Integral Cryptanalysis against Symmetric-Key Cryptosystems

Doctoral Thesis

Integral Cryptanalysis against Symmetric-Key Cryptosystems
共通鍵暗号に対するインテグラル攻撃

January 2017

Graduate School of Engineering,
Kobe University

Yosuke Todo

Acknowledgment

I would like to thank everyone who contributed to this thesis. First of all, I would like to express my sincere gratitude to my superior Prof. Masakatu Morii for introducing me to cryptology as my research topic. His guidance throughout my research helps me to establish the basis of my career in this field. I also would like to thank the judging committee members, Prof. Hiromasa Takeno, Prof. Chikara Ohta, and Associate Prof. Yoshiaki Shiraishi for reviewing this manuscript and giving me valuable feedbacks. Moreover, I would like to thank Associate Prof. Yoshiaki Shiraishi and Assistant Prof. Yasuhiro Takano for giving me helpful advice and supporting my research in Kobe University.

I would like to thank Prof. Hidenori Kuwakado and Associate Prof. Minoru Kuribayashi for giving me helpful advice and supporting my research when I was bachelor and master student in Kobe University. Moreover, I would like to thank Junior Associate Prof. Toshihiro Ohigashi. I have learned much from him when I start to tackle cryptology as my research topic. I am grateful to all the members of my research in Kobe University: Dr. Takanori Isobe, Yuhei Watanabe, Sho Sakikoyama, Yuki Funabiki, and Takeru Koie.

I would like to thank Dr. Kazumaro Aoki, Dr. Kan Yasuda, and Dr. Yu Sasaki. I joined NTT Corporation in 2012, but I did not have enough ability as a researcher. They taught me a lot of things, not only about symmetric-key cryptosystems but also basic things like how to write academic articles, find and approach an interesting research topics. They guided me in a good direction through their collaboration. I am also thankful to all the members in same research group in NTT for constant support.

I greatly appreciate my parents, grandparents, brother and sister for their longtime support. Finally, I express deep gratitude to my wife Ai for her constant encouragement and unconditional help. I would never be possible to write this thesis without her.

Abstract

Cryptosystems is the fundamental technology for the information security. The *cryptology* is one of academical research topics, and the motivation is to support the security of various ongoing services, to design more secure and efficient cryptosystems, and to seek new application of *cryptosystems* for reduction of security risks. Nowadays, a number of information systems and computer networks ensure the security based on cryptosystems.

There are two types of cryptosystems; one is a public-key cryptosystem and another is a symmetric-key cryptosystem. They have its advantages and drawbacks. The public-key cryptosystem encrypts message by using the public key, and the encrypted message (ciphertext) is decrypted by the corresponding secret key. We do not need to share the secret key in advance, but it is unsuitable to handle large data because the processing is not fast. The symmetric-key cryptosystem has to shared the same key in advance. However, the processing is very faster than the public-key cryptosystem. Therefore, we first share the secret key for the symmetric-key cryptosystem by using the public-key cryptosystem and then handle large data by using the symmetric-key cryptosystem.

Cryptanalyses are one of the most important topics in cryptology. Only after many cryptographers analyze the security of cryptosystems and show supporting evidence that the cryptosystem is secure, it is admitted widely used and standardized. Moreover, the knowledge of cryptanalyses contributes to designing more secure new cryptosystem. In fact, cryptology has been developed by iterating designs and analyses. If there are no third-party cryptanalysis and the security is guaranteed only by the designer intuition, it is too dangerous to use such cryptosystems. For example, if one nation designs cryptosystem with trapdoor and they promote its use, it will lead to mass-surveillance society. Therefore, we have to continue the activity of cryptanalyses to avoid such society.

We focus on the cryptanalytic techniques for symmetric-key cryptosystems. Many such techniques have been developed in last three decades. The most famous technique is the *differential cryptanalysis* by Biham and Shamir at 1990 and the *linear cryptanalysis* by Matsui at 1993. Both cryptanalyses were developed to analyze Data Encryption Standard (DES). After their proposals, many symmetric-key cryptosystems which are secure against both cryptanalyses have been proposed. Similarly, many new advanced cryptanalytic techniques have also been developed, e.g., the *higher-order differential cryptanalysis*, *impossible differential cryptanalysis*, *truncated differential cryptanalysis*, *integral cryptanalysis*, *multi-dimensional linear cryptanalysis*, and *zero-correlation linear cryptanalysis*.

In this doctor thesis, we propose many novel techniques for the integral cryptanalysis. The integral cryptanalysis is one of the most powerful cryptanalytic techniques. In a broad sense, attackers recover the secret key by analyzing the pair of chosen-plaintext and corresponding-ciphertext sets. In a narrow sense, the integral cryptanalysis first constructs an integral characteristic and then recovers the secret key by using the characteristic. In the integral characteristic, attackers first prepare N chosen plaintexts. If the XOR of all corresponding texts encrypted by r rounds is 0 for all keys, we say that the cipher has an r-round integral characteristic with N chosen plaintexts. Namely, we prepare the plaintext set $\mathbb{X}$ satisfying $\sum_{p \in \mathbb{X}} T(E_k(p)) = 0$ for all round key k, where the function T is a simple truncation function and E_k denotes the target block cipher whose number of rounds is reduced to r rounds. The probability that above equation holds for ideal ciphers is negligible. Therefore, we can execute a distinguishing attack by construction the integral characteristic. The integral cryptanalysis additionally appends a key recovery step to the r-round integral characteristic. Attackers guess round keys used in the last s rounds and attack $(r + s)$ rounds. If guessed round keys are correct, the sum is always 0. Therefore, if the sum is not 0, the guessed round key is incorrect. By repeating this procedure,

attackers recover the correct round keys used in the last s rounds.

Our contribution mainly consists of two parts; one part is supporting advanced techniques for the integral cryptanalysis, and another part is the development of the *division property*.

Improved Techniques for Integral Cryptanalysis

We first propose generic attacks against several Feistel networks. Such a topic has been discussed since the introduction of Luby-Rackoff construction in 1988. The Luby-Rackoff construction is unrealistic because its round functions must be chosen at random from the set of all functions. We focus on the security of more practical constructions, which are indeed used by some Feistel ciphers in practice. We show new properties using the relationship between the pair of plaintext and ciphertext sets. We propose new generic key recovery attacks by using our properties, and confirm the feasibility by implementing the attack on Feistel ciphers with small block sizes. As a result, we conclude that efficient and practical 6-round Feistel networks are not secure.

We next propose a new technique for the integral cryptanalysis called the *Fast Fourier Transform (FFT) key recovery*. When N chosen plaintexts are required for the integral cryptanalysis and the guessed key is κ bits, a straightforward key recovery requires the time complexity of $O(N2^{\kappa})$. However, our FFT key recovery only requires the time complexity of $O(N + \kappa 2^{\kappa})$. We apply the FFT key recovery technique to three structures, an Even-Mansour scheme, key-alternating cipher, and Feistel structure. As a result, we improve the time complexity for integral cryptanalyses against specific ciphers.

Division Property: Efficient Method to Estimate Upper Bound of Algebraic Degree

How to find effective integral characteristics is the most importance procedure for the integral cryptanalysis. There are two famous methods to find such characteristic: one is the propagation of integral property and anther is the estimation of upper bound of algebraic degree. They have its advantages and drawbacks. While the integral property can mainly exploit the *diffusion part* of block ciphers, the degree estimation can mainly exploit the *confusion part* of block ciphers. The division property is developed by combining the advantages of the integral property and degree estimation. The division property can find more accurate integral characteristics than conventional two methods. To show its advantage, we search for integral characteristics on Feistel, SPN, and AES-like structures using the division property. As a result, we can find improved integral characteristics for all structures.

The usefulness of the division property is not limited to generic attacks. We apply this technique to MISTY1. MISTY1 is a block cipher designed by Matsui in 1997, and it was well evaluated and standardized by projects, such as CRYPTREC, ISO/IEC, and NESSIE. We propose a key recovery attack on the full MISTY1, i.e., we show that 8-round MISTY1 with 5 FL layers does not have 128-bit security. Many attacks against MISTY1 have been proposed, but there is no attack against the full MISTY1. Therefore, our attack is the first cryptanalysis against the full MISTY1. We construct a new integral characteristic by using the propagation of the division property. As a result, we construct a 6-round integral characteristic on MISTY1. Finally, we recover the secret key of the full MISTY1 with $2^{63.58}$ chosen plaintexts and 2^{121} time complexity. Moreover, if we use $2^{63.994}$ chosen plaintexts, the time complexity for our attack is reduced to $2^{108.3}$.

We also apply the division property to a lightweight block cipher LILLIPUT, which is an instantiation of extended generalized Feistel network (EGFN) developed by Berger et al. Division property can find a 13-round integral characteristic which is improved from the previous one by 4 rounds. The new integral characteristic is further extended to a 17-round key recovery attack which are improved from the previous best attack by 3 rounds.

The division property is very powerful tool to find integral characteristics against block ciphers based on S-boxes. However, it has not been applied to non-S-box-based ciphers like the SIMON family effectively. To gain high benefit against non-S-box-based ciphers, we extend the division property to a *bit-based division property*. As a result, we show the existence of 15-round integral characteristic of SIMON32.

We also apply the bit-based division property to a lightweight block cipher PRESENT. We introduce the compact representation for the bit-based division property, and it helps us to evaluate the propagation of the bit-based division property in practical time. As a result, we find 9-round integral characteristics, which is improved by two rounds than previous best one. Moreover, we attack 12-round PRESENT-80 and 13-round PRESENT-128 by using this new characteristic.

Contents

1 Introduction **13**
1.1 Motivation . . . 13
1.2 Our Contribution and Outline . . . 15

2 Symmetric-Key Cryptosystems and Integral Cryptanalysis **18**
2.1 Block Ciphers . . . 18
2.1.1 Definition . . . 18
2.1.2 Security Requirement . . . 18
2.1.3 Goal of Attacks . . . 19
2.1.4 Attack Assumption . . . 19
2.1.5 Structure of Block Ciphers . . . 20
2.2 Integral Cryptanalysis . . . 20
2.2.1 Higher-Order Differential Cryptanalysis . . . 20
2.2.2 Integral Cryptanalysis . . . 21
2.2.3 What is different between Integral and Higher-Order Differential Cryptanalyses? . . . 21
2.2.4 How to Find Integral Characteristics . . . 22

I Improved Techniques for Integral Cryptanalysis **24**

3 Generic Attack using Integral-Like Technique **25**
3.1 Introduction . . . 25
3.1.1 Already Evaluated Constructions . . . 25
3.1.2 Constructions that We Evaluate . . . 26
3.1.3 Our Contributions . . . 26
3.2 Preliminary . . . 27
3.2.1 Notations . . . 27
3.2.2 Feistel Networks . . . 28
3.2.3 Knudsen's attack . . . 28
3.3 New Properties for the K_iF- and K_iFP_i-Feistel . . . 29
3.4 New Key Recovery Attack on K_iF-Feistel . . . 31
3.4.1 New Key Recovery Attack for 5-round K_iF-Feistel . . . 31
3.4.2 New Key Recovery Attack for 6-round K_iF-Feistel . . . 32
3.5 New Key Recovery Attacks on K_iFP_i-Feistel . . . 33
3.5.1 A new property for 5-round K_iFP_i-Feistel . . . 33
3.5.2 New key recovery attack on 6-round K_iFP_i-Feistel . . . 35
3.6 Conclusion . . . 38

4 Fast Fourier Transform (FFT) Key Recovery Technique **39**
4.1 Introduction . . . 39
4.2 Previous Works . . . 40
4.2.1 Integral Cryptanalysis . . . 40
4.2.2 Ferguson et al.'s Partial-Sum Technique . . . 41
4.2.3 Collard et al.'s FFT Key Recovery . . . 42
4.3 FFT Key Recovery for Even-Mansour Scheme . . . 43
4.3.1 Even-Mansour Scheme . . . 43

4.3.2 Attack Framework . . . 43
4.3.3 Integral Cryptanalysis against PRØST Even-Mansour . . . 44
4.4 FFT Key Recovery for Key-Alternating Cipher . . . 45
4.4.1 Key-Alternating Cipher . . . 45
4.4.2 Attack Framework . . . 46
4.4.3 Integral Cryptanalysis against AES . . . 46
4.4.4 Integral Cryptanalysis against PRESENT . . . 47
4.5 FFT Key Recovery for Feistel Structure . . . 48
4.5.1 Feistel Structure . . . 48
4.5.2 Attack Framework . . . 48
4.5.3 Integral Cryptanalysis against CLEFIA . . . 48
4.6 Discussion . . . 50
4.7 Conclusion . . . 50

II Division Property: Efficient Method to Estimate Upper Bound of Algebraic Degree 52

5 Division Property and Application to Structural Evaluation 53
5.1 Introduction . . . 53
5.2 Preliminaries . . . 55
5.2.1 Notation . . . 55
5.2.2 Boolean Function . . . 55
5.2.3 Integral Distinguisher . . . 55
5.3 Division Property . . . 57
5.3.1 Introduction of Division Property . . . 57
5.3.2 Definition of Division Property . . . 57
5.3.3 Vectorial Division Property . . . 60
5.3.4 Collective Division Property . . . 60
5.3.5 Propagation Rules for Simple Operation . . . 63
5.4 Improved Distinguishers on Feistel Network . . . 65
5.4.1 Feistel Network . . . 65
5.4.2 Propagation Characteristic for Feistel Network . . . 66
5.4.3 Path Search Algorithm for (ℓ, d)-Feistel . . . 66
5.5 Improved Distinguishers on Substitute-Permutation Network . . . 68
5.5.1 Substitute-Permutation Network . . . 68
5.5.2 Propagation Characteristic for SPN . . . 69
5.5.3 Path Search Algorithm for (ℓ, d, m)-SPN . . . 71
5.6 Improved Distinguishers on AES-Like Ciphers . . . 73
5.6.1 AES-Like Cipher . . . 73
5.6.2 Path Search Algorithm for (ℓ, d, m)-AES . . . 73
5.6.3 Results for $(4, 3, m)$-AES . . . 74
5.6.4 Results for $(4, 2, m)$-AES . . . 75
5.6.5 Nontrivial Division Property on AES . . . 76
5.7 Conclusion . . . 77

6 Application to MISTY1 –Integral Cryptanalysis on Full MISTY1 79
6.1 Introduction . . . 79
6.2 MISTY1 . . . 80
6.3 Integral Characteristic by Division Property . . . 82
6.3.1 Notations . . . 82
6.3.2 Integral Attack . . . 83
6.3.3 Division Property . . . 83
6.4 Division Property for Public Function . . . 84
6.4.1 Application to MISTY S-boxes . . . 84
6.5 New Integral Characteristic Based on Division Property . . . 85
6.5.1 Division Property for FI function . . . 85
6.5.2 Division Property for FO function . . . 89
6.5.3 Division Property for FL Layer . . . 89

6.5.4 New Path Search for Integral Characteristics on MISTY1 . . . 90
6.5.5 Revisiting Known Characteristic for MISTY1 . . . 91
6.5.6 Optimized Algorithm . . . 93
6.6 Key Recovery Using New Integral Characteristic . . . 93
6.6.1 Passage of First FL Layer . . . 93
6.6.2 Sub Key Recovery Using Partial-Sum Technique . . . 93
6.6.3 Trade-off between Time and Data Complexity . . . 97
6.6.4 Follow-Up Results and Open Problem . . . 97
6.7 Conclusion . . . 98

7 Application to Extended Generalized Feistel Network: Lilliput 99
7.1 Introduction . . . 99
7.2 Lilliput . . . 100
7.2.1 Extended Generalized Feistel Network (EGFN) . . . 100
7.2.2 Specification of Lilliput . . . 101
7.3 New Integral Characteristic Based on Division Property . . . 102
7.3.1 Recall Division Property . . . 102
7.3.2 Integral Distinguisher on Lilliput . . . 103
7.4 Key Recovery Using New Integral Characteristic . . . 105
7.4.1 Detail Procedure for Key Recovery . . . 105
7.5 Conclusion . . . 107

8 Bit-Based Division Property and Application to Simon Family 111
8.1 Introduction . . . 111
8.2 Preliminaries . . . 113
8.2.1 Notations . . . 113
8.2.2 Integral Attack . . . 113
8.2.3 Division Property . . . 113
8.2.4 Simon Family . . . 114
8.2.5 Known Integral Characteristic on Simon Family . . . 114
8.3 Conventional Bit-Based Division Property . . . 115
8.3.1 Comparison between Conventional Bit-Based Division Property and Solving Algebraic Equations . . . 115
8.3.2 Propagation for Core Operation of Simon . . . 116
8.3.3 Application to Simon32 . . . 118
8.4 Bit-Based Division Property using Three Subsets . . . 119
8.4.1 Motivation . . . 119
8.4.2 Characteristic that Conventional Bit-Based Division Property cannot Find 119
8.4.3 Definition of Bit-Based Division Property using Three Subsets . . . 120
8.4.4 Propagation Rules . . . 121
8.4.5 Dependencies between $\mathbb{K}$ and $\mathbb{L}$ . . . 123
8.4.6 Propagation for Core Operation of Simon . . . 124
8.4.7 Application to Simon32 . . . 127
8.4.8 Application to Simeck32 . . . 128
8.5 Provable Security against Integral Cryptanalysis . . . 129
8.5.1 Provable Security for Designers . . . 129
8.5.2 Application to Simon Family . . . 130
8.5.3 Characteristics that Bit-Based Division Property cannot Find . . . 131
8.6 Conclusion . . . 131

9 Compact Representation for Bit-Based Division Property and Application to PRESENT 132
9.1 Introduction . . . 132
9.2 Preliminaries . . . 133
9.2.1 Notations . . . 133
9.2.2 Integral Attack . . . 133
9.2.3 Division Property . . . 133
9.2.4 Bit-Based Division Property . . . 134
9.3 Compact Representation for Division Property . . . 135

9.3.1 Motivation . . . 135
9.3.2 General Idea . . . 136
9.3.3 Toy Cipher using PRESENT S-box . . . 136
9.3.4 Core Function of PRESENT . . . 137
9.4 Improved Integral Attack on PRESENT . . . 138
9.4.1 New Algorithm to Find Integral Characteristics . . . 138
9.4.2 Key Recovery with MTTS Technique and FFT Key Recovery . . . 140
9.5 Conclusion . . . 144

10 Follow-Up Works and Conclusion 145
10.1 Technical Improvement . . . 145
10.1.1 Parity Set. . . . 145
10.1.2 MILP-Based Propagation Search. . . . 145
10.2 Summary of Works about Division Property . . . 146

A Division Property of Texts Encrypted by Six Rounds with Two FL Layers 148

B Propagation Characteristic Table for FI function 149

List of Publications 161

List of Figures

2.1 Block Cipher of n-bit block length and κ-bit secret key 18
2.2 Structure of a block cipher 20
2.3 Outline of the integral cryptanalysis 21
2.4 Integral distinguisher on 4-round AES 23

3.1 Luby-Rackoff construction and Knudsen's construction 26
3.2 Constructions that we evaluate in this chapter 26
3.3 Knudsen's key recovery attacks 29
3.4 New properties for the 5-round Feistel network 30
3.5 Key recovery attack on the 6-round K_iF-Feistel 32
3.6 Key recovery attack on the 5-round K_iFP_i-Feistel 34
3.7 The relationship between plaintext and ciphertext of the 6-round K_iFP_i-Feistel 35
3.8 Experimental results of the attack on the 6-round K_iFP_i-Feistel 38

4.1 Integral cryptanalysis against 6-round AES 41
4.2 FFT key recovery for Even-Mansour scheme 43
4.3 6-round integral characteristic of Prøst P_{128} 44
4.4 FFT key recovery for Key-alternating cipher 46
4.5 Round function of PRESENT 47
4.6 Round function of Feistel structure 48
4.7 Round function of CLEFIA 49
4.8 Key recovery of 12-round CLEFIA 49

5.1 Integral distinguisher on 4-round AES 56
5.2 Propagation characteristic of division property 60
5.3 Division property, vectorial division property, and collective division property 61
5.4 Division Property $\mathcal{D}^{8^2}_{\{[1,5],[3,3],[5,1],[6,0]\}}$ 62
5.5 (ℓ, d)-Feistel 66
5.6 Propagation characteristic for Feistel network 66
5.7 (ℓ, d, m)-SPN 70
5.8 Comparison between new and previous integral characteristics for $(4, 3, m)$-AES 76
5.9 Comparison between new and previous integral characteristics for $(4, 2, m)$-AES-like ciphers 77

6.1 Specification of MISTY1 81
6.2 The difference between Chap. 5 and this chapter. The left figure is an assumption used in Chap. 5. The right one is a new assumption used in this chapter. 84
6.3 Structure of FI function 85
6.4 New 6-round integral characteristic 94
6.5 $KL_{1,2} = 1$ 94
6.6 $KL_{1,1} = 0, KL_{1,2} = 0$ 94
6.7 $KL_{1,1} = 1, KL_{1,2} = 0$ 94
6.8 Key recovery step 94

7.1 Comparison of GFN (Left) and EGFN (Right) with four branches 100
7.2 Round function of Lilliput 101
7.3 Evaluation of $f_{13}(c, \mathcal{K}_{13})$. 108
7.4 Evaluation of $f_{14}(c, \mathcal{K}_{14})$. 108

7.5 Evaluation of $f_{15}(c, \mathcal{K}_{15})$. 109
7.6 Evaluation of $f_{16}(c, \mathcal{K}_{16})$. 109
7.7 Evaluation of Linear Part. 110

8.1 Round function of Simon$2n$ 114
8.2 Core operation of the Simon family. 116
8.3 Round function of the Simon-like ciphers 117
8.4 Core operation of the Simon family 124

9.1 Key-alternating cipher underlying PRESENT S-box 137
9.2 Round function of PRESENT 138
9.3 Equivalent circuit of round function of PRESENT 138
9.4 3-Round key recovery for PRESENT 141
9.5 Involved round keys of PRESENT-80 143
9.6 Involved round keys of PRESENT-128 143

10.1 Summary of works related to the division property 146

List of Tables

3.1 Summary of attacks on several Feistel networks whose round key size is $n/2$ bits 27
3.2 Summary of satisfied properties 30
3.3 Experimental results of the attack on the 6-round K_iF-Feistel 33

4.1 Summary of FFT key recovery, where κ, κ_1, and κ_2 be defined in Sect. 4.3, 4.4, and 4.5. 40
4.2 Comparison of attack results. Time column only includes the time complexity of the key recovery step, and it does not include the time complexity to count the frequency of the partial bit-string of ciphertexts corresponding to the chosen plaintexts (CPs). 40

5.1 The number of required chosen plaintexts to construct r-round integral distinguishers on the SIMONfamily, Serpent, and KECCAK-f 54
5.2 The number of chosen plaintexts to construct r-round integral distinguishers on $(32, 5)$- and $(64, 7)$-Feistel. Our distinguishers are got by implementing Algorithm 1. 68
5.3 The number of chosen plaintexts to construct r-round integral distinguishers on the SIMON family, where the F-function is not bijective. Our distinguishers are got by implementing Algorithm 1. 68
5.4 The number of required chosen plaintexts to construct r-round integral distinguishers on (ℓ, d)-Feistel. We get these values by implementing Algorithm 1. . . 69
5.5 The number of chosen plaintexts to construct r-round integral distinguishers on (ℓ, d, m)-SPN. Our distinguishers are got by implementing Algorithm 2. 71
5.6 The number of chosen plaintexts to construct r-round integral distinguishers on KECCAK-f and Serpent. Our distinguishers are got by implementing Algorithm 2. 71
5.7 The number of required chosen plaintexts to construct r-round integral distinguishers on (ℓ, d, m)-SPN. We get these values by implementing Algorithm 2. . . 72
5.8 The number of chosen plaintexts to construct r-round integral distinguishers on $(4, 3, m)$-AES 75
5.9 The number of chosen plaintexts to construct r-round integral distinguishers on $(4, 2, m)$-AES 76

6.1 Summary of single secret-key attacks against MISTY1 80
6.2 Division property of input is $\mathcal{D}^{7,2,7,7,2,7}_{\{[1,1,2,3,1,5]\}}$ 90
6.3 Propagation from $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\{[6,2,7,7,2,7,7,2,7,7,2,7]\}}$ 91
6.4 Propagation from $\mathcal{D}^{0,0,0,0,0,0,0,0,7,0,0,7}_{\{[6,2,7,7,2,7,7,2,7,7,2,7]\}}$ 92
6.5 Procedure of key recovery step 95
6.6 Trade-off between time and data complexity 97

7.1 Key recovery attacks in the single-key model against LILLIPUT. Related-key attack and chosen-key attacks reach 23 rounds, which are not included in this table. 100
7.2 Propagation from $\mathcal{D}^{4^{16}}_{\{[4,4,\ldots,4,3]\}}$ 104
7.3 Key-recovery procedure for f_{13} 105

8.1 Integral characteristics on SIMON32 112
8.2 Provable secure number of rounds for the SIMON family 112
8.3 Propagation of the conventional bit-based division property for the core operation in the SIMON family 116

8.4 Propagation of the bit-based division property on SIMON32 119
8.5 Propagation of the bit-based division property using three subsets for the core operation in the SIMON family 124
8.6 Propagation of the bit-based division property using three subsets on SIMON32 . 127
8.7 Accurate propagations up to six rounds 130
8.8 Lazy propagation of the bit-based division property for the SIMON family 130

9.1 Sets $V_S(u)$ for all $u \in \mathbb{F}_2^4$ for the PRESENT S-box. All four-bit values are represented in hexadecimal notation. The rightmost bit of the word corresponds to the least significant bit. 135
9.2 Propagation of the bit-based division property for PRESENT S-box. Vectors on $\mathbb{F}_2^4$ are represented an hexadecimal notation. 136
9.3 Compact representation for PRESENT S-box 136
9.4 Propagation from $\mathcal{D}^{1^{64}}_{\texttt{0x00000000000000FF}}$ 139
9.5 Propagation from $\mathcal{D}^{1^{64}}_{\texttt{0xFFFFFFFFFFFFFFF0}}$ 140
9.6 Propagation from $\mathcal{D}^{1^{64}}_{\texttt{0xFFFFFFFFFFFFFFFE}}$ 140

10.1 Summary of works related to the division property 146

B.1 Propagation from $\mathcal{D}^{7,2,7}_{\{[0,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$ 149
B.2 Propagation from $\mathcal{D}^{7,2,7}_{\{[1,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$ 149
B.3 Propagation from $\mathcal{D}^{7,2,7}_{\{[2,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$ 150
B.4 Propagation from $\mathcal{D}^{7,2,7}_{\{[3,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$ 150
B.5 Propagation from $\mathcal{D}^{7,2,7}_{\{[4,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$ 151
B.6 Propagation from $\mathcal{D}^{7,2,7}_{\{[5,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$ 151
B.7 Propagation from $\mathcal{D}^{7,2,7}_{\{[6,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$ 152
B.8 Propagation from $\mathcal{D}^{7,2,7}_{\{[7,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$ 152

Chapter 1

Introduction

1.1 Motivation

Cryptology. Information systems and computer networks become one of the most important social infrastructures, and we need to manage and utilize such sensitive information including personal information. Security technologies are necessary for the secure utilization, and a *cryptosystem* is especially important because it is the fundamental technology underlying every security technology. In fact, we always use some cryptosystems in our routine, e.g., social services via the Internet or online shopping. They use cryptosystem to construct secure channels. The *cryptology* is one of academical research topics, and the motivation is to support the security of various ongoing services, to design more secure and efficient cryptosystems, and to seek new application of cryptosystems for reduction of security risks.

The cryptology is developing research topic. The cryptology, as you imagine, is one of the most sensitive technologies. Actually, the cryptological technologies were managed by every nation for the national defense before World War II. We, i.e., the contemporary cryptographers, call them historical cryptosystems to distinguish with modern cryptosystems. The difference between the historical and modern cryptosystems is whether or not the algorithm of cryptosystems can open to the public. The security of the historical cryptosystem is guaranteed by concealing the algorithm from adversaries. Therefore, once the algorithm open to the public, we cannot use it in secure. We have to manage the cryptosystem itself strictly, and it is practically impossible to apply it to modern services via the Internet. On the other hand, the modern cryptosystem allows the publication of the algorithm, and we conceal only secret key, which is another input of cryptosystems. Since the algorithm open to the public, we can discuss the security in the academical community and the cryptology has been developed.

There are two types of cryptosystems; one is a public-key cryptosystem and another is symmetric-key cryptosystem. They have its advantages and drawbacks. Let us consider two persons: Alice and Bob, and Alice aims to send her message to Bob via the secure channel. In the public-key cryptosystem, Bob first generates his public key from his secret key, and the public key open to the public. It is practically infeasible to compute the secret key only from the public key. Then, Alice encrypts her message using Bob's public key. The ciphertext can be decrypted only using Bob's secret key. The public-key cryptosystem is very useful because we do not need to share their secret keys in advance. On the other hand, it has to use advanced mathematical computation to achieve such rich functionality, and such computation are not always efficient for the computer. In the symmetric-key cryptosystem, Alice and Bob have to share the common secret key in advance. However, it is very efficient and the processing is almost 100 times than that of the public-key cryptosystem. This doctor thesis focuses on the security of the symmetric-key cryptosystem.

Symmetric-Key Cryptosystem. There are several types in the symmetric-key cryptosystem, e.g., a block cipher, stream cipher, hash function, message authentication code, and authenticated encryption.

The most common symmetric-key cryptosystem is the block cipher. The input and output lengths of the block cipher are fixed, and n-bit block ciphers denote block ciphers with n-bit input and output. The claimed security is generally κ bits when block ciphers accept κ-bit secret keys k. Namely, let p and $E(\cdot,\cdot)$ be a plaintext and the block cipher, respectively, and the corresponding ciphertext c is computed as $c = E(k, p)$. The decryption uses the same key k and $p = E^{-1}(k, c)$, where $E^{-1}(\cdot,\cdot)$ is the decryption circuit. Since block ciphers support only fixed-length message, we have to use modes of operations for block ciphers like ECB, CBC, and CTR to support variable-length message.

The stream cipher is constructed by the pseudo-random number generator $F(\cdot,\cdot)$, which accepts κ-bit secret key k and n-bit initialization vector iv. The pseudo-random number generator generates finite-length key streams $z = F(k, iv) \in \{0,1\}^*$, and the ciphertext c is computed as $c = p \oplus z$. Then, the decryption is trivial, i.e., $p = c \oplus z$.

The role of the hash function is completely different from the block cipher and stream cipher. While the block cipher and stream cipher aim to achieve the confidentiality, the hash function aim to achieve the integrity. Let m be the original message, and the hash value is computed as $h = F(m)$, where $F(\cdot)$ denotes the hash function. As you see, the hash function does not accept the secret key. If the original message is forged to m', the hash value is different from the hash value of the original message, i.e., $F(m) \neq F(m')$. Here, the hash function is expected to achieve the following properties: pre-image resistance, second pre-image resistance, and collision resistance.

The message authentication code (MAC) is similar to the hash function, but the required security is different from the hash function. Let m be the message, and the MAC value is computed as $mac = F(k, m)$, where $F(\cdot,\cdot)$ denotes the message authentication code. Let us revisit the communication between Alice and Bob. The motivation of MAC is Bob knows whether or not the message is sent by Alice. Only using hash function cannot achieve this functionality because the adversary can easily compute the hash value for forged message. Therefore, to use MAC, we have to share the secret key in advance like block cipher and stream cipher.

The block cipher and stream cipher provide only confidentiality, and the message authentication code provides only integrity. Both confidentiality and integrity are important in real communication, and we use both symmetric-key cryptosystems under several combinations, e.g., enc-then-mac, mac-then-enc, and enc-and-mac. However, it is not always efficient. The authenticated encryption can provide both confidentiality and integrity in the same time efficiently.

Cryptanalysis. *Cryptanalyses* are one of the most important topics in cryptology. Only after many cryptographer analyze the security of cryptosystems and show supporting evidence that the cryptosystem is secure, it is admitted widely using and standardized. Moreover, the knowledge of cryptanalyses contributes to design more secure new cryptosystem. In fact, cryptology has been developed by iterating designs and analyses. If there are no third-party cryptanalysis and the security is guaranteed only by the designer intuition, it is too dangerous to use such cryptosystems. For example, if one nation designs cryptosystem with trapdoor and they promote its use, it will lead to mass-surveillance society. Therefore, we have to continue the activity of cryptanalyses to avoid such society.

Many cryptanalytic techniques have been proposed in last three decades. The most famous cryptanalytic technique is the *differential cryptanalysis* by Biham and Shamir at 1990 [BS90] and the *linear cryptanalysis* by Matsui at 1993 [Mat93]. Both cryptanalyses were developed to analyze Data Encryption Standard (DES) [Nat77]. After their proposals, many new symmetric-key cryptosystems which are secure against both cryptanalyses have been proposed. Similarly, many new advanced cryptanalytic techniques also have been developed, e.g., the *higher-order differential cryptanalysis* [Lai94], *impossible differential cryptanalysis* [BBS99], *truncated differential cryptanalysis* [Knu94], *integral cryptanalysis* [KW02], *multi-dimensional linear cryptanalysis* [BJV04], *meet-in-the-middle attack* [DH77, DS08] and *zero-correlation linear cryptanalysis* [BR11, BR14].

Integral Cryptanalysis. The integral cryptanalysis is one of the most powerful cryptanalytic techniques, and the concept was introduced by Knudsen and Wagner at FSE2002 [KW02]. However, similar techniques had been applied before the proposal of the integral cryptanalysis. The first this-type cryptanalysis is the higher-order differential cryptanalysis by Lai [Lai94] and is the extension of differential cryptanalysis. The advantage from the classical differential cryptanalysis was discussed by Knudsen in [Knu94]. Then, the similar technique to the higher-order differential cryptanalysis was used as the dedicated attack against the block cipher Square [DKR97]. This attack is widely applied to various block ciphers, where it was referred to the square attack [Dem02, HQ01, YPK02]. Then, some extensions of the square attack were proposed like the multiset [BS01], saturation [Luc01], and internal collision cryptanalyses [GM00]. In 2002, Knudsen and Wagner then formalized the square attack as the integral cryptanalysis.

The integral cryptanalysis first constructs an integral characteristic and then recovers the secret key by using the characteristic. In the integral characteristic, attackers first prepare N chosen plaintexts. If the XOR of all corresponding texts encrypted by r rounds is 0 for all keys, we say that the cipher has an r-round integral characteristic with N chosen plaintexts. Namely, we prepare the plaintext set $\mathbb{X}$ satisfying $\sum_{p \in \mathbb{X}} T(E_k(p)) = 0$ for all round key k, where the function T is a simple truncation function and E_k denotes the target block cipher whose number of rounds is reduced to r rounds. The probability that above equation holds for ideal ciphers is negligible. Therefore, we can execute a distinguishing attack by construction the integral characteristic. The integral cryptanalysis additionally append a key recovery step to the r-round integral characteristic. Attackers guess round keys used in the last s rounds and attack $(r + s)$ rounds. If guessed round keys are correct, the sum is always 0. Therefore, if the sum is not 0, the guessed round key is incorrect. By repeating this procedure, attackers recover the correct round keys used in the last s rounds.

1.2 Our Contribution and Outline

In this doctor thesis, we propose many novel techniques for the integral cryptanalysis. We first show the definition, security requirement, attack assumption, and structure of block ciphers In Chapter 2. Moreover, we show the history of the integral cryptanalysis as preliminaries. Our contribution mainly consists of two parts; one part is supporting advanced techniques for the integral cryptanalysis, and another part is the development of the *division property*.

Improved Technique for Integral Cryptanalysis.

- In Chapter 3, we deal with upper bounds for the security of some Feistel networks. Such a topic has been discussed since the introduction of Luby-Rackoff construction [LR88]. The Luby-Rackoff construction is unrealistic because its round functions must be chosen at random from the set of all functions. Knudsen dealt with a more practical construction whose round functions are chosen at random from a family of 2^{κ} randomly chosen functions, and showed an upper bound for the security by demonstrating generic key recovery attacks [Knu02]. However it is still difficult for designers to choose functions randomly. Then, this chapter considers the security of some Feistel networks which have more efficient and practical round functions, and such Feistel networks are indeed used by some Feistel ciphers in practice. We show new properties using the relationship between the pair of plaintext and ciphertext sets. We propose new generic key recovery attacks by using our properties, and confirm the feasibility by implementing the attack on Feistel ciphers with small block sizes. As a result, we conclude that efficient and practical 6-round Feistel networks are not secure.

- In Chapter 4, we propose a new technique for the integral cryptanalysis called the Fast Fourier Transform (FFT) key recovery. When N chosen plaintexts are required for the integral characteristic and the guessed key is κ bits, a straightforward key recovery requires the time complexity of $O(N2^{\kappa})$. However, the FFT key recovery only requires the time complexity of $O(N + \kappa 2^{\kappa})$. As a previous result using FFT, Collard et al. proposed that FFT can reduce the time complexity of a linear cryptanalysis [CSQ07]. We show that FFT can also reduce the complexity of the integral cryptanalysis. Moreover, the estimation of the complexity is very simple. We first show the complexity of the FFT key recovery against three structures, an Even-Mansour scheme, key-alternating cipher, and Feistel structure. As examples of these structures, we show integral cryptanalyses against PRØST, AES [U.S01], PRESENT [BKL$^+$07], and CLEFIA [SSA$^+$07]. As a result, an 8-round PRØST $\tilde{P}_{128,K}$ can be attacked with an approximate time complexity of $2^{79.6}$. For the key-alternating cipher, a 6-round AES and a 10-round PRESENT can be attacked with approximate time complexities of $2^{51.7}$ and $2^{97.4}$, respectively. For the Feistel structure, a 12-round CLEFIA can be attacked with approximate time complexities of $2^{87.5}$.

Division Property: Efficient Method to Estimate Upper Bound of Algebraic Degree.

- Chapter 5 proposes the *division property* as a novel technique to find integral characteristics. The most important part of the integral cryptanalysis is how to find good integral characteristics. We already have two conventional methods to find them; one method uses the propagation of integral property and another method estimates the upper bound of the algebraic degree of symmetric-key cryptosystems. The former was proposed by Knudsen and Wagner [KW02]. Nowadays, this method has been used as the most common technique to find integral characteristics. The latter was proposed by Lai [Lai94], and this attack is often called the higher-order differential cryptanalysis. Nevertheless both methods find the same non-trivial behavior, i.e., the integral characteristic, they construct the integral characteristic by exploiting different property of symmetric-key cryptosystems. The division property is developed by combining the advantages of the integral property and degree estimation. The division property can find more accurate integral characteristics than conventional two methods. To show its advantage, we search for integral characteristics on Feistel, SPN, and AES-like structures [U.S01] using the division property. As a result, we can find improved integral characteristics for all structures.

- Chapter 6 shows the integral cryptanalysis on full MISTY1 [Mat97], and we show that the division property is also useful for the dedicated attack against specific cryptosystems. MISTY1 is a block cipher designed by Matsui in 1997, and it was well evaluated and standardized by projects, such as CRYPTREC [CRY13], ISO/IEC [ISO05], and NESSIE [NES04]. We propose a key recovery attack on the full MISTY1, i.e., we show that 8-round MISTY1 with 5 FL layers does not have 128-bit security. Many attacks against MISTY1 have been proposed, but there is no attack against the full MISTY1. Therefore, our attack is the first cryptanalysis against the full MISTY1. We construct a new integral characteristic by using the propagation of the division property. As a result, we construct a 6-round integral characteristic on MISTY1. Finally, we recover the secret key of the full MISTY1 with $2^{63.58}$ chosen plaintexts and 2^{121} time complexity. Moreover, if we use $2^{63.994}$ chosen plaintexts, the time complexity for our attack is reduced to $2^{108.3}$.

- Chapter 7 shows the best third-party cryptanalysis against a lightweight block cipher Lilliput [BFMT15], which is an instantiation of an *extended generalized Feistel network (EGFN)* developed by Berger et al. [BMT13]. Its round function updates a part of the state only linearly, which yields several security concerns. Owing to its unique computation structure, the designers expected that EGFN efficiently enhances security against the integral cryptanalysis from generalized Feistel network. However, the security is not enhanced as the designers expect. In fact, division property can find a 13-round integral characteristic which is improved from the previous characteristic by 4 rounds. The new integral characteristic is further extended to a 17-round key recovery attack which are improved from the previous best attack by 3 rounds.

- Chapter 8 extends the division property to bit-based variant. The division property is very powerful tool to find integral characteristics against symmetric-key cryptosystem based on S-boxes. However, it has not been applied to non-S-box-based ciphers like the Simon family [BSS+13] effectively, and only the existence of the 10-round integral characteristic on Simon32 was proven in Chapter 5. On the other hand, the experimental characteristic, which possibly does not work for all keys, covers 15 rounds [WLV+14], and there is a 5-round gap. To fill the gap, we introduce a *bit-based division property*, and we apply it to show that the experimental 15-round integral characteristic always works for all keys. Though the bit-based division property finds more accurate integral characteristics, it requires much time and memory complexity. As a result, we cannot apply it to symmetric-key cryptosystem whose block length is over 32. Therefore, we alternatively propose a method for designers. The method works for ciphers with large block length, and it shows "provable security" against integral cryptanalyses using the division property. We apply this technique to the Simon family and show that Simon48, 64, 96, and 128 probably do not have 17-, 20-, 25-, and 29-round integral characteristics, respectively.

- Chapter 9 apply the bit-based division property to PRESENT [BKL+07]. We first show the application of the bit-based division property to an S-box. The similar observation, which is called the parity set, was introduced by Boura and Canteaut [BC16], and they show the relationship between the parity set and the division property. We first show the relationship between the parity set and the bit-based division property, and the parity set helps us to get the propagation characteristic of the bit-based division property for an S-box. We next propose the compact representation for the bit-based division property. The disadvantage of the bit-based division property is that it cannot be applied to block ciphers whose block length is over 32 because of high time and memory complexity. The compact representation partially solves this problem, and we apply this technique to 64-bit block cipher PRESENT to illustrate our method. We can accurately evaluate the propagation of the bit-based division property thanks to the compact representation. As a result, we find 9-round integral characteristics, and the characteristic is improved by two rounds than previous best characteristic. Moreover, we attack 12-round PRESENT-80 and 13-round PRESENT-128 by using these new characteristic.

Chapter 2

Symmetric-Key Cryptosystems and Integral Cryptanalysis

2.1 Block Ciphers

The security of block ciphers is discussed in this doctor thesis, and we first show the definition and security requirement. Then we show attack assumptions which is well used to analyze block ciphers. We finally show how block ciphers are designed. The understanding of the structure of block ciphers is necessary to understand the concept of the integral cryptanalysis.

2.1.1 Definition

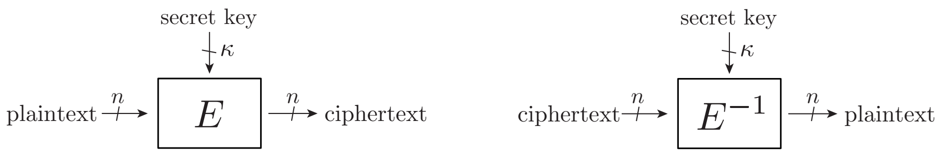


Figure 2.1: Block Cipher of n-bit block length and κ-bit secret key

The input and output lengths of a block cipher are fixed, and we call a block cipher with n-bit input and output an n-bit block cipher. The block cipher accepts a κ-bit secret key. Therefore, the block cipher is keyed permutation such that a mapping $E : \mathbb{F}_2^\kappa \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. This mapping is bijective under the condition the secret key is fixed, and the decryption is also defined as a mapping $E^{-1} : \mathbb{F}_2^\kappa \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. Figure 2.1 shows an n-bit block cipher with κ-bit secret key.

2.1.2 Security Requirement

The ideal block cipher should be modeled as (strong) pseudo-random permutation. The block cipher is an underlying primitive, and some cryptographic functionalities, e.g, modes of operation, hash functions, and authenticated encryptions, are realized by using the block cipher. Then, we can discuss the reduction security of the cryptographic function, i.e., we can prove that the application is always secure as long as the underlying block ciphers behaves like (strong) pseudo-random permutation.

Definition 2.1 (Pseudo-random permutation). *Let $E : \mathbb{F}_2^\kappa \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be an efficient keyed permutation. We say that E is a pseudo-random permutation if*

- *For any $k \in \mathbb{F}_2^\kappa$, E is a bijective from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$.*
- *For any $k \in \mathbb{F}_2^\kappa$, there is an efficient algorithm to evaluate E.*
- *For any polynomial-time algorithm $\mathcal{A}$, we have*

$$\left| \Pr[A^{E(k,\cdot)}] - \Pr[A^{f(\cdot)}] \right| < negl(n),$$

where $k \in \mathbb{F}_2^\kappa$ is chosen uniformly at random and f is chosen uniformly at random from the set of permutation on n-bit strings.

Definition 2.2 (Strong pseudo-random permutation)**.** *Let $E : \mathbb{F}_2^\kappa \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ be an efficient keyed permutation. We say that E is a pseudo-random permutation if*

- *For any $k \in \mathbb{F}_2^\kappa$, E is a bijective from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$.*
- *For any $k \in \mathbb{F}_2^\kappa$, there is an efficient algorithm to evaluate E and E^{-1}.*
- *For any polynomial-time algorithm A, we have*

$$\left|\Pr[A^{E(k,\cdot),E^{-1}(k,\cdot)}] - \Pr[A^{f(\cdot),f^{-1}(\cdot)}]\right| < negl(n),$$

where $k \in \mathbb{F}_2^\kappa$ is chosen uniformly at random and f is chosen uniformly at random from the set of permutation on n-bit strings.

From the definition of the block cipher, the permutation is chosen from at most 2^κ permutations. On the other hand, when f is chosen uniformly at random from the set of permutation on n-bit strings, the permutation is chosen from $(2^n)!$ permutations.

2.1.3 Goal of Attacks

The motivation of cryptanalyses is to reveal whether or not target block ciphers fulfill the security requirements. Generally, goals of attacks on block ciphers are summarized as follows.

Distinguishing Attacks The goal of attackers is to distinguish the target block cipher from an ideal cipher, i.e. strong pseudo-random permutation. The distinguishing attack does not recover the secret information, but it clearly violates security requirements of the block cipher as a (strong) pseudo-random permutation.

Key Recovery Attacks The goal of attackers is to recover the secret key. Once the secret key is recovered, any ciphertext is easily decrypted by attackers.

Message Recovery Attacks The goal of attackers is to recover plaintexts. Therefore, message recovery attack can be also possible if the key recovery attack is possible. If attackers can recover plaintext without recovering the secret key, such attacks are the message recovery attack not key recovery attack.

The secret key is chosen from a finite set. Therefore, if attackers try out all possible secret keys, they can recover the correct key. This is called a *brute force attack*. The brute force attack is not regarded as a flaw of the design because it is theoretically impossible to avoid. Therefore, the block cipher is regarded as secure if there is no an efficient attack than the brute force attack, and cryptographers have following consensus about successful attack.

Definition 2.3 (Successful attack)**.** *If the time complexity of an attack is less than that of the brute force attack, such an attack is regarded as a successful attack.*

If there is an attack to recover the secret key or distinguish the block cipher from a random permutation, such attack is called a shortcut attack. For block ciphers, allowing shortcut attacks is regarded as a significant vulnerability. Therefore, the goal of cryptanalyses is finding a shortcut attack.

2.1.4 Attack Assumption

Several attack assumptions are used to analyze block ciphers. We summarize four common attack assumptions as follows.

Ciphertext-only attacks An attacker is allowed to obtain only ciphertexts.

Known-plaintext attacks An attacker is allowed to know pairs of plaintext and the corresponding ciphertext.

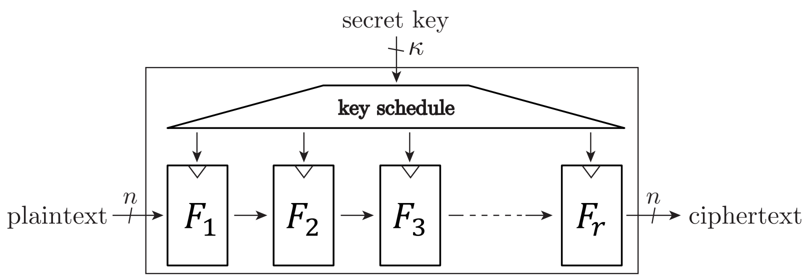


Figure 2.2: Structure of a block cipher

Chosen-plaintext attacks An attacker is allowed to choose plaintexts to be encrypted and know the corresponding ciphertexts.

Chosen-ciphertext attacks An attacker is allowed to choose ciphertexts to be decrypted and know the corresponding plaintexts.

From above definitions, a pseudo-random permutation is secure against chosen plaintext attacks, and a strong pseudo-random permutation is secure even if chosen ciphertext attacks. We cannot execute ciphertext-only attack against modern block ciphers without some information of plaintexts. Therefore, many well-known cryptanalytic techniques use known and chosen plaintext (ciphertext) attacks, e.g., the differential and integral cryptanalyses are a chosen plaintext attack and the linear cryptanalysis is a known plaintext attack.

2.1.5 Structure of Block Ciphers

Block ciphers are generally designed using iterating structure. First, a round function, which is high efficient but weak (keyed) function, are designed, and a block cipher is constructed by iterating the round function several times with different round keys. Here, round keys are generated from the secret key using a key schedule. When the round function is iterated R times, we call it an R-round block cipher. Figure 2.2 shows the structure of the block cipher.

2.2 Integral Cryptanalysis

An integral cryptanalysis is one of the most powerful cryptanalytic techniques, and the concept was introduced by Knudsen and Wagner at FSE2002 [KW02]. However, similar techniques have been applied before the proposal of the integral cryptanalysis. In this section, we summarize the integral cryptanalysis and related works.

2.2.1 Higher-Order Differential Cryptanalysis

The concept of the higher-order differential cryptanalysis was first introduced by Lai [Lai94] and the advantage over the traditional differential cryptanalysis was studied by Knudsen [Knu94]. Let $E(k,\cdot) = E_k(\cdot)$ be an n-bit block cipher, and the traditional differential cryptanalysis focuses on

$$\Delta_\alpha E_k(x) = E_k(x+\alpha) - E_k(x)$$

and recovers the secret key by analyzing the relationship between α and $\Delta_\alpha E_k(x)$. On the other hand, the higher-order differential cryptanalysis focuses on ith order differential as

$$\Delta^{(i)}_{\alpha_1,\alpha_2,\ldots,\alpha_i} E_k(x) = \Delta_{\alpha_i}(\Delta^{(i-1)}_{\alpha_1,\alpha_2,\ldots,\alpha_{i-1}} E_k(x))$$

and recovers the secret key by analyzing the relationship between $(\alpha_1,\alpha_2,\ldots,\alpha_i)$ and $\Delta^{(i)}_{\alpha_1,\alpha_2,\ldots,\alpha_i} E_k(x)$. For example, the 2nd order differential denotes as

$$\begin{aligned}\Delta^{(2)}_{\alpha_1,\alpha_2} E_k(x) &= \Delta_{\alpha_2}(E_k(x+\alpha_1) - E_k(x)) \\ &= E_k(x+\alpha_1+\alpha_2) - E_k(x+\alpha_2) - E_k(x+\alpha_1) + E_k(x).\end{aligned}$$

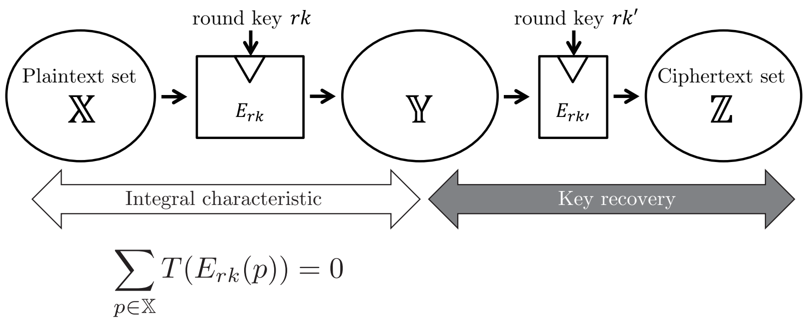

round key rk
round key rk'
Plaintext set $\mathbb{X}$
E_{rk}
$\mathbb{Y}$
$E_{rk'}$
Ciphertext set $\mathbb{Z}$
Integral characteristic
Key recovery
$\sum_{p \in \mathbb{X}} T(E_{rk}(p)) = 0$

Figure 2.3: Outline of the integral cryptanalysis

Let $\deg(E_k)$ be the algebraic degree of E_k, and Lai showed the following relationship.

$$\deg(\Delta_\alpha E_k) \geq \deg(E_k) - 1.$$

Therefore, assuming the algebraic degree of E_k is at most d, the dth and $(d+1)$th order differentials are constant and zero, respectively. It is obvious that the algebraic degree has to be at least $\min(n-1, \kappa-1)$ if block ciphers accept κ-bit secret keys. Then, Knudsen showed the advantage over the conventional differential cryptanalysis by using the toy ciphers [Knu94].

2.2.2 Integral Cryptanalysis

The similar technique to the higher-order differential cryptanalysis was used as the dedicated attack against the block cipher SQUARE [DKR97]. Since this dedicated attack is powerful cryptanalysis, it is widely applied to various block ciphers, where it was referred to the square attack [Dem02, HQ01, YPK02]. Then, some extensions of the square attack were proposed like the multiset [BS01], saturation [Luc01], and internal collision cryptanalyses [GM00]. In 2002, Knudsen and Wagner formalized the square attack as the integral cryptanalysis. Figure 2.3 shows the outline of the integral cryptanalysis. Attackers first prepare N chosen plaintexts. If the XOR of all corresponding ciphertexts is 0 for all keys, we say that the cipher has an integral characteristic with N chosen plaintexts. Namely, we prepare the plaintext set $\mathbb{X}$ satisfying

$$\sum_{p \in \mathbb{X}} T(E_{rk}(p)) = 0 \tag{2.1}$$

for all round key rk, where the function T is an observing function and often used the simple truncation from n bits to $m\,(\leq n)$ bits. The probability that Eq. (2.1) holds for ideal ciphers is 2^{-m}. Therefore, we can execute a distinguishing attack by constructing the integral characteristic.

The integral cryptanalysis additionally appends a key recovery step to the integral characteristic. Assuming that block ciphers has an r-round integral characteristics, attackers guess round keys used in the last s rounds and attack $(r+s)$ rounds. If a guessed round key is correct, Eq. (2.1) always holds. Therefore, if Eq. (2.1) does not hold, the guessed round key is incorrect. By repeating this procedure, attackers recover the correct round keys used in the last s rounds.

2.2.3 What is different between Integral and Higher-Order Differential Cryptanalyses?

The definition of the higher-order differential cryptanalysis is different from that of the integral cryptanalysis. However, they are often regarded as the same cryptanalysis. We discuss why they are regarded as the same cryptanalysis in this section.

To achieve high performance under computer, almost all operations of block ciphers are defined using bit operations like XOR and bit-oriented AND. Then, both the addition of the integral cryptanalysis and the difference of the higher-order differential cryptanalysis become

XOR. For example, the 2nd order differential is represented as

$$\Delta^{(2)}_{\alpha_1,\alpha_2}E_k(x) = E_k(x \oplus \alpha_1 \oplus \alpha_2) \oplus E_k(x \oplus \alpha_1) \oplus E_k(x \oplus \alpha_2) \oplus E_k(x)$$
$$= \bigoplus_{p \in V[\alpha_1,\alpha_2]} E_k(p),$$

where $V[\alpha_1, \alpha_2]$ is linear subspace whose basis is (α_1, α_2). Therefore, the 2nd order differential is the same as the integral characteristic with $\mathbb{X} = V[\alpha_1, \alpha_2]$.

The higher-order differential cryptanalysis uses only a linear subspace as the input set, but the integral cryptanalysis can use arbitrary input set. However, since inputs and outputs of all components of block ciphers are represented as bit strings, almost all previous integral cryptanalyses have never used input sets that are not linear subspace. Therefore, there is no difference between the higher-order and integral characteristics once both cryptanalyses are applied to real ciphers.

2.2.4 How to Find Integral Characteristics

In the practical use, the distinction between the higher-order and integral cryptanalyses is not important. It is rather more important to understand how to find their characteristics. The higher-order differential cryptanalysis focuses on the algebraic degree to construct the characteristics, while the integral cryptanalysis evaluates the propagation of the integral property to construct the characteristic. Nowadays, an analysis that mainly exploits the algebraic degree is called the "higher-order differential cryptanalysis." On the other hand, an analysis that mainly exploits the integral property is called the "integral cryptanalysis."

Degree Estimation. The higher-order differential characteristic is constructed by evaluating the upper bound of the algebraic degree, but it is not easy in general. The most classical method uses the fact that the algebraic degree of an r-round block cipher is upper-bounded by d^r if the algebraic degree of each round function is at most d. However, this method is too rough evaluation, and the upper bound is generally more small. Canteaut and Videau showed tighter bound of the degree of iterated round functions [CV02], and Boura et al. then improved the bound in [BCC11]. This improved bound is useful to evaluate the upper bound on Substitution-Permutation network (SPN).

Theorem 2.1 ([BCC11]). *Let S be a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^n$ corresponding to the concatenation of m smaller S-boxes, defined over $\mathbb{F}_2^{n_0}$. Let δ_k be the maximal degree of the product of any k bits of anyone of these S-boxes. Then, for any function G from $\mathbb{F}_2^n$ into $\mathbb{F}_2$, we have*

$$\deg(G \circ S) \leq n - \frac{n - \deg(G)}{\gamma},$$

where

$$\gamma = \max_{1 \leq i \leq n_0 - 1} \frac{n_0 - i}{n_0 - \delta_i}.$$

For example, let us consider SPN ciphers using four 4-bit bijective S-boxes. Since the algebraic degree of 4-bit bijective S-boxes is at most 3, the algebraic degree of the 2-round cipher is at most $3^2 = 9$. When we use the classical method, the algebraic degree of the 3-round cipher is at most $\min(15, 3^3 = 27) = 15$ but it is not tight. On the other hand, Theorem 2.1 shows that the algebraic degree is at most $\lfloor 16 - \frac{16-9}{3} \rfloor = 13$. Boura et al. showed integral distinguishers on KECCAK [DBPA11] and *Luffa* [CSW08] by using this theorem. We cannot apply Theorem 2.1 to non-SPN ciphers, and real block ciphers have more complicated round function. Therefore, this method is still far from the tight lower bound.

A brute-force method, i.e., all algebraic equations are resolved, is often used because the theoretical estimation of the algebraic degree is difficult [THK99]. The accurate algebraic degree is evaluated by using this method, but it generally requires practically infeasible time complexity. Therefore, the application is very limited.

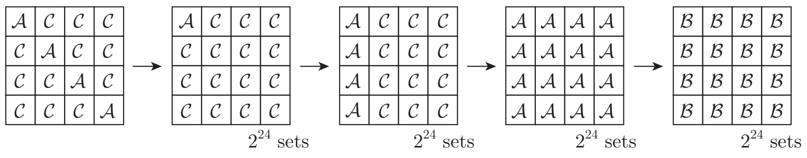


Figure 2.4: Integral distinguisher on 4-round AES

Integral property. The propagation of the integral property is most widely applied to construct integral characteristics, and many integral characteristics have been constructed by evaluating the propagation of the integral property [KW02, LWZ11, WZ11, YPK02, ZRHD08]. It uses four integral properties as follows:

- ALL ($\mathcal{A}$) : Every value appears the same number in the multiset.
- BALANCE ($\mathcal{B}$) : The XOR of all texts in the multiset is 0.
- CONSTANT ($\mathcal{C}$) : The value is fixed to a constant for all texts in the multiset.
- UNKNOWN ($\mathcal{U}$) : The multiset is indistinguishable from one of n-bit random values.

Knudsen and Wagner showed that AES has the 4-round integral distinguisher with 2^{32} chosen plaintexts [KW02] (see Fig. 2.4).

Unfortunately, the integral property does not find effective characteristics if block ciphers consist of non-bijective functions like DES [Nat77] and SIMON [BSS+13]. Moreover, since the propagation does not clearly exploit the algebraic degree of block ciphers, it tends not to construct effective characteristics on block ciphers with low-degree round functions.

Part I

Improved Techniques for Integral Cryptanalysis

Chapter 3

Generic Attack using Integral-Like Technique

Abstract– In this chapter, we deal with the security of some Feistel networks. Such a topic has been discussed since the introduction of a Luby-Rackoff construction, which constructs a pseudo-random permutation from pseudo-random functions. However, the construction is unrealistic because its round functions must be chosen at random from the set of all functions. Knudsen dealt with the security of a more practical construction whose round functions are chosen at random from a family of 2^{κ} randomly chosen functions and showed an upper bound for the security by demonstrating generic key recovery attacks. However it is still difficult for designers to choose functions randomly. Then, this chapter focuses on the security of some Feistel networks which have more efficient and practical round functions, and such Feistel networks are indeed used by some Feistel ciphers in practice. We show new properties using the relationship between plaintexts and ciphertexts. We propose new generic key recovery attacks by using our properties, and confirm the feasibility by implementing the attack on Feistel ciphers with small block sizes. As a result, we conclude that efficient and practical 6-round Feistel networks are not secure.
Keywords– Block ciphers, Feistel networks, Round functions, Key recovery attacks

3.1 Introduction

We can construct n-bit pseudo-random permutations from $(n/2)$-bit pseudo-random functions by using Feistel networks. Luby and Rackoff [LR88] proved that 3- and 4-round Feistel networks are sufficient to make a pseudo-random permutation and a super pseudo-random permutation, respectively, if round functions are pseudo-random functions. Since then, many results for pseudo-randomness on Feistel networks have been proposed [Pat92, Luc96, Pat03, Pat10, LP12]. Luby and Rackoff also showed that the 3-round Feistel network can be distinguished from random permutations by using the adaptive chosen plaintext and ciphertext attack. Since then, many generic attacks on Feistel networks have been proposed [Knu02, Pat04, KR07, SY11]. In this chapter, we discuss the number of rounds that Feistel networks can be attacked by a generic attack. For instance, Patarin showed the distinguishing attack on the 5-round Feistel network whose round functions are pseudo-random functions [Pat04]. The attack is the chosen plaintext attack (CPA) with $O(2^{3n/4})$ texts. Moreover, Knudsen showed the distinguishing attack on the 5-round Feistel network whose round functions are pseudo-random permutation [Knu02]. The attack is CPA with $O(2^{n/2})$ texts.

3.1.1 Already Evaluated Constructions

Luby-Rackoff construction has round functions which are chosen at random from a family of $2^{(n/2)2^{n/2}}$ functions (see the left of Fig. 3.1). This means that the key size for the r-round Feistel cipher is $(r \times (n/2)2^{n/2})$ bits, and it is unrealistic in practice. Then, Knudsen introduced more practical Feistel networks whose round functions are chosen at random from a family of 2^{κ} randomly chosen functions. We call this network the F_{K_i}-Feistel (see the right of Fig. 3.1). In the F_{K_i}-Feistel, attackers can search for the correct round function by an exhaustive search over

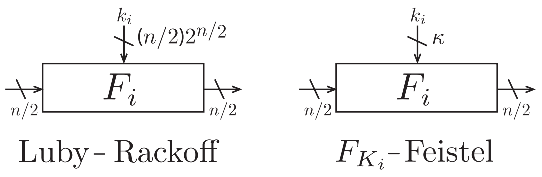


Figure 3.1: Luby-Rackoff construction and Knudsen's construction

all 2^κ possible round functions. Moreover, Knudsen proposed the key recovery attack on 5- and 6-round F_{K_i}-Feistel by using CPA. Time complexity of the attack on the 5-round F_{K_i}-Feistel is $2^{\kappa+\frac{n+6}{4}}$. Time complexity of the attack on the 6-round F_{K_i}-Feistel is $2^{\kappa+\frac{n}{2}+1}$, but this attack is as expensive as the brute force attack when the round key size is the same as the input size of round functions (i.e., $\kappa = n/2$ bits) and the master key size is the same as the block length (i.e., n bits).

3.1.2 Constructions that We Evaluate

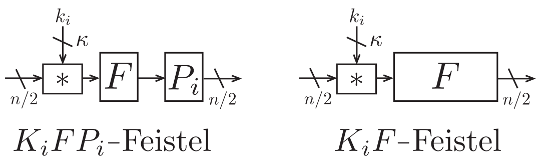


Figure 3.2: Constructions that we evaluate in this chapter

We revisit the Knudsen's result. In order to implement the F_{K_i}-Feistel, designers must design ideal compression functions which have $(\kappa + n/2)$-bit input and $(n/2)$-bit output. However, it is difficult to design such ideal compression functions. Therefore, many block ciphers use more simple construction, e.g., round functions in Feistel networks consist of a key insertion and public permutation rather than a compression function. We split the round function into the *key insert operation* $*$ and *nonlinear bijective function* F. The operation $*$ is a function to mix an input with a round key by a simple operation, e.g., an exclusive OR ($\oplus$) or a modular addition ($\boxplus$). The function F is a public permutation.

We have many methods to construct the function F, and we evaluate two constructions in this chapter. First, we introduce the K_iF-Feistel as the efficient construction (see the right of Fig. 3.2). In this construction, a common nonlinear bijective function F is used for all rounds. For instance, Camellia [AIK+00], SEED [Kor05] and GOST cipher [Nat89] have this construction. Next, we introduce the K_iFP_i-Feistel (see the left of Fig. 3.2). In this construction, a common permutation F and r linear functions P_i are used, and the composition function $(P_i \circ F)$ is used as the ith round function. This construction is called the diffusion switching mechanism [SS06], which is an effective method to design practical and secure Feistel ciphers.

3.1.3 Our Contributions

We summarize existing and our results in Table 3.1, where the target is defined by the key insert operation and nonlinear bijective function. For instance, $\oplus K_iF$ denotes the K_iF-Feistel with the XOR key insertion. From Table 3.1, we can conclude that 6-round K_iF- and K_iFP_i-Feistel with the XOR key insertion are not secure.

In this chapter, we first discuss the security of the 5-round K_iF-Feistel. We show that there exists the non-ideal relationship between plaintexts and corresponding ciphertexts if and only if corresponding internal states satisfy our property. Therefore we can distinguish whether texts have our property by observing the relationship between plaintexts and ciphertexts. We propose new key recovery attacks by using this distinguisher, and this attack is regarded as the meet-in-the-middle-attack using differences. The time complexity is $O(2^{n/2})$ and the attack model is the known plaintext attack (KPA) rather than CPA. We next discuss the security of

Table 3.1: Summary of attacks on several Feistel networks whose round key size is $n/2$ bits

Target	Rounds	Time	Number of texts		Reference
F_{K_i}	5	$2^{3n/4+3/2}$	$\sqrt{n}2^{n/4}$	CPA	[Knu02]
$\oplus K_i F$	5	$2^{n/2+3}$	$2^{n/2+2}$	KPA	Ours
$\oplus K_i F P_i$	5	$t2^{n/2}$	$t2^{n/2}$	KPA	Ours
$\boxplus K_i F$	5	$2^{n/2+3}$	$2^{n/2+2}$	KPA	Ours
F_{K_i}	6	2^{n+1}	$\frac{n}{2}2^{n/2}$	CPA	[Knu02]
$\oplus K_i F$	6	$\frac{n}{2}2^{n/2}$	$\frac{n}{2}2^{n/2}$	CPA	Ours
$\oplus K_i F P_i$	6	$2^{7n/8+1}$	$2^{7n/8}$	CPA	Ours

Let t in the result of $\oplus K_i F P_i$ be a function of n. See Sect. 3.5.1 in detail.

the 6-round K_iF-Feistel. However, we cannot distinguish by using plaintexts and corresponding ciphertexts whether texts have our property. Then we propose a new property which exploits that the key insert operation is XOR. By using the additional property, we can construct non-ideal relationship between plaintexts and ciphertexts. As a result, we can attack the 6-round K_iF-Feistel, and the time complexity is $O(\frac{n}{2}2^{n/2})$.

We discuss the security of the 5-round K_iFP_i-Feistel. We cannot use the property which is used to attack the 5-round K_iF-Feistel because different round functions are used in every round. However if the key insert operation is XOR, we can get non-ideal relationship between plaintexts and ciphertexts. By using this non-ideal relationship, we propose a new key recovery attack on the 5-round K_iFP_i-Feistel. The time complexity is $O(2^{n/2})$ and the attack model is KPA rather than CPA. We next discuss the security of the 6-round K_iFP_i-Feistel. We decrypt one round by exhaustive search over all $2^{n/2}$ round keys, and we use the property which is used to attack the 5-round K_iFP_i-Feistel. Unfortunately, we need about 2^n complexity to execute this attack. Therefore, we first carefully pick texts which are used in our attack. Next, we guess the round key against only picked texts. We can attack the 6-round K_iFP_i-Feistel by using this method, and the time complexity is $O(2^{7n/8})$. We confirm the feasibility by implementing the attack against Feistel ciphers with small block sizes.

This chapter is organized as follows. Section 3.2 gives notations, Feistel networks evaluated in this chapter, and the Knudsen's attack. Section 3.3 gives new properties for the K_iF-Feistel and the K_iFP_i-Feistel. Section 3.4 gives key recovery attacks for the K_iF-Feistel, and Sect. 3.5 gives key recovery attacks for the K_iFP_i-Feistel. We conclude this chapter in Sect. 3.6.

3.2 Preliminary

3.2.1 Notations

Let n be the block length for a Feistel network. Let p and c be plaintexts and ciphertexts, respectively. Let k_i be a round key of the ith round function. For a value x, x_L denotes the left half of x and x_R denotes the right half of x. In the r-round Feistel network, the ciphertext is calculated as follows:

$$c = E_k(p) = Swap \circ \Psi_{k_r} \circ \cdots \circ \Psi_{k_1}(p_L, p_R),$$

where $Swap(x_L, x_R) = (x_R, x_L)$ holds. Moreover, Ψ_{k_i} is defined as follows:

$$\begin{aligned}(L_{i+1}, R_{i+1}) = \Psi_{k_i}(L_i, R_i) &= (y_i \oplus R_i, L_i)\\ &= (\psi_{k_i}(L_i) \oplus R_i, L_i),\end{aligned}$$

where L_i and R_i denote the left half and the right half of the ith round input, respectively. Let ψ_{k_i} be the round function of the ith round. Moreover, let y_i be an output of the ith round function.

3.2.2 Feistel Networks

We define some Feistel networks discussed in this chapter. We first show the Luby-Rackoff construction. In the construction, ψ_{k_i} must be chosen at random from the set of all functions in every round. Unfortunately, it is unrealistic because the key size is enormous.

For more practical construction than the Luby-Rackoff construction, Knudsen defined the ideal round function as follows:

Definition 3.1 (Knudsen's ideal round function)**.** *Let F_k be an $(n/2)$-bit function or permutation chosen from a family of 2^κ function. Then F_k is called ideal if exhaustively search at least 2^κ possible functions is necessary to find the correct function.*

We call this construction F_{K_i}-Feistel. The size of secret information of the r-round F_{K_i}-Feistel is only $(r \times \kappa)$ bits. However, to achieve the F_{K_i}-Feistel, designers must design ideal compression functions which have $(\kappa + n/2)$-bit input and $(n/2)$-bit output, and it is not easy to design such functions. Therefore, many block ciphers use more simple construction, e.g., round functions in Feistel networks consist of key insertions and public permutations rather than compression functions.

