfi, K., Mourtel, C., Olivier, F.: *Electromagnetic Analysis: Concrete Results*. In: Koç, C.K., Naccache, D., Paar, C. (Eds.) CHES 2001. LNCS, vol. 2162, pp. 251-261. Springer, Heidelberg (2001)

[70] Gauravaram, P., Knudsen, L., Matusiewicz, K., Mendel, F., Rechberger, C., Schlffer, M., Thomsen, S.: *Grøstl a SHA-3 Candidate*. Available at http://www.groestl.info/ (2009)

[71] Giraud, C., Knudsen, E.W.: *Fault Attacks on Signature Schemes*. In: Wang, H., Pieprzyk, J., Varadharajan, V. (Eds.) ACISP 2004. LNCS, vol. 3108, pp. 478–491. Springer, Heidelberg (2004)

[72] Goubin, L., Patarin, J.: *DES and Differential Power Analysis - The Duplication Method*. In: Koç, Ç.K., Paar, C. (Eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)

[73] Hell, M., Johansson, T.: *Breaking the F-FCSR-H Stream Cipher in Real Time*. In: Pieprzyk, J. (Ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 557–569. Springer, Heidelberg (2008)

[74] Hell, M., Johansson, T., Meier, M.: *The Grain Family of Stream Ciphers*. In: Robshaw, M., Billet, O. (Eds.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 179–190. Springer, Heidelberg (2008)

[75] Hellman, M.E., Merkle, R., Schroeppel, R., Washington, L., Diffie, W., Pohlig, S., Schweitzer, P.: *Results of an Initial Attempt to Cryptanalyze the Data Encryption Standard*. Technical Report SEL 76-042, Stanford University, Information Systems Laboratory, September (1976)

[76] Hoch, J.J., Shamir, A.: *Fault Analysis of Stream Ciphers*. In: Joye, M., Quisquater, J.-J. (Eds.) CHES 2004. LNCS, vol. 3156, pp. 240–253. Springer, Heidelberg (2004)

[77] Hojík, M., Rudolf, B.: *Differential Fault Analysis of Trivium*. In: Nyberg, K. (Ed.) FSE 2008. LNCS, vol. 5086, pp. 158–172. Springer, Heidelberg (2008)

[78] Hojík, M., Rudolf, B.: *Floating Fault Analysis of Trivium*. In: Chowdhury, D.R., Rijmen, V., Das, A. (Eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 239–250. Springer, Heidelberg (2008)

[79] Hu, Y., Zhang, F., Zhang, Y.: *Hard Fault Analysis of Trivium*. Cryptology ePrint Archive, Report 2009/333 (2009)

[80] Indesteege, S., Keller, N., Dunkelman, O., Biham, E., Preneel, B.: *A Practical Attack on KeeLoq*. In: Smart, N., (Ed.) EUROCRYPT 2008. LNCS, vol. 4065, pp. 1–18. Springer, Heidelberg (2008)

[81] Jakobsen, T., Knudsen, L.R.: *The Interpolation Attack on Block Ciphers*. In: Biham, E. (Ed.) FSE 1997. LNCS, vol. 1267, pp. 28–40. Springer, Heidelberg (1997)

[82] Khazaei, S., Meier, W.: *New Directions in Cryptanalysis of Self-Synchronizing Stream Ciphers*. In: Chowdhury, D.R., Rijmen, V., Das, A. (Eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 15-26. Springer, Heidelberg (2008)

[83] Knudsen, L.R., Rijmen, V.: *Ciphertext-Only Attack on Akellare*. In: Cryptologia, vol. 24(2), pp. 135–147 (2000)

[84] Kocher, P.C.: *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*. In: Koblitz, N. (Ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)

[85] Kocher, J., Jaffe, J., Jun, B.: *Introduction to Differential Power Analysis and Related Attacks*. Technical Report, Cryptography Research, Inc (1998)

[86] Kocher, J., Jaffe, J., Jun, B.: *Differential Power Analysis*. In: Weiner, M.J. (Ed.) CRYPTO 99. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)

[87] Koeune, F., Standaert, F.-X.: *A Tutorial on Physical Security and Side-Channel Attacks*. In: Aldini, A., Gorrieri, R., Martinelli, F. (Eds.) FOSAD 2004/2005. LNCS, vol. 3655, pp. 78–108. Springer, Heidelberg (2005)

[88] Kömmerling, O., Kuhn, M. G.: *Design Principles for Tamper Resistant Smart Card Processors*. In: Proceedings of the USENIX Workshop on Smartcard Technologies, pp. 9-20 (1999)

[89] Lai, X.: *Higher Order Derivatives and Differential Cryptanalysis*. In: Communication and Cryptology, pp. 227–233. Kluwer Academic Publisher (1994)

[90] Mamiya, H., Miyaji, A., Morimoto, H.: *Efficient Countermeasures against RPA, DPA, and SPA*. In: Joye, M., Quisquater, J.J (Eds.) CHES 2004. LNCS, vol. 3156, pp. 243–319. Springer, Heidelberg (2004)

[91] Mangard, S.: *Hardware Countermeasures against DPA A Statistical Analysis of Their Effectiveness*. In: Okamoto, T. (Ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 222–235. Springer, Heidelberg (2004)

[92] Mantin, I., Shamir, A.: *A Practical Attack on Broadcast RC4*. In: Matsui, M. (Ed.) FSE 2001. LNCS, vol. 2355, pp. 152–164. Springer, Heidelberg (2001)

[93] Matsui, M.: *Linear Cryptanalysis Method for DES Cipher*. In: Helleseth, T. (Ed.) EUROCRYPT 93. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1993)

[94] May, T.C., Woods, M.H.: *A New Physical Mechanism for Soft Errors in Dynamic Memories*. In: Proceedings of the 16th International Reliability Physics Symposium. pp. 33–40 (1978)

[95] Mayer-Sommer, R.: *Smartly Analysis the Simplicity and the Power of Simple Power Analysis on Smartcards*. In: Koç, Ç.K., Paar, C. (Eds.) CHES 2000. LNCS, vol. 1965, pp. 78–92. Springer, Heidelberg (2000)

[96] Menezes, A.J., Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptograhy*. CRC Press (1997)

[97] Microchip Technology Inc. *KeeLoq® Authentication Products*. Available at http://www.microchip.com/keeloq/ (2010)

[98] Mroczkowski, P., Szmidt, J.: *The Cube Attack on Courtois Toy Cipher*. Cryptology ePrint Archive, Report 2009/497, IACR (2009)

[99] National Bureau of Standards. *Data Encryption Standard*. FIPS pub. 46, U.S Department of Commerce (1977)

[100] Nyberg, K.: *Differentially Uniform Mappings for Cryptography*. In: Helleseth, T. (Ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 55–64. Springer, Heidelberg (1994)

[101] Nyberg, K., Knudsen, L.R.: *Provable Security against Differential Cryptanalysis*. In: Brickell, E.F. (Ed.) CRYPTO 1992. LNCS, vol. 740, pp. 566–574. Springer, Heidelberg (1993)

[102] ONeil, S.: *Algebraic Structure Defectoscopy*. Cryptology ePrint Archive, Report 2007/378, IACR (2007)

[103] Oswald, E.: *On Side-Channel Attacks and the Application of Algorithmic Countermeasures*. PhD Thesis, University of Technology Graz (IAIK-TUG) (2003)

[104] Oswald, E., Mangard, S., Herbst, C., Tillich, S.: *Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers*. In: Pointcheval, D. (Ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 192–207. Springer, Heidelberg (2006)

[105] Otto, M.: *Fault Attacks and Countermeasures*. PhD Thesis, Universität Paderborn (2004)

[106] Peterson, I.: *Chinks in Digital Armor - Exploiting Faults to Break Smart Card Cryptosystems*. In: Science News, vol. 151(5), pp. 78–79 (1997)

[107] Piret, G.-F.: *Block Ciphers - Security Proofs, Cryptanalysis, Design and Fault Attacks*. PhD Thesis, Université Catholique de Louvain (2005)

[108] Piret, G.-F, Quisquater, J.: *A Differential Fault Attack Technique Against SPN Structures, with Application to the AES and KHAZAD*. In: Walter, C. D., Koç, Ç. K., Paar, C. (Eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003)

[109] Popp, T., Oswald, E., Mangard, S.: *Power Analysis Attacks and Countermeasures*. In: IEEE, Design & Test of Computers, vol. 24(6), pp. 535–543 (2007)

[110] Quisquater, J.-J., Samyde, D.: *A New Tool for Non-Intrusive Analysis of Smart Cards based on Electro-Magnetic Emissions: the SEMA and DEMA Methods*. In: EUROCRYPT 2000 rump session (2000)

[111] Quisquater, J.-J., Samyde, D.: *Electromagnetic Analysis (EMA)- Measures and Countermeasures for Smart Cards*. In: Attali, I., Jensen, T. (Eds.) e-Smart 2001. LNCS, vol. 2140, pp. 200-210. Springer, Heidelberg (2001)

[112] Raddum, H.: *Cryptanalytic Results on Trivium*. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/039 (2006)

[113] Raddum, H., Samaev, I.: *New Technique for Solving Sparse Equation Systems*. Cryptology ePrint Archive, Report 2006/475, IACR (2006)

[114] Renauld, M., Standaert, F.X.: *Algebraic Side-Channel Attacks*. Cryptology ePrint Archive, Report 2009/279 (2009)

[115] Rivest, R., Agre, B., Bailey, D.V., Crutchfield, C., Dodis, Y., Fleming, K.E., Khan, A., Krishnamurthy, J., Lin, Y., Reyzin, L., Shen, E., Sukha, J., Sutherland, D., Tromer, E., Yin, Y.L.: *The MD6 Hash Function - A Proposal to NIST for SHA-3*. Available at http://groups.csail.mit.edu/cis/md6/ (2008)

[116] Rijmen, V., Daemen, J., Preneel, B., Bosselaers, A., Win, E.D.: *The Cipher SHARK*. In: Gollmann, D. (Ed.) FSE 1996. LNCS, vol. 1039, pp. 99–111. Springer, Heidelberg (1996)

[117] Rivest, R.L., Robshaw, M.J.B., Sidney, R., Yin, Y.L.: *The RC6^{TM} Block Cipher*. Available at ftp://ftp.rsasecurity.com/pub/rsalabs/rc6/rc6v11.pdf (2010)

[118] Rivest, R.L., Shamir, A., Adleman, L.: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM. ACM, vol. 21(2), pp. 120–126 (1978)

[119] Rudeanu, S.: *Boolean Functions and Equations*. Elsevier Science Publishing Co Inc., (1974)

[120] Saarinen, M.-J.O.: *Chosen-IV Statistical Attacks on eStream Ciphers*. In: Malek, M., Fernandez-Medina, E., Hernando, J. (Eds.) SECRYPT 2006, pp. 260-266. INSTICC Press (2006)

[121] SATLIB Benchmark Problems. *DIMACS Format*. Available at http://www.cs.ubc.ca/ hoos/SATLIB/benchmarks/SAT/satformat.ps (2011)

[122] SATLive. *The International SAT Competitions*. Available at http://www.satcompetition.org/ (2007)

[123] Schneier, B.: *Applied Cryptography - Protocols, Algorithms and Source Code in C*. John Wiley & Sons (1996)

[124] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: *The Twofish Encryption Algorithm*. Wiley (1999)

[125] Schramn, K.: A*dvanced Methods in Side Channel Cryptanalysis*. PhD Thesis, University of Bochum in Germany (2006)

[126] Shannon, C.: *Communication Theory of Secrecy Systems*. In: Bell System Technical Journal, vol. 28(4), pp. 656–715 (1949)

[127] Skorobogatov, S.P., Anderson, R.J.: *Optical Fault Induction Attacks*. In: Kaliski Jr, B.S., Koç, C.K., Paar, C. (Eds.) CHES 2002. LNCS, vol. 2523, pp. 31–48. Springer, Heidelberg (2002)

[128] Stay, M.: *ZIP Attacks with Reduced Known Plaintext*. In: Matsui, M. (Ed.) FSE 2001. LNCS, vol. 2355, pp. 125–134. Springer, Heidelberg (2001)

[129] Stubblefield, A., Ioannidis, J., Rubin, A.D.: *A Key Recovery Attack on the 802.11b Wired Equivalent Privacy Protocol (WEP)*. ACM Transactions on Information and System Security, vol. 7(2), pp. 319–332 (2004)

[130] Tuchman, W.: *A Brief History of the Data Encryption Standard*. In: Denning, D.E., Denning, P.J. (Eds.) Internet Besieged - Countering Cyberspace Scofflaws, pp. 275–280. ACM Press/Addison-Wesley Publishing (1997)

[131] Vielhaber, M.: *Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack*. Cryptology ePrint Archive, Report 2007/413, IACR (2007)

[132] Vielhaber, M.: *AIDA Breaks BIVIUM (A&B) in 1 Minute Dual Core CPU Time*. Cryptology ePrint Archive, Report 2009/402, IACR (2009)

[133] Wang, X., Yin, Y.L., Yu, H.: *Finding Collision in the Full SHA-1*. In: Shoup, V. (Ed.) Crypto 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)

[134] Wu, H.: The Stream Cipher HC-128. In: Robshaw, M., Billet, O. (Eds.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 39–47. Springer, Heidelberg (2008)

[135] Wu, H.: *The Hash Function JH*. Available at http://www3.ntu.edu.sg/home/wuhj/research/jh/ (2009)

[136] Yang, L., Wang, M., Qiao, S.: *Side Channel Cube Attack on PRESENT*. In: Garay, J.A., Miyaji, A., Otsuka, A. (Eds.) CANS 2009. LNCS, vol. 5888, pp. 379–391. Springer, Heidelberg (2009)

[137] Yen, S.M., Kim, S., Lim, S., Moon, S.: *A Countermeasure against One Physical Cryptanalysis May Benefit Another Attack*. In: Kim, K. (Ed.) ICISC 2001. LNCS, vol. 2288, pp. 414–427. Springer, Heidelberg (2001)

[138] Zhang, A., Lim, C.-W., Khoo, K., Lei, W., Pieprzyk, J.: *Extensions of Cube Attack based on Low Degree Annihilators*. Cryptology ePrint Archive, Report 2009/049, IACR (2009)

[139] Ziegler, J., Lanford, W.A.: *Effect of Cosmic Rays on Computer Memories*. In: Science, vol. 206, pp. 776–788. (1979)

Appendix A

The Result of Cube Attack on PRESENT

Table A.1: Hamming weight bit position, cube indexes and the corresponding superpoly equations for PRESENT-80 from the leakage after the first round.

HW Bit	Cube Indexes	Superpoly Equation	HW Bit	Cube Indexes	Superpoly Equation
2	{59,58}	$k_{72}+k_{75}$	2	{63,62,23,20}	$k_{77}+k_{39}+k_{38}+k_{77}k_{39}+k_{77}k_{38}+1$
2	{63,62}	$k_{76}+k_{79}$	2	{63,61,23,21}	$k_{77}k_{37}$
2	{63,61,57,56}	$k_{77}+k_{77}k_{76}$	2	{63,62,23,22}	$k_{77}+k_{37}+k_{77}k_{37}+1$
2	{63,62,57,56}	$k_{77}+k_{76}+k_{77}k_{76}+1$	2	{63,61,18,16}	$k_{77}k_{36}$
2	{63,61,51,49}	$k_{77}k_{65}$	2	{63,62,18,16}	$k_{36}+k_{77}k_{36}$
2	{63,62,51,49}	$k_{65}+k_{77}k_{65}$	2	{63,61,15,12}	$k_{77}+k_{77}k_{31}+k_{77}k_{30}$
2	{63,62,55,53}	$k_{69}+k_{77}k_{69}$	2	{63,62,15,12}	$k_{77}+k_{31}+k_{30}+k_{77}k_{31}+k_{77}k_{30}+1$
2	{63,61,55,53}	$k_{77}k_{69}$	2	{63,61,15,13}	$k_{77}k_{29}$
2	{63,62,55,54}	$k_{77}+k_{69}+k_{77}k_{69}+1$	2	{63,62,15,14}	$k_{77}+k_{29}+k_{77}k_{29}+1$
2	{63,61,43,41}	$k_{77}k_{57}$	2	{63,61,10,8}	$k_{77}k_{28}$
2	{63,62,43,42}	$k_{77}+k_{57}+k_{77}k_{57}+1$	2	{63,62,9,8}	$k_{77}+k_{28}+k_{77}k_{28}+1$
2	{63,61,35,33}	$k_{77}k_{49}$	2	{63,61,7,4}	$k_{77}+k_{77}k_{23}+k_{77}k_{22}$
2	{63,62,35,34}	$k_{77}+k_{49}+k_{77}k_{49}+1$	2	{63,62,7,4}	$k_{77}+k_{23}+k_{22}+k_{77}k_{23}+k_{77}k_{22}+1$
2	{63,61,27,25}	$k_{77}k_{41}$	2	{63,61,7,5}	$k_{77}k_{21}$
2	{63,62,27,26}	$k_{77}+k_{41}+k_{77}k_{41}+1$	2	{63,62,7,5}	$k_{21}+k_{77}k_{21}$
2	{63,61,19,17}	$k_{77}k_{33}$	2	{63,61,2,0}	$k_{77}k_{20}$
2	{63,62,19,18}	$k_{77}+k_{33}+k_{77}k_{33}+1$	2	{63,62,1,0}	$k_{77}+k_{20}+k_{77}k_{20}+1$
2	{63,61,11,9}	$k_{77}k_{25}$	3	{63,62,61,60,59,57,51,49}	k_{67}
2	{63,62,11,10}	$k_{77}+k_{25}+k_{77}k_{25}+1$	3	{63,62,61,60,59,57,52,50}	k_{64}
2	{63,61,59,57}	$k_{77}k_{73}$	3	{63,62,61,60,59,57,47,45}	k_{63}
2	{63,62,59,58}	$k_{77}+k_{73}+k_{77}k_{73}+1$	3	{63,62,61,60,59,57,43,41}	k_{59}
2	{63,61,3,1}	$k_{77}k_{17}$	3	{63,62,61,60,59,57,44,42}	k_{56}
2	{63,62,3,2}	$k_{77}+k_{17}+k_{77}k_{17}+1$	3	{63,62,61,60,59,57,39,37}	k_{55}
2	{63,60,55,54}	$k_{79}+k_{78}+k_{69}+k_{79}k_{69}+k_{78}k_{69}+1$	3	{63,62,61,60,59,57,35,33}	k_{51}
2	{63,61,55,52}	$k_{77}+k_{77}k_{71}+k_{77}k_{70}$	3	{63,62,61,60,59,57,36,34}	k_{48}
2	{63,62,55,52}	$k_{77}+k_{71}+k_{70}+k_{77}k_{71}+k_{77}k_{70}+1$	3	{63,62,61,60,59,57,31,29}	k_{47}
2	{63,61,50,48}	$k_{77}k_{68}$	3	{63,62,61,60,59,57,27,25}	k_{43}
2	{63,62,50,48}	$k_{68}+k_{77}k_{68}$	3	{63,62,61,60,59,57,28,26}	k_{40}
2	{63,61,47,44}	$k_{77}+k_{77}k_{63}+k_{77}k_{62}$	3	{63,62,61,60,59,57,23,21}	k_{39}
2	{63,62,47,44}	$k_{77}+k_{63}+k_{62}+k_{77}k_{63}+k_{77}k_{62}+1$	3	{63,62,61,60,59,57,19,17}	k_{35}
2	{63,61,47,45}	$k_{77}k_{61}$	3	{63,62,61,60,59,57,20,18}	k_{32}
2	{63,62,47,45}	$k_{61}+k_{77}k_{61}$	3	{63,62,61,60,59,57,15,13}	k_{31}
2	{63,61,42,40}	$k_{77}k_{60}$	3	{63,62,61,60,59,57,11,9}	k_{27}
2	{63,62,42,40}	$k_{60}+k_{77}k_{60}$	3	{63,62,61,60,59,57,12,10}	k_{24}
2	{63,61,39,36}	$k_{77}+k_{77}k_{55}+k_{77}k_{54}$	3	{63,62,61,60,59,57,7,5}	k_{23}
2	{63,62,39,36}	$k_{77}+k_{55}+k_{54}+k_{77}k_{55}+k_{77}k_{54}+1$	3	{63,62,61,60,59,57,3,1}	k_{19}
2	{63,61,39,37}	$k_{77}k_{53}$	3	{63,62,61,60,59,57,4,2}	k_{16}
2	{63,62,39,38}	$k_{77}+k_{53}+k_{77}k_{53}+1$	3	{63,62,61,60,58,52,51,49}	k_{72}
2	{63,61,34,32}	$k_{77}k_{52}$	3	{63,62,61,60,59,57,55,53}	k_{71}
2	{63,62,33,32}	$k_{77}+k_{52}+k_{77}k_{52}+1$	3	{63,62,61,60,59,57,52,49}	$k_{65}+k_{66}+1$
2	{63,61,31,28}	$k_{77}+k_{77}k_{47}+k_{77}k_{46}$	3	{63,62,61,60,59,57,44,41}	$k_{57}+k_{58}+1$
2	{63,62,31,28}	$k_{77}+k_{47}+k_{46}+k_{77}k_{47}+k_{77}k_{46}+1$	3	{63,62,61,60,59,57,36,33}	$k_{49}+k_{50}+1$
2	{63,61,31,29}	$k_{77}k_{45}$	3	{63,62,61,60,59,57,28,25}	$k_{41}+k_{42}+1$
2	{63,62,31,30}	$k_{77}+k_{45}+k_{77}k_{45}+1$	3	{63,62,61,60,59,57,20,17}	$k_{33}+k_{34}+1$
2	{63,61,26,24}	$k_{77}k_{44}$	3	{63,62,61,60,59,57,12,9}	$k_{25}+k_{26}+1$
2	{63,62,25,24}	$k_{77}+k_{44}+k_{77}k_{44}+1$	3	{63,62,61,60,57,52,51,49}	$k_{73}+k_{74}+1$
2	{63,61,23,20}	$k_{77}+k_{77}k_{39}+k_{77}k_{38}$	3	{63,62,61,60,59,57,4,1}	$k_{17}+k_{18}+1$

Table A.2: Hamming weight bit position, cube indexes and the corresponding superpoly equations for PRESENT-128 from the leakage after the first round.