We introduce a construction called K_iF-Feistel. In the K_iF-Feistel, we split a round function into the *key insert operation* $*$ and *nonlinear bijective function* F, where the operation $*$ is a function to mix an input with a round key by a simple operation, e.g., an XOR ($\oplus$) or a modular addition ($\boxplus$). The K_iF-Feistel is used widely to design Feistel ciphers. As an existing result for the security of the K_iF-Feistel, Knudsen and Rijmen showed the 7-round known-key distinguisher [KR07]. As another important result for the security of the K_iF-Feistel, Isobe and Shibutani showed generic key recovery attacks [IS13] by using the all subkeys recovery attack [IS12]. Let n and κ be the block length and the key length, respectively. Then, Isobe et al. showed that 5-, 7- and 9-round K_iF-Feistel ciphers with $\kappa = n$, $\kappa = 3n/2$, and $\kappa = 2n$ can be attacked, respectively[1].

Shirai et al. proposed the diffusion switching mechanism [SS06], which is efficient to design practical and secure Feistel ciphers. In this technique, a linear permutation P is added after a function F, and different linear functions are used in every round. Shirai et al. showed that this technique improves the security against differential and linear cryptanalyses. We call this network K_iFP_i-Feistel.

3.2.3 Knudsen's attack

We show the outline of the Knudsen's attack. The target is the F_{K_i}-Feistel whose round functions consist of permutations, but this attack is also valid against both the K_iF-Feistel and the K_iFP_i-Feistel. For the detailed procedure and evaluation, see the original paper [Knu02].

We show the outline of the Knudsen's attack on the 5-round F_{K_i}-Feistel (see the left of Fig. 3.3). The probability that input differences $(0, \alpha)$ derives output differences $(\cdot, \alpha)$ is 0 for the 4-round F_{K_i}-Feistel, where $\alpha \neq 0$ and $\cdot$ is any differences. On the other hand, the probability satisfying $(0, \alpha) \to (\cdot, \alpha)$ is $2^{-n/2}$ for random permutations. Knudsen showed that the correct key can be recovered by using $(\sqrt{\kappa} \times 2^{n/4+1/2})$ chosen plaintexts. The time complexity is at most $2^{n/4+1/2}(2^\kappa + 2^{\kappa-1} + \cdots + 2 + 1) \approx 2^{\kappa + \frac{n/2+3}{2}}$.

We show the outline of the Knudsen's attack on the 6-round F_{K_i}-Feistel (see the right of Fig. 3.3). The probability that input differences $(0, \alpha)$ derives output differences $(\alpha, 0)$ is 0 for the 5-round F_{K_i}-Feistel, where $\alpha \neq 0$. On the other hand, the probability satisfying $(0, \alpha) \to (\alpha, 0)$ is 2^{-n} for random permutations. Knudsen showed that the correct key can be recovered by using $(\kappa \times 2^{n/2})$ chosen plaintexts. The time complexity is at most $2^{n/2}(2^\kappa + 2^{\kappa-1} + \cdots + 2 + 1) \approx 2^{\kappa+1+n/2}$. For many Feistel ciphers, the total secret key size is the same as the block length (i.e., n bits). Moreover the round key size is the same as the input size of the round function (i.e., $n/2$ bits). Unfortunately, Knudsen's attack is not efficient for such Feistel ciphers, because the time complexity is $2^{\kappa+1+n/2} = 2^{n+1}$, and it is almost the same as the complexity of the brute force attack.

[1] Notice that K_iF-Feistel and K_iFP_i-Feistel are different from Feistel-2 in [IS13], because Feistel-2 can use completely different functions in every round. However, Feistel-2 includes K_iF-Feistel and K_iFP_i-Feistel.

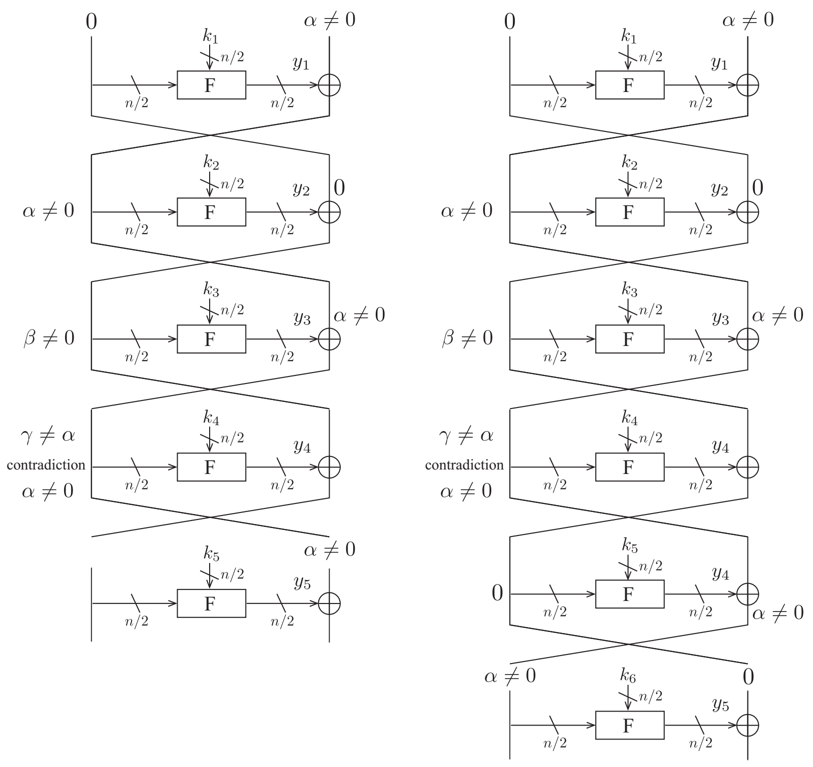


Figure 3.3: Knudsen's key recovery attacks

3.3 New Properties for the K_iF- and K_iFP_i-Feistel

We first propose several properties for 5-round Feistel networks. By exploiting our new properties, we show key recovery attacks on the K_iF-Feistel and the K_iFP_i-Feistel in Section 3.4 and Section 3.5, respectively. Figure 3.4 shows the 5-round Feistel network. When $F_1, F_2, \ldots, F_5$ are the same function for all i, it is the K_iF-Feistel. When F_i is expressed in $P_i \circ F$, it is the K_iFP_i-Feistel. Let A and B be the input of F_2 and F_4, respectively. In our properties, we pay attention to texts satisfying $A = B$.

Property 3.1. *In 5-round K_iF-Feistel, $c_L = p_L$ iff $A = B$.*

Property 3.2. *In 5-round K_iFP_i-Feistel, $c_L = P_4(P_2^{-1}(p_L \oplus L_3)) \oplus L_3$ iff $A = B$.*

Proof of Property 3.1 and 3.2 . For the 5-round Feistel network, c_L is calculated from p_L as follows:

$$c_L = p_L \oplus F_2(A) \oplus F_4(B).$$

For the 5-round K_iF-Feistel, the function F_2 is the same as the function F_4. Then a text always satisfies $c_L = p_L$ if and only if the text satisfies $A = B$.

For the 5-round K_iFP_i-Feistel, we can get the following equation:

$$F(A) = P_2^{-1}(p_L \oplus L_3), \quad F(B) = P_4^{-1}(c_L \oplus L_3).$$

A text always satisfies $F(A) = F(B)$ if and only if the text satisfies $A = B$. Then a text always satisfies the following equation:

$$P_2^{-1}(p_L \oplus L_3) = P_4^{-1}(c_L \oplus L_3),$$

if and only if the text satisfies $A = B$. Then a text always satisfies $c_L = P_4(P_2^{-1}(p_L \oplus L_3)) \oplus L_3$ if and only if the text satisfies $A = B$. □

Both Property 3.1 and 3.2 do not depend on the key insert operation $*$. On the other hand, the following property depends the key insert operation.

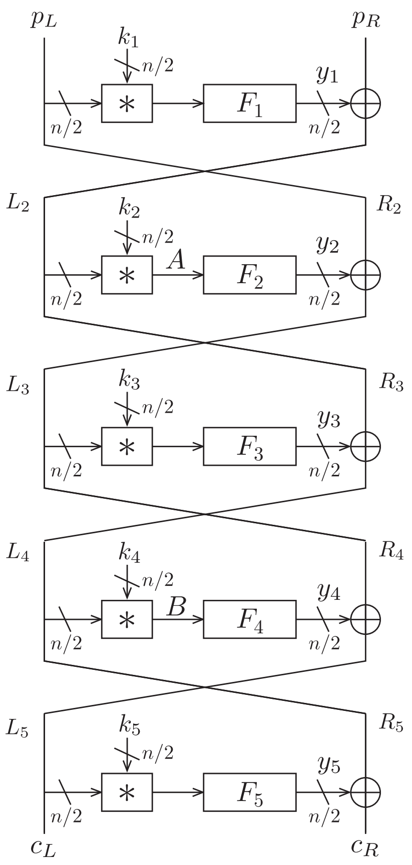


Figure 3.4: New properties for the 5-round Feistel network

Table 3.2: Summary of satisfied properties

Target	Property 3.1	Property 3.2	Property 3.3
$\oplus K_iF$	✓	-	✓
$\boxplus K_iF$	✓	-	-
$\oplus K_iFP_i$	-	✓	✓
$\boxplus K_iFP_i$	-	✓	-

Property 3.3. *In 5-round Feistel network which uses XOR as the key insert operation, $L_3 = k_3 \oplus F_3^{-1}(k_2 \oplus k_4)$ iff $A = B$. In other words, L_3 has the fixed value if the text satisfies $A = B$.*

Proof of Property 3.3. For the 5-round Feistel network, L_3 is calculated from $k_3 \oplus F_3^{-1}(L_2 \oplus L_4)$, where $L_2 = A \oplus k_2$ and $L_4 = B \oplus k_4$. Then L_3 is calculated as follows:

$$L_3 = k_3 \oplus F_3^{-1}(A \oplus B \oplus k_2 \oplus k_4).$$

Then a text always satisfies $L_3 = k_3 \oplus F_3^{-1}(k_2 \oplus k_4)$ if and only if the text satisfies $A = B$. Since k_2, k_3 and k_4 are constant values, L_3 has the fixed value if the text satisfies $A = B$. □

Property 3.3 is satisfied for both the K_iF-Feistel and K_iFP_i-Feistel. We summarize each property in Table 3.2, where "✓" means that the property holds under the corresponding target.

3.4 New Key Recovery Attack on K_iF-Feistel

3.4.1 New Key Recovery Attack for 5-round K_iF-Feistel

We propose a new key recovery attack for the 5-round K_iF-Feistel by using Property 3.1. This attack is more efficient than the Knudsen's attack, and it is KPA rather than CPA.

Overview. We use only texts satisfying $p_L = c_L$, and we can get such a text by observing $2^{n/2}$ known plaintexts and corresponding ciphertexts. These texts always satisfy $A = B$ from Property 3.1. A is calculated by guessing k_1 and k_2, and B is calculated by guessing k_4 and k_5. If $A \neq B$ is satisfied, guessing keys are wrong. Moreover we can curtail the guessing of round keys k_2 and k_4 by calculating differences. We calculate A and A' from two different texts, and calculate ΔA from the difference between A and A'. Similarly, we calculate B and B' from two corresponding texts, and calculate ΔB from the difference between B and B'. Since it satisfies $A = B$ and $A' = B'$, $\Delta A = \Delta B$ holds. Moreover ΔA and ΔB do not depend on k_2 and k_4, respectively. As a result, it satisfies $\Delta A = \Delta L_2$ and $\Delta B = \Delta L_4$. Now ΔL_2 is computed from plaintexts by an exhaustive search over all $2^{n/2}$ possible round keys k_1. Similarly ΔL_4 is computed from ciphertexts by an exhaustive search over all $2^{n/2}$ possible round keys k_5. Consequently, we can execute the meet-in-the-middle-attack using two differences ΔL_2 and ΔL_4.

Procedure and Evaluation. The key recovery attack goes as follows.

1. We analyze about $2^{n/2}$ known plaintexts and pick a text satisfying $p_L = c_L$. We express this text as (p, c). By an exhaustive search over all $2^{n/2}$ possible round keys k_1, we calculate and store $X[k_1] = p_R \oplus F(p_L * k_1)$. In parallel, by an exhaustive search over all $2^{n/2}$ possible round keys k_5, we calculate and store $Y[k_5] = c_R \oplus F(c_L * k_5)$.
2. We pick another text satisfying $p_L = c_L$, and we express this text as (p', c'). Like the first step, we calculate $X'[k_1]$ and $Y'[k_5]$, respectively. Moreover we calculate $\Delta X[k_1]$ and $\Delta Y[k_5]$, respectively. We pick (k_1, k_5) satisfying $\Delta X[k_1] = \Delta Y[k_5]$ as key candidates.
3. By executing the second step several times, we recover the correct key k_1 and k_5.

For each key guess, the probability satisfying $\Delta X[k_1] = \Delta Y[k_5]$ is $2^{-n/2}$, then about $2^n \times 2^{-n/2} = 2^{n/2}$ keys will be left for each execution of the second step. By executing the second step 3 times, we can discard all wrong keys because the probability that wrong keys are left as candidates is $2^n \times (2^{-n/2})^3 = 2^{-n/2}$, and it is negligible. Each time complexity to calculate $X[k_1]$ and $Y[k_5]$ from one text satisfying $p_L = c_L$ is $2^{n/2}$. We analyze 1 text and 3 texts for the first step and the second step, respectively. Therefore the time complexity is $(1 + 3) \times 2 \times 2^{n/2} = 2^{n/2+3}$, and our attack uses $(1 + 3) \times 2^{n/2} = 2^{n/2+2}$ known plaintexts.

Experimental Results. For feasibility evaluation of our attack, we execute experiments for our attacks on the K_iF-Feistel with small block sizes. We evaluate by the average of 1,000 samples in every block length. We choose functions F and round keys randomly in all samples.

We show experimental results for the key recovery attack on the 5-round K_iF-Feistel. We use XOR for the key insert operation. When we use XOR for the key insert operation, this attack can recover two key candidates; one is the correct key (k_1, k_5), and the other is its reverse (k_5, k_1). The reason is as follows. If correct k_1 and k_5 are guessed, it always satisfies

$$\begin{aligned}(p_R \oplus F(p_L \oplus k_1) \oplus k_2) &\oplus (p'_R \oplus F(p'_L \oplus k_1) \oplus k_2) \\ &= (c_R \oplus F(c_L \oplus k_5) \oplus k_4) \oplus (c'_R \oplus F(c'_L \oplus k_5) \oplus k_4).\end{aligned}$$

Then, we can recover the correct key by executing the meet-in-the-middle-attack. However, since both $p_L = c_L$ and $p'_L = c'_L$ hold, it also satisfies

$$\begin{aligned}(p_R \oplus F(p_L \oplus k_5) \oplus k_2) &\oplus (p'_R \oplus F(p'_L \oplus k_5) \oplus k_2) \\ &= (c_R \oplus F(c_L \oplus k_1) \oplus k_4) \oplus (c'_R \oplus F(c'_L \oplus k_1) \oplus k_4).\end{aligned}$$

Then the meet-in-the-middle-attack always find the reverse (k_5, k_1). On the other hand, when we use a modular addition for the key insert operation, this attack can recover (k_1, k_5) correctly.

We evaluate the number of known plaintexts in our experiments. When the block length is 32 bits, we can get 4 texts satisfying $p_L = c_L$ by observing 4.0098×2^{16} known plaintexts. This is almost the same as theoretical value. Moreover, we can recover the correct key from 4 texts in our experiments.

3.4.2 New Key Recovery Attack for 6-round K_iF-Feistel

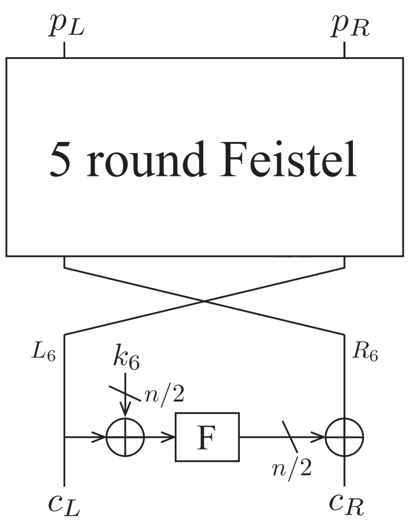


Figure 3.5: Key recovery attack on the 6-round K_iF-Feistel

We propose a new key recovery attack for the 6-round K_iF-Feistel (see Fig. 3.5). It is not clear how to attack 6-round K_iF-Feistel by exploiting only Property 3.1, and thus we additionally use Property 3.3. Therefore the target cipher of our new attack is limited to the 6-round K_iF-Feistel whose key insert operation is XOR.

Overview. For the 5-round K_iF-Feistel, we can exactly pick texts satisfying $A = B$ by observing $p_L = c_L$. However, we cannot pick those texts efficiently for the 6-round K_iF-Feistel, because we cannot know the value of R_6 which is c_L for the 5-round K_iF-Feistel. When the key insert operation is XOR, we can exploit Property 3.3. First, we introduce the following lemma derived from Property 3.3.

Lemma 3.1. *For $2^{n/2}$ chosen plaintexts such that p_L is a constant and p_R is chosen from all values, exactly one text satisfies $A = B$.*

Proof. We prepare $2^{n/2}$ chosen plaintexts such that p_L is a constant and p_R is chosen from all values. L_3 which is calculated from chosen plaintexts has all values ranging from 0 to $2^{n/2} - 1$ exactly once. From Property 3.3, a text satisfies $A = B$ if and only if the text satisfies $L3 = k_3 \oplus F_3^{-1}(k_2 \oplus k_4)$. Then the number of texts satisfying $A = B$ is exactly one in chosen texts. □

From Lemma 3.1, we can introduce the following lemma.

Lemma 3.2. *For $2^{n/2}$ chosen plaintexts such that p_L is a constant and p_R is chosen from all values, exactly one text satisfies $k_6 = F^{-1}(p_L \oplus c_R) \oplus c_L$.*

Proof. From Lemma 3.1, the number of texts satisfying $A = B$ is exactly one in chosen texts. From Property 3.1, a text always satisfies $R_6 = p_L$ if and only if the text satisfies $A = B$. Then, in chosen texts, exactly one text satisfies the following equation:

$$p_L = R_6 = F(c_L \oplus k_6) \oplus c_R.$$

Then exactly one text satisfies the following equation:

$$k_6 = F^{-1}(p_L \oplus c_R) \oplus c_L.$$

□

From Lemma 3.2, we know that the correct k_6 can be calculated only once from $2^{n/2}$ chosen plaintexts. We recover the round key by exploiting this property.

Procedure and Evaluation. The key recovery attack by using Lemma 3.2 goes as follows.

1. We choose $2^{n/2}$ plaintexts such that p_L is a constant and p_R is chosen from all values. Let $p[i] = (p_L, p_R[i])$ be an ith plaintext for $i = 1, \ldots, 2^{n/2}$. We query these chosen plaintexts and get corresponding ciphertexts.
2. We calculate $v = F^{-1}(p_L \oplus c_R[i]) \oplus c_L[i]$ $(1 \leq i \leq 2^{n/2})$, and store the number of occurrences in N_v. We discard v from key candidates if $N_v \neq 1$ holds.
3. By repeating the first and second steps several times, we recover the correct round key k_6.

The probability that $N_v = 1$ holds for wrong keys $v \neq k_6$ is e^{-1} because of the Poisson distribution with parameter $\lambda = 1$. Now we want to recover the $(n/2)$-bit round key k_6. When we repeat the first and second steps $n/2$ times, the probability that a wrong key is left is $2^{n/2} \times e^{-n/2}$ and it is negligible. Therefore attackers can recover the correct key with $\frac{n}{2}2^{n/2}$ complexity and $\frac{n}{2}2^{n/2}$ CPA.

Experimental Result. For feasibility evaluation of our attack, we execute experiments for our attacks on the K_iF-Feistel with small block sizes. We evaluate by the average of 1,000 samples in every block length. We choose functions F and round keys randomly in all samples.

Table 3.3: Experimental results of the attack on the 6-round K_iF-Feistel

Block length	Average number of texts	Theoretical number of texts
20	6.505×2^{10}	10×2^{10}
22	7.333×2^{11}	11×2^{11}
24	8.059×2^{12}	12×2^{12}
26	8.806×2^{13}	13×2^{13}
28	9.635×2^{14}	14×2^{14}
30	10.316×2^{15}	15×2^{15}
32	10.972×2^{16}	16×2^{16}
34	11.702×2^{17}	17×2^{17}
36	12.473×2^{18}	18×2^{18}
38	13.099×2^{19}	19×2^{19}
40	13.839×2^{20}	20×2^{20}

We show experimental results for the key recovery attack on the 6-round K_iF-Feistel. In this attack, we change the block length from 20 bits to 40 bits, and measure the number of texts to recover the correct key. We show experimental results in Table 3.3. In Table 3.3, "average number of texts" denotes the average number of texts to recover the correct key in our experiments, and "theoretical number of texts" denotes the theoretical number of texts to recover the correct key. From Table 3.3, we can know that "theoretical number of texts" is sufficient to recover the correct key.

3.5 New Key Recovery Attacks on K_iFP_i-Feistel

In this section, we propose new key recovery attacks for the K_iFP_i-Feistel by exploiting Property 3.2. It is not clear how to attack the K_iFP_i-Feistel by exploiting only Property 3.2, and thus we additionally use Property 3.3. Namely, the target cipher of our new attack is limited to the K_iFP_i-Feistel whose key insert operation is XOR.

3.5.1 A new property for 5-round K_iFP_i-Feistel

We propose a new key recovery attack for the 5-round K_iFP_i-Feistel by exploiting both Property 3.2 and 3.3. This attack is more efficient than the Knudsen's attack, and it is KPA rather than CPA.

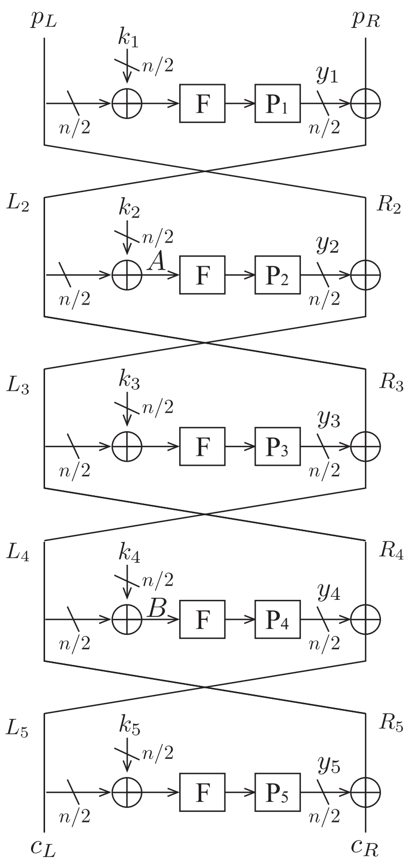


Figure 3.6: Key recovery attack on the 5-round K_iFP_i-Feistel

Overview. Figure 3.6 shows the 5-round K_iFP_i-Feistel. We pay attention to texts satisfying $A = B$. These texts always satisfies $c_L = P_4(P_2^{-1}(p_L \oplus L_3)) \oplus L_3$ because of Property 3.2. Moreover L_3 which is calculated from these texts is constant because of Property 3.3. By using two properties, we show that there is the following relationship between plaintexts and corresponding ciphertexts.

Lemma 3.3. *For the 5-round K_iFP_i-Feistel, the probability that the relationship between plaintexts and corresponding ciphertexts is expressed as*

$$c_L = P_4(P_2^{-1}(p_L)) \oplus k'$$

is about $2^{-n/2+1}$, where k' is a constant value which is determined by the round keys k_2, k_3 and k_4.

Proof. First, we consider texts satisfying $A = B$, and they always have the following relationship between plaintexts and corresponding ciphertexts because of Property 3.2:

$$\begin{aligned} c_L &= P_4(P_2^{-1}(p_L \oplus L_3)) \oplus L_3 \\ &= P_4(P_2^{-1}(p_L)) \oplus P_4(P_2^{-1}(L_3)) \oplus L_3 \\ &= P_4(P_2^{-1}(p_L)) \oplus k', \end{aligned} \tag{3.1}$$

where k' is a value depending on L_3. Moreover, L_3 is a constant value depending on k_2, k_3 and k_4 because of Property 3.3. Therefore, k' is a constant value depending on k_2, k_3 and k_4. Now we consider the probability satisfying Eq. (3.1). The probability satisfying $A = B$ is $2^{-n/2}$, and Eq. (3.1) is always satisfied in this case. The probability satisfying $A \neq B$ is $1 - 2^{-n/2}$, and the probability that Eq. (3.1) is satisfied by chance is $2^{-n/2}$. Then the total probability is $2^{-n/2} + (1 - 2^{-n/2}) \times 2^{-n/2} \approx 2^{-n/2+1}$. □

For a random value v except k', the probability satisfying $c_L = P_4(P_2^{-1}(p_L)) \oplus v$ is $2^{-n/2}$. Now we prepare $t2^{n/2}$ known plaintexts and corresponding ciphertexts. In this case, $N_{k'}$ is about $2t$ and N_v $(v \neq k')$ is about t. Then we can recover the correct k' by using t that we can distinguish the correct k' and the others. We think that we can recover the correct k' by using $t = O(n)$. This attack can only recover k'. However we can easily recover round keys by using k' once k' is recovered.

Procedure and Evaluation. The key recovery attack by using Lemma 3.3 goes as follows.

1. We prepare $t2^{n/2}$ known plaintexts and corresponding ciphertexts.
2. We calculate $v = c_L \oplus P_4(P_2^{-1}(p_L))$ and store the number of occurrences in N_v. We return v that N_v is the maximum number.

The time complexity of this attack is $t2^{n/2}$.

Experimental Results. For feasibility evaluation of our attack, we execute experiments for our attacks on the K_iFP_i-Feistel with small block sizes. We evaluate by the average of 1,000 samples in every block length. We choose functions F and round keys randomly in all samples.

We show experimental results for the key recovery attack on the 5-round K_iFP_i-Feistel. When the correct k' is calculated most (i.e., $N_{k'}$ is the maximum number), we define it as "the success." The time complexity is $t2^{n/2}$ by using KPA with $t2^{n/2}$. For parameter t, we use two parameters $t = n$ and $t = 2n$. When the block length is 32 bits, the success probability of this attack is 74.4% for $t = n$. Moreover the success probability of this attack is 99.5% for $t = 2n$.

3.5.2 New key recovery attack on 6-round K_iFP_i-Feistel

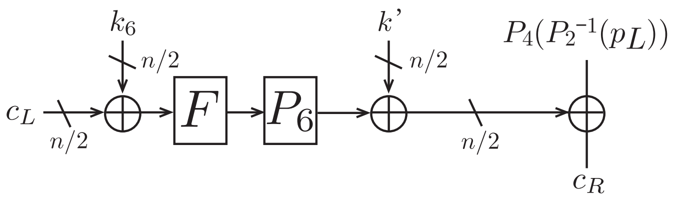


Figure 3.7: The relationship between plaintext and ciphertext of the 6-round K_iFP_i-Feistel

Overview. For the 5-round K_iFP_i-Feistel, there is small bias depending on k' between p_L and c_L. Then we can recover k' from p_L and c_L. However, we cannot attack the 6-round K_iFP_i-Feistel using this method because we cannot know the value of R_6 which is c_L for the 5-round K_iFP_i-Feistel. Then we first assume that texts satisfy $A = B$, and calculate R_6 as follows:

$$R_6 = P_4(P_2^{-1}(p_L)) \oplus k'.$$

The relationship between plaintexts and corresponding ciphertexts is expressed as follows:

$$\begin{aligned} P_6(F(c_L \oplus k_6)) &= c_R \oplus R_6 \\ &= c_R \oplus P_4(P_2^{-1}(p_L)) \oplus k', \end{aligned} \tag{3.2}$$

where k_6 is the round key for the 6-round. Figure 3.7 shows the circuit of the Eq. (3.2). Then if we exhaustively search for over all $2^{n/2}$ possible round keys k_6, we can recover k' from Eq. (3.2). However, this attack is not effective. The reason is that the probability satisfying the Eq. (3.2) is at most probability $2^{-n/2+1}$. If we recover k' by using this bias, we have to search for $2^{n/2}$ possible round keys k_6 against over $2^{n/2}$ texts. This time complexity is over 2^n.

We use the chosen ciphertext attack (CCA). We first prepare enormous chosen ciphertexts and corresponding plaintexts. Next, we pick some texts which are used to recover round keys. Finally, we guess k_6, and get k'. Since we guess k_6 against only picked texts, we can reduce the complexity. Note that we can also attack the 6-round K_iFP_i-Feistel by using CPA not CCA, because Feistel networks are involution. We show this attack by using CCA for simplicity in this section.

Picking Texts. We show how to pick some texts from chosen ciphertexts. We first prepare $2^{n/2}$ chosen ciphertexts such that c_L is a constant and c_R is chosen from all values. If c_L is a constant, we know $c_R \oplus P_4(P_2^{-1}(p_L))$ is uniquely determined by k_6 and k'. Moreover we know that Eq. (3.2) is satisfied with probability $2^{-n/2+1}$. Namely, the probability that $c_R \oplus P_4(P_2^{-1}(p_L))$ is the same as $P_6(F(c_L \oplus k_6)) \oplus k'$ is $2^{-n/2+1}$, and it is double compared with the others. Then we calculate $v = c_R \oplus P_4(P_2^{-1}(p_L))$ and store the number of occurrences in N_v. The number N_v is about $2^{n/2} \times 2^{-n/2+1} = 2$ when it satisfies $v = P_6(F(c_L \oplus k_6)) \oplus k'$. However the number N_v is about $2^{n/2} \times 2^{-n/2} = 1$ when it satisfies $v \neq P_6(F(c_L \oplus k_6)) \oplus k'$. By using this slight bias, we pick some texts which are used to recover round keys.

Lemma 3.4. *Let us choose $2^{n/2}$ ciphertexts such that c_L is a constant and c_R is chosen from all values, and query these chosen ciphertexts and get corresponding plaintexts. Next, let us calculate $v = c_R \oplus P_4(P_2^{-1}(p_L))$ and store the number of occurrences in N_v. Finally, let us pick (c_L, v) that N_v is the maximum number. Then, P_T denotes the probability that the picked text satisfies Eq. (3.2), and P_T is expressed as follows:*

$$P_T = \sum_{i=0}^{2^{n/2}} \frac{2^i e^{-2}}{i!} \left(\frac{1}{e} \sum_{j=0}^{i} \frac{1}{j!} \right)^{2^{n/2}-1} > 2^{-n/2}.$$

Proof. We calculate $v = c_R \oplus P_4(P_2^{-1}(p_L))$ and store the number of occurrences in N_v. Moreover we choose v that N_v is the maximum number. The probability that the chosen v is equal to $P_6(F(c_L \oplus k_6)) \oplus k'$ has the Poisson distribution with parameter $\lambda = 2$, then the probability that the number N_v is equal to i is given as follows:

$$P_c(i) = \frac{2^i e^{-2}}{i!},$$

On the other hand, the probability that the chosen v is equal to another value has the Poisson distribution with parameter $\lambda = 1$, then the probability that the number N_v is less than or equal to i is given as follows:

$$P_f(i) = \sum_{j=0}^{i} \frac{1^i e^{-1}}{j!} = \frac{1}{e} \sum_{j=0}^{i} \frac{1}{j!}.$$

Then, the probability of Lemma 3.4 is given as follows:

$$\begin{aligned} P_t &= \sum_{i=0}^{2^{n/2}} P_c(i) \times (P_f(i))^{2^{n/2}-1} \\ &= \sum_{i=0}^{2^{n/2}} \frac{2^i e^{-2}}{i!} \left(\frac{1}{e} \sum_{j=0}^{i} \frac{1}{j!} \right)^{2^{n/2}-1} \\ &= \frac{1}{e^{2^{n/2}+1}} \sum_{i=0}^{2^{n/2}} \frac{2^i}{i!} \left(\sum_{j=0}^{i} \frac{1}{j!} \right)^{2^{n/2}-1}. \end{aligned}$$

Since it satisfies the following equation

$$\sum_{j=0}^{i} \frac{1}{j!} = e - \frac{e}{(i+1)!},$$

it satisfies the following equation

$$\begin{aligned} P_t &= \frac{1}{e^{2^{n/2}+1}} \sum_{i=0}^{2^{n/2}} \frac{2^i}{i!} \left(e - \frac{e}{(i+1)!} \right)^{2^{n/2}-1} \\ &= \frac{1}{e^2} \sum_{i=0}^{2^{n/2}} \frac{2^i}{i!} \left(1 - \frac{1}{(i+1)!} \right)^{2^{n/2}-1}. \end{aligned}$$

Next we show the behavior of P_t. We pay attention to i that $(i+1)! \approx 2^{n/2}$ is satisfied. Then $(1 - 1/(i+1)!)^{2^{n/2}-1}$ is close to e^{-1}. Moreover it satisfies $i = \Gamma^{-1}(2^{n/2})$ and $i! = \frac{(i+1)!}{i+1} = \frac{2^{n/2}}{\Gamma^{-1}(2^{n/2})+1}$. Therefore, it satisfies the following equation

$$P_t > \frac{1}{e^3} \frac{(\Gamma^{-1}(2^{n/2}) + 1) \times 2^{\Gamma^{-1}(2^{n/2})}}{2^{n/2}}.$$

Moreover it satisfies the following equation

$$2^{n/2} \times P_t \approx \frac{(\Gamma^{-1}(2^{n/2}) + 1) 2^{\Gamma^{-1}(2^{n/2})}}{e^3} > 1,$$

for $\Gamma^{-1}(2^{n/2}) \geq 3$. Therefore P_t is higher than $2^{-n/2}$. $\square$

For the picked text (c_L, v), we exhaustively search for over all $2^{n/2}$ possible round keys k_6 and calculate k' from $k' = P_6(F(c_L \oplus k_6)) \oplus v$. When we use $2^d \times 2^{n/2}$ chose ciphertexts, the number that the correct (k_6, k') is calculated is about $2^d \times P_t$. On the other hand, the number that the wrong tuple is calculated is about $2^d \times 2^{-n/2}$ on average. Since P_t is greater than $2^{-n/2}$, we can distinguish the correct (k_6, k').

Procedure and Evaluation. We show a key recovery attack on the 6-round K_iFP_i-Feistel by using Lemma 3.4. The attack goes as follows.

1. We choose $2^{n/2}$ ciphertexts such that c_L is a constant and c_R is chosen from all values, i.e., $c[i] = (c_L, c_R[i])$ for $i = 1, \ldots, 2^{n/2}$. We query these chosen ciphertexts and get corresponding plaintexts.

2. We pick some texts by using Lemma 3.4. We calculate $v = c_R[i] \oplus P_4(P_2^{-1}(p_L[i]))$ and store the number of occurrences in N_v. We pick (c_L, v) that N_v is the maximum number.

3. For the picked text (c_L, v), we exhaustively search for over all $2^{n/2}$ possible round keys k_6 and calculate k'. We count the number that (k_6, k') is calculated.

4. We repeat the first, second and third steps 2^d times. We return (k_6, k') that is calculated most.

Now we give the suitable number of chosen ciphertext. We consider $n = 128$ because many block ciphers have the 128-bit block size. For $n = 128$, it satisfies $P_t > 2^{-44}$ because of Lemma 3.4. By repeating the first, second and third steps $2^{3n/8}$ times (namely, using $2^{7n/8}$ chosen ciphertexts), the number that the correct (k_6, k') is calculated is at least $2^{3n/8} \times 2^{-44} = 16$. Next, we consider the number that one of wrong keys is calculated, and the probability that the number is over 16 is at most $\frac{1}{16!}$ [STKT06]. Consequently, our new attack can recover the correct (k_6, k').

We show the time complexity. The complexity for the second step is about $2^{n/2}$, and the complexity for the third step is about $2^{n/2}$. Since we repeat the first, second and third steps $2^{3n/8}$ times, the total time complexity is $2^{7n/8+1}$.

Experimental Result. For feasibility evaluation of our attack, we execute experiments for our attacks on the K_iFP_i-Feistel with small block sizes. We evaluate by the average of 1,000 samples in every block length. We choose functions F and round keys randomly in all samples.

We show experimental results for the key recovery attack on the 6-round K_iFP_i-Feistel, and our attack uses $2^{7n/8}$ chosen plaintexts. When the correct key (k, k') is calculated most, we define it as "the success." We summarize experimental results in Fig. 3.8, where the horizontal axis shows block lengths, the left vertical axis shows the number that the correct key is calculated, and the right vertical axis shows the success probability on our attack. From Fig. 3.8, the number that picked texts satisfy Eq. (3.2) is always higher than the theoretical value. Unfortunately the success probability is small when the block length is small. However the success probability is big when the block length is big. Therefore we expect that our attack can recover the correct key of the 6-round K_iFP_i-Feistel.

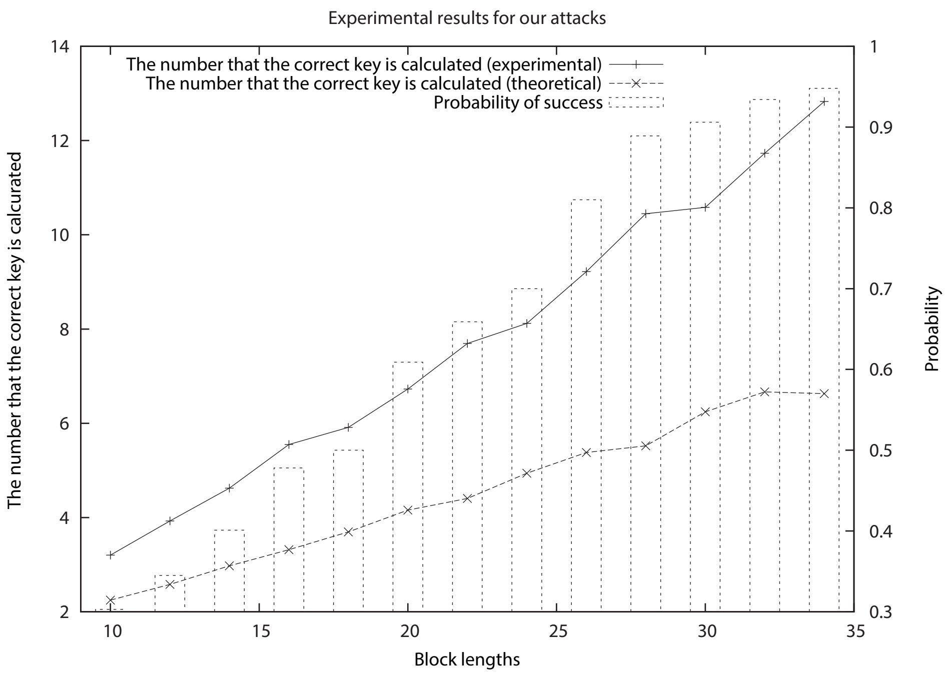


Figure 3.8: Experimental results of the attack on the 6-round K_iFP_i-Feistel

3.6 Conclusion

We revisit Knudsen's Feistel networks. It is still difficult for designers to design Knudsen's Feistel networks, and many Feistel ciphers use more efficient and practical Feistel networks. In this chapter, we introduce such Feistel networks, the K_iF-Feistel and the K_iFP_i-Feistel. We show new properties using the relationship between plaintexts and ciphertexts. By using these properties, we show that the 6-round K_iF- and K_iFP_i-Feistel are not secure.

Chapter 4

Fast Fourier Transform (FFT) Key Recovery Technique

Abstract– We propose a new technique for the integral cryptanalysis called the Fast Fourier Transform (FFT) key recovery. When N chosen plaintexts are required for the integral characteristic and the guessed key is κ bits, a straightforward key recovery requires the time complexity of $O(N2^{\kappa})$. However, the FFT key recovery only requires the time complexity of $O(N+\kappa 2^{\kappa})$. As a previous result using FFT, at ICISC 2007, Collard et al. proposed that FFT can reduce the time complexity of a linear cryptanalysis. We show that FFT can also reduce the complexity of the integral cryptanalysis. Moreover, the advantage of the FFT key recovery is that the estimation of the complexity is very simple. We first show the complexity of the FFT key recovery against three structures, an Even-Mansour scheme, key-alternating cipher, and Feistel structure. As examples of these structures, we show integral cryptanalyses against PRØST, AES, PRESENT, and CLEFIA. An 8-round PRØST $\tilde{P}_{128,K}$ can be attacked with an approximate time complexity of $2^{79.6}$. For the key-alternating cipher, a 6-round AES and 10-round PRESENT can be attacked with approximate time complexities of $2^{51.7}$ and $2^{97.4}$, respectively. For the Feistel structure, a 12-round CLEFIA can be attacked with approximate time complexities of $2^{87.5}$.
Keywords– Block ciphers, Integral cryptanalysis, Even-Mansour, Key-alternating cipher, Feistel structure, FFT, FWHT

4.1 Introduction

An integral cryptanalysis was first proposed by Daemen et al. to evaluate the security of SQUARE [DKR97], and then Knudsen and Wagner formalized this attack as the integral cryptanalysis [KW02]. This attack uses N chosen plaintexts (CPs) and the corresponding ciphertexts. Generally, the integral cryptanalysis first constructs an integral characteristic and then recovers the secret key by using the characteristic. In the integral characteristic, plaintexts are prepared in which the XOR of the Rth round output is 0. In the key recovery, Rth round outputs are recovered from ciphertexts by guessing round keys used in the last several rounds. If the guessed key is correct, the XOR of the recovered texts is always 0. In this chapter, we focus on the key recovery for the integral cryptanalysis.

For the key recovery using integral characteristic, we can use an exhaustive search for the subkey bits effectively related to computing the XOR that is expected to be 0 for the correct subkey [DKR97]. There exist several studies to improve the key recovery. Ferguson et al. proposed the partial-sum technique to reduce the number of S-box lookups [FKL+00] for a reduced round AES using memory to save a partial-sum of the integrals. Shimoyama et al. proposed a technique that regards key recovery as to solve the system of equations using a linearization, and the technique requires more chosen texts but reduces the time complexity of the attack when the algebraic degree of the round function is small [SMKT99]. This chapter proposes another technique to reduce the time complexity of the key recovery without requiring more texts.

We propose a new improved technique for the integral cryptanalysis called the Fast Fourier Transform (FFT) key recovery. The FFT is recently used to accelerate the key recovery step in several attacks. In 2007, Collard et al. first proposed a linear cryptanalysis using the

Table 4.1: Summary of FFT key recovery, where κ, κ_1, and κ_2 be defined in Sect. 4.3, 4.4, and 4.5.

Target structure	Previous	FFT	Reference
Even-Mansour	$O(2^{2\kappa})$	$O(\kappa 2^{\kappa})$	Sect. 4.3
Key-alternating	$O(2^{\kappa_1+2\kappa_2})$	$O(\kappa_2 2^{\kappa_1+\kappa_2})$	Sect. 4.4
Feistel	$O(2^{2\kappa_1+2\kappa_2})$	$O(\kappa_1 2^{\kappa_1} + \kappa_2 2^{\kappa_2})$	Sect. 4.5

Table 4.2: Comparison of attack results. Time column only includes the time complexity of the key recovery step, and it does not include the time complexity to count the frequency of the partial bit-string of ciphertexts corresponding to the chosen plaintexts (CPs).

Target cipher	#Round	Data (CP)	Time	Technique	Reference
PRØST $\tilde{P}_{128,K}$	8	2^{64}	$2^{79.6}$	FFT	Sect. 4.3
PRØST $\tilde{P}_{256,K}$	9	2×2^{64}	$2^{80.9}$	FFT	Sect. 4.3
AES	6	6×2^{32}	$6 \times 2^{50} \approx 2^{52.6}$	Partial-sum	[FKL+00]
AES	6	6×2^{32}	$2^{51.7}$	FFT	Sect. 4.4
PRESENT	10	$2^{22.4}$	$2^{99.3}$	Partial-sum	[WW13]
PRESENT	10	$2^{22.4}$	$2^{97.4}$	FFT	Sect. 4.4
CLEFIA	12	13×2^{112}	$13 \times 2^{106} \approx 2^{109.7}$	MITM, Partial-sum	[SW12b]
CLEFIA	12	5×2^{112}	$2^{87.5}$	MITM, FFT	Sect. 4.5

FFT [CSQ07], and then Nguyen et al. extended it to a multi-dimensional linear cryptanalysis [NWWL10, NWW11]. Moreover, Bogdanov et al. proposed a zero correlation cryptanalysis using the FFT in 2013 [BGW+13]. We now point out that the FFT can also be applied to the integral cryptanalysis, and propose attack frameworks for the integral cryptanalysis with the FFT key recovery. We focus on three structures for block ciphers, i.e., an Even-Mansour scheme [EM97], key-alternating cipher [BKL+12], and Feistel structure, and we estimate the security against the integral cryptanalysis. Table 4.1 shows results of the FFT key recovery, and Table 4.2 summarizes results of integral cryptanalyses against specific ciphers.

4.2 Previous Works

4.2.1 Integral Cryptanalysis

Knudsen and Wagner proposed the integral cryptanalysis [KW02]. It first searches for an integral characteristic of a target block cipher and then recovers the secret key by using the characteristic. Nowadays, many integral cryptanalyses have been proposed against specific ciphers [KW02, LWZ11, WZ11, YPK02, ZRHD08].

Integral Characteristic. An integral characteristic is generally constructed by the propagation of the integral property. We define four integral properties as follows:

- ALL ($\mathcal{A}$) : Every value appears the same number in the multiset.
- BALANCE ($\mathcal{B}$) : The XOR of all texts in the multiset is 0.
- CONSTANT ($\mathcal{C}$) : The value is fixed to a constant for all texts in the multiset.
- UNKNOWN ($\mathcal{U}$) : The multiset is indistinguishable from one of n-bit random values.

Knudsen and Wagner showed that AES has the 4-round integral characteristic by the propagation of integral properties [KW02]. It uses 2^{32} chosen plaintexts, and each byte after encrypting 4 rounds satisfies $\mathcal{B}$.

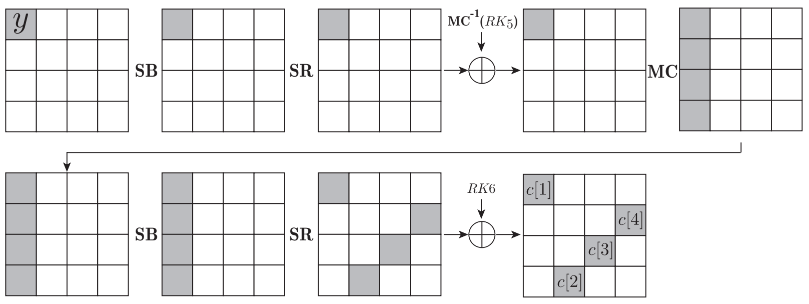


Figure 4.1: Integral cryptanalysis against 6-round AES

Key Recovery. The Rth round output is recovered from ciphertexts by guessing round keys used in the last several rounds. If the guessed key is incorrect, the recovered texts are expected to behave as random texts. On the other hand, if the guessed key is correct, the XOR of the recovered texts is always 0.

For instance, Fig. 4.1 shows the key recovery of the integral cryptanalysis against a 6-round AES. Here, we now have 2^{32} ciphertexts for the 6-round AES, and know that y satisfies $\mathcal{B}$. Let $c[i]$ be bytes in the ciphertexts as shown in Fig. 4.1, and c_n denotes the nth ciphertext. In this case, the XOR of y is calculated from 2^{32} ciphertexts as

$$\bigoplus_{n=1}^{2^{32}} S_5(S_1(c_n[1] \oplus K_1) \oplus S_2(c_n[2] \oplus K_2) \\ \oplus S_3(c_n[3] \oplus K_3) \oplus S_4(c_n[4] \oplus K_4) \oplus K_5) = 0, \qquad (4.1)$$

where $S_1, S_2, \ldots, S_5$ are S-boxes, each of which consists of the inverse of the AES S-box and a multiplication by a field element from the inverse of the AES MDS matrix. Moreover, K_1, K_2, K_3, and K_4 are calculated from RK_6, and K_5 is calculated from RK_5. Therefore, the total bit length of the guessed keys is 40 bits. Analysis using a straightforward method incurs the approximate time complexity of $2^{32+40} = 2^{72}$.

4.2.2 Ferguson et al.'s Partial-Sum Technique

Ferguson et al. proposed an improved technique to reduce the time complexity in the key recovery of the integral cryptanalysis, and it is refereed as the partial-sum technique [FKL$^+$00]. While five keys are guessed in the same time in the straightforward method, a part of five keys are first guessed in the partial-sum technique. Specifically, the frequency of 32 bits $(c[1], c[2], c[3], c[4])$ is first stored into a voting table, two keys in K_1, K_2, K_3, and K_4 are then guessed, and reduce the size of the voting table.

1. The frequency of $(c[1], c[2], c[3], c[4])$ is stored into a voting table. The size of the voting table is 2^{32}.
2. For every element in the voting table, K_1 and K_2 are guessed, compute $x_{K_1,K_2} = S_1(c[1]) \oplus K_1) \oplus S_2(c[2]) \oplus K_2)$. Elements in the voting table are updated as $(x_{K_1,K_2}, c[3], c[4])$. The time complexity in this step is $2^{32+16} = 2^{48}$, and the size of the voting table is reduced to 2^{24}.
3. For every element in the voting table, K_3 is guessed, compute $y_{K_1,K_2,K_3} = x_{K_1,K_2} \oplus S_3(c[3]) \oplus K_3)$. Elements in the voting table are updated as $(y_{K_1,K_2,K_3}, c[4])$. The time complexity in this step is $2^{24+24} = 2^{48}$, and the size of the voting table is reduced to 2^{16}.
4. For every element in the voting table, K_4 is guessed, compute $z_{K_1,K_2,K_3} = y_{K_1,K_2,K_3} \oplus S_4(c[4]) \oplus K_4)$. Elements in the voting table are updated as $(z_{K_1,K_2,K_3,K_4}$. The time complexity in this step is $2^{16+32} = 2^{48}$, and the size of the voting table is reduced to 2^8.
5. For every element in the voting table, K_5 is guessed, compute $S_5(z_{K_1,K_2,K_3,K_4} \oplus K_5)$. The time complexity in this step is $2^{8+40} = 2^{48}$.

Since each time complexity from step 2 to 5 is 2^{48}, the total time complexity is $4 \times 2^{48} = 2^{50}$ S-box lookups.

4.2.3 Collard et al.'s FFT Key Recovery

Collard et al. showed a linear cryptanalysis using FFT in 2007. When the key recovery of the linear cryptanalysis [Mat93] uses N ciphertexts $c_1, c_2, \ldots, c_N$, it guesses keys K and calculates

$$\sum_{n=1}^{N} f(c_n \oplus K). \tag{4.2}$$

It finally recovers the correct K to evaluate Eq. (4.2) for several possible Ks. Here, let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function, which is generated from a linear approximate equation. The evaluation of Eq.(4.2) requires the time complexity of $O(N2^\kappa)$ using a straightforward method, and the size of N is often larger than 2^κ. Collard et al. showed that the evaluation of Eq. (4.2) requires the time complexity of approximately $O(\kappa 2^\kappa)$. Nguyen et al. then noticed that the Fast Walsh-Hadamard Transform (FWHT) can be used instead of the FFT [NWWL10]. Hereinafter, we show the calculation method using the FWHT.

Two κ-dimensional vectors v and w are first created, where v is generated from Boolean function f, and w is generated from the set of ciphertexts as indicated below.

$$\begin{aligned} v_i &= f(i), \\ w_i &= \#\{1 \le n \le N | c_n = i\}. \end{aligned}$$

A κ-dimensional vector u is calculated from v and w as

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{2^\kappa-1} \end{bmatrix} = \begin{bmatrix} v_0 & v_1 & v_2 & \cdots & v_{2^\kappa-1} \\ v_1 & v_0 & v_3 & \cdots & v_{2^\kappa-2} \\ v_2 & v_3 & v_0 & \cdots & v_{2^\kappa-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{2^\kappa-1} & v_{2^\kappa-2} & v_{2^\kappa-3} & \cdots & v_0 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{2^\kappa-1} \end{bmatrix}. \tag{4.3}$$

In this case, u_K is equal to the results of Eq. (4.2). Therefore, if Eq. (4.3) can be calculated quickly, the time complexity is reduced. Equation (4.3) is simply expressed as $u = \boldsymbol{V} \times w$. Here, matrix $\boldsymbol{V}$ consists of four $(2^{\kappa-1})$-dimensional block matrices $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ as

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{V}_2 \\ \boldsymbol{V}_2 & \boldsymbol{V}_1 \end{bmatrix}.$$

From the diagonalization of $\boldsymbol{V}$, we have

$$\boldsymbol{V} = \frac{1}{2} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{I} & -\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_1 + \boldsymbol{V}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{V}_1 - \boldsymbol{V}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{I} & -\boldsymbol{I} \end{bmatrix},$$

where $\boldsymbol{I}$ is an identity matrix. Since $\boldsymbol{V}_1 + \boldsymbol{V}_2$ and $\boldsymbol{V}_1 - \boldsymbol{V}_2$ have the same structure as $\boldsymbol{V}$, we obtain

$$\boldsymbol{V} = \frac{1}{2^\kappa} \times \boldsymbol{H}_{2^\kappa} \times \mathrm{diag}(\boldsymbol{H}_{2^\kappa} v) \times \boldsymbol{H}_{2^\kappa},$$

where $\boldsymbol{H}_{2^\kappa}$ is the 2^κ-dimensional Walsh matrix[1], and $\mathrm{diag}(\boldsymbol{H}_{2^\kappa} v)$ is a diagonal matrix whose element in the ith row and ith column is the ith element of $\boldsymbol{H}_{2^\kappa} v$. Therefore, Eq. (4.3) is expressed as

$$u = \boldsymbol{V} \times w = \frac{1}{2^\kappa} \boldsymbol{H}_{2^\kappa} \times \mathrm{diag}(\boldsymbol{H}_{2^\kappa} v) \times \boldsymbol{H}_{2^\kappa} w.$$

The procedure to calculate u is given below.

1. Let us calculate $\hat{v} = \boldsymbol{H}_{2^\kappa} v$. Then, Eq. (4.3) is expressed as $u = \frac{1}{2^\kappa} \boldsymbol{H}_{2^\kappa} \times \mathrm{diag}(\hat{v}) \times \boldsymbol{H}_{2^\kappa} w$.

[1] The Walsh matrix is defined as the following recursive formulae.

$$\boldsymbol{H}_{2^1} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \boldsymbol{H}_{2^\kappa} = \begin{bmatrix} \boldsymbol{H}_{2^{\kappa-1}} & \boldsymbol{H}_{2^{\kappa-1}} \\ \boldsymbol{H}_{2^{\kappa-1}} & -\boldsymbol{H}_{2^{\kappa-1}} \end{bmatrix} \quad (\kappa \ge 2).$$

2. Let us calculate $\hat{w} = \boldsymbol{H}_{2^\kappa} w$. Then, Eq. (4.3) is expressed as $u = \frac{1}{2^\kappa}\boldsymbol{H}_{2^\kappa} \times \mathrm{diag}(\hat{v})\hat{w}$.
3. Let us calculate $\hat{u}$ whose $\hat{u}_i$ is calculated from $\hat{v}_i \times \hat{w}_i$, and then calculate $u = \frac{1}{2^\kappa}\boldsymbol{H}_{2^\kappa}\hat{u}$.

In the first and second steps, we calculate the multiplication of the Walsh matrix using the FWHT, and each time complexity is approximately the time of $\kappa 2^\kappa$ additions. In the third step, we first calculate 2^κ multiplications of the κ-bit integers, where we regard that the complexity of one multiplication is equal to that for κ additions. We next calculate the FWHT, and the time complexity is approximately the time of $\kappa 2^\kappa$ additions. We finally calculate the division by 2^κ, but the time complexity is negligible because it can be computed by a κ-bit shift. Therefore, the time complexity of the third step is approximately $2\kappa 2^\kappa$. Thus, the total time complexity is approximately the time of $4\kappa 2^\kappa$ additions.

4.3 FFT Key Recovery for Even-Mansour Scheme

4.3.1 Even-Mansour Scheme

The Even-Mansour scheme is a famous scheme to construct an n-bit block cipher from an n-bit permutation P. The encryption is executed using two n-bit keys K_1 and K_2 as

$$c = K_2 \oplus P(p \oplus K_1),$$

where p and c denote a plaintext and ciphertext, respectively [EM97]. The Even-Mansour scheme has recently been a popular topic of discussion [DKS12, BKL+12].

4.3.2 Attack Framework

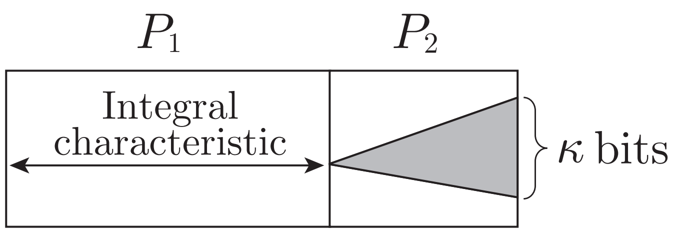


Figure 4.2: FFT key recovery for Even-Mansour scheme

We first split permutation P into two permutations, P_1 and P_2, as $P = P_2 \circ P_1$ (see Fig. 4.2). We assume that P_1 has an integral characteristic with N chosen plaintexts. Moreover, any one bit is diffused to κ bits by P_2. The straightforward key recovery requires $O(\min\{N2^\kappa, 2^{2\kappa}\})$ time complexity.

Let f be a Boolean function such that the input is κ bits of the output of P_2 and the output is any one bit of the input of P_2. In this case, the key recovery can be expressed as

$$\bigoplus_{i=1}^{N} f(c'_i \oplus K'_2) = 0,$$

where c'_i and K'_2 are truncated to κ bits from c_i and K_2, respectively. The FFT key recovery calculates the summation of integers, and we have

$$\sum_{i=1}^{N} f(c'_i \oplus K'_2) = 0 \bmod 2.$$

We can efficiently evaluate this equation using the FWHT, and the time complexity is $4\kappa 2^\kappa$ additions.

4.3.3 Integral Cryptanalysis against PRØST Even-Mansour

Specification of PRØST. PRØST is an authenticated encryption scheme, which was submitted to the CAESAR competition [KLL+14a, KLL+14b]. PRØST has a PRØST permutation P_n ($n = 128$ or 256), and the input and output size is $2n$ bits. PRØST permutation adopts a substitution-permutation network, and the state is represented as $4 \times d$ matrix, where $d = 16$ and 32 for $n = 128$ and 256, respectively. PRØST permutation consists of T rounds, where $T = 16$ and 18 for $n = 128$ and 256, respectively. The round function of PRØST consists of four component functions: `SubRows`, `ShiftPlanes`, `MixSlices`, and `AddConstant`. Each function is defined as follows:

- `SubRows` substitutes each 4-bit value in the matrix into another 4-bit value by an S-box.
- `ShiftPlanes` cyclically shifts the jth row by $\pi_{2-(i \bmod 2)}(j)$ nibbles, where a nibble stands for 4 bits, to the left for the ith round, where π_i is defined as

$$(\pi_1, \pi_2) = \begin{cases} (\{0,2,4,6\}, \{0,1,8,9\}) & \text{for } n{=}128, \\ (\{0,4,12,26\}, \{1,24,26,31\}) & \text{for } n{=}256. \end{cases}$$

- `MixSlices` diffuses four 4-bit values within each column by a linear function.
- `AddConstant` is an XOR operation where the round constant is XORed to the matrix.

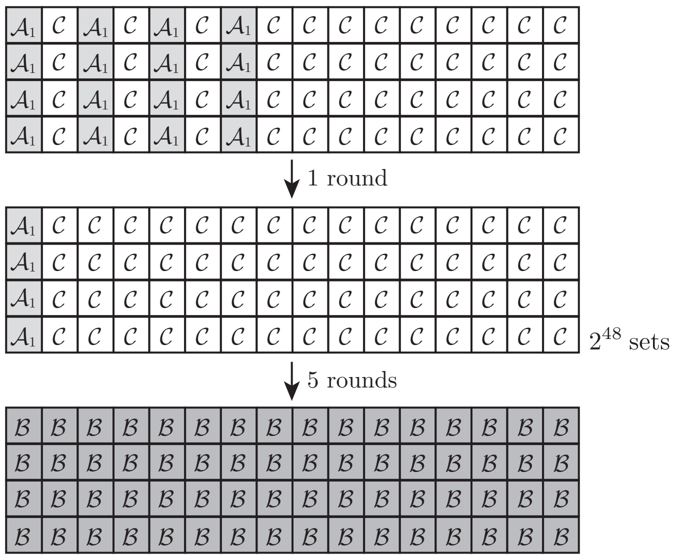


Figure 4.3: 6-round integral characteristic of PRØST P_{128}

Integral Characteristic of PRØST. We experimentally search for integral characteristics of PRØST. We set a column of the second round input as $\mathcal{A}$ and observe the 6th round output. The XOR of the output depends on the value of the constant nibbles of the second round input. However, we can expect that bit positions whose XOR values are always zero are $\mathcal{B}$ by changing the value of the constant nibbles of the second round input. We try 1024 randomly chosen values for the constant nibbles, and the number of trials is sufficient to determine that the output bits satisfy $\mathcal{B}$ by assuming that the non-$\mathcal{B}$ bits uniformly take 0 and 1.

Through the experiment, we obtain the 5-round integral characteristic of $P_{128}(x)$ with 2^{16} chosen plaintexts (see Fig. 4.3). This characteristic is extended to the 6-round one as shown in Fig. 4.3, and it uses 2^{64} chosen plaintexts.

Similarly, we show the integral characteristic of $P_{256}(x)$. We first prepare chosen plaintexts where each column satisfies $\mathcal{A}$, and this characteristic is extended to the 7-round one with 2^{64} chosen plaintexts. When the first column (16 bits) satisfies $\mathcal{A}$, the 7th round output satisfies the following integral property.

```
0x30f0c00d3dc930cd090f0d0000d09000
0xd00c9fd390dc030d0f0000c0300090d0
0xd0930c0f0d000030c000d0d09003df9c
0x90d000c0f03009cd3dc0390d0d0f0000
```

Here, if the hexadecimal value in the ith row and jth column is `0x3`, the upper two bits satisfy $\mathcal{B}$ and the lower two bits do not satisfy $\mathcal{B}$. As another example, when the 17th column (16 bits) satisfies $\mathcal{A}$, the 7th round output satisfies the following integral property.

```
0x090f0d0000d0900030f0c00d3dc930cd
0x0f0000c0300090d0d00c9fd390dc030d
0xc000d0d09003df9cd0930c0f0d000030
0x3dc0390d0d0f000090d000c0f03009cd
```

In both integral characteristics, 348 bits of the 7th round output satisfy $\mathcal{B}$.

Detailed Attack Procedure. PRØST P_{128} has 6-round integral characteristics with 2^{64} chosen plaintexts. Moreover, any one bit is diffused to 64 bits in 2 rounds. Therefore, the time complexity of the FFT key recovery is $4 \times 64 \times 2^{64} = 2^{72}$ additions. The probability that incorrect keys are discarded in one FFT key recovery is 2^{-1}. All the bits of the output of the 6-round integral characteristic satisfy $\mathcal{B}$. Therefore, we repeat this analysis for 256 bits. The probability that all 256 bits satisfy $\mathcal{B}$ for incorrect key K is 2^{-256}. Thus, we can recover the correct key, and the total complexity is approximately $256 \times 2^{72} = 2^{80}$ additions.

PRØST P_{256} has 7-round integral characteristics with 2^{64} chosen plaintexts. Moreover, any one bit is diffused to 64 bits in 2 rounds. Therefore, the time complexity of the FFT key recovery is $4 \times 64 \times 2^{64} = 2^{72}$ additions. The probability that incorrect keys are discarded in one FFT key recovery is 2^{-1}. In each characteristic, only 348 bits of the 7th round output satisfy $\mathcal{B}$. Therefore, we use two 7-round characteristics, and then the sum of the number of bits that do not satisfy $\mathcal{B}$ in both characteristics is 24 bits. Since 488 bits of the output of the 7-round integral characteristic satisfy $\mathcal{B}$, we repeat the FFT key recovery 488 times. The time complexity is approximately $488 \times 2^{72} = 2^{80.9}$ additions. Since the probability that all 488 bits satisfy $\mathcal{B}$ for incorrect key K is 2^{-488}, and 2^{24} incorrect keys are expected to remain. We exhaustively search for the correct key. The time complexity is 2^{24}, and it is negligible compared to the time complexity for the two FFT key recoveries.

Further Optimization. The attack with the FFT key recovery can further be optimized. We look again the principle of the FFT key recovery, which is shown in Sect. 4.2.3. Since a vector $\hat{v}$ is calculated only from a Boolean function f, it does not depend on the multiset of ciphertexts. Therefore, we do not need to calculate $\hat{v}$ every time if we repeat the attack procedure for different chosen plaintext sets. Similarly, since a vector $\hat{w}$ is calculated only from the multiset of ciphertexts, it does not depend on a Boolean function f. Therefore, we do not need to calculate $\hat{w}$ every time if we use the same set of ciphertexts.

We now revisit the 8-round integral cryptanalysis against PRØST P_{128}. Any one bit is diffused to 64 bits in 2 rounds, and there are 16 bits that are diffused to the same 64 bits. Therefore, we only calculate $\hat{w}$ once per 16, and the time complexity becomes $(256/16) \times 64 \times 2^{64}$ additions. Moreover, it requires $256 \times 64 \times 2^{64}$ and $256 \times 2 \times 64 \times 2^{64}$ additions to get $\hat{v}$ and u, respectively. Therefore, the total time complexity is about

$$(16 + 256 + 256 \times 2) \times 64 \times 2^{64} \approx 2^{79.6}$$

additions.

4.4 FFT Key Recovery for Key-Alternating Cipher

4.4.1 Key-Alternating Cipher

The key-alternating cipher [BKL+12] is a common type of block cipher, and AES can be viewed as a 10-round key-alternating cipher [ABD+13]. Let P_i be an n-bit permutation, and the key-alternating cipher is expressed as

$$c = K_r \oplus P_r(\cdots \oplus P_3(K_2 \oplus P_2(K_1 \oplus P_1(K_0 \oplus p)))),$$

where $K_0, K_1, \ldots, K_r$ are round keys that are calculated from the master key. Let p and c be a plaintext and a ciphertext, respectively.

4.4.2 Attack Framework

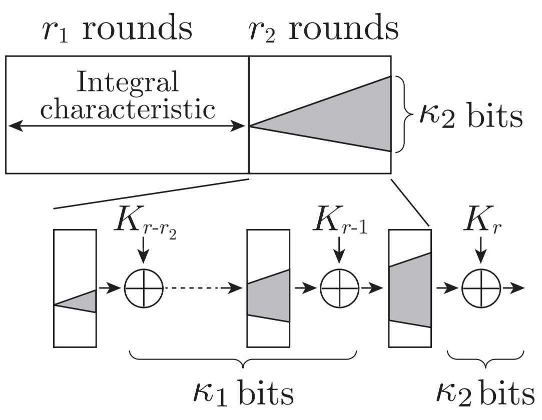


Figure 4.4: FFT key recovery for Key-alternating cipher

We first split r rounds into r_1 and r_2 rounds as $r = r_1 + r_2$ (see Fig. 4.4). We assume that a key-alternating cipher has an r_1-round integral characteristic with N chosen plaintexts. Moreover, we need to guess a κ-bit key that is required for this characteristic for the ciphertext side, and any one bit of output from the r_1 rounds is diffused to κ_2 bits by r_2 rounds. Let $F_{2,K'}$ be a function from κ_2 bits to one bit, where K' is κ_1-bit key calculated from $K_{r-r_2}, K_{r-r_2+1}, \ldots, K_{r-1}$. In this case, the key recovery can be expressed as

$$\bigoplus_{i=1}^{N} F_{2,K'}(c'_i \oplus K'_r) = 0,$$

where c'_i and K'_r are truncated to κ_2 bits from c_i and K_r, respectively. To apply the FFT key recovery, we first guess correct K', and then we have

$$\sum_{i=1}^{N} F_{2,K'}(c'_i \oplus K'_r) = 0 \bmod 2.$$

We can efficiently evaluate this equation using the FWHT. Thus, the time complexity is $4\kappa_2 2^{\kappa_1+\kappa_2}$ additions.

4.4.3 Integral Cryptanalysis against AES

Specification of AES. AES is standardized by the National Institute of Standards and Technology (NIST) in 2001 [U.S01]. The state of AES is expressed as a 4×4 matrix whose elements take an 8-bit value, i.e. the block length is 128 bits. AES consists of 10 rounds, and the round function is composed of four functions: `SubBytes`, `ShifrRows`, `MixColumns`, and `AddRoundKeys`. Each function is defined as follows:

- `SubBytes` substitutes each 8-bit value in the matrix into another 8-bit value by an S-box.
- `ShifrRows` rotates the position of 8-bit values within each row.
- `MixColumns` diffuses four 8-bit values within each column by a linear function.
- `AddRoundKeys` is an XOR operation where the round key is XORed to the matrix.

Detailed Attack Procedure. We show the FFT key recovery for the integral cryptanalysis against a 6-round AES (see Fig. 4.1). Since the FFT key recovery only calculates the summation of integers, we transform Eq. (4.1) to

$$\begin{aligned}&\sum_{n=1}^{2^{32}} S_5^{(i)}(S_1(c_n[1] \oplus K_1) \oplus S_2(c_n[2] \oplus K_2)\\ &\qquad \oplus S_3(c_n[3] \oplus K_3) \oplus S_4(c_n[4] \oplus K_4) \oplus K_5)\\ &= \sum_{n=1}^{2^{32}} f_{K_5}^{(i)}(F(c_n \oplus (K_1 \| K_2 \| K_3 \| K_4))),\end{aligned} \tag{4.4}$$

where F is a function from $\{0,1\}^{32}$ to $\{0,1\}^{32}$. Moreover, $f_{K_5}^{(i)}$ is a Boolean function whose output is the ith bit of the output of S_5. We first guess K_5, and then calculate this summation using the FWHT. The time complexity is $4 \times 32 \times 2^{32} = 2^{39}$ additions for every K_5, and the time complexity is $2^8 \times 2^{39} = 2^{47}$ additions because K_5 is 8 bits. The probability that incorrect keys are discarded in one FFT key recovery is 2^{-1}.

We estimate the time complexity to recover 5 keys $K_1, K_2, \ldots, K_5$ because Ferguson et al. estimated it in [FKL+00]. Since the output of S_5 is 8 bits, we repeat this analysis using the 8 bits. The probability that all 8 bits satisfy $\mathcal{B}$ for incorrect key is 2^{-8}. Since the total bit length of $K_1, K_2, \ldots, K_5$ is 40 bits, we repeat the above attack using 6 different sets. Thus, the total complexity is approximately the time of $6 \times 8 \times 2^{47} = 6 \times 2^{50}$ additions. When we use the partial-sum technique, the total time complexity is approximately the time of 6×2^{50} S-box lookups. We discuss the differences between them in Sect. 4.6.