HW Bit	Cube Indexes	Superpoly Equation	HW Bit	Cube Indexes	Superpoly Equation
2	{59,58}	$k_{120} + k_{123}$	2	{63,62,23,20}	$k_{125} + k_{87} + k_{86} + k_{125}k_{87} + k_{125}k_{86} + 1$
2	{63,62}	$k_{124} + k_{127}$	2	{63,61,23,21}	$k_{125}k_{85}$
2	{63,61,57,56}	$k_{125} + k_{125}k_{124}$	2	{63,62,23,22}	$k_{125} + k_{85} + k_{125}k_{85} + 1$
2	{63,62,57,56}	$k_{125} + k_{124} + k_{125}k_{124} + 1$	2	{63,61,18,16}	$k_{125}k_{84}$
2	{63,61,51,49}	$k_{125}k_{113}$	2	{63,62,18,16}	$k_{84} + k_{125}k_{84}$
2	{63,62,51,49}	$k_{113} + k_{125}k_{113}$	2	{63,61,15,12}	$k_{125} + k_{125}k_{79} + k_{125}k_{78}$
2	{63,62,55,53}	$k_{117} + k_{125}k_{117}$	2	{63,62,15,12}	$k_{125} + k_{79} + k_{78} + k_{125}k_{79} + k_{125}k_{78} + 1$
2	{63,61,55,53}	$k_{125}k_{117}$	2	{63,61,15,13}	$k_{125}k_{77}$
2	{63,62,55,54}	$k_{125} + k_{117} + k_{125}k_{117} + 1$	2	{63,62,15,14}	$k_{125} + k_{77} + k_{125}k_{77} + 1$
2	{63,61,43,41}	$k_{125}k_{105}$	2	{63,61,10,8}	$k_{125}k_{76}$
2	{63,62,43,42}	$k_{125} + k_{105} + k_{125}k_{105} + 1$	2	{63,62,9,8}	$k_{125} + k_{76} + k_{125}k_{76} + 1$
2	{63,61,35,33}	$k_{125}k_{97}$	2	{63,61,7,4}	$k_{125} + k_{125}k_{71} + k_{125}k_{70}$
2	{63,62,35,34}	$k_{125} + k_{97} + k_{125}k_{97} + 1$	2	{63,62,7,4}	$k_{125} + k_{71} + k_{70} + k_{125}k_{71} + k_{125}k_{70} + 1$
2	{63,61,27,25}	$k_{125}k_{89}$	2	{63,61,7,5}	$k_{125}k_{69}$
2	{63,62,27,26}	$k_{125} + k_{89} + k_{125}k_{89} + 1$	2	{63,62,7,5}	$k_{69} + k_{125}k_{69}$
2	{63,61,19,17}	$k_{125}k_{81}$	2	{63,61,2,0}	$k_{125}k_{68}$
2	{63,62,19,18}	$k_{125} + k_{81} + k_{125}k_{81} + 1$	2	{63,62,1,0}	$k_{125} + k_{68} + k_{125}k_{68} + 1$
2	{63,61,11,9}	$k_{125}k_{73}$	3	{63,62,61,60,59,57,51,49}	k_{115}
2	{63,62,11,10}	$k_{125} + k_{73} + k_{125}k_{73} + 1$	3	{63,62,61,60,59,57,52,50}	k_{112}
2	{63,61,59,57}	$k_{125}k_{121}$	3	{63,62,61,60,59,57,47,45}	k_{111}
2	{63,62,59,58}	$k_{125} + k_{121} + k_{125}k_{121} + 1$	3	{63,62,61,60,59,57,43,41}	k_{107}
2	{63,61,3,1}	$k_{125}k_{65}$	3	{63,62,61,60,59,57,44,42}	k_{104}
2	{63,62,3,2}	$k_{125} + k_{65} + k_{125}k_{65} + 1$	3	{63,62,61,60,59,57,39,37}	k_{103}
2	{63,60,55,54}	$k_{127} + k_{126} + k_{117} + k_{127}k_{117} + k_{126}k_{117} + 1$	3	{63,62,61,60,59,57,35,33}	k_{99}
2	{63,61,55,52}	$k_{125} + k_{125}k_{119} + k_{125}k_{118}$	3	{63,62,61,60,59,57,36,34}	k_{96}
2	{63,62,55,52}	$k_{125} + k_{119} + k_{118} + k_{125}k_{119} + k_{125}k_{118} + 1$	3	{63,62,61,60,59,57,31,29}	k_{95}
2	{63,61,50,48}	$k_{125}k_{116}$	3	{63,62,61,60,59,57,27,25}	k_{91}
2	{63,62,50,48}	$k_{116} + k_{125}k_{116}$	3	{63,62,61,60,59,57,28,26}	k_{88}
2	{63,61,47,44}	$k_{125} + k_{125}k_{111} + k_{125}k_{110}$	3	{63,62,61,60,59,57,23,21}	k_{87}
2	{63,62,47,44}	$k_{125} + k_{111} + k_{110} + k_{125}k_{111} + k_{125}k_{110} + 1$	3	{63,62,61,60,59,57,19,17}	k_{83}
2	{63,61,47,45}	$k_{125}k_{109}$	3	{63,62,61,60,59,57,20,18}	k_{80}
2	{63,62,47,45}	$k_{109} + k_{125}k_{109}$	3	{63,62,61,60,59,57,15,13}	k_{79}
2	{63,61,42,40}	$k_{125}k_{108}$	3	{63,62,61,60,59,57,11,9}	k_{75}
2	{63,62,42,40}	$k_{108} + k_{125}k_{108}$	3	{63,62,61,60,59,57,12,10}	k_{72}
2	{63,61,39,36}	$k_{125} + k_{125}k_{103} + k_{125}k_{102}$	3	{63,62,61,60,59,57,7,5}	k_{71}
2	{63,62,39,36}	$k_{125} + k_{103} + k_{102} + k_{125}k_{103} + k_{125}k_{102} + 1$	3	{63,62,61,60,59,57,3,1}	k_{67}
2	{63,61,39,37}	$k_{125}k_{101}$	3	{63,62,61,60,59,57,4,2}	k_{64}
2	{63,62,39,38}	$k_{125} + k_{101} + k_{125}k_{101} + 1$	3	{63,62,61,60,58,52,51,49}	k_{120}
2	{63,61,34,32}	$k_{125}k_{100}$	3	{63,62,61,60,59,57,55,53}	k_{119}
2	{63,62,33,32}	$k_{125} + k_{100} + k_{125}k_{100} + 1$	3	{63,62,61,60,59,57,52,49}	$k_{113} + k_{114} + 1$
2	{63,61,31,28}	$k_{125} + k_{125}k_{95} + k_{125}k_{94}$	3	{63,62,61,60,59,57,44,41}	$k_{105} + k_{106} + 1$
2	{63,62,31,28}	$k_{125} + k_{95} + k_{94} + k_{125}k_{95} + k_{125}k_{94} + 1$	3	{63,62,61,60,59,57,36,33}	$k_{97} + k_{98} + 1$
2	{63,61,31,29}	$k_{125}k_{93}$	3	{63,62,61,60,59,57,28,25}	$k_{89} + k_{90} + 1$
2	{63,62,31,30}	$k_{125} + k_{93} + k_{125}k_{93} + 1$	3	{63,62,61,60,59,57,20,17}	$k_{81} + k_{82} + 1$
2	{63,61,26,24}	$k_{125}k_{92}$	3	{63,62,61,60,59,57,12,9}	$k_{73} + k_{74} + 1$
2	{63,62,25,24}	$k_{125} + k_{92} + k_{125}k_{92} + 1$	3	{63,62,61,60,57,52,51,49}	$k_{121} + k_{122} + 1$
2	{63,61,23,20}	$k_{125} + k_{125}k_{87} + k_{125}k_{86}$	3	{63,62,61,60,59,57,4,1}	$k_{65} + k_{66} + 1$

Appendix B

The Result of Fault Attack on KATAN32

Tables B.1, B.2, B.3 and B.4 show the polynomial equations resulting from proper and faulty ciphertext differential for faulty rounds $t = 231, 237, 243, 249$.

Table B.1: Fault injection after $t = 231$ rounds of KATAN32

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{8+t}	Δc_7	s_{10+t}
s_{9+t}	Δc_8	s_{11+t}
s_{10+t}	Δc_9	s_{12+t}
s_{11+t}	Δc_{17}	s_{9+t}
s_{19+t}	Δc_{28}	s_{22+t}
s_{20+t}	Δc_{29}	s_{23+t}
s_{21+t}	Δc_{30}	s_{24+t}
s_{22+t}	Δc_{31}	s_{25+t}

Table B.2: Fault injection after $t = 237$ rounds of KATAN32

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{1+t}	Δc_{28}	$s_{19+t} + s_{23+t} + s_{28+t} + k_{480} + s_{21+t}s_{24+t}$
	Δc_{24}	$s_{22+t} + s_{26+t} + s_{31+t} + k_{474} + s_{24+t}s_{27+t}$
	Δc_6	s_{6+t}
	Δc_4	$s_{4+t} + s_{15+t} + k_{481} + s_{0+t}s_{5+t} + s_{7+t}s_{9+t}$

Table B.2 (Continued)

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{2+t}	Δc_{29}	$s_{20+t} + s_{24+t} + s_{29+t} + k_{478} + s_{22+t}s_{25+t}$
	Δc_{27}	s_{4+t}
	Δc_{25}	s_{0+t}
	Δc_5	$s_{5+t} + s_{16+t} + k_{479} + s_{1+t}s_{6+t} + s_{8+t}s_{10+t}$
s_{3+t}	Δc_{30}	$s_{25+t} + s_{30+t} + k_{476} + s_{23+t}s_{26+t}$
	Δc_{28}	s_{5+t}
	Δc_{26}	s_{1+t}
	Δc_{12}	s_{8+t}
	Δc_6	$s_{6+t} + s_{17+t} + k_{477} + s_{2+t}s_{7+t} + s_{9+t}s_{11+t}$
s_{4+t}	Δc_{27}	s_{2+t}
	Δc_7	$s_{7+t} + s_{18+t} + k_{475} + s_{3+t}s_{8+t} + s_{10+t}s_{12+t}$
s_{5+t}	Δc_{30}	s_{7+t}
	Δc_{28}	s_{3+t}
	Δc_{21}	$s_{21+t} + s_{26+t} + k_{484} + s_{19+t}s_{22+t}$
	Δc_8	s_{19+t}
s_{9+t}	Δc_2	s_{11+t}
s_{10+t}	Δc_{12}	s_{12+t}
s_{11+t}	Δc_7	s_{9+t}
s_{12+t}	Δc_{12}	s_{10+t}
s_{19+t}	Δc_{22}	s_{22+t}
s_{20+t}	Δc_{23}	s_{23+t}
s_{21+t}	Δc_{24}	s_{24+t}
s_{22+t}	Δc_{25}	s_{25+t}
s_{23+t}	Δc_{26}	s_{26+t}
	Δc_{12}	s_{20+t}
s_{24+t}	Δc_{27}	s_{27+t}
	Δc_{20}	$s_{7+t} + s_{18+t} + s_{22+t} + s_{27+t} + k_{475} + k_{482} + s_{3+t}s_{8+t} +$ $s_{10+t}s_{12+t} + s_{20+t}s_{23+t} + 1$
	Δc_{13}	s_{21+t}

Table B.3: Fault injection after $t = 243$ rounds of KATAN32

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{0+t}	Δc_{21}	$s_{22+t} + s_{27+t} + k_{494} + s_{20+t}s_{23+t}$
s_{1+t}	Δc_{27}	s_{6+t}
	Δc_{22}	$s_{23+t} + s_{28+t} + k_{492} + s_{21+t}s_{24+t}$
	Δc_{20}	s_{3+t}
s_{2+t}	Δc_{28}	s_{7+t}
	Δc_{23}	$s_{24+t} + s_{29+t} + k_{490} + s_{22+t}s_{25+t}$
	Δc_{21}	s_{4+t}
	Δc_{19}	s_{0+t}
s_{3+t}	Δc_{24}	$s_{25+t} + s_{30+t} + k_{488} + s_{23+t}s_{26+t}$
	Δc_{22}	s_{5+t}
	Δc_{20}	s_{1+t}
	Δc_0	$s_{6+t} + s_{17+t} + k_{489} + s_{2+t}s_{7+t} + s_{9+t}s_{11+t}$
s_{4+t}	Δc_{25}	$s_{26+t} + s_{31+t} + k_{486} + s_{24+t}s_{27+t}$
	Δc_{21}	s_{2+t}
	Δc_1	$s_{7+t} + s_{18+t} + k_{487} + s_{3+t}s_{8+t} + s_{10+t}s_{12+t}$
s_{18+t}	Δc_4	s_{21+t}
	Δc_1	$s_{4+t} + s_{15+t} + k_{493} + s_{0+t}s_{5+t} + s_{7+t}s_{9+t}$
s_{19+t}	Δc_5	s_{22+t}
	Δc_2	$s_{5+t} + s_{16+t} + k_{491} + s_{1+t}s_{6+t} + s_{8+t}s_{10+t}$
s_{21+t}	Δc_7	s_{24+t}
s_{22+t}	Δc_8	s_{25+t}
	Δc_5	s_{19+t}
s_{23+t}	Δc_9	s_{26+t}
	Δc_6	s_{20+t}
s_{24+t}	Δc_{10}	s_{27+t}
	Δc_7	s_{21+t}
s_{26+t}	Δc_{20}	$s_{21+t} + s_{23+t} + s_{31+t} + k_{486} + k_{496} + s_{19+t}s_{22+t} + s_{24+t}s_{27+t} + 1$
	Δc_9	s_{23+t}

Table B.4: Fault injection after $t = 249$ rounds of KATAN32

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{4+t}	Δc_{19}	$s_{22+t} + s_{26+t} + s_{31+t} + k_{498} + s_{24+t}s_{27+t}$
s_{5+t}	Δc_{20}	s_{0+t}
s_{21+t}	Δc_1	s_{24+t}
s_{23+t}	Δc_3	s_{26+t}
	Δc_0	s_{20+t}
s_{25+t}	Δc_2	s_{22+t}

Tables B.5, B.6, B.7 and B.8 below show the differential characteristics for faulty rounds $t = 231, 237, 243, 249$ respectively. Denote '0' and '1' as differential values 0 and 1 respectively of the corresponding ciphertext bit differential Δc_j, and '-' as unknown differential value.

Table B.5: Differential characteristics for KATAN32 (faulty round t=231)

		Δc_j															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
s_{j+t}	0	-	0	-	-	0	-	-	-	-	-	-	-	-	0	0	0
	1	-	-	-	0	-	-	-	-	-	-	-	-	-	0	0	-
	2	-	-	0	-	-	-	-	-	-	-	-	-	-	0	-	0
	3	-	0	-	-	-	-	-	-	-	-	-	-	-	0	0	-
	4	0	0	-	-	-	-	-	-	-	-	-	-	-	0	0	0
	5	-	0	-	-	-	-	-	-	-	-	-	-	-	0	0	1
	6	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	-
	7	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-
	8	-	0	-	-	-	-	-	-	-	-	-	-	-	0	0	-
	9	0	-	0	-	-	-	-	-	-	-	-	-	-	0	-	0
	10	-	0	-	-	-	-	-	-	-	-	-	-	-	0	0	-
	11	0	-	0	-	-	-	-	-	-	-	-	-	-	0	-	0
	12	-	0	-	-	-	-	-	-	-	-	-	-	-	0	0	-
	13	0	0	0	-	0	-	0	-	-	-	-	1	-	0	0	0
	14	0	0	-	0	-	0	-	-	-	-	1	-	-	0	0	0
	15	0	0	0	-	0	-	-	-	-	1	-	-	-	0	0	0
	16	-	0	-	0	-	-	-	-	1	-	-	-	-	0	0	1
	17	0	-	0	-	-	-	-	1	-	-	-	-	-	0	1	0
	18	-	0	-	-	-	-	1	-	-	-	-	-	-	0	0	-
	19	0	-	-	-	-	1	-	-	-	-	-	-	-	0	-	0
	20	-	-	-	-	1	-	-	-	-	-	-	-	-	-	0	1
	21	-	-	-	1	-	-	-	-	-	-	-	-	-	0	1	-
	22	-	-	1	-	-	-	-	-	-	-	-	-	-	1	-	0
	23	-	1	-	-	-	-	-	-	-	-	-	-	-	-	0	0
	24	1	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0
	25	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1
	26	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	0
	27	0	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0
	28	0	1	-	0	-	0	-	-	0	-	-	-	-	0	0	0
	29	1	-	0	-	-	-	-	0	-	-	-	-	-	0	0	0
	30	-	0	-	-	-	-	0	-	-	-	-	-	-	0	0	0
	31	0	-	-	-	-	0	-	-	-	-	-	-	-	0	0	0

Table B.5 (Continued)

s_{j+t} \ Δc_j	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	-	0	-	-	-	-	1	-	-	-	-	-	-	-
1	0	-	0	-	-	-	-	1	-	-	-	-	-	-	-	-
2	-	0	-	-	-	-	1	-	-	-	-	-	-	-	-	-
3	0	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-
4	0	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-
5	0	-	-	1	-	-	-	-	-	-	0	-	-	-	-	-
6	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
7	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8	0	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-
9	-	0	-	-	-	-	0	-	-	-	-	-	-	-	-	-
10	0	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-
11	-	0	-	-	0	0	-	-	-	-	-	-	-	-	-	-
12	0	-	-	1	0	-	-	1	-	-	0	-	-	-	-	-
13	0	0	1	0	-	0	1	-	0	0	0	1	-	0	-	-
14	0	1	0	-	0	1	-	0	0	0	1	-	0	-	-	-
15	0	0	-	0	1	-	0	0	0	1	0	0	-	0	-	-
16	0	-	0	1	-	0	0	0	1	-	0	-	-	-	-	-
17	-	0	1	-	0	0	0	1	-	0	-	-	-	-	-	-
18	0	1	-	0	0	0	1	0	0	-	0	-	-	-	-	-
19	1	-	0	0	0	1	0	0	-	0	-	-	-	-	-	-
20	-	0	0	0	1	0	0	-	0	-	-	-	-	-	-	-
21	0	0	0	1	0	0	-	0	-	-	-	-	-	-	-	-
22	0	0	1	-	0	-	0	-	-	-	-	-	-	-	-	-
23	0	1	-	0	-	0	-	-	-	-	-	-	-	-	-	-
24	1	-	0	-	0	-	-	-	-	-	-	-	-	-	-	-
25	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
26	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
27	0	-	0	-	0	-	-	-	-	-	-	-	-	-	-	-
28	0	0	0	0	0	0	-	0	-	-	-	-	1	-	-	-
29	0	0	0	-	0	-	0	-	-	-	-	-	-	-	-	-
30	0	0	-	0	-	0	-	-	-	-	1	-	-	-	-	-
31	0	-	0	-	0	-	-	-	-	1	-	-	-	-	-	-

Table B.6: Differential characteristics for KATAN32 (faulty round t=237)

		Δc_j															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
s_{j+t}	0	0	0	0	1	-	0	-	0	-	-	0	-	-	0	1	0
	1	0	0	1	-	0	-	0	-	-	0	-	-	-	1	0	0
	2	0	1	-	0	-	0	-	-	0	-	-	-	-	0	0	0
	3	1	-	0	-	-	-	-	0	-	-	-	-	-	0	0	0
	4	-	0	-	0	-	0	0	0	0	-	-	-	-	0	0	0
	5	0	-	0	-	0	0	0	0	-	-	-	-	-	0	0	0
	6	-	0	-	0	0	0	0	-	-	-	-	-	-	0	0	0
	7	0	-	0	0	-	0	-	-	-	-	-	-	-	0	0	0
	8	-	0	0	0	-	0	-	0	-	-	-	-	-	0	0	0
	9	0	0	0	0	0	1	0	-	0	-	-	-	-	0	0	0
	10	0	0	0	0	-	0	-	0	-	-	-	-	-	0	0	0
	11	0	0	0	1	0	0	0	-	0	-	-	-	-	0	0	0
	12	0	0	1	0	-	0	-	0	-	-	-	-	-	0	0	0
	13	0	1	0	0	0	0	0	0	0	0	0	-	0	0	0	0
	14	1	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0
	15	0	0	0	0	0	0	0	0	0	-	0	-	-	0	0	0
	16	0	0	0	0	0	0	0	0	-	0	-	-	-	0	0	0
	17	0	0	0	0	0	0	0	-	0	-	-	-	-	0	0	0
	18	0	0	0	0	-	0	-	0	-	-	-	-	-	0	0	0
	19	0	0	0	-	0	-	0	-	-	-	-	1	-	0	0	0
	20	0	0	-	0	-	0	-	-	-	-	1	-	-	0	0	0
	21	0	0	0	-	0	-	-	-	-	1	-	-	-	0	0	0
	22	-	0	-	0	-	-	-	-	1	-	-	-	-	0	0	1
	23	0	-	0	-	-	-	-	1	-	-	-	-	-	0	0	0
	24	-	0	-	-	-	-	1	-	-	-	-	-	-	0	0	-
	25	0	-	-	0	0	1	-	-	-	-	-	-	-	0	0	0
	26	-	-	0	0	1	-	-	-	-	-	-	-	-	0	0	1
	27	-	0	0	0	-	-	0	-	1	-	-	-	-	0	0	-
	28	0	0	0	-	0	0	0	1	-	0	-	0	-	0	0	0
	29	0	0	-	0	0	0	1	-	0	-	0	-	-	0	0	0
	30	0	-	0	0	0	1	-	0	-	0	-	-	0	0	0	0
	31	-	0	0	0	1	-	0	-	0	-	-	0	-	0	0	1

Table B.6 (Continued)

s_{j+t} \ Δc_j	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	-	0	-	-	-	-	1	-
1	0	0	0	0	0	0	0	-	0	-	-	-	-	1	-	-
2	0	0	0	0	0	0	-	0	-	-	-	-	1	-	-	-
3	0	0	0	-	0	-	0	-	-	-	-	-	-	-	-	-
4	0	0	0	0	0	0	0	0	-	-	1	-	-	-	-	-
5	0	0	0	0	0	0	0	-	-	1	-	-	-	-	-	-
6	0	0	0	0	0	0	-	-	1	-	-	-	-	-	-	0
7	0	0	0	1	0	-	-	1	-	-	-	-	-	-	-	-
8	0	0	0	-	0	-	0	-	-	-	-	-	-	-	-	-
9	0	0	0	0	0	0	-	0	-	-	-	-	0	-	-	-
10	0	0	0	-	0	-	0	-	-	-	-	-	-	-	-	-
11	0	0	0	0	0	0	-	0	-	-	0	0	-	-	-	-
12	0	0	0	-	0	-	0	-	-	0	0	-	-	-	-	-
13	0	0	0	0	0	0	0	0	0	0	-	0	1	-	0	0
14	0	0	0	0	0	0	0	0	0	-	0	1	-	0	0	0
15	0	0	0	0	0	0	0	0	-	0	1	-	0	0	0	1
16	0	0	0	0	0	0	0	-	0	1	-	0	0	0	1	0
17	0	0	0	0	0	0	-	0	1	-	0	0	0	1	-	0
18	0	0	0	1	0	-	0	1	-	0	0	-	1	-	0	-
19	0	0	1	0	-	0	1	-	0	0	0	1	-	0	-	-
20	0	1	0	-	0	1	-	0	0	0	1	-	0	-	-	-
21	0	0	-	0	1	-	0	0	0	1	0	0	-	0	-	-
22	0	-	0	1	-	0	0	0	1	-	0	-	-	-	-	-
23	-	0	1	-	0	0	0	1	0	0	-	-	-	-	-	-
24	0	1	-	0	0	0	1	0	0	-	0	-	-	-	-	-
25	1	-	0	0	0	1	0	0	0	0	-	-	-	-	-	-
26	-	0	0	0	1	0	0	0	0	-	-	-	-	-	-	-
27	0	0	0	1	0	0	0	0	0	-	0	-	-	-	-	-
28	0	0	1	0	0	0	0	0	0	0	0	0	-	0	-	-
29	0	1	0	0	0	0	0	0	0	0	0	-	0	-	-	-
30	1	0	0	0	0	0	0	0	0	0	-	0	-	-	-	-
31	0	0	0	0	0	0	0	0	0	-	0	-	-	-	-	1

Table B.7: Differential characteristics for KATAN32 (faulty round t=243)

		Δc_j															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
s_{j+t}	0	0	0	0	0	0	-	0	0	0	1	-	0	-	0	0	0
	1	0	0	0	0	-	0	0	0	1	-	0	-	-	0	0	0
	2	0	0	0	-	0	0	0	1	-	0	-	0	-	0	0	0
	3	0	0	-	0	0	0	1	-	0	-	0	-	-	0	0	0
	4	0	0	0	0	0	1	-	0	-	0	-	0	0	0	0	0
	5	0	0	0	0	1	-	0	-	0	-	0	0	-	0	0	1
	6	0	0	0	1	-	0	-	0	-	0	0	0	-	0	1	0
	7	0	0	1	-	0	-	0	-	0	0	0	0	-	1	0	0
	8	0	0	-	0	-	0	-	0	0	0	0	0	-	0	0	0
	9	0	0	0	-	0	-	0	0	0	0	0	1	0	0	0	0
	10	0	0	-	0	-	0	0	0	0	0	1	0	-	0	0	0
	11	0	0	0	-	0	0	0	0	0	1	0	0	0	0	0	0
	12	0	0	-	0	0	0	0	0	1	0	0	0	-	0	0	0
	13	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	16	0	0	0	0	1	0	0	0	0	0	0	0	-	0	0	0
	17	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	18	0	0	1	0	0	0	0	0	0	0	0	0	-	0	0	0
	19	0	1	0	0	0	0	0	0	0	0	0	-	0	0	0	0
	20	1	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0
	21	0	0	0	0	0	0	0	0	0	-	0	-	-	0	0	0
	22	0	0	0	0	0	0	0	0	-	0	-	-	-	0	0	0
	23	0	0	0	0	0	0	0	-	0	-	-	-	-	0	0	0
	24	0	0	0	0	0	0	-	0	-	-	-	-	1	0	0	0
	25	0	0	0	0	0	0	0	-	-	0	0	1	-	0	0	0
	26	0	0	0	0	0	0	-	-	0	0	1	-	-	0	0	0
	27	0	0	0	0	0	0	-	0	0	0	-	-	0	0	0	0
	28	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0
	29	0	0	0	0	0	0	0	0	-	0	0	0	1	0	0	0
	30	0	0	0	0	0	0	0	-	0	0	0	1	-	0	0	0
	31	0	0	0	0	0	0	-	0	0	0	1	-	0	0	0	0

Table B.7 (Continued)

s_{j+t} \ Δc_j	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	-	0
1	0	0	0	1	0	0	0	0	0	0	0	-	0	-	0	-
2	0	0	1	0	0	0	0	0	0	0	0	0	-	0	-	-
3	0	1	0	0	0	0	0	0	0	0	0	-	0	-	-	-
4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-
5	0	0	0	0	0	0	0	0	0	0	0	1	0	-	-	1
6	0	0	0	0	0	0	0	0	0	0	0	-	-	-	1	-
7	0	0	0	0	0	0	0	0	0	0	0	-	-	1	-	-
8	0	0	0	0	0	0	0	0	0	0	0	-	0	-	-	-
9	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	-
10	0	0	0	0	0	0	0	0	0	0	0	-	0	-	-	-
11	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	-
12	0	0	0	0	0	0	0	0	0	0	0	-	0	-	-	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
16	0	0	0	0	0	0	0	0	0	0	0	1	0	-	0	1
17	0	0	0	0	0	0	0	0	0	0	0	0	-	0	1	-
18	0	0	0	0	0	0	0	0	0	0	0	-	0	1	-	0
19	0	0	0	0	0	0	0	0	0	0	-	0	1	-	0	0
20	0	0	0	0	0	0	0	0	0	-	0	1	-	0	0	0
21	0	0	0	0	0	0	0	0	-	0	1	-	0	0	0	1
22	0	0	0	0	0	0	0	-	0	1	-	0	0	0	1	0
23	0	0	0	0	0	0	-	0	1	-	0	0	0	1	-	0
24	0	0	0	0	0	-	0	1	-	0	0	0	1	0	0	-
25	0	0	0	0	0	0	1	-	0	0	0	1	0	0	-	0
26	0	0	0	0	0	1	-	0	0	0	1	0	0	0	-	-
27	0	0	0	0	0	-	0	0	0	1	0	0	0	0	0	-
28	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	1	0	0	0	0	0	0	0	-	0
31	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	-

Table B.8: Differential characteristics for KATAN32 (faulty round t=249)

		Δc_j															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
s_{j+t}	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	-	0	0	0	1	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	0	1	-	0	0	0
	5	0	0	0	0	0	0	0	0	0	0	1	-	0	0	0	0
	6	0	0	0	0	0	0	0	0	0	1	-	0	-	0	0	0
	7	0	0	0	0	0	0	0	0	1	-	0	-	0	0	0	0
	8	0	0	0	0	0	0	0	0	-	0	-	0	-	0	0	0
	9	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0	0
	10	0	0	0	0	0	0	0	0	-	0	-	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	1
	12	0	0	0	0	0	0	0	0	-	0	0	0	0	0	1	0
	13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	16	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	17	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	18	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	19	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	20	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	21	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	22	0	0	0	0	1	0	0	0	0	0	0	0	-	0	0	0
	23	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	24	0	0	1	0	0	0	0	0	0	0	0	0	-	0	0	0
	25	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	26	1	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
	27	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	31	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0

Table B.8 (Continued)

s_{j+t} \ Δc_j	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
22	0	0	0	0	0	0	0	0	0	0	0	1	0	-	0	1
23	0	0	0	0	0	0	0	0	0	0	0	0	-	0	1	-
24	0	0	0	0	0	0	0	0	0	0	0	-	0	1	-	0
25	0	0	0	0	0	0	0	0	0	0	0	0	1	-	0	0
26	0	0	0	0	0	0	0	0	0	0	0	1	-	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	1
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Appendix C

The Result of Fault Attack on KATAN48

Tables C.1, C.2, C.3, C.4 and C.5 show the polynomial equations resulting from proper and faulty ciphertext differential for faulty rounds $t = 234, 238, 242, 246, 250$.

Table C.1: Fault injection after t=234 rounds of KATAN48

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{6+t}	Δc_{26}	s_{15+t}
	Δc_{25}	s_{14+t}
s_{9+t}	Δc_{28}	s_{17+t}
s_{11+t}	Δc_{18}	$s_{19+t} + 1$
s_{29+t}	Δc_{41}	s_{37+t}
s_{30+t}	Δc_{42}	s_{38+t}

Table C.2: Fault injection after t=238 rounds of KATAN48

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{4+t}	Δc_{9}	$s_{12+t} + 1$
	Δc_{16}	s_{13+t}
s_{5+t}	Δc_{17}	s_{14+t}
s_{6+t}	Δc_{24}	s_{15+t}
s_{8+t}	Δc_{47}	s_{0+t}
	Δc_{13}	s_{16+t}
s_{9+t}	Δc_{14}	s_{17+t}
s_{10+t}	Δc_{15}	s_{18+t}

Table C.2 (Continued)

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{11+t}	Δc_{16}	s_{19+t}
s_{12+t}	Δc_{17}	s_{20+t}
	Δc_{16}	s_{29+t}
	Δc_{15}	s_{3+t}
s_{13+t}	Δc_{17}	s_{30+t}
s_{14+t}	Δc_{18}	s_{31+t}
	Δc_{17}	s_{5+t}
s_{15+t}	Δc_{19}	s_{32+t}
	Δc_{6}	s_{7+t}
s_{16+t}	Δc_{20}	s_{33+t}
	Δc_{13}	s_{8+t}
s_{17+t}	Δc_{38}	$s_{29+t} + s_{35+t} + s_{41+t} + k_{482} + s_{30+t}s_{38+t}$
	Δc_{21}	s_{34+t}
	Δc_{14}	s_{9+t}
	Δc_{12}	$s_{16+t} + s_{25+t} + k_{479} + s_{3+t}s_{12+t} + s_{10+t}s_{18+t}$
s_{18+t}	Δc_{22}	s_{35+t}
	Δc_{15}	s_{10+t}
s_{19+t}	Δc_{46}	s_{1+t}
	Δc_{40}	$s_{31+t} + s_{37+t} + s_{43+t} + k_{480} + s_{32+t}s_{40+t}$
	Δc_{14}	$s_{18+t} + s_{27+t} + k_{477} + s_{5+t}s_{14+t} + s_{12+t}s_{20+t}$
s_{29+t}	Δc_{33}	s_{37+t}
s_{30+t}	Δc_{25}	s_{38+t}
	Δc_{17}	$s_{13+t} + s_{22+t} + k_{483} + s_{0+t}s_{9+t} + s_{7+t}s_{15+t}$
s_{31+t}	Δc_{36}	$s_{33+t} + s_{39+t} + s_{45+t} + k_{478} + s_{34+t}s_{42+t} + 1$
	Δc_{35}	s_{39+t}
	Δc_{18}	$s_{14+t} + s_{23+t} + k_{481} + s_{1+t}s_{10+t} + s_{8+t}s_{16+t}$
	Δc_{7}	s_{4+t}
	Δc_{1}	$s_{4+t} + s_{40+t} + s_{46+t} + k_{476} + s_{35+t}s_{43+t}$
s_{32+t}	Δc_{36}	s_{40+t}
s_{33+t}	Δc_{37}	s_{41+t}
s_{34+t}	Δc_{38}	s_{42+t}
s_{35+t}	Δc_{39}	s_{43+t}

Table C.3: Fault injection after t=242 rounds of KATAN48

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{6+t}	Δc_{45}	s_{14+t}
	Δc_{10}	s_{15+t}
s_{12+t}	Δc_9	$s_{20+t} + 1$
	Δc_8	s_{29+t}
	Δc_7	s_{3+t}
s_{15+t}	Δc_{46}	s_{7+t}
s_{16+t}	Δc_{47}	s_{8+t}
	Δc_{12}	s_{33+t}
s_{17+t}	Δc_{13}	s_{34+t}
	Δc_6	s_{9+t}
s_{18+t}	Δc_{14}	s_{35+t}
	Δc_7	s_{10+t}
s_{19+t}	Δc_{15}	s_{36+t}
s_{30+t}	Δc_{17}	s_{38+t}
	Δc_9	$s_{13+t} + s_{22+t} + k_{491} + s_{0+t}s_{9+t} + s_{7+t}s_{15+t}$
s_{31+t}	Δc_{35}	$s_{40+t} + s_{46+t} + k_{484} + s_{35+t}s_{43+t}$
	Δc_{29}	$s_{34+t} + s_{40+t} + k_{490} + s_{29+t}s_{37+t}$
	Δc_{18}	s_{39+t}
	Δc_{10}	$s_{14+t} + s_{23+t} + k_{489} + s_{1+t}s_{10+t} + s_{8+t}s_{16+t}$
s_{33+t}	Δc_{31}	$s_{36+t} + s_{42+t} + k_{488} + s_{31+t}s_{39+t}$
	Δc_{29}	s_{41+t}
	Δc_{12}	$s_{16+t} + s_{25+t} + k_{487} + s_{3+t}s_{12+t} + s_{10+t}s_{18+t}$
s_{34+t}	Δc_{38}	s_{1+t}
	Δc_{30}	s_{42+t}
s_{35+t}	Δc_{33}	$s_{38+t} + s_{44+t} + k_{486} + s_{33+t}s_{41+t}$
	Δc_{31}	s_{43+t}
	Δc_{14}	$s_{18+t} + s_{27+t} + k_{485} + s_{5+t}s_{14+t} + s_{12+t}s_{20+t}$
s_{36+t}	Δc_{32}	s_{44+t}
s_{39+t}	Δc_{37}	s_{0+t}
	Δc_{36}	s_{5+t}
	Δc_{18}	s_{31+t}