Further Optimization. Similar to Sect. 4.3.3, we can further optimize our attack by removing the redundant calculation of $\hat{v}$ and $\hat{u}$. A vector $\hat{v}$ is calculated only from a Boolean function f, and the time complexity is $2^8 \times 8 \times 32 \times 2^{32} = 2^{48}$ additions. Next, we calculate $\hat{w}$ by using FFT, and the complexity of this step is $32 \times 2^{32} = 2^{37}$ additions. Finally, we calculate u from $\hat{v}$ and $\hat{w}$, and the time complexity is $2 \times 2^8 \times 8 \times 32 \times 2^{32} = 2^{49}$ additions. We execute our cryptanalysis with 6 sets, but they always use the same $\hat{v}$. Therefore, the total complexity is about $2^{48} + 6 \times (2^{49} + 2^{37}) \approx 2^{51.7}$ additions.

4.4.4 Integral Cryptanalysis against PRESENT

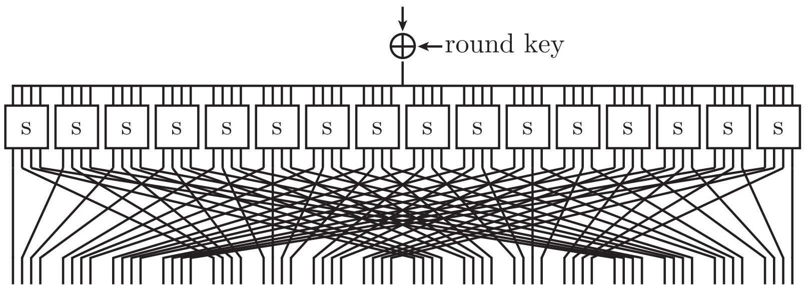


Figure 4.5: Round function of PRESENT

Specification of PRESENT. PRESENT [BKL+07] is a lightweight block cipher and it is adopted as an ISO/IEC standard [ISO12]. The block length is 64 bits, and the key size is chosen from 80 bits or 128 bits. PRESENT consists of 31 rounds, and Fig. 4.5 shows the round function of PRESENT. Round keys are first XORed and then sixteen 4-bit S-boxes are applied in the first of the round function. Finally, a bit-wise permutation is applied.

Detailed Attack Procedure. PRESENT has 7-round integral characteristics with 2^{16} chosen plaintexts [WW13], and any one bit is diffused to full 64 bits in 3 rounds. Moreover, 20-bit round key is required to calculate the inverse function. Therefore, the time complexity of the FFT key recovery is $2^{20} \times 4 \times 64 \times 2^{64} = 2^{92}$ additions. The probability that incorrect keys are discarded in one FFT key recovery is 2^{-1}.

We estimate the time complexity to recover $(20 + 64)$-bit keys. Since only one bit of the output of the 7-round integral characteristic satisfies $\mathcal{B}$, we repeat this analysis by using 84 chosen plaintext sets. The probability that it satisfies $\mathcal{B}$ for incorrect key in all 84 sets is 2^{-84}. Therefore, we can recover the correct 84-bit key, and the total complexity is approximately $84 \times 2^{92} \approx 2^{98.4}$ additions.

Further Optimization. Similar to Sect. 4.3.3, we can further optimize our attack by removing the redundant calculation of $\hat{v}$ and $\hat{u}$. A vector $\hat{v}$ is calculated only from a Boolean function f, and the time complexity is $2^{20} \times 64 \times 2^{64} = 2^{90}$ additions. Next, we calculate $\hat{w}$ by using FFT, and the complexity of this step is $64 \times 2^{64} = 2^{70}$ additions. Finally, we calculate u from $\hat{v}$ and $\hat{w}$, and the time complexity is $2 \times 2^{20} \times 64 \times 2^{64} = 2^{91}$ additions. We execute our cryptanalysis with 84 sets, but they always use the same $\hat{v}$. Therefore, the total complexity is about $2^{90} + 84 \times (2^{91} + 2^{70}) \approx 2^{97.4}$ additions.

4.5 FFT Key Recovery for Feistel Structure

4.5.1 Feistel Structure

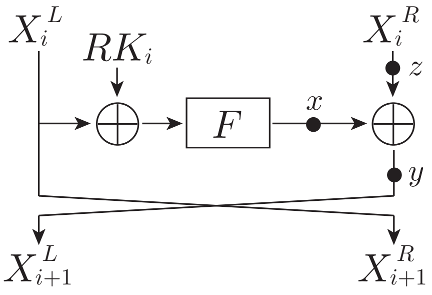


Figure 4.6: Round function of Feistel structure

The Feistel structure is also commonly used to construct a block cipher. In this chapter, we only focus on the Feistel structure whose round key is XORed before an F-function (see Fig. 4.6). Let (x_i^L, x_i^R) be the ith round output, and the $(i+1)$th round output is calculated as $(F(x_i^L \oplus RK_i) \oplus x_i^R, x_i^L)$, where RK_i denotes the ith round key.

4.5.2 Attack Framework

In 2012, Sasaki et al. proposed the MITM technique for the integral cryptanalysis [SW12b]. Generally, since the integral characteristics of the Feistel structure satisfy $\mathcal{B}$ in the right half, $\bigoplus z$ in Fig. 4.6 becomes 0. In the MITM technique, we evaluate $\bigoplus x$ and $\bigoplus y$ independently instead of $\bigoplus z$. Then, we search for keys satisfying $\bigoplus x = \bigoplus y$ through analysis such as the MITM attack [DH77]. In [SW12b], the partial-sum technique is used to evaluate $\bigoplus x$ and $\bigoplus y$, but the FFT can also be used to evaluate them.

We assume that we need to guess κ_1 and κ_2 bits to evaluate $\bigoplus x$ and $\bigoplus y$, respectively. If the round key is XORed with input from function F, the FFT key recovery can evaluate $\bigoplus x$ and $\bigoplus y$ with $4\kappa_1 2^{\kappa_1}$ and $4\kappa_2 2^{\kappa_2}$ additions, respectively. Moreover, the matching step of MITM analysis requires the time complexity of $O(\max\{2^{\kappa_1}, 2^{\kappa_2}\})$.

4.5.3 Integral Cryptanalysis against CLEFIA

Specification of CLEFIA. CLEFIA is a 128-bit block cipher, which was proposed by Shirai et al. in 2007 [SSA07]. It has a 4-branch type-2 generalized Feistel structure [ZMI89], and is adopted as an ISO/IEC standard. The round function is defined as Fig. 4.7, and the ith round output is calculated from the $(i-1)$th round output, RK_{2i-2} and RK_{2i-1}. Moreover, the

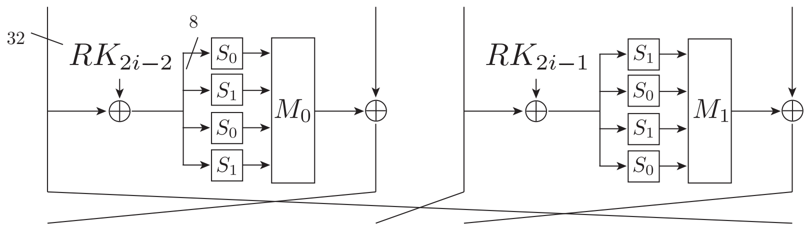


Figure 4.7: Round function of CLEFIA

whitening keys WK_0 and WK_1 are used in the first round, and WK_2 and WK_3 are used in the last round. For the 128-bit security version, the number of rounds is 18.

Detailed Attack Procedure. Li et al. showed that it has a 9-round integral characteristic with 2^{112} chosen plaintexts [LWZ11]. Then, Sasaki and Wang showed that the complexity of the integral cryptanalysis against a 12-round CLEFIA is 13×2^{106} S-box lookups using the MITM technique. We show the FFT key recovery against a 12-round CLEFIA. We first define some

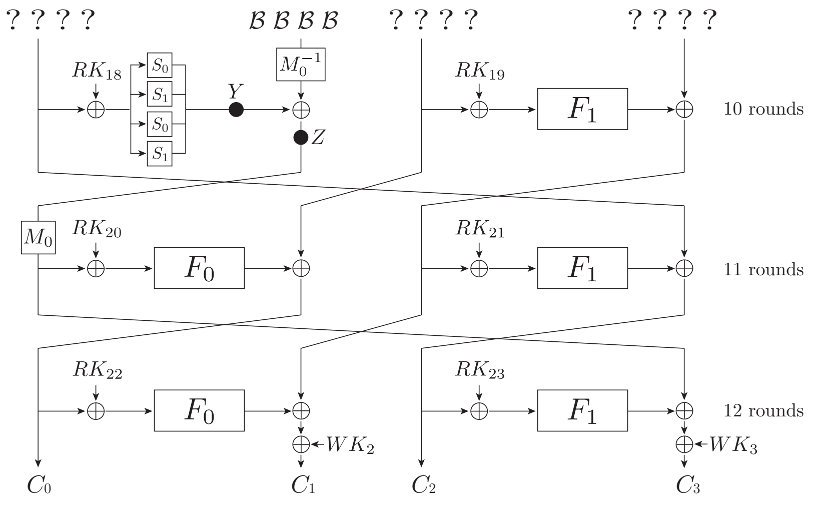


Figure 4.8: Key recovery of 12-round CLEFIA

notations. Let C_1, C_2, C_3, and C_4 be ciphertexts and each value is 32 bits (see Fig. 4.8). Let X be any 32-bit value, and $X[i]$ denotes the ith byte of X, namely $X = X[1]\|X[2]\|X[3]\|X[4]$. We define function $f_i : \{0,1\}^{32} \to \{0,1\}^8$ as

$$f_1(X)\|f_2(X)\|f_3(X)\|f_4(X) = F_1(X).$$

We first use the same method as the MITM technique [SW12b]. It uses a 9-round integral characteristic [LWZ11], where the second branch of the 9th round output satisfies $\mathcal{B}$. To optimize the MITM technique, we equivalently move the position of M_0 in the 10th round as shown in Fig. 4.8. As a result, we have $\bigoplus Y = \bigoplus Z$, where Y and Z are defined in Fig. 4.8. Therefore, if $\bigoplus Y$ and $\bigoplus Z$ can be calculated with the guessed round keys, we can recover the secret key from the MITM technique. Let $\bigoplus Y$ and $\bigoplus Z$ be calculated as

$$\begin{aligned}\bigoplus Y &= \bigoplus S(F_1(F_0(C_0 \oplus RK_{22}) \oplus C_1 \oplus RK'_{21}) \\ &\qquad \oplus C_2 \oplus RK_{18}), \\ \bigoplus Z &= \bigoplus M_0^{-1}(F_1(C_2 \oplus RK_{23}) \oplus C_3 \oplus WK_3),\end{aligned}$$

where $RK'_{21} = RK_{21} \oplus WK_2$, and S denotes the concatenation of 4 S-boxes S_0, S_1, S_0, and S_1. In [SW12b], each value is calculated using the partial-sum technique.

Hereinafter, we use the FFT key recovery instead of the partial-sum technique. We need to guess 96 bits to evaluate $\bigoplus Y$. Since WK_3 does not affect $\bigoplus Z$, we need to guess 32 bits to evaluate $\bigoplus Z$. Clearly, the time complexity to evaluate $\bigoplus Z$ is negligible compared to that to evaluate $\bigoplus Y$. Therefore, we show the complexity required to evaluate $\bigoplus Y$. For instance, the first byte of $\bigoplus Y$ is calculated as

$$\begin{aligned}\bigoplus Y[1] = \bigoplus S_0(&f_1(F_0(C_0 \oplus RK_{22}) \oplus C_1 \oplus RK'_{21}) \\ &\oplus C_2[1] \oplus RK_{18}[1]).\end{aligned}$$

To execute the FFT key recovery, we transform the above equation to

$$\begin{aligned}\sum Y[1]^{(i)} = \sum S_0^{(i)}(&f_1(F_0(C_0 \oplus RK_{22}) \oplus C_1 \oplus RK'_{21}) \\ &\oplus C_2[1] \oplus RK_{18}[1]),\end{aligned}$$

where $Y[1]^{(i)}$ denotes the ith bit of $Y[1]$ and the output of $S_0^{(i)}$ is the ith bit of the output of S_0. Moreover, this equation is transformed by defining function f as

$$\begin{aligned}&\sum Y[1]^{(i)} \\ &= \sum f\left((C_0\|C_1\|C_2[1]) \oplus (RK_{22}\|RK'_{21}\|RK_{18}[1])\right).\end{aligned}$$

We can evaluate this equation for all possible $(RK_{22}\|RK'_{21}\|RK_{18}[1])$ with $4 \times 72 \times 2^{72} \approx 2^{80.2}$ additions. Similarly, we evaluate $\sum Z[1]^{(i)}$ using the FFT key recovery, but the time complexity is negligible. Finally, we search for round keys satisfying $\sum Y[1]^{(i)} = \sum Z[1]^{(i)} \bmod 2$ using analysis such as the MITM attack. Since the complexity is approximately 2^{72}, it is also negligible.

Since the output of S_0 is 8 bits, we repeat this analysis for the eight bits. Moreover, we similarly calculate the second, third, and fourth bytes of $\bigoplus Y$ and $\bigoplus Z$. Therefore, the time complexity is approximately $4 \times 8 \times 2^{80.2} = 2^{85.2}$ additions. The probability that all 32 bits satisfy $\mathcal{B}$ for incorrect keys is expected to be 2^{-32}. Since the total bit length of RK_{18}, $RK_{21} \oplus WK_2$, RK_{22}, and RK_{23} is 128 bits, we repeat above analysis using 5 different chosen plaintext sets. Thus, the total complexity is approximately $5 \times 2^{85.2} = 2^{87.5}$ additions.

4.6 Discussion

We compare the FFT key recovery and the partial-sum technique. We first compare them based on their units of complexity. The complexity of the partial-sum technique is estimated from the number of S-box lookups. On the other hand, that of the FFT key recovery is estimated from the number of additions. Since the two processing speeds depend on the environment, we cannot directly compare them. However, we can roughly compare them. In the partial-sum technique, we need at least the time complexity of $O(2^{\kappa+\ell})$, where κ denotes the bit length of the guessing key. Moreover, let ℓ be the bit length of guessed key when we partially compute the sum, e.g., $\ell = 8$ for AES and $\ell = 32$ for CLEFIA. We expect that the FFT key recovery is superior to the partial-sum technique when ℓ is greater than 8. Second, we compare them based on memory access. The partial-sum technique randomly accesses memories. On the other hand, the FFT key recovery sequentially accesses memories. Generally, sequential access is more efficient than the random access.

We have an open problem regarding the FFT key recovery. Since round keys of block ciphers are calculated from the secret key, some bits of round keys are automatically recovered if some bits of the secret key are recovered. The partial-sum technique can utilize this property and efficiently reduce the complexity. For instance, the integral cryptanalysis against a 22-round LBlock utilizes this property [SW12a]. However, in the FFT key recovery, we do not yet know how to utilize this property.

4.7 Conclusion

We proposed a new technique for the integral cryptanalysis called the FFT key recovery. The FFT is commonly known technique, which reduces the time complexity from $O(N^2)$ to $O(N \log_2 N)$.

For the integral cryptanalysis, N is the number of guessed key bit, namely $N = 2^{\kappa}$ if we guess κ bits of round keys. The straightforward integral cryptanalysis requires the time complexity of $O(2^{2\kappa})$, but the FFT key recovery is roughly requires the time complexity of $O(\kappa 2^{\kappa})$. Moreover, the time complexity only depends on the bit length of keys that are required for an integral characteristic from the ciphertext side for most cases. Therefore, we can easily estimate the time complexity. We focused on three structures, i.e., the Even-Mansour scheme, key-alternating cipher, and Feistel structure, and showed attack strategies for these three structures. As applications of the three structures, we showed that an 8-round PRØST $\tilde{P}_{128,K}$, 6-round AES, 10-round PRESENT, and 12-round CLEFIA can be attacked with $2^{79.6}$, $2^{51.7}$, $2^{97.4}$, and $2^{87.5}$ additions, respectively.

Part II

Division Property: Efficient Method to Estimate Upper Bound of Algebraic Degree

Chapter 5

Division Property and Application to Structural Evaluation

Abstract– In this chapter, we show structural cryptanalyses against two popular networks, i.e., the Feistel Network and Substitute-Permutation Network (SPN). Our cryptanalyses are distinguishing attacks by an improved integral distinguisher. The integral distinguisher is one of the most powerful distinguishers against block ciphers, and it is usually constructed by evaluating the propagation of integral properties, e.g., ALL or BALANCE property. However, the integral property does not derive useful distinguishers against block ciphers with non-bijective functions and bit-oriented structures. Moreover, since the integral property does not clearly exploit the algebraic degree of block ciphers, it tends not to construct useful distinguishers against block ciphers with low-degree functions. In this chapter, we propose a new property called the *division property*, which is the generalization of the integral property. It can effectively construct the integral distinguisher even if the block cipher has non-bijective functions, bit-oriented structures, and low-degree functions. From viewpoints of the attackable number of rounds or chosen plaintexts, the division property can construct better distinguishers than previous methods. Although our attack is a generic attack, it can improve several integral distinguishers against specific cryptographic primitives. For instance, it can reduce the required number of chosen plaintexts for the 10-round distinguisher on KECCAK-f from 2^{1025} to 2^{515}. For the Feistel cipher, it theoretically proves that SIMON32, 48, 64, 96, and 128 have 9-, 11-, 11-, 13-, and 13-round integral distinguishers, respectively.
Keywords– Block cipher, Integral distinguisher, Division property, Feistel Network, Substitute-Permutation Network, KECCAK, SIMON, Boolean function

5.1 Introduction

The structural evaluation of cryptographic networks is an important topic of cryptology, and it helps us to design strong symmetric-key primitives. There are several structural evaluations against the Feistel Network and Substitute-Permutation Network (SPN) [BS01, IS13, Knu02, LR88, Pat04]. As one direction of the structural evaluation, there are the security evaluation by "the generic attack," which exploits only the feature of the network and does not exploit the particular weaknesses of a specific cipher. It is applicable to large classes of block ciphers, but it is not often effective than the dedicated attack against the specific cipher. This chapter focuses on generic attacks against both the Feistel Network and SPN. The existing generic attack shows that the Feistel Network whose F-functions are chosen from random functions or permutations is vulnerable up to 5 rounds [Pat04, Knu02]. Moreover, Biryukov and Shamir showed that the SPN is vulnerable up to 2.5 rounds [BS01].

Our Contribution. This chapter shows generic attacks against two networks by improving an integral distinguisher. The integral attack was first proposed by Daemen et al. to evaluate the security of SQUARE [DKR97], and then it was formalized by Knudsen and Wagner [KW02].

Table 5.1: The number of required chosen plaintexts to construct r-round integral distinguishers on the SIMONfamily, Serpent, and KECCAK-f

Target	$\log_2(\#\text{texts})$								Method	Reference
	6R	7R	8R	9R	10R	11R	12R	13R		
SIMON32	17	25	29	31	-	-	-	-	our	Sect. 5.4.3
	-	-	-	-	-	-	-	-	degree	[Knu94, BC13]
SIMON48	17	29	39	44	46	47	-	-	our	Sect. 5.4.3
	17	-	-	-	-	-	-	-	degree	[Knu94, BC13]
SIMON64	17	33	49	57	61	63	-	-	our	Sect. 5.4.3
	17	-	-	-	-	-	-	-	degree	[Knu94, BC13]
SIMON96	17	33	57	77	87	92	94	95	our	Sect. 5.4.3
	17	33	-	-	-	-	-	-	degree	[Knu94, BC13]
SIMON128	17	33	65	97	113	121	125	127	our	Sect. 5.4.3
	17	33	-	-	-	-	-	-	degree	[Knu94, BC13]
Target	$\log_2(\#\text{texts})$								Method	Reference
	3R	4R	5R	6R	7R	8R	9R	10R		
Serpent	12	28	84	113	124	-	-	-	our	Sect. 5.5.3
	28	82	113	123	127	-	-	-	degree	[BCC11]
Target	$\log_2(\#\text{texts})$								Method	Reference
	8R	9R	10R	11R	12R	13R	14R	15R		
KECCAK-f	130	258	515	1025	1410	1538	1580	1595	our	Sect. 5.5.3
	257	513	1025	1409	1537	1579	1593	1598	degree	[BCC11]

Nowadays, many integral distinguishers have been proposed against specific ciphers [KW02, LWZ11, WZ11, YPK02, ZRHD08], and they are often constructed by evaluating the propagation characteristic of integral properties, e.g., ALL property or BALANCE property. In this chapter, we revisit the integral property and then introduce the *division property* by generalizing the integral property. The division property can effectively construct integral distinguishers even if block ciphers have non-bijective functions, bit-oriented structures, and low-degree functions.

The Feistel Network is a generic construction to create a (2ℓ)-bit pseudo-random permutation from an ℓ-bit pseudo-random function. We call the ℓ-bit function the F-function and assume that an attacker cannot know the specification of the F-function. Our distinguishing attack can attack up to 3 rounds, and it can attack up to 5 rounds if the F-function is limited to a permutation. Unfortunately, they are not improved compared with the previous ones. However, assuming that the algebraic degree of the F-function is smaller than the bit length of the F-function, our attack can attack more rounds than the previous attacks exploiting the low-degree function. We summarize new integral distinguishers in Table 5.4. Although the assumption of our attack is only the algebraic degree of the F-function, it can construct new integral distinguishers on the SIMON family [BSS$^+$13]. Since SIMON has a non-bijective F-function and a bit-oriented structure, it is complicated task to construct the integral distinguisher. In [WLV$^+$14], Wang et al. showed that SIMON32 has the 15-round integral distinguisher by experiments. However, it is difficult to experimentally search for integral distinguishers on other SIMON families because the block lengths are large. The division property theoretically introduces that SIMON32, 48, 64, 96, and 128 have at least 9-, 11-, 11-, 13-, and 13-round integral distinguishers, respectively. Table 5.1 shows the comparison between our distinguishers and previous ones.

The SPN consists of an S-Layer and P-Layer, where the S-Layer has m ℓ-bit bijective S-boxes and the P-Layer has an (ℓm)-bit bijective linear function. The attacker cannot know the specifications of the S-boxes and linear function. Surprisingly, our generic attack is able to attack more rounds as the number of S-boxes is larger than the bit length of S-boxes. This fact implies that the design of the P-Layer that can diffuse more outputs of S-boxes may not derive prospective security improvements. We summarize new integral distinguishers in Table 5.7. Similar to the result against the Feistel Network, the division property is also useful to construct integral distinguishers against specific cryptographic primitives. For instance, we can reduce the required number of chosen plaintexts for the 7-round distinguisher on Serpent [ABK98] from 2^{127} to 2^{124}. Moreover, for the integral distinguisher on KECCAK-f [DBPA11], we can reduce the required number of chosen plaintexts compared with previous ones constructed by Boura et al. [BCC11]. Table 5.1 shows the comparison between our distinguishers and previous ones.

Organization. This chapter is organized as follows: In Sect. 5.2, we show notations, Boolean functions, and the framework of integral distinguishers. In Sect. 5.3, we propose the division property by generalizing the integral property and show propagation characteristics. In Sect. 5.4 and Sect. 5.5, we show new distinguishing attacks on the Feistel Network and SPN, respectively.

5.2 Preliminaries

5.2.1 Notation

We make the distinction between the addition over $\mathbb{F}_2^n$ and addition over $\mathbb{Z}$, and we use $\oplus$ and $+$ as the addition over $\mathbb{F}_2^n$ and addition over $\mathbb{Z}$, respectively. For any $a \in \mathbb{F}_2^n$, the ith element is expressed as $a[i]$, and the Hamming weight $w(a)$ is calculated as $w(a) = \sum_{i=1}^{n} a[i]$. Let $1^n \in \mathbb{F}_2^n$ be a value whose all elements are 1. Moreover, let $0^n \in \mathbb{F}_2^n$ be a value whose all elements are 0. For any set $\mathbb{K}$, let $|\mathbb{K}|$ be the number of elements. Moreover, let ϕ be an empty set. For any $\vec{a} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$, the vectorial Hamming weight is defined as $W(\vec{a}) = [w(a_1), w(a_2), \ldots, w(a_m)] \in \mathbb{Z}^m$, where a_i denotes the ith element of $\vec{a}$. Moreover, for any $\vec{k} \in \mathbb{Z}^m$ and $\vec{k}' \in \mathbb{Z}^m$, we define $\vec{k} \succeq \vec{k}'$ if $k_i \geq k_i'$ for all $i\,(1 \leq i \leq m)$. Otherwise, $\vec{k} \not\succeq \vec{k}'$.

Bit Product Functions. Let $\pi_u : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function for any $u \in \mathbb{F}_2^n$. Let $x \in \mathbb{F}_2^n$ be an input of π_u, and $\pi_u(x)$ is the AND of $x[i]$ satisfying $u[i] = 1$, namely, it is defined as

$$\pi_u(x) := \prod_{i=1}^{n} x[i]^{u[i]}.$$

Let $\pi_{\vec{u}} : (\mathbb{F}_2^n)^m \to \mathbb{F}_2$ be a function for any $\vec{u} \in (\mathbb{F}_2^n)^m$. Let $\vec{x} \in (\mathbb{F}_2^n)^m$ be an input of $\pi_{\vec{u}}$, namely, $\pi_{\vec{u}}(\vec{x})$ is calculated as

$$\pi_{\vec{u}}(\vec{x}) := \prod_{i=1}^{m} \pi_{u_i}(x_i).$$

5.2.2 Boolean Function

A Boolean function is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. Let $\deg(f)$ be the algebraic degree of a Boolean function f. As representations of the Boolean function, we use Algebraic Normal Form, which is defined as follows.

Algebraic Normal Form. Algebraic Normal Form (ANF) is a representation of a Boolean function. Any $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be represented as

$$f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^f \left(\prod_{i=1}^{n} x[i]^{u[i]} \right) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^f \pi_u(x),$$

where $a_u^f \in \mathbb{F}_2$ is a constant value depending on f and u. If $\deg(f)$ is at most d, all a_u^f satisfying $d < w(u)$ are 0. An n-bit S-box can be regarded as the concatenation of n Boolean functions. If algebraic degrees of n Boolean functions are at most d, we say the algebraic degree of the S-box is at most d.

5.2.3 Integral Distinguisher

An integral distinguisher was first proposed by Daemen et al. to evaluate the security of Square [DKR97], and then it was formalized by Knudsen and Wagner [KW02]. It uses a set of chosen plaintexts that contains all possible values for some bits and has a constant value for the other bits. Corresponding ciphertexts are calculated from plaintexts in the set by using an encryption oracle. If the XOR of the corresponding ciphertexts always becomes 0, we say that this cipher has the integral distinguisher.

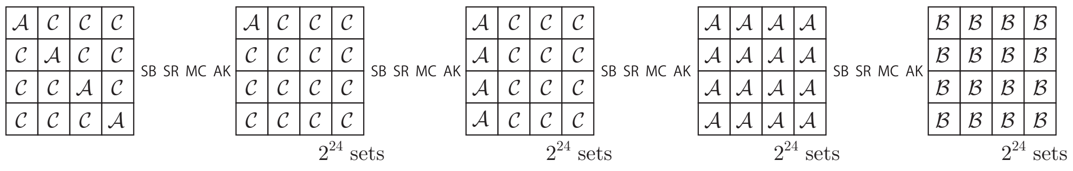


Figure 5.1: Integral distinguisher on 4-round AES

Integral Property. Nowadays, many integral distinguishers have been proposed against specific ciphers [KW02, LWZ11, WZ11, YPK02, ZRHD08], and they are often constructed by evaluating the propagation characteristic of the integral property. We define four integral properties as follows:

- ALL ($\mathcal{A}$) : Every value appears the same number in the multiset.
- BALANCE ($\mathcal{B}$) : The XOR of all texts in the multiset is 0.
- CONSTANT ($\mathcal{C}$) : The value is fixed to a constant for all texts in the multiset.
- UNKNOWN ($\mathcal{U}$) : The multiset is indistinguishable from one of n-bit random values.

Knudsen and Wagner showed that AES has the 4-round integral distinguisher with 2^{32} chosen plaintexts [KW02]. Figure 5.1 shows the integral distinguisher.

Unfortunately, the integral property does not derive effective distinguishers if block ciphers consist of non-bijective functions, e.g., DES [Nat77] and SIMON [BSS+13] consist of non-bijection functions. Moreover, since the propagation characteristic does not clearly exploit the algebraic degree of block ciphers, it tends not to construct effective distinguishers against block ciphers with low-degree round functions.

Degree Estimation. As another method to construct the integral distinguisher, there is a higher-order differential attack [Lai94, Knu94], which exploits the algebraic degree of block ciphers. When the algebraic degree of a block cipher is at most D, the cipher has the integral distinguisher with 2^{D+1} chosen plaintexts. Canteaut and Videau showed the bound of the degree of iterated round functions [CV02]. Then, Boura et al. improved the bound [BCC11], and showed integral distinguishers on KECCAK [DBPA11] and *Luffa* [CSW08].

If the degree of r iterated round functions is at most D, we can construct the r-round integral distinguisher with 2^{D+1} chosen plaintexts. In a classical method, if the degree of the round function is at most d, the degree of r iterated round functions is bounded by d^r. In 2011, Boura et al. showed tighter bound as follows.

Theorem 5.1 ([BCC11]). *Let S be a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^n$ corresponding to the concatenation of m smaller S-boxes, defined over $\mathbb{F}_2^{n_0}$. Let δ_k be the maximal degree of the product of any k bits of anyone of these S-boxes. Then, for any function G from $\mathbb{F}_2^n$ into $\mathbb{F}_2$, we have*

$$\deg(G \circ S) \leq n - \frac{n - \deg(G)}{\gamma},$$

where

$$\gamma = \max_{1 \leq i \leq n_0 - 1} \frac{n_0 - i}{n_0 - \delta_i}.$$

By using this bound, we can estimate the degree of (ℓ, d, m)-SPN. For instance, we show the degree of $(4, 3, 64)$-SPN as follows.

Number of rounds	1	2	3	4	5	6	7	8	9
Bound on degree	3	9	27	81	197	236	249	253	255

Therefore, we can construct the 8-round integral distinguisher on $(4, 3, 64)$-SPN with 2^{254} chosen plaintexts.

5.3 Division Property

5.3.1 Introduction of Division Property

The development of the division property is motivated from following two observations.

Observation 5.1. *Let us consider the integral characteristic of AES. The propagation of the integral property can find 4-round integral characteristics with 2^{32} chosen plaintext, while the algebraic degree of 4-round AES is upper-bounded by 126 even if Theorem 2.1 is used. Therefore, the propagation of the integral property is superior to the degree estimation.*

Observation 5.2. *Let us consider the integral characteristic of a modified AES, where S-boxes with degree 2 are used instead of the original S-box. Since the propagation of the integral property is not affected by the algebraic degree of S-boxes, the found characteristics is the same as the original AES. On the other hand, the algebraic degree of 7-round modified AES is upper-bounded by 96 when Theorem 2.1 is used. Therefore, the degree estimation is superior to the propagation of the integral property.*

Observations 5.1 and 5.2 implies that each of the two methods has its own advantages and drawbacks. Specifically, S-boxes are regarded as black boxes in the propagation of the integral property, i.e., it constructs integral characteristics by mainly exploiting the diffusion part of block ciphers. On the other hand, the degree estimation can exploit the algebraic degree of S-boxes but is difficult to well exploit the diffusion part, i.e., it constructs integral characteristics by mainly exploiting the confusion part of block ciphers. Therefore, it is natural that the method that can exploit both diffusion and confusion parts of block ciphers is desirable.

Redefinition of Integral Property. Let $\mathbb{X}$ be a multiset whose elements take an n-bit value. We first consider features of the multiset $\mathbb{X}$ satisfying $\mathcal{A}$. If we choose one bit from n bits and calculate the XOR of the chosen bit in the multiset, the calculated value is always 0. Moreover, if we choose at most $(n-1)$ bits from n bits and calculate the XOR of the AND of chosen bits in the multiset, the calculated value is also always 0. However, if we choose all bits from n bits and calculate the XOR of the AND of n bits in the multiset, the calculated value becomes unknown[1]. Above features are expressed by using the bit product function π_u, which is defined in Sect. 5.2.1, as follows. We evaluate the parity of $\pi_u(x)$ for all $x \in \mathbb{X}$, namely, evaluate $\bigoplus_{x\in\mathbb{X}} \pi_u(x)$. The parity is always even for any u satisfying $w(u) < n$. On the other hand, the parity becomes unknown for $u = 1^n$.

We next consider features of the multiset $\mathbb{X}$ satisfying $\mathcal{B}$. If we choose one bit from n bits and calculate the XOR of the chosen bit in the multiset, the calculated value is always 0. However, if we choose at least two bits from n bits and calculate the XOR of the AND of chosen bits in the multiset, the calculated value becomes unknown. Above features are expressed by using the bit product function π_u as follows. We evaluate the parity of $\pi_u(x)$ for all $x \in \mathbb{X}$. The parity is always even for any u satisfying $w(u) < 2$. On the other hand, the parity becomes unknown for any u satisfying $w(u) \geq 2$.

5.3.2 Definition of Division Property

Section 5.3.1 redefines both the ALL and BALANCE properties by the same notation. Since the redefinition can be parameterized by the number of product bits $w(u)$ of the bit product function π_u, we generalize the integral property as follows.

Definition 5.1 (Division Property)**.** *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$, and k takes a value between 0 and n. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_k^n$, it fulfils the following conditions: The parity of $\pi_u(x)$ for all $x \in \mathbb{X}$ is always even if $w(u) < k$. Moreover, the parity becomes unknown if $w(u) \geq k$.*

In the division property, the set of u is divided into the subset that $\bigoplus_{x\in\mathbb{X}} \pi_u(x)$ becomes unknown and the subset that $\bigoplus_{x\in\mathbb{X}} \pi_u(x)$ becomes 0.

[1] If all values appear the same even number in the multiset, the calculated value is always 0. If all values appear the same odd number in the multiset, the calculated value is always 1. Thus, we cannot guarantee whether the calculated value is 0 or not when we consider the multiset satisfying $\mathcal{A}$. In this case, we say the calculated value becomes unknown.

Example 5.1. *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^4$. As an example, we prepare the input multiset $\mathbb{X}$ as*

$$\mathbb{X} := \{\texttt{0x0}, \texttt{0x3}, \texttt{0x3}, \texttt{0x3}, \texttt{0x5}, \texttt{0x6}, \texttt{0x8}, \texttt{0xB}, \texttt{0xD}, \texttt{0xE}\}.$$

A following table calculates the summation of $\pi_u(x)$.

	0x0	0x3	0x3	0x3	0x5	0x6	0x8	0xB	0xD	0xE	$\sum \pi_u(x)$
	0000	0011	0011	0011	0101	0110	1000	1011	1101	1110	$(\bigoplus \pi_u(x))$
$u = 0000$	1	1	1	1	1	1	1	1	1	1	10 (0)
$u = 0001$	0	1	1	1	1	0	0	1	1	0	6 (0)
$u = 0010$	0	1	1	1	0	1	0	1	0	1	6 (0)
$u = 0011$	0	1	1	1	0	0	0	1	0	0	4 (0)
$u = 0100$	0	0	0	0	1	1	0	0	1	1	4 (0)
$u = 0101$	0	0	0	0	1	0	0	0	1	0	2 (0)
$u = 0110$	0	0	0	0	0	1	0	0	0	1	2 (0)
$u = 0111$	0	0	0	0	0	0	0	0	0	0	0 (0)
$u = 1000$	0	0	0	0	0	0	1	1	1	1	4 (0)
$u = 1001$	0	0	0	0	0	0	0	1	1	0	2 (0)
$u = 1010$	0	0	0	0	0	0	0	1	0	1	2 (0)
$u = 1011$	0	0	0	0	0	0	0	1	0	0	1 (1)
$u = 1100$	0	0	0	0	0	0	0	0	1	1	2 (0)
$u = 1101$	0	0	0	0	0	0	0	0	1	0	1 (1)
$u = 1110$	0	0	0	0	0	0	0	0	0	1	1 (1)
$u = 1111$	0	0	0	0	0	0	0	0	0	0	0 (0)

For all u satisfying $w(u) < 3$, $\bigoplus_{x \in \mathbb{X}} \pi_u(x)$ becomes 0. Therefore, the multiset has the division property $\mathcal{D}_3^4$.

Each definition of $\mathcal{B}$ and $\mathcal{U}$ is essentially the same as that of $\mathcal{D}_2^n$ and $\mathcal{D}_1^n$, respectively. However, the definition of $\mathcal{A}$ is different from that of $\mathcal{D}_n^n$. The multiset satisfying $\mathcal{A}$ always has the division property $\mathcal{D}_n^n$ but not vice versa. For instance, the multiset satisfying the EVEN property, which is defined that the number of occurrences is even for all values [SK12], does not always have $\mathcal{A}$, but it always has $\mathcal{D}_n^n$. In this chapter, we use only $\mathcal{D}_n^n$ instead of $\mathcal{A}$ because it is sufficient to use $\mathcal{D}_n^n$ from the viewpoint of the construction of integral distinguishers.

Propagation Characteristic of Division Property. Let s be an S-box whose degree is d. Let $\mathbb{X}$ be an input multiset whose elements take a value of $\mathbb{F}_2^n$. Let $S(\mathbb{X})$ be an output multiset whose elements are calculated from $s(x)$ for all $x \in \mathbb{X}$. We assume that $\mathbb{X}$ has $\mathcal{D}_k^n$ and want to evaluate the division property of $S(\mathbb{X})$. In the division property, the set of u is divided into the subset that $\bigoplus_{x \in \mathbb{X}} \pi_u(x)$ becomes unknown and the subset that $\bigoplus_{x \in \mathbb{X}} \pi_u(x)$ becomes 0. Therefore, we divide the set of v into the subset that $\bigoplus_{s(x) \in S(\mathbb{X})} \pi_v(s(x))$ becomes unknown and the subset that $\bigoplus_{s(x) \in S(\mathbb{X})} \pi_v(s(x))$ becomes 0. Since the parity of $\pi_v(s(x))$ for all $s(x) \in S(\mathbb{X})$ is equal to that of $(\pi_v \circ s)(x)$ for all $x \in \mathbb{X}$, we evaluate $\bigoplus_{x \in \mathbb{X}} (\pi_v \circ s)(x)$.

Proposition 5.1 (Propagation Characteristic of Division Property)**.** *Let s be an function (S-box) from n bits to n bits, and the degree is d. Assuming that an input multiset $\mathbb{X}$ has the division property $\mathcal{D}_k^n$, the output multiset $S(\mathbb{X})$ has $\mathcal{D}_{\lceil \frac{k}{d} \rceil}^n$. In addition, assuming that the S-box is a permutation, the output multiset $S(\mathbb{X})$ has $\mathcal{D}_n^n$ when the input multiset has $\mathcal{D}_n^n$.*

Proof. We represent $\bigoplus_{x \in \mathbb{X}} (\pi_v \circ s)(x)$ by using ANF as

$$\begin{aligned}
\bigoplus_{x \in \mathbb{X}} (\pi_v \circ s)(x) &= \bigoplus_{x \in \mathbb{X}} \left(\bigoplus_{u \in \mathbb{F}_2^n} a_u^{\pi_v \circ s} \pi_u(x) \right) \\
&= \bigoplus_{u \in \mathbb{F}_2^n | w(u) \geq k} a_u^{\pi_v \circ s} \left(\bigoplus_{x \in \mathbb{X}} \pi_u(x) \right) \oplus \bigoplus_{u \in \mathbb{F}_2^n | w(u) < k} a_u^{\pi_v \circ s} \left(\bigoplus_{x \in \mathbb{X}} \pi_u(x) \right).
\end{aligned}$$

Since the multiset $\mathbb{X}$ has $\mathcal{D}_k^n$, $\bigoplus_{x\in\mathbb{X}} \pi_u(x)$ is always 0 for any $\{u \in \mathbb{F}_2^n | w(u) < k\}$. Therefore, it satisfies

$$\bigoplus_{x\in\mathbb{X}} (\pi_v \circ s)(x) = \bigoplus_{u\in\mathbb{F}_2^n|w(u)\geq k} a_u^{\pi_v\circ s} \left(\bigoplus_{x\in\mathbb{X}} \pi_u(x)\right).$$

If $a_u^{\pi_v\circ s}$ is 0 for all $\{u \in \mathbb{F}_2^n | w(u) \geq k\}$, $\bigoplus_{x\in\mathbb{A}}(\pi_v \circ s)(x)$ always becomes 0. In other words, if there exists $\{u \in \mathbb{F}_2^n | w(u) \geq k\}$ such that $a_u^{\pi_v\circ s}$ is 1, $\bigoplus_{x\in\mathbb{A}}(\pi_v \circ s)(x)$ becomes unknown. Since the function π_v is the AND of $w(v)$ bits and the degree of S-box is d, the degree of the Boolean function $(\pi_v \circ s)$ has the following properties:

- The degree of $(\pi_v \circ s)$ is at most $\min\{n, w(v) \times d\}$.
- If the S-box is a permutation, the degree of $(\pi_v \circ s)$ is at most $n - 1$ for $w(v) < n$.

We first assume that the multiset $\mathbb{X}$ has $\mathcal{D}_k^n$. In this case, we consider only u satisfying $w(u) \geq k$. When $w(v) \times d < k$ holds, $a_u^{\pi_v\circ s}$ is always 0. Thus, the necessary condition that $a_u^{\pi_v\circ s}$ becomes 1 is $w(v) \times d \geq k$, and it is $w(v) \geq \lceil \frac{k}{d} \rceil$. Namely, the necessary condition that $\bigoplus_{x\in\mathbb{X}}(\pi_v \circ s)(x)$ becomes unknown is $w(v) \geq \lceil \frac{k}{d} \rceil$, and $S(\mathbb{X})$ has $\mathcal{D}_{\lceil \frac{k}{d} \rceil}^n$. We next assume that the multiset $\mathbb{X}$ has $\mathcal{D}_n^n$ and the S-box is a permutation. In this case, we consider only $u = 1^n$. When $w(v) < n$ holds, $a_{1^n}^{\pi_v\circ s}$ is always 0 because the degree of the Boolean function $(\pi_v \circ s)$ is at most $n - 1$. Thus, the necessary condition that $a_{1^n}^{\pi_v\circ s}$ becomes 1 is $v = 1^n$. Namely, the necessary condition that $\bigoplus_{x\in\mathbb{X}}(\pi_v \circ s)(x)$ becomes unknown is $v = 1^n$, and $S(\mathbb{X})$ has $\mathcal{D}_n^n$. □

Example 5.2. *Let us consider a following 4-bit S-box.*

x	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
$s(x)$	0x8	0xC	0x0	0xB	0x9	0xD	0xE	0x5	0xA	0x1	0x2	0x6	0x4	0xF	0x3	0x7

The S-box is bijective and the algebraic degree is 2. We now prepare the input multiset $\mathbb{X}$ as

$$\mathbb{X} := \{\texttt{0x0}, \texttt{0x3}, \texttt{0x3}, \texttt{0x3}, \texttt{0x5}, \texttt{0x6}, \texttt{0x8}, \texttt{0xB}, \texttt{0xD}, \texttt{0xE}\},$$

which is the same as Example 5.1 and the division property is $\mathcal{D}_3^4$. The output multiset is calculated as

$$S(\mathbb{X}) := \{\texttt{0x8}, \texttt{0xB}, \texttt{0xB}, \texttt{0xB}, \texttt{0xD}, \texttt{0xE}, \texttt{0xA}, \texttt{0x6}, \texttt{0xF}, \texttt{0x3}\},$$

and a following table calculates the summation of $(\pi_v \circ s)(x)$.

	0x8	0xB	0xB	0xB	0xD	0xE	0xA	0x6	0xF	0x3	$\sum(\pi_v \circ s)(x)$
	1000	1011	1011	1011	1101	1110	1010	0110	1111	0011	$(\bigoplus(\pi_v \circ s)(x))$
$v = 0000$	1	1	1	1	1	1	1	1	1	1	10 (0)
$v = 0001$	0	1	1	1	1	0	0	0	1	1	6 (0)
$v = 0010$	0	1	1	1	0	1	1	1	1	1	8 (0)
$v = 0011$	0	1	1	1	0	0	0	0	1	1	5 (1)
$v = 0100$	0	0	0	0	1	1	0	1	1	0	4 (0)
$v = 0101$	0	0	0	0	1	0	0	0	1	0	2 (0)
$v = 0110$	0	0	0	0	0	1	0	1	1	0	3 (1)
$v = 0111$	0	0	0	0	0	0	0	0	1	0	1 (1)
$v = 1000$	1	1	1	1	1	1	1	0	1	0	8 (0)
$v = 1001$	0	1	1	1	1	0	0	0	1	0	5 (1)
$v = 1010$	0	1	1	1	0	1	1	0	1	0	6 (0)
$v = 1011$	0	1	1	1	0	0	0	0	1	0	4 (0)
$v = 1100$	0	0	0	0	1	1	0	0	1	0	3 (1)
$v = 1101$	0	0	0	0	1	0	0	0	1	0	2 (0)
$v = 1110$	0	0	0	0	0	1	0	0	1	0	2 (0)
$v = 1111$	0	0	0	0	0	0	0	0	1	0	1 (1)

For all v satisfying $w(v) < 2$, $\bigoplus_{s(x)\in S(\mathbb{X})} \pi_v(s(x))$ becomes 0. Therefore, the multiset $S(\mathbb{X})$ has the division property $\mathcal{D}_2^4$.

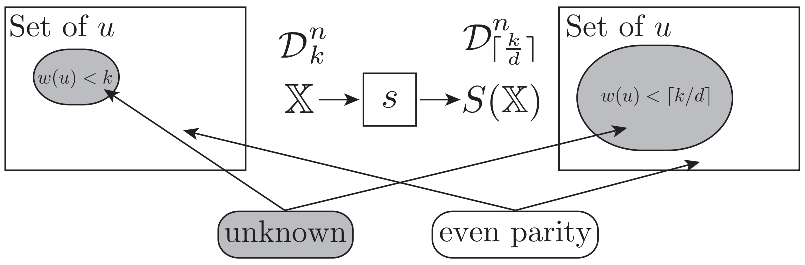


Figure 5.2: Propagation characteristic of division property

Figure 5.2 shows the outline of the propagation characteristic of the division property. Let $\mathbb{X}$ and $S(\mathbb{X})$ be input and output multisets, respectively. First, the size of the set of u that $\bigoplus_{x \in \mathbb{X}} \pi_u(x)$ becomes unknown is small. However, the size of the set of u that $\bigoplus_{x \in \mathbb{X}} \pi_u(s(x))$ becomes unknown expands. If the size expands to the universal set except for 0^n, we regard that the output multiset is indistinguishable from the multiset of random texts.

5.3.3 Vectorial Division Property

Section 5.3.2 only shows the division property for one S-box. However, since practical ciphers use several S-boxes in every round, we cannot construct integral distinguishers by only using Proposition 5.1. Therefore, we vectorize the division property.

Let an S-Layer be any function that consists of m n-bit S-boxes with degree d in parallel. We now consider the propagation characteristic of the division property against the S-Layer. Let $\mathbb{X}$ be the input multiset of the S-Layer, and $x \in \mathbb{X}$ takes a value of $(\mathbb{F}_2^n)^m$. The vectorization is the natural extension of the division property. Namely, the set of $\vec{u}$ is divided into the subset that $\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x})$ becomes unknown and the subset that $\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x})$ becomes 0, where $\vec{u}$ is an m-dimensional vector whose elements take a value of $\mathbb{F}_2^n$. Figure 5.3 shows the difference between the division property and vectorial one.

Definition 5.2 (Vectorial Division Property)**.** *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. Let $\vec{k}$ be an m-dimensional vector whose ith element takes a value between 0 and n_i. When the multiset $\mathbb{X}$ has the (vectorial) division property $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\vec{k}}$, it fulfils the following conditions:*

$$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} \text{unknown} & \text{if } W(\vec{u}) \succeq \vec{k}, \\ 0 & \text{otherwise.} \end{cases}$$

For the simplicity, the division property for $(\mathbb{F}_2^n)^m$ is referred to as $\mathcal{D}^{n^m}_{\mathbb{K}}$.

Propagation Characteristic of Vectorial Division Property. Assume that the input multiset of the S-Layer has the division property $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\vec{k}}$. The output of the S-Layer is calculated as $S(\vec{x}) = (s_1(x_1), s_2(x_2), \ldots, s_m(x_m))$ for $(x_1, x_2, \ldots, x_m) \in \mathbb{X}$. We now consider the set of $\vec{v}$ that $\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{v}}(S(\vec{x}))$ becomes unknown and the set of $\vec{v}$ that $\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{v}}(S(\vec{x}))$ becomes 0. Since the output of each S-box is calculated independently, the propagation characteristic of the division property can also be evaluated independently. Namely, the output multiset has $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\vec{k}'}$, where $k'_i = \lceil k_i/d \rceil$ holds. Moreover, if ith S-box is bijective and $k_i = n$ holds, $k'_i = n$ holds.

5.3.4 Collective Division Property

By vectorizing of the division property, we can evaluate the multiset whose elements take a value of $(\mathbb{F}_2^n)^m$. However, it is still insufficient to use only vectorial division property. For simplicity, we consider a multiset $\mathbb{X}$ whose elements take a value of $(\mathbb{F}_2^8)^2$. Assume that the number of elements in $\mathbb{X}$ is 256, and two elements of $\vec{x}$ take all values from 0 to 255 independently. We consider the set of $\vec{u}$ that the parity of $\pi_{\vec{u}}(\vec{x})$ for all $\vec{x} \in \mathbb{X}$ becomes unknown and the set of $\vec{u}$ that the parity becomes 0.

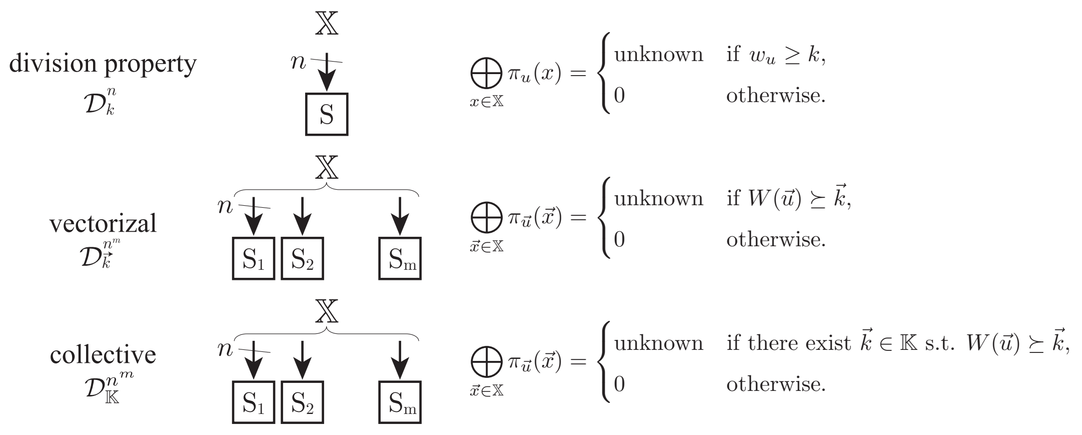

division property
$\mathcal{D}_k^n$
$\mathbb{X}$
n
S
$\bigoplus_{x \in \mathbb{X}} \pi_u(x) = \begin{cases} \text{unknown} & \text{if } w_u \geq k, \\ 0 & \text{otherwise.} \end{cases}$
vectorizal
$\mathcal{D}_{\vec{k}}^{n^m}$
S1
S2
Sm
$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} \text{unknown} & \text{if } W(\vec{u}) \succeq \vec{k}, \\ 0 & \text{otherwise.} \end{cases}$
collective
$\mathcal{D}_{\mathbb{K}}^{n^m}$
$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} \text{unknown} & \text{if there exist } \vec{k} \in \mathbb{K} \text{ s.t. } W(\vec{u}) \succeq \vec{k}, \\ 0 & \text{otherwise.} \end{cases}$

Figure 5.3: Division property, vectorial division property, and collective division property

- The parity becomes unknown if $\vec{u} \succeq [8, 0]$.
- The parity becomes unknown if $\vec{u} \succeq [0, 8]$.
- The parity becomes unknown if $\vec{u} \succeq [1, 1]$.
- Otherwise, i.e., $(\vec{u} \not\succeq [8, 0]) \wedge (\vec{u} \not\succeq [0, 8]) \wedge (\vec{u} \not\succeq [1, 1])$, the parity is always even.

We cannot express this property by using the vectorial division property. Therefore, we collect several vectorial division properties. Figure 5.3 shows the difference between the vectorial division property and collective one.

Definition 5.3 (Collective Division Property)**.** *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. Let $\mathbb{K}$ be a set whose elements take an m-dimensional vector whose ith element takes a value between 0 and n_i. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_1, n_2, \ldots, n_m}$, it fulfils the following conditions:*

$$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} \text{unknown} & \text{if there exist } \vec{k} \in \mathbb{K} \text{ s.t. } W(\vec{u}) \succeq \vec{k}, \\ 0 & \text{otherwise.} \end{cases}$$

It is obvious that the collective division property with $|\mathbb{K}| = 1$ is the same as the vectorial division property.

Example 5.3. *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2^8 \times \mathbb{F}_2^8)$. Assume that the multiset $\mathbb{X}$ has the division property $\mathcal{D}^{8^2}_{\{[1,5],[3,3],[4,5],[5,1],[6,0]\}}$. Then, if $[u_1, u_2]$ is chosen from the gray part in Fig 5.4, $\bigoplus_{[x_1,x_2] \in \mathbb{X}} \pi_{[u_1,u_2]}([x_1, x_2])$ becomes unknown. For example, when $\vec{u} = [\texttt{0x3F}, \texttt{0xFC}]$ is used, we cannot determine $\bigoplus_{[x_1,x_2] \in \mathbb{X}} \pi_{[\texttt{0x3F},\texttt{0xFC}]}([x_1, x_2])$ because $W(\vec{u}) = [6, 6]$. On the other hand, if (u_1, u_2) is chosen from the white part in Fig 5.4, $\bigoplus_{[x_1,x_2] \in \mathbb{X}} \pi_{[u_1,u_2]}([x_1, x_2])$ is 0. Note that the division property $\mathcal{D}^{8^2}_{\{[1,5],[3,3],[5,1],[6,0]\}}$ is the same as $\mathcal{D}^{8^2}_{\{[1,5],[3,3],[4,5],[5,1],[6,0]\}}$ because the unknown space is invariant.*

A similar example is shown in [SHZ+15] and may help to further understand the division property.

Propagation Characteristic for Collective Division Property.

Proposition 5.2 (Propagation Characteristic of Collective Division Property)**.** *Let S be a function that consists of m S-boxes, where the bit length and the algebraic degree of the ith S-box is n_i bits and d_i, respectively. The input and the output take a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$, and $\mathbb{X}$ and $S(\mathbb{X})$ denote the input multiset and the output multiset, respectively. Assuming that*

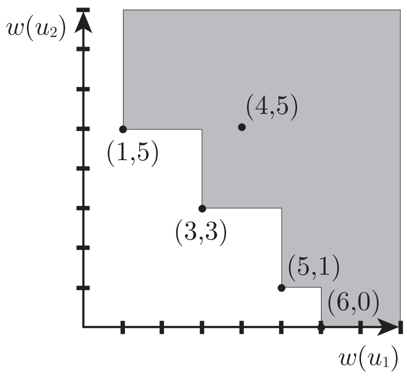


Figure 5.4: Division Property $\mathcal{D}^{8^2}_{\{[1,5],[3,3],[5,1],[6,0]\}}$

the multiset $\mathbb{X}$ has the division property $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\mathbb{K}}$, the multiset $S(\mathbb{X})$ has the division property $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\mathbb{K}'}$, where $\mathbb{K}'$ is calculated as follows: First, $\mathbb{K}'$ is initialized to ϕ. Then, for all $\vec{k} \in \mathbb{K}$,

$$\mathbb{K}' = \mathbb{K}' \cup \left[\left\lceil \frac{k_1}{d_1} \right\rceil, \left\lceil \frac{k_2}{d_2} \right\rceil, \ldots, \left\lceil \frac{k_m}{d_m} \right\rceil \right],$$

is calculated. Here, when the ith S-box is bijective and $k_i = n_i$, the ith element of the propagated property becomes n_i not $\lceil n_i/d_i \rceil$.

Proof. Let S be a function that consists of m S-boxes, where S_i denotes the ith S-box and the bit length and the algebraic degree is n_i bits and d_i, respectively. The input and the output take a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$, and $\mathbb{X}$ and $S(\mathbb{X})$ denote the input multiset and the output multiset, respectively.

First, we only apply the first S-box and evaluate the division property of the multiset whose elements are represented by $[S_1(x_1), x_2, \ldots, x_m]$. Assuming that the multiset $\mathbb{X}$ has the division property $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\mathbb{K}}$, the parity $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{v}}([S_1(x_1), x_2, \ldots, x_m])$ is evaluated as follows:

$$\begin{aligned}
\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{v}}([S_1(x_1), x_2, \ldots, x_m]) &= \bigoplus_{\vec{x}\in\mathbb{X}} \left((\pi_{v_1} \circ S_1)(x_1) \times \prod_{i=2}^{m} \pi_{v_i}(x_i) \right) \\
&= \bigoplus_{\vec{x}\in\mathbb{X}} \left(\left(\bigoplus_{u_1 \in \mathbb{F}_2^{n_1}} a_{u_1}^{(\pi_{v_1}\circ S_1)} \pi_{u_1}(x_1) \right) \times \left(\prod_{i=2}^{m} \pi_{v_i}(x_i) \right) \right) \\
&= \bigoplus_{u_1 \in \mathbb{F}_2^{n_1}} \left(\bigoplus_{\vec{x}\in\mathbb{X}} \left(a_{u_1}^{(\pi_{v_1}\circ S_1)} \pi_{u_1}(x_1) \times \prod_{i=2}^{m} \pi_{v_i}(x_i) \right) \right) \\
&= \bigoplus_{u_1 \in \mathbb{F}_2^{n_1}} \left(a_{u_1}^{(\pi_{v_1}\circ S_1)} \bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[u_1, v_2, v_3, \ldots, v_m]}(\vec{x}) \right).
\end{aligned}$$

Therefore, for any $\vec{v} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$, the parity $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{v}}([S_1(x_1), x_2, \ldots, x_m])$ is 0 if

$$a_{u_1}^{(\pi_{v_1}\circ S_1)} \bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[u_1, v_2, v_3, \ldots, v_m]}(\vec{x})$$

is 0 for all $u_1 \in \mathbb{F}_2^{n_1}$. Since the algebraic degree of $(\pi_{v_1} \circ S_1)$ is at most $w(v_1) \times d_1$, $a_{u_1}^{(\pi_{v_1}\circ S_1)} = 0$ when $w(u_1) > w(v_1) \times d_1$. Therefore, the parity becomes unknown only if we cannot determine the value of $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[u_1, v_2, v_3, \ldots, v_m]}(\vec{x})$ when $w(u_1) \leq w(v_1) \times d_1$. Now, since the multiset $\mathbb{X}$ has the division property $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\mathbb{K}}$,

$$\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} \text{unknown} & \text{if there exist } \vec{k} \in \mathbb{K} \text{ s.t. } W(\vec{u}) \succeq \vec{k}, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, the necessary condition that $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[u_1,v_2,v_3,\ldots,v_m]}(\vec{x})$ becomes unknown is expressed as follows:

$$\begin{aligned}
&W([u_1, v_2, v_3, \ldots, v_m]) \succeq \vec{k}, \\
\Rightarrow &[w(v_1) \times d_1, w(v_2), \ldots, w(v_m)] \succeq \vec{k}, \\
\Rightarrow &[w(v_1), w(v_2), \ldots, w(v_m)] \succeq \left[\left\lceil \frac{k_1}{d_1} \right\rceil, k_2, k_3, \ldots, k_m \right].
\end{aligned}$$

Namely, $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{v}}([S_1(x_1), x_2, \ldots, x_m])$ is unknown only if there exists $\vec{k} \in \mathbb{K}$ satisfying

$$W(v_1, v_2, v_3, \ldots, v_m) \succeq \left[\left\lceil \frac{k_1}{d_1} \right\rceil, k_2, k_3, \ldots, k_m \right].$$

Therefore, the division property of the output multiset is $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\mathbb{K}'}$, where $\mathbb{K}'$ has the following vectors

$$\left[\left\lceil \frac{k_1}{d_1} \right\rceil, k_2, \ldots, k_m \right] \text{ for all } \vec{k} \in \mathbb{K}.$$

Next, assume that S_1 is bijective and $k_1 = n_1$. Then, the algebraic degree of $(\pi_{v_1} \circ S_1)$ is less than n_1 for $w(v_1) < n_1$ and becomes n_1 for only $w(v_1) = n_1$. Therefore, the necessary condition that $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[u_1,v_2,v_3,\ldots,v_m]}(\vec{x})$ becomes unknown is $w(v_1) = n_1$. Namely, if $k_1 = n_1$, $[n_1, k_2, k_3, \ldots, k_m]$ is inserted into $\mathbb{K}'$ instead of $[\lceil k_1/d_1 \rceil, k_2, \ldots, k_m]$. Finally, Proposition 5.2 is proven by repeating the same procedure for other S-boxes. □

5.3.5 Propagation Rules for Simple Operation

We summarize some propagation rules for simple operation as follows.

Proposition 5.3 (COPY)**.** *Let F be a copy function, where the input x takes a value of $\mathbb{F}_2^n$ and the output is calculated as $[y_1, y_2] = [x, x]$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and the output multiset, respectively. Assuming that the multiset $\mathbb{X}$ has the division property $\mathcal{D}^n_k$, the multiset $F(\mathbb{X})$ has the division property $\mathcal{D}^{n,n}_{\mathbb{K}'}$, where $\mathbb{K}'$ is calculated as follows: First, $\mathbb{K}'$ is initialized to ϕ. Then, for all i $(0 \le i \le k)$,*

$$\mathbb{K}' = \mathbb{K}' \cup [k-i, i],$$

is calculated.

Proof. Let F be a copy function, where the input x takes a value of $\mathbb{F}_2^n$ and the output is calculated as $[y_1, y_2] = [x, x]$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and the output multiset, respectively.

Assuming that the multiset $\mathbb{X}$ has the division property $\mathcal{D}^n_k$, the parity $\bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ is evaluated as follows:

$$\bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y}) = \bigoplus_{x\in\mathbb{X}} \pi_{[v_1,v_2]}([x,x]) = \bigoplus_{x\in\mathbb{X}} (\pi_{v_1}(x) \times \pi_{v_2}(x)) = \bigoplus_{x\in\mathbb{X}} (\pi_{v_1 \vee v_2}(x)).$$

Since the multiset $\mathbb{X}$ has the division property $\mathcal{D}^n_k$,

$$\bigoplus_{x\in\mathbb{X}} \pi_u(x) = \begin{cases} \text{unknown} & w(u) \ge k, \\ 0 & w(u) < k. \end{cases}$$

When $w(v_1) + w(v_2) < k$, the parity $\bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ is 0 because $w(v_1 \vee v_2) \le w(v_1) + w(v_2) < k$. Moreover, the necessary condition that the parity becomes unknown is $w(v_1) + w(v_2) \ge k$. Therefore, the division property of $F(\mathbb{X})$ is $\mathcal{D}^{n,n}_{\mathbb{K}'}$, where $\mathbb{K}'$ has the following vectors

$$[k-i, i] \text{ for } 0 \le i \le k.$$

□

Proposition 5.4 (XOR)**.** *Let F be a function compressed by an XOR, where the input $[x_1, x_2]$ takes a value of $(\mathbb{F}_2^n \times \mathbb{F}_2^n)$ and the output is calculated as $y = x_1 \oplus x_2$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and the output multiset, respectively. Assuming that the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n,n}$, the division property of the multiset $F(\mathbb{X})$ is $\mathcal{D}_{k'}^{n}$ as*

$$k' = \min_{[k_1,k_2]\in\mathbb{K}} \{k_1 + k_2\}.$$

Here, if the minimum value of k' is larger than n, the propagation characteristic of the division property is aborted. Namely, a value of $\oplus_{y\in F(\mathbb{X})}\pi_v(y)$ is 0 for all $v \in \mathbb{F}_2^n$.

Proof. Let F be a compression function by an XOR, where the input $[x_1, x_2]$ takes a value of $(\mathbb{F}_2^n \times \mathbb{F}_2^n)$ and the output is calculated as $y = x_1 \oplus x_2$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and the output multiset, respectively.

Assuming that the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n,n}$, the parity $\bigoplus_{y\in F(\mathbb{X})} \pi_v(y)$ is evaluated as follows:

$$\begin{aligned}
\bigoplus_{y\in F(\mathbb{X})} \pi_v(y) = \bigoplus_{[x_1,x_2]\in\mathbb{X}} \pi_v(x_1 \oplus x_2) &= \bigoplus_{[x_1,x_2]\in\mathbb{X}} \left(\prod_{i=1}^{n} (x_1[i] \oplus x_2[i])^{v[i]} \right) \\
&= \bigoplus_{[x_1,x_2]\in\mathbb{X}} \left(\bigoplus_{\vec{w}\in\{1,2\}^n} \left(\prod_{i=1}^{n} x_{w_i}[i]^{v[i]} \right) \right) \\
&= \bigoplus_{\vec{w}\in\{1,2\}^n} \left(\bigoplus_{[x_1,x_2]\in\mathbb{X}} \left(\prod_{i=1}^{n} x_{w_i}[i]^{v[i]} \right) \right) \\
&= \bigoplus_{\vec{w}\in\{1,2\}^n} \left(\bigoplus_{[x_1,x_2]\in\mathbb{X}} \left(\pi_{\delta_1(v,\vec{w})}(x_1) \times \pi_{\delta_2(v,\vec{w})}(x_2) \right) \right),
\end{aligned}$$

where

$$\delta_j(v, \vec{w})[i] = \begin{cases} 1 & v[i] = 1 \text{ and } w_i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Since the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n,n}$,

$$\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[u_1,u_2]}(\vec{x}) = \begin{cases} \text{unknown} & \text{if there exist } [k_1, k_2] \in \mathbb{K} \text{ s.t. } [w(u_1), w(u_2)] \succeq [k_1, k_2], \\ 0 & \text{otherwise.} \end{cases}$$

When $w(v) = w(\delta_1(v, \vec{w})) + w(\delta_2(v, \vec{w})) < \min_{\vec{k}\in\mathbb{K}}\{k_1 + k_2\}$, the parity $\bigoplus_{y\in F(\mathbb{X})} \pi_v(y)$ is 0 because there is not $[k_1, k_2] \in \mathbb{K}$ satisfying $[w(\delta_1(v, \vec{w})), w(\delta_2(v, \vec{w}))] \succeq [k_1, k_2]$. Moreover, the necessary condition that the parity becomes unknown is $w(v) \geq \min_{\vec{k}\in\mathbb{K}}\{k_1 + k_2\}$. Therefore, the division property of $F(\mathbb{X})$ is $\mathcal{D}_{k'}^{n}$, where $k' = \min_{\vec{k}\in\mathbb{K}}\{k_1 + k_2\}$. Note that the parity is 0 for all v if k' is greater than n. □

Proposition 5.5 (SPLIT)**.** *Let F be a split function, where the input x takes a value of $\mathbb{F}_2^n$ and the output is calculated as $y_1\|y_2 = x$, where $[y_1, y_2]$ takes a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n-n_1})$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and the output multiset, respectively. Assuming that the multiset $\mathbb{X}$ has the division property $\mathcal{D}_k^n$, the multiset $F(\mathbb{X})$ has the division property $\mathcal{D}_{\mathbb{K}'}^{n_1,n-n_1}$, where $\mathbb{K}'$ is calculated as follows: First, $\mathbb{K}'$ is initialized to ϕ. Then, for all i $(0 \leq i \leq k)$,*

$$\mathbb{K}' = \mathbb{K}' \cup [k - i, i],$$

is calculated. Here, $(k - i)$ is less than or equal to n_1, and i is less than or equal to $n - n_1$.

Proof. Let F be a split function, where the input x takes a value of $\mathbb{F}_2^n$ and the output is calculated as $y_1\|y_2 = x$, where $[y_1, y_2]$ takes a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n-n_1})$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and the output multiset, respectively.

Assuming that the multiset $\mathbb{X}$ has the division property $\mathcal{D}_k^n$, the parity $\bigoplus_{\vec{y} \in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ is evaluated as follows:

$$\bigoplus_{\vec{y} \in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y}) = \bigoplus_{x \in \mathbb{X}} \pi_{[v_1 \| v_2]}(x).$$

Since the multiset $\mathbb{X}$ has the division property $\mathcal{D}_k^n$,

$$\bigoplus_{x \in \mathbb{X}} \pi_u(x) = \begin{cases} \text{unknown} & w(u) \geq k, \\ 0 & w(u) < k. \end{cases}$$

When $w(v_1) + w(v_2) < k$, the parity $\bigoplus_{\vec{y} \in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ is 0 because $w(v_1 \| v_2) = w(v_1) + w(v_2) < k$. Moreover, the necessary condition that the parity becomes unknown is $w(v_1) + w(v_2) \geq k$. Therefore, the division property of $F(\mathbb{X})$ is $\mathcal{D}_{\mathbb{K}'}^{n_1, n-n_1}$, where $\mathbb{K}'$ has the following vectors

$$[k - i, i] \text{ for } 0 \leq i \leq k.$$

Note that we cannot choose more than n_1 and $n - n_1$ bits from y_1 and y_2, respectively. □

Proposition 5.6 (CONCATENATION). *Let F be a concatenation function, where the input $[x_1, x_2]$ takes a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2})$ and the output is calculated as $y = x_1 \| x_2$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and the output multiset, respectively. Assuming that the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_1, n_2}$, the division property of the multiset $F(\mathbb{X})$ is $\mathcal{D}_{k'}^{n_1+n_2}$ as*

$$k' = \min_{[k_1, k_2] \in \mathbb{K}} \{k_1 + k_2\}.$$

Proof. Let F be a concatenation function, where the input $[x_1, x_2]$ takes a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2})$ and the output is calculated as $y = x_1 \| x_2$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and the output multiset, respectively.