Table C.4: Fault injection after t=246 rounds of KATAN48

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{2+t}	Δc_{33}	s_{10+t}
	Δc_{31}	$s_{29+t} + s_{35+t} + s_{41+t} + k_{498} + s_{30+t}s_{38+t}$
s_{3+t}	Δc_{41}	s_{12+t}
	Δc_{32}	$s_{36+t} + s_{42+t} + k_{496} + s_{31+t}s_{39+t}$
s_{5+t}	Δc_{43}	s_{14+t}
	Δc_{36}	s_{13+t}
	Δc_{34}	$s_{38+t} + s_{44+t} + k_{494} + s_{33+t}s_{41+t}$
s_{8+t}	Δc_{39}	s_{16+t}
	Δc_{37}	$s_{41+t} + s_{47+t} + k_{492} + s_{36+t}s_{44+t}$
	Δc_{31}	s_{0+t}
s_{10+t}	Δc_{41}	s_{18+t}
	Δc_{39}	s_{1+t}
s_{12+t}	Δc_0	s_{29+t}
s_{13+t}	Δc_{44}	s_{21+t}
	Δc_1	s_{30+t}
s_{14+t}	Δc_2	s_{31+t}
s_{15+t}	Δc_{44}	s_{6+t}
	Δc_{38}	s_{7+t}
s_{16+t}	Δc_4	s_{33+t}
s_{18+t}	Δc_6	s_{35+t}
s_{19+t}	Δc_7	s_{36+t}
s_{30+t}	Δc_9	s_{38+t}
	Δc_1	$s_{13+t} + s_{22+t} + k_{499} + s_{0+t}s_{9+t} + s_{7+t}s_{15+t}$
s_{31+t}	Δc_{10}	s_{39+t}
	Δc_2	$s_{14+t} + s_{23+t} + k_{497} + s_{1+t}s_{10+t} + s_{8+t}s_{16+t}$
s_{33+t}	Δc_{12}	s_{41+t}
	Δc_4	$s_{16+t} + s_{25+t} + k_{495} + s_{3+t}s_{12+t} + s_{10+t}s_{18+t}$
s_{36+t}	Δc_{32}	s_{3+t}
	Δc_{15}	s_{44+t}
	Δc_7	$s_{19+t} + s_{28+t} + k_{493} + s_{6+t}s_{15+t} + s_{13+t}s_{21+t}$

Table C.5: Fault injection after t=250 rounds of KATAN48

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{8+t}	Δc_{29}	$s_{41+t} + s_{47+t} + k_{500} + s_{36+t}s_{44+t}$
s_{44+t}	Δc_7	s_{36+t}

Tables C.6, C.7, C.8, C.9 and C.10 below show the differential characteristics for faulty rounds $t = 234, 238, 242, 246, 250$ respectively. Denote '0' and '1' as differential values 0 and 1 respectively of the corresponding ciphertext bit differential Δc_j, and '-' as unknown differential value.

Table C.6: Differential characteristics for KATAN48 (faulty round t=234)

	Δc_j															
s_{j+t}	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	-	0	0	0	0	-	-	-	-	-	0	-	-	-	-	-
1	0	0	0	0	-	-	-	-	-	0	-	-	-	-	-	-
2	0	0	0	-	-	-	-	-	0	-	-	-	-	-	-	-
3	0	0	-	-	-	-	-	0	-	-	-	-	-	-	-	-
4	0	-	0	-	-	-	0	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-
6	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-
7	-	-	-	0	0	-	-	-	-	-	-	-	-	-	-	-
8	-	-	0	0	-	-	-	-	-	-	-	-	-	-	-	-
9	-	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-
10	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
11	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
17	0	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-
18	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
20	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-
21	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-
22	-	0	0	0	0	-	-	-	-	-	0	-	-	-	-	-
23	0	0	0	0	-	-	-	-	-	0	-	-	-	-	-	0
24	0	0	0	-	-	-	-	-	0	-	-	-	-	-	0	-
25	0	0	-	-	-	-	-	0	-	-	-	-	-	0	-	-
26	0	-	0	-	-	-	0	-	-	-	-	-	0	-	-	0
27	-	0	-	-	-	0	-	-	-	-	-	0	-	-	0	-
28	-	-	-	-	0	-	-	-	-	-	0	-	-	-	-	-
29	-	-	-	0	-	-	-	-	-	0	-	-	-	-	-	-
30	-	-	0	-	-	-	-	-	0	-	-	-	-	-	-	-
31	-	0	-	-	-	-	-	0	-	-	-	-	-	-	-	-
32	0	-	-	-	-	-	0	-	-	0	-	-	-	-	-	-
33	-	-	-	-	-	0	-	-	0	-	-	-	-	-	-	-
34	-	-	-	-	0	-	-	0	-	-	-	-	-	-	-	-
35	-	-	-	0	-	-	0	-	-	-	-	-	-	-	-	-
36	-	-	0	-	-	0	-	-	-	-	-	-	-	-	-	-
37	-	0	-	-	0	-	-	0	-	1	-	-	-	-	-	-
38	0	-	-	0	-	-	0	-	1	-	-	-	-	-	-	-
39	-	-	0	-	-	0	-	1	-	-	-	-	-	-	-	-
40	-	-	-	-	0	-	1	-	-	-	-	-	-	-	-	-
41	-	-	-	0	-	1	-	-	-	-	-	-	-	-	-	-
42	-	-	0	-	1	0	-	0	-	0	-	-	-	-	-	-
43	-	-	-	1	0	-	0	-	-	-	-	-	-	-	-	-
44	-	-	1	0	-	0	-	-	-	-	-	-	-	-	-	-
45	0	1	0	-	0	0	0	0	-	0	-	-	-	0	-	-
46	1	-	-	0	0	0	0	-	-	-	-	-	0	-	-	-
47	-	-	0	0	0	0	-	-	-	-	-	0	-	-	-	-

Table C.6 (Continued)

s_{j+t} \ Δc_j	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	-	-	-	0	0	-	-	0	0	0	0	-	0	-	-	-
1	-	-	-	0	-	-	0	0	0	0	-	0	-	-	-	0
2	-	-	-	-	-	0	0	0	0	-	0	-	-	-	0	-
3	-	-	-	-	0	0	0	0	-	0	-	-	-	0	-	-
4	-	-	-	0	0	0	0	-	0	-	-	-	0	-	-	-
5	-	-	-	0	0	0	-	-	-	-	-	0	-	-	-	-
6	-	-	-	0	0	-	-	-	-	-	0	-	-	-	-	-
7	-	-	-	0	0	-	-	-	-	0	0	-	-	-	-	0
8	-	-	-	0	-	-	-	-	0	0	-	-	-	-	0	-
9	-	-	-	-	-	-	-	0	0	-	-	-	-	0	-	-
10	-	-	-	-	-	-	0	0	-	-	-	-	0	-	-	0
11	-	-	-	-	-	0	0	-	-	-	-	0	-	-	0	-
12	-	-	-	-	0	0	-	-	-	-	0	-	-	0	-	-
13	-	-	-	0	0	-	-	-	-	0	-	-	0	-	-	-
14	-	-	-	0	1	-	-	-	0	-	1	0	-	-	-	-
15	-	-	-	1	-	-	-	0	-	1	0	-	-	-	-	-
16	-	-	-	-	-	-	0	0	1	0	-	-	-	-	1	0
17	-	-	-	-	-	0	0	1	0	-	-	-	-	1	0	-
18	-	-	-	-	0	0	1	0	-	-	-	-	1	0	-	1
19	-	-	-	0	0	1	0	-	-	-	-	1	0	-	-	-
20	-	-	-	0	0	0	-	1	-	-	0	0	-	1	0	0
21	-	-	-	0	0	-	1	-	-	0	0	-	1	0	-	-
22	0	-	-	0	0	1	-	0	0	0	0	1	0	0	-	0
23	-	-	-	0	1	-	0	0	0	0	1	0	0	-	0	0
24	-	-	-	1	-	0	0	0	0	1	0	0	-	0	0	1
25	-	-	-	-	0	0	0	0	1	0	0	-	0	0	1	0
26	-	-	-	0	0	0	0	1	0	0	-	0	0	1	0	0
27	-	-	-	0	0	0	1	0	0	-	0	0	1	0	0	0
28	-	-	-	0	0	1	0	0	-	0	0	1	0	0	-	-
29	-	-	-	0	1	0	0	-	0	0	1	0	0	-	-	0
30	-	-	-	1	0	0	-	0	0	1	0	0	-	-	0	0
31	-	-	-	0	0	-	0	0	1	0	0	-	-	0	0	0
32	-	-	-	0	-	0	0	1	0	0	0	-	0	0	0	0
33	-	-	-	-	0	0	1	0	0	0	-	0	0	0	0	-
34	-	-	-	0	0	1	0	0	0	-	0	0	0	0	-	0
35	-	-	-	0	1	0	0	-	-	0	0	0	0	-	0	-
36	-	-	-	1	0	0	-	-	0	0	0	0	-	0	-	-
37	-	-	-	0	0	0	0	0	0	0	0	-	0	0	-	0
38	-	-	-	0	0	0	0	0	0	0	-	0	0	-	0	-
39	-	-	-	0	0	0	0	0	0	-	0	0	-	0	-	-
40	-	-	-	0	0	0	0	-	-	0	0	-	0	-	-	-
41	-	-	-	0	0	0	-	-	0	0	-	0	-	-	-	-
42	-	-	-	0	0	0	0	0	0	-	0	0	-	0	-	0
43	-	-	-	0	0	0	0	-	-	0	0	-	0	-	0	-
44	-	-	-	0	0	0	-	-	0	0	-	0	-	0	-	-
45	-	-	-	0	0	0	0	0	0	-	0	0	0	0	-	0
46	-	-	-	0	0	0	0	-	-	0	0	0	0	-	0	-
47	-	-	-	0	0	0	-	-	0	0	0	0	-	0	-	-

Table C.6 (Continued)

s_{j+t} \ Δc_j	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-
1	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-
2	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-
4	-	-	0	-	-	0	-	-	-	-	-	-	-	-	-	-
5	-	0	-	-	0	-	-	-	-	-	-	-	-	-	-	-
6	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
7	-	-	0	-	-	-	-	-	0	-	-	-	-	-	-	-
8	-	0	-	-	-	-	-	0	-	-	-	-	-	-	-	-
9	0	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-
10	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-
11	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-
12	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-
13	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-
14	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-
17	1	-	-	-	0	0	-	-	-	-	-	-	-	-	-	-
18	-	-	-	0	0	-	-	-	-	-	-	-	-	-	-	-
19	0	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-
20	-	0	0	1	0	-	-	-	-	-	-	-	-	-	-	-
21	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-
22	0	1	0	0	-	-	0	0	0	-	-	-	-	-	-	-
23	1	0	0	-	-	0	0	0	-	-	-	-	-	-	-	-
24	0	0	-	-	0	0	0	-	-	-	-	-	-	-	-	-
25	0	-	-	0	0	0	-	-	-	-	-	-	-	-	-	-
26	0	-	0	0	0	0	-	-	-	-	-	-	-	-	-	-
27	-	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-
28	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-
29	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
30	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
31	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
32	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
33	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
34	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
35	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
36	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
37	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
38	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
39	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
40	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
41	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
42	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
43	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
44	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
45	-	-	-	0	-	-	-	-	-	0	-	-	-	-	-	-
46	-	-	0	-	-	-	-	-	0	-	-	-	-	-	-	-
47	-	0	-	-	-	-	-	0	-	-	-	-	-	-	-	-

Table C.7: Differential characteristics for KATAN48 (faulty round t=238)

	Δc_j															
s_{j+t}	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	-	0	-	0	0	0	1	0	-	0	0	0	0	-	0	-
1	0	-	0	0	0	1	0	-	0	0	0	0	-	0	-	-
2	-	0	0	0	1	0	-	0	0	0	0	-	0	-	-	-
3	0	0	0	1	0	-	0	0	0	0	-	0	-	-	-	0
4	0	0	1	0	-	0	0	0	0	-	0	-	-	-	0	-
5	0	1	0	-	0	0	0	0	-	0	-	-	-	0	-	-
6	1	-	-	0	0	0	0	-	-	-	-	-	0	-	-	-
7	0	-	0	0	0	0	0	0	-	0	-	0	0	-	-	-
8	-	0	0	0	0	0	0	-	0	-	0	0	-	-	-	-
9	0	0	0	0	0	0	-	0	-	0	0	-	-	-	-	-
10	0	0	0	0	0	-	0	-	0	0	-	-	-	-	-	-
11	0	0	0	0	-	0	-	0	0	-	-	-	-	-	-	-
12	0	0	0	-	0	-	0	0	-	-	-	-	-	-	-	-
13	0	-	-	0	-	0	0	-	-	-	-	-	-	-	-	-
14	0	1	0	-	0	0	0	-	-	-	-	-	-	0	-	-
15	1	-	-	0	0	0	-	-	-	-	-	-	0	-	-	-
16	0	0	0	0	0	-	0	-	0	0	-	0	-	-	-	-
17	0	0	0	0	-	0	-	0	0	-	0	-	-	-	-	-
18	0	0	0	-	0	-	0	0	-	0	-	-	-	-	-	-
19	0	-	-	0	-	0	0	-	-	-	-	-	-	-	-	-
20	0	0	0	-	0	0	0	0	-	0	-	-	-	0	-	-
21	0	-	-	0	0	0	0	-	-	-	-	-	0	-	-	-
22	0	0	0	0	0	0	0	0	-	0	0	0	0	-	0	-
23	0	0	0	0	0	0	0	-	0	0	0	0	-	0	-	-
24	0	0	0	0	0	0	-	0	0	0	0	-	0	-	-	-
25	0	0	0	0	0	-	0	0	0	0	-	0	-	-	-	0
26	0	0	0	0	-	0	0	0	0	-	0	-	-	-	0	-
27	0	0	0	-	0	0	0	0	-	0	-	-	-	0	-	-
28	0	-	-	0	0	0	0	-	-	-	-	-	0	-	-	-
29	-	-	0	0	0	0	-	-	-	-	-	0	-	-	-	-
30	-	0	0	0	0	-	-	-	-	-	0	-	-	-	-	-
31	0	0	0	0	-	-	-	-	-	0	-	-	-	-	-	0
32	0	0	0	-	-	-	-	-	0	-	-	-	-	-	0	-
33	0	0	-	-	-	-	-	0	-	-	-	-	-	0	-	-
34	0	-	0	-	-	-	0	-	-	-	-	-	0	-	-	0
35	-	0	-	-	-	0	-	-	-	-	-	0	-	-	0	-
36	0	-	-	-	0	-	-	-	-	-	0	-	-	0	-	-
37	0	-	0	0	-	-	0	-	-	0	-	-	0	-	-	0
38	-	0	0	-	-	0	-	-	0	-	-	0	-	-	0	-
39	0	0	-	-	0	-	-	0	-	-	0	-	-	0	-	1
40	0	-	-	0	-	-	0	-	-	0	-	-	0	-	1	-
41	-	-	0	-	-	0	-	-	0	-	-	0	-	1	-	-
42	0	0	0	-	0	-	-	0	-	-	0	-	1	0	-	0
43	0	0	-	0	-	-	0	-	-	0	-	1	0	-	0	-
44	0	-	0	-	-	0	-	-	0	-	1	0	-	0	-	0
45	0	0	0	-	0	-	0	0	0	1	0	-	0	0	0	0
46	0	0	-	0	-	0	0	0	1	0	-	0	0	0	0	-
47	0	-	0	-	0	0	0	1	0	-	0	0	0	0	-	0

Table C.7 (Continued)

		Δc_j															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
s_{j+t}	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	-	0
	1	-	0	-	0	0	0	0	0	0	0	0	0	0	-	0	0
	2	0	-	-	0	0	0	0	0	0	0	0	0	-	0	0	0
	3	-	-	-	0	0	0	0	0	0	0	0	-	0	0	0	0
	4	-	-	-	0	0	0	0	0	0	0	-	0	0	0	0	-
	5	-	-	-	0	0	0	0	0	0	-	0	0	0	0	-	0
	6	-	-	-	0	0	0	0	-	-	0	0	0	0	-	0	-
	7	-	-	-	0	0	0	0	0	0	0	0	0	0	0	-	0
	8	-	-	-	0	0	0	0	0	0	0	0	0	0	-	0	-
	9	-	-	-	0	0	0	0	0	0	0	0	0	-	0	-	0
	10	-	-	-	0	0	0	0	0	0	0	0	-	0	-	0	0
	11	-	-	-	0	0	0	0	0	0	0	-	0	-	0	0	-
	12	-	-	-	0	0	0	0	0	0	-	0	-	0	0	-	-
	13	-	-	-	0	0	0	0	-	-	0	-	0	0	-	-	-
	14	-	-	-	0	0	0	0	0	0	-	0	0	0	-	-	-
	15	-	-	-	0	0	0	0	-	-	0	0	0	-	-	-	0
	16	-	-	-	0	0	0	0	0	0	0	0	-	0	-	0	0
	17	-	-	-	0	0	0	0	0	0	0	-	0	-	0	0	1
	18	-	-	-	0	0	0	0	0	0	-	0	-	0	0	1	0
	19	-	-	-	0	0	0	0	1	-	0	-	0	0	1	0	-
	20	-	-	-	0	0	0	0	0	0	-	0	0	0	0	-	0
	21	-	-	-	0	0	0	0	-	-	0	0	0	0	-	0	-
	22	-	-	0	0	0	0	0	0	0	0	0	0	0	0	-	0
	23	-	0	-	0	0	0	0	0	0	0	0	0	0	-	0	0
	24	0	-	-	0	0	0	0	0	0	0	0	0	-	0	0	0
	25	-	-	-	0	0	0	0	0	0	0	0	-	0	0	0	0
	26	-	-	-	0	0	0	0	0	0	0	-	0	0	0	0	1
	27	-	-	-	0	0	0	0	0	0	-	0	0	0	0	1	0
	28	-	-	0	0	0	0	0	1	-	0	0	0	0	1	0	0
	29	-	0	-	0	0	0	1	-	0	0	0	0	1	0	0	-
	30	0	-	-	0	0	1	-	0	0	0	0	1	0	0	-	0
	31	-	-	-	0	1	-	0	0	0	0	1	0	0	-	0	0
	32	-	-	-	1	-	0	0	0	0	1	0	0	-	0	0	1
	33	-	-	-	-	0	0	0	0	1	0	0	-	0	0	1	0
	34	-	-	-	0	0	0	0	1	0	0	-	0	0	1	0	0
	35	-	-	-	0	0	0	1	0	0	-	0	0	1	0	0	0
	36	-	-	-	0	0	1	0	0	-	0	0	1	0	0	0	-
	37	-	1	-	0	1	0	0	-	0	0	1	0	0	0	0	0
	38	1	-	-	1	0	0	-	0	0	1	0	0	0	0	0	0
	39	-	-	-	0	0	-	0	0	1	0	0	0	0	0	0	0
	40	-	-	-	0	-	0	0	1	0	0	0	0	0	0	0	0
	41	-	-	-	-	0	0	1	0	0	0	0	0	0	0	0	-
	42	-	-	-	0	0	1	0	0	0	0	0	0	0	0	0	0
	43	-	-	-	0	1	0	0	0	0	0	0	0	0	0	0	0
	44	-	-	-	1	0	0	0	0	0	0	0	0	0	0	0	-
	45	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
	46	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
	47	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	-

Table C.7 (Continued)

s_{j+t} \ Δc_j	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	-	0	-	-	-	0	-	-	-	-	-	0	-
1	0	0	-	0	-	-	-	0	-	-	-	-	-	0	-	-
2	0	-	0	-	-	-	0	-	-	-	-	-	0	-	-	0
3	-	0	-	-	-	0	-	-	-	-	-	0	-	-	0	-
4	0	-	-	-	0	-	-	-	-	-	0	-	-	-	-	-
5	-	-	-	0	-	-	-	-	-	0	-	-	-	-	-	-
6	-	-	0	-	-	-	-	-	0	-	-	-	-	-	-	-
7	-	0	0	-	-	-	-	0	-	-	0	-	-	-	-	-
8	0	0	-	-	-	-	0	-	-	0	-	-	-	-	-	0
9	0	-	-	-	-	0	-	-	0	-	-	-	-	-	0	-
10	-	-	-	-	0	-	-	0	-	-	-	-	-	0	-	-
11	-	-	-	0	-	-	0	-	-	-	-	-	0	-	-	-
12	-	-	0	-	-	0	-	-	-	-	-	0	-	-	-	-
13	-	0	-	-	0	-	-	-	-	-	0	-	-	-	-	-
14	0	-	1	0	-	-	-	-	-	0	-	1	-	-	-	-
15	-	1	0	-	-	-	-	-	0	-	1	-	-	-	-	-
16	1	0	-	-	-	-	1	0	-	1	-	0	-	0	0	-
17	0	-	-	-	-	1	0	-	1	-	0	-	0	-	-	0
18	-	-	-	-	1	0	-	-	-	0	-	0	-	-	0	-
19	-	-	-	1	0	-	-	-	0	-	0	-	-	-	-	-
20	-	-	0	0	-	1	0	-	-	0	0	1	-	-	0	-
21	-	0	0	-	1	0	-	-	0	0	1	-	-	-	-	-
22	0	0	0	1	0	0	-	0	0	1	0	0	-	-	0	0
23	0	0	1	0	0	-	0	0	1	0	0	0	-	0	0	0
24	0	1	0	0	-	0	0	1	0	0	0	-	0	0	0	0
25	1	0	0	-	0	0	1	0	0	0	-	0	0	0	0	-
26	0	0	-	0	0	1	0	0	0	-	0	0	0	-	-	0
27	0	-	0	0	1	0	0	-	-	0	0	0	-	-	0	-
28	-	0	0	1	0	0	-	-	0	0	0	-	-	-	-	-
29	0	0	1	0	0	-	-	0	0	0	-	-	-	-	-	-
30	0	1	0	0	-	-	0	0	0	-	-	-	-	-	-	-
31	1	0	0	-	-	0	0	0	-	-	-	-	-	-	-	-
32	0	0	-	-	0	0	0	-	-	-	-	-	-	-	-	-
33	0	-	-	0	0	0	-	-	-	-	-	-	-	-	-	-
34	0	-	0	0	0	0	-	-	-	-	-	-	-	-	-	-
35	-	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-
36	0	0	0	0	-	0	-	-	-	-	-	-	-	-	-	-
37	0	0	0	-	0	0	-	0	-	-	-	-	-	-	-	-
38	0	0	-	0	0	-	0	-	-	-	-	-	-	-	-	-
39	0	-	0	0	-	0	-	-	-	-	-	-	-	-	-	-
40	-	0	0	-	0	-	-	-	-	-	-	-	-	-	-	-
41	0	0	-	0	-	-	-	-	-	-	-	-	-	-	-	-
42	0	-	0	0	-	0	-	-	-	-	-	-	-	-	-	-
43	-	0	0	-	0	-	-	-	-	-	-	-	-	-	-	-
44	0	0	-	0	-	0	-	-	-	-	-	-	-	-	-	-
45	0	-	0	0	0	0	-	-	-	-	-	0	-	-	-	-
46	-	0	0	0	0	-	-	-	-	-	0	-	-	-	-	-
47	0	0	0	0	-	0	-	-	-	0	-	-	-	-	-	0

Table C.8: Differential characteristics for KATAN48 (faulty round t=242)

	Δc_j															
s_{j+t}	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	0	-	0	0	0	0	0	0	-	0	-	0	0	0	1	0
1	-	0	0	0	0	0	0	-	0	-	0	0	0	1	0	-
2	0	0	0	0	0	0	-	0	-	0	0	0	1	0	-	0
3	0	0	0	0	0	-	0	-	0	0	0	1	0	-	0	0
4	0	0	0	0	-	0	-	0	0	0	1	0	-	0	0	0
5	0	0	0	-	0	-	0	0	0	1	0	-	0	0	0	0
6	0	0	-	0	-	0	0	0	1	0	-	0	0	0	0	-
7	0	-	0	-	0	0	0	1	0	-	0	0	0	0	0	0
8	-	0	-	0	0	0	1	0	-	0	0	0	0	0	0	-
9	0	-	0	0	0	1	0	-	0	0	0	0	0	0	-	0
10	-	0	0	0	1	0	-	0	0	0	0	0	0	-	0	-
11	0	0	0	1	0	-	0	0	0	0	0	0	-	0	-	0
12	0	0	1	0	-	0	0	0	0	0	0	-	0	-	0	0
13	0	1	0	-	0	0	0	0	0	0	-	0	-	0	0	-
14	1	0	-	0	0	0	0	0	0	1	0	-	0	0	0	-
15	0	-	0	0	0	0	0	0	1	0	-	0	0	0	-	-
16	-	0	0	0	0	0	0	1	0	0	0	0	0	-	0	-
17	0	0	0	0	0	0	1	0	0	0	0	0	-	0	-	0
18	0	0	0	0	0	1	0	0	0	0	0	-	0	-	0	0
19	0	0	0	0	1	0	0	0	0	0	-	0	-	0	0	-
20	0	0	0	1	0	0	0	0	0	0	0	-	0	0	0	0
21	0	0	1	0	0	0	0	0	0	0	-	0	0	0	0	-
22	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	-
29	0	0	0	0	0	0	0	0	0	-	0	0	0	0	-	0
30	0	0	0	0	0	0	0	0	-	0	0	0	0	-	0	-
31	0	0	0	0	0	0	0	-	0	0	0	0	-	0	-	-
32	0	0	0	0	0	0	-	0	0	0	0	-	0	-	-	-
33	0	0	0	0	0	-	0	0	0	0	-	0	-	-	-	0
34	0	0	0	0	-	0	0	0	0	-	0	-	-	-	0	-
35	0	0	0	-	0	0	0	0	-	0	-	-	-	0	-	-
36	0	0	-	0	0	0	0	-	0	-	-	-	0	-	-	-
37	0	0	0	0	0	0	-	0	0	-	0	0	-	-	0	-
38	0	0	0	0	0	-	0	0	-	0	0	-	-	0	-	-
39	0	0	0	0	-	0	0	-	0	0	-	-	0	-	-	0
40	0	0	0	-	0	0	-	0	0	-	-	0	-	-	0	-
41	0	0	-	0	0	-	0	0	-	-	0	-	-	0	-	-
42	0	0	0	0	-	0	0	-	0	0	0	-	0	-	-	0
43	0	0	0	-	0	0	-	0	0	0	-	0	-	-	0	-
44	0	0	-	0	0	-	0	0	0	-	0	-	-	0	-	-
45	0	0	0	0	-	0	0	0	0	0	0	-	0	-	0	0
46	0	0	0	-	0	0	0	0	0	0	-	0	-	0	0	0
47	0	0	-	0	0	0	0	0	0	-	0	-	0	0	0	1

Table C.8 (Continued)

s_{j+t} \ Δc_j	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	-	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0	-	0	1	0	0	0	0	0	0	0	0	0	0	0
4	0	-	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
6	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
7	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
8	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
12	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
13	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
14	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
15	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
20	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
21	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
22	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
29	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	-
30	-	-	0	0	0	0	0	0	0	0	0	0	0	0	-	0
31	-	0	-	0	0	0	0	0	0	0	0	0	0	-	0	0
32	0	-	-	0	0	0	0	0	0	0	0	0	-	0	0	0
33	-	-	-	0	0	0	0	0	0	0	0	-	0	0	0	0
34	-	-	-	0	0	0	0	0	0	0	-	0	0	0	0	1
35	-	-	-	0	0	0	0	0	0	-	0	0	0	0	1	0
36	-	-	0	0	0	0	0	0	-	0	0	0	0	1	0	0
37	-	0	-	0	0	0	0	0	0	0	0	0	1	0	0	-
38	0	-	-	0	0	0	0	0	0	0	0	1	0	0	-	0
39	-	-	0	0	0	0	0	0	0	0	1	0	0	-	0	0
40	-	-	-	0	0	0	0	0	0	1	0	0	-	0	0	1
41	-	-	-	0	0	0	0	0	1	0	0	-	0	0	1	0
42	-	-	0	0	0	0	0	0	0	0	-	0	0	1	0	0
43	-	-	-	0	0	0	0	0	0	-	0	0	1	0	0	0
44	-	-	1	0	0	0	0	0	-	0	0	1	0	0	0	0
45	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
46	1	-	-	0	0	0	0	0	0	1	0	0	0	0	0	0
47	-	-	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table C.8 (Continued)

s_{j+t} \ Δc_j	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	-	0	0	0	0	-	-	-	-	-
1	0	0	0	0	0	-	0	0	0	0	-	0	-	-	-	0
2	0	0	0	0	-	0	0	0	0	-	0	-	-	-	0	-
3	0	0	0	-	0	0	0	0	-	0	-	-	-	0	-	-
4	0	0	-	0	0	0	0	-	0	-	-	-	0	-	-	-
5	0	-	0	0	0	0	-	-	-	-	-	0	-	-	-	-
6	-	0	0	0	0	-	-	-	-	-	0	-	-	-	-	-
7	0	0	0	0	0	0	-	-	-	0	0	-	-	-	-	0
8	0	0	0	0	0	-	-	-	0	0	-	-	-	-	0	-
9	0	0	0	0	-	0	-	0	0	-	-	-	-	0	-	-
10	0	0	0	-	0	-	0	0	-	-	-	-	0	-	-	0
11	0	0	-	0	-	0	0	-	-	-	-	0	-	-	0	-
12	0	-	0	-	0	0	-	-	-	-	0	-	-	0	-	-
13	-	0	-	0	0	-	-	-	-	0	-	-	0	-	-	-
14	0	-	0	0	0	-	-	-	0	-	1	0	-	-	-	-
15	-	0	0	0	-	-	-	0	-	1	0	-	-	-	-	-
16	0	0	0	-	0	-	0	0	1	0	-	-	-	-	1	0
17	0	0	-	0	-	0	0	1	0	-	-	-	-	1	0	-
18	0	-	0	-	0	0	1	0	-	-	-	-	1	0	-	1
19	-	0	-	0	0	1	0	-	-	-	-	1	0	-	1	-
20	0	-	0	0	0	0	-	1	-	-	0	0	-	1	0	0
21	-	0	0	0	0	-	1	-	-	0	0	-	1	0	0	-
22	0	0	0	0	0	0	-	0	0	0	0	1	0	0	-	0
23	0	0	0	0	0	-	0	0	0	0	1	0	0	-	0	0
24	0	0	0	0	-	0	0	0	0	1	0	0	-	0	0	1
25	0	0	0	-	0	0	0	0	1	0	0	-	0	0	1	0
26	0	0	-	0	0	0	0	1	0	0	-	0	0	1	0	0
27	0	-	0	0	0	0	1	0	0	-	0	0	1	0	0	0
28	-	0	0	0	0	1	0	0	-	0	0	1	0	0	0	-
29	0	0	0	0	1	0	0	-	0	0	1	0	0	-	-	0
30	0	0	0	1	0	0	-	0	0	1	0	0	-	-	0	0
31	0	0	1	0	0	-	0	0	1	0	0	0	-	0	0	0
32	0	1	0	0	-	0	0	1	0	0	0	-	0	0	0	0
33	1	0	0	-	0	0	1	0	0	0	-	0	0	0	0	-
34	0	0	-	0	0	1	0	0	0	-	0	0	0	-	-	0
35	0	-	0	0	1	0	0	-	-	0	0	0	-	-	0	-
36	-	0	0	1	0	0	-	-	0	0	0	0	-	-	-	-
37	0	0	1	0	0	0	0	0	0	0	0	-	0	-	-	0
38	0	1	0	0	0	0	0	0	0	0	-	0	-	-	0	-
39	1	0	0	0	0	0	0	0	0	-	0	0	-	0	-	-
40	0	0	0	0	0	0	0	-	-	0	0	-	0	-	-	-
41	0	0	0	0	0	0	-	-	0	0	-	0	-	-	-	-
42	0	0	0	0	0	0	0	0	0	-	0	0	-	0	-	0
43	0	0	0	0	0	0	0	-	-	0	0	-	0	-	0	-
44	0	0	0	0	0	0	-	-	0	0	-	0	-	-	-	-
45	0	0	0	0	0	0	0	0	0	-	0	0	0	0	-	0
46	0	0	0	0	0	0	0	-	-	0	0	0	0	-	0	-
47	0	0	0	0	0	0	-	-	0	0	0	0	-	-	-	-

Table C.9: Differential characteristics for KATAN48 (faulty round t=246)

	Δc_j															
s_{j+t}	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	-
2	0	0	0	0	0	0	0	-	0	0	0	0	0	0	-	0
3	0	0	0	0	0	0	-	0	0	0	0	0	0	-	0	-
4	0	0	0	0	0	-	0	0	0	0	0	0	-	0	-	0
5	0	0	0	0	-	0	0	0	0	0	0	-	0	-	0	0
6	0	0	0	-	0	0	0	0	0	0	-	0	-	0	0	0
7	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0	1
8	0	0	0	0	0	0	0	0	-	0	-	0	0	0	1	0
9	0	0	0	0	0	0	0	-	0	-	0	0	0	1	0	-
10	0	0	0	0	0	0	-	0	-	0	0	0	1	0	-	0
11	0	0	0	0	0	-	0	-	0	0	0	1	0	-	0	0
12	0	0	0	0	-	0	-	0	0	0	1	0	-	0	0	0
13	0	0	0	-	0	-	0	0	0	1	0	-	0	0	0	0
14	0	0	0	0	-	0	0	0	1	0	-	0	0	0	0	0
15	0	0	0	-	0	0	0	1	0	-	0	0	0	0	0	0
16	0	0	0	0	0	0	1	0	-	0	0	0	0	0	0	1
17	0	0	0	0	0	1	0	-	0	0	0	0	0	0	1	0
18	0	0	0	0	1	0	-	0	0	0	0	0	0	1	0	0
19	0	0	0	1	0	-	0	0	0	0	0	0	1	0	0	0
20	0	0	0	0	-	0	0	0	0	0	0	1	0	0	0	0
21	0	0	0	-	0	0	0	0	0	0	1	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	-
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	-
40	0	0	0	0	0	0	0	0	0	0	0	-	0	0	-	0
41	0	0	0	0	0	0	0	0	0	0	-	0	0	-	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	-
43	0	0	0	0	0	0	0	0	0	0	0	-	0	0	-	0
44	0	0	0	0	0	0	0	0	0	0	-	0	0	-	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0

Table C.9 (Continued)

s_{j+t} \ Δc_j	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	-	0	0	0	0	0	0	0	0	0	0	0	0	1	0
2	-	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
4	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
5	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	1	-	-	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	-	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8	-	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
13	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	1	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
29	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
41	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
44	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table C.9 (Continued)

s_{j+t} \ Δc_j	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0
1	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	-
5	0	0	0	0	0	0	0	0	0	-	0	0	0	0	-	0
6	0	0	0	0	0	0	0	-	-	0	0	0	0	-	0	-
7	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0
8	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	-
9	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	0
10	0	0	0	0	0	0	0	0	0	0	0	-	-	-	0	0
11	0	0	0	0	0	0	0	0	0	0	-	0	-	0	0	-
12	0	0	0	0	0	0	0	0	0	-	0	-	0	0	-	-
13	0	0	0	0	0	0	0	-	-	0	-	0	0	-	-	-
14	0	0	0	0	0	0	0	0	0	-	0	0	1	-	-	-
15	0	0	0	0	0	0	0	-	-	0	0	0	-	-	-	0
16	0	0	0	0	0	0	0	0	0	0	0	-	-	-	0	0
17	0	0	0	0	0	0	0	0	0	0	-	0	-	0	0	1
18	0	0	0	0	0	0	0	0	0	-	0	-	0	0	1	0
19	0	0	0	0	0	0	0	1	-	0	-	0	0	1	0	-
20	0	0	0	0	0	0	0	0	0	-	0	0	0	0	-	0
21	0	0	0	0	0	0	0	-	-	0	0	0	0	-	0	-
22	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-	0
23	0	0	0	0	0	0	0	0	0	0	0	0	1	-	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	1
27	0	0	0	0	0	0	0	0	0	-	0	0	0	0	1	0
28	0	0	0	0	0	0	0	1	-	0	0	0	0	1	0	0
29	0	0	0	0	0	0	1	-	0	0	0	0	1	0	0	-
30	0	0	0	0	0	0	-	0	0	0	0	1	0	0	-	0
31	0	0	0	0	0	-	0	0	0	0	1	0	0	-	0	0
32	0	0	0	0	-	0	0	0	0	1	0	0	-	0	0	1
33	0	0	0	-	0	0	0	0	1	0	0	-	0	0	1	0
34	0	0	-	0	0	0	0	1	0	0	-	0	0	1	0	0
35	0	-	0	0	0	0	1	0	0	-	0	0	1	0	0	0
36	-	0	0	0	0	1	0	0	-	0	0	1	0	0	0	-
37	0	0	0	0	1	0	0	-	0	0	1	0	0	0	0	0
38	0	0	0	1	0	0	-	0	0	1	0	0	0	0	0	0
39	0	0	1	0	0	-	0	0	1	0	0	0	0	0	0	0
40	0	1	0	0	-	0	0	1	0	0	0	0	0	0	0	0
41	1	0	0	-	0	0	1	0	0	0	0	0	0	0	0	-
42	0	0	-	0	0	1	0	0	0	0	0	0	0	0	0	0
43	0	-	0	0	1	0	0	0	0	0	0	0	0	0	0	0
44	-	0	0	1	0	0	0	0	0	0	0	0	0	0	0	-
45	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-

Table C.10: Differential characteristics for KATAN48 (faulty round t=250)

	Δc_j															
s_{j+t}	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-
12	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	0
13	0	0	0	0	0	0	0	0	0	0	0	-	0	-	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	-
18	0	0	0	0	0	0	0	0	0	0	0	0	1	0	-	0
19	0	0	0	0	0	0	0	0	0	0	0	1	0	-	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table C.10 (Continued)

		Δc_j															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
s_{j+t}	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	1
	9	0	-	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	10	-	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	11	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	12	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
	13	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	14	1	0	-	0	0	0	0	0	0	1	0	0	0	0	0	0
	15	0	-	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	16	-	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	17	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	18	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	19	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	20	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	21	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	22	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	23	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	36	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	41	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
	42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	44	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	47	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Table C.10 (Continued)

s_{j+t} \ Δc_j	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
30	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-	0
31	0	0	0	0	0	0	0	0	0	0	0	0	1	-	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	1
35	0	0	0	0	0	0	0	0	0	-	0	0	0	0	1	0
36	0	0	0	0	0	0	0	0	-	0	0	0	0	1	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-
38	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-	0
39	0	0	0	0	0	0	0	0	0	0	1	0	0	-	0	0
40	0	0	0	0	0	0	0	0	0	1	0	0	-	0	0	1
41	0	0	0	0	0	0	0	0	1	0	0	-	0	0	1	0
42	0	0	0	0	0	0	0	0	0	0	-	0	0	1	0	0
43	0	0	0	0	0	0	0	0	0	-	0	0	1	0	0	0
44	0	0	0	0	0	0	0	0	-	0	0	1	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Appendix D

The Result of Fault Attack on KATAN64

Tables D.1, D.2, D.3, D.4 and D.5 show the polynomial equations resulting from proper and faulty ciphertext differential for faulty rounds $t = 236, 238, 242, 246, 250$.