Assuming that the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_1, n_2}$, the parity $\bigoplus_{y \in F(\mathbb{X})} \pi_v(y)$ is evaluated as follows:

$$\bigoplus_{y \in F(\mathbb{X})} \pi_v(y) = \bigoplus_{[x_1, x_2] \in \mathbb{X}} \pi_{v_1 \| v_2}(x_1 \| x_2) = \bigoplus_{[x_1, x_2] \in \mathbb{X}} \pi_{[v_1, v_2]}([x_1, x_2]),$$

where $v = v_1 \| v_2$, and the bit length of v_1 and that of v_2 is n_1 and n_2, respectively. Since the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_1, n_2}$,

$$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{[u_1, u_2]}(\vec{x}) = \begin{cases} \text{unknown} & \text{if there exist } [k_1, k_2] \in \mathbb{K} \text{ s.t. } [w(u_1), w(u_2)] \succeq [k_1, k_2], \\ 0 & \text{otherwise.} \end{cases}$$

When $w(v) = w(v_1) + w(v_2) < \min_{\vec{k} \in \mathbb{K}}\{k_1 + k_2\}$, the parity $\bigoplus_{y \in F(\mathbb{X})} \pi_v(y)$ is 0 because there is not $[k_1, k_2] \in \mathbb{K}$ satisfying $[w(v_1), w(v_2)] \succeq [k_1, k_2]$. Moreover, the necessary condition that the parity becomes unknown is $w(v) \geq \min_{\vec{k} \in \mathbb{K}}\{k_1 + k_2\}$. Therefore, the division property of $F(\mathbb{X})$ is $\mathcal{D}_{k'}^n$, where $k' = \min_{\vec{k} \in \mathbb{K}}\{k_1 + k_2\}$. □

5.4 Improved Distinguishers on Feistel Network

5.4.1 Feistel Network

(ℓ, d)-Feistel. The Feistel Network is one of the most popular network to design block ciphers. When n-bit block ciphers are constructed by the Feistel Network, the input of the round function is expressed in two $(n/2)$-bit values. Moreover, an $(n/2)$-bit non-linear function F is used in the round function, and we call this function the F-function. Let (w_1, w_2) be the input of the round function, and the output is calculated as $(z_1, z_2) = (F(w_1) \oplus w_2, w_1)$. We now define an (ℓ, d)-Feistel, whose F-function is an ℓ-bit non-linear function with degree d (this function is not limited to a permutation). Figure 5.5 shows the round function of the Feistel Network. There are many block ciphers adopting (ℓ, d)-Feistel, e.g. DES [Nat77], Camellia [AIK+00], and SIMON2n [BSS+13] adopt $(32, 5)$-, $(64, 7)$-, and $(n, 2)$-Feistel, respectively.

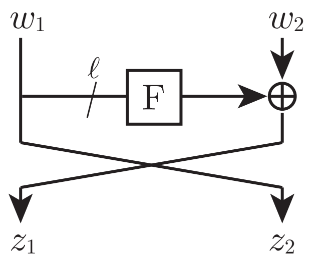


Figure 5.5: (ℓ, d)-Feistel

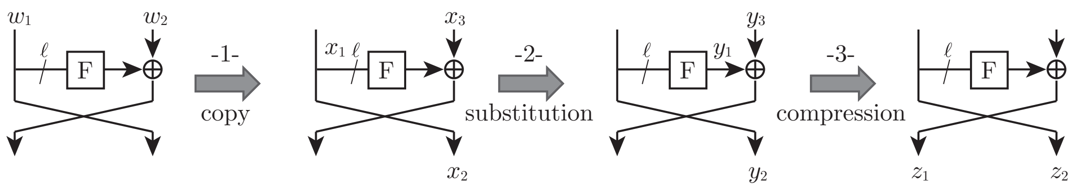


Figure 5.6: Propagation characteristic for Feistel network

5.4.2 Propagation Characteristic for Feistel Network

This section shows that the division property is useful to construct integral distinguishers on (ℓ, d)-Feistel. Since the Feistel Network has "copy," "substitution," and "xor," where the "copy" creates the input of the F-function, the "substitution" processes the input by the F-function, and finally the "xor" creates the left half of the output by XOR. Therefore, we use Proposition 5.3, 5.2, and 5.4. Figure 5.6 shows the outline of the propagation characteristic.

-1- Copy Assume that the input set has the division property $\mathcal{D}^{\ell^2}_{\mathbb{K}}$. From Proposition 5.3, the output division property $\mathcal{D}^{\ell^3}_{\mathbb{K}'}$ is computed as follows: First, $\mathbb{K}'$ is initialized to ϕ. Then, for all $\vec{k} \in \mathbb{K}$

$$\mathbb{K}' = \mathbb{K}' \cup \{[0, k_1, k_2], [1, k_1 - 1, k_2], \dots, [k_1, 0, k_2]\}.$$

-2- Substitution The F-function is an ℓ-bit function with degree d. Assume that the input set has the division property $\mathcal{D}^{\ell^3}_{\mathbb{K}}$. From Proposition 5.2, the output division property has $\mathcal{D}^{\ell^3}_{\mathbb{K}'}$ is computed as follows: First, $\mathbb{K}'$ is initialized to ϕ. Then, for all $\vec{k} \in \mathbb{K}$

$$\mathbb{K}' = \mathbb{K}' \cup \{[\lceil k_1/d \rceil, k_2, k_3]\}.$$

If the F-function is limited to a permutation, k'_1 is ℓ when $k_1 = \ell$.

-3- Xor Assume that the input set has the division property $\mathcal{D}^{\ell^3}_{\mathbb{K}}$. From Proposition 5.4, the output division property has $\mathcal{D}^{\ell^2}_{\mathbb{K}'}$ is computed as follows: First, $\mathbb{K}'$ is initialized to ϕ. Then, for all $\vec{k} \in \mathbb{K}$ satisfying $k_1 + k_3 \leq \ell$

$$\mathbb{K}' = \mathbb{K}' \cup \{[k_1 + k_3, k_2]\}.$$

5.4.3 Path Search Algorithm for (ℓ, d)-Feistel

This section shows the path search algorithm for integral distinguishers against (ℓ, d)-Feistel. The algorithm is based on the propagation characteristic shown in Sect. 5.4.2. Assume that k_1 bits of the left half of the input are active and the rest $(\ell - k_1)$ bits are constant. Moreover, assume that k_2 bits of the right half of the input are active and the rest $(\ell - k_2)$ bits are constant. Namely, we prepare $2^{k_1+k_2}$ chosen plaintexts. The input set has the division property $\mathcal{D}^{\ell^2}_{[k_1,k_2]}$. Algorithm 1 shows the path search algorithm to create the integral distinguisher on (ℓ, d)-Feistel. Algorithm 1 does not limit the F-function to be a permutation. If the F-function is limited to be a permutation, L becomes $k_2 + \ell$ when $X = \ell$ holds (see the 5th line in Algorithm 1). Algorithm 1 calls `SizeReduce`, which eliminates $\vec{k} \in \mathbb{K}$ if there exists $\vec{k}' \in \mathbb{K}$ satisfying $\vec{k} \succeq \vec{k}'$.

Algorithm 1 Path search algorithm for integral distinguishers on (ℓ, d)-Feistel

```
procedure FeistelFuncEval($\ell, d, \mathbb{K}$)
    $\mathbb{K}' \Leftarrow \phi$
    for all $\vec{k} \in \mathbb{K}$ do
        for $X = 0$ to $k_1$ do
            $L \Leftarrow k_2 + \lceil X/d \rceil$
            if $L \leq \ell$ then
                $\mathbb{K}' = \mathbb{K}' \cup (L, k_1 - X)$
            end if
        end for
    end for
    return SizeReduce($\mathbb{K}'$)
end procedure

procedure IntegralPathSearch($\ell, d, r = 0, k_1, k_2$)
    $\mathbb{K} \Leftarrow$ FeistelFuncEval($\ell, d, \{[k_1, k_2]\}$)
    $D \Leftarrow \max_{\vec{k} \in \mathbb{K}} \{k_1 + k_2\}$
    while $1 < D$ do
        $r \Leftarrow r + 1$
        $\mathbb{K} \Leftarrow$ FeistelFuncEval($\ell, d, \mathbb{K}$)
        $D \Leftarrow \max_{\vec{k} \in \mathbb{K}} \{k_1 + k_2\}$
    end while
    return $r$
end procedure
```

Results. Table 5.2 shows the number of required chosen plaintexts to construct r-round integral distinguishers on $(32, 5)$- and $(64, 7)$-Feistel, where DES [Nat77] is classified into $(32, 5)$-Feistel with non-bijective function and Camellia [AIK$^+$00] is classified into $(64, 7)$-Feistel with bijective function. When we construct the integral distinguisher on (ℓ, d)-Feistel with 2^D chosen plaintexts, we use (k_1, k_2) satisfying

$$(k_1, k_2) = \begin{cases} (D - \ell, \ell) & \text{for } \ell \leq D, \\ (0, D) & \text{for } D < \ell. \end{cases}$$

For the comparison with our integral distinguishers, we consider two previous methods, one is the propagation characteristic of the integral property and another is the estimation of the algebraic degree. We first consider the propagation characteristic of the integral property. If the F-function is a non-bijective function, the propagation characteristic does not construct effective distinguishers. Therefore, results introduced by the integral property are only shown when the F-function is bijective. We next consider the estimation of the algebraic degree. However, since we do not know the improved bound against the Feistel Network, we use the trivial bound for the Feistel Network. Assume that the left half of the plaintext is constant. For any r-round (ℓ, d)-Feistel, it can be observed that the function, which associates the right half of the ciphertext with the right half of the plaintext, has degree at most d^{r-2} for $2 \leq r$. Therefore, we can construct the r-round integral distinguishers with $2^{d^{r-2}+1}$ chosen plaintexts. Since the right half of the plaintext is at most ℓ bits, the distinguisher can be constructed with $d^{r-2} + 1 \leq \ell$.

As a result, as far as we try, all distinguishers constructed by the division property are "better" than those by previous methods. We summarize integral distinguishers on (ℓ, d, m)-SPN in Table 5.4. We already know a better integral distinguisher on Camellia in [YPK02], but it is constructed by using the specific feature of Camellia. On the other hand, our method is generic distinguishing attacks against (ℓ, d)-Feistel. From the result of $(64, 7)$-Feistel, it shows that even if the F-function of Camellia is chosen from any functions with degree 7, the modified Camellia has the 6-round integral distinguisher.

If we construct the dedicated path search algorithm for the specific cipher, we expect that the algorithm can create better integral distinguishers.

Table 5.2: The number of chosen plaintexts to construct r-round integral distinguishers on $(32,5)$- and $(64,7)$-Feistel. Our distinguishers are got by implementing Algorithm 1.

Target [Application]	F-function	$\log_2(\#\text{texts})$						Method	Reference
		4R	5R	6R	7R	8R	9R		
$(32,5)$-Feistel [DES]		26	51	62	-	-	-	our	Sect. 5.4.3
		26	-	-	-	-	-	degree	[Knu94, BC13]
$(64,7)$-Feistel [Camellia]	bijection	50	98	124	-	-	-	our	Sect. 5.4.3
		50	-	-	-	-	-	degree	[Knu94, BC13]
		64	-	-	-	-	-	integral	[KW02]

Table 5.3: The number of chosen plaintexts to construct r-round integral distinguishers on the Simon family, where the F-function is not bijective. Our distinguishers are got by implementing Algorithm 1.

Target [Application]	$\log_2(\#\text{texts})$								Method	Reference
	6R	7R	8R	9R	10R	11R	12R	13R		
$(16,2)$-Feistel [Simon32]	17	25	29	31	-	-	-	-	our	Sect. 5.4.3
	-	-	-	-	-	-	-	-	degree	[Knu94, BC13]
$(24,2)$-Feistel [Simon48]	17	29	39	44	46	47	-	-	our	Sect. 5.4.3
	17	-	-	-	-	-	-	-	degree	[Knu94, BC13]
$(32,2)$-Feistel [Simon64]	17	33	49	57	61	63	-	-	our	Sect. 5.4.3
	17	-	-	-	-	-	-	-	degree	[Knu94, BC13]
$(48,2)$-Feistel [Simon96]	17	33	57	77	87	92	94	95	our	Sect. 5.4.3
	17	33	-	-	-	-	-	-	degree	[Knu94, BC13]
$(64,2)$-Feistel [Simon128]	17	33	65	97	113	121	125	127	our	Sect. 5.4.3
	17	33	-	-	-	-	-	-	degree	[Knu94, BC13]

Integral Distinguishers on Simon Family. Although our attack is a generic attack, it can create new integral distinguishers on the Simon family [BSS+13]. Simon is a lightweight block ciphers proposed by the National Security Agency (NSA). Since Simon has a non-bijective F-function and bit-oriented structure, it is complicated task to construct the integral distinguisher. The division property theoretically shows that Simon32, 48, 64, 96, and 128 have at least 9-, 11-, 11-, 13-, and 13-round integral distinguishers, respectively. Table 5.3 shows the comparison between our distinguishers and previous ones by the degree estimation. On the other hand, Wang et al. showed that Simon32 has the 15-round integral distinguisher by experiments [WLV+14]. Therefore, there are 6-round differences between our theoretical result and Wang's experimental result. Our distinguisher is valid against all $(32,2)$-Feistel and it does not exploit the feature of the round function. Namely, we expect that the 6-round difference is derived from the specification of the round function of Simon32. We discuss this topic in Chap. 8.

5.5 Improved Distinguishers on Substitute-Permutation Network

5.5.1 Substitute-Permutation Network

(ℓ,d,m)-SPN. The Substitute-Permutation Network (SPN) is another important structure for block ciphers. The SPN has a round function that consists of an S-Layer and P-Layer, and a block cipher is designed by iterating the round function. We now define an (ℓ,d,m)-SPN, whose round function has m ℓ-bit S-boxes in the S-Layer and one (ℓm)-bit linear function in the P-Layer. Here, each S-box is any bijective function whose degree is at most d, and an (ℓm)-bit linear function is any bijective function whose degree is at most 1. Figure 5.7 shows the round function of the SPN. Nowadays, many block ciphers adopting (ℓ,d,m)-SPN have been proposed, e.g. AES [U.S01], PRESENT [BKL+07], and Serpent [ABK98] adopt $(8,7,16)$-, $(4,3,16)$-, and $(4,3,32)$-SPN, respectively. Moreover, Keccak-f [DBPA11], which is a permutation in the hash function Keccak, can be regarded as $(5,2,320)$-SPN.

Table 5.4: The number of required chosen plaintexts to construct r-round integral distinguishers on (ℓ, d)-Feistel. We get these values by implementing Algorithm 1.

Target	F-function	$\log_2(\#\text{texts})$									Examples
		6R	7R	8R	9R	10R	11R	12R	13R	14R	
$(16, 2)$		17	25	29	31	-	-	-	-	-	SIMON32 [BSS+13]
	bijection	16	23	28	30	31	-	-	-	-	
$(24, 2)$		17	29	39	44	46	47	-	-	-	SIMON48 [BSS+13]
	bijection	17	27	38	43	46	47	-	-	-	
$(32, 2)$		17	33	49	57	61	63	-	-	-	SIMON64 [BSS+13]
	bijection	17	32	47	56	60	62	63	-	-	
$(48, 2)$		17	33	57	77	87	92	94	95	-	SIMON96 [BSS+13]
	bijection	17	33	55	76	86	91	94	95	-	
$(64, 2)$		17	33	65	97	113	121	125	127	-	SIMON128 [BSS+13]
	bijection	17	33	64	95	112	120	124	126	127	

Target	F-function	$\log_2(\#\text{texts})$									Examples
		3R	4R	5R	6R	7R	8R	9R	10R	11R	
$(32, 5)$		6	26	51	62	-	-	-	-	-	DES [Nat77]
	bijection	6	26	46	61	-	-	-	-	-	
$(48, 5)$		6	26	64	90	95	-	-	-	-	
	bijection	6	26	59	89	95	-	-	-	-	
$(64, 5)$		6	26	77	118	126	-	-	-	-	
	bijection	6	26	72	117	126	-	-	-	-	

Target	F-function	$\log_2(\#\text{texts})$									Examples
		3R	4R	5R	6R	7R	8R	9R	10R	11R	
$(32, 7)$		8	35	60	-	-	-	-	-	-	
	bijection	8	32	59	-	-	-	-	-	-	
$(48, 7)$		8	49	90	-	-	-	-	-	-	
	bijection	8	48	84	95	-	-	-	-	-	
$(64, 7)$		8	50	104	125	-	-	-	-	-	
	bijection	8	50	98	124	-	-	-	-	-	Camellia [AIK+00]

Target	F-function	$\log_2(\#\text{texts})$									Examples
		3R	4R	5R	6R	7R	8R	9R	10R	11R	
$(32, 31)$		32	62	-	-	-	-	-	-	-	
	bijection	32	32	63	-	-	-	-	-	-	
$(48, 47)$		48	94	-	-	-	-	-	-	-	
	bijection	48	48	95	-	-	-	-	-	-	
$(64, 63)$		64	126	-	-	-	-	-	-	-	
	bijection	64	64	127	-	-	-	-	-	-	
$(32, 32)$		33	-	-	-	-	-	-	-	-	
$(48, 48)$		49	-	-	-	-	-	-	-	-	
$(64, 64)$		65	-	-	-	-	-	-	-	-	

5.5.2 Propagation Characteristic for SPN

This section shows that the division property is useful to construct integral distinguishers on (ℓ, d, m)-SPN. We first prepare the set of the input of the S-Layer such that k_i bits of the input of the ith S-box are active and the rest $(\ell - k_i)$ bits are constant. In this case, the input set has the division property $\mathcal{D}^{\ell^m}_{\vec{k}}$. We first evaluate the propagation characteristic against the S-Layer. Next, the P-Layer is applied but the input and output take a value of $\mathbb{F}_2^{\ell m}$. Therefore, we need to convert the division property $\mathcal{D}^{\ell^m}_{\vec{k}}$ into $\mathcal{D}^{\ell m}_{k}$ and then evaluate the propagation characteristic against the P-Layer. Since the S-Layer is applied again after the P-Layer, we convert the division property $\mathcal{D}^{\ell m}_{k}$ into $\mathcal{D}^{\ell^m}_{\mathbb{K}}$. After the second round, we evaluate the propagation characteristic of this collective division property.

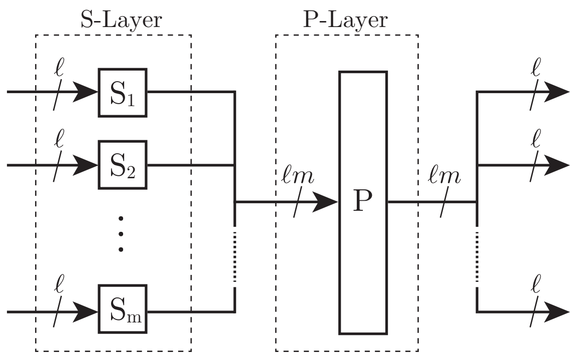


Figure 5.7: (ℓ, d, m)-SPN

-1- S-Layer Assume that the input set has the division property $\mathcal{D}^{\ell^m}_{\mathbb{K}}$. From Proposition 5.2, the output division property $\mathcal{D}^{\ell^m}_{\mathbb{K}'}$ is computed from $\mathbb{K}$ as

$$k'_i = \begin{cases} \lceil k_i/d \rceil & \text{if } k_i < \ell, \\ \ell & \text{if } k_i = \ell, \end{cases}$$

for all $\vec{k} \in \mathbb{K}$.

-2- Concatenation (Conversion form S-Layer to P-Layer) The output of the S-Layer is expressed in a value of $(\mathbb{F}_2^\ell)^m$, but the input of the P-Layer is expressed in a value of $\mathbb{F}_2^{\ell m}$. The transformation is generally implemented by a simple bit concatenation. Assume that the input set has the division property $\mathcal{D}^{\ell^m}_{\mathbb{K}}$. From Proposition 5.6, the output division property $\mathcal{D}^{\ell m}_{k'}$ is computed from $\mathbb{K}$ as

$$k' = \min_{\vec{k} \in \mathbb{K}} \left\{ \sum_{i=1}^{m} k_i \right\}.$$

-3- P-Layer The P-Layer consists of an (ℓm)-bit linear function. Since the degree of the linear function is at most 1, there is no change in the division property.

-4- Partition (Conversion form P-Layer to S-Layer) The output of the P-Layer is expressed in a value of $\mathbb{F}_2^{\ell m}$, but the input of the S-Layer is expressed in a value of $(\mathbb{F}_2^\ell)^m$. The transformation is generally implemented by a simple bit partition. Assume that the input set has the division property $\mathcal{D}^{\ell m}_{k}$. From Proposition 5.5, the output division property has $\mathcal{D}^{\ell^m}_{\mathbb{K}'}$ is computed as follows: First, $\mathbb{K}'$ is initialized to ϕ. Then, for all $\vec{k}'$ satisfying $\sum_{i=1}^{m} k'_i = k$

$$\mathbb{K}' = \mathbb{K}' \cup \{\vec{k}'\}.$$

We can construct the integral distinguisher by evaluating the propagation characteristic of the collective division property. However, since $|\mathbb{K}|$ extremely expands, it is infeasible to execute the straightforward implementation. Therefore, we show more efficient technique. Let $\mathbb{X}$ be the input set of the S-Layer, and the elements take a value of $(\mathbb{F}_2^\ell)^m$. Assume that the input set has the division property $\mathcal{D}^{\ell^m}_{\mathbb{K}}$ that is created by the partition of the division property $\mathcal{D}^{\ell m}_{k}$. If $k > (\ell-1)m$, at least $(m - \ell m + k)$ elements of $\vec{k} \in \mathbb{K}$ have to become ℓ. In this case, the rest elements have to become $\ell - 1$. Since the S-Layer derives $\lceil \frac{\ell-1}{d} \rceil$ and ℓ from $(\ell-1)$ and ℓ, respectively, the output set has the division property $\mathcal{D}^{\ell m}_{k'}$, where k' is calculated as

$$k' = \begin{cases} \left\lceil \frac{\ell-1}{d} \right\rceil (\ell m - k) + \ell(m - \ell m + k) & \text{for } k > (\ell-1)m, \\ \left\lceil \frac{k}{d} \right\rceil & \text{for } k \le (\ell-1)m. \end{cases}$$

Here, if $k \le (\ell-1)m$ holds, we simply regard the round function of (ℓ, d, m)-SPN as one (ℓm)-bit S-box with degree d.

Algorithm 2 Path search algorithm for integral distinguishers on (ℓ, d, m)-SPN

```
procedure IntegralPathSearch($\ell, d, m, r = 0, k_1, k_2, \ldots, k_m$)
    if $k_i < \ell$ then $k_i \Leftarrow \lceil k_i/d \rceil$                    ▷ 1-st round S-Layer
    end if
    $k \Leftarrow \sum_{i=1}^{m} k_i$                    ▷ 1-st round Concatenation and P-Layer
    while $1 < k$ do
        $r \Leftarrow r + 1$
        if $k \leq (\ell - 1)m$ then $k \Leftarrow \lceil k/d \rceil$                    ▷ $(r + 1)$th round
        else $k \Leftarrow \lceil \frac{\ell-1}{d} \rceil (\ell m - k) + \ell(m - \ell m + k)$                    ▷ $(r + 1)$th round
        end if
    end while
    return $r$
end procedure
```

Table 5.5: The number of chosen plaintexts to construct r-round integral distinguishers on (ℓ, d, m)-SPN. Our distinguishers are got by implementing Algorithm 2.

Target	$\log_2(\#\text{texts})$					Method	Reference
	3R	4R	5R	6R	7R		
$(4, 3, 16)$-SPN	12	28	52	60	-	our	Sect. 5.5.3
[PRESENT]	28	52	60	63	-	degree	[BCC11]
$(8, 7, 16)$-SPN	56	120	-	-	-	our	Sect. 5.5.3
[AES]	117	127	-	-	-	degree	[BCC11]

Table 5.6: The number of chosen plaintexts to construct r-round integral distinguishers on Keccak-f and Serpent. Our distinguishers are got by implementing Algorithm 2.

Target [Application]	$\log_2(\#\text{texts})$								Method	Reference
	3R	4R	5R	6R	7R	8R	9R	10R		
$(4, 3, 32)$-SPN	12	28	84	113	124	-	-	-	our	Sect. 5.5.3
[Serpent]	28	82	113	123	127	-	-	-	degree	[BCC11]
Target [Application]									**Method**	**Reference**
	8R	9R	10R	11R	12R	13R	14R	15R		
$(5, 2, 320)$-SPN	130	258	515	1025	1410	1538	1580	1595	our	Sect. 5.5.3
[Keccak-f]	257	513	1025	1409	1537	1579	1593	1598	degree	[BCC11]

5.5.3 Path Search Algorithm for (ℓ, d, m)-SPN

We now consider integral distinguishers on (ℓ, d, m)-SPN. We first prepare the set of chosen plaintexts such that k_i bits of the input of the ith S-box are active and the rest $(\ell - k_i)$ bits are constant. Namely, we prepare $2^{\sum_{i=1}^{m} k_i}$ chosen plaintexts. The input set has the division property $\mathcal{D}_{\vec{k}}^{\ell^m}$. Algorithm 2 shows the path search algorithm to construct the integral distinguisher.

Results. Table 5.5 shows the number of required chosen plaintexts to construct the r-round integral distinguisher on $(4, 3, 16)$- and $(8, 7, 16)$-SPN, where PRESENT [BKL+07] and AES [U.S01] are classified into $(4, 3, 16)$- and $(8, 7, 16)$-SPN, respectively. When we construct the integral distinguisher on (ℓ, d, m)-SPN with 2^D chosen plaintexts, we use a vector $\vec{k}$ satisfying

$$k_i = \begin{cases} \ell & \text{for } i\ell \leq D, \\ D - (i-1)\ell & \text{for } (i-1)\ell \leq D < i\ell, \\ 0 & \text{for } D < (i-1)\ell. \end{cases}$$

For the comparison with our integral distinguishers, we first consider the propagation characteristic of the integral property. However, it does not construct effective distinguishers because the P-Layer is any linear function. Next, we estimate the algebraic degree by using the method proposed by Boura et al. (see Theorem 5.1).

As a result, as far as we try, all distinguishers constructed by the division property are "better" than those by previous methods. We summarize integral distinguishers on (ℓ, d, m)-SPN in Table 5.7. We already know the 7-round integral distinguisher on PRESENT in [WW13] and the 4-round integral distinguisher on AES in [KW02]. However, they are constructed by using the specific feature of each block cipher. On the other hand, our method is generic distinguishing attacks against (ℓ, d, m)-SPN. From the result of $(4, 3, 16)$-SPN, it shows that even if the P-Layer of PRESENT is chosen from any bijective linear functions, the modified PRESENT has the 6-round integral distinguisher. Similarly, from the result of $(8, 7, 16)$-SPN, it shows that even if the P-Layer of AES is chosen from any bijective linear function, the modified AES still has the 4-round integral distinguisher.

If we construct the dedicated path search algorithm for the specific cipher, we expect that the algorithm can create better integral distinguishers.

Table 5.7: The number of required chosen plaintexts to construct r-round integral distinguishers on (ℓ, d, m)-SPN. We get these values by implementing Algorithm 2.

Target	Size	$\log_2(\#\text{texts})$							Examples
	(bits)	4R	5R	6R	7R	8R	9R	10R	
$(4,3,16)$	64	28	52	60	-	-	-	-	PRESENT [BKL+07] LED [GPPR11]
$(4,3,24)$	96	28	76	89	-	-	-	-	
$(4,3,32)$	128	28	84	113	124	-	-	-	Serpent [ABK98] NOEKEON [DPAR00]
$(4,3,40)$	160	28	84	136	152	-	-	-	
$(4,3,48)$	192	28	84	156	180	188	-	-	
$(4,3,56)$	224	28	84	177	209	220	-	-	
$(4,3,64)$	256	28	84	200	237	252	-	-	Prøst-128 [KLL+14a] Minalpher-P [STA+14]
$(4,3,128)$	512	28	84	244	424	484	504	509	Prøst-256 [KLL+14a]

Target	Size	$\log_2(\#\text{texts})$							Examples
	(bits)	5R	6R	7R	8R	9R	10R	11R	
$(5,2,40)$	200	18	35	65	130	178	195	-	PRIMATE-80 [ABB+14]
$(5,2,56)$	280	18	35	65	130	230	265	275	PRIMATE-120 [ABB+14]
$(5,2,64)$	320	18	35	65	130	258	300	315	ASCON P [DEMS14]

Target	Size	$\log_2(\#\text{texts})$							Examples
	(bits)	9R	10R	11R	12R	13R	14R	15R	
$(5,2,160)$	800	258	515	705	770	790	798	-	KECCAK-f[800] [DBPA11]
$(5,2,256)$	1280	258	515	1025	1195	1253	1271	1278	
$(5,2,320)$	1600	258	515	1025	1410	1538	1580	1595	KECCAK-f[1600] [DBPA11]

Target	Size	$\log_2(\#\text{texts})$							Examples
	(bits)	3R	4R	5R	6R	7R	8R	9R	
$(5,4,40)$	200	20	65	170	195	-	-	-	
$(5,4,56)$	280	20	65	230	270	-	-	-	
$(5,4,64)$	320	20	65	260	305	-	-	-	
$(5,4,160)$	800	20	65	260	665	770	795	-	
$(5,4,256)$	1280	20	65	260	1025	1220	1265	-	ICEPOLE P [MGH+14]
$(5,4,320)$	1600	20	65	260	1025	1460	1565	1595	

Target	Size	$\log_2(\#\text{texts})$							Examples
	(bits)	3R	4R	5R	6R	7R	8R	9R	
$(8,7,16)$	128	56	120	-	-	-	-	-	AES [U.S01]
$(8,7,24)$	192	56	176	-	-	-	-	-	Rijndael-192 [DR02]
$(8,7,32)$	256	56	232	-	-	-	-	-	Rijndael-256 [DR02]
$(8,7,64)$	512	56	344	488	-	-	-	-	WHIRLPOOL primitive [BR03]

Integral Distinguishers on Serpent and KECCAK-f. Although our attack is a generic attack, it can create new integral distinguishers on Serpent and KECCAK-f. Serpent is one of AES finalists and is classified into $(4, 3, 32)$-SPN. The existing integral distinguisher is shown in [ZRHD08], and it shows that Serpent has 3.5-round integral distinguisher. On the other hand, we show that all $(4, 3, 32)$-SPNs have at least 7-round integral distinguishers with 2^{124} chosen plaintexts. Table 5.6 shows the comparison between our distinguishers and previous ones by the degree estimation.

Keccak is chosen as SHA-3, and the core function Keccak-f is classified into $(5, 2, 320)$-SPN. Boura et al. estimated the algebraic degree of Keccak-f in [BCC11]. We search for the integral distinguisher by using Algorithm 2. As a result, our distinguishers can reduce the number of chosen plaintexts compared with previous ones. Table 5.6 shows the comparison between our distinguishers and previous ones.

5.6 Improved Distinguishers on AES-Like Ciphers

We introduced the division property and proposed distinguishing attacks against the Feistel and Substitute-Permutation Networks. In this section, we show that the division property is also useful to construct the dedicated attack against specific ciphers. As an example, we apply the division property to AES-like ciphers. AES is the most famous block cipher and is standardized by the National Institute of Standards and Technology (NIST) in 2001. Many block ciphers, hash functions, and authenticated encryptions with AES-like design have been proposed after the standardization. The division property can find non trivial property on AES and improves integral distinguishers on many AES-like ciphers.

5.6.1 AES-Like Cipher

(ℓ, d, m)-AES. AES is a 128-bit block cipher, and an intermediate text of AES is expressed in a 4×4 matrix whose elements are 8 bits. The round function of AES consists of SubBytes, ShiftRows, MixColumns, and AddRoundKey, where each function is defined as follows:

- SubBytes : It substitutes each byte in the matrix into another byte by an S-box.
- ShiftRows : Each byte of the ith row is rotated $i - 1$ bytes to the left.
- MixColumns : It diffuses bytes within each column by a linear function.
- AddRoundKey : A round key is XORed with the intermediate text.

We define an (ℓ, d, m)-AES, where ℓ, d, and m denote the bit length of an S-box, the algebraic degree of an S-box, and the size of the matrix, respectively. The intermediate text is expressed in an $m \times m$ matrix whose elements are ℓ bits. Let $\vec{x} \in (\mathbb{F}_2^\ell)^{m^2}$ be an input of the round function, which is arranged as

$$\begin{bmatrix} x_1 & x_{m+1} & \cdots & x_{m^2-m+1} \\ x_2 & x_{m+2} & \cdots & x_{m^2-m+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_{2m} & \cdots & x_{m^2-m+m} \end{bmatrix}.$$

Let $\vec{y} \in (\mathbb{F}_2^\ell)^{m^2}$ be an output of the round function, which is calculated as

$$\vec{y} = (\text{AddRoundKey} \circ \text{MixColumns} \circ \text{ShiftRows} \circ \text{SubBytes})(\vec{x}).$$

Each function is the same as that of AES except for the scale. For instance, AES [U.S01] and `LED` [GPPR11] adopt $(8, 7, 4)$-AES and $(4, 3, 4)$-AES, respectively. Moreover, P_{256} of `PHOTON` [GPP11] adopts $(4, 3, 8)$-AES[2].

5.6.2 Path Search Algorithm for (ℓ, d, m)-AES

Section 5.5 shows how to construct integral distinguishers on (ℓ, d, m)-SPN, but practical block ciphers have a specific P-Layer. For instance, the P-Layer in AES consists of ShiftRows and MixColumns, and it is not any linear function. Taking into account the structure of the P-Layer, we can construct more effective algorithm. In this section, as an example, we show a path search algorithm to construct integral distinguishers on (ℓ, d, m)-AES. `AesFuncEval` in Algorithm 3

[2]Since `PHOTON` is a hash function, it uses AddConstant instead of AddRoundKey.

Algorithm 3 Path search for integral characteristics on (ℓ, d, m)-AES

```
 1: procedure AesFuncEval($\ell, d, m, \vec{k}$)
 2:     for $i = 1$ to $m^2$ do
 3:         if $k_i < \ell$ then $k_i \Leftarrow \lceil k_i/d \rceil$                ▷ SubBytes
 4:         end if
 5:     end for
 6:     $\vec{k} \Leftarrow$ ShiftRows($\vec{k}$)                                 ▷ ShiftRows
 7:     $k'_c \Leftarrow \sum_{r=1}^{m} k'_{m(c-1)+r}$ for all $c$                ▷ MixColumns
 8:     $\vec{k}' \Leftarrow$ sort($\vec{k}'$)
 9:     return $\vec{k}'$
10: end procedure

11: procedure IntegralPathSearch($\ell, d, m, r = 0, \vec{k} \in \{0, 1, \dots, \ell\}^{m^2}$)
12:     $\mathbb{K} \Leftarrow$ AesFuncEval($\ell, d, m, \vec{k}$)                ▷ 1-st round
13:     $D \Leftarrow \max_{\vec{k} \in \mathbb{K}}(\sum_{c=1}^{m} k_c)$
14:     while $1 < D$ do
15:         $r \Leftarrow r + 1$
16:         $\mathbb{K}' = \phi$
17:         for all $\vec{k} \in \mathbb{K}$ do
18:             $\mathbb{K}'' \Leftarrow$ Partition($\vec{k}$)
19:             for all $\vec{k}'' \in \mathbb{K}''$ do
20:                 $\mathbb{K}' \Leftarrow \mathbb{K}' \cup$ AesFuncEval($\ell, d, m, \vec{k}''$)
21:             end for
22:         end for
23:         $\mathbb{K} \Leftarrow$ SizeReduce($\mathbb{K}'$)
24:         $D \Leftarrow \max_{\vec{k} \in \mathbb{K}}(\sum_{c=1}^{m} k_c)$
25:     end while
26:     return $r$
27: end procedure
```

evaluates the propagation characteristic of the division property against the round function of AES-like ciphers, and it calls `ShiftRows` and `sort`. `ShiftRows` performs a similar transformation to `ShifrRows`. `sort` is the sorting algorithm, which is useful for feasible implementation. `IntegralPathSearch` in Algorithm 3 shows the path search algorithm, and it calls `Partition`, `AesFuncEval`, and `SizeReduce`. `Partition`($\vec{k}$) calculates all possible $\vec{k}' \in \{0, 1, \dots, \ell\}^{mm}$ satisfying

$$\left(\sum_{r=1}^{m} k'_r, \sum_{r=1}^{m} k'_{m+r}, \dots, \sum_{r=1}^{m} k'_{m(m-1)+r}\right) = (k_1, k_2, \dots, k_m),$$

`SizeReduce` eliminates $\vec{k} \in \mathbb{K}$ if there exists $\vec{k}' \in \mathbb{K}'$ satisfying $\vec{k} \succeq \vec{k}'$.

Note that $|\mathbb{K}|$ extremely expands when the partition of the division property is executed (see the 18th line in Algorithm 3). Namely, our algorithm takes large execution time and memory capacity if we straightforwardly implement our algorithm. Therefore, we use an effective method, which uses the feature of (ℓ, d, m)-AES, for the feasible implementation. Note that each column of (ℓ, d, m)-AES is equivalent each other. Assuming that the input set has $\mathcal{D}^{(\ell m)^m}_{\vec{k}, \vec{k}'}$ that $\vec{k}'$ is a permutation of elements of $\vec{k}$, the division property of the next round calculated from $\vec{k}$ is exactly the same as that from $\vec{k}'$ because columns of (ℓ, d, m)-AES are equivalent each other. Namely, it is enough to save either, and we implement it by a sorting algorithm (see the 8th line in Algorithm 3). This technique enables us to execute our path search algorithm feasibly in many parameters.

5.6.3 Results for $(4, 3, m)$-AES

Table 5.8 shows the number of required chosen plaintexts to construct r-round integral distinguishers on $(4, 3, m)$-AES. When we construct the integral distinguisher on $(4, 3, m)$-AES with

Table 5.8: The number of chosen plaintexts to construct r-round integral distinguishers on $(4, 3, m)$-AES

Target [Application]	$\log_2(\#\text{texts})$						Method	Reference
	3R	4R	5R	6R	7R	8R		
$(4,3,4)$-AES [LED]	4	12	32	52	-	-	our (AES)	Sect. 5.6.2
	12	28	52	60	-	-	our (SPN)	Sect. 5.5.3
	28	52	60	63	-	-	degree	[BCC11]
	4	16	-	-	-	-	integral	[DKR97, KW02]
$(4,3,5)$-AES [P_{100} in PHOTON]	4	12	20	72	97	-	our (AES)	Sect. 5.6.2
	12	28	76	92	-	-	our (SPN)	Sect. 5.5.3
	28	76	92	98	-	-	degree	[BCC11]
	4	20	-	-	-	-	integral	[DKR97, KW02]
$(4,3,6)$-AES [P_{144} in PHOTON]	4	12	24	84	132	-	our (AES)	Sect. 5.6.2
	12	28	84	124	140	-	our (SPN)	Sect. 5.5.3
	28	82	124	138	142	-	degree	[BCC11]
	4	24	-	-	-	-	integral	[DKR97, KW02]
$(4,3,7)$-AES [P_{196} in PHOTON]	4	12	24	84	164	192	our (AES)	Sect. 5.6.2
	12	28	84	160	184	192	our (SPN)	Sect. 5.5.3
	28	82	158	184	192	195	degree	[BCC11]
	4	28	-	-	-	-	integral	[DKR97, KW02]
$(4,3,8)$-AES [P_{256} in PHOTON]	4	12	28	92	204	249	our (AES)	Sect. 5.6.2
	12	28	84	200	237	252	our (SPN)	Sect. 5.5.3
	28	82	198	237	250	254	degree	[BCC11]
	4	32	-	-	-	-	integral	[DKR97, KW02]

2^D chosen plaintexts, we carefully choose the input matrix $\vec{k}$.

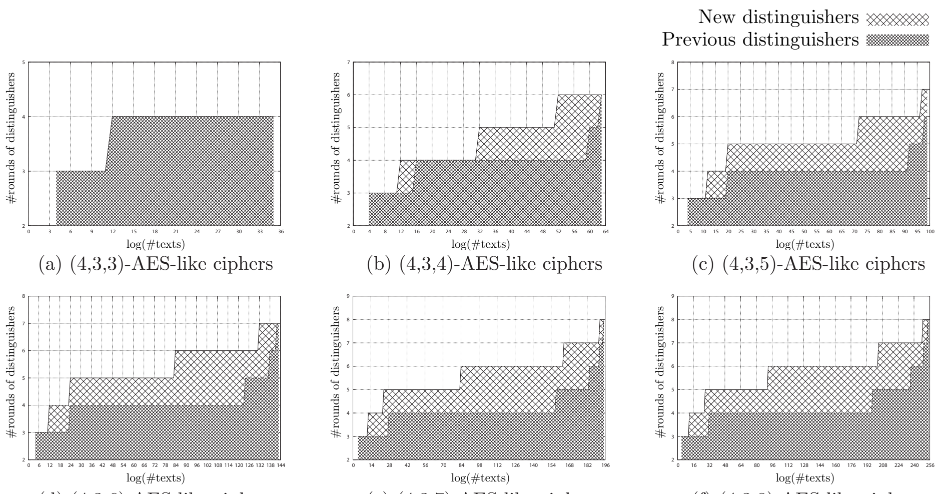
For the comparison with our improved integral distinguishers, we also show integral distinguishers by using the propagation characteristic of the integral property. We also estimate the algebraic degree by the method proposed Boura et al. (see Theorem 5.1). Moreover, since $(4, 3, m)$-AES are classified into $(4, 3, m^2)$-SPN, we construct integral distinguishers by Algorithm 2.

Figure 5.8 shows the comparison between the existing-best distinguisher and our distinguisher. Here the horizontal axis denotes the number of required chosen plaintexts and the vertical axis denotes the number of rounds in integral distinguishers. As a result, as far as we try, all distinguishers constructed by the division property are at least better than those by previous methods. Especially, the advantage of our method is large when we construct the integral distinguisher with the small number of texts. For instance, our method shows that $(4, 3, 8)$-AES, which is adopted by P_{256} in PHOTON, has the 6-round distinguisher with 2^{92} chosen plaintexts. If we regard $(4, 3, 8)$-AES as $(4, 3, 64)$-SPN, 2^{200} chosen plaintexts are required to construct the distinguisher.

5.6.4 Results for $(4, 2, m)$-AES

As the algebraic degree of S-box decreases, the division property can construct longer integral distinguisher because it can exploit the algebraic degree effectively. Therefore, we also evaluate the integral distinguisher on $(4, 2, m)$-AES. Table 5.9 shows the number of required chosen plaintexts to construct r-round integral distinguishers on $(4, 2, m)$-AES, where we carefully choose the input matrix $\vec{K}$ when we construct the integral distinguisher on $(4, 2, m)$-AES with 2^D chosen plaintexts. Note that the propagation of integral properties finds only 4-round integral distinguisher independent on the algebraic degree.

Figure 5.9 shows the comparison between the existing-best distinguisher and our distinguisher. Here the horizontal axis denotes the number of required chosen plaintexts and the vertical axis denotes the number of rounds in integral distinguishers. Similarly to results for $(4, 3, m)$-AES, all distinguishers constructed by the division property are at least better than those by previous methods.



(a) (4,3,3)-AES-like ciphers (b) (4,3,4)-AES-like ciphers (c) (4,3,5)-AES-like ciphers

(d) (4,3,6)-AES-like ciphers (e) (4,3,7)-AES-like ciphers (f) (4,3,8)-AES-like ciphers

Figure 5.8: Comparison between new and previous integral characteristics for $(4, 3, m)$-AES

Table 5.9: The number of chosen plaintexts to construct r-round integral distinguishers on $(4, 2, m)$-AES

Target	$\log_2(\#\text{texts})$							Method
	7R	8R	9R	10R	11R	12R	13R	
$(4, 2, 3)$-AES	34	35	-	-	-	-	-	degree
	25	35	-	-	-	-	-	our (AES)
$(4, 2, 4)$-AES	57	61	63	-	-	-	-	degree
	33	49	61	-	-	-	-	our (AES)
$(4, 2, 5)$-AES	83	92	96	98	99	-	-	degree
	37	60	80	96	99	-	-	our (AES)
$(4, 2, 6)$-AES	105	125	135	140	142	143	-	degree
	37	69	105	125	140	143	-	our (AES)
$(4, 2, 7)$-AES	129	163	180	188	192	194	195	degree
	36	69	116	163	184	192	195	our (AES)
$(4, 2, 8)$-AES	129	193	225	241	249	253	255	degree
	33	65	129	193	225	249	253	our (AES)

5.6.5 Nontrivial Division Property on AES

The most important parameter on AES-like ciphers is $(8, 7, 4)$-AES because AES adopts the parameter. We apply the division property to $(8, 7, 4)$-AES and show nontrivial property.

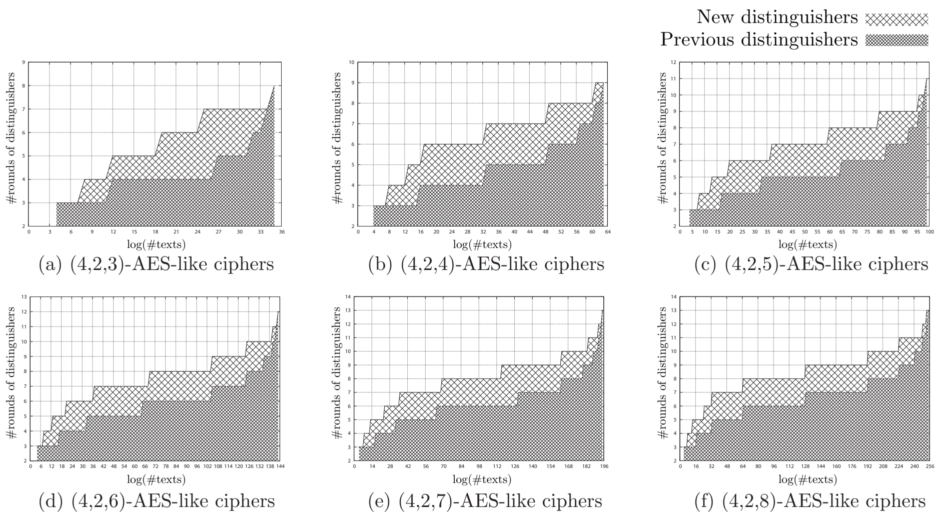
Revisiting 4-round Distinguisher. We first revisit the 4-round integral distinguisher on AES. When the following input for Algorithm 3

$$\vec{k} = \begin{bmatrix} 8 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

is used, we get the following propagation of the division property.

$$\mathcal{D}^{8^{16}}_{\vec{k}} \xrightarrow{4R} \mathcal{D}^{1^{128}}_{2}$$

Therefore, 128 bits encrypted by 4 rounds are balanced.



(a) (4,2,3)-AES-like ciphers (b) (4,2,4)-AES-like ciphers (c) (4,2,5)-AES-like ciphers (d) (4,2,6)-AES-like ciphers (e) (4,2,7)-AES-like ciphers (f) (4,2,8)-AES-like ciphers

Figure 5.9: Comparison between new and previous integral characteristics for $(4, 2, m)$-AES-like ciphers

Nontrivial Property. We next consider the propagation of the division property using more number of chosen plaintexts. When the following input for Algorithm 3

$$\vec{k} = \begin{bmatrix} 8 & 8 & 0 & 0 \\ 0 & 8 & 8 & 0 \\ 0 & 0 & 8 & 8 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

is used, we get the following propagation of the division property.

$$\mathcal{D}^{8^{16}}_{\vec{k}} \xrightarrow{4R} \mathcal{D}^{128}_{3}$$

Therefore, the and of any 2 bits from 128 bits encrypted by 4 rounds is also balanced. Moreover, when the following input for Algorithm 3

$$\vec{k} = \begin{bmatrix} 8 & 8 & 8 & 0 \\ 0 & 8 & 8 & 8 \\ 8 & 0 & 8 & 8 \\ 8 & 8 & 0 & 8 \end{bmatrix}$$

is used, we get the following propagation of the division property.

$$\mathcal{D}^{8^{16}}_{\vec{k}} \xrightarrow{4R} \mathcal{D}^{128}_{4}$$

Therefore, the and of any 3 bits from 128 bits encrypted by 4 rounds is also balanced.

5.7 Conclusion

In this chapter, we proposed the fundamental technique to improve integral distinguishers and showed structural cryptanalyses against the Feistel Network and the SPN. Our new technique uses the division property, which is the generalization of the integral property. It can effectively construct integral distinguishers even if block ciphers have non-bijective functions, bit-oriented structures, and low-degree functions. For the Feistel Network, when the algebraic degree of the F-function is smaller than the bit length of the F-function, our method can attack more rounds than previous generic attacks. Moreover, we theoretically showed that Simon48, 64, 96, and 128 have 11-, 11-, 13-, and 13-round integral distinguishers, respectively. For the SPN, our method extremely reduces the required number of chosen plaintexts compared with previous methods. Moreover, we improved integral distinguishers on Keccak-f and Serpent. The division property is useful to construct integral distinguishers against specific ciphers. As one example, we showed a path search algorithm to construct integral distinguishers on the AES-like cipher, which is the sub class of the SPN. From this fact, we expect that the division property can construct many improved integral distinguishers against specific ciphers by constructing the dedicated path search algorithm.

Chapter 6

Application to MISTY1 –Integral Cryptanalysis on Full MISTY1

Abstract– MISTY1 is a block cipher designed by Matsui in 1997. It was well evaluated and standardized by projects, such as CRYPTREC, ISO/IEC, and NESSIE. In this chapter, we propose a key recovery attack on the full MISTY1, i.e., we show that 8-round MISTY1 with 5 FL layers does not have 128-bit security. Many attacks against MISTY1 have been proposed, but there is no attack against the full MISTY1. Therefore, our attack is the first cryptanalysis against the full MISTY1. We construct a new integral characteristic by using the propagation characteristic of the division property. We first improve the division property by optimizing a public S-box and then construct a 6-round integral characteristic on MISTY1. Finally, we recover the secret key of the full MISTY1 with $2^{63.58}$ chosen plaintexts and 2^{121} time complexity. Moreover, if we use $2^{63.994}$ chosen plaintexts, the time complexity for our attack is reduced to $2^{108.3}$. Note that our cryptanalysis is a theoretical attack. Therefore, the practical use of MISTY1 will not be affected by our attack.
Keywords– MISTY1, Integral attack, Division property

6.1 Introduction

MISTY [Mat97] is a block cipher designed by Matsui in 1997 and is based on the theory of provable security [Nyb94, NK95] against the differential attack [BS90] and linear attack [Mat93]. MISTY has a recursive structure, and the component function has a unique structure, the so-called MISTY structure [Mat96]. There are two types of MISTY, MISTY1 and MISTY2. MISTY1 adopts the Feistel structure whose F-function is designed by the recursive MISTY structure. MISTY2 does not adopt the Feistel structure and uses only the MISTY structure. Both ciphers achieve provable security against differential and linear attacks. MISTY1 is designed for practical use, and MISTY2 is designed for experimental use.

MISTY1 is a 64-bit block cipher with 128-bit key, and it has a Feistel structure with FL layers. MISTY1 is in the candidate recommended ciphers list of CRYPTREC [CRY13], and it is standardized by ISO/IEC 18033-3 [ISO05]. Moreover, it is a NESSIE-recommended cipher [NES04] and is described in RFC 2994 [OM00]. There are many existing attacks against reduced MISTY1, and we summarize these attacks in Table 6.1. A higher-order differential attack is the most powerful attack against MISTY1 [Bar15b]. However, there is no attack against the full MISTY1, i.e., 8-round MISTY1 with 5 FL layers.

Integral Attack. The integral attack [KW02] was first proposed by Daemen et al. to evaluate the security of SQUARE [DKR97] and was then formalized by Knudsen and Wagner. There are two major techniques to construct an integral characteristic; one uses the propagation characteristic of integral properties [KW02], and the other estimates the algebraic degree [Knu94, Lai94]. We often call the second technique a "higher-order differential attack." A new technique to construct integral characteristics was proposed in EUROCRYPT 2015 [Tod15b], and it introduced a new property, the so-called division property, by generalizing the integral property [KW02]. It showed the propagation characteristic of the division property for any function restricted by an

Table 6.1: Summary of single secret-key attacks against MISTY1

Rounds	#FL layers	Attack algorithm	Data	Time	Reference
5	0	higher order differential	11×2^7 CP	2^{17}	[THK99]
5	3	integral	2^{34} CP	2^{48}	[KW02]
5	4	higher order differential	2^{22} CP	2^{28}	[HTK04]
5	4	impossible differential	2^{38} CP	$2^{46.45}$	[DK08]
6	4	higher order differential	$2^{53.7}$ CP	$2^{53.7}$	[TSSK08]
6	4	impossible differential	2^{51} CP	$2^{123.4}$	[DK08]
7	0	impossible differential	$2^{50.2}$ KP	$2^{114.1}$	[DK08]
7	4	higher order differential	$2^{54.1}$ CP	$2^{120.7}$	[TSSK08]
7	4	higher order differential	$2^{50.1}$ CP	$2^{100.4}$	[Bar15b]
7	5	higher order differential	$2^{51.4}$ CP	2^{121}	[Bar15b]
8	5	integral by division property	$2^{63.58}$ CP	2^{121}	This chapter
8	5	integral by division property	$2^{63.994}$ CP	$2^{108.3}$	This chapter

algebraic degree. As a result, several improved results were reported on the structural evaluation of the Feistel network and the Substitution-Permutation network. Please refer to Chap. 5 in detail. Moreover, the division property was applied to the generalized Feistel network [ZW15].

Our Contribution. In Chap. 5, S-boxes are randomly chosen depending on round keys but the algebraic degree is restricted. However, many realistic block ciphers use more efficient structures, e.g., a public S-box and key addition. In this chapter, we show that the division property becomes more useful if an S-box is a public function. Then, we apply our technique to the cryptanalysis of MISTY1. We first evaluate the propagation characteristic of the division property for public S-boxes S_7 and S_9 and show that S_7 has a vulnerable property. We next evaluate the propagation characteristic of the division property for the FI function and then evaluate it for the FO function. Moreover, we evaluate the propagation characteristic for the FL layer. Finally, we devise an algorithm to search for integral characteristics on MISTY1 by assembling these propagation characteristics. As a result, we can construct a new 6-round integral characteristic, where the left 7-bit value of the output is balanced. We recover the round key by using the partial-sum technique [FKL+00]. As a result, the secret key of the full MISTY1 can be recovered with $2^{63.58}$ chosen plaintexts and 2^{121} time complexity. Moreover, if we can use $2^{63.994}$ chosen plaintexts, the time complexity is reduced to $2^{108.3}$. Unfortunately, we have to use almost all chosen plaintexts, and recovering the secret key by using fewer chosen plaintexts is left as an open problem.

6.2 MISTY1

MISTY1 is a Feistel cipher whose F-function has the MISTY structure, and the recommended parameter is 8 rounds with 5 FL layers. Figure 6.1 shows the structure of MISTY1. Let X_i^L (resp. X_i^R) be the left half (resp. the right half) of an i-round input. Moreover, $X_i^L[j]$ (resp. $X_i^R[j]$) denotes the jth bit of X_i^L (resp. X_i^R) from the left. MISTY1 is a 64-bit block cipher with 128-bit key, and it has a Feistel structure with FL layers, where the FO function is used in the F-function of the Feistel structure. The component function FO_i is constructed by using the 3-round MISTY structure, where $FI_{i,1}$, $FI_{i,2}$, and $FI_{i,3}$ are used as the F-function of the MISTY structure, and the four 16-bit round keys $KO_{i,1}$, $KO_{i,2}$, $KO_{i,3}$, and $KO_{i,4}$ are used. Moreover, the function $FI_{i,j}$ is constructed by using the 3-round MISTY structure, where a 9-bit S-box S_9 and 7-bit S-box S_7 are used in the F-function, and a 16-bit round key $KI_{i,j}$ is

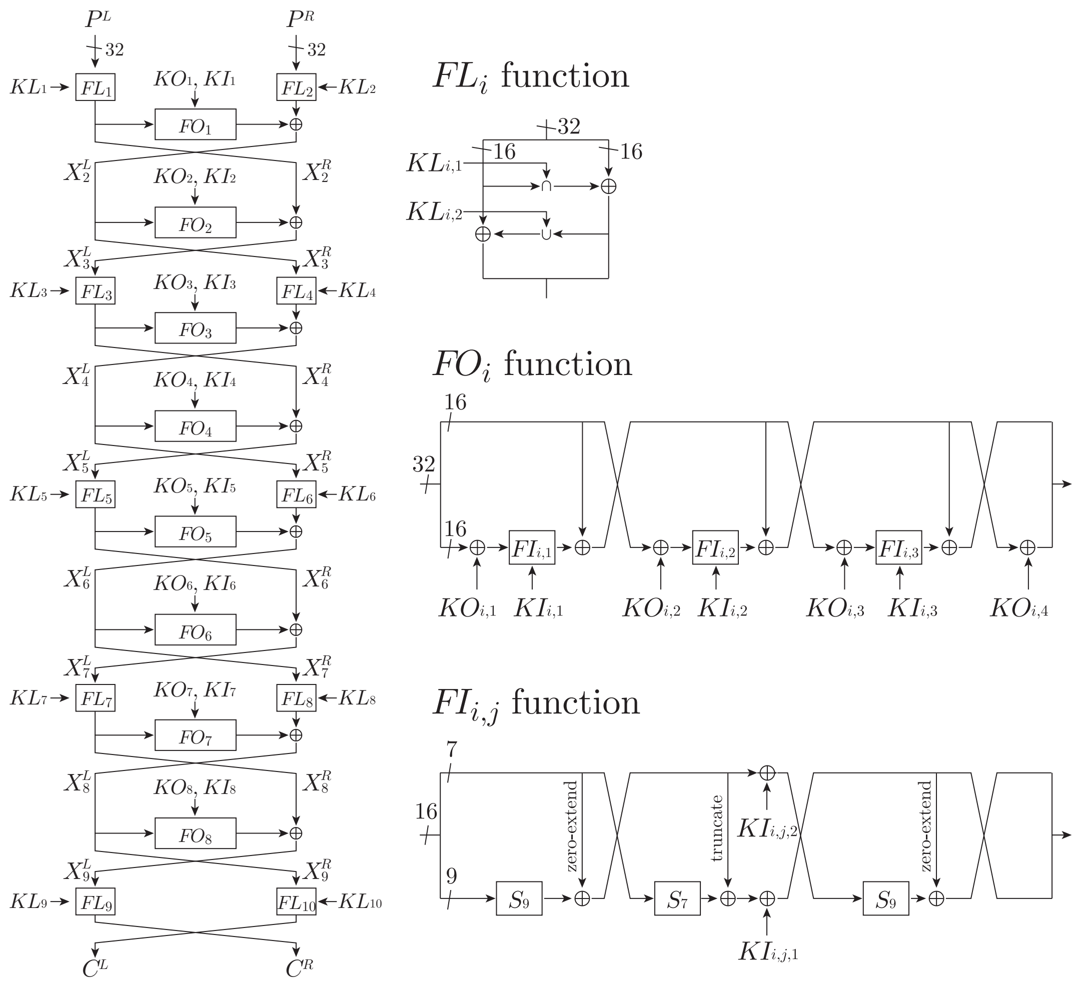


Figure 6.1: Specification of MISTY1

used. Here, the ANF of S_7 is represented as

$$\begin{aligned}
y[0] &= x[0] \oplus x[1]x[3] \oplus x[0]x[3]x[4] \oplus x[1]x[5] \oplus x[0]x[2]x[5] \oplus x[4]x[5] \\
&\quad \oplus x[0]x[1]x[6] \oplus x[2]x[6] \oplus x[0]x[5]x[6] \oplus x[3]x[5]x[6] \oplus 1, \\
y[1] &= x[0]x[2] \oplus x[0]x[4] \oplus x[3]x[4] \oplus x[1]x[5] \oplus x[2]x[4]x[5] \oplus x[6] \oplus x[0]x[6] \\
&\quad \oplus x[3]x[6] \oplus x[2]x[3]x[6] \oplus x[1]x[4]x[6] \oplus x[0]x[5]x[6] \oplus 1, \\
y[2] &= x[1]x[2] \oplus x[0]x[2]x[3] \oplus x[4] \oplus x[1]x[4] \oplus x[0]x[1]x[4] \oplus x[0]x[5] \oplus x[0]x[4]x[5] \\
&\quad \oplus x[3]x[4]x[5] \oplus x[1]x[6] \oplus x[3]x[6] \oplus x[0]x[3]x[6] \oplus x[4]x[6] \oplus x[2]x[4]x[6], \\
y[3] &= x[0] \oplus x[1] \oplus x[0]x[1]x[2] \oplus x[0]x[3] \oplus x[2]x[4] \oplus x[1]x[4]x[5] \oplus x[2]x[6] \\
&\quad \oplus x[1]x[3]x[6] \oplus x[0]x[4]x[6] \oplus x[5]x[6] \oplus 1, \\
y[4] &= x[2]x[3] \oplus x[0]x[4] \oplus x[1]x[3]x[4] \oplus x[5] \oplus x[2]x[5] \oplus x[1]x[2]x[5] \oplus x[0]x[3]x[5] \\
&\quad \oplus x[1]x[6] \oplus x[1]x[5]x[6] \oplus x[4]x[5]x[6] \oplus 1, \\
y[5] &= x[0] \oplus x[1] \oplus x[2] \oplus x[0]x[1]x[2] \oplus x[0]x[3] \oplus x[1]x[2]x[3] \oplus x[1]x[4] \\
&\quad \oplus x[0]x[2]x[4] \oplus x[0]x[5] \oplus x[0]x[1]x[5] \oplus x[3]x[5] \oplus x[0]x[6] \oplus x[2]x[5]x[6], \\
y[6] &= x[0]x[1] \oplus x[3] \oplus x[0]x[3] \oplus x[2]x[3]x[4] \oplus x[0]x[5] \oplus x[2]x[5] \oplus x[3]x[5] \\
&\quad \oplus x[1]x[3]x[5] \oplus x[1]x[6] \oplus x[1]x[2]x[6] \oplus x[0]x[3]x[6] \oplus x[4]x[6] \oplus x[2]x[5]x[6].
\end{aligned}$$

Moreover, the ANF of S_9 is represented as

$$\begin{aligned}
y[0] = {} & x[0]x[4] \oplus x[0]x[5] \oplus x[1]x[5] \oplus x[1]x[6] \oplus x[2]x[6] \oplus x[2]x[7] \oplus x[3]x[7] \oplus x[3]x[8] \\
& \oplus x[4]x[8] \oplus 1, \\
y[1] = {} & x[0]x[2] \oplus x[3] \oplus x[1]x[3] \oplus x[2]x[3] \oplus x[3]x[4] \oplus x[4]x[5] \oplus x[0]x[6] \oplus x[2]x[6] \\
& \oplus x[7] \oplus x[0]x[8] \oplus x[3]x[8] \oplus x[5]x[8] \oplus 1, \\
y[2] = {} & x[0]x[1] \oplus x[1]x[3] \oplus x[4] \oplus x[0]x[4] \oplus x[2]x[4] \oplus x[3]x[4] \oplus x[4]x[5] \oplus x[0]x[6] \\
& \oplus x[5]x[6] \oplus x[1]x[7] \oplus x[3]x[7] \oplus x[8], \\
y[3] = {} & x[0] \oplus x[1]x[2] \oplus x[2]x[4] \oplus x[5] \oplus x[1]x[5] \oplus x[3]x[5] \oplus x[4]x[5] \oplus x[5]x[6] \\
& \oplus x[1]x[7] \oplus x[6]x[7] \oplus x[2]x[8] \oplus x[4]x[8], \\
y[4] = {} & x[1] \oplus x[0]x[3] \oplus x[2]x[3] \oplus x[0]x[5] \oplus x[3]x[5] \oplus x[6] \oplus x[2]x[6] \oplus x[4]x[6] \\
& \oplus x[5]x[6] \oplus x[6]x[7] \oplus x[2]x[8] \oplus x[7]x[8], \\
y[5] = {} & x[2] \oplus x[0]x[3] \oplus x[1]x[4] \oplus x[3]x[4] \oplus x[1]x[6] \oplus x[4]x[6] \oplus x[7] \oplus x[3]x[7] \\
& \oplus x[5]x[7] \oplus x[6]x[7] \oplus x[0]x[8] \oplus x[7]x[8], \\
y[6] = {} & x[0]x[1] \oplus x[3] \oplus x[1]x[4] \oplus x[2]x[5] \oplus x[4]x[5] \oplus x[2]x[7] \oplus x[5]x[7] \oplus x[8] \\
& \oplus x[0]x[8] \oplus x[4]x[8] \oplus x[6]x[8] \oplus x[7]x[8] \oplus 1, \\
y[7] = {} & x[1] \oplus x[0]x[1] \oplus x[1]x[2] \oplus x[2]x[3] \oplus x[0]x[4] \oplus x[5] \oplus x[1]x[6] \oplus x[3]x[6] \\
& \oplus x[0]x[7] \oplus x[4]x[7] \oplus x[6]x[7] \oplus x[1]x[8] \oplus 1, \\
y[8] = {} & x[0] \oplus x[0]x[1] \oplus x[1]x[2] \oplus x[4] \oplus x[0]x[5] \oplus x[2]x[5] \oplus x[3]x[6] \oplus x[5]x[6] \\
& \oplus x[0]x[7] \oplus x[0]x[8] \oplus x[3]x[8] \oplus x[6]x[8] \oplus 1.
\end{aligned}$$

The component function FL_i uses two 16-bit round keys, $KL_{i,1}$ and $KL_{i,2}$, where $\cap$ and $\cup$ denote a bitwise AND and OR, respectively. These round keys are calculated from the secret key $(K_1, K_2, \ldots, K_8)$ as follows.

Symbol	$KO_{i,1}$	$KO_{i,2}$	$KO_{i,3}$	$KO_{i,4}$	$KI_{i,1}$	$KI_{i,2}$	$KI_{i,3}$	$KL_{i,1}$	$KL_{i,2}$
Key	K_i	K_{i+2}	K_{i+7}	K_{i+4}	K'_{i+5}	K'_{i+1}	K'_{i+3}	$K_{\frac{i+1}{2}}$ (odd i) $K'_{\frac{i}{2}+2}$ (even i)	$K'_{\frac{i+1}{2}+6}$ (odd i) $K_{\frac{i}{2}+4}$ (even i)

Here, K_i and K'_i are identified with K_{i-8} and K'_{i-8}, respectively, when i exceeds 8. Moreover, K'_i is defined as the output of $FI_{i,j}$ where the input is K_i and the key is K_{i+1}.

6.3 Integral Characteristic by Division Property

6.3.1 Notations

We make the distinction between the addition over $\mathbb{F}_2^n$ and addition over $\mathbb{Z}$, and we use $\oplus$ and $+$ as the addition over $\mathbb{F}_2^n$ and addition over $\mathbb{Z}$, respectively. For any $a \in \mathbb{F}_2^n$, the ith element is expressed as $a[i]$, and the Hamming weight $w(a)$ is calculated as $w(a) = \sum_{i=1}^{n} a[i]$. Moreover, $a[i_1, i_2, \ldots, i_j]$ denotes a j-bit substring of a as $a[i_1, i_2, \ldots, i_j] = a[i_1] \| a[i_2] \| \cdots \| a[i_j]$. Let $1^n \in \mathbb{F}_2^n$ be a value whose all elements are 1. Moreover, let $0^n \in \mathbb{F}_2^n$ be a value whose all elements are 0. For any set $\mathbb{K}$, let $|\mathbb{K}|$ be the number of elements. Moreover, let ϕ be an empty set. For any $\vec{a} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$, the vectorial Hamming weight is defined as $W(\vec{a}) = [w(a_1), w(a_2), \ldots, w(a_m)] \in \mathbb{Z}^m$, where a_i denotes the ith element of $\vec{a}$. Moreover, for any $\vec{k} \in \mathbb{Z}^m$ and $\vec{k}' \in \mathbb{Z}^m$, we define $\vec{k} \succeq \vec{k}'$ if $k_i \geq k'_i$ for all $i \, (1 \leq i \leq m)$. Otherwise, $\vec{k} \not\succeq \vec{k}'$.

Boolean Function. A Boolean function is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. Let $\deg(f)$ be the algebraic degree of a Boolean function f. Algebraic Normal Form (ANF) is often used as representation of the Boolean function. Let f be any Boolean function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. Then, it can be represented as

$$f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^f \left(\prod_{i=1}^{n} x[i]^{u[i]} \right),$$

where $a_u^f \in \mathbb{F}_2$ is a constant value depending on f and u. If $\deg(f)$ is at most d, all a_u^f satisfying $d < w(u)$ are 0. An n-bit S-box can be regarded as the collection of n Boolean functions. If the algebraic degrees of its n Boolean functions are at most d, we say the algebraic degree of the S-box is at most d.

6.3.2 Integral Attack

An integral attack [KW02] is one of the most powerful cryptanalyses against block ciphers. Attackers prepare N chosen plaintexts and get the corresponding ciphertexts. If the XOR of all corresponding ciphertexts is 0 for all secret keys, we say that the block cipher has an integral characteristic with N chosen plaintexts. In an integral attack, attackers first create an integral characteristic against a reduced-round block cipher. Then, they guess the round keys that are used in the last several rounds and calculate the XOR of the ciphertexts of the reduced-round block cipher. Finally, they evaluate whether or not the XOR is 0. If the XOR is not 0, they can discard the guessed round keys from the candidates of the correct key.

6.3.3 Division Property

A division property, which was proposed in Chap. 5, is used to search for integral characteristics. The division property of chosen plaintexts is first evaluated, and that of texts encrypted by one round is evaluated by the propagation characteristic. By repeating this procedure, the division property of texts encrypted by arbitrary rounds is evaluated. Finally, we can easily determine from the propagated division property whether r rounds have integral characteristics or not.

Bit Product Function. We first define two bit product functions π_u and $\pi_{\vec{u}}$, which are used to evaluate the division property of a multiset[1]. Let $\pi_u : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function for any $u \in \mathbb{F}_2^n$. Let $x \in \mathbb{F}_2^n$ be the input, and $\pi_u(x)$ be the AND of $x[i]$ satisfying $u[i] = 1$, i.e., it is defined as

$$\pi_u(x) := \prod_{i=1}^{n} x[i]^{u[i]}.$$

Let $\pi_{\vec{u}} : (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m}) \to \mathbb{F}_2$ be a function for any $\vec{u} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. Let $\vec{x} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$ be the input, and $\pi_{\vec{u}}(\vec{x})$ be defined as

$$\pi_{\vec{u}}(\vec{x}) := \prod_{i=1}^{m} \pi_{u_i}(x_i).$$

Definition of Division Property. The division property is given against a multiset, and it is calculated by using the bit product function. Let $\mathbb{X}$ be an input multiset whose elements take a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. In the division property, we first evaluate a value of $\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x})$ for all $\vec{u} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. Then, we divide the set of $\vec{u}$ into a subset whose sum is 0 and a subset whose sum becomes unknown[2]. In Chap. 5, the focus was on using the Hamming weight of $\vec{u}$ to divide the set.

Definition 5.3 (Division Property). *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_1,n_2,\ldots,n_m}$, where $\mathbb{K}$ denotes a set of m-dimensional vectors whose elements take a value between 0 and n_i, it fulfills the following conditions:*

$$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} \textit{unknown} & \textit{if there are } \vec{k} \in \mathbb{K} \textit{ s.t. } W(\vec{u}) \succeq \vec{k}, \\ 0 & \textit{otherwise.} \end{cases}$$

If there are $\vec{k} \in \mathbb{K}$ and $\vec{k}' \in \mathbb{K}$ satisfying $\vec{k} \succeq \vec{k}'$, $\vec{k}$ can be removed from $\mathbb{K}$ because it is redundant. Assume that the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_1,n_2,\ldots,n_m}$. If there is no unit vector $\vec{e}_j$ in $\mathbb{K}$, where $\vec{e}_j$ is a vector whose jth element is 1 and the others are 0, $\bigoplus_{x \in \mathbb{X}} x_j$ is 0. See Chap. 5 to better understand the concept in detail.

[1] A multiset allows multiple instances of the elements unlike a set.

[2] If we know all accurate values in a multiset, we can divide the set of $\vec{u}$ into subsets whose evaluated value is 0 or 1. However, in the application to cryptanalysis, we evaluate the multiset whose elements are texts encrypted for several rounds. Such elements change depending on the sub keys and the constant bit of plaintexts. Therefore, we consider subsets whose sum is 0 for all sub keys, and otherwise, we consider the sum as unknown.

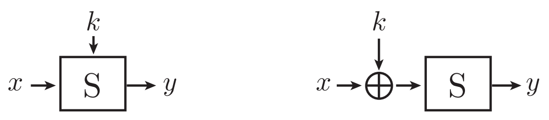


Figure 6.2: The difference between Chap. 5 and this chapter. The left figure is an assumption used in Chap. 5. The right one is a new assumption used in this chapter.

6.4 Division Property for Public Function

In an assumption of Chap. 5, attackers do not know the specification of an S-box and only know the algebraic degree of the S-box. However, many specific block ciphers usually use a public S-box and addition of secret sub keys, where an XOR is typically used for the addition. In this chapter, we show that the propagation characteristic of the division property can be improved if an S-box is a public function. The difference between Chap. 5 and this chapter is shown in Fig. 6.2.

We consider the propagation characteristic of the division property for the function shown in the right figure in Fig. 6.2. The key XORing is first applied, but it does not affect the division property because it is a linear function. Therefore, when we evaluate the propagation characteristic of the division property, we can remove the key XORing. Next, a public S-box is applied, and we can determine the ANF of the S-box. Assuming that an S-box is a function from n bits to m bits, the ANF is represented as

$$\begin{aligned} y[1] &= f_1(x[1], x[2], \dots, x[n]), \\ y[2] &= f_2(x[1], x[2], \dots, x[n]), \\ &\vdots \\ y[m] &= f_m(x[1], x[2], \dots, x[n]), \end{aligned}$$

where $x[i]$ $(1 \leq i \leq n)$ is an input, $y[j]$ $(1 \leq j \leq m)$ is an output, and f_j $(1 \leq j \leq m)$ is a Boolean function. The division property evaluates the input multiset and the output one by using the bit product function π_u, and we then divide the set of u into a subset whose evaluated value is 0 and a subset whose evaluated value becomes unknown. Namely, we evaluate the equation

$$F_u(x[1], x[2], \dots, x[n]) = \prod_{i=1}^{m} f_i(x[1], x[2], \dots, x[n])^{u[i]}$$

and divide the set of u. In Chap. 5, a fundamental property of the product of some functions is used, i.e., the algebraic degree of F_u is at most $w(u) \times d$ if the algebraic degree of functions f_i is at most d. However, since we now know the ANF of functions $f_1, f_2, \dots, f_m$, we can calculate the accurate algebraic degree of F_u for all $u \in \mathbb{F}_2^n$. In this case, if the algebraic degree of F_u is less than $w(u) \times d$ for all u for which $w(u)$ is constant, we can improve the propagation characteristic.

6.4.1 Application to MISTY S-boxes

Evaluation of S_7. The S_7 of MISTY is a 7-bit S-box with degree 3. We evaluate the property of $(\pi_v \circ S_7)$ to get the propagation characteristic of the division property. The algebraic degree of $(\pi_v \circ S_7)$ increases in accordance with the Hamming weight of v, and it is summarized as follows.

$w(v)$	0	1	2	3	4	5	6	7
degree	0	3	5	5	6	6	6	7

One can easily choose a modified S-box S_7' with algebraic degree 3, such that the algebraic degree of $(\pi_v \circ S_7')$ is at least 6 with $w(v) \geq 2$. However, for the S_7, the increment of the algebraic degree is bounded by 5 when $w(v) = 2$ or $w(v) = 3$ holds[3]. Then, $\bigoplus_{x \in \mathbb{X}} (\pi_v \circ S_7)(x)$ is 0 for $w(v) \leq 3$ if $\mathbb{X}$ has $\mathcal{D}_6^7$. It means that the necessary condition that $\bigoplus_{x \in \mathbb{X}} (\pi_v \circ S_7)(x)$ becomes unknown is $w(v) \geq 4$, and $\mathcal{D}_4^7$ is propagated from $\mathcal{D}_6^7$. Thus, the propagation characteristic is represented as the following.

[3] This observation was also provided by Theorem 3.1 in [BC13].

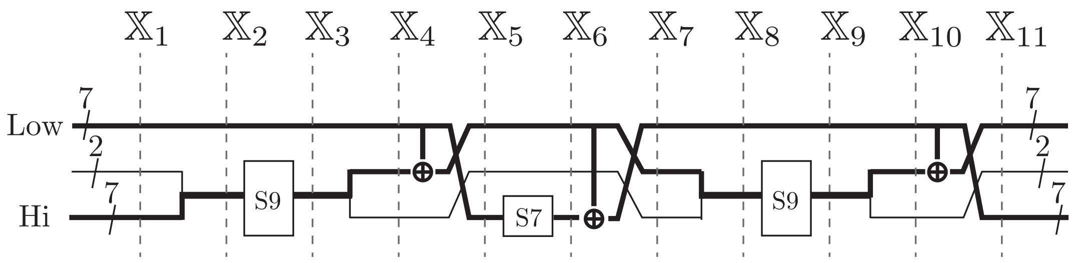


Figure 6.3: Structure of FI function

$\mathcal{D}^7_k$ for input set $\mathbb{X}$	$\mathcal{D}^7_0$	$\mathcal{D}^7_1$	$\mathcal{D}^7_2$	$\mathcal{D}^7_3$	$\mathcal{D}^7_4$	$\mathcal{D}^7_5$	$\mathcal{D}^7_6$	$\mathcal{D}^7_7$
$\mathcal{D}^7_k$ for output set $S(\mathbb{X})$	$\mathcal{D}^7_0$	$\mathcal{D}^7_1$	$\mathcal{D}^7_1$	$\mathcal{D}^7_1$	$\mathcal{D}^7_2$	$\mathcal{D}^7_2$	$\mathcal{D}^7_4$	$\mathcal{D}^7_7$

Note that all propagations except for $\mathcal{D}^7_6 \to \mathcal{D}^7_4$ are calculated by following Proposition 5.1. If the modified S-box is applied, the division property $\mathcal{D}^7_2$ is propagated from the division property $\mathcal{D}^7_6$ because of Proposition 5.1. Therefore, the deterioration of the division property for the S_7 is smaller than expected for a randomly chosen 7-bit S-box with algebraic degree 3.

Evaluation of S_9. The S_9 of MISTY is a 9-bit S-box with degree 2. We evaluate the property of $(\pi_v \circ S_9)$ to get the propagation characteristic of the division property. The algebraic degree of $(\pi_v \circ S_9)$ increases in accordance with the Hamming weight of v, and it is summarized as follows.

$w(v)$	0	1	2	3	4	5	6	7	8	9
degree	0	2	4	6	8	8	8	8	8	9

Thus, the propagation characteristic is represented as

$\mathcal{D}^9_k$ for input set $\mathbb{X}$	$\mathcal{D}^9_0$	$\mathcal{D}^9_1$	$\mathcal{D}^9_2$	$\mathcal{D}^9_3$	$\mathcal{D}^9_4$	$\mathcal{D}^9_5$	$\mathcal{D}^9_6$	$\mathcal{D}^9_7$	$\mathcal{D}^9_8$	$\mathcal{D}^9_9$
$\mathcal{D}^9_k$ for output set $S(\mathbb{X})$	$\mathcal{D}^9_0$	$\mathcal{D}^9_1$	$\mathcal{D}^9_1$	$\mathcal{D}^9_2$	$\mathcal{D}^9_2$	$\mathcal{D}^9_3$	$\mathcal{D}^9_3$	$\mathcal{D}^9_4$	$\mathcal{D}^9_4$	$\mathcal{D}^9_9$

Unlike the propagation characteristic of the division property for S_7, the one for S_9 is essentially optimal among 9-bit S-boxes with algebraic degree 2.

6.5 New Integral Characteristic Based on Division Property

This section shows how to create integral characteristics for MISTY1 by using the propagation characteristic of the division property. We first evaluate the propagation characteristic for the component functions of MISTY1, i.e., the FI function, the FO function, and the FL layer. Finally, by assembling these characteristics, we devise an algorithm to search for integral characteristics on MISTY1.

6.5.1 Division Property for FI function

We evaluate the propagation characteristic of the division property for the FI function by using those for MISTY S-boxes shown in Sect. 6.4.1. Since there are a zero-extended XOR and a truncated XOR in the FI function, we use a new representation, in which the internal state is expressed as two 7-bit values and one 2-bit value. Figure 6.3 shows the structure of the FI function with our representation, where we remove the XOR of sub keys because it does not affect the division property.

Let $\mathbb{X}_1$ be the input multiset of the FI function. We define every multiset $\mathbb{X}_2, \mathbb{X}_3, \ldots, \mathbb{X}_{11}$ in Fig. 6.3. Here, elements of the multiset $\mathbb{X}_1$, $\mathbb{X}_5$, $\mathbb{X}_6$, and $\mathbb{X}_{11}$ take a value of $(\mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7)$. Elements of the multiset $\mathbb{X}_2$, $\mathbb{X}_3$, $\mathbb{X}_8$, and $\mathbb{X}_9$ take a value of $(\mathbb{F}_2^9 \times \mathbb{F}_2^7)$. Elements of the multiset $\mathbb{X}_4$, $\mathbb{X}_7$, and $\mathbb{X}_{10}$ take a value of $(\mathbb{F}_2^2 \times \mathbb{F}_2^7 \times \mathbb{F}_2^7)$. Since elements of $\mathbb{X}_1$ and $\mathbb{X}_{11}$ take a value of $(\mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7)$, the propagation for the FI function is calculated on $\mathcal{D}_{\mathbb{K}}^{7,2,7}$. Here, the propagation is calculated with the following steps.

From $\mathbb{X}_1$ to $\mathbb{X}_2$: A 9-bit value is created by concatenating the first 7-bit value with the second 2-bit value. The propagation characteristic can be evaluated by using Proposition 5.6.

From $\mathbb{X}_2$ to $\mathbb{X}_3$: The 9-bit S-box S_9 is applied to the first 9-bit value. The propagation characteristic can be evaluated by using the table shown in Sect. 6.4.1.

From $\mathbb{X}_3$ to $\mathbb{X}_4$: The 9-bit output value is split into a 2-bit value and a 7-bit value. The propagation characteristic can be evaluated by using Proposition 5.5.

From $\mathbb{X}_4$ to $\mathbb{X}_5$: The second 7-bit value is XORed with the last 7-bit value, and then, the order is rotated. The propagation characteristic can be evaluated by using Proposition 5.3 and 5.4.

From $\mathbb{X}_5$ to $\mathbb{X}_6$: The 7-bit S-box S_7 is applied to the first 7-bit value. The propagation characteristic can be evaluated by using the table shown in Sect. 6.4.1.

From $\mathbb{X}_6$ to $\mathbb{X}_7$: The first 7-bit value is XORed with the last 7-bit value, and then, the order is rotated. The propagation characteristic can be evaluated by using Proposition 5.3 and 5.4.

From $\mathbb{X}_7$ to $\mathbb{X}_8$: A 9-bit value is created by concatenating the first 2-bit value with the second 7-bit value. The propagation characteristic can be evaluated by using Proposition 5.6.

From $\mathbb{X}_8$ to $\mathbb{X}_{11}$: The propagation characteristic is the same as that from $\mathbb{X}_2$ to $\mathbb{X}_5$.

As an example, we show the propagation characteristic when $\mathbb{X}_1$ has the division property $\mathcal{D}^{7,2,7}_{\{[4,2,6]\}}$.

Example 6.1 (Propagation from $\mathcal{D}^{7,2,7}_{[4,2,6]}$ for FI function)**.** *We consider the propagation characteristic of the division property for the FI function (see Fig. 6.3). Assume that $\mathbb{X}_1$ has the division property $\mathcal{D}^{7,2,7}_{\{[4,2,6]\}}$.*

From $\mathbb{X}_1$ to $\mathbb{X}_2$: *Since the first 7-bit value and the second 2-bit value are concatenated, Rule 5 is applied. Thus, the multiset $\mathbb{X}_2$ has the division property $\mathcal{D}^{9,7}_{\{[6,6]\}}$.*

From $\mathbb{X}_2$ to $\mathbb{X}_3$: *The 9-bit S-box S_9 is applied. Thus, the multiset $\mathbb{X}_3$ has the division property $\mathcal{D}^{9,7}_{\{[3,6]\}}$.*

From $\mathbb{X}_3$ to $\mathbb{X}_4$: *Since the first 9-bit value are split to 2-bit and 7-bit values, Rule 4 is applied. Thus, the multiset $\mathbb{X}_4$ has the division property $\mathcal{D}^{2,7,7}_{\{[0,3,6],[1,2,6],[2,1,6]\}}$.*

From $\mathbb{X}_4$ to $\mathbb{X}_5$: *Since the second 7-bit value is XORed with the last 7-bit value, Rule 2 and Rule 3 are applied. In this case, the propagation of the division property is calculated as*

$$[0,3,6] \Rightarrow [0,3,6],[0,4,5],[0,5,4],[0,6,3],[0,7,2],$$
$$[1,2,6] \Rightarrow [1,2,6],[1,3,5],[1,4,4],[1,5,3],[1,6,2],[1,7,1],$$
$$[2,1,6] \Rightarrow [2,1,6],[2,2,5],[2,3,4],[2,4,3],[2,5,2],[2,6,1],[2,7,0].$$

The position is rotated, and then the division property of $\mathbb{X}_5$ has $\mathcal{D}^{7,2,7}_{\mathbb{K}}$, where $\mathbb{K}$ has 18 vectors as

$$[6,0,3],[5,0,4],[4,0,5],[3,0,6],[2,0,7],$$
$$[6,1,2],[5,1,3],[4,1,4],[3,1,5],[2,1,6],[1,1,7],$$
$$[6,2,1],[5,2,2],[4,2,3],[3,2,4],[2,2,5],[1,2,6],[0,2,7].$$

From $\mathbb{X}_5$ to $\mathbb{X}_6$: *The 7-bit S-box S_7 is applied. Here, we exploit the vulnerable property of S_7. Thus, the following 18 vectors*

$$[4,0,3],[2,0,4],[2,0,5],[1,0,6],[1,0,7],$$
$$[4,1,2],[2,1,3],[2,1,4],[1,1,5],[1,1,6],[1,1,7],$$
$$[4,2,1],[2,2,2],[2,2,3],[1,2,4],[1,2,5],[1,2,6],[0,2,7],$$

are calculated. For example, the vector $[2,0,5]$ *is removed because* $[2,0,5] \succ [2,0,4]$*. Similarly, after removing redundant vectors, and the division property of* $\mathbb{X}_6$ *has* $\mathcal{D}_{\mathbb{K}}^{7,2,7}$*, where* $\mathbb{K}$ *has* 10 *vectors as*

$$[0,2,7],[1,0,6],[1,1,5],[1,2,4],[2,0,4],$$
$$[2,1,3],[2,2,2],[4,0,3],[4,1,2],[4,2,1].$$

From $\mathbb{X}_6$ **to** $\mathbb{X}_7$ **:** *Since the first* 7*-bit value is XORed with the last* 7*-bit value, Rule 2 and Rule 3 are applied. In this case, the propagation of the division property is calculated as*

$$\begin{aligned}
&[0,2,7] \Rightarrow [0,2,7],[1,2,6],[2,2,5],[3,2,4],[4,2,3],[5,2,2],[6,2,1],[7,2,0],\\
&[1,0,6] \Rightarrow [1,0,6],[2,0,5],[3,0,4],[4,0,3],[5,0,2],[6,0,1],[7,0,0],\\
&[1,1,5] \Rightarrow [1,1,5],[2,1,4],[3,1,3],[4,1,2],[5,1,1],[6,1,0],\\
&[1,2,4] \Rightarrow [1,2,4],[2,2,3],[3,2,2],[4,2,1],[5,2,0],\\
&[2,0,4] \Rightarrow [2,0,4],[3,0,3],[4,0,2],[5,0,1],[6,0,0],\\
&[2,1,3] \Rightarrow [2,1,3],[3,1,2],[4,1,1],[5,1,0],\\
&[2,2,2] \Rightarrow [2,2,2],[3,2,1],[4,2,0],\\
&[4,0,3] \Rightarrow [4,0,3],[5,0,2],[6,0,1],[7,0,0],\\
&[4,1,2] \Rightarrow [4,1,2],[5,1,1],[6,1,0],\\
&[4,2,1] \Rightarrow [4,2,1],[5,2,0].
\end{aligned}$$

After removing redundant vectors, the position is rotated, and then the division property of $\mathbb{X}_7$ *has* $\mathcal{D}_{\mathbb{K}}^{2,7,7}$*, where* $\mathbb{K}$ *has* 16 *vectors as*

$$[0,0,6],[0,1,5],[0,2,4],[0,3,3],[0,4,2],[0,6,1],[1,0,5],[1,1,4],$$
$$[1,2,3],[1,3,2],[1,5,1],[2,0,4],[2,1,3],[2,2,2],[2,4,1],[2,7,0].$$

From $\mathbb{X}_7$ **to** $\mathbb{X}_8$ **:** *Since the first* 2*-bit value and the second* 7*-bit value are concatenated, Rule 5 is applied. Then, the following* 16 *vectors*

$$[0,6],[1,5],[2,4],[3,3],[4,2],[6,1],[1,5],[2,4],$$
$$[3,3],[4,2],[6,1],[2,4],[3,3],[4,2],[6,1],[9,0],$$

are calculated. After removing redundant vectors, the division property of $\mathbb{X}_8$ *has* $\mathcal{D}_{\mathbb{K}}^{9,7}$*, where* $\mathbb{K}$ *has* 7 *vectors as*

$$[0,6],[1,5],[2,4],[3,3],[4,2],[6,1],[9,0].$$

From $\mathbb{X}_8$ **to** $\mathbb{X}_9$ **:** *The* 9*-bit S-box* S_9 *is applied. Then, the following* 7 *vectors*

$$[0,6],[1,5],[1,4],[2,3],[2,2],[3,1],[9,0],$$

are calculated. After removing redundant vectors, the division property of $\mathbb{X}_9$ *has* $\mathcal{D}_{\mathbb{K}}^{9,7}$*, where* $\mathbb{K}$ *has* 5 *vectors as*

$$[0,6],[1,4],[2,2],[3,1],[9,0].$$

From $\mathbb{X}_9$ **to** $\mathbb{X}_{10}$ **:** *Since the first* 9*-bit value are split to* 2*-bit and* 7*-bit values, Rule 4 is applied. Thus, the multiset* $\mathbb{X}_{10}$ *has the division property* $\mathcal{D}_{\mathbb{K}}^{2,7,7}$*, where* $\mathbb{K}$ *has* 10 *vectors as*

$$\begin{aligned}
&[0,6] \Rightarrow [0,0,6],\\
&[1,4] \Rightarrow [0,1,4],[1,0,4],\\
&[2,2] \Rightarrow [0,2,2],[1,1,2],[2,0,2],\\
&[3,1] \Rightarrow [0,3,1],[1,2,1],[2,1,1],\\
&[9,0] \Rightarrow [2,7,0].
\end{aligned}$$

Algorithm 4 Propagation for FI function

```
 1: procedure FIEval($k_1, k_2, k_3$)
 2:     $\mathbb{K} \Leftarrow$ S9Eval($\vec{k}$)                    ▷ $\mathbb{X}_1 \to \mathbb{X}_5$
 3:     $\mathbb{K}' \Leftarrow$ S7Eval($\mathbb{K}$)                  ▷ $\mathbb{X}_5 \to \mathbb{X}_7$
 4:     $\mathbb{K}'' \Leftarrow$ S9Eval($\mathbb{K}'$)                ▷ $\mathbb{X}_7 \to \mathbb{X}_{11}$
 5:     return $\mathbb{K}''$
 6: end procedure
```

```
 1: procedure S9Eval($\mathbb{K}$)
 2:     $\mathbb{K}' \Leftarrow \phi$
 3:     for all $\vec{k} \in \mathbb{K}$ do
 4:         $[\ell, c, r] \Leftarrow [k_1, k_2, k_3]$
 5:         $k \Leftarrow \ell + c$
 6:         if $k < 9$ then
 7:             $k \Leftarrow \lceil k/2 \rceil$
 8:         end if
 9:         for $c' \Leftarrow 0$ to $\min(2, k)$ do
10:             for $x \Leftarrow 0$ to $r$ do
11:                 $\ell' \Leftarrow r - x$
12:                 $r' \Leftarrow k - c' + x$
13:                 if $r' \leq 7$ then
14:                     $\mathbb{K}' \Leftarrow \mathbb{K}' \cup [\ell', c', r']$
15:                 end if
16:             end for
17:         end for
18:     end for
19:     return SizeReduce($\mathbb{K}'$)
20: end procedure
21: procedure S7Eval($\mathbb{K}$)
22:     $\mathbb{K}' \Leftarrow \phi$
23:     for all $\vec{k} \in \mathbb{K}$ do
24:         $[\ell, c, r] \Leftarrow [k_1, k_2, k_3]$
25:         $k \Leftarrow \ell$
26:         if $k = 6$ then
27:             $k \Leftarrow 4$
28:         else if $k < 6$ then
29:             $k \Leftarrow \lceil k/3 \rceil$
30:         end if
31:         for $x \Leftarrow 0$ to $r$ do
32:             $\ell' \Leftarrow c$
33:             $c' \Leftarrow r - x$
34:             $r' \Leftarrow k + x$
35:             if $r' \leq 7$ then
36:                 $\mathbb{K}' \Leftarrow \mathbb{K}' \cup [\ell', c', r']$
37:             end if
38:         end for
39:     end for
40:     return SizeReduce($\mathbb{K}'$)
41: end procedure
```

From $\mathbb{X}_{10}$ to $\mathbb{X}_{11}$: *Since the second 7-bit value is XORed with the last 7-bit value, Rule 2 and Rule 3 are applied. In this case, the propagation of the division property is calculated as*

$$\begin{aligned}
&[0,0,6] \Rightarrow [0,0,6],[0,1,5],[0,2,4],[0,3,3],[0,4,2],[0,5,1],[0,6,0],\\
&[0,1,4] \Rightarrow [0,1,4],[0,2,3],[0,3,2],[0,4,1],[0,5,0],\\
&[1,0,4] \Rightarrow [1,0,4],[1,1,3],[1,2,2],[1,3,1],[1,4,0],\\
&[0,2,2] \Rightarrow [0,2,2],[0,3,1],[0,4,0],\\
&[1,1,2] \Rightarrow [1,1,2],[1,2,1],[1,3,0],\\
&[2,0,2] \Rightarrow [2,0,2],[2,1,1],[2,2,0],\\
&[0,3,1] \Rightarrow [0,3,1],[0,4,0],\\
&[1,2,1] \Rightarrow [1,2,1],[1,3,0],\\
&[2,1,1] \Rightarrow [2,1,1],[2,2,0],\\
&[2,7,0] \Rightarrow [2,7,0].
\end{aligned}$$

After removing redundant vectors, the position is rotated, and then the division property of $\mathbb{X}_{11}$ has $\mathcal{D}_{\mathbb{K}}^{7,2,7}$, where $\mathbb{K}$ has 12 vectors as

$$\begin{aligned}
&[0,0,4],[0,1,3],[0,2,2],[1,0,3],[1,1,2],[1,2,1],\\
&[2,0,2],[2,1,1],[2,2,0],[4,0,1],[4,1,0],[6,0,0].
\end{aligned}$$

Algorithm 4 creates the propagation characteristic table for the FI function. It calls SizeReduce($\mathbb{K}$), where redundant vectors are eliminated, i.e., it eliminates $\vec{k} \in \mathbb{K}$ if there exists $\vec{k}' \in \mathbb{K}$ satisfying $\vec{k} \succeq \vec{k}'$. Algorithm 4 only creates the propagation characteristic table for which the input property is represented by $\mathcal{D}_{\{\vec{k}\}}^{7,2,7}$. If any input multiset is evaluated, we need to know the propagation characteristic from $\mathcal{D}_{\mathbb{K}}^{7,2,7}$ with $|\mathbb{K}| \geq 2$. However, we do not need to evaluate such propagation in advance because it can be easily evaluated by the table for which the input property is represented by $\mathcal{D}_{\{\vec{k}\}}^{7,2,7}$. For example, we consider the propagation characteristic from $\mathcal{D}_{\{\vec{k},\vec{k}'\}}^{7,2,7}$ to $\mathcal{D}_{\mathbb{K}}^{7,2,7}$. We first get $\mathbb{K}_1$ and $\mathbb{K}_2$ from the propagation characteristic tables for $\mathcal{D}_{\{\vec{k}\}}^{7,2,7}$ and $\mathcal{D}_{\{\vec{k}'\}}^{7,2,7}$, respectively. Then, $\mathbb{K}$ is calculated as $\mathbb{K} = \mathbb{K}_1 \cup \mathbb{K}_2$.

We show all propagation characteristic tables for the *FI* function in Appendix B. Here, the propagation table from $\vec{k}$ to $\mathbb{K}$ is generated, and the number of entries of this table is $8 \cdot 3 \cdot 8 = 192$. Moreover, we experimentally evaluated the propagation characteristic for the *FI* function. In our experimental search, for any $\mathcal{D}_{\{[k_1,k_2,k_3]\}}^{7,2,7}$, we created 100 random input multisets and then evaluated the propagation characteristic. As a result, we confirmed that the experimental propagation characteristics are the same as the theoretical ones shown in Appendix B.

6.5.2 Division Property for *FO* function

Algorithm 5 Propagation for *FO* function

```
1:  procedure FOEval($k_1, k_2, k_3, k_4, k_5, k_6$)
2:      $\mathbb{K} \Leftarrow$ FORound($\vec{k}$)
3:      $\mathbb{K}' \Leftarrow$ FORound($\mathbb{K}$)
4:      $\mathbb{K}'' \Leftarrow$ FORound($\mathbb{K}'$)
5:      return $\mathbb{K}''$
6:  end procedure
1:  procedure FORound($\mathbb{K}$)
2:      $\mathbb{K}' \Leftarrow \phi$
3:      for all $\vec{k} \in \mathbb{K}$ do
4:          $\mathbb{Y} \Leftarrow$ FIEval($k_1, k_2, k_3$)
5:          for all $\vec{y} \in \mathbb{Y}$ do
6:              for all $\vec{x}$ s.t. $(x_1 \leq k_4) \wedge (x_2 \leq k_5) \wedge (x_3 \leq k_6)$ do
7:                  $\vec{k'} \Leftarrow [k_4 - x_1, k_5 - x_2, k_6 - x_3, y_1 + x_1, y_2 + x_2, y_3 + x_3]$
8:                  if $(k'_4 \leq 7) \wedge (k'_5 \leq 2) \wedge (k'_6 \leq 7)$ then
9:                      $\mathbb{K}' \Leftarrow \mathbb{K}' \cup \vec{k'}$
10:                 end if
11:             end for
12:         end for
13:     end for
14:     return SizeReduce($\mathbb{K}'$)
15: end procedure
```

We next evaluate the propagation characteristic of the division property for the *FO* function by using the propagation characteristic table of the *FI* function. Here, we remove the XOR of sub keys because it does not affect the division property. The input and output of the *FO* function take the value of $(\mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7 \times \mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7)$. Therefore, the propagation for the *FO* function is calculated on $\mathcal{D}_{\mathbb{K}}^{7,2,7,7,2,7}$. Similar to the one created for the *FI* function, we create the propagation characteristic table for the *FO* function (see Algorithm 5). We create only a table for which the input property is represented by $\mathcal{D}_{\{\vec{k}\}}^{7,2,7,7,2,7}$ and the output property is represented by $\mathcal{D}_{\mathbb{K}}^{7,2,7,7,2,7}$. Here, the propagation table from $\vec{k}$ to $\mathbb{K}$ is generated, and the number of entries of this table is $8 \cdot 3 \cdot 8 \cdot 8 \cdot 3 \cdot 8 = 36864$. As an example, the propagation characteristic table from $\mathcal{D}_{\{[1,1,2,3,1,5]\}}^{7,2,7,7,2,7}$ is shown in Table 6.2.

6.5.3 Division Property for FL Layer

MISTY1 has the FL layer, which consists of two *FL* functions and is applied once every two rounds. In the *FL* function, the right half of the input is XORed with the AND between the left half and a sub key $KL_{i,1}$. Then, the left half of the input is XORed with the OR between the right half and a sub key $KL_{i,2}$.

Table 6.2: Division property of input is $\mathcal{D}^{7,2,7,7,2,7}_{\{[1,1,2,3,1,5]\}}$

$\vec{k}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\{\vec{k}\}}$	$\mathbb{K}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\mathbb{K}}$
[1 1 2 3 1 5]	[0 0 0 0 0 4] [0 0 0 0 1 3] [0 0 0 0 2 2] [0 0 0 1 0 3] [0 0 0 1 1 2] [0 0 0 1 2 1]
	[0 0 0 2 0 2] [0 0 0 2 1 1] [0 0 0 2 2 0] [0 0 0 3 0 1] [0 0 0 3 1 0] [0 0 0 5 0 0]
	[0 0 1 0 0 3] [0 0 1 0 1 2] [0 0 1 0 2 1] [0 0 1 1 0 2] [0 0 1 1 1 1] [0 0 1 1 2 0]
	[0 0 1 2 0 1] [0 0 1 2 1 0] [0 0 1 3 0 0] [0 0 2 0 0 2] [0 0 2 0 1 1] [0 0 2 0 2 0]
	[0 0 2 1 0 1] [0 0 2 1 1 0] [0 0 2 2 0 0] [0 0 3 0 0 1] [0 0 3 0 1 0] [0 0 3 1 0 0]
	[0 0 5 0 0 0] [0 1 0 0 0 3] [0 1 0 0 1 2] [0 1 0 0 2 1] [0 1 0 1 0 2] [0 1 0 1 1 1]
	[0 1 0 1 2 0] [0 1 0 2 0 1] [0 1 0 2 1 0] [0 1 0 3 0 0] [0 1 1 0 0 2] [0 1 1 0 1 1]
	[0 1 1 0 2 0] [0 1 1 1 0 1] [0 1 1 1 1 0] [0 1 1 2 0 0] [0 1 2 0 0 1] [0 1 2 0 1 0]
	[0 1 2 1 0 0] [0 1 4 0 0 0] [0 2 0 0 0 2] [0 2 0 0 1 1] [0 2 0 0 2 0] [0 2 0 1 0 1]
	[0 2 0 1 1 0] [0 2 0 2 0 0] [0 2 1 0 0 1] [0 2 1 0 1 0] [0 2 1 1 0 0] [0 2 3 0 0 0]
	[1 0 0 0 0 3] [1 0 0 0 1 2] [1 0 0 0 2 1] [1 0 0 1 0 2] [1 0 0 1 1 1] [1 0 0 1 2 0]
	[1 0 0 2 0 1] [1 0 0 2 1 0] [1 0 0 4 0 0] [1 0 1 0 0 2] [1 0 1 0 1 1] [1 0 1 0 2 0]
	[1 0 1 1 0 1] [1 0 1 1 1 0] [1 0 1 2 0 0] [1 0 2 0 0 1] [1 0 2 0 1 0] [1 0 2 1 0 0]
	[1 0 4 0 0 0] [1 1 0 0 0 2] [1 1 0 0 1 1] [1 1 0 0 2 0] [1 1 0 1 0 1] [1 1 0 1 1 0]
	[1 1 0 2 0 0] [1 1 1 0 0 1] [1 1 1 0 1 0] [1 1 1 1 0 0] [1 1 3 0 0 0] [1 2 0 0 0 1]
	[1 2 0 0 1 0] [1 2 0 1 0 0] [1 2 2 0 0 0] [2 0 0 0 0 2] [2 0 0 0 1 1] [2 0 0 0 2 0]
	[2 0 0 1 0 1] [2 0 0 1 1 0] [2 0 0 3 0 0] [2 0 1 0 0 1] [2 0 1 0 1 0] [2 0 1 1 0 0]
	[2 0 3 0 0 0] [2 1 0 0 0 1] [2 1 0 0 1 0] [2 1 0 1 0 0] [2 1 2 0 0 0] [2 2 1 0 0 0]
	[3 0 0 0 0 1] [3 0 0 0 1 0] [3 0 0 2 0 0] [3 0 2 0 0 0] [3 1 1 0 0 0] [3 2 0 0 0 0]
	[4 0 0 1 0 0] [4 0 1 0 0 0] [4 1 0 0 0 0] [6 0 0 0 0 0]

Since the input and output of the FL function take the value of $(\mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7 \times \mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7)$, the propagation for the FL function is calculated on $\mathcal{D}^{7,2,7,7,2,7}_{\mathbb{K}}$. `FLEval` in Algorithm 6 calculates the propagation characteristic table for the FL function. Here, the propagation table from $\vec{k}$ to $\mathbb{K}$ is generated, and the number of entries of this table is $8 \cdot 3 \cdot 8 \cdot 8 \cdot 3 \cdot 8 = 36864$. Moreover, the FL layer consists of two FL functions. Therefore, we have to consider the propagation characteristic of the division property $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\{\vec{k}\}}$, where each FL function is applied to the left and right halves. `FLLayerEval` in Algorithm 6 calculates the propagation characteristic of the division property for the FL layer.

6.5.4 New Path Search for Integral Characteristics on MISTY1

We created the propagation characteristic table for the FI and FO functions in Sect. 6.5.1 and 6.5.2, respectively. Moreover, we showed the propagation characteristic for the FL layer in Sect. 6.5.3. By assembling these propagation characteristics, we devise an algorithm to search for integral characteristics on MISTY1. Since the input and output are represented as eight 7-bit values and four 2-bit values, the propagation is calculated on $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\mathbb{K}}$.

The FL layer is first applied to plaintexts, and it deteriorates the propagation of the division property. Therefore, we first remove only the first FL layer and search for integral characteristics on MISTY1 without the first FL layer. The method for passing through the first FL layer is shown in the next section. Algorithm 7 shows the search algorithm for integral characteristics on MISTY1 without the first FL layer.

As a result, we find 6-round integral characteristics without the first and last FL layers by using Algorithm 7. Each characteristic uses 2^{63} chosen plaintexts, where any one bit of the first seven bits is constant and the others take all values. Then, such input has the division property $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\{[6,2,7,7,2,7,7,2,7,7,2,7]\}}$. Therefore, we use $\vec{k} = [6,2,7,7,2,7,7,2,7,7,2,7]$ as the input of Algorithm 7. We perfectly execute `SizeReduce` every round, and Table 6.3 shows the propagation of $\mathbb{K}$, where $\min_w(\mathbb{K})$ and $\max_w(\mathbb{K})$ are calculated as

$$\min_w(\mathbb{K}) = \min_{\vec{k} \in \mathbb{K}} \left\{ \sum_{i=1}^{12} k_i \right\}, \quad \max_w(\mathbb{K}) = \max_{\vec{k} \in \mathbb{K}} \left\{ \sum_{i=1}^{12} k_i \right\}.$$

After the 6th round function, we have 131 vectors, which are listed in Appendix A.

Algorithm 6 Propagation for FL layer

```
 1: procedure FLLayerEval($\mathbb{K}$)
 2:     $\mathbb{K}' \Leftarrow \phi$
 3:     for all $\vec{k} \in \mathbb{K}$ do
 4:         $\mathbb{L} \Leftarrow$ FlEval$(k_1, k_2, \ldots, k_6)$
 5:         $\mathbb{R} \Leftarrow$ FlEval$(k_7, k_8, \ldots, k_{12})$
 6:         for all $\vec{\ell} \in \mathbb{L}$ do
 7:             for all $\vec{r} \in \mathbb{R}$ do
 8:                 $\mathbb{K}' \Leftarrow \mathbb{K}' \cup [\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6, r_1, r_2, r_3, r_4, r_5, r_6]$
 9:             end for
10:         end for
11:     end for
12:     return $\mathbb{K}'$
13: end procedure
```

```
 1: procedure FLEval$(k_1, k_2, \ldots, k_6)$
 2:     $\mathbb{K}' \Leftarrow \phi$
 3:     $[\ell, c, r] \Leftarrow [k_1 + k_4, k_2 + k_5, k_3 + k_6]$
 4:     for $k_1' \Leftarrow 0$ to $\min(7, \ell)$ do
 5:         for $k_2' \Leftarrow 0$ to $\min(2, c)$ do
 6:             for $k_3' \Leftarrow 0$ to $\min(7, r)$ do
 7:                 $(k_4', k_5', k_6') \Leftarrow (\ell - k_1', c - k_2', r - k_3')$
 8:                 if $(k_4' \leq 7) \wedge (k_5' \leq 2) \wedge (k_6' \leq 7)$ then
 9:                     $\mathbb{K}' \Leftarrow \mathbb{K}' \cup [k_1', k_2', k_3', k_4', k_5', k_6']$
10:                 end if
11:             end for
12:         end for
13:     end for
14:     return SizeReduce$(\mathbb{K}')$
15: end procedure
```

Table 6.3: Propagation from $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\{[6,2,7,7,2,7,7,2,7,7,2,7]\}}$

#rounds	0 (plaintexts)	1	2	FL	3	4	FL	5	6
$\lvert\mathbb{K}\rvert$	1	1	9	16	2596	2617429	12268480	58962	131
$\max_w(\mathbb{K})$	63	63	63	63	62	55	47	27	8
$\min_w(\mathbb{K})$	63	63	61	61	43	19	19	4	1

Assume that $\mathbb{X}$ has the division property $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\mathbb{K}}$. Let $e_i \in \mathbb{Z}^{12}$ be a unit vector whose ith element is one and the others are zero. When there do not exist e_i in $\mathbb{K}$, $\bigoplus_{\vec{x} \in \mathbb{X}} x_i = 0$. Since the vector $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ is not included in 131 vectors, we are certain that the first 7 bits are balanced. Our algorithm is written by C++, and the execution time is about one day with Core i7-4770 Processor (4 cores) in 16 GB RAM. Figure 6.4 shows the 6-round integral characteristic, where the bit strings labeled B, i.e., the first 7 bits and last 32 bits, are balanced. Note that the 6-round characteristic becomes a 7-round characteristic if the FL layer after the 6th round function is removed. Compared with the previous 4-round characteristic [HTK04, TSSK08], our characteristic is improved by two rounds.

As shown in Sect. 6.4, the S_7 of MISTY1 has the vulnerable property that $\mathcal{D}^7_4$ is provided from $\mathcal{D}^7_6$. Interestingly, assuming that S_7 does not have this property (changing lines 26–30 in `S7Eval`), our algorithm cannot construct the 6-round characteristic.

6.5.5 Revisiting Known Characteristic for MISTY1

It was already shown in [THK99] that reduced MISTY1 has a 14th order differential characteristic, and the principle was also discussed in [BF00, CV02]. In the 14th order differential characteristic, 14 bits $P^R[10, \ldots, 16, 26, \ldots, 32]$ are active and the others are constant. Then, the first seven bits of X^R_5 are balanced. We evaluate the principle of the characteristic by using

Algorithm 7 Path search for r-round characteristics without first FL layer

```
1: procedure Misty1Eval($k_1, k_2, \dots, k_{12}, r$)
2:     $\mathbb{K} \Leftarrow$ RoundFuncEval($\vec{k}$)                    ▷ 1st round
3:     for $i = 1$ to $r$ do
4:         if $i$ is even then
5:             $\mathbb{K} \Leftarrow$ FlLayerEval($\mathbb{K}$)             ▷ FL Layer
6:         end if
7:         $\mathbb{K} \Leftarrow$ RoundFuncEval($\mathbb{K}$)               ▷ (i+1)th round
8:     end for
9:     return $\mathbb{K}$
10: end procedure
1: procedure RoundFuncEval($\mathbb{K}$)
2:     $\mathbb{K}' \Leftarrow \phi$
3:     for all $\vec{k} \in \mathbb{K}$ do
4:         for all $\vec{x}$ s.t. $x_j \leq k_j$ for all $j = 1, 2, \dots, 6$ do
5:             $[r_1, r_2, r_3] \Leftarrow [k_1 - x_1, k_2 - x_2, k_3 - x_3]$
6:             $[r_4, r_5, r_6] \Leftarrow [k_4 - x_4, k_5 - x_5, k_6 - x_6]$
7:             $\mathbb{Y} \Leftarrow$ FOEval($x_1, x_2, x_3, x_4, x_5, x_6$)
8:             for all $\vec{y} \in \mathbb{Y}$ do
9:                 $[\ell_1, \ell_2, \ell_3] \Leftarrow [k_7 + y_1, k_8 + y_2, k_9 + y_3]$
10:                $[\ell_4, \ell_5, \ell_6] \Leftarrow [k_{10} + y_4, k_{11} + y_5, k_{12} + y_6]$
11:                if $\ell_{j'} \leq 7$ for $j' \in \{1, 3, 4, 6\}$ and $\ell_{j'} \leq 2$ for $j' \in \{2, 5\}$ then
12:                    $\mathbb{K}' \Leftarrow \mathbb{K}' \cup [\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6, r_1, r_2, r_3, r_4, r_5, r_6]$
13:                end if
14:            end for
15:        end for
16:    end for
17:    return SizeReduce($\mathbb{K}'$)
18: end procedure
```

Table 6.4: Propagation from $\mathcal{D}^{0,0,0,0,0,0,0,0,7,0,0,7}_{\{[6,2,7,7,2,7,7,2,7,7,2,7]\}}$

#rounds	0 (plaintexts)	1	2	FL	3	4	FL
$\lvert\mathbb{K}\rvert$	1	1	460	400	125	12	12
$\max_w(\mathbb{K})$	14	14	14	14	4	2	1
$\min_w(\mathbb{K})$	14	14	4	4	1	1	1

the propagation characteristic of the division property. We search for the integral characteristics by using Algorithm 7 with perfect SizeReduce. We use $\vec{k} = [0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 7]$ as the input of Algorithm 7, and Table 6.4 shows the propagation of $\mathbb{K}$. The output of the 4th round function has the division property $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\mathbb{K}}$, where $\mathbb{K}$ has 12 vectors as follows:

$$\begin{array}{lll}
[1,0,0,0,0,0,0,0,0,0,0,0] & [0,1,0,0,0,0,0,0,0,0,0,0] & [0,0,1,0,0,0,0,0,0,0,0,0] \\
[0,0,0,1,0,0,0,0,0,0,0,0] & [0,0,0,0,1,0,0,0,0,0,0,0] & [0,0,0,0,0,1,0,0,0,0,0,0] \\
[0,0,0,0,0,0,2,0,0,0,0,0] & [0,0,0,0,0,0,0,1,0,0,0,0] & [0,0,0,0,0,0,0,0,1,0,0,0] \\
[0,0,0,0,0,0,0,0,0,1,0,0] & [0,0,0,0,0,0,0,0,0,0,1,0] & [0,0,0,0,0,0,0,0,0,0,0,1]
\end{array}$$

This result implies the existence of a 14th order differential characteristic, where the left seven bits of X^R_5 are balanced.

The 14th order differential characteristic is extended to a 46th order differential characteristic, where 14 bits $P^L[10, \dots, 16, 26, \dots, 32]$ and 32 bits P^R are active and the others are constant. Then, the first seven bits of X^L_5 are balanced. We also revisit the 46th order differential characteristic. Namely, we evaluate the propagation characteristic of the division property, where the input set has the division property $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\{[0,0,7,0,0,7,7,2,7,7,2,7]\}}$. As a result, we can get an integral characteristic that the first 16 bits of X^L_5 are balanced. In the simple extension shown in [HTK04] and [TSSK08], only the first 7 bits are balanced. Thus, our method proves that the number of balanced bits is extended from 7 bits to 16 bits.

6.5.6 Optimized Algorithm

If we execute `SizeReduce` perfectly, it requires $O(|\mathbb{K}|^2)$ time complexity, and the execution time of Algorithm 7 is increased. Therefore, we use a more reasonable method.

Let $\mathcal{D}_{\mathbb{K}}$ be any division property, where $\mathbb{K}$ contains redundant vectors. Moreover, by executing `SizeReduce`, we get $\mathbb{K}'$ from $\mathbb{K}$. Then, as shown in Sect. 6.3.3, the unknown set indicated by $\mathcal{D}_{\mathbb{K}}$ is the same as that by $\mathcal{D}_{\mathbb{K}'}$. Namely, the result of Algorithm 7 does not change even if we do not perform `SizeReduce` perfectly. Therefore, we execute a rough `SizeReduce` which performs faster. The rough `SizeReduce` first sorts every vector in $\mathbb{K}$ by using lexicographic order and obtains the following $|\mathbb{K}|$ vectors,

$$\vec{k}^{(1)}, \vec{k}^{(2)}, \dots, \vec{k}^{(|\mathbb{K}|)}.$$

Then, there is no $(\vec{k}^{(i)}, \vec{k}^{(j)})$ satisfying $\vec{k}^{(i)} \succeq \vec{k}^{(j)}$ such that $i < j$. We initialize two indices, $i = 1$ and $j = 2$, and evaluate whether or not $\vec{k}^{(j)} \succeq \vec{k}^{(i)}$. If $\vec{k}^{(j)} \succeq \vec{k}^{(i)}$, we remove $\vec{k}^{(j)}$, and increment j. If $\vec{k}^{(j)} \not\succeq \vec{k}^{(i)}$, increment j. Moreover, if we cannot remove $\vec{k}^{(j)}$ "th" times consecutively, increment i and set $j = i+1$. We can choose th freely. If $th = |\mathbb{K}|$, the above algorithm executes `SizeReduce` perfectly. From our experiments, $th = 10$ or $th = 100$ are reasonable parameters.

6.6 Key Recovery Using New Integral Characteristic

This section shows the key recovery step of our cryptanalysis, which uses the 6-round integral characteristic shown in Sect. 6.5. In the characteristic, the left 7-bit value of X_7^L is balanced. Since the integral characteristic does not cover the first FL layer, we first show how to pass through the first FL layer. Then, we calculate two FL layers and one FO function by guessing round keys from ciphertexts, and we evaluate the balanced seven bits.

6.6.1 Passage of First FL Layer

Our new characteristic removes the first FL layer. Therefore, we have to create a set of chosen plaintexts to construct integral characteristics by using guessed round keys $KL_{1,1}$ and $KL_{1,2}$. Here, we have to carefully choose the set of chosen plaintexts to avoid the use of the full code book (see Fig. 6.5, Fig. 6.6, and Fig. 6.7). In every figure, A_i denotes for which we prepare an input set that i bits are active. As an example, we consider an integral characteristic for which the first one bit is constant and the remaining 63 bits are active. Since all bits of the right half are active, we focus only on the left half. We first guess that $KL_{1,2}[1] = 1$, and we then prepare the set of plaintexts as in Fig. 6.5. We next guess that $(KL_{1,1}[1], KL_{1,2}[1]) = (0, 0)$, and we then prepare the set of plaintexts as in Fig. 6.6. Moreover, we guess that $(KL_{1,1}[1], KL_{1,2}[1]) = (1, 0)$, and we then prepare the set of plaintexts as in Fig. 6.7. These chosen plaintexts construct 6-round integral characteristics if the guessed key bits are correct. Note that we do not use 2^{62} chosen plaintexts of the form $(1A_{15}\ 1A_{15}\ A_{16}\ A_{16})$, i.e., we do not use chosen plaintexts satisfying $P^L[1] = P^L[16] = 1$. Thus, our integral characteristics use $2^{64} - 2^{62} \approx 2^{63.58}$ chosen plaintexts.

6.6.2 Sub Key Recovery Using Partial-Sum Technique

Figure 6.8 shows the structure of our key recovery step. We guess $KL_{1,1}[i](= K_1[i])$ and $KL_{1,2}[i](= K_7'[i])$ and then prepare a set of chosen plaintexts to construct an integral characteristic. In the characteristic, seven bits $X_7^L[1, \dots, 7]$ are balanced. Therefore, we evaluate whether or not $X_7^L[j]$ is balanced for $j \in \{1, 2, \dots, 7\}$ by using the partial-sum technique [FKL+00].

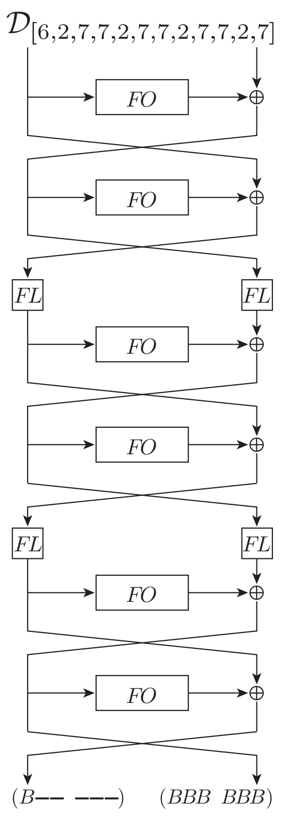


Figure 6.4: New 6-round integral characteristic

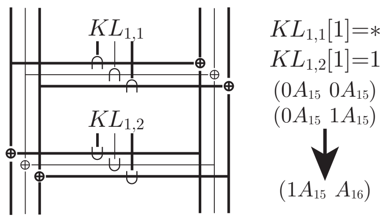


Figure 6.5: $KL_{1,2} = 1$

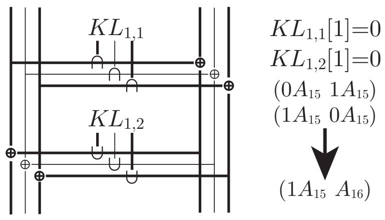


Figure 6.6: $KL_{1,1} = 0, KL_{1,2} = 0$

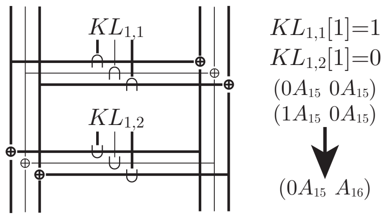


Figure 6.7: $KL_{1,1} = 1, KL_{1,2} = 0$

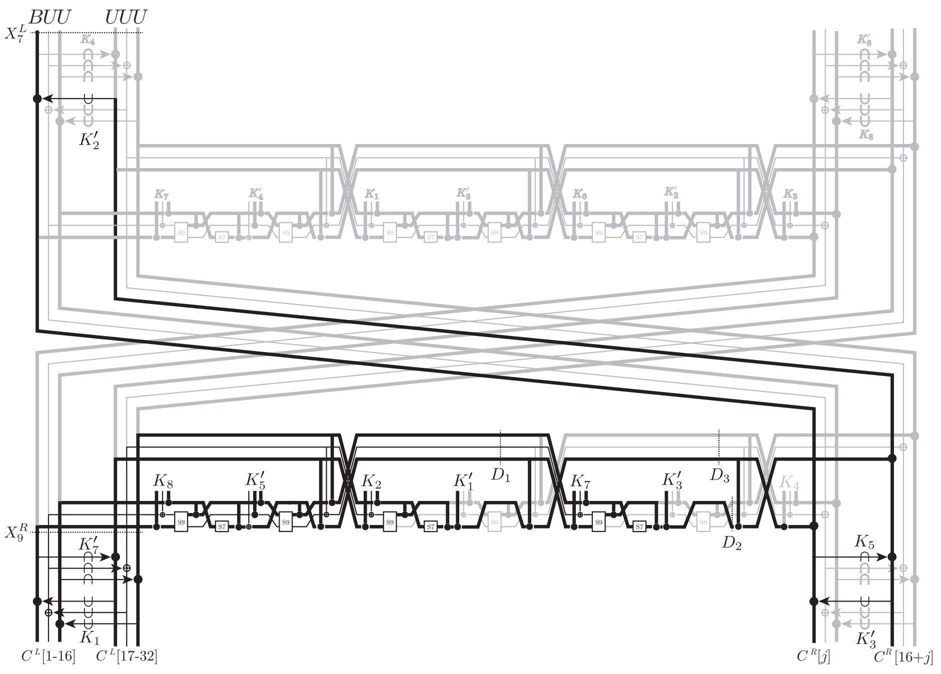


Figure 6.8: Key recovery step

Table 6.5: Procedure of key recovery step

Step	Guessed key	#guessed total bits	New value	Discarded values	#texts	Values in set	Complexity
1		0			2^{34}	$C^L, C^R[j, 16+j]$	
2	K_1, K'_7	32	X_9^R	C^L	2^{34}	$X_9^R, C^R[j, 16+j]$	$2^{34+32} = 2^{66}$
3	K_8, K'_5	64	D_1	$X_9^R[1, \dots, 16]$	2^{34}	$D_1, X_9^R[17, \dots, 32], C^R[j, 16+j]$	$2^{34+64} = 2^{98}$
4	$K'_3[j], (K_7)$	65	$D_2[j]$	D_1 w/o $D_1[j]$	2^{20}	$D_1[j], D_2[j], X_9^R[17, \dots, 32], C^R[j, 16+j]$	$2^{34+65} = 2^{99}$
5	$K_2, (K'_1[j])$	81	$D_3[j]$	$X_9^R[17, \dots, 32], D_1[j]$	2^4	$D_2[j], D_3[j], C^R[j, 16+j]$	$2^{20+81} = 2^{101}$
6	$K_5[j], K'_2[j]$	83	$X_7^L[j]$	$D_2[j], D_3[j], C^R[j, 16+j]$	2^1	$X_7^L[j]$	$2^{4+83} = 2^{87}$

In the first step, we store the frequency of 34 bits $(C^L, C^R[j, 16+j])$ into a voting table for $j \in \{1, 2, \ldots, 7\}$. Then, we partially guess round keys, reduce the size of the voting table, and calculate the XOR of $X_7^L[j]$. Table 6.5 summarizes the procedure of the key recovery step, where every value is defined in Fig. 6.8.

Step 1 Prepare the memory that stores how many times each 34-bit value $(C^L, C^R[j, 16+j])$ appears, and pick the values that appear an odd number of times.

Step 2 Guess 32-bit (K_1, K_7'), and calculate X_9^R from C^L. Delete the parity of the number of occurrences of C^L from the memory, and store that of X_9^R into the memory. Namely, the memory contains a 2^{34}-bit array that stores the parity of the number of occurrences of the 34-bit string $(X_9^R, C^R[j, 16+j])$. The time complexity of Step 2 is $2^{34} \times 2^{32} = 2^{66}$.

Step 3 Additionally guess 32-bit (K_8, K_5'), and calculate D_1 from X_9^R. Delete the parity of the number of occurrences of $X_9^R[1, \ldots, 16]$ from the memory, and store that of D_1 into the memory. Namely, the memory contains a 2^{34}-bit array that stores the parity of the number of occurrences of the 34-bit string $(D_1, X_9^R[17, \ldots, 32], C^R[j, 16+j])$. The time complexity of Step 3 is $2^{34} \times 2^{64} = 2^{98}$.

Step 4 Additionally guess 1-bit $K_3'[j]$, get K_7 from (K_7', K_8), which is already guessed in Step 2 and Step 3, and calculate $D_2[j]$ from D_1. Delete the parity of the number of occurrences of D_1 without $D_1[j]$ from the memory, and store that of $D_2[j]$ into the memory. Namely, the memory contains a 2^{20}-bit array that stores the parity of the number of occurrences of the 20-bit string $(D_1[j], D_2[j], X_9^R[17, \ldots, 32], C^R[j, 16+j])$. The time complexity of Step 4 is $2^{34} \times 2^{65} = 2^{99}$.

Step 5 Additionally guess 32-bit K_2, get $K_1'[j]$ from (K_1, K_2), which is already guessed in Step 2 and Step 5, and calculate $D_3[j]$ from $(X_9^R[17, \ldots, 32], D_1[j])$. Delete the parity of the number of occurrences of $(X_9^R[17, \ldots, 32], D_1[j])$ from the memory, and store that of $D_3[j]$ into the memory. Namely, the memory contains a 2^4-bit array that stores the parity of the number of occurrences of the 4-bit string $(D_2[j], D_3[j], C^R[j, 16+j])$. The time complexity of Step 5 is $2^{20} \times 2^{81} = 2^{101}$.

Step 6 Additionally guess 2-bit $(K_5[j], K_2'[j])$, get $K_3'[j]$, which is already guessed in Step 4, and calculate $X_7^L[j]$ from $(D_2[j], D_3[j], C^R[j, 16+j])$. The time complexity of Step 6 is $2^4 \times 2^{83} = 2^{87}$.

The total time complexity is

$$2^{66} + 2^{98} + 2^{99} + 2^{101} + 2^{87} \approx 2^{101.5}.$$

We repeat the above six steps for $j \in \{1, 2, \ldots, 7\}$. Therefore, the time complexity of the key recovery step is $7 \times 2^{101.5} = 2^{104.3}$.

The key recovery step has to guess the 124-bit key

$$\begin{aligned}&K_1, K_2, K_5[1, \ldots, 7], K_7, K_8,\\ &K_1'[1, \ldots, 7], K_2'[1, \ldots, 7], K_3'[1, \ldots, 7], K_5', K_7'.\end{aligned}$$

Here, K_7' and $K_1'[1, \ldots, 7]$ are uniquely determined by guessing K_7, K_8 and K_1, K_2, respectively. Thus, the guessed key material is reduced to

$$\begin{aligned}&K_1, K_2, K_5[1, \ldots, 7], K_7, K_8,\\ &K_2'[1, \ldots, 7], K_3'[1, \ldots, 7], K_5',\end{aligned}$$

and its size becomes 101 bits. Moreover, since we already guessed 2 bits, i.e., $K_1[i]$ and $K_7'[i]$, to construct integral characteristics, the guessed key bit size is reduced to 99 bits. For wrong keys, the probability that $X_7^L[1, \ldots, 7]$ is balanced is 2^{-7}. Therefore, the number of the candidates of round keys is reduced to 2^{92}. Finally, we guess the 27 bits:

$$K_5[8, \ldots, 16], K_2'[8, \ldots, 16], K_3'[8, \ldots, 16].$$

Note that K_3, K_4, and K_6 are uniquely determined from (K_2, K_2'), (K_3, K_3'), and (K_5, K_5'), respectively. Therefore, the total time complexity is $2^{92+27} = 2^{119}$. We guess the correct key from 2^{119} candidates by using two plaintext-ciphertext pairs, and the time complexity is $2^{119} + 2^{119-64} \approx 2^{119}$. We have to execute the above procedure against $(K_1[i], K_7'[i]) = (0, 0), (0, 1), (1, 0), (1, 1)$, and the time complexity becomes $4 \times 2^{119} = 2^{121}$.

Table 6.6: Trade-off between time and data complexity

#characteristics	Complexity for partial-sum	Complexity for brute-force	Total
1	$1 \times 4 \times 2^{104.3}$	2^{121}	2^{121}
2	$2 \times 4 \times 2^{104.3}$	2^{114}	2^{114}
3	$3 \times 4 \times 2^{104.3}$	2^{107}	$2^{108.5}$
4	$4 \times 4 \times 2^{104.3}$	2^{100}	$2^{108.3}$
5	$5 \times 4 \times 2^{104.3}$	2^{93}	$2^{108.6}$

6.6.3 Trade-off between Time and Data Complexity

In Sect. 6.6.2, we use only one set of chosen plaintexts, where $(2^{64} - 2^{62})$ chosen plaintexts are required. Since the probability that wrong keys are not discarded is 2^{-7}, a brute-force search is required with a time complexity of $2^{128-7} = 2^{121}$, and it is larger than the time complexity of the partial-sum technique. Therefore, if we have a higher number of characteristics, the total time complexity can be reduced.

To exploit several characteristics, we choose some constant bits from seven bits ($i \in \{1, 2, \ldots, 7\}$). If we use a characteristic with $i = 1$, we use chosen plaintexts for which plaintext P^L takes the following values

$$\begin{aligned}
&(00A_{14} \;\; 00A_{14}), (00A_{14} \;\; 01A_{14}), (01A_{14} \;\; 00A_{14}), (01A_{14} \;\; 01A_{14}),\\
&(00A_{14} \;\; 10A_{14}), (00A_{14} \;\; 11A_{14}), (01A_{14} \;\; 10A_{14}), (01A_{14} \;\; 11A_{14}),\\
&(10A_{14} \;\; 00A_{14}), (10A_{14} \;\; 01A_{14}), (11A_{14} \;\; 00A_{14}), (11A_{14} \;\; 01A_{14}),
\end{aligned}$$

where A_{14} denotes that all values appear the same number independently of other bits, e.g., $(00A_{14} \;\; 00A_{14})$ uses 2^{60} chosen plaintexts because P^R also takes all values. Moreover, if we use a characteristic with $i = 2$, we use chosen plaintexts for which P^L takes the following values

$$\begin{aligned}
&(00A_{14} \;\; 00A_{14}), (00A_{14} \;\; 10A_{14}), (10A_{14} \;\; 00A_{14}), (10A_{14} \;\; 10A_{14}),\\
&(00A_{14} \;\; 01A_{14}), (00A_{14} \;\; 11A_{14}), (10A_{14} \;\; 01A_{14}), (10A_{14} \;\; 11A_{14}),\\
&(01A_{14} \;\; 00A_{14}), (01A_{14} \;\; 10A_{14}), (11A_{14} \;\; 00A_{14}), (11A_{14} \;\; 10A_{14}).
\end{aligned}$$

When both characteristics are used, they do not require choosing plaintexts for which P^L takes $(11A_{14} \;\; 11A_{14})$. Therefore, $(2^{64} - 2^{60})$ chosen plaintexts are required, and the probability that wrong keys are not discarded becomes 2^{-14}. Similarly, when three characteristics, which require $(2^{64} - 2^{58})$ chosen plaintexts, are used, the probability that wrong keys are not discarded becomes 2^{-21}.

Table 6.6 summarizes the trade-off between time and data complexity. For the use of each characteristic, we have to execute four key recoveries with the partial-sum technique, i.e., for $(KL_{1,1}[1], KL_{1,2}[1]) \in \{(0,1), (1,1), (0,0), (1,0)\}$. It shows that the use of four characteristics is optimized from the perspective of time complexity. Namely, when $(2^{64} - 2^{56}) \approx 2^{63.994}$ chosen plaintexts are required, the time complexity to recover the secret key is $2^{108.3}$.

6.6.4 Follow-Up Results and Open Problem

After a preliminary version [Tod15a] was published, Achiya Bar-On improved the key recovery step [Bar15a] by using the same integral characteristic shown in this chapter. Then, this paper was accepted in CRYPTO2016 [BK16]. The improved key recovery technique uses the meet-in-the-middle technique [SW12b] under the chosen ciphertext setting. It dramatically reduces the time complexity where the secret key is recovered, and the time complexity is $2^{69.5}$. On the other hand, it requires the full code book. When we consider the data complexity optimization, our attack, which requires 2^{121} time complexity and $2^{63.58}$ chosen plaintexts, is still the best attack. We need to construct a more efficient integral characteristic if we want to improve the data complexity, and it is left as an open problem.

6.7 Conclusion

In this chapter, we showed a cryptanalysis of the full MISTY1. MISTY1 was well evaluated and standardized by several projects, such as CRYPTREC, ISO/IEC, and NESSIE. We constructed a new integral characteristic by using the propagation characteristic of the division property. Here, we improved the division property by optimizing the division property for a public S-box. As a result, a new 6-round integral characteristic is constructed, and we can recover the secret key of the full MISTY1 with $2^{63.58}$ chosen plaintexts and 2^{121} time complexity. If we can use $2^{63.994}$ chosen plaintexts, our attack can recover the secret key with a time complexity of $2^{108.3}$.

Chapter 7

Application to Extended Generalized Feistel Network: Lilliput

Abstract– This chapter provides security analysis of lightweight block cipher Lilliput, which is an instantiation of extended generalized Feistel network (EGFN) developed by Berger *et al.* at SAC 2013. Its round function updates a part of the state only linearly, which yields several security concerns. In this chapter, we propose the best third-party cryptanalysis. Owing to its unique computation structure, the designers expected that EGFN efficiently enhances the security against integral cryptanalysis. However, the security is not enhanced as the designers expect. In fact, division property finds a 13-round distinguisher which improves the previous distinguisher by 4 rounds. The new distinguisher is further extended to a 17-round key recovery attack which improves the previous best attack by 3 rounds.
Keywords– Block cipher, Lilliput, extended generalized Feistel network, division property

7.1 Introduction

Lightweight cryptography is one of the most actively discussed topics in the current community of symmetric-key cryptosystems. A huge number of designs have been proposed especially for the last decade. Here, we omit the list of all the lightweight primitives. Please refer to [AB15] for such a list. The motivation of lightweight cryptography is to design a secure symmetric-key cryptosystem under area-constraining environment.

One of the major approaches to design lightweight cipher is using Feistel network or generalized Feistel network (GFN), which has a property that its transformation is basically involutive thus the overhead to implement decryption circuit is minimized. Meanwhile, diffusion speed of the standard Feistel network is often much slower than other design approaches. To overcome this drawback, several researches have developed new ideas. Suzaki *et al.* pointed out that security of GFN can be enhanced by replacing the way of mixing branches [SM10]. This is called *block-shuffle* and TWINE [SMMK12] was designed based on this idea. Zhang and Wu used modified Feistel network to design LBlock [WZ11], which turned out to be the same network as one in TWINE. The latest approach, which is a main focus in this chapter, is *extended GFN (EGFN)* proposed by Berger *et al.* [BMT13], in which an additional linear diffusion layer is inserted between the application to F-function and branch network. The comparison of GFN and EGFN is depicted in Fig. 7.1. In many designs, the nonlinear layer is the most expensive, thus the linear layer leads to better diffusion speed with a small extra cost.

Berger et al. [BMT13] specified two concrete examples of EGFN with security analysis. Unfortunately, mistakes in the security analysis were pointed out by Zhang and Wu [ZW14] and very effective differential trails were constructed for those original choices of EGFN. To fix this drawback, Berger et al. combined block-shuffle [SM10] with EGFN, and proposed a new cipher preventing the attack by Zhang and Wu. The cipher was named Lilliput [BFMT15].

Lilliput is a lightweight block cipher, supporting 64-bit block and 80-bit key. Lilliput is a 16-branch EGFN with block-shuffle, in which the size of each branch is 4 bits (nibble) and

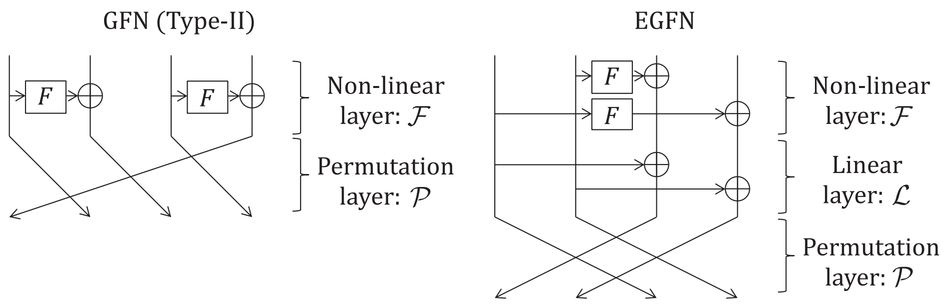

GFN (Type-II)
EGFN
F
Non-linear layer: $\mathcal{F}$
Permutation layer: $\mathcal{P}$
Linear layer: $\mathcal{L}$

Figure 7.1: Comparison of GFN (Left) and EGFN (Right) with four branches

the nonlinear function is an application of a 4-bit S-box. Those parameter sizes are the same as TWINE and LBlock. The number of rounds is 30, which is 2 rounds less than TWINE and LBlock. This shows that the additional linear layer of Lilliput allows to ensure its security with a smaller number of rounds than TWINE and LBlock. The designers of Lilliput provided several security analysis, including minimal number of active S-boxes for every round, impossible differential attack, integral attack, differential/linear cryptanalysis, related-key attacks and chosen-key attacks. Single-key attacks are summarized in Table 7.1.

Table 7.1: Key recovery attacks in the single-key model against Lilliput. Related-key attack and chosen-key attacks reach 23 rounds, which are not included in this table.

approaches	distinguisher	key recovery	data	time	ref.
integral	9 rounds	13 rounds	2^{62}	2^{72}	[BFMT15]
impossible differential	8 rounds	14 rounds	2^{63}	2^{77}	[BFMT15]
division property	13 rounds	17 rounds	2^{63}	2^{77}	**Ours**

Our Contributions. In this chapter, we show that the linear layer of EGFN and Lilliput yields security concern to be carefully discussed. The designers evaluated the security in [BFMT15, BMT13], where the propagation characteristic of the integral property [KW02] was used to search for the integral distinguisher. They showed that EGFN and Lilliput have higher security than GFN with block-shuffle. Actually, while TWINE and LBlock allow 15-round integral distinguisher, Lilliput only allows the 9-round integral distinguisher. It implies that the linear layer enhances security against the integral cryptanalysis by $6(= 15 - 9)$ rounds. On the other hand, the linear layer does not increase the algebraic degree. Hence by constructing the integral characteristic by estimating the algebraic degree, which is often called the higher order differential cryptanalysis, the attack may be improved drastically. The division property is a new method to find integral distinguisher, which is a generalization of the integral property and can exploit low algebraic degree in the same time. Thus security contribution of the linear layer can be evaluated more accurately by using the division property. As a result, we show that the division property finds a 13-round integral distinguisher, and it implies that the security is not enhanced as the designers expected. Moreover, the new distinguisher leads to the attack against 17-round Lilliput (see Table 7.1), which is the current best attack against Lilliput.

7.2 Lilliput

7.2.1 Extended Generalized Feistel Network (EGFN)

Extended generalized Feistel network (EGFN) [BMT13] was invented by Berger et al. so as to achieve faster diffusion than ordinary generalized Feistel network. Previous GFN has two computation layers per round; one is applying nonlinear functions to some of branches and XORing the results to other branches (nonlinear layer $\mathcal{F}$), and the other is permuting branches (permutation layer $\mathcal{P}$), which is often designed as a simple cyclic shift of branches. EGFN [BMT13] adds a new diffusion layer (linear layer $\mathcal{L}$). In many designs, the nonlinear layer $\mathcal{F}$ is the most expensive part, thus the linear layer $\mathcal{L}$ helps to increase the diffusion speed with a small additional cost. Berger et al. showed two concrete choices of $\mathcal{F}$ and $\mathcal{L}$ when the number of branches is 8 and 16 along with some security analysis. It is notable that the permutation $\mathcal{P}$ was assumed to be a simple swap of the left half and right half of the state.

Zhang and Wu [ZW14] pointed out that the security evaluation in [BMT13] was wrong and presented efficient differential characteristics against concrete examples in [BMT13]. The attack relies on the choice of $\mathcal{P}$, which is a simple swap of branches.