Table D.1: Fault injection after t=236 rounds of KATAN64

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{8+t}	Δc_{37}	s_{3+t}
s_{16+t}	Δc_{38}	s_{28+t}
s_{20+t}	Δc_{36}	$s_{32+t} + s_{42+t}$
s_{28+t}	Δc_{38}	s_{16+t}
s_{30+t}	Δc_{34}	s_{18+t}
s_{31+t}	Δc_{35}	s_{19+t}
s_{33+t}	Δc_{36}	s_{42+t}
s_{45+t}	Δc_{61}	$s_{2+t} + s_{54+t}$
s_{46+t}	Δc_{62}	$s_{3+t} + s_{55+t}$
	Δc_{25}	s_{2+t}
s_{54+t}	Δc_{61}	$s_{2+t} + s_{45+t}$

Table D.2: Fault injection after t=238 rounds of KATAN64

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{2+t}	Δc_{55}	$s_{39+t} + s_{45+t} + s_{54+t} + k_{482} + s_{41+t}s_{50+t}$
	Δc_{10}	$s_{52+t} + s_{61+t} + k_{476} + s_{48+t}s_{57+t}$
s_{5+t}	Δc_{27}	s_{10+t}
	Δc_{13}	s_{0+t}
s_{6+t}	Δc_{28}	s_{11+t}
s_{8+t}	Δc_{16}	s_{3+t}
s_{10+t}	Δc_{27}	s_{5+t}
s_{17+t}	Δc_{27}	$s_{29+t} + s_{39+t}$
s_{18+t}	Δc_{28}	$s_{30+t} + s_{40+t}$
s_{21+t}	Δc_{37}	s_{33+t}
	Δc_{19}	s_{9+t}
	Δc_{18}	$s_{17+t} + s_{30+t} + k_{481} + s_{1+t}s_{6+t} + s_{13+t}s_{25+t} + 1$
s_{24+t}	Δc_{53}	s_{0+t}
	Δc_{34}	s_{46+t}
	Δc_{22}	s_{12+t}
s_{25+t}	Δc_{54}	s_{1+t}
	Δc_{35}	s_{47+t}
	Δc_{26}	$s_{16+t} + s_{29+t} + k_{483} + s_{0+t}s_{5+t} + s_{12+t}s_{24+t}$
	Δc_{23}	s_{13+t}
	Δc_{22}	$s_{21+t} + s_{34+t} + k_{479} + s_{5+t}s_{10+t} + s_{17+t}s_{29+t} + 1$
s_{26+t}	Δc_{24}	s_{14+t}
s_{27+t}	Δc_{25}	s_{15+t}
s_{28+t}	Δc_{39}	s_{16+t}
s_{29+t}	Δc_{40}	s_{17+t}
s_{31+t}	Δc_{28}	s_{40+t}
s_{32+t}	Δc_{36}	s_{20+t}
s_{33+t}	Δc_{37}	s_{21+t}
	Δc_{30}	s_{42+t}
s_{36+t}	Δc_{33}	s_{45+t}

Table D.2 (Continued)

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{39+t}	Δc_{55}	s_{2+t}
	Δc_{36}	s_{48+t}
s_{41+t}	Δc_{38}	s_{50+t}
s_{43+t}	Δc_{59}	s_{6+t}
	Δc_{53}	$s_{0+t} + s_{52+t}$
s_{45+t}	Δc_{55}	$s_{2+t} + s_{54+t}$
s_{46+t}	Δc_{56}	$s_{3+t} + s_{55+t}$
s_{47+t}	Δc_{44}	$s_{56+t} + 1$
	Δc_{35}	$s_{25+t} + s_{38+t} + k_{477} + s_{9+t}s_{14+t} + s_{21+t}s_{33+t}$
s_{48+t}	Δc_{49}	$s_{39+t} + s_{45+t} + s_{51+t} + s_{60+t} + k_{478} + s_{47+t}s_{56+t}$
	Δc_{45}	$s_{57+t} + 1$
	Δc_{36}	s_{39+t}
s_{50+t}	Δc_{63}	$s_{40+t} + s_{46+t} + s_{55+t} + k_{480} + s_{42+t}s_{51+t}$
	Δc_{38}	s_{41+t}

Table D.3: Fault injection after t=242 rounds of KATAN64

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{3+t}	Δc_{56}	s_{15+t}
	Δc_{13}	s_{8+t}
s_{5+t}	Δc_{58}	s_{17+t}
	Δc_{15}	s_{10+t}
s_{7+t}	Δc_{60}	s_{19+t}
	Δc_{17}	s_{12+t}
s_{8+t}	Δc_{4}	s_{3+t}
s_{17+t}	Δc_{58}	s_{5+t}
s_{19+t}	Δc_{60}	s_{7+t}
	Δc_{17}	$s_{31+t} + s_{41+t}$

Table D.3 (Continued)

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{24+t}	Δc_{22}	s_{46+t}
s_{25+t}	Δc_{48}	$s_{43+t} + s_{52+t} + k_{490} + s_{39+t}s_{48+t}$
	Δc_{23}	s_{47+t}
	Δc_{14}	$s_{16+t} + s_{29+t} + k_{491} + s_{0+t}s_{5+t} + s_{12+t}s_{24+t}$
	Δc_{10}	$s_{21+t} + s_{34+t} + k_{487} + s_{5+t}s_{10+t} + s_{17+t}s_{29+t} + 1$
s_{28+t}	Δc_{14}	s_{16+t}
s_{34+t}	Δc_{19}	s_{43+t}
s_{39+t}	Δc_{24}	s_{48+t}
s_{40+t}	Δc_{25}	s_{49+t}
s_{41+t}	Δc_{51}	$s_{46+t} + s_{55+t} + k_{488} + s_{42+t}s_{51+t}$
	Δc_{39}	$s_{50+t} + s_{53+t} + s_{62+t} + k_{484} + s_{49+t}s_{58+t}$
	Δc_{26}	s_{50+t}
	Δc_{17}	$s_{19+t} + s_{32+t} + k_{489} + s_{3+t}s_{8+t} + s_{15+t}s_{27+t}$
s_{45+t}	Δc_{55}	$s_{50+t} + s_{59+t} + k_{486} + s_{46+t}s_{55+t}$
	Δc_{21}	$s_{23+t} + s_{36+t} + k_{485} + s_{7+t}s_{12+t} + s_{19+t}s_{31+t}$
s_{46+t}	Δc_{31}	s_{55+t}
s_{47+t}	Δc_{32}	s_{56+t}
s_{48+t}	Δc_{24}	s_{39+t}
s_{49+t}	Δc_{34}	s_{58+t}
s_{50+t}	Δc_{60}	s_{0+t}
	Δc_{35}	s_{59+t}
	Δc_{26}	s_{41+t}

Table D.4: Fault injection after t=246 rounds of KATAN64

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{3+t}	Δc_{56}	s_{8+t}
	Δc_{44}	s_{15+t}
s_{4+t}	Δc_{57}	s_{9+t}
	Δc_{45}	s_{16+t}
s_{10+t}	Δc_{58}	s_{5+t}
s_{11+t}	Δc_{52}	s_{23+t}
	Δc_{40}	$s_{54+t} + s_{63+t} + k_{492} + s_{50+t}s_{59+t}$
s_{15+t}	Δc_{56}	s_{27+t}
	Δc_{44}	s_{3+t}
s_{16+t}	Δc_{45}	s_{4+t}
s_{19+t}	Δc_{60}	s_{31+t}
	Δc_{48}	s_{7+t}
s_{20+t}	Δc_{61}	s_{32+t}
s_{23+t}	Δc_{9}	s_{45+t}
s_{30+t}	Δc_{3}	s_{39+t}
s_{33+t}	Δc_{6}	s_{42+t}
s_{38+t}	Δc_{2}	$s_{16+t} + s_{29+t} + k_{499} + s_{0+t}s_{5+t} + s_{12+t}s_{24+t}$
s_{41+t}	Δc_{5}	$s_{19+t} + s_{32+t} + k_{497} + s_{3+t}s_{8+t} + s_{15+t}s_{27+t}$
s_{42+t}	Δc_{15}	s_{51+t}
	Δc_{6}	$s_{20+t} + s_{33+t} + k_{495} + s_{4+t}s_{9+t} + s_{16+t}s_{28+t}$
s_{45+t}	Δc_{43}	$s_{50+t} + s_{59+t} + k_{494} + s_{46+t}s_{55+t}$
	Δc_{18}	s_{54+t}
	Δc_{9}	$s_{23+t} + s_{36+t} + k_{493} + s_{7+t}s_{12+t} + s_{19+t}s_{31+t}$
s_{50+t}	Δc_{48}	s_{0+t}
	Δc_{39}	$s_{46+t} + s_{55+t} + k_{496} + s_{42+t}s_{51+t}$
	Δc_{23}	s_{59+t}
	Δc_{14}	s_{41+t}
s_{55+t}	Δc_{19}	s_{46+t}
s_{59+t}	Δc_{43}	$s_{39+t} + s_{45+t} + s_{54+t} + k_{498} + s_{41+t}s_{50+t}$
	Δc_{23}	s_{50+t}

Table D.5: Fault injection after t=250 rounds of KATAN64

Faulty Bit	Ciphertext Bit Differential	Polynomial Equations
s_{3+t}	Δc_{39}	$s_{53+t} + s_{62+t} + k_{500} + s_{49+t}s_{58+t}$
s_{40+t}	Δc_1	s_{49+t}

Tables D.6, D.7, D.8, D.9 and D.10 below show the differential characteristics for faulty rounds $t = 236, 238, 242, 246, 250$ respectively. Denote '0' and '1' as differential values 0 and 1 respectively of the corresponding ciphertext bit differential Δc_j, and '-' as unknown differential value.

Table D.6: Differential characteristics for KATAN64 (faulty round t=236)

	Δc_j															
s_{j+t}	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
0	0	0	-	0	-	0	-	-	0	1	0	-	-	0	-	-
1	0	-	0	-	0	-	-	0	1	0	-	-	0	-	-	-
2	-	0	-	-	-	-	0	1	0	-	-	0	-	-	-	0
3	0	-	-	-	-	0	1	0	-	-	0	-	-	-	0	-
4	-	-	-	-	0	1	0	-	-	0	-	-	-	0	-	-
5	-	-	-	0	1	0	-	-	0	-	-	-	0	-	-	-
6	-	-	0	1	0	-	-	0	-	-	-	0	-	-	-	-
7	-	0	1	-	-	-	0	-	-	-	-	-	-	-	-	-
8	0	1	-	-	-	0	-	-	-	-	-	-	-	-	-	-
9	1	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-
10	-	-	-	0	-	0	-	-	-	-	-	-	-	-	0	-
11	-	-	0	-	0	-	-	-	-	-	-	-	-	0	-	-
12	-	0	-	0	-	-	-	-	-	-	-	-	0	-	-	-
13	0	-	0	-	-	-	-	-	-	-	-	0	-	-	-	-
14	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	0	0	-	-	-	0	0	-	-	-	0	-	-	-	0	-
16	0	-	-	-	0	0	-	-	-	0	-	-	-	0	-	-
17	-	-	-	0	0	-	-	-	0	-	-	-	0	-	-	-
18	-	-	0	-	-	-	-	0	-	-	-	0	-	-	-	0
19	-	0	-	-	-	-	0	-	-	-	0	-	-	-	0	-
20	0	-	-	-	-	0	-	-	-	0	-	-	-	0	-	-
21	-	-	-	-	0	-	-	-	0	-	-	-	0	-	-	-
22	-	-	0	0	-	-	-	0	-	-	-	0	-	-	-	0
23	-	0	0	-	-	-	0	-	-	-	0	-	-	-	0	-
24	0	0	-	-	-	0	-	-	-	0	-	-	-	0	-	-
25	0	-	-	-	0	-	-	-	0	-	-	-	0	-	-	-
26	-	0	0	-	-	-	0	0	0	-	-	0	-	-	-	0
27	0	0	-	-	-	0	0	0	-	-	0	-	-	-	0	-
28	0	-	-	-	0	0	0	-	-	0	-	-	-	0	-	-
29	-	-	-	0	0	0	-	-	0	-	-	-	0	-	-	-
30	-	-	0	0	0	-	-	0	-	-	-	0	-	-	-	-
31	-	0	0	-	-	-	0	-	-	-	-	-	-	-	-	-
32	0	0	-	-	-	0	-	-	-	-	-	-	-	-	-	-
33	0	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-
34	-	-	-	0	0	0	-	-	0	-	-	-	0	-	0	-
35	-	-	0	0	0	-	-	0	-	-	-	0	-	0	-	0
36	-	0	0	0	-	-	0	-	-	-	0	-	0	-	0	-
37	0	0	0	-	-	0	-	-	-	0	-	0	-	0	-	-
38	0	0	-	-	0	-	-	-	0	-	0	-	0	-	-	-
39	0	-	-	0	-	-	-	0	-	0	-	0	-	-	-	0
40	-	0	0	-	-	-	0	-	0	-	0	1	-	-	0	-
41	0	0	-	-	-	0	-	0	-	0	1	-	-	0	-	-
42	0	-	-	-	0	-	0	-	0	1	-	-	0	-	-	-
43	-	-	-	0	-	0	-	0	-	-	-	0	-	-	-	-
44	-	-	0	-	0	-	0	-	-	-	0	-	-	-	-	-
45	-	0	-	0	-	0	-	-	-	0	-	-	-	-	-	-
46	0	-	0	-	0	-	-	-	0	-	-	-	-	-	-	-
47	-	0	-	0	-	-	-	0	-	-	-	-	-	-	-	-
48	0	-	0	-	-	-	0	-	-	-	-	-	-	-	-	-
49	-	0	1	-	-	0	-	-	-	-	-	-	-	-	-	-
50	0	1	-	-	0	-	-	-	-	-	-	-	-	-	-	-
51	1	-	0	0	0	-	-	0	-	1	-	-	0	-	0	-
52	-	0	0	0	-	-	0	-	1	-	-	0	-	0	-	-
53	0	0	0	-	-	0	-	1	-	-	0	-	0	-	-	-
54	0	0	-	-	0	-	1	-	-	0	-	0	-	-	-	-
55	0	-	0	0	0	1	0	-	0	-	0	-	-	-	-	-
56	-	0	0	0	1	0	-	0	-	0	-	-	-	-	-	-
57	0	0	0	1	0	-	0	-	0	-	-	-	-	-	-	-
58	0	0	1	0	-	0	-	0	-	-	-	-	-	-	-	1
59	0	1	0	-	0	-	0	-	-	-	-	-	-	-	1	-
60	1	0	0	0	0	0	-	0	-	0	-	-	0	1	0	-
61	0	0	0	0	0	-	0	-	0	-	-	0	1	0	-	-
62	0	0	0	0	-	0	-	0	-	-	0	1	0	-	-	0
63	0	0	0	-	0	-	0	-	-	0	1	0	-	-	0	-

Table D.6 (Continued)

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

Table D.6 (Continued)

		Δc_j															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
s_{j+t}	0	0	0	0	-	-	0	0	0	-	0	0	-	-	-	0	-
	1	0	0	-	-	0	0	0	-	0	0	-	-	-	0	-	0
	2	0	-	-	0	0	0	-	0	0	-	-	-	0	-	0	-
	3	-	-	0	0	0	-	0	0	-	-	-	0	-	0	-	0
	4	-	0	0	0	-	0	0	-	-	-	0	-	0	-	0	-
	5	0	0	0	-	0	0	-	-	-	0	-	0	-	0	-	-
	6	0	0	-	0	0	-	-	-	0	-	0	-	0	-	-	-
	7	0	-	0	0	-	-	-	0	-	0	-	0	-	-	-	0
	8	-	0	0	-	-	-	0	-	0	-	0	-	-	-	0	-
	9	0	0	-	-	-	0	-	0	-	0	-	-	-	0	-	-
	10	0	-	0	-	0	-	0	-	0	1	-	0	0	-	-	0
	11	-	0	-	0	-	0	-	0	1	-	0	0	-	-	0	-
	12	0	-	0	-	0	-	0	1	-	0	0	-	-	0	-	-
	13	-	0	-	0	-	0	1	-	0	0	-	-	0	-	-	-
	14	0	-	0	-	0	1	-	0	0	-	-	0	-	-	-	-
	15	-	0	0	0	1	-	0	0	-	-	0	0	-	-	0	0
	16	0	0	0	1	-	0	0	-	-	0	0	-	-	0	0	-
	17	0	0	1	-	0	0	-	-	0	0	-	-	0	0	-	-
	18	0	1	-	0	0	-	-	0	0	-	-	0	0	-	-	-
	19	1	-	0	0	-	-	0	0	-	-	0	0	-	-	-	0
	20	-	0	0	-	-	0	0	-	-	0	0	-	-	-	0	-
	21	0	0	-	-	0	0	-	-	0	0	-	-	-	0	-	-
	22	0	-	-	0	0	-	-	0	0	0	-	0	0	-	-	-
	23	-	-	0	0	-	-	0	0	0	-	0	0	-	-	-	0
	24	-	0	0	-	-	0	0	0	-	0	0	-	-	-	0	-
	25	0	0	-	-	0	0	0	-	0	0	-	-	-	0	-	-
	26	0	-	-	0	0	0	-	0	0	0	-	-	0	0	0	1
	27	-	-	0	0	0	-	0	0	0	-	-	0	0	0	1	0
	28	-	0	0	0	-	0	0	0	-	-	0	0	0	1	0	-
	29	0	0	0	-	0	0	0	-	-	0	0	0	1	0	-	0
	30	0	0	-	0	0	0	-	-	0	0	0	1	0	-	0	-
	31	0	-	0	0	0	-	-	0	0	0	1	0	-	0	-	0
	32	-	0	0	0	-	-	0	0	0	1	0	-	0	-	0	-
	33	0	0	0	-	-	0	0	0	1	0	-	0	-	0	-	-
	34	0	0	0	-	0	0	0	1	0	0	0	0	0	0	-	0
	35	0	0	-	0	0	0	1	0	0	0	0	0	0	-	0	-
	36	0	-	0	0	0	1	0	0	0	0	0	0	-	0	-	0
	37	-	0	0	0	1	0	0	0	0	0	0	-	0	-	0	-
	38	0	0	0	1	0	0	0	0	0	0	-	0	-	0	-	-
	39	0	0	1	0	0	0	0	0	0	-	0	-	0	-	-	-
	40	0	1	0	0	0	0	0	0	0	0	-	0	0	0	-	-
	41	1	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0
	42	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0
	43	0	0	0	0	0	-	0	-	0	-	-	-	-	0	-	-
	44	0	0	0	0	-	0	-	0	-	-	-	-	0	-	-	-
	45	0	0	0	-	0	-	0	-	-	-	-	0	-	-	-	-
	46	0	0	-	0	-	0	-	-	-	-	0	-	-	-	-	-
	47	0	-	0	-	0	-	-	-	-	0	-	-	-	-	-	-
	48	-	0	-	0	-	-	-	-	0	-	-	-	-	-	-	-
	49	0	-	0	0	0	-	-	0	0	-	-	0	0	-	-	-
	50	-	0	0	0	-	-	0	0	-	-	0	0	-	-	-	0
	51	0	0	0	-	0	0	0	-	-	0	0	0	-	-	0	-
	52	0	0	-	0	0	0	-	-	0	0	0	-	-	0	-	-
	53	0	-	0	0	0	-	-	0	0	0	-	-	0	-	-	-
	54	-	0	0	0	-	-	0	0	0	-	-	0	-	-	-	0
	55	0	0	0	0	-	0	0	0	-	-	0	0	-	-	0	0
	56	0	0	0	-	0	0	0	-	-	0	0	-	-	0	0	-
	57	0	0	-	0	0	0	-	-	0	0	-	-	0	0	-	-
	58	0	-	0	0	0	-	-	0	0	-	-	0	0	-	-	-
	59	-	0	0	0	-	-	0	0	-	-	0	0	-	-	-	0
	60	0	0	0	-	0	0	0	-	-	0	0	0	-	0	0	-
	61	0	0	-	0	0	0	-	-	0	0	0	-	0	0	-	-
	62	0	-	0	0	0	-	-	0	0	0	-	0	0	-	-	-
	63	-	0	0	0	-	-	0	0	0	-	0	0	-	-	-	0