7.2.2 Specification of Lilliput

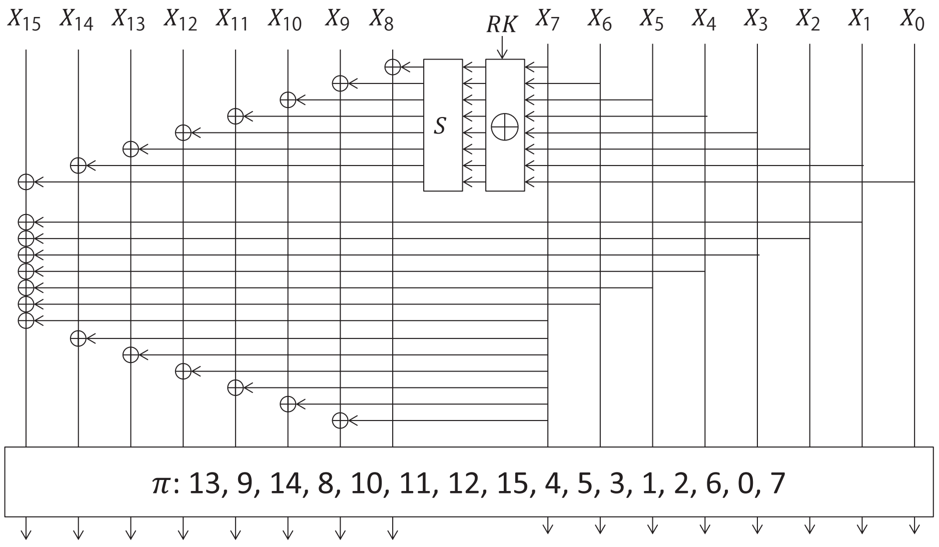

X_{15} X_{14} X_{13} X_{12} X_{11} X_{10} X_9 X_8
RK
X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0
S
π: 13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7

Figure 7.2: Round function of Lilliput

Lilliput [BFMT15] was designed by Berger et al. in 2015. So as to prevent the attack by Zhang and Wu [ZW14], the designers adopted block-shuffle network [SM10] proposed by Suzaki et al. on top of EGFN so as to achieve even faster diffusion.

The block size and key size of Lilliput are 64 bits and 80 bits, respectively. Its round function consists of 16 branches of size 4 bits. 64-bit plaintext is first loaded to sixteen 4-bit array $X_{15}, X_{14} \ldots, X_0$. Then, the round function consisting of three layers $\mathcal{F}$, $\mathcal{L}$, and $\mathcal{P}$ is iterated 30 times. The permutation layer $\mathcal{P}$ is omitted in the last round for involution reasons. An illustration of the round function is shown in Fig. 7.2.

The key schedule first expands the 80-bit key to 32-bit round keys for round j ($j = 0, \ldots, 29$) dented by RK^j. We omit its description because we do not analyze the key schedule.

Non-linear Layer $\mathcal{F}$. At first, the state input and round key are XORed. Then, a 4-bit S-box is applied to each of eight nibbles in the right half of the state, and the results are xored to the left half of the state. Let RK_i^j and X_i^j be the ith nibble of the jth round key RK^j and jth round state X^j, respectively. Then, the nonlinear layer can be defined as

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	4	8	7	1	9	3	2	E	0	B	6	F	A	5	D	C

Linear Layer $\mathcal{L}$. The idea in $\mathcal{L}$ is, along with diffusion by $\mathcal{F}$, having X_7^j propagate to all nibbles in the left half of the state and having X_{15}^j be propagated from all nibbles from the right half of the state. $\mathcal{L}$ is defined as follows.

$$X_{15}^j \leftarrow X_{15}^j \oplus X_7^j \oplus X_6^j \oplus X_5^j \oplus X_4^j \oplus X_3^j \oplus X_2^j \oplus X_1^j,$$
$$X_{15-i}^j \leftarrow X_{15-i}^j \oplus X_7^j \text{ for } i = 1, 2, \ldots, 6.$$

Permutation Layer $\mathcal{P}$. Nibble positions are permuted with permutation π defined as

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi(x)$	13	9	14	8	10	11	12	15	4	5	3	1	2	6	0	7

The designers chose π to achieve the highest number of active S-boxes after 18, 19 and 20 rounds.

7.3 New Integral Characteristic Based on Division Property

The designers of EGFN and Lilliput already showed the security against the integral cryptanalysis in [BFMT15, BMT13], and the propagation characteristic of the integral property [KW02] was used to search for the integral distinguisher. When a d-round EGFN reaches the full diffusion, the integral distinguisher of the EGFN covers at most $2d+2$ rounds. Moreover, Lilliput, which is a specific block cipher based EGFN with $d = 4$, has the 9-round integral distinguisher. Compared with 15-round integral distinguishers of TWINE and LBlock, it implies that the linear layer enhances the security against the integral cryptanalysis by $6(= 15 - 9)$ rounds. On the other hand, if we construct the integral distinguisher by estimating the algebraic degree, which is often called the higher order differential cryptanalysis, the security is not likely to dramatically improve because the linear layer does not increase the algebraic degree.

The division property is a new method to find integral distinguishers, and it is the generalization of the integral property so that can exploit the algebraic degree in the same time. Therefore, we can more accurately evaluate the contribution of the linear layer by using the division property. In this section, we show a new integral distinguisher with the division property, and it covers 13 rounds, which is beyond $2d + 2 = 10$. Zhang and Wu showed that TWINE and LBlock have 16-round integral distinguishers by using the division property [ZW15]. Therefore, the true contribution by the linear layer is $3(= 16-13)$ rounds. Moreover, this 13-round integral distinguisher leads to a 17-round attack, which is a current best attack against Lilliput.

7.3.1 Recall Division Property

This section briefly shows the definition and propagation rules to understand this chapter. Please refer to Chapter 5 for details.

The division property of a multiset is evaluated by using the bit product function defined as follows. Let $\pi_{\vec{u}} : (\mathbb{F}_2^n)^m \to \mathbb{F}_2$ be a bit product function for any $\vec{u} \in (\mathbb{F}_2^n)^m$. Let $\vec{x} \in (\mathbb{F}_2^n)^m$ be the input, and $\pi_{\vec{u}}(\vec{x})$ is defined as

$$\pi_{\vec{u}}(\vec{x}) := \prod_{i=1}^{m} \left(\prod_{j=1}^{n} x_i[j]^{u_i[j]} \right).$$

Notice that $x_i[j]^1 = x_i[j]$ and $x_i[j]^0 = 1$.

Definition 5.3 (Division Property)**.** *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2^n)^m$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n^m}$, where $\mathbb{K}$ denotes a set of m-dimensional vectors whose elements take a value between 0 and n, it fulfills the following conditions:*

$$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} \textit{unknown} & \textit{if there are } \vec{k} \in \mathbb{K} \textit{ s.t. } W(\vec{u}) \succeq \vec{k}, \\ 0 & \textit{otherwise,} \end{cases}$$

where $W(\vec{u}) = (w(u_m), \ldots, w(u_1)) \in \mathbb{Z}^m$ and $w(u_j) = \sum_{i=1}^{n} u_j[i]$. Moreover, $\vec{k} \succeq \vec{k}'$ denotes $k_i \geq k_i'$ for all $i \in \{1, 2, \ldots, m\}$.

If there are $\vec{k} \in \mathbb{K}$ and $\vec{k}' \in \mathbb{K}$ satisfying $\vec{k} \succeq \vec{k}'$ in the division property $\mathcal{D}_{\mathbb{K}}^{n^m}$, $\vec{k}$ can be removed from $\mathbb{K}$ because the vector $\vec{k}$ is redundant. Let $\mathbb{X}$ be the set of texts encrypted by r rounds, and $e_i \in \mathbb{Z}^m$ denotes an unit vector whose ith element is one and the others are zero. Assuming that $\mathbb{X}$ fulfills the division property $\mathcal{D}_{\mathbb{K}}^{n^m}$ and e_i does not belong to $\mathbb{K}$, the cipher has the r-round integral distinguisher, where the ith element is balanced. We summarize the propagation of the division property in Proposition 5.2 and Sect. 5.3.5.

Algorithm 8 Propagation from $\mathcal{D}_{\mathbb{K}}^{4^{16}}$ for the round function of Lilliput

```
procedure nonLinear($\mathbb{K}, i, j$)
    $\mathbb{K}' \Leftarrow \phi$
    for all $\vec{k} \in \mathbb{K}$ do
        $\vec{k}' \Leftarrow \vec{k}$
        for $x = 0$ to $k_j$ do
            $k_i' \Leftarrow k_i + D_S(x)$
            $k_j' \Leftarrow k_j - x$
            if $k_i' \leq 4$ then
                $\mathbb{K}' \Leftarrow \mathbb{K}' \cup \{\vec{k}'\}$
            end if
        end for
    end for
    remove redundant vectors from $\mathbb{K}'$
    return $\mathbb{K}'$
end procedure
```

```
procedure linear($\mathbb{K}, i, j$)
    $\mathbb{K}' \Leftarrow \phi$
    for all $\vec{k} \in \mathbb{K}$ do
        $\vec{k}' \Leftarrow \vec{k}$
        for $x = 0$ to $k_j$ do
            $k_j' \Leftarrow k_j - x$
            $k_i' \Leftarrow k_i + x$
            if $k_i' \leq 4$ then
                $\mathbb{K}' \Leftarrow \mathbb{K}' \cup \{\vec{k}'\}$
            end if
        end for
    end for
    remove redundant vectors from $\mathbb{K}'$
    return $\mathbb{K}'$
end procedure
```

7.3.2 Integral Distinguisher on Lilliput

The state of Lilliput is represented as sixteen 4-bit values, and the use of the division property $\mathcal{D}_{\mathbb{K}}^{4^{16}}$ is appropriate. Let $|\mathbb{K}|$ be the number of elements in $\mathbb{K}$, and the upper bound of $|\mathbb{K}|$ is $5^{16} \approx 2^{37.15}$. Since we can reduce $|\mathbb{K}|$ by removing redundant vectors in general, we can practically evaluate the propagation characteristic of $\mathcal{D}_{\mathbb{K}}^{4^{16}}$.

Propagation Characteristic. The round function of EGFN consists of three layers: the nonlinear layer, linear layer, and permutation layer. In the nonlinear layer of EGFN, the core operation is

$$x_i = x_i \oplus F(x_j)$$

for appropriate i and j. We only focus on the case that F is permutation because the most important instantiation Lilliput uses a bijective S-box. Let $\mathcal{D}_k^4$ and $\mathcal{D}_{k'}^4$ be the input and output division property for the S-box, respectively. As the algebraic degree of F is at most three, it holds

$$k' = D_S(k) = \begin{cases} 4 & \text{if } k = 4, \\ 1 & \text{if } k = 1, 2, 3, \\ 0 & \text{if } k = 0. \end{cases}$$

Assuming $\mathcal{D}_{(k_i,k_j)}^{4^2}$ be the input division property of the Feistel structure, the output division property $\mathcal{D}_{\mathbb{K}}^{4^2}$ is

$$\mathbb{K} = \{(k_i + D_S(x), k_j - x) \mid 0 \leq x \leq k_j, D_S(x) \leq 4 - k_i\}.$$

The propagation characteristic for the nonlinear layer is shown in `nonLinear` of Algorithm 8.

The linear layer of EGFN consists of the iteration of XORs as

$$x_i = x_i \oplus x_j$$

for appropriate i and j. Therefore, assuming $\mathcal{D}_{(k_i,k_j)}^{4^2}$ be the input division property of the Feistel structure, the output division property $\mathcal{D}_{\mathbb{K}}^{4^2}$ is

$$\mathbb{K} = \{(k_i + x, k_j - x) \mid 0 \leq x \leq \min\{k_j, 4 - k_i\}\}.$$

The propagation characteristic for the linear layer is shown in `linear` of Algorithm 8.

About the permutation layer, the propagation characteristic is the only modification of the corresponding index. The entire algorithm to evaluate the propagation characteristic of the round function is shown in `roundFunction` of Algorithm 9.

Algorithm 9 Propagation from $\mathcal{D}^{4^{16}}_{\mathbb{K}}$ for the round function of Lilliput

```
1:  procedure roundFunction($\mathbb{K}$)
2:      for all $(i, j) \in \{(8, 7), (9, 6), (10, 5), (11, 4), (12, 3), (13, 2), (14, 1), (15, 0)\}$ do
3:          $\mathbb{K} =$ nonLinear($\mathbb{K}, i, j$)
4:      end for
5:      for all $(i, j) \in \{(15, 1), (15, 2), (15, 3), (15, 4), (15, 5), (15, 6), (15, 7)\}$ do
6:          $\mathbb{K} =$ linear($\mathbb{K}, i, j$)
7:      end for
8:      for all $(i, j) \in \{(14, 7), (13, 7), (12, 7), (11, 7), (10, 7), (9, 7)\}$ do
9:          $\mathbb{K} =$ linear($\mathbb{K}, i, j$)
10:     end for
11:     $\mathbb{K}' \Leftarrow \phi$
12:     for all $\vec{k} \in \mathbb{K}$ do
13:         for $i = 0$ to 16 do
14:             $k'_{\pi(i)} \Leftarrow k_i$
15:         end for
16:         $\mathbb{K}' \Leftarrow \mathbb{K}' \cup \{\vec{k}'\}$
17:     end for
18:     return $\mathbb{K}'$
19: end procedure
```

Table 7.2: Propagation from $\mathcal{D}^{4^{16}}_{\{[4,4,\ldots,4,3]\}}$

#rounds	0	1	2	3	4	5	6 $\star$
$\lvert\mathbb{K}\rvert$	1	1	3	14	377	33948	5513237
$\max_w(\mathbb{K})$	63	63	63	63	63	55	$\leq$57
$\min_w(\mathbb{K})$	63	63	61	59	55	19	35

#rounds	7 $\star$	8 $\star$	9 $\star$	10	11	12	13
$\lvert\mathbb{K}\rvert$	266813452	70804820	1385951	16960	572	52	16
$\max_w(\mathbb{K})$	$\leq$51	$\leq$43	$\leq$25	13	6	4	2
$\min_w(\mathbb{K})$	22	9	6	3	2	1	1

In rounds labeled $\star$, the set $\mathbb{K}$ includes redundant vectors.

New Integral Distinguisher. As the number of exploiting chosen plaintexts increases, the integral distinguisher can analyze more rounds in general. Therefore, we evaluate all integral distinguishers with 2^{63} chosen plaintexts where only one bit in the right half is constant. Note that these distinguishers are always better than distinguishers whose only one bit in the left half is constant. We choose one 4-bit value from X_0 to X_7, and we prepare chosen plaintexts such that any one bit in the chosen value is constant and the others are active.

We implemented Algorithm 9 and searched for non-trivial integral distinguishers. Let $\mathcal{D}^{4^{16}}_{\vec{k}}$ be the plaintext division property. When we choose one-bit constant from X_p, we use $\vec{k}$ as

$$k_i = \begin{cases} 4 & \text{if } i \neq p \\ 3 & \text{if } i = p \end{cases}$$

for $i \in \{0, 1, \ldots, 16\}$. We coded our algorithm with C++, and we executed it in Xeon Processor E5-2699 (18 cores) in 128 GB RAM. As a result, our algorithm found 13-round integral distinguishers for $p = 0$ and $p = 6$. For other p, our algorithm found 12-round integral distinguishers.

When $p = 0$ i.e., $\vec{k} = [4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3]$, we find a 13-round integral distinguisher, and the position X_9^{13} is balanced. Table 7.2 shows the propagation characteristic, where $\min_w(\mathbb{K})$ and $\max_w(\mathbb{K})$ are calculated as

$$\min_w(\mathbb{K}) = \min_{\vec{k} \in \mathbb{K}} \left\{ \sum_{i=1}^{16} k_i \right\}, \quad \max_w(\mathbb{K}) = \max_{\vec{k} \in \mathbb{K}} \left\{ \sum_{i=1}^{16} k_i \right\}.$$

Round 0 denotes the division property of the plaintext set, and we perfectly remove redundant vectors except for 6, 7, 8, and 9 rounds.

7.4 Key Recovery Using New Integral Characteristic

Let X_i^j be the j-round nibble value in X_i, where the plaintext is represented as $(X_{15}^0, \dots, X_0^0)$. Moreover, let Y_i^j be the output of the S-box as $Y_i^j = S(X_i^j \oplus RK_i^j)$. We prepare 2^{63} chosen plaintexts such that any one bit of X_0^0 is constant and the other 63 bits are active. Then, it holds $\bigoplus X_9^{13} = 0$, and we can attack 17-round LILLIPUT by using the 13-round integral distinguisher. In our attack, let $c = (c_{15}, \dots, c_0)$ be the ciphertext, where the linear layer of the last round is removed. Note that the last round of LILLIPUT has the linear layer but this c is equivalent with the ciphertext of 17-round LILLIPUT because the linear layer is public.

Since LILLIPUT has many XORs in the round function, the procedure of the key recovery is very complicating. For simplicity, we use the following strategy. We first decompose four rounds of LILLIPUT into five subfunctions denoted by f_{13}, f_{14}, f_{15}, f_{16}, and L. Here the output of f_i is the XOR of Y^i involved in X_9^{13}, and the output of L is the linear part to compute X_9^{13} from ciphertext. Then

$$X_9^{13} = f_{13}(c, \mathcal{K}_{13}) \oplus f_{14}(c, \mathcal{K}_{14}) \oplus f_{15}(c, \mathcal{K}_{15}) \oplus f_{16}(c, \mathcal{K}_{16}) \oplus L(c),$$

where $\mathcal{K}_i$ is the set of round keys involved in f_i. The bit sizes of $\mathcal{K}_{13}$, $\mathcal{K}_{14}$, $\mathcal{K}_{15}$, and $\mathcal{K}_{16}$ are 44, 16, 48, and 28 bits, respectively. Then,

$$\begin{aligned}
f_{13}(c, \mathcal{K}_{13}) &= Y_6^{13}, \\
f_{14}(c, \mathcal{K}_{14}) &= Y_0^{14}, \\
f_{15}(c, \mathcal{K}_{15}) &= Y_0^{15} \oplus Y_1^{15} \oplus Y_3^{15} \oplus Y_5^{15} \oplus Y_6^{15} \oplus Y_7^{15}, \\
f_{16}(c, \mathcal{K}_{16}) &= Y_0^{15} \oplus Y_1^{15} \oplus Y_3^{15} \oplus Y_4^{15} \oplus Y_5^{15} \oplus Y_6^{15} \oplus Y_7^{15}.
\end{aligned}$$

We compute the sum of $f_i(c, \mathcal{K}_i)$ by guessing $\mathcal{K}_i$ independently of i. Then, we compute keys satisfying

$$\bigoplus_{X^0} f_{13}(c, \mathcal{K}_{13}) \oplus f_{14}(c, \mathcal{K}_{14}) \oplus f_{16}(c, \mathcal{K}_{16}) = \bigoplus_{X^0} f_{15}(c, \mathcal{K}_{15}) \oplus L(c) \tag{7.1}$$

Note that we do not need to guess round keys to compute the sum of $L(c)$. Note that $\mathcal{K}_{13} \cup \mathcal{K}_{14} \cup \mathcal{K}_{15} \cup \mathcal{K}_{16}$ is 72 bits, and the probability that Eq. (7.1) holds randomly is 2^{-4}. Therefore, we reduce the space of key candidates from 2^{72} to 2^{68}. Finally, we recover the correct key by additionally guessing the remaining 8 bits. It is enough to determine the correct key by using two known plaintexts. Thus, the total time complexity is $2^{76} \times 2 = 2^{77}$.

Note that the time complexity that we evaluate whether Eq. (7.1) holds or not is less than 2^{61} and it is negligible.

7.4.1 Detail Procedure for Key Recovery

Table 7.3: Key-recovery procedure for f_{13}

Step	Guessed key	#guessed total bits	New value	Discarded values	#texts	Values in set	Complexity
1		0			2^{56}	$c_{15}, c_{13}, \dots, c_2, c_0$	
2	RK_7^{16}	4	t_1	c_7, c_8	2^{52}	$c_{15}, c_{13}, \dots, c_9, c_6, \dots, c_2, c_0, t_1$	$2^{56+4} = 2^{60}$
3	RK_6^{16}	8	t_2	c_6, c_9, t_1	2^{44}	$c_{15}, c_{13}, \dots, c_{10}, c_5, \dots, c_2, c_0, t_2$	$2^{52+8} = 2^{60}$
4	RK_5^{16}	12	t_3	c_5, c_{10}, t_2	2^{36}	$c_{15}, c_{13}, c_{12}, c_{11}, c_4, c_3, c_2, c_0, t_3$	$2^{44+12} = 2^{56}$
5	RK_4^{16}	16	t_4	c_{11}, t_3	2^{32}	$c_{15}, c_{13}, c_{12}, c_4, c_3, c_2, c_0, t_4$	$2^{36+16} = 2^{52}$
6	RK_3^{16}	20	t_5	c_3, c_{12}, t_4	2^{24}	$c_{15}, c_{13}, c_4, c_2, c_0, t_5$	$2^{32+20} = 2^{52}$
7	RK_2^{16}, RK_0^{15}	28	t_6	c_2, c_{13}, t_5	2^{16}	c_{15}, c_4, c_0, t_6	$2^{24+28} = 2^{52}$
8	RK_0^{16}	32	t_7, X_{15}^{16}	c_0, c_{15}, t_6	2^{12}	c_4, t_7, X_{15}^{16}	$2^{16+32} = 2^{48}$
9	RK_7^{15}	36	X_8^{15}	c_4, X_{15}^{16}	2^{8}	t_7, X_8^{15}	$2^{12+36} = 2^{48}$
10	RK_3^{14}	40	X_{12}^{14}	t_7, X_8^{15}	2^{4}	X_{12}^{14}	$2^{8+40} = 2^{48}$
11	RK_6^{13}	44	Y_6^{13}	X_{12}^{14}	2^{4}	Y_6^{13}	$2^{4+44} = 2^{48}$

1. Evaluation of $f_{13}(c, \mathcal{K}_{13})$. Round keys including $\mathcal{K}_{13}$ is

$$\mathcal{K}_{13} = \{RK_6^{13}, RK_3^{14}, RK_0^{15}, RK_7^{15}, RK_0^{16}, RK_2^{16}, \dots, RK_7^{16}\}.$$

Therefore, we compute the sum of Y_6^{13} from 56-bit $c_{15}, c_{13}, \dots, c_2, c_0$ by guessing 44-bit $\mathcal{K}_{13}$. First,

$$\begin{aligned} f_{13}(c, \mathcal{K}_{13}) &= S(X_{12}^{14} \oplus RK_6^{13}) \\ &= S(S(t_7 \oplus X_8^{15} \oplus RK_3^{14}) \oplus RK_6^{13}) \\ &= S(S(t_7 \oplus S(X_{15}^{16} \oplus RK_7^{15}) \oplus c_4 \oplus RK_3^{14}) \oplus RK_6^{13}), \end{aligned}$$

where $t_7 = X_2^{15} \oplus X_{15}^{15}$ is calculated as

$$\begin{aligned} t_7 &= t_6 \oplus c_{15} \oplus S(c_0 \oplus RK_0^{16}), \\ t_6 &= t_5 \oplus S(S(c_2 \oplus RK_2^{16}) \oplus c_{13} \oplus RK_0^{15}), \\ t_5 &= t_4 \oplus c_{12} \oplus S(c_3 \oplus RK_3^{16}), \\ t_4 &= t_3 \oplus c_{11} \oplus S(c_4 \oplus RK_4^{16}), \\ t_3 &= t_2 \oplus c_{10} \oplus S(c_5 \oplus RK_5^{16}), \\ t_2 &= t_1 \oplus c_9 \oplus S(c_6 \oplus RK_6^{16}), \\ t_1 &= c_7 \oplus c_8 \oplus S(c_7 \oplus RK_7^{16}). \end{aligned}$$

We can evaluate the value $t_7 = X_2^{15} \oplus X_{15}^{15}$ efficiently by using a partial-sum technique [FKL+00]. Table 7.3 shows the key-recovery procedure. Finally, we can create a table $T_{13} = \{\oplus f_{13}(c, \mathcal{K}_{13}), \mathcal{K}_{13}\}$ with about 2^{61} time complexity.

2. Evaluation of $f_{14}(c, \mathcal{K}_{14})$. Round keys including $\mathcal{K}_{14}$ is

$$\mathcal{K}_{14} = \{RK_0^{14}, RK_2^{15}, RK_0^{16}, RK_1^{16}\}.$$

Therefore, we compute the sum of Y_0^{14} from 20-bit $c_{15}, c_{14}, c_6, c_1 c_0$ by guessing 16-bit $\mathcal{K}_{14}$. Clearly, the time complexity to create a table $T_{14} = \{\oplus f_{14}(c, \mathcal{K}_{14}), \mathcal{K}_{14}\}$ is at most 2^{36}, which is negligible.

3. Evaluation of $f_{15}(c, \mathcal{K}_{15})$. Round keys including $\mathcal{K}_{15}$ is

$$\mathcal{K}_{15} = \{RK_0^{15}, RK_1^{15}, RK_3^{15}, RK_5^{15}, RK_6^{15}, RK_7^{15}, RK_0^{16}, RK_2^{16}, RK_3^{16}, RK_4^{16}, RK_6^{16}, RK_7^{16}\}.$$

Therefore, we compute the sum of $f_{15}(c, \mathcal{K}_{15})$ from 52-bit $c_{15}, c_{13}, c_{12}, c_{11}, c_9, \dots, c_2, c_0$ by guessing 48-bit $\mathcal{K}_{15}$. However, since the output of f_{15} is calculated as

$$f_{15}(c, \mathcal{K}_{15}) = Y_0^{15} \oplus Y_1^{15} \oplus Y_3^{15} \oplus Y_5^{15} \oplus Y_6^{15} \oplus Y_7^{15},$$

we can compute the sum of Y_i^{15} independently. Let $\mathcal{K}_{15,i}$ be 8-bit round keys involved in Y_i^{15}. Then, since each Y_i^{15} involves 8-bit ciphertext and 8-bit round key, the time complexity to create a table $T_{15,i} = \{\oplus Y_i^{15}, \mathcal{K}_{15,i}\}$ is at most 2^{16}. Finally, we generate a table $T_{15,i} = \{\oplus f_{15}(c, \mathcal{K}_{15}), \mathcal{K}_{15}\}$ from $T_{15,0}$, $T_{15,1}$, $T_{15,3}$, $T_{15,5}$, $T_{15,6}$, and $T_{15,7}$. Since the size of each table is 2^8, the time complexity to create a table T_{15} is at most 2^{48}, which is negligible.

4. Evaluation of $f_{16}(c, \mathcal{K}_{16})$. Round keys including $\mathcal{K}_{13}$ is

$$\mathcal{K}_{16} = \{RK_0^{16}, RK_1^{16}, RK_3^{16}, RK_4^{16}, RK_5^{16}, RK_6^{16}, RK_7^{16}\},$$

Therefore, we compute the sum of $f_{16}(c, \mathcal{K}_{16})$ from 28-bit $c_7, \dots, c_3, c_1, c_0$ by guessing 28-bit $\mathcal{K}_{16}$. Clearly, the time complexity to create a table $T_{16} = \{\oplus f_{16}(c, \mathcal{K}_{16}), \mathcal{K}_{16}\}$ is at most 2^{48}, which is negligible.

5. Merge tables and meet-in-the-middle technique. Recall that $\mathcal{K}_{13}$, $\mathcal{K}_{14}$, $\mathcal{K}_{15}$, and $\mathcal{K}_{16}$ is 44, 16, 48, and 28 bits, respectively. Therefore, the size of T_{13}, T_{14}, T_{15}, and T_{16} is 2^{44}, 2^{16}, 2^{48}, and 2^{28}, respectively. Then, $\mathcal{K}_{13} \cup \mathcal{K}_{14} \cup \mathcal{K}_{16}$ is 56 bits. Therefore, we first merge three tables T_{13}, T_{14}, and T_{16}, and let

$$T_A = \{f_{13}(c, \mathcal{K}_{13}) \oplus f_{14}(c, \mathcal{K}_{14}) \oplus f_{16}(c, \mathcal{K}_{16}), \mathcal{K}_{13} \cup \mathcal{K}_{14} \cup \mathcal{K}_{16}\}$$

be the merged table, and we can compute T_A with 2^{56} time complexity.

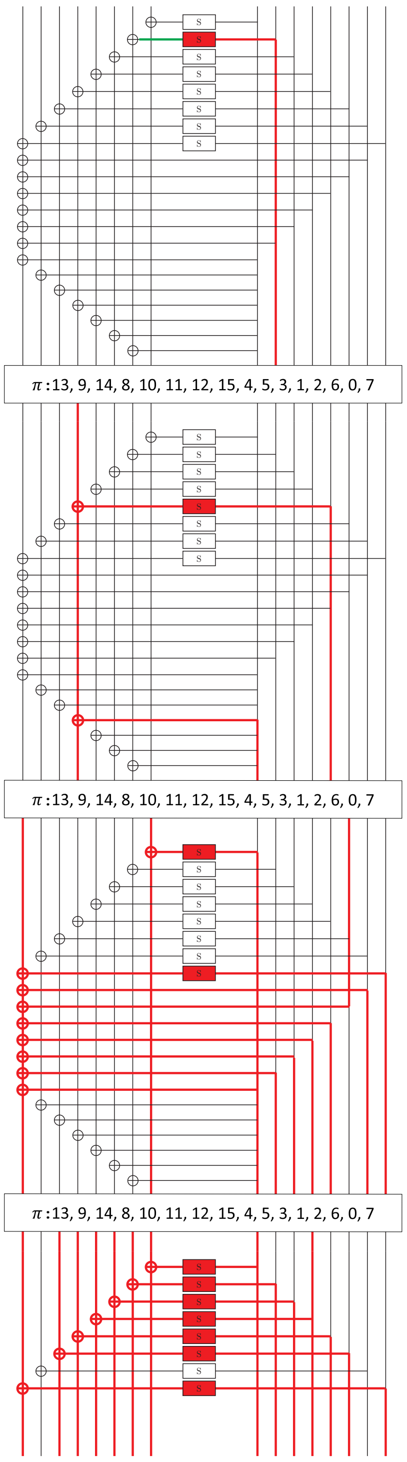
Finally, we search for round keys satisfying

$$\bigoplus f_{13}(c, \mathcal{K}_{13}) \oplus f_{14}(c, \mathcal{K}_{14}) \oplus f_{16}(c, \mathcal{K}_{16}) = \bigoplus f_{15}(c, \mathcal{K}_{15}) \oplus L(c) \tag{7.2}$$

from T_A and T_{15} by using the meet-in-the-middle technique [SW12b]. In the meet-in-the-middle technique, we can compute round keys with about time complexity $\max\{2^{56}, 2^{48}\} = 2^{56}$. Note that $\mathcal{K}_{13} \cup \mathcal{K}_{14} \cup \mathcal{K}_{15} \cup \mathcal{K}_{16}$ is 72 bits, and the probability that Eq. (7.2) randomly holds is 2^{-4}. Therefore, we reduce the space of key candidates from 2^{72} to 2^{68}. Finally, we recover the correct key by additionally guessing the remaining 8 bits. It is enough to determine the correct key by using two known plaintexts. Thus, the total time complexity is $2^{76} \times 2 = 2^{77}$.

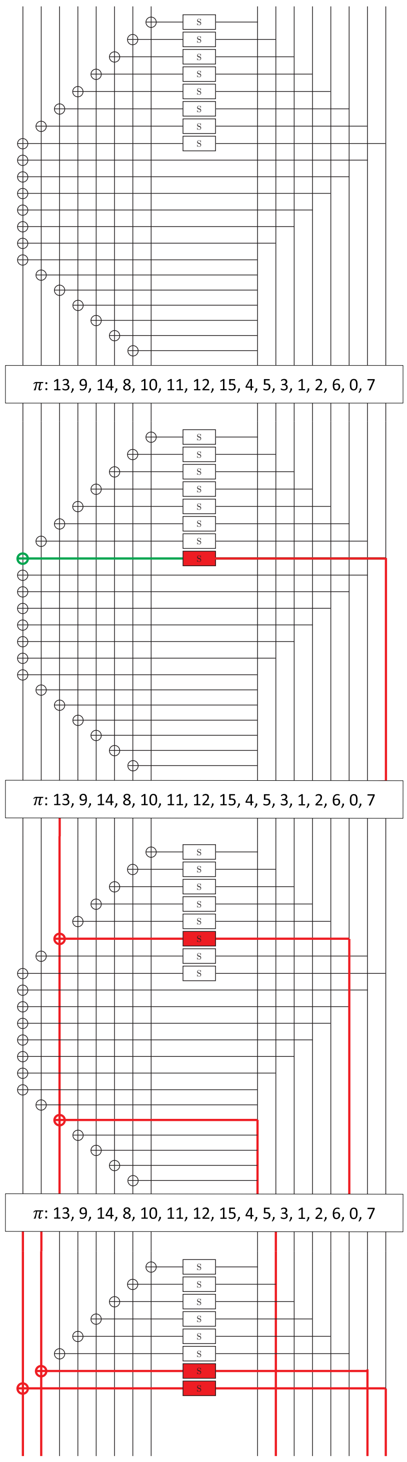
7.5 Conclusion

This chapter proposed the integral cryptanalysis on a lightweight block cipher Lilliput. The designers also evaluated the integral characteristic using the propagation of the integral property. Thanks to the linear layer $\mathcal{L}$ to achieve faster diffusion, the number of rounds in the found integral characteristic was 9 rounds. Therefore, the designers expected that the advantage from the GFN is 6(=15-9) rounds. On the other hand, if we focus on the upper bound of the algebraic degree, the security is not likely to dramatically improve because $\mathcal{L}$ does not increase the algebraic degree. The division property is generalized from the integral property so that can also exploit the algebraic degree. As a result, the division property can find 13-round integral characteristics, and the true contribution by $\mathcal{L}$ is at most 3(=16-13) rounds. Moreover, this 13-round integral distinguisher leads to a 17-round attack, which is a current best attack against Lilliput.



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

Figure 7.3: Evaluation of $f_{13}(c, \mathcal{K}_{13})$.



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

Figure 7.4: Evaluation of $f_{14}(c, \mathcal{K}_{14})$.

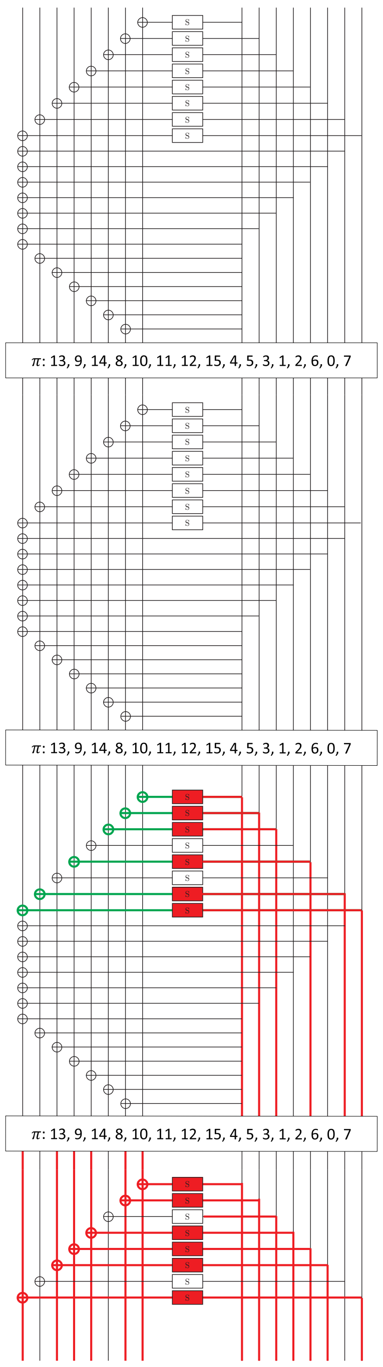

π: 13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7
π: 13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7
π: 13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7

Figure 7.5: Evaluation of $f_{15}(c, \mathcal{K}_{15})$.

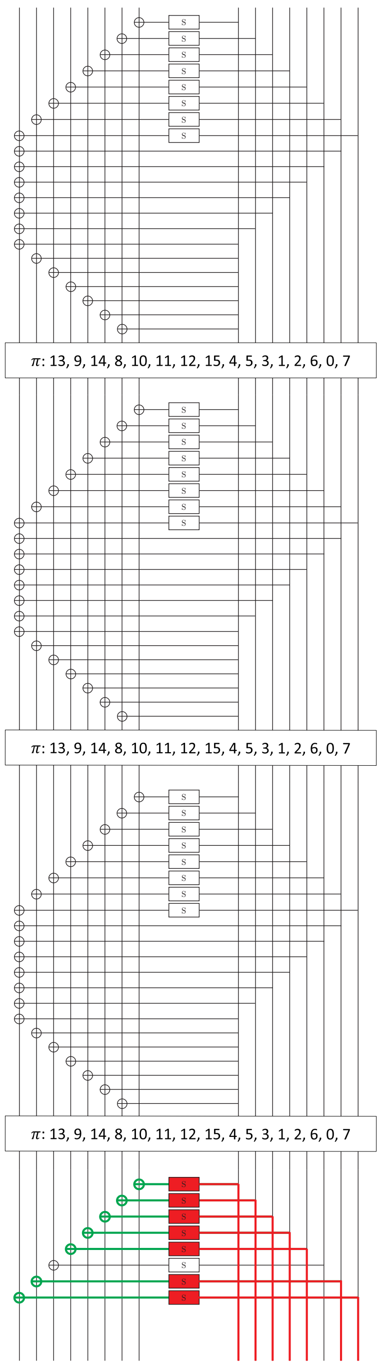

π: 13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7
π: 13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7
π: 13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7

Figure 7.6: Evaluation of $f_{16}(c, \mathcal{K}_{16})$.

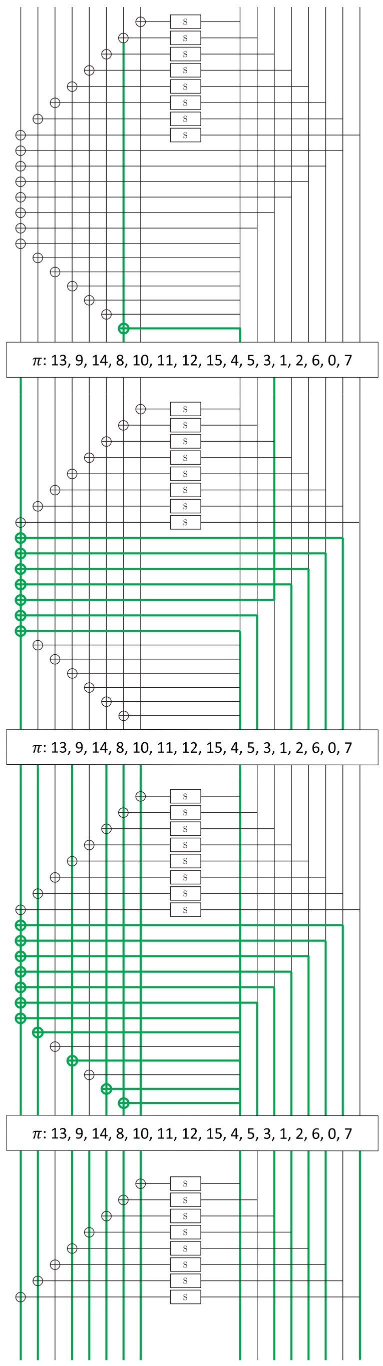

S
π: 13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7
π: 13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7
π: 13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7

Figure 7.7: Evaluation of Linear Part.

Chapter 8

Bit-Based Division Property and Application to Simon Family

Abstract– Ciphers that do not use S-boxes have been discussed for the demand on lightweight cryptosystems, and their round functions consist of `and`, `rotation`, and `xor`. Especially, the Simon family is one of the most famous such ciphers, and there are many cryptanalyses against the Simon family. However, it is very difficult to guarantee the security because we cannot use useful techniques for S-box-based ciphers. Very recently, the division property, which is a new technique to find integral characteristics, was shown in EUROCRYPT 2015. The technique is powerful for S-box-based ciphers, and it was used to break, for the first time, the full MISTY1 in CRYPTO 2015. However, it has not been applied to non-S-box-based ciphers like the Simon family effectively, and only the existence of the 10-round integral characteristic on Simon32 was proven. On the other hand, the experimental characteristic, which possibly does not work for all keys, covers 15 rounds, and there is a 5-round gap. To fill the gap, we introduce a bit-based division property, and we apply it to show that the experimental 15-round integral characteristic always works for all keys. Though the bit-based division property finds more accurate integral characteristics, it requires much time and memory complexity. As a result, we cannot apply it to symmetric-key ciphers whose block length is over 32. Therefore, we alternatively propose a method for designers. The method works for ciphers with large block length, and it shows "provable security" against integral cryptanalyses using the division property. We apply this technique to the Simon family and show that Simon48, 64, 96, and 128 probably do not have 17-, 20-, 25-, and 29-round integral characteristics, respectively.
Keywords– Integral cryptanalysis, Division property, Provable security, Simon family

8.1 Introduction

Non-S-box-based ciphers have been proposed for the demand on lightweight cryptosystems [AJN15, BSS+13]. Such ciphers are superior in lightweight environments because they are implemented by logical operations and do not have a lookup table like S-boxes. In 2013, the NSA proposed a lightweight block cipher family, called the Simon family, that follows this design principle [BSS+13]. However, it is too difficult to guarantee the security against several cryptanalyses because we cannot use many useful techniques for S-box-based ciphers. Therefore, many cryptanalyses have been proposed against the Simon family, e.g., [ALLW14, BRV14, BNS14, KLT15, SHW+14, WLV+14], and the designers recently summarized cryptanalyses in [BSS+15]. In this chapter, we investigate the security of non-S-box-based ciphers against integral cryptanalyses and illustrate our methods on the Simon family.

Division Property. Very recently, the division property, which is a new technique to find integral characteristics [KW02], was proposed in [Tod15b]. The new technique permitted us to find a 6-round integral characteristic on MISTY1, leading to the first complete theoretical cryptanalysis of the full MISTY1 [Tod15a]. Please refer to Chap. 5 and 6 in detail. Moreover, this technique was applied to generalized Feistel structures in [ZW15], leading to improved integral cryptanalyses against LBlock and TWINE.

Table 8.1: Integral characteristics on SIMON32

Methods	#Rounds	Balanced bit (right half)	Reference
Experiment (no proof)	15	`(?b??,????,b???,???b)`	[WLV+14]
Division	10	`(bbbb,bbbb,bbbb,bbbb)`	[Tod15b], Chap. 5
Conventional bit-based division	14	`(bbbb,bbbb,bbbb,bbbb)`	Sect. 8.3
Bit-based division using 3 subsets	15	`(?b??,????,b???,???b)`	Sect. 8.4

Table 8.2: Provable secure number of rounds for the SIMON family

Ciphers	SIMON48	SIMON64	SIMON96	SIMON128	reference
Vulnerable number	14 rounds	17 rounds	21 rounds	25 rounds	[ZWW15]
Provable security	17 rounds	20 rounds	25 rounds	29 rounds	this chapter

The division property also proves integral characteristics on the SIMON family in [Tod15b], and SIMON32, 48, 64, 96, and 128 have 9-, 11-, 11-, 13-, 13-round integral characteristics, respectively[1]. However, the round function is regarded as any function of degree 2. Therefore, we can expect that integral characteristics can be extended to more rounds if one is able to exploit the concrete structure of the round function. In fact, the experimental integral characteristic, which possibly does not work for all keys, covers 15 rounds [WLV+14], and there is a large gap between the proved characteristic and experimental one.

Our Contribution. The round function of the SIMON family is regarded as any function of degree 2 in Chap. 5 because we cannot decompose the round function into several sub blocks like S-boxes. However, we can decompose the round function into every bit, and we call the division property that focuses on every bit a *bit-based division property.*

First, we apply the conventional bit-based division property to SIMON32, which is not against the definition of the division property. Therefore, we can directly use the propagation rules of the division property. As a result, the conventional bit-based division property proves that SIMON32 has a 14-round integral characteristic. However, there is still a gap of one round between the proof and experiment. Namely, this means that either the experimental 15-round characteristic does not work for all keys or the conventional bit-based division property cannot find the accurate characteristic. As a result, we conclude that the conventional bit-based division property is insufficient to find the accurate characteristic. The conventional division property divides the set of $\vec{u}$ according to whether the parity is 0 or unknown. However, we should divide the set of $\vec{u}$ according to whether the parity is 0, 1, or unknown because we can also exploit the fact that the parity is not only 0 but also 1. To exploit this fact, we newly introduce a variant of the bit-based division property, which divides the set of $\vec{u}$ into three subsets. Since the variant is completely different from the definition of the conventional division property, we show the propagation characteristic also. Finally, we apply the variant to SIMON32 and show that the experimental 15-round characteristic always works for all keys. The proved characteristic is the completely same as the experimental one including the position of balanced bits. Table 8.1 shows the comparison of integral characteristics, where balanced and unknown bits are labeled as `b` and `?`, respectively.

Although the bit-based division property can find more accurate integral characteristics, their propagations require much time and memory complexity. When we evaluate the propagation for n-bit block ciphers, it roughly requires 2^n complexity because the bit-based division property has to manage the set of n-dimensional vectors whose elements take values in $\mathbb{F}_2$. This is feasible for SIMON32 because the block length is 32 bits, but it is infeasible for other SIMON family members. Therefore, we introduce a new technique, which is useful for designers but is not useful for attackers. We call this technique a *lazy propagation*, where we evaluate only a part of all propagations. The lazy propagation cannot find the integral characteristic, but it can evaluate the number of rounds that the bit-based division property cannot find integral characteristics even if we can evaluate the accurate propagation. Namely, the technique shows "provable security" for the integral cryptanalysis using the division property, and we expect that it becomes a useful

[1] Since the round key is XORed after the round function in SIMON, we can trivially get one-round extended integral characteristics.

technique for designers. Our provable security guarantees the security against only the integral cryptanalysis using the division property, and it does not always guarantee the security against all integral-like cryptanalyses. However, for SIMON32, the bit-based division property can find the accurate integral characteristic. Therefore, we expect that it also finds the best integral characteristic for the other SIMON family if it is feasible. Table 8.2 shows the number of rounds of SIMON48, 64, 96, and 128, where the division property never finds integral characteristics. As a result, we expect that SIMON48, 64, 96, and 128 do not have 17-, 20-, 25-, and 29-round integral characteristics, respectively[2]. Moreover, as the comparison, Table 8.2 also shows the number of rounds that SIMON48, 64, 96, and 128 have integral characteristics [ZWW15].

8.2 Preliminaries

8.2.1 Notations

We make the distinction between the addition of $\mathbb{F}_2^n$ and addition of $\mathbb{Z}$, and we use $\oplus$ and $+$ as the addition of $\mathbb{F}_2^n$ and addition of $\mathbb{Z}$, respectively. For any $a \in \mathbb{F}_2^n$, the ith element is expressed in $a[i]$, and the Hamming weight $w(a)$ is calculated as $w(a) = \sum_{i=1}^{n} a[i]$. For any $\vec{a} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$, the vectorial Hamming weight of $\vec{a}$ is defined as $W(\vec{a}) = (w(a_1), w(a_2), \ldots, w(a_m)) \in \mathbb{Z}^m$. Moreover, for any $\vec{k} \in \mathbb{Z}^m$ and $\vec{k}' \in \mathbb{Z}^m$, we define $\vec{k} \succeq \vec{k}'$ if $k_i \geq k_i'$ for all i. Otherwise, $\vec{k} \not\succeq \vec{k}'$. In this chapter, we often treat the set of $\vec{k}$, and $\mathbb{K}$ denotes this set. Then, let $|\mathbb{K}|$ be the number of vectors. We simply write $\mathbb{K} \leftarrow \vec{k}$ when $\mathbb{K} := \mathbb{K} \cup \{\vec{k}\}$. Moreover, we simply write $\mathbb{K} \xleftarrow{x} \vec{k}$, where the new $\mathbb{K}$ computed as

$$\mathbb{K} := \begin{cases} \mathbb{K} \cup \{\vec{k}\} & \text{if the original } \mathbb{K} \text{ does not include } \vec{k}, \\ \mathbb{K} \setminus \{\vec{k}\} & \text{if the original } \mathbb{K} \text{ includes } \vec{k}. \end{cases}$$

8.2.2 Integral Attack

The integral attack was first introduced by Daemen et al. to evaluate the security of SQUARE [DKR97], and then it was formalized by Knudsen and Wagner [KW02]. Attackers first prepare N chosen plaintexts and encrypt them R rounds. If the XOR of all encrypted texts becomes 0, we say that the cipher has an R-round integral characteristic with N chosen plaintexts. Finally, we analyze the entire cipher by using the integral characteristic. Therefore, it is very important to find integral characteristic. There are two main approaches to find integral characteristics. The first one is the propagation of the integral property [KW02] and the second one is based on the degree estimation [Knu94, Lai94].

8.2.3 Division Property

The division property, which was proposed in [Tod15b], is a new method to find integral characteristics. This section briefly shows the definition and propagation rules. Please refer to Chap. 5 in detail.

Bit Product Function. The division property of a multiset is evaluated by using the bit product function defined as follows. Let $\pi_u : \mathbb{F}_2^n \to \mathbb{F}_2$ be a bit product function for any $u \in \mathbb{F}_2^n$. Let $x \in \mathbb{F}_2^n$ be the input, and $\pi_u(x)$ is the AND of $x[i]$ satisfying $u[i] = 1$, i.e., it is defined as

$$\pi_u(x) := \prod_{i=1}^{n} x[i]^{u[i]}.$$

Notice that $x[i]^1 = x[i]$ and $x[i]^0 = 1$. Let $\pi_{\vec{u}} : (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m}) \to \mathbb{F}_2$ be a bit product function for any $\vec{u} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. Let $\vec{x} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$ be the input, and $\pi_{\vec{u}}(\vec{x})$ is defined as

$$\pi_{\vec{u}}(\vec{x}) := \prod_{i=1}^{m} \pi_{u_i}(x_i).$$

[2] If we truly guarantee the security against integral attack, we have to consider the key recovery part.

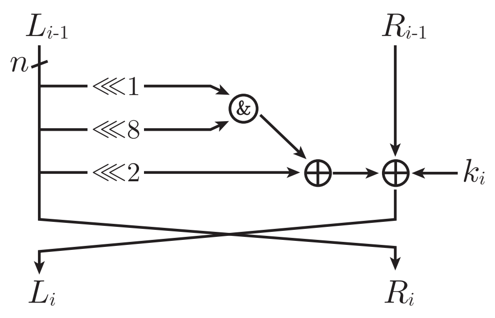


Figure 8.1: Round function of SIMON$2n$

The bit product function also appears in the Algebraic Normal Form (ANF) of a Boolean function. The ANF of a Boolean function f is represented as

$$f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^f \left(\prod_{i=1}^{n} x[i]^{u[i]} \right) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^f \pi_u(x),$$

where $a_u^f \in \mathbb{F}_2$ is a constant value depending on f and u.

Definition of Division Property.

Definition 5.3 (Division Property)**.** *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_1,n_2,\ldots,n_m}$, where $\mathbb{K}$ denotes a set of m-dimensional vectors whose elements take a value between 0 and n_i, it fulfills the following conditions:*

$$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} \textit{unknown} & \textit{if there are } \vec{k} \in \mathbb{K} \textit{ s.t. } W(\vec{u}) \succeq \vec{k}, \\ 0 & \textit{otherwise.} \end{cases}$$

In this chapter, the division property for $(\mathbb{F}_2^n)^m$ is referred to as $\mathcal{D}_{\mathbb{K}}^{n^m}$ for the simplicity. If there are $\vec{k} \in \mathbb{K}$ and $\vec{k}' \in \mathbb{K}$ satisfying $\vec{k} \succeq \vec{k}'$, $\vec{k}$ can be removed from $\mathbb{K}$ because it is redundant. Assume that the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_1,n_2,\ldots,n_m}$. If there is no unit vector $\vec{e}_j$ in $\mathbb{K}$, where $\vec{e}_j$ is a vector whose jth element is 1 and the others are 0, $\bigoplus_{x \in \mathbb{X}} x_j$ is 0. See Chap. 5 to better understand the concept in detail, and [SHZ+15] and [Tod15a] help readers understand the division property.

8.2.4 SIMON Family

The SIMON family is a lightweight block cipher family [BSS+13] based on the Feistel construction. Let SIMON$2n$ be the SIMON block ciphers with $2n$-bit block length, where n is chosen from 16, 24, 32, 48, and 64. Moreover, SIMON$2n$ with mn-bit secret key is referred to as SIMON$2n/mn$. Since we only care about integral characteristics on the SIMON family, this chapter only uses SIMON$2n$.

The output of the ith round function is denoted by (L_i, R_i) and is calculated as

$$(L_i, R_i) = (L_{i-1}^{\lll 1} \wedge L_{i-1}^{\lll 8}) \oplus L_{i-1}^{\lll 2} \oplus R_{i-1} \oplus k_i, L_{i-1}),$$

where $L^{\lll j}$ denotes the j-bit left rotation of L, and k_i denotes the ith round key. Moreover, (L_0, R_0) denotes a plaintext. The round function consists of `and`, `rotation`, and `xor`, and Fig. 8.1 shows the round function. For more details, please refer to [BSS+13].

8.2.5 Known Integral Characteristic on SIMON Family

It is difficult to find effective integral characteristics on ciphers which consist of `and`, `rotation`, and `xor`. In [WLV+14], authors experimentally showed that SIMON32 has the 15-round integral characteristic with 2^{31} chosen plaintexts. Since their characteristic is confirmed under 2^{13} secret keys, they expected that the success probability of this characteristic is at least $1 - 2^{-13}$. Therefore, this approach does not guarantee that the characteristic works for all secret keys. Moreover, it is practically infeasible to find integral characteristics of other SIMON family members because the block length is too large for proceeding to an experimental evaluation.

Integral characteristics proved under all secret keys are shown in Chap. 5, but in this approach the round function of SIMON$2n$ is seen as any n-bit function of degree 2. Therefore, the detailed structure of the round function is not exploited. As a result, it shows that SIMON32, 48, 64, 96, and 128 has 9-, 11-, 11-, 13-, and 13-round integral characteristic, respectively. Since the round key is XORed after the round function, we can trivially get one-round extended integral characteristics using the same technique in [WLV+14]. Therefore, 10-, 12-, 12-, 14-, and 14-round integral characteristics are proved in SIMON32, 48, 64, 96, and 128, respectively. Thus, there is a 5-round gap between the proved characteristic and experimental one.

8.3 Conventional Bit-Based Division Property

When n-bit block ciphers are analyzed, the conventional division property uses $\mathcal{D}_{\mathbb{K}}^{\ell_1,\ell_2,\ldots,\ell_m}$, where ℓ_i and m are chosen by attackers in the range of $n = \sum_{i=1}^{m} \ell_i$. This section considers the conventional bit-based division property, i.e., $\mathcal{D}_{\mathbb{K}}^{1^n}$. Here, we summarize three propagation rules for the bit-based division property as follows. Note that these rules are trivially proven from the propagation rules shown in Sect. 8.2.3 because it is not against the definition of the conventional division property.

Proposition 8.1 (Copy for $\mathcal{D}_{\mathbb{K}}$). *Let F be a copy function, where the input $(x_1, x_2, \ldots, x_m)$ takes values of $\mathbb{F}_2^m$, and the output is calculated as $(x_1, x_1, x_2, x_3, \ldots, x_m)$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and output multiset, respectively. Assuming that $\mathbb{X}$ has $\mathcal{D}_{\mathbb{K}}^{1^m}$, $F(\mathbb{X})$ has $\mathcal{D}_{\mathbb{K}'}^{1^{m+1}}$, where $\mathbb{K}'$ is computed as*

$$\mathbb{K}' \leftarrow \begin{cases} (0, 0, k_2, \ldots, k_m), & \text{if } k_1 = 0, \\ (1, 0, k_2, \ldots, k_m), (0, 1, k_2, \ldots, k_m), & \text{if } k_1 = 1, \end{cases}$$

from all $\vec{k} \in \mathbb{K}$.

Proposition 8.2 (AND for $\mathcal{D}_{\mathbb{K}}$). *Let F be a function compressed by an AND, where the input $(x_1, x_2, \ldots, x_m)$ takes values of $\mathbb{F}_2^m$, and the output is calculated as $(x_1 \wedge x_2, x_3, \ldots, x_m)$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and output multiset, respectively. Assuming that $\mathbb{X}$ has $\mathcal{D}_{\mathbb{K}}^{1^m}$, $F(\mathbb{X})$ has $\mathcal{D}_{\mathbb{K}'}^{1^{m-1}}$, where $\mathbb{K}'$ is computed from all $\vec{k} \in \mathbb{K}$ as*

$$\mathbb{K}' \leftarrow \left(\left\lceil \frac{k_1 + k_2}{2} \right\rceil, k_3, k_4, \ldots, k_m \right).$$

Proposition 8.3 (XOR for $\mathcal{D}_{\mathbb{K}}$). *Let F be a function compressed by an XOR, where the input $(x_1, x_2, \ldots, x_m)$ takes values of $\mathbb{F}_2^m$, and the output is calculated as $(x_1 \oplus x_2, x_3, \ldots, x_m)$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and output multiset, respectively. Assuming that $\mathbb{X}$ has $\mathcal{D}_{\mathbb{K}}^{1^m}$, $F(\mathbb{X})$ has $\mathcal{D}_{\mathbb{K}'}^{1^{m-1}}$, where $\mathbb{K}'$ is computed from all $\vec{k} \in \mathbb{K}$ s.t. $(k_1, k_2) = (0, 0)$, $(1, 0)$, or $(0, 1)$ as*

$$\mathbb{K}' \leftarrow (k_1 + k_2, k_3, k_4, \ldots, k_m).$$

8.3.1 Comparison between Conventional Bit-Based Division Property and Solving Algebraic Equations

Before the application to SIMON family, we roughly show the relation between the bit-based division property and the resolution of algebraic equations by brute force. When entire ciphers are represented by algebraic equations, such equations involve both the plaintext and secret key. Therefore, if we solve such equations for an n-bit block cipher with a κ-bit secret key, this roughly requires $2^{\kappa+n}$ complexity. On the other hand, XORing with a constant value does not change the conventional bit-based division property because such XORing is a linear function [Tod15a]. Therefore, the propagation of the conventional bit-based division property does not involve the secret key. It may miss some useful cryptographic properties, but it dramatically reduces the complexity.

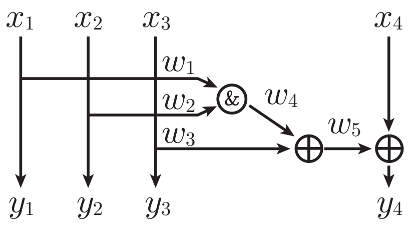


Figure 8.2: Core operation of the SIMON family.

Table 8.3: Propagation of the conventional bit-based division property for the core operation in the SIMON family

Input $\mathcal{D}^{1^4}_{\vec{k}}$	Output $\mathcal{D}^{1^4}_{\mathbb{K}}$
$\vec{k} = [0,0,0,0]$	$\mathbb{K} = \{[0,0,0,0]\}$
$\vec{k} = [1,0,0,0]$	$\mathbb{K} = \{[1,0,0,0],[0,0,0,1]\}$
$\vec{k} = [0,1,0,0]$	$\mathbb{K} = \{[0,1,0,0],[0,0,0,1]\}$
$\vec{k} = [1,1,0,0]$	$\mathbb{K} = \{[1,1,0,0],[0,0,0,1]\}$
$\vec{k} = [0,0,1,0]$	$\mathbb{K} = \{[0,0,1,0],[0,0,0,1]\}$
$\vec{k} = [1,0,1,0]$	$\mathbb{K} = \{[1,0,1,0],[0,0,1,1],[1,0,0,1]\}$
$\vec{k} = [0,1,1,0]$	$\mathbb{K} = \{[0,1,1,0],[0,0,1,1],[0,1,0,1]\}$
$\vec{k} = [1,1,1,0]$	$\mathbb{K} = \{[1,1,1,0],[0,0,1,1],[1,1,0,1]\}$
$\vec{k} = [k_1,k_2,k_3,1]$	$\mathbb{K} = \{[k_1,k_2,k_3,1]\}$

8.3.2 Propagation for Core Operation of SIMON

As an example, we analyze SIMON$2n$ by using the conventional bit-based division property. We focus on only one bit of the right half in SIMON$2n$. The core operation of the round function is represented by Fig. 8.2. Since the input and output bit length is 4 bits, we use the division property $\mathcal{D}^{1^4}_{\mathbb{K}}$.

We consider the propagation characteristic. For instance, let assume that the input multiset has $\mathcal{D}^{1^4}_{[k_1,k_2,k_3,1]}$, where k_i denotes any value, i.e., 0 or 1. Then, if the multiset of $(y_1, y_2, y_3, w_5, x_4)$ has $\mathcal{D}^{1^5}_{[*,*,*,1,1]}$, where $*$ is propagated values, the propagation always abort in the XOR, $x_4 \oplus w_5$. Consequently, the bit-based division property of (y_1, y_2, y_3, y_4) is the same as that of (x_1, x_2, x_3, x_4). On the other hand, assuming that the input multiset has $\mathcal{D}^{1^4}_{[k_1,k_2,k_3,0]}$, the output property is different from the input one.

Let $\mathcal{D}^{1^4}_{\mathbb{K}}$ and $\mathcal{D}^{1^4}_{\mathbb{K}'}$ be the division property of the input and output, respectively. When we get $\mathbb{K}'$ from $\mathbb{K}$, we first independently calculate vectors belonging to $\mathbb{K}'$ by evaluating the propagation from every vector in $\mathbb{K}$. Then, $\mathbb{K}'$ is represented as the union of all calculated vectors. Finally, if there are $\vec{k} \in \mathbb{K}'$ and $\vec{k}' \in \mathbb{K}'$ such that $\vec{k} \succeq \vec{k}'$, $\vec{k}$ is removed from $\mathbb{K}'$ because the vector is redundant.

Table 8.3 summarizes the propagation characteristics from $\mathcal{D}^{1^4}_{\vec{k}}$ to $\mathcal{D}^{1^4}_{\mathbb{K}}$. The round function of SIMON$2n$ repeats the core operation for all n-bit values in the right half. Therefore, we use $\mathcal{D}^{1^{2n}}_{\mathbb{K}}$. In every core operation, we only focus on four bits and evaluate the propagation independent of other $(2n-4)$ bits. Evaluating the propagation is a quite technical task. To help readers understand the propagation, we show a toy example, where we evaluate the propagation on a SIMON-like cipher whose block length is only 8 bits.

Example 8.1 (Propagation of Conventional Bit-Based Division Property.)**.** *To help readers understand the propagation of the bit-based division property, we show the specific propagation on a* SIMON*-like cipher. The block length of the* SIMON*-like cipher is 8 bits, and* (L_{i+1}, R_{i+1}) *is computed as*

$$(L_{i+1}, R_{i+1}) = (L_i^{\lll 1} \wedge L_i) \oplus L_i^{\lll 2} \oplus R_i \oplus k_i, L_i).$$

The left figure in Fig. 8.3 shows the round function.

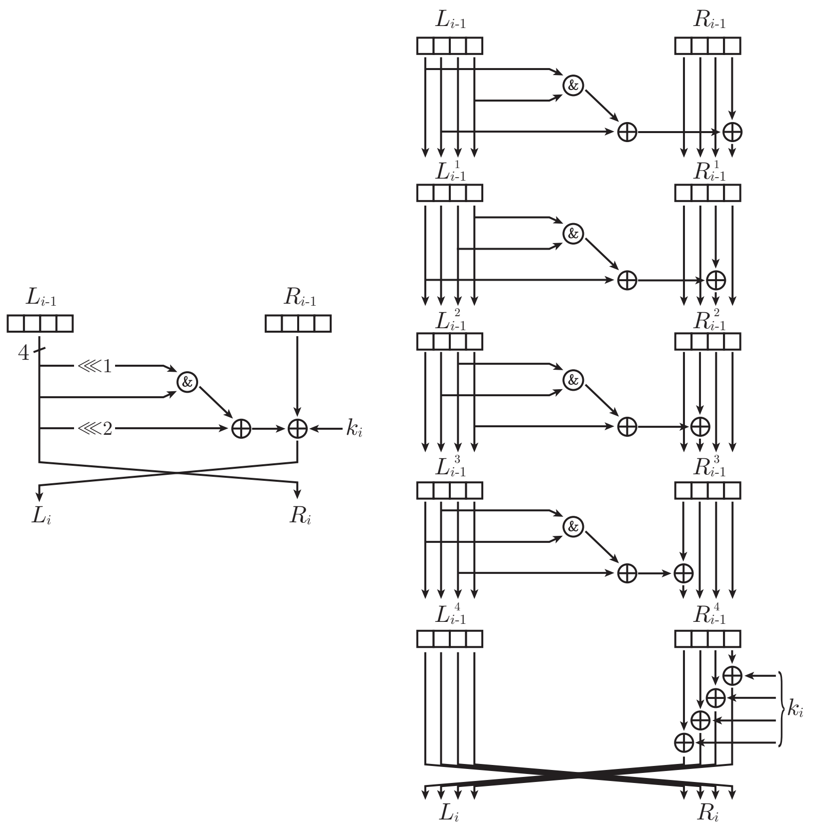


Figure 8.3: Round function of the SIMON-like ciphers

Let us consider the propagation of the bit-based division property $\mathcal{D}^{1^8}_{\mathbb{K}}$. *We evaluate the propagation on the equivalent circuit shown in the right figure in Fig. 8.3. Notice that we can remove XORing with the round key when the conventional bit-based division property is evaluated.*

As an example, let us consider 2^3 *chosen plaintexts that all bits in* L_0 *and the first one bit in* R_0 *is constant and the others are active. Such set has the bit-based division property* $\mathcal{D}^{1^8}_{\{[0,0,0,0,0,1,1,1]\}}$. *Hereinafter,* $[0,0,0,0,0,1,1,1]$ *is referred to as* $[00000111]$ *for the simplicity.*

Round 1 *Now, the bit-based division property of* (L_0, R_0) *is* $\mathcal{D}^{1^8}_{\{[00000111]\}}$.

Step 1 *Since the 8th bit of* $[00000111]$ *is one, there is no change in* $\mathbb{K}$.

Step 2 *Since the 7th bit of* $[00000111]$ *is one, there is no change in* $\mathbb{K}$.

Step 3 *Since the 6th bit of* $[00000111]$ *is one, there is no change in* $\mathbb{K}$.

Step 4 *Since the 1st, 2nd, and 4th bits of* $[00000111]$ *are zero, there is no change in* $\mathbb{K}$.

After the swapping, the bit-based division property of (L_1, R_1) *is* $\mathcal{D}^{1^8}_{\{[01110000]\}}$.

Round 2 *Now, the bit-based division property of* (L_1, R_1) *is* $\mathcal{D}^{1^8}_{\{[01110000]\}}$.

Step 1 *The propagation from* $[01110000]$ *generates three vectors as*

$$[01110000] \Rightarrow [01110000], [01100001], [00110001].$$

Step 2 *From every vector, the following six vectors*

$$\begin{aligned}[01110000] &\Rightarrow [01110000], [01000010],\\ [01100001] &\Rightarrow [01100001], [01000011],\\ [00110001] &\Rightarrow [00110001], [00000011],\end{aligned}$$

are generated. Here, $[01000011]$ *is redundant because* $[01000011] \succeq [01000010]$.

Step 3 *From every vector, the following 11 vectors*

$$\begin{aligned}
&[01110000] \Rightarrow [01110000], [00010100], [01100100],\\
&[01000010] \Rightarrow [01000010], [00000110],\\
&[01100001] \Rightarrow [01100001], [00000101],\\
&[00110001] \Rightarrow [00110001], [00010101], [00100101],\\
&[00000011] \Rightarrow [00000011],
\end{aligned}$$

are generated. Here, $[00010101]$ *is redundant because* $[00010101] \succeq [00010100]$. *Moreover,* $[00100101]$ *is redundant because* $[00100101] \succeq [00000101]$.

Step 4 *From every vector, the following 17 vectors*

$$\begin{aligned}
&[01110000] \Rightarrow [01110000], [00111000], [01011000],\\
&[00010100] \Rightarrow [00010100],\\
&[01100100] \Rightarrow [01100100], [00101100], [01001100],\\
&[01000010] \Rightarrow [01000010], [00001010],\\
&[00000110] \Rightarrow [00000110],\\
&[01100001] \Rightarrow [01100001], [00101001], [01001001],\\
&[00000101] \Rightarrow [00000101],\\
&[00110001] \Rightarrow [00110001], [00011001],\\
&[00000011] \Rightarrow [00000011],
\end{aligned}$$

are generated.

After the swapping, the bit-based division property of (L_2, R_2) *is* $\mathcal{D}^{1^8}_{\mathbb{K}'}$, *where* $\mathbb{K}'$ *has 17 vectors.*

After the 2nd round, we similarly evaluate the propagation of the bit-based division property. As a result, the bit-based division property of (L_4, R_4) *is* $\mathcal{D}^{1^8}_{\mathbb{K}}$, *where*

$$\mathbb{K} = \{[00010000], [00100000], [01000000], [10000000], [00000010], [00000100], [00001001]\}.$$

It means that the first one bit and the last one bit of R_4 *are balanced.*

8.3.3 Application to Simon32

We evaluate the propagation characteristic of the conventional bit-based division property on Simon32. We prepare chosen plaintexts such that the first bit is constant and the others are active. Then, the set of chosen plaintexts has the division property $\mathcal{D}^{1^{32}}_{\mathbb{K}}$, where $\mathbb{K} = \{[0, 1, 1, \ldots, 1]\}$. Table 8.4 shows $|\mathbb{K}|$, which is the number of vectors, in every round, where we perfectly remove redundant vectors from $\mathbb{K}$. Moreover, $\min_w(\mathbb{K})$ and $\max_w(\mathbb{K})$ are defined as

$$\min_w(\mathbb{K}) = \min_{\vec{k} \in \mathbb{K}} \left(\sum_{i=1}^{32} w(k_i) \right), \quad \max_w(\mathbb{K}) = \max_{\vec{k} \in \mathbb{K}} \left(\sum_{i=1}^{32} w(k_i) \right).$$

The output of the 14th round function has the division property $\mathcal{D}^{1^{32}}_{\mathbb{K}}$, where $\mathbb{K}$ has 32 distinct vectors whose Hamming weight is one. Therefore, the conventional bit-based division property cannot show whether or not the output of the 14th round function is balanced. On the other hand, the output of the 13th round function has the division property $\mathcal{D}^{1^{32}}_{\mathbb{K}}$, where $\mathbb{K}$ is represented as 16 vectors, whose Hamming weight of the left half is 1 and that of the right half is 0, and 120 $(= \binom{16}{2})$ vectors, whose Hamming weight of the left half is 0 and that of the right half is 2. This division property means that the output of the 13th round function takes the following integral property

`(????,????,????,????, bbbb,bbbb,bbbb,bbbb)`,

where balanced and unknown bits are labeled as `b` and `?`, respectively.