Table D.6 (Continued)

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

Table D.7: Differential characteristics for KATAN64 (faulty round t=238)

	Δc_j															
s_{j+t}	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
0	0	0	1	0	0	0	0	0	-	0	-	0	-	-	0	1
1	0	1	0	0	0	0	0	-	0	-	0	-	-	0	1	0
2	1	0	0	0	0	0	-	0	-	0	-	-	0	1	0	-
3	0	0	0	0	0	-	0	-	0	-	-	0	1	0	-	-
4	0	0	0	0	-	0	-	0	-	-	0	1	0	-	-	0
5	0	0	0	-	0	-	0	-	-	0	1	0	-	-	0	-
6	0	0	-	0	-	0	-	-	0	1	0	-	-	0	-	-
7	0	-	0	-	0	-	-	0	1	0	-	-	0	-	-	-
8	-	0	-	-	-	-	0	1	0	-	-	0	-	-	-	0
9	0	-	-	-	-	0	1	0	-	-	0	-	-	-	0	-
10	-	0	0	-	0	1	0	-	-	0	-	0	-	0	-	-
11	0	0	-	0	1	0	-	-	0	-	0	-	0	-	-	-
12	0	-	0	1	0	-	-	0	-	0	-	0	-	-	-	-
13	-	0	1	-	-	-	0	-	0	-	-	-	-	-	-	-
14	0	1	-	-	-	0	-	0	-	-	-	-	-	-	-	-
15	1	0	0	0	0	-	0	0	0	-	-	0	0	-	-	0
16	0	0	0	0	-	0	0	0	-	-	0	0	-	-	0	0
17	0	0	0	-	0	0	0	-	-	0	0	-	-	0	0	-
18	0	0	-	0	0	0	-	-	0	0	-	-	0	0	-	-
19	0	-	0	0	0	-	-	0	0	-	-	0	0	-	-	-
20	-	0	0	-	-	-	0	0	-	-	-	0	-	-	-	0
21	0	0	-	-	-	0	0	-	-	-	0	-	-	-	0	-
22	0	0	-	0	0	0	-	-	0	0	-	-	0	0	-	-
23	0	-	0	0	0	-	-	0	0	-	-	0	0	-	-	-
24	-	0	0	-	-	-	0	0	-	-	-	0	-	-	-	0
25	0	0	-	-	-	0	0	-	-	-	0	-	-	-	0	-
26	0	0	0	0	0	0	-	0	0	0	-	-	0	0	0	-
27	0	0	0	0	0	-	0	0	0	-	-	0	0	0	-	-
28	0	0	0	0	-	0	0	0	-	-	0	0	0	-	-	0
29	0	0	0	-	0	0	0	-	-	0	0	0	-	-	0	-
30	0	0	-	0	0	0	-	-	0	0	0	-	-	0	-	-
31	0	-	0	0	0	-	-	0	0	0	-	-	0	-	-	-
32	-	0	0	-	-	-	0	0	0	-	-	0	-	-	-	0
33	0	0	-	-	-	0	0	0	-	-	0	-	-	-	0	-
34	0	0	0	-	0	0	0	-	-	0	0	0	-	0	0	-
35	0	0	-	0	0	0	-	-	0	0	0	-	0	0	-	-
36	0	-	0	0	0	-	-	0	0	0	-	0	0	-	-	-
37	-	0	0	-	-	-	0	0	0	-	-	0	-	-	-	0
38	0	0	-	-	-	0	0	0	-	-	0	-	-	-	0	-
39	0	-	-	-	0	0	0	-	-	0	-	-	-	0	-	0
40	-	-	-	0	0	0	-	-	0	-	-	-	0	-	0	-
41	-	-	0	0	0	-	-	0	-	-	-	0	-	0	-	0
42	-	0	0	0	-	-	0	-	-	-	0	-	0	-	0	-
43	0	0	0	-	-	0	-	-	-	0	-	0	-	0	-	-
44	0	0	-	-	0	-	-	-	0	-	0	-	0	-	-	-
45	0	-	-	0	-	-	-	0	-	0	-	0	-	-	-	0
46	-	0	0	-	-	-	0	-	0	-	0	1	-	-	0	-
47	0	0	-	-	-	0	-	0	-	0	1	-	-	0	-	-
48	0	-	-	-	0	-	0	-	0	1	-	-	0	-	-	-
49	-	-	-	0	-	0	-	0	1	-	-	0	-	-	-	-
50	-	-	0	-	0	-	0	1	-	-	0	-	-	-	-	-
51	-	0	-	0	-	0	1	-	0	0	0	-	-	0	-	1
52	0	-	0	-	0	1	-	0	0	0	-	-	0	-	1	-
53	-	0	-	0	1	-	0	0	0	-	-	0	-	1	-	-
54	0	-	0	1	-	0	0	0	-	-	0	-	1	-	-	0
55	-	0	0	-	0	0	0	-	0	0	0	1	0	-	0	-
56	0	0	-	0	0	0	-	0	0	0	1	0	-	0	-	0
57	0	-	0	0	0	-	0	0	0	1	0	-	0	-	0	-
58	-	0	0	0	-	0	0	0	1	0	-	0	-	0	-	-
59	0	0	0	-	0	0	0	1	0	-	0	-	0	-	-	-
60	0	0	-	0	0	0	1	0	0	0	0	0	-	0	-	0
61	0	-	0	0	0	1	0	0	0	0	0	-	0	-	0	-
62	-	0	0	0	1	0	0	0	0	0	-	0	-	0	-	-
63	0	0	0	1	0	0	0	0	0	-	0	-	0	-	-	0

Table D.7 (Continued)

	Δc_j															
	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	0	-	-	0	-	-	-	0	-	0	0	0	0	0	0	0
1	-	-	0	-	-	-	0	-	-	0	0	0	0	0	0	0
2	-	0	-	-	-	0	-	-	-	0	0	0	0	0	0	0
3	0	-	-	-	0	-	-	-	-	0	0	0	0	0	0	0
4	-	-	-	0	-	-	-	-	-	0	0	0	0	0	0	0
5	-	-	0	-	-	-	-	-	-	0	0	0	0	0	0	0
6	-	0	-	-	-	-	-	-	-	0	0	0	0	0	0	-
7	0	-	-	-	-	-	-	-	-	0	0	0	0	0	-	0
8	-	-	-	-	-	-	-	-	-	0	0	-	0	-	0	0
9	-	-	-	-	-	-	-	-	-	0	-	0	-	0	0	0
10	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
11	-	-	-	0	-	-	-	-	-	0	0	0	0	0	0	-
12	-	-	0	-	-	-	-	-	-	0	0	0	0	0	-	0
13	-	0	-	-	-	-	-	-	-	0	0	-	0	-	0	0
14	0	-	-	-	-	-	-	-	-	0	-	0	-	0	0	0
15	0	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
16	-	-	-	0	-	-	-	0	-	0	0	0	0	0	0	0
17	-	-	0	-	-	-	0	-	-	0	0	0	0	0	0	0
18	-	0	-	-	-	0	-	-	-	0	0	0	0	0	0	-
19	0	-	-	-	0	-	-	-	0	0	0	0	0	0	-	0
20	-	-	-	0	-	-	-	0	-	0	0	-	0	-	0	0
21	-	-	0	-	-	-	-	-	-	0	-	0	-	0	0	0
22	-	0	-	-	-	0	-	-	-	0	0	0	0	0	0	-
23	0	-	-	-	0	-	-	-	0	0	0	0	0	0	-	0
24	-	-	-	0	-	-	-	0	-	0	0	1	0	-	0	0
25	-	-	0	-	-	-	-	-	-	0	1	0	-	0	0	0
26	-	0	-	-	-	0	-	-	-	0	0	0	0	0	0	0
27	0	-	-	-	0	-	-	-	-	0	0	0	0	0	0	0
28	-	-	-	0	-	-	-	-	-	0	0	0	0	0	0	0
29	-	-	0	-	-	-	-	-	-	0	0	0	0	0	0	0
30	-	0	-	-	-	-	-	-	-	0	0	0	0	0	0	-
s_{j+t} 31	0	-	-	-	-	-	-	-	-	0	0	0	0	0	-	0
32	-	-	-	-	-	-	-	-	-	0	0	-	0	-	0	0
33	-	-	-	-	-	-	-	-	-	0	-	0	-	0	0	0
34	-	-	0	-	0	-	0	1	-	0	0	0	0	0	0	0
35	-	0	-	0	-	0	-	-	-	0	0	0	0	0	0	-
36	0	-	0	-	0	-	-	-	0	0	0	0	0	0	-	0
37	-	0	-	0	-	-	-	0	-	0	0	1	0	-	0	0
38	0	-	0	-	-	-	-	-	-	0	1	0	-	0	0	0
39	-	0	-	-	-	-	-	-	-	1	0	-	0	0	0	1
40	0	-	-	-	-	-	-	-	-	0	-	0	0	0	1	0
41	-	-	-	0	-	-	-	-	-	-	0	0	0	1	0	0
42	-	-	0	-	-	-	-	-	-	0	0	0	1	0	0	0
43	-	0	-	-	-	-	-	-	-	0	0	1	0	0	0	0
44	0	-	-	-	-	-	-	-	-	0	1	0	0	0	0	-
45	-	-	-	-	-	-	-	-	-	1	0	0	0	0	-	0
46	-	-	-	-	-	-	-	-	-	0	0	0	0	-	0	0
47	-	-	-	-	-	-	-	-	-	0	0	0	-	0	0	0
48	-	-	-	-	-	-	-	-	-	0	0	-	0	0	0	1
49	-	-	-	-	-	-	-	-	-	0	-	0	0	0	1	0
50	-	-	-	-	-	-	-	-	-	-	0	0	0	1	0	0
51	-	-	0	-	-	-	-	-	-	0	0	0	1	0	0	0
52	-	0	-	0	-	-	-	-	-	0	0	1	0	0	0	0
53	0	-	0	-	-	-	-	-	-	0	1	0	0	0	0	0
54	-	0	-	-	-	-	-	-	-	1	0	0	0	0	0	0
55	0	-	-	-	-	-	-	-	1	0	0	0	0	0	0	0
56	-	-	-	-	-	-	-	1	-	0	0	0	0	0	0	0
57	-	-	-	-	-	-	1	-	-	0	0	0	0	0	0	0
58	-	-	-	-	-	1	-	-	-	0	0	0	0	0	0	0
59	-	-	-	-	1	-	-	-	0	0	0	0	0	0	0	0
60	-	-	0	1	0	-	-	0	-	0	0	0	0	0	0	0
61	-	0	1	0	-	-	0	-	-	0	0	0	0	0	0	0
62	0	1	0	-	-	0	-	-	-	0	0	0	0	0	0	0
63	1	0	-	-	0	-	-	-	0	0	0	0	0	0	0	0

Table D.7 (Continued)

	Δc_j															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	-	0	0	0	-	-	0	0	0	-	0
1	0	0	0	0	-	0	0	0	-	-	0	0	0	-	0	0
2	0	0	0	-	0	0	0	-	-	0	0	0	-	0	0	-
3	0	0	-	0	0	0	-	-	0	0	0	-	0	0	-	-
4	0	-	0	0	0	-	-	0	0	0	-	0	0	-	-	-
5	-	0	0	0	-	-	0	0	0	-	0	0	-	-	-	0
6	0	0	0	-	-	0	0	0	-	0	0	-	-	-	0	-
7	0	0	-	-	0	0	0	-	0	0	-	-	-	0	-	0
8	0	-	-	0	0	0	-	0	0	-	-	-	0	-	0	-
9	-	-	0	0	0	-	0	0	-	-	-	0	-	0	-	0
10	-	0	0	0	-	0	0	-	0	-	0	-	0	-	0	1
11	0	0	0	-	0	0	-	0	-	0	-	0	-	0	1	-
12	0	0	-	0	0	-	0	-	0	-	0	-	0	1	-	0
13	0	-	0	0	-	0	-	0	-	0	-	0	1	-	0	0
14	-	0	0	-	0	-	0	-	0	-	0	1	-	0	0	-
15	0	0	-	0	0	0	-	0	0	0	1	-	0	0	-	-
16	0	-	0	0	0	-	0	0	0	1	-	0	0	-	-	0
17	-	0	0	0	-	0	0	0	1	-	0	0	-	-	0	0
18	0	0	0	-	0	0	0	1	-	0	0	-	-	0	0	-
19	0	0	-	0	0	0	1	-	0	0	-	-	0	0	-	-
20	0	-	0	0	0	1	-	0	0	-	-	0	0	-	-	0
21	-	0	0	0	1	-	0	0	-	-	0	0	-	-	0	0
22	0	0	0	1	0	0	0	-	-	0	0	-	-	0	0	0
23	0	0	1	0	0	0	-	-	0	0	-	-	0	0	0	-
24	0	1	0	0	0	-	-	0	0	-	-	0	0	0	-	0
25	1	0	0	0	-	-	0	0	-	-	0	0	0	-	0	0
26	0	0	0	-	0	0	0	-	-	0	0	0	-	0	0	0
27	0	0	-	0	0	0	-	-	0	0	0	-	0	0	0	-
28	0	-	0	0	0	-	-	0	0	0	-	0	0	0	-	-
29	-	0	0	0	-	-	0	0	0	-	0	0	0	-	-	0
30	0	0	0	-	-	0	0	0	-	0	0	0	-	-	0	0
31	0	0	-	-	0	0	0	-	0	0	0	-	-	0	0	0
s_{j+t} 32	0	-	-	0	0	0	-	0	0	0	-	-	0	0	0	1
33	-	-	0	0	0	-	0	0	0	-	-	0	0	0	1	0
34	-	0	0	0	1	0	0	0	0	-	0	0	0	1	0	0
35	0	0	0	1	0	0	0	0	-	0	0	0	1	0	0	0
36	0	0	1	0	0	0	0	-	0	0	0	1	0	0	0	0
37	0	1	0	0	0	0	-	0	0	0	1	0	0	0	0	0
38	1	0	0	0	0	-	0	0	0	1	0	0	0	0	0	0
39	0	0	0	0	-	0	0	0	1	0	0	0	0	0	0	-
40	0	0	0	-	0	0	0	1	0	0	0	0	0	0	-	0
41	0	0	-	0	0	0	1	0	0	0	0	0	0	-	0	-
42	0	-	0	0	0	1	0	0	0	0	0	0	-	0	-	0
43	-	0	0	0	1	0	0	0	0	0	0	-	0	-	0	-
44	0	0	0	1	0	0	0	0	0	0	-	0	-	0	-	-
45	0	0	1	0	0	0	0	0	0	-	0	-	0	-	-	-
46	0	1	0	0	0	0	0	0	0	0	-	0	0	0	-	-
47	1	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0
48	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0
49	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0	-
50	0	0	0	0	0	0	-	0	0	0	-	-	0	0	-	-
51	0	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0
52	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0
53	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0	0
54	0	0	0	0	0	0	-	0	0	0	-	-	0	0	0	-
55	0	0	0	0	0	0	0	0	0	0	-	0	0	0	-	-
56	0	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0
57	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0
58	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0	-
59	0	0	0	0	0	0	-	0	0	0	-	-	0	0	-	-
60	0	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0
61	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0
62	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0	0
63	0	0	0	0	0	0	-	0	0	0	-	-	0	0	0	-

Table D.7 (Continued)

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

Table D.8: Differential characteristics for KATAN64 (faulty round t=242)

	Δc_j															
s_{j+t}	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
0	0	0	0	-	0	0	0	0	0	0	-	0	0	0	1	0
1	0	0	-	0	0	0	0	0	0	-	0	0	0	1	0	0
2	0	-	0	0	0	0	0	0	-	0	0	0	1	0	0	0
3	-	0	0	0	0	0	0	-	0	0	0	1	0	0	0	0
4	0	0	0	0	0	0	-	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	-	0	0	0	1	0	0	0	0	0	-
6	0	0	0	0	-	0	0	0	1	0	0	0	0	0	-	0
7	0	0	0	-	0	0	0	1	0	0	0	0	0	-	0	-
8	0	0	-	0	0	0	1	0	0	0	0	0	-	0	-	0
9	0	-	0	0	0	1	0	0	0	0	0	-	0	-	0	-
10	-	0	0	0	1	0	0	0	0	0	0	0	-	0	0	-
11	0	0	0	1	0	0	0	0	0	0	0	-	0	0	-	0
12	0	0	1	0	0	0	0	0	0	0	-	0	0	-	0	1
13	0	1	0	0	0	0	0	0	0	-	0	0	-	0	1	0
14	1	0	0	0	0	0	0	0	-	0	0	-	0	1	0	-
15	0	0	0	0	0	0	0	-	0	0	0	0	1	0	0	0
16	0	0	0	0	0	0	-	0	0	0	0	1	0	0	0	0
17	0	0	0	0	0	-	0	0	0	0	1	0	0	0	0	-
18	0	0	0	0	-	0	0	0	0	1	0	0	0	0	-	0
19	0	0	0	-	0	0	0	0	1	0	0	0	0	-	0	0
20	0	0	-	0	0	0	0	1	0	0	0	0	-	0	0	0
21	0	-	0	0	0	0	1	0	0	0	0	-	0	0	0	-
22	-	0	0	0	0	1	0	0	0	0	0	0	0	0	-	0
23	0	0	0	0	1	0	0	0	0	0	0	0	0	-	0	0
24	0	0	0	1	0	0	0	0	0	0	0	0	-	0	0	0
25	0	0	1	0	0	0	0	0	0	0	0	-	0	0	0	-
26	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	-
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	-
39	0	0	0	0	0	0	0	0	0	0	-	0	0	0	-	-
40	0	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0
41	0	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0
42	0	0	0	0	0	0	0	-	0	0	0	-	-	0	0	0
43	0	0	0	0	0	0	-	0	0	0	-	-	0	0	0	-
44	0	0	0	0	0	-	0	0	0	-	-	0	0	0	-	0
45	0	0	0	0	-	0	0	0	-	-	0	0	0	-	0	0
46	0	0	0	-	0	0	0	-	-	0	0	0	-	0	0	-
47	0	0	-	0	0	0	-	-	0	0	0	-	0	0	-	-
48	0	-	0	0	0	-	-	0	0	0	-	0	0	-	-	-
49	-	0	0	0	-	-	0	0	0	-	0	0	-	-	-	0
50	0	0	0	-	-	0	0	0	-	0	0	-	-	-	0	-
51	0	0	-	0	0	0	0	-	0	0	0	-	-	0	-	0
52	0	-	0	0	0	0	-	0	0	0	-	-	0	-	0	-
53	-	0	0	0	0	-	0	0	0	-	-	0	-	0	-	0
54	0	0	0	0	-	0	0	0	-	-	0	-	0	-	0	1
55	0	0	0	-	0	0	0	-	-	0	0	0	-	0	0	-
56	0	0	-	0	0	0	-	-	0	0	0	-	0	0	-	0
57	0	-	0	0	0	-	-	0	0	0	-	0	0	-	0	0
58	-	0	0	0	-	-	0	0	0	-	0	0	-	0	0	0
59	0	0	0	-	-	0	0	0	-	0	0	-	0	0	0	-
60	0	0	-	0	0	0	0	-	0	0	0	0	0	0	-	0
61	0	-	0	0	0	0	-	0	0	0	0	0	0	-	0	0
62	-	0	0	0	0	-	0	0	0	0	0	0	-	0	0	0
63	0	0	0	0	-	0	0	0	0	0	0	-	0	0	0	1