Table 8.4: Propagation of the bit-based division property on SIMON32

#rounds	$\lvert\mathbb{K}\rvert$	$\min_w(\mathbb{K})$	$\max_w(\mathbb{K})$
0 (plaintexts)	1	31	31
1	1	31	31
2	3	30	31
3	11	29	31
4	65	27	31
5	774	24	31
6	18165	19	31
7	587692	14	30
8	5191387	8	25
9	1595164	5	18
10	95768	3	14
11	5894	2	6
12	682	2	4
13	136	1	2
14	32	1	1

Round keys are XORed with the right half only after the round function is applied to the left half in the SIMON family. Then, when a plaintext is represented as $(L_0, F(L_0) \oplus R_0)$, texts encrypted by one round are represented as

$$\begin{aligned} L_1 &= F(L_0) \oplus k_1 \oplus F(L_0) \oplus R_0 = R_0 \oplus k_1, \\ R_1 &= L_1. \end{aligned}$$

Therefore, we can easily get a 14-round integral characteristic from the 13-round one. The same technique is used in [WLV+14]. Therefore, we conclude that 14-round SIMON32 has the integral characteristic with 2^{31} chosen plaintexts.

8.4 Bit-Based Division Property using Three Subsets

8.4.1 Motivation

The conventional bit-based division property proved the existence of the 14-round integral characteristic of SIMON32. However, the experimental characteristic covers 15 rounds [WLV+14], and there is still a one-round gap between the experiment and proof. In [WLV+14], the authors experimentally confirmed the characteristic by randomly choosing 2^{13} secret keys. Therefore, they concluded that the success probability of the characteristic is at least $1 - 2^{-13}$. Thus, we consider that this gap is derived from either the experimental result does not work for all keys or the conventional bit-based division property cannot find the accurate characteristic.

We first show that the conventional bit-based division property is insufficient to find integral characteristics on SIMON32, and we then introduce a new variant of the bit-based division property. The conventional bit-based division property focuses on that the parity $\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x})$ is 0 or unknown. On the other hand, the new variant focuses on that the parity $\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x})$ is 0, 1, or unknown. Therefore we call the new variant the *bit-based division property using three subsets*. The new variant can find more accurate integral characteristics and prove that the experimental characteristic shown in [WLV+14] works for all keys.

8.4.2 Characteristic that Conventional Bit-Based Division Property cannot Find

The conventional division property divides the set of $\vec{u}$ according to whether the parity is 0 or unknown. However, it sometimes overlooks useful characteristics. We show it by using a simple example.

We again evaluate the propagation of the conventional bit-based division property for the circuit in Fig 8.2, and $F : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ denotes the circuit. Moreover, let $\mathbb{X}$ and $F(\mathbb{X})$ be the input and output multiset, respectively. Assuming that $\mathbb{X}$ has $\mathcal{D}^{1^4}_{\{[1,1,0,0],[0,0,1,0]\}}$, $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[1,1,0,0]}(\vec{x})$ and $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[0,0,1,0]}(\vec{x})$ are unknown. Then, the output multiset $F(\mathbb{X})$ has $\mathcal{D}^{1^4}_{\{[1,1,0,0],[0,0,1,0],[0,0,0,1]\}}$ from Table 8.3.

Let us assume that both $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[1,1,0,0]}(\vec{x})$ and $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[0,0,1,0]}(\vec{x})$ are 1. Even if we know the parity is always one, the division property of $\mathbb{X}$ is $\mathcal{D}^{1^4}_{\{[1,1,0,0],[0,0,1,0]\}}$. However, we can get the following equation.

$$\begin{aligned}
\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[0,0,0,1]}(F(\vec{x})) &= \bigoplus_{\vec{x}\in\mathbb{X}} (x_1x_2 \oplus x_3 \oplus x_4) \\
&= \bigoplus_{\vec{x}\in\mathbb{X}} (x_1x_2) \bigoplus_{\vec{x}\in\mathbb{X}} (x_3) \bigoplus_{\vec{x}\in\mathbb{X}} (x_4) \\
&= \bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[1,1,0,0]}(\vec{x}) \bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[0,0,1,0]}(\vec{x}) \bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[0,0,0,1]}(\vec{x}) \\
&= 1 \oplus 1 \oplus 0 = 0.
\end{aligned}$$

Therefore, $\oplus_{\vec{x}\in\mathbb{X}} \pi_{[0,0,0,1]}(F(\vec{x}))$ is always 0 not unknown, and the division property of $F(\mathbb{X})$ is $\mathcal{D}^{1^4}_{\{[1,1,0,0],[0,0,1,0],[0,1,0,1],[1,0,0,1]\}}$ not $\mathcal{D}^{1^4}_{\{[1,1,0,0],[0,0,1,0],[0,0,0,1]\}}$.

Since the conventional division property focuses on the case the parity is 0, it cannot find characteristics that appear by cancelling like the above example. Therefore, we newly introduce a variant of the bit-based division property to exploit this fact. The variant divides the set of $\vec{u}$ into three subsets, i.e., 0, 1, and unknown.

8.4.3 Definition of Bit-Based Division Property using Three Subsets

The conventional division property uses the set $\mathbb{K}$ to represent the subset of $\vec{u}$ such that $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(\vec{x})$ is unknown. The bit-based division property using three subsets needs to represent not only the subset of $\vec{u}$ such that $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(\vec{x})$ is unknown but also the subset of $\vec{u}$ such that $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(\vec{x})$ is one. Therefore, we use the set $\mathbb{K}$ to represent the subset of $\vec{u}$ such that $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(\vec{x})$ is unknown, and we also use the set $\mathbb{L}$ to represent the subset of $\vec{u}$ such that $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(\vec{x})$ is one.

Definition 8.1 (Bit-based division property using three subsets)**.** *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2)^m$, and $\vec{k}$ is an m-dimensional vector whose ith element takes 0 or 1. When the multiset $\mathbb{X}$ has the bit-based division property using three subsets $\mathcal{D}^{1^m}_{\mathbb{K},\mathbb{L}}$, it fulfils the following conditions:*

$$\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} \text{unknown} & \textit{if there are } \vec{k}\in\mathbb{K} \textit{ s.t. } W(\vec{u}) \succeq \vec{k}, \\ 1 & \textit{else if there is } \vec{\ell}\in\mathbb{L} \textit{ s.t. } W(\vec{u}) = \vec{\ell}, \\ 0 & \textit{otherwise.} \end{cases}$$

If there are $\vec{k}\in\mathbb{K}$ and $\vec{k}'\in\mathbb{K}$ satisfying $\vec{k}\succeq\vec{k}'$, $\vec{k}$ can be removed from $\mathbb{K}$ because the vector $\vec{k}$ is redundant. Moreover, when there is $\vec{k}\in\mathbb{K}$ satisfying $W(\vec{u})\succeq\vec{k}$, $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(\vec{x})$ is unknown even if there is $\vec{\ell}\in\mathbb{L}$ satisfying $W(\vec{u}) = \vec{\ell}$. Therefore, if there are $\vec{\ell}\in\mathbb{L}$ and $\vec{k}\in\mathbb{K}$ satisfying $\vec{\ell}\succeq\vec{k}$, the vector $\vec{\ell}$ is redundant. Notice that redundant vectors in $\mathbb{K}$ and $\mathbb{L}$ do not affect whether $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(\vec{x})$ is 0, 1, or unknown for any $\vec{u}$.

Example 8.2. *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2)^4$. Assume the multiset $\mathbb{X}$ has the bit-based division property $\mathcal{D}^{1^4}_{\mathbb{K},\mathbb{L}}$, where $\mathbb{K} = \{[0,0,0,1],[0,1,1,0]\}$ and $\mathbb{L} = \{[1,0,0,0],[1,0,1,0],[0,0,1,0],[0,0,1,1]\}$. Then, every parity satisfies the following, where the value of $\vec{u}$ is represented as hexadecimal notation of $(u_1\|u_2\|u_3\|u_4)$.*

$\vec{u}$	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
Parity	0	?	1	?	0	?	?	?	1	?	1	?	0	?	?	?

Notice that the parity of $\pi_{[0,0,1,1]}(\vec{x})$ over all $\vec{x}\in\mathbb{X}$ is unknown because there is $[0,0,0,1]\in\mathbb{K}$ and $W([0,0,1,1])\succeq W([0,0,0,1])$. Thus, $[0,0,1,1]\in\mathbb{L}$ is redundant.

8.4.4 Propagation Rules

We show propagation rules for the bit-based division property using three subsets. There rules are very similar to those of the conventional division property. Here, we show three rules, "Copy," "AND," and "XOR," because any Boolean function can be evaluated by using these three rules.

Proposition 8.4 (Copy for $\mathcal{D}_{\mathbb{K},\mathbb{L}}$). *Let F be a copy function, where the input $(x_1, x_2, \ldots, x_m)$ takes values of $(\mathbb{F}_2)^m$, and the output is calculated as $(x_1, x_1, x_2, x_3, \ldots, x_m)$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and output multiset, respectively. Assuming that $\mathbb{X}$ has $\mathcal{D}^{1^m}_{\mathbb{K},\mathbb{L}}$, $F(\mathbb{X})$ has $\mathcal{D}^{1^{m+1}}_{\mathbb{K}',\mathbb{L}'}$, where $\mathbb{K}'$ and $\mathbb{L}'$ are computed as*

$$\mathbb{K}' \leftarrow \begin{cases} (0,0,k_2,\ldots,k_m), & \text{if } k_1 = 0 \\ (1,0,k_2,\ldots,k_m),(0,1,k_2,\ldots,k_m), & \text{if } k_1 = 1 \end{cases},$$

$$\mathbb{L}' \leftarrow \begin{cases} (0,0,\ell_2,\ldots,\ell_m), & \text{if } \ell_1 = 0 \\ (1,0,\ell_2,\ldots,\ell_m),(0,1,\ell_2,\ldots,\ell_m),(1,1,\ell_2,\ldots,\ell_m) & \text{if } \ell_1 = 1 \end{cases}.$$

from all $\vec{k} \in \mathbb{K}$ and all $\vec{\ell} \in \mathbb{L}$, respectively.

Proof. Let F be a copy function, where the input $(x_1, x_2, \ldots, x_m)$ takes values of $\mathbb{F}_2^m$, and the output is calculated as $(x_1, x_1, x_2, x_3, \ldots, x_m)$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and output multiset, respectively. Now, we want to evaluate the parity $\bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ for any $\vec{v} \in \mathbb{F}_2^{m+1}$.

$$\begin{aligned} \bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y}) &= \bigoplus_{\vec{x}\in\mathbb{X}} (\pi_{\vec{v}} \circ F)(\vec{x}) \\ &= \bigoplus_{\vec{x}\in\mathbb{X}} x_1^{v_1} x_1^{v_2} x_2^{v_3} x_3^{v_4} \cdots x_m^{v_{m+1}} \\ &= \bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[v_1 \vee v_2, v_3, \ldots, v_{m+1}]}(\vec{x}). \end{aligned}$$

Assuming that $\mathbb{X}$ has $\mathcal{D}^{1^m}_{\mathbb{K},\mathbb{L}}$, $\pi_{\vec{u}}(\vec{x})$ is satisfied as

$$\bigoplus_{x\in\mathbb{X}} \pi_{\vec{u}}(x) = \begin{cases} \text{unknown} & \text{if there are } \vec{k} \in \mathbb{K} \text{ s.t. } \vec{u} \succeq \vec{k}, \\ 1 & \text{else if there is } \vec{\ell} \in \mathbb{L} \text{ s.t. } \vec{u} = \vec{\ell}, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, $\bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ is unknown if and only if there are $\vec{k} \in \mathbb{K}$ satisfying

$$W([v_1 \vee v_2, v_3, \ldots, v_{m+1}]) \succeq \vec{k}.$$

Thus, the parity is unknown for any $\vec{v}$ satisfying

$$W(\vec{v}) \succeq (1,0,k_2,k_3,\ldots,k_m) \text{ or } W(\vec{v}) \succeq (0,1,k_2,k_3,\ldots,k_m)$$

for all $\vec{k} \in \mathbb{K}$ s.t. $k_1 = 1$. Moreover, the parity is unknown for any $\vec{v}$ satisfying

$$W(\vec{v}) \succeq (0,0,k_2,k_3,\ldots,k_m)$$

for all $\vec{k} \in \mathbb{K}$ s.t. $k_1 = 0$.

Next, if there are not $\vec{k} \in \mathbb{K}$ satisfying $W([v_1 \vee v_2, v_3, \ldots, v_{m+1}]) \succeq \vec{k}$, the parity $\bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ is exactly determined. Then, the parity is 1 if there is $\vec{\ell} \in \mathbb{L}$ satisfying $W([v_1 \vee v_2, v_3, \ldots, v_{m+1}]) = \vec{\ell}$. Otherwise, the parity is 0. □

Proposition 8.5 (AND for $\mathcal{D}_{\mathbb{K},\mathbb{L}}$). *Let F be a function compressed by an AND, where the input $(x_1, x_2, \ldots, x_m)$ takes values of $(\mathbb{F}_2)^m$, and the output is calculated as $(x_1 \wedge x_2, x_3, \ldots, x_m)$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and output multiset, respectively. Assuming that $\mathbb{X}$ has $\mathcal{D}^{1^m}_{\mathbb{K},\mathbb{L}}$, $F(\mathbb{X})$ has $\mathcal{D}^{1^{m-1}}_{\mathbb{K}',\mathbb{L}'}$, where $\mathbb{K}'$ is computed from all $\vec{k} \in \mathbb{K}$ as*

$$\mathbb{K}' \leftarrow \left(\left\lceil \frac{k_1 + k_2}{2} \right\rceil, k_3, k_4, \ldots, k_m \right).$$

Moreover, $\mathbb{L}'$ *is computed from all* $\vec{\ell} \in \mathbb{L}$ *s.t.* $(\ell_1, \ell_2) = (0,0)$ *or* $(1,1)$ *as*

$$\mathbb{L}' \leftarrow \left(\left\lceil \frac{\ell_1 + \ell_2}{2} \right\rceil, \ell_3, \ell_4, \dots, \ell_m \right).$$

Proof. Let F be a non-linear function, where the input $(x_1, x_2, \dots, x_m)$ takes values of $(\mathbb{F}_2)^m$, and the output is calculated as $(x_1 \wedge x_2, x_3, \dots, x_m)$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and output multiset, respectively. Now, we want to evaluate the parity $\bigoplus_{\vec{y} \in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ for any $\vec{v} \in \mathbb{F}_2^{m-1}$.

$$\begin{aligned}
\bigoplus_{\vec{y} \in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y}) &= \bigoplus_{\vec{x} \in \mathbb{X}} (\pi_{\vec{v}} \circ F)(\vec{x}) \\
&= \bigoplus_{\vec{x} \in \mathbb{X}} (x_1 x_2)^{v_1} x_3^{v_2} x_4^{v_3} \cdots x_m^{v_{m-1}} \\
&= \bigoplus_{\vec{x} \in \mathbb{X}} \pi_{[v_1, v_1, v_2, v_3, \dots, v_{m-1}]}(\vec{x}).
\end{aligned}$$

Assuming that $\mathbb{X}$ has $\mathcal{D}^{1^m}_{\mathbb{K},\mathbb{L}}$, $\pi_{\vec{u}}(\vec{x})$ is satisfied as

$$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(x) = \begin{cases} \text{unknown} & \text{if there are } \vec{k} \in \mathbb{K} \text{ s.t. } \vec{u} \succeq \vec{k}, \\ 1 & \text{else if there is } \vec{\ell} \in \mathbb{L} \text{ s.t. } \vec{u} = \vec{\ell}, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, $\bigoplus_{\vec{y} \in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ is unknown if and only if there is $\vec{k} \in \mathbb{K}$ satisfying

$$W([v_1, v_1, v_2, v_3, \dots, v_{m-1}]) \succeq \vec{k}.$$

It means that the parity is unknown for any $\vec{v}$ satisfying

$$W(\vec{v}) \succeq \left(\left\lceil \frac{k_1 + k_2}{2} \right\rceil, k_3, k_4, \dots, k_m \right).$$

Next, if there are not $\vec{k} \in \mathbb{K}$ satisfying

$$W([v_1, v_1, v_2, v_3, \dots, v_{m-1}]) \succeq \vec{k},$$

the parity $\bigoplus_{y \in F(\mathbb{X})} \pi_{\vec{v}}(y)$ is exactly determined. Then, the parity is 1 if there is $\ell \in \mathbb{L}$ satisfying

$$W([v_1, v_1, v_2, v_3, \dots, v_{m-1}]) = \vec{\ell}.$$

Otherwise, the parity is 0. □

Proposition 8.6 (XOR for $\mathcal{D}_{\mathbb{K},\mathbb{L}}$)**.** *Let F be a function compressed by an XOR, where the input $(x_1, x_2, \dots, x_m)$ takes values of $(\mathbb{F}_2)^m$, and the output is calculated as $(x_1 \oplus x_2, x_3, \dots, x_m)$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and output multiset, respectively. Assuming that $\mathbb{X}$ has $\mathcal{D}^{1^m}_{\mathbb{K},\mathbb{L}}$, $F(\mathbb{X})$ has $\mathcal{D}^{1^{m-1}}_{\mathbb{K}',\mathbb{L}'}$, where $\mathbb{K}'$ is computed from all $\vec{k} \in \mathbb{K}$ s.t. $(k_1, k_2) = (0,0)$, $(1,0)$, or $(0,1)$ as*

$$\mathbb{K}' \leftarrow (k_1 + k_2, k_3, k_4, \dots, k_m).$$

Moreover, $\mathbb{L}'$ *is computed from all* $\vec{\ell} \in \mathbb{L}$ *s.t.* $(\ell_1, \ell_2) = (0,0)$, $(1,0)$, *or* $(0,1)$ *as*

$$\mathbb{L}' \xleftarrow{\mathtt{x}} (\ell_1 + \ell_2, \ell_3, \ell_4, \dots, \ell_m).$$

Proof. Let F be a function compressed by an XOR, where the input $(x_1, x_2, \dots, x_m)$ takes values of $\mathbb{F}_2^m$, and the output is calculated as $(x_1 \oplus x_2, x_3, \dots, x_m)$. Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and output multiset, respectively. Now, we want to evaluate the parity $\bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ for any $\vec{v} \in \mathbb{F}_2^{m-1}$.

$$\begin{aligned}
\bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y}) &= \bigoplus_{\vec{x}\in\mathbb{X}} (\pi_{\vec{v}} \circ F)(\vec{x}) \\
&= \bigoplus_{\vec{x}\in\mathbb{X}} (x_1 \oplus x_2)^{v_1} x_3^{v_2} x_4^{v_3} \cdots x_m^{v_{m-1}} \\
&= \bigoplus_{\vec{x}\in\mathbb{X}} x_1^{v_1} x_3^{v_2} x_4^{v_3} \cdots x_m^{v_{m-1}} \oplus x_2^{v_1} x_3^{v_2} x_4^{v_3} \cdots x_m^{v_{m-1}} \\
&= \bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[v_1,0,v_2,v_3,\ldots,v_{m-1}]}(\vec{x}) \bigoplus_{\vec{x}\in\mathbb{X}} \pi_{[0,v_1,v_2,v_3,\ldots,v_{m-1}]}(\vec{x}).
\end{aligned}$$

Assuming that $\mathbb{X}$ has $\mathcal{D}^{1^m}_{\mathbb{K},\mathbb{L}}$, $\pi_{\vec{u}}(\vec{x})$ is satisfied as

$$\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(x) = \begin{cases} \text{unknown} & \text{if there are } \vec{k} \in \mathbb{K} \text{ s.t. } \vec{u} \succeq \vec{k}, \\ 1 & \text{else if there is } \vec{\ell} \in \mathbb{L} \text{ s.t. } \vec{u} = \vec{\ell}, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, $\bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ is unknown if and only if there are $\vec{k} \in \mathbb{K}$ satisfying

$$W([v_1, 0, v_2, v_3, \ldots, v_{m-1}]) \succeq \vec{k} \text{ or } W([0, v_1, v_2, v_3, \ldots, v_{m-1}]) \succeq \vec{k}.$$

It means that the parity is unknown for any $\vec{v}$ satisfying

$$W(\vec{v}) \succeq (k_1 + k_2, k_3, k_4, \ldots, k_m),$$

where $(k_1, k_2) = (0,0)$, $(1,0)$, or $(0,1)$. From $\vec{k} \in \mathbb{K}$ satisfying $(k_1, k_2) = (1,1)$, we cannot get $\vec{v}$ that the parity is unknown.

Next, if there is not $\vec{k} \in \mathbb{K}$ satisfying

$$W([v_1, 0, v_2, v_3, \ldots, v_{m-1}]) \succeq \vec{k} \text{ or } W([0, v_1, v_2, v_3, \ldots, v_{m-1}]) \succeq \vec{k},$$

the parity $\bigoplus_{\vec{y}\in F(\mathbb{X})} \pi_{\vec{v}}(\vec{y})$ is exactly determined. If there is either

$$\vec{\ell}_1 = W([v_1, 0, v_2, v_3, \ldots, v_{m-1}]) \text{ or } \vec{\ell}_2 = W([0, v_1, v_2, v_3, \ldots, v_{m-1}])$$

in $\mathbb{L}$, the parity is 1. If there are both $\vec{\ell}_1$ and $\vec{\ell}_2$ in $\mathbb{L}$, the parity is 0. Otherwise, the parity is 0. □

8.4.5 Dependencies between $\mathbb{K}$ and $\mathbb{L}$

Propagation for Public Function. In the propagation rules shown in Sect. 8.4.4, $\mathbb{K}'$ and $\mathbb{L}'$ are computed from $\mathbb{K}$ and $\mathbb{L}$, respectively. Therefore, we can evaluate the propagation from $\mathbb{K}$ and that from $\mathbb{L}$ independently. However, independent propagations generate many redundant vectors in $\mathbb{K}'$ and $\mathbb{L}'$. Note that redundant vectors in $\mathbb{K}'$ and $\mathbb{L}'$ do not affect whether the parity is 0, 1, or unknown for any $\vec{u}$. Therefore, when we consider the propagation for public functions, we do not need to care about the dependencies between $\mathbb{K}$ and $\mathbb{L}$. On the other hand, if there are many redundant vectors, the propagation requires much time complexity. Therefore, we should remove redundant vectors if possible because of the reason of only complexity.

XORing with Secret Round Key. For the public function, the propagation from $\mathbb{K}$ and that from $\mathbb{L}$ are independently evaluated. However, if the secret round key is XORed, every vector in $\mathbb{L}$ affects $\mathbb{K}$.

Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input and output multiset, respectively. Then, $\vec{y} \in F(\mathbb{X})$ is computed as $\vec{y} = \vec{x} \oplus \vec{rk}$ for $\vec{x} \in \mathbb{X}$, where $\vec{rk}$ is the secret round key. Moreover, let $\mathcal{D}^{1^m}_{\mathbb{K},\mathbb{L}}$ and $\mathcal{D}^{1^m}_{\mathbb{K}',\mathbb{L}'}$ be the bit-based division property using three subsets on $\mathbb{X}$ and $F(\mathbb{X})$, respectively. We want to get $\mathbb{K}'$ and $\mathbb{L}'$ from $\mathbb{K}$ and $\mathbb{L}$. We cannot know the secret round key. Therefore, the parity

$\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{v}}(\vec{x} \oplus \vec{rk})$ satisfying $\vec{v} \succ \vec{\ell}$ is unknown because the parity depends on the secret round key.

In many ciphers, round keys are XORed with a part of entire bits. Assuming a round key is XORed with the ith bit, $\mathbb{K}'$ is computed as

$$\mathbb{K}' \leftarrow (\ell_1, \ell_2, \dots, \ell_i \vee 1, \dots, \ell_m)$$

for all $\vec{\ell} \in \mathbb{L}$ satisfying $\ell_i = 0$.

8.4.6 Propagation for Core Operation of Simon

We search for integral characteristics on Simon32 by the bit-based division property using three subsets. Similar to the conventional bit-based division property, we focus on only one bit of the right half and consider the core operation of the Simon family (see Fig. 8.2).

Table 8.5: Propagation of the bit-based division property using three subsets for the core operation in the Simon family

Input $\mathcal{D}^{1^4}_{\mathbb{K},\{\vec{\ell}\}}$	Output $\mathcal{D}^{1^4}_{\mathbb{K}',\mathbb{L}'}$
$\vec{\ell} = [0,0,0,0]$	$\mathbb{L}' = \{[0,0,0,0]\}$
$\vec{\ell} = [1,0,0,0]$	$\mathbb{L}' = \{[1,0,0,0]\}$
$\vec{\ell} = [0,1,0,0]$	$\mathbb{L}' = \{[0,1,0,0]\}$
$\vec{\ell} = [1,1,0,0]$	$\mathbb{L}' = \{[1,1,0,0],[0,0,0,1],[1,0,0,1],[0,1,0,1],[1,1,0,1]\}$
$\vec{\ell} = [0,0,1,0]$	$\mathbb{L}' = \{[0,0,1,0],[0,0,0,1],[0,0,1,1]\}$
$\vec{\ell} = [1,0,1,0]$	$\mathbb{L}' = \{[1,0,1,0],[1,0,0,1],[1,0,1,1]\}$
$\vec{\ell} = [0,1,1,0]$	$\mathbb{L}' = \{[0,1,1,0],[0,1,0,1],[0,1,1,1]\}$
$\vec{\ell} = [1,1,1,0]$	$\mathbb{L}' = \{[1,1,1,0],[0,0,1,1],[1,0,1,1],[0,1,1,1],[1,1,0,1]\}$
$\vec{\ell} = [\ell_1,\ell_2,\ell_3,1]$	$\mathbb{L}' = \{[\ell_1,\ell_2,\ell_3,1]\}$

The core operation is a public function and it does not involve any secret information. Therefore, we can evaluate the propagation from $\mathbb{K}$ and that from $\mathbb{L}$ independently. Table 8.5 summarizes the propagation characteristics from $\mathcal{D}^{1^4}_{\mathbb{K},\{\vec{\ell}\}}$ to $\mathcal{D}^{1^4}_{\mathbb{K}',\mathbb{L}'}$, where the propagation from $\mathbb{K}$ to $\mathbb{K}'$ is the same as that in Table 8.3. To help readers understand the propagation, we show an example as follows.

Example 8.3 (Example of Propagation of Bit-Based Division Property using Three Subsets).

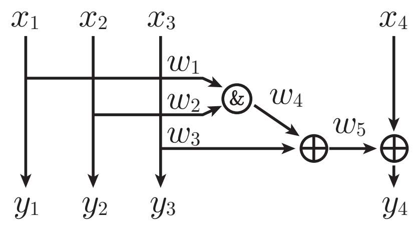


Figure 8.4: Core operation of the Simon family

We show an example of the propagation of the bit-based division property using three subsets. Let F be the core operation of the Simon *family (see 8.4). Let $\mathbb{X}$ and $F(\mathbb{X})$ be the input multiset and the output multiset, respectively.*

Assuming the input multiset has the division property $\mathcal{D}^{1^4}_{\phi,[1,1,1,0]}$. We consider the intermediate state $[y_1, y_2, y_3, w_1, w_2, w_3, x_4]$, and evaluate the division property. From Proposition 8.4, *such state has* $\mathcal{D}^{1^7}_{\phi,\mathbb{L}}$*, where* $\mathbb{L}$ *is represented as*

$$\begin{aligned}
&[1,1,1,0,0,0,0][1,1,0,0,0,1,0][1,1,1,0,0,1,0][1,0,1,0,1,0,0],\\
&[1,0,0,0,1,1,0][1,0,1,0,1,1,0][1,1,1,0,1,0,0][1,1,0,0,1,1,0],\\
&[1,1,1,0,1,1,0][0,1,1,1,0,0,0][0,1,0,1,0,1,0][0,1,1,1,0,1,0],\\
&[0,0,1,1,1,0,0][0,0,0,1,1,1,0][0,0,1,1,1,1,0][0,1,1,1,1,0,0],\\
&[0,1,0,1,1,1,0][0,1,1,1,1,1,0][1,1,1,1,0,0,0][1,1,0,1,0,1,0],\\
&[1,1,1,1,0,1,0][1,0,1,1,1,0,0][1,0,0,1,1,1,0][1,0,1,1,1,1,0],\\
&[1,1,1,1,1,0,0][1,1,0,1,1,1,0][1,1,1,1,1,1,0].
\end{aligned}$$

Next, we consider the intermediate state $[y_1, y_2, y_3, w_4, w_3, x_4]$*, and evaluate the division property. From Proposition 8.5, such state has* $\mathcal{D}^{1^6}_{\phi,\mathbb{L}'}$*, where* $\mathbb{L}'$ *is represented as*

$$\begin{aligned}
&[1,1,1,0,0,0][1,1,0,0,1,0][1,1,1,0,1,0][0,0,1,1,0,0],\\
&[0,0,0,1,1,0][0,0,1,1,1,0][0,1,1,1,0,0][0,1,0,1,1,0],\\
&[0,1,1,1,1,0][1,0,1,1,0,0][1,0,0,1,1,0][1,0,1,1,1,0],\\
&[1,1,1,1,0,0][1,1,0,1,1,0][1,1,1,1,1,0].
\end{aligned}$$

Third, we consider the intermediate state $[y_1, y_2, y_3, w_5, x_4]$*, and evaluate the division property. From Proposition 8.6, such state has* $\mathcal{D}^{1^5}_{\phi,\mathbb{L}''}$*, where* $\mathbb{L}''$ *is represented as*

$$[1,1,1,0,0][1,1,0,1,0][0,0,1,1,0][0,1,1,1,0][1,0,1,1,0].$$

Here, two $[1,1,1,1,0]$ *is propagated from* $[1,1,1,0,1,0]$ *and* $[1,1,1,1,0,0]$*, and* $[1,1,1,1,0]$ *is removed from* $\mathbb{L}$*. Finally, we get* $[y_1, y_2, y_3, y_4]$ *has the division property* $\mathcal{D}^{1^5}_{\phi,\mathbb{L}'''}$*, where* $\mathbb{L}'''$ *is represented as*

$$[1,1,1,0][1,1,0,1][0,0,1,1][0,1,1,1][1,0,1,1].$$

Next, the propagation on the round function can be evaluated by repeating for all bits of the right half. Finally, when round keys are XORed with the right half, new vectors are generated from $\mathbb{L}$, and the new vectors are inserted into $\mathbb{K}$. The following shows a toy example, which is the same as Example 8.1, to help readers understand the propagation.

Example 8.4 (Propagation of Bit-Based Division Property Using Three Subsets.)**.** *To help readers understand the propagation of the bit-based division property using three subsets, we revisit the toy* SIMON*-like cipher shown in Example 8.1 (see Fig. 8.3).*

Let us consider the propagation of the bit-based division property $\mathcal{D}^{1^8}_{\mathbb{K},\mathbb{L}}$*. We evaluate the propagation on the equivalent circuit shown in the right figure in Fig. 8.3.*

As an example, let us consider 2^3 *chosen plaintexts that all bits in* L_0 *and the first one bit in* R_0 *is constant and the others are active. Such set has the bit-based division property* $\mathcal{D}^{1^8}_{\phi,\{[0,0,0,0,0,1,1,1]\}}$*. Hereinafter,* $[0,0,0,0,0,1,1,1]$ *is referred to as* $[00000111]$ *for the simplicity.*

Round 1 *Now, the bit-based division property of* (L_0, R_0) *is* $\mathcal{D}^{1^8}_{\phi,\{[00000111]\}}$*.*

Step 1 *Since the 8th bit of* $[00000111]$ *is one, there is no change in* $\mathbb{L}$*. Moreover, there is no vector in* $\mathbb{K}$*.*

Step 2 *Since the 7th bit of* $[00000111]$ *is one, there is no change in* $\mathbb{L}$*. Moreover, there is no vector in* $\mathbb{K}$*.*

Step 3 *Since the 6th bit of* $[00000111]$ *is one, there is no change in* $\mathbb{L}$*. Moreover, there is no vector in* $\mathbb{K}$*.*

Step 4 *Since the 1st, 2nd, and 4th bits of* $[00000111]$ *are zero, there is no change in* $\mathbb{L}$*. Moreover, there is no vector in* $\mathbb{K}$*.*

Finally, since the round key is XORed with the right half, the parity of $\pi_{[00001111]}(\vec{x})$ *for all* $\vec{x} \in \mathbb{X}$ *becomes unknown. After the swapping, the bit-based division property of* (L_1, R_1) *is* $\mathcal{D}^{1^8}_{\{[11110000]\},\{[01110000]\}}$*.*

Round 2 *Now, the bit-based division property of* (L_1, R_1) *is* $\mathcal{D}^{1^8}_{\{[11110000]\},\{[01110000]\}}$.

Step 1 *The propagation from* [11110000] $\in \mathbb{K}$ *generates three vectors as*

$$[11110000] \Rightarrow [11110000], [01100001], [10110001].$$

Moreover, the propagation from [01110000] $\in \mathbb{L}$ *generates three vectors as*

$$[01110000] \Rightarrow [01110000], [00110001], [01110001],$$

where [01110001] *is redundant because* [01110001] $\succeq$ [01100001].

Step 2 *From every vector in* $\mathbb{K}$*, the following eight vectors*

$$\begin{aligned}
&[11110000] \Rightarrow [11110000], [11000010], [01110010],\\
&[01100001] \Rightarrow [01100001], [01000011],\\
&[10110001] \Rightarrow [10110001], [10000011], [00110011],
\end{aligned}$$

are generated. Moreover, from every vector in $\mathbb{L}$*, the following 10 vectors*

$$\begin{aligned}
&[01110000] \Rightarrow [01110000], [01000010], [01010010], [01100010], [01110010],\\
&[00110001] \Rightarrow [00110001], [00000011], [00010011], [00100011], [00110011],
\end{aligned}$$

are generated. Here, [01110010] *and* [00110011] *are redundant because there are the same vector in* $\mathbb{K}'$.

Step 3 *From every vector in* $\mathbb{K}$*, the following 19 vectors*

$$\begin{aligned}
&[11110000] \Rightarrow [11110000], [10010100], [11100100],\\
&[11000010] \Rightarrow [11000010], [10000110],\\
&[01110010] \Rightarrow [01110010], [00010110], [01100110],\\
&[01100001] \Rightarrow [01100001], [00000101],\\
&[01000011] \Rightarrow [01000011], [00000111],\\
&[10110001] \Rightarrow [10110001], [10010101], [10100101],\\
&[10000011] \Rightarrow [10000011],\\
&[00110011] \Rightarrow [00110011], [00010111], [00100111],
\end{aligned}$$

are generated. Here, [00000111]*,* [10010101]*,* [10100101]*,* [00010111]*, and* [00100111] *are redundant. Moreover, from every vector in* $\mathbb{L}$*, the following 22 vectors*

$$\begin{aligned}
&[01110000] \Rightarrow [01110000], [00010100], [01010100], [00110100], [01100100],\\
&[01000010] \Rightarrow [01000010],\\
&[01010010] \Rightarrow [01010010], [01000110], [01010110],\\
&[01100010] \Rightarrow [01100010], [00000110], [01000110], [00100110], [01100110],\\
&[00110001] \Rightarrow [00110001], [00100101], [00110101],\\
&[00000011] \Rightarrow [00000011],\\
&[00010011] \Rightarrow [00010011], [00000111], [00010111],\\
&[00100011] \Rightarrow [00100011],
\end{aligned}$$

are generated. where [01000110] *is cancelled because it is propagated from* [01010010] *and* [01100010] *twice. Moreover,* [01010110]*,* [00100101]*,* [00110101]*,* [01100110]*,* [00000111]*, and* [00010111] *are redundant.*

Step 4 *Repeating the similar procedure, elements in* $\mathbb{K}$ *are represented as*

$$\begin{aligned}
&[11110000], [01100001], [10110001], [01110010], [11000010], [00110011],\\
&[01000011], [10000011], [10010100], [11100100], [00000101], [00010110],\\
&[01100110], [10000110], [00111000], [11011000], [00101001], [01001001],\\
&[10011001], [00001010], [00011100], [00101100], [11001100].
\end{aligned}$$

Moreover, elements in $\mathbb{L}$ are represented as

$$[01110000], [00110001], [01000010], [01010010], [01100010], [00000011],$$
$$[00010011], [00100011], [00010100], [00110100], [01010100], [01100100],$$
$$[00000110], [00100110], [01011000], [00011001], [01001100].$$

Finally, the round key is XORed with the right half. Some new vectors, which are generated from $\vec{\ell} \in \mathbb{L}$, are inserted into $\mathbb{K}$. Moreover, some vectors in $\mathbb{L}$ becomes redundant because of the new vectors of $\mathbb{K}$. As a result, we get 17 vectors in $\mathbb{L}$ and 29 vectors in $\mathbb{K}$.

After the 2nd round, we similarly evaluate the propagation of the bit-based division property. As a result, the bit-based division property of (L_4, R_4) is $\mathcal{D}^{1^8}_{\mathbb{K},\mathbb{L}}$, where

$$\mathbb{K} = \{[00010000], [00100000], [01000000], [10000000], [00000011], [00000101], [00000110], [00001001], [00001010], [00001100]\},$$

and

$$\mathbb{L} = \{[00000010], [00000100]\}.$$

It means that the sum of R_4 is `0x6`*. The conventional bit-based division property proves that the first and last bits of R_4 are balanced in Example 8.1. By using the bit-based division property using three subsets, we additionally know that the second and third bits of R_4 are always one.*

8.4.7 Application to Simon32

We evaluate the propagation characteristic of the bit-based division property using three subsets on Simon32. We prepare chosen plaintexts such that the first bit is constant and the others are active, and the set of chosen plaintexts has $\mathcal{D}^{1^{32}}_{\{[1,1,1,\ldots,1]\},\{[0,1,1,\ldots,1]\}}$.

Table 8.6: Propagation of the bit-based division property using three subsets on Simon32

#rounds	$\lvert\mathbb{K}\rvert$	$\min_w(\mathbb{K})$	$\max_w(\mathbb{K})$	$\lvert\mathbb{L}\rvert$	$\min_w(\mathbb{L})$	$\max_w(\mathbb{L})$
0 (plaintexts)	1	32	32	1	31	31
1	1	32	32	1	31	31
2	1	32	32	5	30	31
3	6	30	31	19	29	31
4	43	28	30	138	27	31
5	722	25	29	2236	24	31
6	23321	20	29	89878	19	31
7	996837	15	28	4485379	14	30
8	9849735	8	26	47149981	8	25
9	2524718	5	19	2453101	5	18
10	130724	3	14	20360	3	13
11	7483	2	7	168	2	6
12	852	2	4	8	2	3
13	181	1	2	0	-	-
14	32	1	2	0	-	-
15	32	1	1	0	-	-

Table 8.6 shows the propagation characteristic, where there are not redundant vectors in $\mathbb{K}$ and $\mathbb{L}$, and $\min_w(\mathbb{K})$, $\max_w(\mathbb{K})$, $\min_w(\mathbb{L})$, and $\max_w(\mathbb{L})$ are defined as

$$\min_w(\mathbb{K}) = \min_{\vec{k}\in\mathbb{K}} \left(\sum_{i=1}^{32} w(k_i) \right), \quad \max_w(\mathbb{K}) = \max_{\vec{k}\in\mathbb{K}} \left(\sum_{i=1}^{32} w(k_i) \right),$$
$$\min_w(\mathbb{L}) = \min_{\vec{\ell}\in\mathbb{L}} \left(\sum_{i=1}^{32} w(\ell_i) \right), \quad \max_w(\mathbb{L}) = \max_{\vec{\ell}\in\mathbb{L}} \left(\sum_{i=1}^{32} w(\ell_i) \right),$$

where we perfectly remove redundant vectors from $\mathbb{K}$ and $\mathbb{L}$. As a result, the output of the 14th round function has $\mathcal{D}^{1^{32}}_{\mathbb{K},\phi}$, where vector in $\mathbb{K}$ are represented by hexadecimal notation as

```
(0001 0000)  (0002 0000)  (0004 0000)  (0008 0000)  (0010 0000)  (0020 0000)
(0040 0000)  (0080 0000)  (0100 0000)  (0200 0000)  (0400 0000)  (0800 0000)
(1000 0000)  (2000 0000)  (4000 0000)  (8000 0000)  (0000 0002)  (0000 0004)
(0000 0008)  (0000 0010)  (0000 0020)  (0000 0040)  (0000 0081)  (0000 0100)
(0000 0200)  (0000 0400)  (0000 0800)  (0000 1000)  (0000 2000)  (0000 4001)
(0000 4080)  (0000 8000),
```

and ϕ denotes the empty set. This division property means that the output of the 14th round function takes the following integral property

$$(\texttt{????,????,????,????, ?b??,????,b???,???b}),$$

where balanced and unknown bits are labeled as `b` and `?`, respectively. In the Simon family, we can easily get a 15-round integral characteristic from the 14-round one, and this proved integral characteristic is completely the same as the experimental one. Therefore, we conclude that the experimental characteristic is not probabilistic characteristic, and it works for all keys.

Experimental Verification. Table 8.6 shows that the output of the 12th round function has the division property $\mathcal{D}^{1^{32}}_{\mathbb{K},\mathbb{L}}$, where $|\mathbb{K}| = 852$ and $|\mathbb{L}| = 8$. Here, eight vectors in $\mathbb{L}$ are represented by hexadecimal notation as

```
(0004 0001)  (0004 0003)  (0004 0401)  (0004 0402)
(0000 0403)  (0000 0440)  (0000 1001)  (0000 2100).
```

We experimentally confirmed whether or not the parity is 1 by randomly choosing the secret key, and we repeat the confirmation 20 times. As a result, we confirmed that the parity is always 1.

8.4.8 Application to Simeck32

Simeck was recently proposed in [YZS+15], and its round function is very similar to that of Simon. Let (L_i, R_i) be the output of the ith round function, and it is calculated as

$$(L_i, R_i) = (L_{i-1} \wedge L_{i-1}^{\lll 5}) \oplus L_{i-1}^{\lll 1} \oplus R_{i-1} \oplus k_i, L_{i-1}).$$

The rotation number is changed from $(1, 8, 2)$ to (0,5,1). Similar to Simon, Simeck has different parameters according to the block length. Let Simeck$2n$ be the Simeck block ciphers with $2n$-bit block length, where n is chosen from 16, 24, and 32.

We also evaluated the propagation of the bit-based division property using three subsets against Simeck32. As a result, the output of the 14th round function has $\mathcal{D}^{1^{32}}_{\mathbb{K},\phi}$, where vectors in $\mathbb{K}$ are represented by hexadecimal notation as

```
(0001 0000)  (0002 0000)  (0004 0000)  (0008 0000)  (0010 0000)  (0020 0000)
(0040 0000)  (0080 0000)  (0100 0000)  (0200 0000)  (0400 0000)  (0800 0000)
(1000 0000)  (2000 0000)  (4000 0000)  (8000 0000)  (0000 0002)  (0000 0004)
(0000 0008)  (0000 0011)  (0000 0021)  (0000 0030)  (0000 0040)  (0000 0080)
(0000 0100)  (0000 0201)  (0000 0210)  (0000 0220)  (0000 0401)  (0000 0410)
(0000 0420)  (0000 0600)  (0000 0800)  (0000 1000)  (0000 2000)  (0000 4001)
(0000 4010)  (0000 4020)  (0000 4200)  (0000 4400)  (0000 8001)  (0000 8010)
(0000 8020)  (0000 8200)  (0000 8400)  (0000 C000).
```

This division property means that the output of the 14th round function takes the following integral property

$$(\texttt{????,????,????,????, bb??,?bb?,??bb,???b}).$$

Since round keys are XORed after the round function in Simeck, we can trivially get the 15-round integral characteristic. Here, 2^{31} plaintexts are chosen as $(L_0, F(L_0) \oplus R_0)$, where the first bit of R_0 is constant and the others are active.

8.5 Provable Security against Integral Cryptanalysis

We introduced the bit-based division property using three subsets in Sect. 8.4, and we proved that this method can find more accurate integral characteristics than those found by the conventional division property. In particular, we showed that the new method can discover the tight characteristic on SIMON32. However, a problem is left about the feasibility, i.e., the propagation of the division property requires much time and memory complexity. For instance, if we want to evaluate the propagation of the division property $\mathcal{D}_{\mathbb{K}}^{n^m}$, the time and memory complexity is upper-bounded by $(n+1)^m$. Therefore, if the upper bound is too large, e.g., $(n+1)^m \gg 2^{32}$, it is difficult to evaluate the propagation[3]. In the bit-based division property, the time and memory complexity is upper-bounded by 2^n, where n denotes the block length. Moreover, the bit-based division property using three subsets requires more complexity than that using two subsets. Therefore, we cannot apply the bit-based division property to the SIMON family except for SIMON32.

8.5.1 Provable Security for Designers

We cannot apply the bit-based division property to the SIMON family except for SIMON32, but we can show the "provable security" alternatively. When we design new symmetric-key primitives, we have to guarantee the security against several cryptanalyses. Provable security has been discussed in detail for differential and linear cryptanalyses [Mat96, NK95], but such tools do not exist for integral cryptanalysis.

Let $\mathcal{D}_{\mathbb{K}_i,\mathbb{L}_i}^{1^m}$ denotes the division property of the output set of the ith round function. We want to find R-round integral characteristics. Then, for any $\vec{u}$ with $w(\vec{u}) = 1$, we have to evaluate that there are not $\vec{k} \in \mathbb{K}_R$ satisfying $W(\vec{u}) \succeq \vec{k}$ and $\vec{\ell} \in \mathbb{L}_R$ satisfying $W(\vec{u}) = \vec{\ell}$. Therefore, we have to get all vectors in $\mathbb{K}_R$ and $\mathbb{L}_R$, and such vectors are searched by an algorithm like breadth-first search. On the other hand, we want to show that an R-round integral characteristic cannot exist. Then, it is enough to show that $\mathbb{K}_R$ has m distinct vectors whose Hamming weight is one, i.e., there is not balanced bits, and such vectors are searched by an algorithm like depth-first search. In our provable security, we aim to get such number of rounds efficiently, and a *lazy propagation* is useful to find such number of rounds.

Definition 8.2 (Lazy propagation)**.** *Let $\mathcal{D}_{\mathbb{K}_{i-1},\mathbb{L}_{i-1}}^{1^m}$ be the bit-based division property of the input set of the ith round function. The ith round function is applied, and let $\mathcal{D}_{\bar{\mathbb{K}}_i,\bar{\mathbb{L}}_i}^{1^m}$ be the bit-based division property from the lazy propagation. Then, $\bar{\mathbb{K}}_i$ is computed from only a part of vectors in $\mathbb{K}_{i-1}$, and $\bar{\mathbb{L}}_i$ always becomes the empty set ϕ.*

The lazy propagation first removes all vectors from $\mathbb{L}_{i-1}$. Moreover, it only evaluates the propagation from vectors with low Hamming weight in $\mathbb{K}_{i-1}$ because such vectors are more close to unknown. Therefore, it is more efficiently evaluated than the accurate propagation.

Let us consider the meaning of the lazy propagation. Assuming the input set of the $(i-1)$th round function has $\mathcal{D}_{\mathbb{K}_{i-1},\mathbb{L}_{i-1}}^{1^m}$, we get $\mathcal{D}_{\mathbb{K}_i,\mathbb{L}_i}^{1^m}$ and $\mathcal{D}_{\bar{\mathbb{K}}_i,\phi}^{1^m}$ by the accurate propagation and the lazy propagation, respectively. Then, the set of $\vec{u}$ that the parity is unknown is represented as

$$\mathbb{S}_{\mathbb{K}} := \left\{ \vec{u} \in (\mathbb{F}_2)^m \mid \text{there are } \vec{k} \in \mathbb{K}_i \text{ satisfying } W(\vec{u}) \succeq \vec{k} \right\}.$$

On the other hand, $\mathbb{S}_{\bar{\mathbb{K}}_i}$ cannot completely represent the set of $\vec{u}$ that the parity is unknown. However, $\mathbb{S}_{\bar{\mathbb{K}}_i} \subseteq \mathbb{S}_{\mathbb{K}_i}$ always holds.

Next, we repeat the lazy propagation, and we assume that $\mathcal{D}_{\bar{\mathbb{K}}_{i+1},\phi}^{1^m}$ is propagated from $\mathcal{D}_{\bar{\mathbb{K}}_i,\phi}^{1^m}$ by the lazy propagation. Similarly, assuming that $\mathcal{D}_{\mathbb{K}_{i+1},\mathbb{L}_{i+1}}^{1^m}$ is the division property from $\mathcal{D}_{\mathbb{K}_i,\mathbb{L}_i}^{1^m}$ by the accurate propagation, $\mathbb{S}_{\bar{\mathbb{K}}_{i+1}} \subseteq \mathbb{S}_{\mathbb{K}_{i+1}}$ always holds because $\mathbb{S}_{\bar{\mathbb{K}}_i} \subseteq \mathbb{S}_{\mathbb{K}_i}$. Therefore, if the lazy propagation creates $\mathcal{D}_{\bar{\mathbb{K}}_R,\phi}^{1^m}$, where $\bar{\mathbb{K}}_R$ has m distinct vectors whose Hamming weight is one, the accurate propagation also creates the same m distinct vectors in at least the same round.

[3] In [Tod15a], the propagation for MISTY1 was evaluated, and the division property $\mathcal{D}_{\mathbb{K}}^{7,2,7,7,2,7,7,2,7,7,2,7}$ was used. Then, $|\mathbb{K}|$ is upper bounded by $8^8 \times 3^4 = 1358954496 \approx 2^{30.3}$, and it is feasible.

Table 8.7: Accurate propagations up to six rounds

#rounds	SIMON48		SIMON64		SIMON96		SIMON128	
	$\min_w(\mathbb{L})$	$\min_w(\mathbb{K})$	$\min_w(\mathbb{L})$	$\min_w(\mathbb{K})$	$\min_w(\mathbb{L})$	$\min_w(\mathbb{K})$	$\min_w(\mathbb{L})$	$\min_w(\mathbb{K})$
0	47	48	63	64	95	96	127	128
1	47	48	63	64	95	96	127	128
2	46	47	62	63	94	96	126	128
3	45	46	61	62	93	94	125	126
4	43	44	59	60	91	92	123	124
5	40	41	56	57	88	89	120	121
6	35	36	51	52	83	84	115	116

Table 8.8: Lazy propagation of the bit-based division property for the SIMON family

#rounds	SIMON48		SIMON64		SIMON96		SIMON128	
	$\min_w(\mathbb{K})$	Limit	$\min_w(\mathbb{K})$	Limit	$\min_w(\mathbb{K})$	Limit	$\min_w(\mathbb{K})$	Limit
7	30	33	46	61	78	81	110	113
8	20	23	35	38	68	71	100	103
9	11	14	23	26	55	57	87	88
10	7	10	13	15	40	41	71	71
11	5	8	9	10	27	28	59	59
12	3	8	6	8	17	17	42	42
13	2	5	4	7	11	11	32	32
14	2	3	3	7	8	9	21	21
15	1	2	2	7	5	6	15	15
16	1(u)	1	2	4	4	6	10	10
17			1	3	3	6	8	8
18			1	1	2	6	5	6
19			1(u)	1	2	6	4	6
20					1	6	3	6
21					1	6	2	6
22					1	6	2	6
23					1	1	2	6
24					1(u)	1	1	6
25							1	6
26							1	6
27							1	6
28							1(u)	1

8.5.2 Application to SIMON Family

We evaluate the lazy propagation of the bit-based division property on SIMON48, SIMON64, SIMON96, and SIMON128. Here, we only evaluate integral characteristics when they use chosen plaintexts that only one bit of the left half is constant and the other bits are active. We calculate the accurate propagation up to 6 rounds[4]. Table 8.7 shows $\min_w(\mathbb{L})$ and $\min_w(\mathbb{K})$ in the accurate propagation of $\mathcal{D}^{1^{2n}}_{\mathbb{K},\mathbb{L}}$ up to 6 rounds, where $\min_w(\mathbb{L})$ and $\min_w(\mathbb{K})$ are calculated as

$$\min_w(\mathbb{K}) = \min_{\vec{k}\in\mathbb{K}}\left(\sum_{i=1}^{2n} w(k_i)\right), \quad \min_w(\mathbb{L}) = \max_{\vec{\ell}\in\mathbb{L}}\left(\sum_{i=1}^{2n} w(\ell_i)\right).$$

From the 7th round function, we repeat the lazy propagation. We first remove all vectors from $\mathbb{L}$, and then the bit-based division property is represented as $\mathcal{D}^{1^{2n}}_{\mathbb{K},\phi}$, where ϕ denotes the empty set. Moreover, we remove vectors with high Hamming weight from $\mathbb{K}$. Table 8.8 shows the lazy

[4] In our implementation, we could not calculate the accurate propagation up to 7 rounds because of the limitation of the memory size.

propagation of the bit-based division property $\mathcal{D}^{1^{2n}}_{\mathbb{K},\phi}$, where we only store vectors $\vec{k} \in \mathbb{K}$ satisfying

$$\min_w(\mathbb{K}) \leq \sum_{i=1}^{2n} w(k_i) \leq \text{Limit}.$$

Here, $\mathtt{u}$ means that the $\mathbb{K}$ has $2n$ distinct vectors whose Hamming weight is one, and then, we simply say that the propagation reaches the unknown.

Even if there is a vector $\vec{k} \in \mathbb{K}$ satisfying Limit $< \sum_{i=1}^{2n} w(k_i)$, we do not evaluate the propagation from such $\vec{k}$. Therefore, if the propagation from the removed vector $\vec{k}$ immediately reaches the unknown, there is a gap between the accurate propagation and lazy propagation. However, if the lazy propagation reaches the unknown in a specific number of rounds, the accurate propagation at least reaches the unknown in the same number of rounds. Therefore, the lazy propagation is not useful for attackers, but it guarantees the number of rounds that the bit-based division property cannot find integral characteristics.

As a result, the lazy propagation shows that 16-, 19-, 24-, and 28-round Simon48, 64, 96, and 128 probably do not have integral characteristics, respectively. However, we can get $(r+1)$-round integral characteristics from r-round integral characteristics because round keys are XORed after the round function. Therefore, we expect that 17-, 20-, 25-, and 29-round Simon48, 64, 96, and 128 probably do not have integral characteristics, respectively.

8.5.3 Characteristics that Bit-Based Division Property cannot Find

We consider characteristics that the bit-based division property cannot find. Our provable security supposes that all round keys are randomly and secretly chosen. However, practical ciphers generate round keys from the secret key using the key schedule. Therefore, our provable security does not suppose integral characteristics that exploit the key scheduling algorithm.

The bit-based division property using three subsets focuses on the parity $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}}(\vec{x})$ and divides the set of $\vec{u}$ into three subsets. Then, the propagation simply regards $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}_1}(\vec{x}) \oplus \pi_{\vec{u}_2}(\vec{x})$ as unknown if either $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}_1}(\vec{x})$ or $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}_2}(\vec{x})$ is unknown. For instance, if $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}_1}(\vec{x}) \oplus \pi_{\vec{u}_2}(\vec{x})$ is always 0 or 1 although $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}_1}(\vec{x})$ and $\bigoplus_{\vec{x}\in\mathbb{X}} \pi_{\vec{u}_2}(\vec{x})$ are unknown, the bit-based division property cannot exploit such property.

8.6 Conclusion

The division property is a useful technique to find integral characteristics, but it has not been applied to non-S-box-based ciphers effectively. This chapter focused on the bit-based division property. More precisely, this chapter proposed a new variant using three subsets. The conventional bit-based division property divides the set of $\vec{u}$ into two subsets, but the new variant divides the set of $\vec{u}$ into three subsets. The bit-based division property using three subsets can prove that the experimental integral characteristic for Simon32 shown in [WLV+14] works for all keys. Moreover, we focused on the propagation of the division property and showed that the lazy propagation is useful to guarantee the security against integral cryptanalyses using the division property. As a result, we showed that 17-, 20-, 25-, and 29-round Simon48, 64, 96, and 128 probably do not have integral characteristics, respectively.

Chapter 9

Compact Representation for Bit-Based Division Property and Application to PRESENT

***Abstract*–** After the proposal of the division property, some applications and improvements have been proposed. The bit-based division property is also one of such improvements, and the accurate integral characteristic of SIMON32 is theoretically proved. In this chapter, we propose the compact representation for the bit-based division property. The drawback of the bit-based division property is that it cannot be applied to block ciphers whose block length is over 32 because of high time and memory complexity. The compact representation partially solves this problem, and we apply this technique to 64-bit block cipher PRESENT to illustrate our method. We can accurately evaluate the propagation of the bit-based division property thanks to the compact representation. As a result, we find 9-round integral characteristics, and it is improved by two rounds than previous best characteristic. Moreover, we attack 12-round PRESENT-80 and 13-round PRESENT-128 by using this new characteristic.
***Keywords*–** Integral cryptanalysis, Division property, Compact representation, PRESENT

9.1 Introduction

The concept of an integral cryptanalysis was first introduced as the dedicated attack against block cipher SQUARE [DKR97], and Knudsen and Wagner then formalized the dedicated attack as the integral attack [KW02]. The integral cryptanalysis is applied to many ciphers, and this is nowadays one of the most powerful cryptanalyses [KW02, LWZ11, YPK02, ZRHD08]. The integral cryptanalysis mainly consists of two parts: a search for integral characteristics and key recovery. The propagation of the integral property [KW02] and the degree estimation[1] [Knu94, Lai94] have been used as well-known methods to find integral characteristics.

The division property, which is a novel technique to find integral characteristics, was proposed [Tod15b]. This technique is the generalization of the integral property that can also exploit the algebraic degree at the same time. After the proposal, the new understanding of the division property and new applications have been proposed [BC16, SHZ+15, Tod15a, TM16a, ZW15].

The bit-based division property, which is a new variant of the division property, was proposed [TM16a], and please refer to Chap. 8 in detail[2]. To analyze n-bit block ciphers with m ℓ-bit S-boxes, the conventional division property decomposes n-bit value into m ℓ-bit values, and the division property $\mathcal{D}_{\mathbb{K}}^{\ell^m}$ is used. For convenience, we call this-type division property an integer-based division property. On the other hand, the bit-based division property decomposes n-bit value into n 1-bit values, i.e., $\mathcal{D}_{\mathbb{K}}^{1^n}$ is used. The bit-based division property can find more accurate integral characteristics than the integer-based division property. Actually, the bit-based division property proves the 15-round integral characteristic of SIMON32, and it is tight [TM16a].

[1] This method is often called the higher-order differential attack [Knu94, Lai94].

[2] In Chap. 8, we proposed two variants of the bit-based division property: the conventional bit-based division property and the bit-based division property using three subsets. However, we focus on the conventional bit-based division property in this chapter.

Our Contribution. In this chapter, we propose a *compact representation* for the bit-based division property against S-box-based ciphers. A drawback of the bit-based division property is that it requires about 2^n time and memory complexity to evaluate n-bit block ciphers. Therefore, the application is limited to block ciphers with small block length like SIMON32 in [TM16a]. Moreover, at CRYPTO 2016, Boura and Canteaut introduced the parity set, which is the so-called bit-based division property for an S-box [BC16], but the application is also limited to the low-data distinguisher for a few rounds of PRESENT [BKL⁺07]. The compact representation partially solves this problem, and we can get high-data distinguishers by reducing time and memory complexity. To demonstrate the advantage of the compact representation, we apply our new technique to PRESENT. As a result, we find new 9-round integral characteristics. Since the previous best characteristic is 7-round one [WW13], our new characteristic is improved by two rounds. Moreover, we attack 12-round PRESENT-80 and 13-round PRESENT-128 by using the new integral characteristic. Zhang et al. discussed the security of PRESENT against the integral attack in [ZWW15] and attacked 10-round PRESENT-80 and 11-round PRESENT-128 by using the match-through-the-S-box (MTTS) technique. Therefore, our new attack is also improved by two rounds.

9.2 Preliminaries

9.2.1 Notations

We make the distinction between the addition of $\mathbb{F}_2^n$ and addition of $\mathbb{Z}$, and we use $\oplus$ and $+$ as the addition of $\mathbb{F}_2^n$ and addition of $\mathbb{Z}$, respectively. For any $a \in \mathbb{F}_2^n$, the ith element is expressed in $a[i]$, and the Hamming weight $w(a)$ is calculated as $w(a) = \sum_{i=1}^{n} a[i]$. For any $\vec{a} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$, the vectorial Hamming weight of $\vec{a}$ is defined as $W(\vec{a}) = (w(a_1), w(a_2), \ldots, w(a_m)) \in \mathbb{Z}^m$. Moreover, for any $\vec{k} \in \mathbb{Z}^m$ and $\vec{k}' \in \mathbb{Z}^m$, we define $\vec{k} \succeq \vec{k}'$ if $k_i \geq k_i'$ for all i $(1 \leq i \leq m)$. Otherwise, $\vec{k} \not\succeq \vec{k}'$. Let $\mathbb{K}$ be the set of $\vec{k}$, and $|\mathbb{K}|$ denotes the number of elements in $\mathbb{K}$.

9.2.2 Integral Attack

The integral attack was first introduced by Daemen et al. to evaluate the security of SQUARE [DKR97], and then it was formalized by Knudsen and Wagner [KW02]. Attackers first prepare N chosen plaintexts and encrypt them R rounds. If the XOR of all encrypted texts becomes 0, we say that the cipher has an R-round integral characteristic with N chosen plaintexts. Finally, we analyze the entire cipher by using the integral characteristic. There are two classical approaches to find integral characteristics. The first one is the propagation of the integral property [KW02] and another is based on the degree estimation [Knu94, Lai94].

9.2.3 Division Property

The division property proposed in [Tod15b] is a new method to find integral characteristics. This section briefly shows the definition and propagation rules. Please refer to Chap. 5 in detail.

Bit Product Function. The division property of a multiset is evaluated by using the bit product function defined as follows. Let $\pi_u : \mathbb{F}_2^n \to \mathbb{F}_2$ be a bit product function for any $u \in \mathbb{F}_2^n$. Let $x \in \mathbb{F}_2^n$ be the input and $\pi_u(x)$ be the AND of $x[i]$ satisfying $u[i] = 1$, i.e., it is defined as

$$\pi_u(x) := \prod_{i=1}^{n} x[i]^{u[i]}.$$

Notice that $x[i]^1 = x[i]$ and $x[i]^0 = 1$. Let $\pi_{\vec{u}} : (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m}) \to \mathbb{F}_2$ be a bit product function for any $\vec{u} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. Let $\vec{x} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$ be the input and $\pi_{\vec{u}}(\vec{x})$ be defined as

$$\pi_{\vec{u}}(\vec{x}) := \prod_{i=1}^{m} \pi_{u_i}(x_i).$$

The bit product function also appears in the Algebraic Normal Form (ANF) of a Boolean function. The ANF of a Boolean function f is represented as

$$f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^f \left(\prod_{i=1}^{n} x[i]^{u[i]} \right) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^f \pi_u(x),$$

where $a_u^f \in \mathbb{F}_2$ is a constant value depending on f and u.

Definition of Division Property.

Definition 5.3 (Division Property)**.** *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_1,n_2,\ldots,n_m}$, where $\mathbb{K}$ denotes a set of m-dimensional vectors whose elements take a value between 0 and n_i, it fulfills the following conditions:*

$$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} \textit{unknown} & \textit{if there are } \vec{k} \in \mathbb{K} \textit{ s.t. } W(\vec{u}) \succeq \vec{k}, \\ 0 & \textit{otherwise.} \end{cases}$$

See Chap. 5 to better understand the concept in detail. In this chapter, the division property for $(\mathbb{F}_2^n)^m$ is referred to as $\mathcal{D}_{\mathbb{K}}^{n^m}$ for the simplicity. If there are $\vec{k} \in \mathbb{K}$ and $\vec{k}' \in \mathbb{K}$ satisfying $\vec{k} \succeq \vec{k}'$ in the division property $\mathcal{D}_{\mathbb{K}}^{n_1,n_2,\ldots,n_m}$, $\vec{k}$ can be removed from $\mathbb{K}$ because the vector $\vec{k}$ is redundant.

Some propagation rules for the division property are proven and summarized in Chap. 5. We omit the description of the propagation rules in this chapter because it is not always necessary to understand this chapter.

9.2.4 Bit-Based Division Property

The bit-based division property introduced in [TM16a] is variant of the division property. There are two bit-based division properties: the conventional bit-based division property and the bit-based division property using three subsets. Please refer to Chap 8 in detail.

In this chapter, we only focus on the conventional bit-based division property. To analyze n-bit block ciphers, the conventional division property uses $\mathcal{D}_{\mathbb{K}}^{\ell_1,\ell_2,\ldots,\ell_m}$, where ℓ_i and m are chosen by attackers in the range of $n = \sum_{i=1}^{m} \ell_i$. The conventional bit-based division property uses $\mathcal{D}_{\mathbb{K}}^{1^n}$, and it is not against the definition of the conventional division property.

Propagation Characteristic for S-box. Let us consider the propagation characteristic of the bit-based division property for an S-box. Similar observation was shown by Boura and Canteaut in [BC16], and they introduced a new concept called the *parity set* as follows.

Definition 9.1 (Parity Set)**.** *Let $\mathbb{X}$ be a set whose elements take a value of $\mathbb{F}_2^n$. Its parity set is defined as*

$$\mathcal{U}(\mathbb{X}) = \left\{ u \in \mathbb{F}_2^n \mid \bigoplus_{x \in \mathbb{X}} \pi_u(x) = 1 \right\}.$$

Assuming $\mathbb{X}$ has the division property $\mathcal{D}_k^n$,

$$\mathcal{U}(\mathbb{X}) \subseteq \{u \in \mathbb{F}_2^n : w(u) \geq k\}.$$

Let $\mathbb{X}$ and $S(\mathbb{X})$ denote the input set and output set of the S-box, respectively. Then, the parity set of $S(\mathbb{X})$ fulfills

$$\mathcal{U}(S(\mathbb{X})) \subseteq \cup_{u \in \mathcal{U}(\mathbb{X})} V_s(u),$$

where

$$V_s(u) = \{v \in \mathbb{F}_2^n \mid \text{ ANF of } (\pi_v \circ S) \text{ contains } \pi_u(x)\}.$$

The definition of the parity set trivially derives the following proposition.

Table 9.1: Sets $V_S(u)$ for all $u \in \mathbb{F}_2^4$ for the PRESENT S-box. All four-bit values are represented in hexadecimal notation. The rightmost bit of the word corresponds to the least significant bit.

	$V_S(u)$															
	0x0	0x1	0x2	0x4	0x8	0x3	0x5	0x9	0x6	0xA	0xC	0x7	0xB	0xD	0xE	0xF
$u =$ 0x0	x			x	x						x					
$u =$ 0x1		x			x		x				x					
$u =$ 0x2			x		x				x		x					
$u =$ 0x4		x		x				x			x					
$u =$ 0x8		x	x	x	x	x					x					
$u =$ 0x3				x		x	x	x	x	x	x		x			
$u =$ 0x5							x	x			x					
$u =$ 0x9				x		x	x		x	x					x	
$u =$ 0x6		x			x			x	x	x	x					
$u =$ 0xA			x	x			x	x		x		x	x	x	x	x
$u =$ 0xC			x			x		x			x					
$u =$ 0x7			x		x	x		x	x				x	x		
$u =$ 0xB			x	x	x	x			x	x	x	x		x		x
$u =$ 0xD			x	x	x			x		x		x			x	
$u =$ 0xE							x					x	x	x	x	x
$u =$ 0xF																x

Proposition 9.1. *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. When the multiset $\mathbb{X}$ has the bit-based division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, the parity set of $\mathbb{X}$ fulfills*

$$\mathcal{U}(\mathbb{X}) \subseteq \{u \in \mathbb{F}_2^n : \text{ there are } k \in \mathbb{K} \text{ satisfying } u \succeq k\}.$$

Moreover, assuming $\mathcal{U}(\mathbb{X}) \subseteq \mathbb{K}'$, the set $\mathbb{X}$ has the bit-based division property $\mathcal{D}_{\mathbb{K}'}^{1^n}$.

Proposition 9.1 shows that the bit-based division property of $S(\mathbb{X})$ can be evaluated from that of $\mathbb{X}$ via the parity set.

Case of PRESENT S-box. As an example, let us consider the case of the PRESENT S-box. Let (x_4, x_3, x_2, x_1) and (y_4, y_3, y_2, y_1) be the input and output of the S-box, respectively, and the algebraic normal form of the PRESENT S-box is described as

$$\begin{aligned}
y_4 &= x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3 + x_1 + x_2 + x_4 + 1, \\
y_3 &= x_1x_2x_4 + x_1x_3x_4 + x_1x_2 + x_1x_4 + x_2x_4 + x_3 + x_4 + 1, \\
y_2 &= x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_4 + x_3x_4 + x_2 + x_4, \\
y_1 &= x_2x_3 + x_1 + x_3 + x_4.
\end{aligned}$$

Table 9.1 shows sets of $V_S(u)$ for all $u \in \mathbb{F}_2^4$ for the PRESENT S-box. Assuming that $\mathbb{X}$ fulfills $\mathcal{D}_k^{1^4}$, let $\mathcal{D}_{\mathbb{K}'}^{1^4}$ be the bit-based division property of $S(\mathbb{X})$ and $\mathbb{K}'$ is

$$\mathbb{K}' = \cup_{u \in \mathcal{U}(\mathbb{X})} V_s(u), \quad \mathcal{U}(\mathbb{X}) \subseteq \{u \in \mathbb{F}_2^n : u \succeq k\}$$

from Proposition 9.1. We compute $\mathbb{K}'$ for any $k \in \mathbb{F}_2^4$ and then remove redundant vectors. Table 9.2 shows the propagation characteristic of the bit-based division property for the PRESENT S-box.

9.3 Compact Representation for Division Property

9.3.1 Motivation

We can find more accurate integral characteristics by using the bit-based division property than the integer-based division property. However, this evaluation requires about 2^n time and memory complexity for n-bit block ciphers. Therefore, the bit-based division property is applied

Table 9.2: Propagation of the bit-based division property for PRESENT S-box. Vectors on $\mathbb{F}_2^4$ are represented an hexadecimal notation.

$\vec{k}$ of input $\mathcal{D}^{1^4}_{\vec{k}}$	$\mathbb{K}$ of output $\mathcal{D}^{1^4}_{\mathbb{K}}$	$\vec{k}$ of input $\mathcal{D}^{1^4}_{\vec{k}}$	$\mathbb{K}$ of output $\mathcal{D}^{1^4}_{\mathbb{K}}$
0x0	{0x0}	0x8	{0x1, 0x2, 0x4, 0x8}
0x1	{0x1, 0x2, 0x4, 0x8}	0x9	{0x2, 0x4, 0x8}
0x2	{0x1, 0x2, 0x4, 0x8}	0xA	{0x2, 0x4, 0x8}
0x3	{0x2, 0x4, 0x8}	0xB	{0x2, 0x4, 0x8}
0x4	{0x1, 0x2, 0x4, 0x8}	0xC	{0x2, 0x4, 0x8}
0x5	{0x2, 0x4, 0x8}	0xD	{0x2, 0x4, 0x8}
0x6	{0x1, 0x2, 0x8}	0xE	{0x5, 0xB, 0xE}
0x7	{0x2, 0x8}	0xF	{0xF}

Table 9.3: Compact representation for PRESENT S-box

Compact	Real property	Output property	Redundant
$\bar{0}$	{0x0}	{0x0}	$\bar{0}, \bar{1}, \bar{3}, \bar{6}, \bar{7}, \bar{E}, \bar{F}$
$\bar{1}$	{0x1, 0x2, 0x4, 0x8}	{0x1, 0x2, 0x4, 0x8}	$\bar{1}, \bar{3}, \bar{6}, \bar{7}, \bar{E}, \bar{F}$
$\bar{3}$	{0x3, 0x5, 0x9, 0xA, 0xB, 0xC, 0xD}	{0x2, 0x4, 0x8}	$\bar{3}, \bar{7}, \bar{E}, \bar{F}$
$\bar{6}$	{0x6}	{0x1, 0x2, 0x8}	$\bar{6}, \bar{7}, \bar{E}, \bar{F}$
$\bar{7}$	{0x7}	{0x2, 0x8}	$\bar{7}, \bar{F}$
$\bar{E}$	{0xE}	{0x5, 0xB, 0xE}	$\bar{E}, \bar{F}$
$\bar{F}$	{0xF}	{0xF}	$\bar{F}$

to small block-length ciphers like Simon32 in [TM16a]. Moreover, the application of the parity set is limited to the low-data distinguisher for a few rounds of PRESENT [BC16]. It is an open problem to apply the bit-based division property to high-data distinguishers for non small block-length ciphers.