Table D.8 (Continued)

	Δc_j															
s_{j+t}	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	0	0	0	0	-	0	-	0	-	0	0	1	0	0	0	0
1	0	0	0	-	0	-	-	-	-	0	1	0	0	0	0	0
2	0	0	-	0	-	-	-	-	0	1	0	0	0	0	0	0
3	0	-	0	-	-	-	-	0	1	0	0	0	0	0	0	0
4	-	0	-	0	-	-	0	1	0	0	0	0	0	0	0	0
5	0	-	0	-	-	0	1	0	-	0	0	0	0	0	0	0
6	-	0	-	-	0	1	0	-	-	0	0	0	0	0	0	0
7	0	-	-	0	1	0	-	-	0	0	0	0	0	0	0	0
8	-	-	0	1	0	-	-	0	-	0	0	0	0	0	0	0
9	-	0	1	0	-	-	0	-	-	0	0	0	0	0	0	0
10	0	1	0	-	-	0	-	0	-	0	0	0	0	0	0	0
11	1	0	-	-	0	-	0	-	0	0	0	0	0	0	0	0
12	0	-	-	0	-	0	-	0	-	0	0	0	0	0	0	0
13	-	-	0	-	0	-	0	-	-	0	0	0	0	0	0	0
14	-	0	-	0	-	0	-	-	-	0	0	0	0	0	0	0
15	0	-	0	0	0	-	-	0	0	0	0	0	0	0	0	0
16	-	0	0	0	-	-	0	0	-	0	0	0	0	0	0	0
17	0	0	0	-	-	0	0	-	-	0	0	0	0	0	0	0
18	0	0	-	-	0	0	-	-	0	0	0	0	0	0	0	0
19	0	-	-	0	0	-	-	0	0	0	0	0	0	0	0	0
20	-	-	0	0	-	-	-	0	-	0	0	0	0	0	0	0
21	-	0	0	-	-	-	0	-	-	0	0	0	0	0	0	0
22	0	0	-	-	-	0	-	-	0	0	0	0	0	0	0	0
23	0	-	-	0	0	-	-	0	0	0	0	0	0	0	0	0
24	-	-	0	0	-	-	0	0	-	0	0	0	0	0	0	0
25	-	0	0	-	-	0	-	-	-	0	0	0	0	0	0	0
26	0	0	-	0	0	-	-	-	0	0	0	0	0	0	0	0
27	0	-	0	0	-	-	-	0	0	0	0	0	0	0	0	0
28	-	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0
29	0	0	0	-	-	0	0	0	-	0	0	0	0	0	0	0
30	0	0	-	-	0	0	0	-	-	0	0	0	0	0	0	0
31	0	-	-	0	0	0	-	-	0	0	0	0	0	0	0	0
32	-	-	0	0	0	-	-	0	-	0	0	0	0	0	0	0
33	-	0	0	0	-	-	0	-	-	0	0	0	0	0	0	0
34	0	0	0	-	-	0	0	0	-	0	0	0	0	0	0	0
35	0	0	-	-	0	0	0	-	0	0	0	0	0	0	0	0
36	0	-	-	0	0	0	-	0	0	0	0	0	0	0	0	0
37	-	-	0	0	0	-	0	0	-	0	0	0	0	0	0	0
38	-	0	0	0	-	0	0	-	-	0	0	0	0	0	0	0
39	0	0	0	-	0	0	-	-	-	0	0	0	0	0	0	0
40	0	0	-	0	0	-	-	-	0	0	0	0	0	0	0	0
41	0	-	0	0	-	-	-	0	-	0	0	0	0	0	0	0
42	-	0	0	-	-	-	0	-	0	0	0	0	0	0	0	0
43	0	0	-	-	-	0	-	0	-	0	0	0	0	0	0	0
44	0	-	-	-	0	-	0	-	0	0	0	0	0	0	0	0
45	-	-	-	0	-	0	-	0	1	0	0	0	0	0	0	0
46	-	-	0	-	0	-	0	1	-	0	0	0	0	0	0	0
47	-	0	-	0	-	0	-	-	-	0	0	0	0	0	0	-
48	0	-	0	-	0	-	-	-	0	0	0	0	0	0	-	0
49	-	0	-	0	-	-	-	0	-	0	0	0	0	-	0	0
50	0	-	0	1	-	-	0	-	-	0	0	0	-	0	0	0
51	-	0	1	-	0	0	-	-	-	0	0	0	0	0	0	1
52	0	1	-	0	0	-	-	-	0	0	0	0	0	0	1	0
53	1	-	0	0	-	-	-	0	-	0	0	0	0	1	0	0
54	-	0	0	0	-	-	0	-	1	0	0	0	1	0	0	0
55	0	0	0	-	0	0	0	1	0	0	0	0	0	0	0	0
56	0	0	-	0	0	0	-	0	-	0	0	0	0	0	0	-
57	0	-	0	0	0	-	0	-	0	0	0	0	0	0	-	0
58	-	0	0	0	-	0	-	0	-	0	0	0	0	-	0	0
59	0	0	0	1	0	-	0	-	0	0	0	0	-	0	0	0
60	0	0	1	0	0	0	-	0	-	0	0	0	0	0	0	1
61	0	1	0	0	0	-	0	-	0	0	0	0	0	0	1	0
62	1	0	0	0	-	0	-	0	-	0	0	0	0	1	0	0
63	0	0	0	0	0	-	0	-	0	0	0	0	1	0	0	0

Table D.8 (Continued)

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

Table D.8 (Continued)

		Δc_j															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s_{j+t}	0	0	-	0	0	0	-	-	0	0	0	-	-	0	-	-	-
	1	-	0	0	-	-	-	0	0	0	-	-	0	-	-	-	0
	2	0	0	-	-	-	0	0	0	-	0	0	-	-	-	0	-
	3	0	-	-	-	0	0	0	-	0	0	-	-	-	0	-	0
	4	0	-	-	0	0	0	-	0	0	-	-	-	0	-	0	-
	5	-	-	0	0	0	-	0	0	-	-	-	0	-	0	-	0
	6	-	0	0	0	-	0	0	-	-	-	0	-	0	-	0	1
	7	0	0	0	-	0	0	-	-	-	0	-	0	-	0	1	-
	8	0	0	-	-	0	-	-	-	0	-	0	-	0	1	-	-
	9	0	-	-	0	-	-	-	0	-	0	-	0	-	-	-	0
	10	-	-	0	-	0	-	0	-	0	-	0	1	-	0	0	-
	11	0	0	-	0	-	0	-	0	-	0	1	-	-	0	-	-
	12	0	-	0	-	0	-	0	-	0	1	-	-	0	-	-	0
	13	-	0	-	0	-	0	-	0	1	-	-	0	-	-	0	-
	14	0	-	0	-	0	-	0	1	-	0	0	-	-	0	-	-
	15	0	0	-	0	0	0	1	-	0	0	-	-	0	0	-	-
	16	0	-	0	0	0	1	-	0	0	-	-	0	1	-	-	0
	17	-	0	0	0	1	-	0	0	-	-	0	1	-	-	0	0
	18	0	0	0	1	-	0	0	-	-	0	1	-	-	0	0	0
	19	0	0	1	-	0	0	-	-	0	0	-	-	0	0	0	-
	20	0	1	-	-	0	-	-	0	0	-	-	0	0	0	-	-
	21	1	-	-	0	-	-	0	0	-	-	0	0	-	-	-	0
	22	0	-	0	-	-	0	0	-	-	0	0	0	-	0	0	0
	23	0	0	-	-	0	0	-	-	0	0	0	-	0	0	0	-
	24	0	-	-	0	0	-	-	0	0	0	-	0	0	0	-	-
	25	-	-	0	1	-	-	0	0	0	-	0	0	-	-	-	0
	26	0	0	1	-	-	0	0	0	-	0	0	0	-	-	0	0
	27	0	1	-	-	0	0	0	-	0	0	0	-	-	0	0	0
	28	0	-	-	0	0	0	-	0	0	0	-	-	0	0	0	1
	29	-	-	0	0	0	-	0	0	0	-	-	0	0	0	1	0
	30	-	0	0	0	-	0	0	0	-	-	0	0	0	1	0	0
	31	0	0	0	-	0	0	0	-	-	0	0	0	1	0	0	0
	32	0	0	-	0	0	0	-	-	0	0	0	1	0	0	0	-
	33	0	-	0	0	0	-	-	0	0	0	1	0	-	0	-	0
	34	1	0	0	0	0	-	0	0	0	1	0	0	0	0	0	0
	35	0	0	0	0	-	0	0	0	1	0	0	0	0	0	0	0
	36	0	0	0	-	0	0	0	1	0	0	0	0	0	0	0	0
	37	0	0	-	0	0	0	1	0	0	0	0	0	0	0	0	-
	38	0	-	0	0	0	1	0	0	0	0	0	0	-	0	-	0
	39	-	0	0	0	1	0	0	0	0	0	0	-	0	-	0	0
	40	0	0	0	1	0	0	0	0	0	0	-	0	-	0	0	0
	41	0	0	1	0	0	0	0	0	0	0	0	-	0	0	0	-
	42	0	1	0	0	0	0	0	0	0	0	-	0	-	0	-	-
	43	1	0	0	0	0	0	0	0	0	-	0	-	-	-	-	0
	44	0	0	0	0	0	0	0	0	-	0	-	-	-	-	0	0
	45	0	0	0	0	0	0	0	-	0	0	-	-	-	0	0	-
	46	0	0	0	0	0	0	-	0	0	0	-	-	0	0	-	-
	47	0	0	0	-	0	-	0	0	0	-	-	0	-	-	-	0
	48	0	0	-	0	-	0	0	0	-	-	0	-	-	-	0	0
	49	0	-	0	-	0	0	0	-	-	0	-	-	-	0	0	-
	50	0	0	-	0	0	0	-	-	0	0	-	-	-	0	-	-
	51	0	0	0	-	0	-	0	0	0	-	-	0	-	0	-	-
	52	0	0	-	0	-	0	0	0	-	-	0	-	-	-	-	0
	53	0	-	0	-	0	0	0	-	-	0	-	-	-	-	0	-
	54	0	0	-	0	0	0	-	-	0	0	-	-	-	0	-	-
	55	0	0	0	0	0	0	-	0	0	0	-	-	0	0	-	-
	56	0	0	0	-	0	-	0	0	0	-	-	0	-	-	-	0
	57	0	0	-	0	-	0	0	0	-	-	0	-	-	-	0	0
	58	0	-	0	-	0	0	0	-	-	0	-	-	-	0	0	-
	59	0	0	-	0	0	0	-	-	0	0	-	-	-	0	-	-
	60	0	0	0	-	0	-	0	0	0	-	-	0	0	0	-	0
	61	0	0	-	0	-	0	0	0	-	-	0	0	0	-	0	0
	62	0	-	0	-	0	0	0	-	-	0	0	0	-	0	0	-
	63	0	0	-	0	0	0	-	-	0	0	0	-	-	0	-	-

Table D.9: Differential characteristics for KATAN64 (faulty round t=246)

		Δc_j															
		63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
s_{j+t}	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	-
	1	0	0	0	0	0	0	0	0	0	-	0	0	0	0	-	0
	2	0	0	0	0	0	0	0	0	-	0	0	0	0	-	0	0
	3	0	0	0	0	0	0	0	-	0	0	0	0	-	0	0	0
	4	0	0	0	0	0	0	-	0	0	0	0	-	0	0	0	0
	5	0	0	0	0	0	-	0	0	0	0	-	0	0	0	0	0
	6	0	0	0	0	-	0	0	0	0	-	0	0	0	0	0	0
	7	0	0	0	-	0	0	0	0	-	0	0	0	0	0	0	-
	8	0	0	-	0	0	0	0	-	0	0	0	0	0	0	-	0
	9	0	-	0	0	0	0	-	0	0	0	0	0	0	-	0	0
	10	0	0	0	0	0	-	0	0	0	0	0	0	-	0	0	0
	11	0	0	0	0	-	0	0	0	0	0	0	-	0	0	0	1
	12	0	0	0	-	0	0	0	0	0	0	-	0	0	0	1	0
	13	0	0	-	0	0	0	0	0	0	-	0	0	0	1	0	0
	14	0	-	0	0	0	0	0	0	-	0	0	0	1	0	0	0
	15	0	0	0	0	0	0	0	-	0	0	0	1	0	0	0	0
	16	0	0	0	0	0	0	-	0	0	0	1	0	0	0	0	0
	17	0	0	0	0	0	-	0	0	0	1	0	0	0	0	0	0
	18	0	0	0	0	-	0	0	0	1	0	0	0	0	0	0	0
	19	0	0	0	-	0	0	0	1	0	0	0	0	0	0	0	-
	20	0	0	-	0	0	0	1	0	0	0	0	0	0	0	-	0
	21	0	-	0	0	0	1	0	0	0	0	0	0	0	-	0	0
	22	0	0	0	0	1	0	0	0	0	0	0	0	-	0	0	0
	23	0	0	0	1	0	0	0	0	0	0	0	-	0	0	0	0
	24	0	0	1	0	0	0	0	0	0	0	-	0	0	0	0	1
	25	0	1	0	0	0	0	0	0	0	-	0	0	0	0	1	0
	26	0	0	0	0	0	0	0	0	-	0	0	0	0	1	0	0
	27	0	0	0	0	0	0	0	-	0	0	0	0	1	0	0	0
	28	0	0	0	0	0	0	-	0	0	0	0	1	0	0	0	0
	29	0	0	0	0	0	-	0	0	0	0	1	0	0	0	0	0
	30	0	0	0	0	-	0	0	0	0	1	0	0	0	0	0	0
	31	0	0	0	-	0	0	0	0	1	0	0	0	0	0	0	0
	32	0	0	-	0	0	0	0	1	0	0	0	0	0	0	0	0
	33	0	-	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	34	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	35	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	36	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	37	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	38	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	39	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
	47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
	48	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
	49	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
	50	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	-
	51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
	52	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
	53	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
	54	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
	55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
	57	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
	58	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
	59	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	-
	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
	61	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
	62	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
	63	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0

Table D.9 (Continued)

	Δc_j															
	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	-	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	-	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	-	0	0	0	1	0	0	0	0	0	0	0	0	0
5	0	-	0	0	0	1	0	0	0	0	0	0	0	0	0	0
6	-	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	-	0	-	0	0	0	0	0	0	1
9	0	1	0	0	0	-	0	-	0	0	0	0	0	0	1	0
10	1	0	0	0	0	0	0	0	-	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	-	0	0	0	0	1	0	0	0
12	0	0	0	0	0	0	-	0	0	0	0	1	0	0	0	0
13	0	0	0	0	0	-	-	0	-	0	1	0	0	0	0	0
14	0	0	0	0	-	-	0	-	0	1	0	0	0	0	0	0
15	0	0	0	-	0	0	0	0	1	0	0	0	0	0	0	0
16	0	0	-	0	0	0	0	1	0	0	0	0	0	0	0	0
17	0	-	0	0	0	0	1	0	0	0	0	0	0	0	0	0
18	-	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	1	0	0	-	0	-	0	0	0	0	0	0	0
21	0	0	1	0	0	-	0	-	0	0	0	0	0	0	0	0
22	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	-	0	-	0	0	0	0	0	0	0
25	0	0	0	0	0	-	0	-	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_{j+t} 31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	-	0	-	0	0	0	0	0	0	0
33	0	0	0	0	0	-	0	-	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	-	0	-	0	0	0	0	0	0	0
38	0	0	0	0	0	-	0	-	0	0	0	0	0	0	0	0
39	0	0	0	0	-	0	-	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	-	0	0	0	-	0	0	0	0	0	0	0
42	0	0	0	-	0	0	0	-	-	0	0	0	0	0	0	0
43	0	0	-	0	0	0	-	-	0	0	0	0	0	0	0	0
44	0	-	0	0	0	-	-	0	0	0	0	0	0	0	0	0
45	-	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0
46	0	0	0	-	-	0	0	0	-	0	0	0	0	0	0	0
47	0	0	-	-	0	0	0	-	0	0	0	0	0	0	0	0
48	0	-	-	0	0	0	-	0	0	0	0	0	0	0	0	0
49	-	-	0	0	0	-	0	0	-	0	0	0	0	0	0	0
50	-	0	0	0	-	0	0	-	-	0	0	0	0	0	0	0
51	0	0	0	-	0	0	0	-	-	0	0	0	0	0	0	0
52	0	0	-	0	0	0	-	-	0	0	0	0	0	0	0	0
53	0	-	0	0	0	-	-	0	-	0	0	0	0	0	0	0
54	-	0	0	0	-	-	0	-	0	0	0	0	0	0	0	0
55	0	0	0	-	-	0	0	0	-	0	0	0	0	0	0	0
56	0	0	-	-	0	0	0	-	0	0	0	0	0	0	0	0
57	0	-	-	0	0	0	-	0	0	0	0	0	0	0	0	0
58	-	-	0	0	0	-	0	0	-	0	0	0	0	0	0	0
59	-	0	0	0	-	0	0	-	0	0	0	0	0	0	0	0
60	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0
61	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0
62	0	-	0	0	0	0	0	0	-	0	0	0	0	0	0	0
63	-	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0

Table D.9 (Continued)

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

Table D.9 (Continued)

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

Table D.10: Differential characteristics for KATAN64 (faulty round t=250)

	Δc_j															
s_{j+t}	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
48	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
49	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
50	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
51	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table D.10 (Continued)

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

Table D.10 (Continued)

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

Table D.10 (Continued)

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