9.3.2 General Idea

The compact representation for the bit-based division property partially solves this problem. We focus on the fact that different division properties cause the same division property through an S-box. Then, we regard the different properties as the same property, and it helps us to evaluate the propagation characteristic efficiently.

Compact Representation for PRESENT S-box. The focus is that there are some input division properties whose output division property is the same. For example, the output division property from $\mathcal{D}^{1^4}_{\{\texttt{0x1}\}}$ is $\mathcal{D}^{1^4}_{\{\texttt{0x1},\texttt{0x2},\texttt{0x4},\texttt{0x8}\}}$, which is the same as that from $\mathcal{D}^{1^4}_{\{\texttt{0x2}\}}$. In the compact representation, we regard their input properties as the same input property. Table 9.3 shows the compact representation for PRESENT S-box. While sixteen values are used to represent the bit-based division property, only seven values $\{\bar{0}, \bar{1}, \bar{3}, \bar{6}, \bar{7}, \bar{E}, \bar{F}\}$ are used in the compact representation. For simplicity, let $\mathbb{S}_c$ be

$$\mathbb{S}_c = \{\bar{0}, \bar{1}, \bar{3}, \bar{6}, \bar{7}, \bar{E}, \bar{F}\}.$$

Note that we have to check the original vectors when we remove redundant vectors. Assuming that the division property is $\mathcal{D}_{\{\bar{3},\bar{6},\bar{E}\}}$, each original vectors are represented as

$$\bar{3} \to \{\texttt{0x3}, \texttt{0x5}, \texttt{0x9}, \texttt{0xA}, \texttt{0xB}, \texttt{0xC}, \texttt{0xD}\}, \quad \bar{6} \to \{\texttt{0x6}\}, \quad \bar{E} \to \{\texttt{0xE}\}.$$

Therefore, $\bar{E}$ is redundant because $\texttt{0xE} \succeq \texttt{0xA}$. On the other hand, there is not a vector $\vec{k}$ satisfying $\texttt{0x6} \succeq \bar{k}$ in $\vec{k} \in \{\texttt{0x3}, \texttt{0x5}, \texttt{0x9}, \texttt{0xA}, \texttt{0xB}, \texttt{0xC}, \texttt{0xD}\}$. As a result, after remove redundant vectors, the division property becomes $\mathcal{D}_{\{\bar{3},\bar{6}\}}$. The right-end column in Table 9.3 shows redundant vectors by the compact representation.

9.3.3 Toy Cipher using PRESENT S-box

Even if we apply the compact representation to the input division property of S-boxes, the propagated output division property is not represented by the compact representation. We need

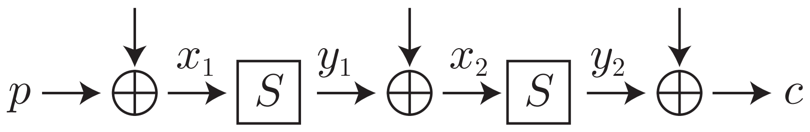


Figure 9.1: Key-alternating cipher underlying PRESENT S-box

to carefully apply the compact representation to the output division property, which depends on the structure of a target cipher. For simplicity, let us consider a key-alternating cipher underlying PRESENT S-box, where the block length is 4 bits, and Figure 9.1 shows the 2-round cipher. Let p and c be the plaintext and ciphertext, and x_i and y_i denote the input and output of the ith S-box, respectively. Note that the division property does not change for constant addition. Then, our aim is to evaluate the division property of c, and it is enough to manage only the compact representation of the division property in x_2. Our next aim is to evaluate the compact representation in x_2. Then, it is enough to manage only the compact representation in x_1, and the following propagation characteristic is applied.

$$\begin{aligned}
&\{\bar{\texttt{0}}\} \to \{\texttt{0x0}\} \to \{\bar{\texttt{0}}\},\\
&\{\bar{\texttt{1}}\} \to \{\texttt{0x1}, \texttt{0x2}, \texttt{0x4}, \texttt{0x8}\} \to \{\bar{\texttt{1}}\},\\
&\{\bar{\texttt{3}}\} \to \{\texttt{0x2}, \texttt{0x4}, \texttt{0x8}\} \to \{\bar{\texttt{1}}\},\\
&\{\bar{\texttt{6}}\} \to \{\texttt{0x1}, \texttt{0x2}, \texttt{0x8}\} \to \{\bar{\texttt{1}}\},\\
&\{\bar{\texttt{7}}\} \to \{\texttt{0x2}, \texttt{0x8}\} \to \{\bar{\texttt{1}}\},\\
&\{\bar{\texttt{E}}\} \to \{\texttt{0x5}, \texttt{0xB}, \texttt{0xE}\} \to \{\bar{\texttt{3}}, (\bar{\texttt{E}})\},\\
&\{\bar{\texttt{F}}\} \to \{\texttt{0xF}\} \to \{\bar{\texttt{F}}\}.
\end{aligned}$$

Note that the property $\bar{\texttt{E}}$ derives $\bar{\texttt{3}}$ and $\bar{\texttt{E}}$, but $\bar{\texttt{E}}$ is redundant.

Example 9.1. *Assuming the division property of p is $\{\bar{\texttt{E}}\}$, the division property of x_1 is also $\{\bar{\texttt{E}}\}$ because the division property is independent of the constant XORing. Applying the first S-box, the division property of y_1 is $\{\bar{\texttt{3}}, \bar{\texttt{E}}\}$, and $\bar{\texttt{E}}$ is redundant. Since the division property is independent of the constant XORing, the division property of x_2 is $\{\bar{\texttt{3}}\}$. Applying the second S-box, the bit-based division property of y_2 is $\mathcal{D}^{1^4}_{\{\texttt{0x2},\texttt{0x4},\texttt{0x8}\}}$, and the bit-based division property of c is also $\mathcal{D}^{1^4}_{\{\texttt{0x2},\texttt{0x4},\texttt{0x8}\}}$. Therefore, the least significant bit of c is balanced.*

9.3.4 Core Function of PRESENT

PRESENT does not have simple key-alternating structure like Fig. 9.1. There is a bit permutation in the diffusion part of the round function, and we can decompose the round function of PRESENT into four subfunctions. Figure 9.3 shows the equivalent circuit of the round function of PRESENT. The input and output of every sub function are four four-bit values, and the position of each four-bit value then moves. Since this equivalent circuit does not have bit-oriented permutation except the interior of sub functions, we first generate the propagation characteristic table of sub functions under the compact representation. Then, we evaluate the propagation characteristic of round functions from the table under the compact representation.

Propagation Characteristic for Sub Function. Let $\vec{k} = (k_4, k_3, k_2, k_1) \in (\mathbb{S}_c)^4$ be the input division property of the sub function. Then, the output division property $\mathbb{K}$ is the set whose elements are vectors in $(\mathbb{S}_c)^4$. Algorithm 10 generates the propagation characteristic table under the compact representation for the sub function. Here, `compact` is a function that converts from the bit-based division property to the compact representation.

Example 9.2 (Propagation characteristic from $(\bar{\texttt{3}}, \bar{\texttt{6}}, \bar{\texttt{7}}, \bar{\texttt{F}})$)**.** *The output bit-based division property of each S-box is evaluated from the corresponding compact representation as*

$$\bar{\texttt{3}} \to \{\texttt{0x2}, \texttt{0x4}, \texttt{0x8}\}, \quad \bar{\texttt{6}} \to \{\texttt{0x1}, \texttt{0x2}, \texttt{0x8}\}, \quad \bar{\texttt{7}} \to \{\texttt{0x2}, \texttt{0x8}\}, \quad \bar{\texttt{F}} \to \{\texttt{0xF}\}.$$

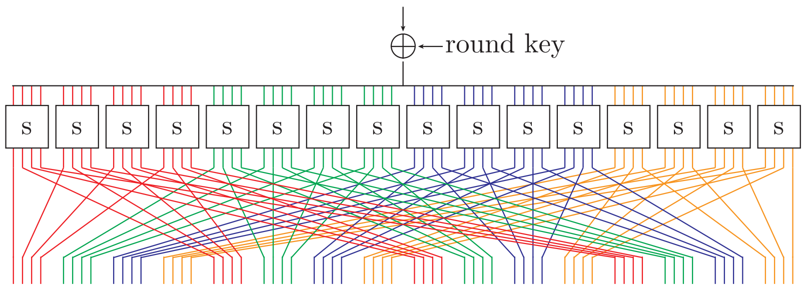


Figure 9.2: Round function of PRESENT

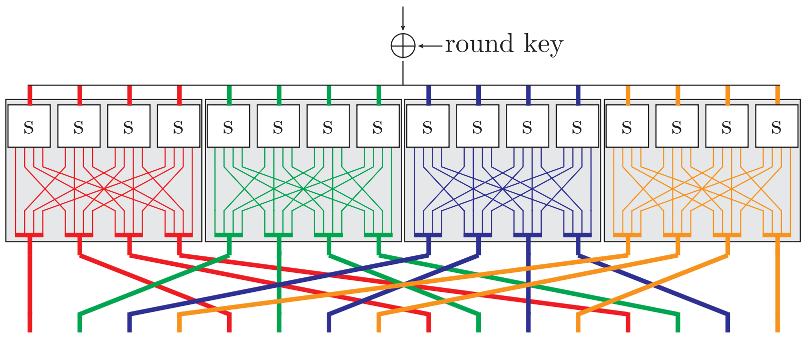


Figure 9.3: Equivalent circuit of round function of PRESENT

Then, let $\mathcal{D}^{1^4}_{\mathbb{K}'}$ *be the output bit-based division property, and* $\mathbb{K}'$ *is represented as* $18(= 3 \times 3 \times 2 \times 1)$ *vectors*

$$(\mathtt{0x11B5}), (\mathtt{0x3195}), (\mathtt{0x11F1}), (\mathtt{0x31D1}), (\mathtt{0x51B1}), (\mathtt{0x7191}),$$
$$(\mathtt{0x1935}), (\mathtt{0x3915}), (\mathtt{0x1971}), (\mathtt{0x3951}), (\mathtt{0x5931}), (\mathtt{0x7911}),$$
$$(\mathtt{0x9135}), (\mathtt{0xB115}), (\mathtt{0x9171}), (\mathtt{0xB151}), (\mathtt{0xD131}), (\mathtt{0xF111}).$$

Then, the compact representation of 18 vectors is

$$(\overline{\mathtt{1}}\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{3}}), (\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{3}}), (\overline{\mathtt{1}}\overline{\mathtt{1}}\overline{\mathtt{F}}\overline{\mathtt{1}}), (\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{1}}), (\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{1}}), (\overline{\mathtt{7}}\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{1}}), (\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{3}}\overline{\mathtt{3}}), (\overline{\mathtt{3}}\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{3}}), (\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{7}}\overline{\mathtt{1}}),$$
$$(\overline{\mathtt{3}}\overline{\mathtt{3}}\overline{\mathtt{3}}\overline{\mathtt{1}}), (\overline{\mathtt{3}}\overline{\mathtt{3}}\overline{\mathtt{3}}\overline{\mathtt{1}}), (\overline{\mathtt{7}}\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{1}}), (\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{3}}), (\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{1}}\overline{\mathtt{3}}), (\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{7}}\overline{\mathtt{1}}), (\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{1}}), (\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{1}}), (\overline{\mathtt{F}}\overline{\mathtt{1}}\overline{\mathtt{1}}\overline{\mathtt{1}}).$$

After remove redundant vectors, the output division property is represented as

$$(\overline{\mathtt{1}}\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{3}}), (\overline{\mathtt{1}}\overline{\mathtt{1}}\overline{\mathtt{F}}\overline{\mathtt{1}}), (\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{1}}), (\overline{\mathtt{1}}\overline{\mathtt{3}}\overline{\mathtt{7}}\overline{\mathtt{1}}), (\overline{\mathtt{7}}\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{1}}), (\overline{\mathtt{3}}\overline{\mathtt{1}}\overline{\mathtt{1}}\overline{\mathtt{3}}), (\overline{\mathtt{F}}\overline{\mathtt{1}}\overline{\mathtt{1}}\overline{\mathtt{1}})$$

by the compact representation.

9.4 Improved Integral Attack on PRESENT

9.4.1 New Algorithm to Find Integral Characteristics

We show a new algorithm to find integral characteristics of PRESENT by using the compact representation of the division property. Note that the given integral characteristic is the same as that given by the accurate propagation characteristic of the bit-based division property.

The input of the algorithm is the bit-based division property of the plaintext set. The algorithm first converts from this bit-based division property to the corresponding compact representation. In every round function, the algorithm evaluates the propagation characteristic for four sub functions independently and the relocation of 16 four-bit values. Algorithm 11 evaluates the propagation characteristic for round functions. This evaluation is repeated until there is no integral characteristic in the output of the round function.

Algorithm 10 Generate propagation characteristic table for the sub function

```
1: procedure evalSubFunction($\vec{k} \in (\mathbb{S}_c)^4$)
2:     $\mathbb{K}_i$ is the set of the propagated division property from $k_i$ through the S-box.
3:     $\mathbb{K}'$ is an empty set.
4:     for all $(x, y, z, w) \in (\mathbb{K}_4 \times \mathbb{K}_3 \times \mathbb{K}_2 \times \mathbb{K}_1)$ do
5:         $k'_4 \Leftarrow \texttt{compact}(x_4 \| y_4 \| z_4 \| w_4)$
6:         $k'_3 \Leftarrow \texttt{compact}(x_3 \| y_3 \| z_3 \| w_3)$
7:         $k'_2 \Leftarrow \texttt{compact}(x_2 \| y_2 \| z_2 \| w_2)$
8:         $k'_1 \Leftarrow \texttt{compact}(x_1 \| y_1 \| z_1 \| w_1)$
9:         $\mathbb{K}' = \mathbb{K}' \cup \{\vec{k}'\}$
10:    end for
11:    remove redundant vectors from $\mathbb{K}'$
12:    return $\mathbb{K}'$
13: end procedure
```

Algorithm 11 Generate propagation characteristic table for the sub function

```
1: procedure evalRoundFunction($\vec{k} \in (\mathbb{S}_c)^{16}$)
2:     $\mathbb{K}_i \Leftarrow \texttt{evalSubFunction}([k_{4*i+4}, k_{4*i+3}, k_{4*i+2}, k_{4*i+1}])$
3:     for all $(\vec{x}, \vec{y}, \vec{z}, \vec{w}) \in (\mathbb{K}_4 \times \mathbb{K}_3 \times \mathbb{K}_2 \times \mathbb{K}_1)$ do
4:         $k'_{16} = x_4, k'_{12} = x_3, k'_8 = x_2, k'_4 = x_1$
5:         $k'_{15} = y_4, k'_{11} = y_3, k'_7 = y_2, k'_3 = y_1$
6:         $k'_{14} = z_4, k'_{10} = z_3, k'_6 = z_2, k'_2 = z_1$
7:         $k'_{13} = w_4, k'_9 = w_3, k'_5 = w_2, k'_1 = w_1$
8:         $\mathbb{K}' = \mathbb{K}' \cup \{\vec{k}'\}$
9:     end for
10:    remove redundant vectors from $\mathbb{K}'$
11:    return $\mathbb{K}'$
12: end procedure
```

7-Round Integral Characteristic Revisited. We first revisit the 16th order integral characteristic [WW13], where the least significant bit (lsb) in the output of the 7-round PRESENT is balanced when the least sixteen bits are active and the others are constant. The bit-based division property of the plaintext set is $\mathcal{D}^{1^{64}}_{\texttt{0x00000000000000FF}}$, and the compact representation is

$$\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{F}}\bar{\texttt{F}}.$$

Ciphertexts encrypted by one round have the following compact representation

$$\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{F}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{F}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{F}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{0}}\bar{\texttt{F}}.$$

Moreover, ciphertexts encrypted by two rounds have the following compact representation

$$\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}\bar{\texttt{1}}.$$

Table 9.4 shows the propagation characteristic, where we perfectly remove redundant vectors. After six rounds, we get 70 elements in the compact representation. We finally apply additional one-round function, and the propagated bit-based division property does not include `0x0000000000000001`. Therefore, the lsb in the output of the 7-round PRESENT is balanced.

New 9-Round Integral Characteristic. We next search for integral characteristics exploiting more number of active bits. Let us recall Table 9.3. Then, the propagated characteristic

Table 9.4: Propagation from $\mathcal{D}^{1^{64}}_{\texttt{0x00000000000000FF}}$

#rounds	0	1	2	3	4	5	6	7 ⋆
$\lvert\mathbb{K}\rvert$	1	1	1	707281	349316	1450	70	63

⋆We do not use the compact representation in the final round.

Table 9.5: Propagation from $\mathcal{D}^{1^{64}}_{\texttt{0xFFFFFFFFFFFFFFF0}}$

#rounds	0	1	2	3	4	5	6	7	8	9 ⋆
$\lvert\mathbb{K}\rvert$	1	1	81	8277	136421	2497368	343121	1393	70	63

⋆We do not use the compact representation in the final round.

Table 9.6: Propagation from $\mathcal{D}^{1^{64}}_{\texttt{0xFFFFFFFFFFFFFFFE}}$

#rounds	0	1	2	3	4	5	6	7	8	9 ⋆
$\lvert\mathbb{K}\rvert$	1	3	15	174	1053	9s6251	444174	19749	188	376

⋆We do not use the compact representation in the final round.

from $\bar{\texttt{E}}$ is $\{\texttt{0x5}, \texttt{0xB}, \texttt{0xE}\}$, and the output bit-based division property is most far from *unknown property* except for $\bar{\texttt{F}}$.

We prepare the plaintext set that the least significant four bits are passive and the others are active, and the compact representation is

$$\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{0}}.$$

Ciphertexts encrypted by one round have the following compact representation

$$\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{E}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{E}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}, \bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{E}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}, \bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{E}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}, \bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{E}}.$$

Moreover, the compact representation of ciphertexts encrypted by two rounds consists of 81 elements, where all representations are represented by only $\bar{\texttt{E}}$ and $\bar{\texttt{F}}$. After eight rounds, we get 70 elements in the compact representation. We finally apply additional one-round function, and the propagated bit-based division property does not include `0x0000000000000001`. Therefore, the lsb in the output of the 9-round PRESENT is balanced. Table 9.5 shows the propagation characteristic, where we perfectly remove redundant vectors.

Integral Characteristics with Maximum Number of Chosen Plaintexts. The number of rounds that integral characteristics cover is clearly maximized when the number of active bits is 63. Therefore, we moreover search for integral characteristics exploiting 2^{63} chosen plaintexts. Then, we prepare the plaintext set that the lsb is passive and the others are active, and the compact representation is

$$\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{E}}.$$

Ciphertexts encrypted by one round have the following compact representation

$$\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{E}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{E}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}, \bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{E}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}, \bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{F}}\bar{\texttt{E}}.$$

Moreover, the compact representation of ciphertexts encrypted by two rounds consists of 15 elements, where all compact representations are represented by only $\bar{\texttt{E}}$ and $\bar{\texttt{F}}$. After eight rounds, we get 188 elements in the compact representation. We finally apply additional one-round function, and the integral property is

$$\texttt{0xEEE0EEE0EEE0EEE0},$$

where `E` means that the 1st bit is balanced, and `0` means that all bits are balanced, i.e., 28 bits are balanced. Table 9.6 shows the propagation characteristic, where we perfectly remove redundant vectors.

9.4.2 Key Recovery with MTTS Technique and FFT Key Recovery

We attack 12-round PRESENT-80 and 13-round PRESENT-128 by using new 9-round integral characteristics. Our attack uses the match-through-the-S-box (MTTS) technique [ZWW15] and FFT key recovery shown in Chap. 4 [TA14, TA15]. We briefly explain their previous techniques.

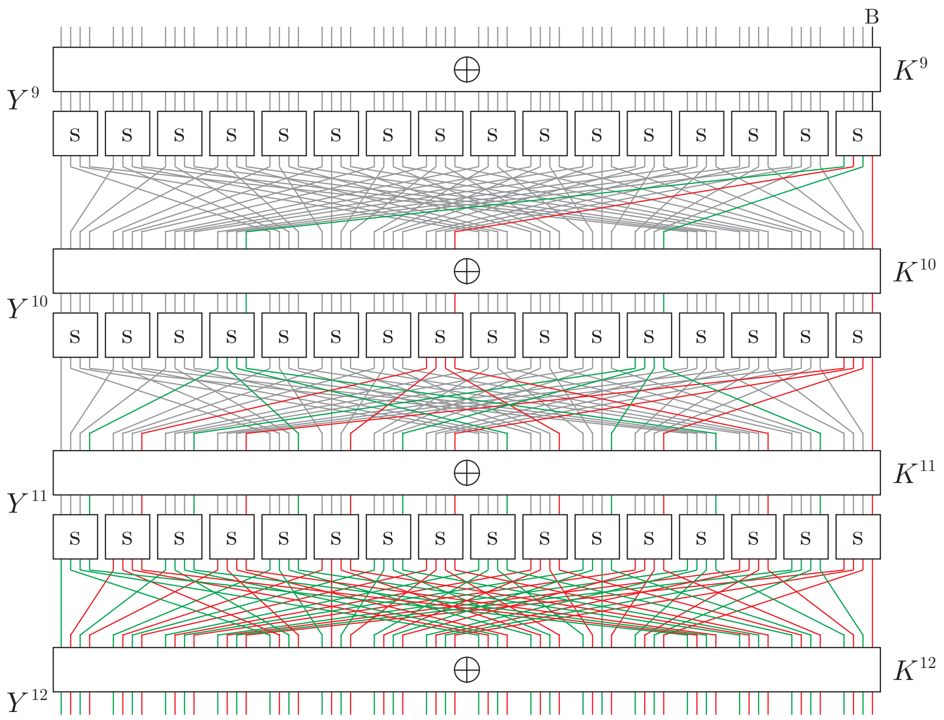


Figure 9.4: 3-Round key recovery for PRESENT

Match-through-the-S-box (MTTS) Technique [ZWW15]**.** The MTTS technique was proposed by Zhang et al., and it is the extension of the meet-in-the-middle technique [SW12b]. Let $x = (x_4, x_3, x_2, x_1)$ and $y = (y_4, y_3, y_2, y_1)$ be the input and output of the PRESENT S-box. Assuming that x_1 is balanced over a chosen plaintext set Λ, the aim is to recover round keys such that $\bigoplus_\Lambda x_1 = 0$. Then, $x_1 = y_4 y_2 \oplus y_3 \oplus y_1 \oplus 1$ from the ANF of S^{-1}, and $\bigoplus_\Lambda y_4 y_2 = \bigoplus_\Lambda y_3 \oplus y_1$ because $\bigoplus_\Lambda x_1 = 0$. Therefore, we independently evaluate the XOR of $y_4 y_2$ and that of $y_3 \oplus y_1$, and we then search for round keys that two XORs take the same value. In [ZWW15], Zhang et al. attacked 10-round PRESENT-80 and 11-round PRESENT-128 by using the MTTS technique.

Fast Fourier Transform (FFT) Key Recovery Technique [TA14, TA15]**.** The FFT key recovery was proposed in Chap. 4, and it was originally used for the linear cryptanalysis in [CSQ07]. We now evaluate the XOR

$$\bigoplus_\Lambda f_{k_1}(c \oplus k_2),$$

where f_{k_1} is a Boolean function depending on a round key k_1. Moreover, κ_1 and κ_2 are bit lengths of k_1 and k_2, respectively. Then, we can evaluate XORs over all (k_1, k_2) with $3\kappa_2 2^{\kappa_1+\kappa_2}$ time complexity. Note that the time complexity does not depend on the number of chosen plaintexts. Therefore, we can easily evaluate the time complexity by only counting the bit length of involved round keys. Please refer to Chap. 4 in detail.

Integral Attack against 12-round PRESENT-80. Let X^i be the input of the $(i+1)$th round function, and Y^i is computed as $Y^i = X^i \oplus K^i$, where K^i denotes the round key. Moreover, $X^i[j]$, $Y^i[j]$, and $K^i[j]$ denote the jth bit of X^i, Y^i, and K^i from the right hand, respectively. Here, X^0 is plaintexts, and Y^i is ciphertexts in i-round PRESENT. Figure 9.4 shows the 3-round key recovery for PRESENT.

In the first step, we choose 2^{60}-plaintext sets (denoted by Λ) and get corresponding ciphertexts after 12-round encryption. We store frequencies of two 32-bit values

$$Y_E = (Y^{12}[0], Y^{12}[2], \ldots, Y^{12}[62]) \quad Y_O = (Y^{12}[1], Y^{12}[3], \ldots, Y^{12}[63])$$

into voting tables.

In the second step, we compute the XOR of $(X^{10}[16] \times X^{10}[48])$ from Y_E by guessing involved round keys. The XOR is computed as

$$\bigoplus_{\Lambda}(X^{10}[16] \times X^{10}[48]) = f_{K^{10}[16,48],K^{11}[0,8,\ldots,56]}(Y_E \oplus K_E^{12}),$$

where $K_E^{12} = (K^{12}[0], K^{12}[2], \ldots, K^{12}[62])$. The FFT key recovery can evaluate the XOR with the time complexity $3 \times 32 \times 2^{2+8+32} = 3 \times 2^{47}$. Note that this time complexity is negligible because we already use 2^{60} time complexity to prepare chosen plaintexts.

In the third step, we compute the XOR of $(X^{10}[0] \oplus X^{10}[32])$ from Y_O by guessing involved round keys. The XOR is computed as

$$\bigoplus_{\Lambda}(X^{10}[0] \oplus X^{10}[32]) = f'_{K^{11}[4,12,\ldots,60]}(Y_O \oplus K_O^{12}).$$

where $K_O^{12} = (K^{12}[1], K^{12}[3], \ldots, K^{12}[63])$. Note that we do not need to guess $K^{10}[0]$ and $K^{10}[32]$ because they are linearly involved to $\bigoplus_{\Lambda}(X^{10}[0] \oplus X^{10}[32])$. Then, the XOR is evaluated with the time complexity $3 \times 32 \times 2^{8+32} = 3 \times 2^{45}$, and it is also negligible.

In the fourth step, we search for round keys satisfying

$$\bigoplus_{\Lambda}(X^{10}[16]X^{10}[48]) = \bigoplus_{\Lambda}(X^{10}[0] \oplus X^{10}[32]).$$

Since involved round keys are 42 bits and 40 bits, the total is over 80 bits. However, the total bit length of involved round keys reduces to 68 bits from the key schedule (see Fig. 9.5). Therefore, by repeating this procedure N times, we can reduce the key space to 2^{68-N}.

Finally, we exhaustively search remaining keys, and the time complexity is 2^{80-N}. Therefore, the data complexity is $N \times 2^{60}$, and the time complexity is $(N \times 2^{60} + 2^{80-N})$ for $N \in \{1, 2, \ldots, 16\}$.

Integral Attack against 13-round PRESENT-128. We attack 13-round PRESENT-128 by using the similar strategy as the 12-round attack.

As a result, the FFT key recovery can evaluate the XOR of $(X^{10}[16] \times X^{10}[48])$ with the time complexity $3 \times 64 \times 2^{2+8+32+64} = 3 \times 2^{112}$. Moreover, the FFT key recovery can evaluate the XOR of $(X^{10}[0] \oplus X^{10}[32])$ with the time complexity $3 \times 64 \times 2^{8+32+64} = 3 \times 2^{110}$. While involved round keys are 112 bits and 110 bits, the total bit length of involved round keys reduces to 126 bits because of the key schedule (see Fig. 9.6). Therefore, by repeating the procedure N times, we can reduce the key space to 2^{126-N}. Finally, we exhaustively search remaining keys. The time complexity is 2^{128-N}, and it is the dominant complexity. Therefore, the data complexity is $N \times 2^{60}$, and the time complexity is 2^{128-N} for $N \in \{1, 2, \ldots, 16\}$.

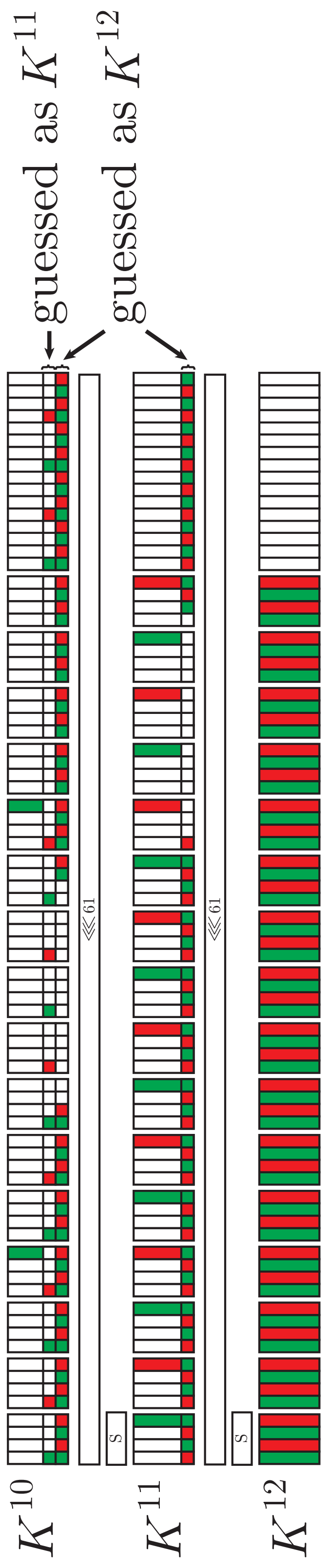


Figure 9.5: Involved round keys of PRESENT-80

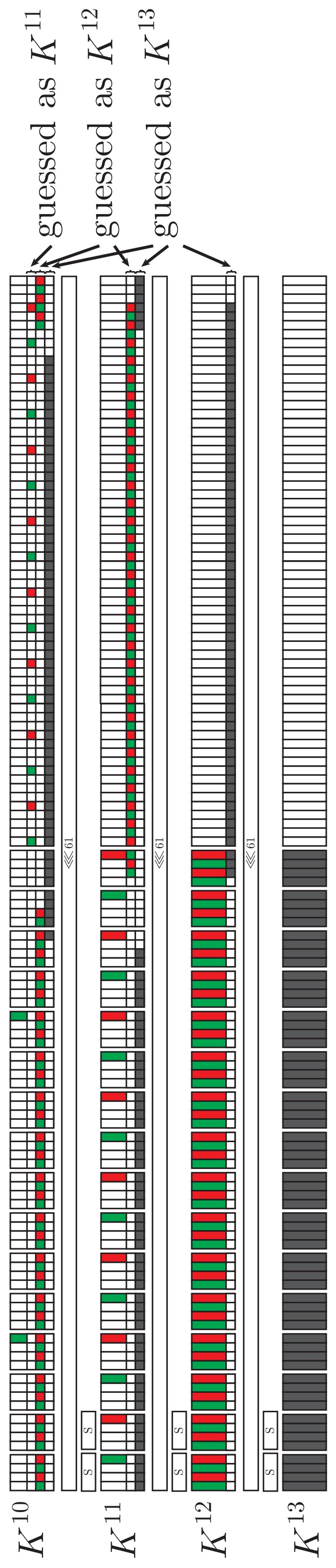


Figure 9.6: Involved round keys of PRESENT-128

9.5 Conclusion

We proposed the compact representation for the bit-based division property in this chapter. It is difficult to apply the bit-based division property to block ciphers whose block length is over 32 because of high time and memory complexity. The compact representation partially solves this problem. To demonstrate the advantage of our method, we applied this technique to 64-bit block cipher PRESENT. As a result, we attacked 12-round PRESENT-80 and 13-round PRESENT-128 by using new 9-round integral characteristic, and they are improved by two rounds than the previous best integral attacks.

Chapter 10

Follow-Up Works and Conclusion

After the proposal of the division property at EUROCRYPT 2015 [Tod15b], many third-party follow-up results have been reported. At INDOCRYPT 2015, Zhang and Wu applied the division property to generalized Feistel networks [ZW15] and analyzed two lightweight block ciphers LBlock [WZ11] and TWINE [SMMK12]. At CRYPTO 2016, Boura and Canteaut showed another view of the division property called the *parity set*. At ASIACRYPT 2016, Xiang, Zhang, Bao, and Lin proposed that the propagation of the division property can be easily evaluated by using mixed integer linear programming [XZBL16]. Nowadays, new ciphers that discuss the security for the analysis using the division property in advance have been proposed like SKINNY [BJK+16], Mysterion [JSV16], and Sparx and LAX [DPU+16].

10.1 Technical Improvement

This section summarizes two important third-party follow-up works.

10.1.1 Parity Set.

Independently of the bit-based division property, Boura and Canteaut also showed another view of the division property and introduced the concept of the parity set [BC16] and the link with the Reed-Muller codes. Actually, the parity set is regarded as the bit-based division property for an S-box. While the propagation of the integer-based division property focuses on the Hamming weight of $u \in \mathbb{F}_2^n$ that the parity is unknown, the parity set focus on $u \in \mathbb{F}_2^n$ itself that the parity is unknown. They evaluated the propagation for the PRESENT S-box and showed the low-data distinguishers. Moreover, the relationship between the parity set and the bit-based division property were discussed in [TM16b].

10.1.2 MILP-Based Propagation Search.

The most important and difficult open problem that we left is to construct efficient algorithm to evaluate the propagation of the division property. In the application to MISTY1 [Tod15b], we implemented the propagation using C++, but it requires much memory and time complexities. Moreover, at FSE2016 [TM16a], we could not apply the bit-based division property to Simon family except for Simon32 because of the high complexity. This problem was solved by using the state-of-the-art technique using mixed integer linear programming by Xiang et al. at ASIACRYPT 2016 [XZBL16]. The approach using MILP was first introduced by Mouha et al. for evaluating the lower bound of the number of active S-boxes on word-oriented ciphers,[MWGP11]. However, several ciphers do not have word-oriented structure. Therefore, Sun et al. then developed a method to model all possible differential propagations bit by bit even for the S-box [SHW+14]. At ASIACRYPT 2016, Xiang et al. first introduced the *division trail*. To analyze r-round ciphers, the following propagation of the division property

$$\mathbb{K}_0 \to \mathbb{K}_1 \to \cdots \to \mathbb{K}_r$$

is evaluated, where $\mathcal{D}_{\mathbb{K}_{i-1}}$ is the division property for the input of ith round function. Then, for $(\vec{k}_0, \vec{k}_1, \ldots, \vec{k}_r) \in (\mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r)$, if $\vec{k}_{i-1}$ can propagate to $\vec{k}_i$ for all $i \in \{1, 2, \ldots, r\}$,

$(\vec{k}_0, \vec{k}_1, \dots, \vec{k}_r)$ is called r-round division trail. If there is a division trail that r-round output becomes unknown, the r-round output is unknown. They developed a method to model the propagation rules for the division property and showed that all division trails are effectively evaluated using MILP. As a result, they get new integral characteristics on SIMON family, Simeck family, PRESENT, and RECTANGLE [1].

10.2 Summary of Works about Division Property

Table 10.1: Summary of works related to the division property

Author	Note	Month	Year	Reference
Todo	proposal of Division Property	May	2015	EUROCRYPT [Tod15b]
Todo	break of full MISTY1	Aug.	2015	CRYPTO [Tod15a]
Zhang and Wu	application to GFN	Dec.	2015	INDOCRYPT [ZW15]
Todo and Morii	bit-based division property	Mar.	2016	FSE [TM16a]
Journault et al.	Mysterion	Mar.	2016	DCC [JSV16]
Sasaki and Todo	application to LILLIPUT	Aug.	2016	SAC [ST16]
Boura and Canteaut	parity set	Aug.	2016	CRYPTO [BC16]
Bar-On and Keller	improved analysis on MISTY1	Aug.	2016	CRYPTO [BK16]
Beierle et al.	SKINNY and MANTIS	Aug.	2016	CRYPTO [BJK+16]
Todo and Morii	application to PRESENT	Nov.	2016	CANS [TM16b]
Xiang et al.	MILP-based evaluation	Dec.	2016	ASIACRYPT [XZBL16]
Diny et al.	SPARX and LAX	Dec.	2016	ASIACRYPT [DPU+16]

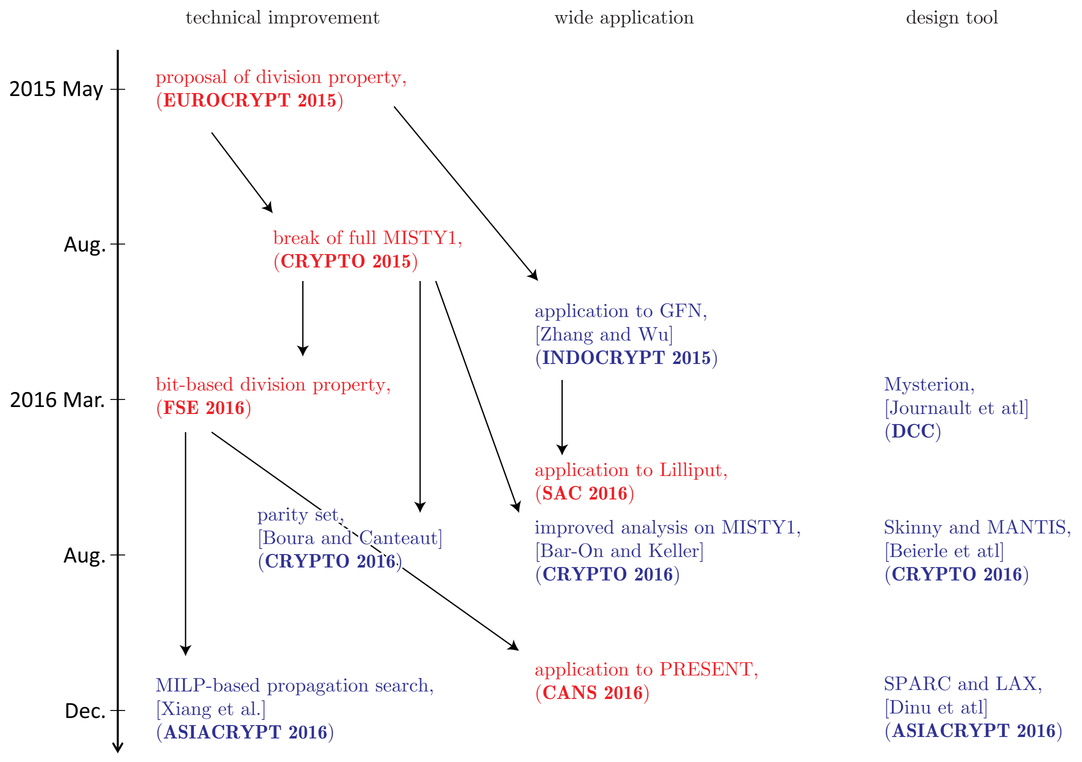

technical improvement
wide application
design tool
2015 May
Aug.
2016 Mar.
Aug.
Dec.
proposal of division property, (EUROCRYPT 2015)
break of full MISTY1, (CRYPTO 2015)
application to GFN, [Zhang and Wu] (INDOCRYPT 2015)
bit-based division property, (FSE 2016)
Mysterion, [Journault et atl] (DCC)
application to Lilliput, (SAC 2016)
parity set, [Boura and Canteaut] (CRYPTO 2016)
improved analysis on MISTY1, [Bar-On and Keller] (CRYPTO 2016)
Skinny and MANTIS, [Beierle et atl] (CRYPTO 2016)
application to PRESENT, (CANS 2016)
MILP-based propagation search, [Xiang et al.] (ASIACRYPT 2016)
SPARC and LAX, [Dinu et atl] (ASIACRYPT 2016)

Figure 10.1: Summary of works related to the division property

[1]We also independently evaluated the propagation of the division property on PRESENT in [TM16b] and get the same integral characteristics. In that paper, we introduced the compact representation for the division property to evaluate the propagation efficiently.

We finally summarize all works[2] related to the division property. Table 10.1 summarizes the publication list. Figure 10.1 shows each relationship, where papers labeled red color are proposed by ourself and ones labeled blue color are proposed by third party. We expect further improvement and evolution of the division property.

[2]Only internationally published works are surveyed.

Appendix A

Division Property of Texts Encrypted by Six Rounds with Two FL Layers

When the input set has the division property $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\{[6,2,7,7,2,7,7,2,7,7,2,7]\}}$, the division property of the set of texts encrypted by 6 rounds without the first and last FL layers is represented as $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\mathbb{K}}$, where $\mathbb{K}$ has 131 vectors as follows:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 1]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 1, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 3]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 1]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 2]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 1]	[0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 0, 1]	[0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 1, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 4, 1, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 3]
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 2]	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2, 1]	[0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 2]	[0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1]
[0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 2, 0]	[0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 1]	[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, 0, 1, 1, 0, 0, 2]	[0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1]	[0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 2, 0]	[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0]	[0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 1]	[0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 1, 5, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2]	[0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 1]
[0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0]	[0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 1]	[0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 1, 0]	[0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 1]	[0, 0, 0, 0, 0, 0, 0, 2, 1, 0, 1, 0]	[0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 2, 4, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 3]	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 2]	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 1]	[0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 2]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1]	[0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 2, 0]	[0, 0, 0, 0, 0, 0, 1, 0, 0, 2, 0, 1]	[0, 0, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 3, 0, 0]	[0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 2]	[0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1]	[0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 2, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1]	[0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0]	[0, 0, 0, 0, 0, 0, 1, 0, 1, 2, 0, 0]	[0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 1, 0]	[0, 0, 0, 0, 0, 0, 1, 0, 2, 1, 0, 0]	[0, 0, 0, 0, 0, 0, 1, 0, 5, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 2]
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1]	[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 2, 0]	[0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1]	[0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0]
[0, 0, 0, 0, 0, 0, 1, 1, 0, 2, 0, 0]	[0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1]	[0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0]	[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 1, 4, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 1]	[0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 1, 0]	[0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 2, 3, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2]	[0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 1]	[0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0]
[0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 0, 1]	[0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 1, 0]	[0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0]	[0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 1, 0]	[0, 0, 0, 0, 0, 0, 2, 0, 1, 1, 0, 0]	[0, 0, 0, 0, 0, 0, 2, 0, 4, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 1, 0]	[0, 0, 0, 0, 0, 0, 2, 1, 0, 1, 0, 0]	[0, 0, 0, 0, 0, 0, 2, 1, 3, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 1, 0]	[0, 0, 0, 0, 0, 0, 3, 0, 3, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 3, 1, 2, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 3, 2, 1, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 4, 0, 0, 1, 0, 0]	[0, 0, 0, 0, 0, 0, 5, 0, 2, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 5, 1, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 5, 2, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 7, 0, 1, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 7, 1, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]	[1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]	[1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
[1, 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, 0, 0, 0, 0, 0, 0, 0]	

Assume that $\mathbb{X}$ has the division property $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\mathbb{K}}$. Let $e_i \in \mathbb{Z}^{12}$ be a unit vector whose ith element is one and the others are zero. When there is not e_i in $\mathbb{K}$, $\bigoplus_{\vec{x}\in\mathbb{X}} x_i = 0$. Since the vector $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ is not included in 131 vectors, the first 7 bits are balanced.

Appendix B

Propagation Characteristic Table for FI function

Table B.1: Propagation from $\mathcal{D}^{7,2,7}_{\{[0,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$

$\vec{k}$	$\mathbb{K}$
[0 0 0]	[0 0 0]
[0 0 1]	[0 0 1] [0 1 0] [1 0 0]
[0 0 2]	[0 0 1] [0 1 0] [1 0 0]
[0 0 3]	[0 0 1] [0 2 0] [1 0 0]
[0 0 4]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[0 0 5]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [2 0 0]
[0 0 6]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [3 0 0]
[0 0 7]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[0 1 0]	[0 0 1] [0 1 0] [1 0 0]
[0 1 1]	[0 0 1] [0 1 0] [2 0 0]
[0 1 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[0 1 3]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[0 1 4]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[0 1 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [3 0 0]
[0 1 6]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[0 1 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[0 2 0]	[0 0 1] [0 1 0] [1 0 0]
[0 2 1]	[0 0 1] [0 1 0] [2 0 0]
[0 2 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[0 2 3]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[0 2 4]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[0 2 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [3 0 0]
[0 2 6]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[0 2 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]

Table B.2: Propagation from $\mathcal{D}^{7,2,7}_{\{[1,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$

$\vec{k}$	$\mathbb{K}$
[1 0 0]	[0 0 1] [0 1 0] [1 0 0]
[1 0 1]	[0 0 1] [0 1 0] [2 0 0]
[1 0 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[1 0 3]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[1 0 4]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[1 0 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [3 0 0]
[1 0 6]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[1 0 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[1 1 0]	[0 0 1] [0 1 0] [1 0 0]
[1 1 1]	[0 0 1] [0 1 0] [2 0 0]
[1 1 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[1 1 3]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[1 1 4]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[1 1 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [3 0 0]
[1 1 6]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[1 1 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[1 2 0]	[0 0 1] [0 1 0] [2 0 0]
[1 2 1]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[1 2 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[1 2 3]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[1 2 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[1 2 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[1 2 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[1 2 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]

Table B.3: Propagation from $\mathcal{D}^{7,2,7}_{\{[2,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$

$\vec{k}$	$\mathbb{K}$
[2 0 0]	[0 0 1] [0 1 0] [1 0 0]
[2 0 1]	[0 0 1] [0 1 0] [2 0 0]
[2 0 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[2 0 3]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [2 0 0]
[2 0 4]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[2 0 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [3 0 0]
[2 0 6]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[2 0 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[2 1 0]	[0 0 1] [0 1 0] [2 0 0]
[2 1 1]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[2 1 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[2 1 3]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[2 1 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[2 1 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[2 1 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[2 1 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[2 2 0]	[0 0 1] [0 1 0] [2 0 0]
[2 2 1]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[2 2 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[2 2 3]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[2 2 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[2 2 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[2 2 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[2 2 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]

Table B.4: Propagation from $\mathcal{D}^{7,2,7}_{\{[3,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$

$\vec{k}$	$\mathbb{K}$
[3 0 0]	[0 0 1] [0 1 0] [2 0 0]
[3 0 1]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[3 0 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[3 0 3]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[3 0 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[3 0 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[3 0 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[3 0 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[3 1 0]	[0 0 1] [0 1 0] [2 0 0]
[3 1 1]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[3 1 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[3 1 3]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[3 1 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[3 1 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[3 1 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[3 1 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[3 2 0]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[3 2 1]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[3 2 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[3 2 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[3 2 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[3 2 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[3 2 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[3 2 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]

Table B.5: Propagation from $\mathcal{D}^{7,2,7}_{\{[4,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$

$\vec{k}$	$\mathbb{K}$
[4 0 0]	[0 0 1] [0 1 0] [2 0 0]
[4 0 1]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[4 0 2]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[4 0 3]	[0 0 2] [0 1 1] [1 0 1] [1 1 0] [3 0 0]
[4 0 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[4 0 5]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[4 0 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[4 0 7]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[4 1 0]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[4 1 1]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[4 1 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[4 1 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[4 1 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[4 1 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[4 1 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[4 1 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]
[4 2 0]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[4 2 1]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[4 2 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[4 2 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[4 2 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[4 2 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[4 2 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[4 2 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]

Table B.6: Propagation from $\mathcal{D}^{7,2,7}_{\{[5,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$

$\vec{k}$	$\mathbb{K}$
[5 0 0]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[5 0 1]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[5 0 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[5 0 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[5 0 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[5 0 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[5 0 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[5 0 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]
[5 1 0]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[5 1 1]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[5 1 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[5 1 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[5 1 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[5 1 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[5 1 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[5 1 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]
[5 2 0]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[5 2 1]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[5 2 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[5 2 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[5 2 4]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[5 2 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[5 2 6]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]
[5 2 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [4 0 2] [4 1 1] [4 2 0] [6 0 1] [6 1 0]

Table B.7: Propagation from $\mathcal{D}^{7,2,7}_{\{[6,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$

$\vec{k}$	$\mathbb{K}$
[6 0 0]	[0 0 2] [0 1 1] [0 2 0] [1 0 1] [1 1 0] [3 0 0]
[6 0 1]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[6 0 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[6 0 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [2 0 1] [2 1 0] [4 0 0]
[6 0 4]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[6 0 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [3 0 1] [3 1 0] [5 0 0]
[6 0 6]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[6 0 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]
[6 1 0]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[6 1 1]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[6 1 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[6 1 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[6 1 4]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[6 1 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[6 1 6]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]
[6 1 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [4 0 2] [4 1 1] [4 2 0] [6 0 1] [6 1 0]
[6 2 0]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[6 2 1]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[6 2 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[6 2 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[6 2 4]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[6 2 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[6 2 6]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]
[6 2 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [4 0 2] [4 1 1] [4 2 0] [6 0 1] [6 1 0]

Table B.8: Propagation from $\mathcal{D}^{7,2,7}_{\{[7,*,*]\}}$ to $\mathcal{D}^{7,2,7}_{\mathbb{K}}$

$\vec{k}$	$\mathbb{K}$
[7 0 0]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[7 0 1]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[7 0 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[7 0 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[7 0 4]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[7 0 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[7 0 6]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]
[7 0 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [4 0 2] [4 1 1] [4 2 0] [6 0 1] [6 1 0]
[7 1 0]	[0 0 2] [0 1 1] [0 2 0] [2 0 1] [2 1 0] [4 0 0]
[7 1 1]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[7 1 2]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[7 1 3]	[0 0 3] [0 1 2] [0 2 1] [1 0 2] [1 1 1] [1 2 0] [3 0 1] [3 1 0] [5 0 0]
[7 1 4]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[7 1 5]	[0 0 4] [0 1 3] [0 2 2] [1 0 3] [1 1 2] [1 2 1] [2 0 2] [2 1 1] [2 2 0] [4 0 1] [4 1 0] [6 0 0]
[7 1 6]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [3 0 2] [3 1 1] [3 2 0] [5 0 1] [5 1 0] [7 0 0]
[7 1 7]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [2 0 3] [2 1 2] [2 2 1] [4 0 2] [4 1 1] [4 2 0] [6 0 1] [6 1 0]
[7 2 0]	[0 0 5] [0 1 4] [0 2 3] [1 0 4] [1 1 3] [1 2 2] [3 0 3] [3 1 2] [3 2 1] [5 0 2] [5 1 1] [5 2 0] [7 0 1] [7 1 0]
[7 2 1]	[0 0 6] [0 1 5] [0 2 4] [1 0 5] [1 1 4] [1 2 3] [2 0 4] [2 1 3] [2 2 2] [4 0 3] [4 1 2] [4 2 1] [6 0 2] [6 1 1] [6 2 0]
[7 2 2]	[0 0 6] [0 1 5] [0 2 4] [1 0 5] [1 1 4] [1 2 3] [2 0 4] [2 1 3] [2 2 2] [4 0 3] [4 1 2] [4 2 1] [6 0 2] [6 1 1] [6 2 0]
[7 2 3]	[0 0 6] [0 1 5] [0 2 4] [1 0 5] [1 1 4] [1 2 3] [2 0 4] [2 1 3] [2 2 2] [4 0 3] [4 1 2] [4 2 1] [6 0 2] [6 1 1] [6 2 0]
[7 2 4]	[0 0 7] [0 1 6] [0 2 5] [1 0 6] [1 1 5] [1 2 4] [2 0 5] [2 1 4] [2 2 3] [3 0 4] [3 1 3] [3 2 2] [5 0 3] [5 1 2] [5 2 1] [7 0 2] [7 1 1] [7 2 0]
[7 2 5]	[0 0 7] [0 1 6] [0 2 5] [1 0 6] [1 1 5] [1 2 4] [2 0 5] [2 1 4] [2 2 3] [3 0 4] [3 1 3] [3 2 2] [5 0 3] [5 1 2] [5 2 1] [7 0 2] [7 1 1] [7 2 0]
[7 2 6]	[0 2 7] [1 1 7] [1 2 6] [2 0 7] [2 1 6] [2 2 5] [3 0 6] [3 1 5] [3 2 4] [4 0 5] [4 1 4] [4 2 3] [5 0 4] [5 1 3] [5 2 2] [7 0 3] [7 1 2] [7 2 1]
[7 2 7]	[7 2 7]

Bibliography

[AB15] Yann Le Corre Alex Biryukov, Johann Großschädl. CryptoLUX, Lightweight Cryptography. https://www.cryptolux.org/index.php/Lightweight_Cryptography, 2015.

[ABB+14] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Florian Mendel, Bart Mennink, Nicky Mouha, Qingju Wang, and Kan Yasuda. PRIMATEs v1.02, 2014. Submission to CAESAR competition.

[ABD+13] Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the indifferentiability of key-alternating ciphers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO Part I*, volume 8042 of *LNCS*, pages 531–550, 2013.

[ABK98] Ross Anderson, Eli Biham, and Lars Knudsen. Serpent: A proposal for the Advanced Encryption Standard, 1998. One of the five finalists of the AES contest.

[AIK+00] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis. In Douglas R. Stinson and Stafford E. Tavares, editors, *SAC*, volume 2012 of *LNCS*, pages 39–56. Springer, 2000.

[AJN15] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX v2.0, 2015. Submission to CAESAR competition.

[ALLW14] Farzaneh Abed, Eik List, Stefan Lucks, and Jakob Wenzel. Differential cryptanalysis of round-reduced Simon and Speck. In Carlos Cid and Christian Rechberger, editors, *FSE*, volume 8540 of *LNCS*, pages 525–545. Springer, 2014.

[Bar15a] Achiya Bar-On. A 2^{70} attack on the full MISTY1. IACR Cryptology ePrint Archive, Report 2015/746, available at http://eprint.iacr.org/2015/746, 2015.

[Bar15b] Achiya Bar-On. Improved higher-order differential attacks on MISTY1. In Gregor Leander, editor, *FSE*, volume 9054 of *LNCS*, pages 28–47. Springer, 2015.

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *LNCS*, pages 12–23. Springer, 1999.

[BC13] Christina Boura and Anne Canteaut. On the influence of the algebraic degree of f^{-1} on the algebraic degree of $\mathrm{G} \circ \mathrm{F}$. *IEEE Transactions on Information Theory*, 59(1):691–702, 2013.

[BC16] Christina Boura and Anne Canteaut. Another view of the division property. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO Part I*, volume 9814 of *LNCS*, pages 654–682. Springer, 2016.

[BCC11] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of Keccak and *Luffa*. In Antoine Joux, editor, *FSE*, volume 6733 of *LNCS*, pages 252–269. Springer, 2011.

[BF00] Steve Babbage and Laurent Frisch. On MISTY1 higher order differential cryptanalysis. In Dongho Won, editor, *ICISC*, volume 2015 of *LNCS*, pages 22–36. Springer, 2000.

[BFMT15] Thierry P. Berger, Julien Francq, Marine Minier, and Gaël Thomas. Extended generalized Feistel networks using matrix representation to propose a new lightweight block cipher: Lilliput. *IEEE Transactions on Computers*, 65:2074–2089, 2015.

[BGW$^+$13] Andrey Bogdanov, Huizheng Geng, Meiqin Wang, Long Wen, and Baudoin Collard. Zero-correlation linear cryptanalysis with FFT and improved attacks on ISO standards camellia and CLEFIA. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *SAC*, volume 8282 of *LNCS*, pages 306–323. Springer, 2013.

[BJK$^+$16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO Part II*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016.

[BJV04] Thomas Baignères, Pascal Junod, and Serge Vaudenay. How far can we go beyond linear cryptanalysis? In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *LNCS*, pages 432–450. Springer, 2004.

[BK16] Achiya Bar-On and Nathan Keller. A 2ˆ70 attack on the full MISTY1. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO Part I*, volume 9814 of *LNCS*, pages 435–456. Springer, 2016.

[BKL$^+$07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.

[BKL$^+$12] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, François-Xavier Standaert, John P. Steinberger, and Elmar Tischhauser. Key-alternating ciphers in a provable setting: Encryption using a small number of public permutations - (extended abstract). In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 45–62. Springer, 2012.

[BMT13] Thierry P. Berger, Marine Minier, and Gaël Thomas. Extended generalized Feistel networks using matrix representation. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *SAC*, volume 8282 of *LNCS*, pages 289–305. Springer, 2013.

[BNS14] Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and improving impossible differential attacks: Applications to clefia, camellia, lblock and simon. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT Part I*, volume 8873 of *LNCS*, pages 179–199. Springer, 2014.

[BR03] Paulo S. L. M. Barreto and Vincent Rijmen. The Whirlpool hashing function, 2003. submitted to the NESSIE project, available at http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html.

[BR11] Andrey Bogdanov and Vincent Rijmen. Zero-correlation linear cryptanalysis of block ciphers. IACR Cryptology ePrint Archive, Report 2011/123, available at http://eprint.iacr.org/2011/123, 2011.

[BR14] Andrey Bogdanov and Vincent Rijmen. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Des. Codes Cryptography*, 70(3):369–383, 2014.

[BRV14] Alex Biryukov, Arnab Roy, and Vesselin Velichkov. Differential analysis of block ciphers SIMON and SPECK. In Carlos Cid and Christian Rechberger, editors, *FSE*, volume 8540 of *LNCS*, pages 546–570. Springer, 2014.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO*, volume 537 of *LNCS*, pages 2–21. Springer, 1990.

[BS01] Alex Biryukov and Adi Shamir. Structural cryptanalysis of SASAS. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *LNCS*, pages 394–405. Springer, 2001.

[BSS$^+$13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers, 2013.

[BSS$^+$15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. SIMON and SPECK: block ciphers for the internet of things. IACR Cryptology ePrint Archive, Report 2015/585, available at http://eprint.iacr.org/2015/585, 2015.

[CRY13] CRYPTREC. Specifications of e-government recommended ciphers. available at http://www.cryptrec.go.jp/english/method.html, 2013.

[CSQ07] Baudoin Collard, François-Xavier Standaert, and Jean-Jacques Quisquater. Improving the time complexity of Matsui's linear cryptanalysis. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *LNCS*, pages 77–88. Springer, 2007.

[CSW08] Christophe De Cannière, Hisayoshi Sato, and Dai Watanabe. Hash function *Luffa* - a SHA-3 candidate, 2008. Available at http://hitachi.com/rd/yrl/crypto/luffa/round1archive/Luffa_Specification.pdf.

[CV02] Anne Canteaut and Marion Videau. Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *LNCS*, pages 518–533. Springer, 2002.

[DBPA11] Joan Daemen, Guido Bertoni, Michaël Peeters, and Gilles Van Assche. The Keccak reference version 3.0, 2011.

[Dem02] Hüseyin Demirci. Square-like attacks on reduced rounds of IDEA. In Kaisa Nyberg and Howard M. Heys, editors, *SAC*, volume 2595 of *LNCS*, pages 147–159. Springer, 2002.

[DEMS14] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1, 2014. Submission to CAESAR competition.

[DH77] Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *IEEE Computer*, 10(6):74–84, 1977.

[DK08] Orr Dunkelman and Nathan Keller. An improved impossible differential attack on MISTY1. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *LNCS*, pages 441–454. Springer, 2008.

[DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher Square. In Eli Biham, editor, *FSE*, volume 1267 of *LNCS*, pages 149–165. Springer, 1997.

[DKS12] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The Even-Mansour scheme revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 336–354. Springer, 2012.

[DPAR00] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. The Noekeon block cipher., 2000. submitted to the NESSIE project, available at http://gro.noekeon.org/.

[DPU$^+$16] Daniel Dinu, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Johann Großschädl, and Alex Biryukov. Design strategies for ARX with provable bounds: Sparx and LAX. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT Part I*, volume 10031 of *LNCS*, pages 484–513. Springer, 2016.

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

[DS08] Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *LNCS*, pages 116–126. Springer, 2008.

[EM97] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *J. Cryptology*, 10(3):151–162, 1997.

[FKL$^+$00] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In Bruce Schneier, editor, *FSE*, volume 1978 of *LNCS*, pages 213–230. Springer, 2000.

[GM00] Henri Gilbert and Marine Minier. A collision attack on 7 rounds of Rijndael. In *AES Candidate Conference*, pages 230–241, 2000.

[GPP11] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *LNCS*, pages 222–239. Springer, 2011.

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *LNCS*, pages 326–341. Springer, 2011.

[HQ01] Yeping He and Sihan Qing. Square attack on reduced Camellia cipher. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS*, volume 2229 of *LNCS*, pages 238–245. Springer, 2001.

[HTK04] Yasuo Hatano, Hidema Tanaka, and Toshinobu Kaneko. Optimization for the algebraic method and its application to an attack of MISTY1. *IEICE Transactions*, 87-A(1):18–27, 2004.

[IS12] Takanori Isobe and Kyoji Shibutani. All subkeys recovery attack on block ciphers: Extending meet-in-the-middle approach. In Lars R. Knudsen and Huapeng Wu, editors, *SAC*, volume 7707 of *LNCS*, pages 202–221. Springer, 2012.

[IS13] Takanori Isobe and Kyoji Shibutani. Generic key recovery attack on Feistel scheme. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT Part I*, volume 8269 of *LNCS*, pages 464–485. Springer, 2013.

[ISO05] ISO/IEC. JTC1: ISO/IEC 18033: Security techniques – encryption algorithms – part 3: Block ciphers, 2005.

[ISO12] ISO/IEC. ISO/IEC 29192-2. information technology - security techniques - lightweight cryptography – part 2: Block ciphers, 2012.

[JSV16] Anthony Journault, François-Xavier Standaert, and Kerem Varici. Improving the security and efficiency of block ciphers based on LS-designs. *Des. Codes Cryptography*, pages 1–15, 2016.

[KLL$^+$14a] Elif Bilge Kavun, Martin Mehl Lauridsen, Gregor Leander, Christian Rechberger, Peter Schwabe, and Tolga Yalçin. PRØST v1.1, 2014. Submission to CAESAR competition.

[KLL$^+$14b] Elif Bilge Kavun, Martin Mehl Lauridsen, Gregor Leander, Christian Rechberger, Peter Schwabe, and Tolga Yalçin. PRØST, reference implemenations in C for SUPERCOP, 2014. Submission to CAESAR competition.

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO Part I*, volume 9215, pages 161–185. Springer, 2015.

[Knu94] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *FSE*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.

[Knu02] Lars R. Knudsen. The security of Feistel ciphers with six rounds or less. *J. Cryptology*, 15(3):207–222, 2002.

[Kor05] Korea Information Security Agency. SEED 128 algorithm specification, 2005. RFC4269, Available at `https://seed.kisa.or.kr/html/egovframework/iwt/ds/ko/ref/[2]_SEED+128_Specification_english_M.pdf`.

[KR07] Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *LNCS*, pages 315–324. Springer, 2007.

[KW02] Lars R. Knudsen and David Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *FSE*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.

[Lai94] Xuejia Lai. Higher order derivatives and differential cryptanalysis. In *Communications and Cryptography*, volume 276 of *The Springer International Series in Engineering and Computer Science*, pages 227–233. Springer, 1994.

[LP12] Rodolphe Lampe and Jacques Patarin. Security of Feistel schemes with new and various tools. 2012.

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.

[Luc96] Stefan Lucks. Faster Luby-Rackoff ciphers. In Dieter Gollmann, editor, *FSE*, volume 1039 of *LNCS*, pages 189–203. Springer, 1996.

[Luc01] Stefan Lucks. The saturation attack - A bait for Twofish. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *LNCS*, pages 1–15. Springer, 2001.

[LWZ11] Yanjun Li, Wenling Wu, and Lei Zhang. Improved integral attacks on reduced-round CLEFIA block cipher. In Souhwan Jung and Moti Yung, editors, *WISA*, volume 7115 of *LNCS*, pages 28–39. Springer, 2011.

[Mat93] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *EUROCRYPT*, volume 765 of *LNCS*, pages 386–397. Springer, 1993.

[Mat96] Mitsuru Matsui. New structure of block ciphers with provable security against differential and linear cryptanalysis. In Dieter Gollmann, editor, *FSE*, volume 1039 of *LNCS*, pages 205–218. Springer, 1996.

[Mat97] Mitsuru Matsui. New block encryption algorithm MISTY. In Eli Biham, editor, *FSE*, volume 1267 of *LNCS*, pages 54–68. Springer, 1997.

[MGH$^+$14] Pawel Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wøjcik. ICEPOLE v1, 2014. Submission to CAESAR competition.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Inscrypt*, volume 7537 of *LNCS*, pages 57–76. Springer, 2011.

[Nat77] National Bureau of Standards. *DATA ENCRYPTION STANDARD (DES)*, 1977. Federal Information Processing Standards Publication 46.

[Nat89] National Soviet Bureau of Standards. Information Processing System – Cryptographic Protection – Cryptographic Algorithm GOST 28147-89, 1989.

[NES04] NESSIE. New european schemes for signatures, integrity, and encryption. available at https://www.cosic.esat.kuleuven.be/nessie/, 2004.

[NK95] Kaisa Nyberg and Lars R. Knudsen. Provable security against a differential attack. *J. Cryptology*, 8(1):27–37, 1995.

[NWW11] Phuong Ha Nguyen, Hongjun Wu, and Huaxiong Wang. Improving the algorithm 2 in multidimensional linear cryptanalysis. In Udaya Parampalli and Philip Hawkes, editors, *ACISP*, volume 6812 of *LNCS*, pages 61–74. Springer, 2011.

[NWWL10] Phuong Ha Nguyen, Lei Wei, Huaxiong Wang, and San Ling. On multidimensional linear cryptanalysis. In Ron Steinfeld and Philip Hawkes, editors, *ACISP*, volume 6168 of *LNCS*, pages 37–52. Springer, 2010.

[Nyb94] Kaisa Nyberg. Linear approximation of block ciphers. In Alfredo De Santis, editor, *EUROCRYPT*, volume 950 of *LNCS*, pages 439–444. Springer, 1994.

[OM00] Hidenori Ohta and Mitsuru Matsui. A description of the MISTY1 encryption algorithm. available at https://tools.ietf.org/html/rfc2994, 2000.

[Pat92] Jacques Patarin. How to construct pseudorandom and super pseudorandom permutations from one single pseudorandom function. In Rainer A. Rueppel, editor, *EUROCRYPT*, volume 658 of *LNCS*, pages 256–266. Springer, 1992.

[Pat03] Jacques Patarin. Luby-Rackoff: 7 rounds are enough for $2^{n(1-\epsilon)}$security. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *LNCS*, pages 513–529. Springer, 2003.

[Pat04] Jacques Patarin. Security of random Feistel schemes with 5 or more rounds. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *LNCS*, pages 106–122. Springer, 2004.

[Pat10] Jacques Patarin. Security of balanced and unbalanced Feistel schemes with linear non equalities. 2010.

[SHW$^+$14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT Part I*, volume 8873 of *LNCS*, pages 158–178. Springer, 2014.

[SHZ$^+$15] Bing Sun, Xin Hai, Wenyu Zhang, Lei Cheng, and Zhichao Yang. New observation on division property. IACR Cryptology ePrint Archive, Report 2015/459, available at http://eprint.iacr.org/2015/459, 2015.

[SK12] Naoki Shibayama and Toshinobu Kaneko. A peculiar higher order differential of CLEFIA. In *ISITA*, pages 526–530. IEEE, 2012.

[SM10] Tomoyasu Suzaki and Kazuhiko Minematsu. Improving the generalized Feistel. In Seokhie Hong and Tetsu Iwata, editors, *FSE*, volume 6147 of *LNCS*, pages 19–39. Springer, 2010.

[SMKT99] Takeshi Shimoyama, Shiho Moriai, Toshinobu Kaneko, and Shigeo Tsujii. Improved higher order differential attack and its application to Nyberg-Knudsen's designed block cipher. *IEICE Transactions*, 82-A(9):1971–1980, 1999.

[SMMK12] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. TWINE : A lightweight block cipher for multiple platforms. In Lars R. Knudsen and Huapeng Wu, editors, *SAC*, volume 7707 of *LNCS*, pages 339–354. Springer, 2012.

[SS06] Taizo Shirai and Kyoji Shibutani. On Feistel structures using a diffusion switching mechanism. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *LNCS*, pages 41–56. Springer, 2006.

[SSA$^+$07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher CLEFIA (extended abstract). In Alex Biryukov, editor, *FSE*, volume 4593 of *LNCS*, pages 181–195. Springer, 2007.

[ST16] Yu Sasaki and Yosuke Todo. New differential bounds and division property of Lilliput: Block cipher with extended generalized Feistel network. In *SAC*, 2016. (in press).

[STA$^+$14] Yu Sasaki, Yosuke Todo, Kazumaro Aoki, Yusuke Naito, Takeshi Sugawara, Yumiko Murakami, Mitsuru Matsui, and Shoichi Hirose. Minalpher v1, 2014. Submission to CAESAR competition.

[STKT06] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In Min Surp Rhee and Byoungcheon Lee, editors, *ICISC*, volume 4296 of *LNCS*, pages 29–40. Springer, 2006.

[SW12a] Yu Sasaki and Lei Wang. Comprehensive study of integral analysis on 22-round LBlock. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *ICISC*, volume 7839 of *LNCS*, pages 156–169. Springer, 2012.

[SW12b] Yu Sasaki and Lei Wang. Meet-in-the-middle technique for integral attacks against Feistel ciphers. In Lars R. Knudsen and Huapeng Wu, editors, *SAC*, volume 7707 of *LNCS*, pages 234–251. Springer, 2012.

[SY11] Yu Sasaki and Kan Yasuda. Known-key distinguishers on 11-round Feistel and collision attacks on its hashing modes. In Antoine Joux, editor, *FSE*, volume 6733 of *LNCS*, pages 397–415. Springer, 2011.

[TA14] Yosuke Todo and Kazumaro Aoki. FFT key recovery for integral attack. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *CANS*, volume 8813 of *LNCS*, pages 64–81. Springer, 2014.

[TA15] Yosuke Todo and Kazumaro Aoki. Fast fourier transform key recovery for integral attacks. *IEICE Transactions*, 98-A(9):1944–1952, 2015.

[THK99] Hidema Tanaka, Kazuyuki Hisamatsu, and Toshinobu Kaneko. Strength of MISTY1 without FL function for higher order differential attack. In Marc P. C. Fossorier, Hideki Imai, Shu Lin, and Alain Poli, editors, *AAECC-13*, volume 1719 of *LNCS*, pages 221–230. Springer, 1999.

[TM16a] Yosuke Todo and Masakatu Morii. Bit-based division property and application to Simon family. In Thomas Peyrin, editor, *FSE*, volume 9783 of *LNCS*, pages 357–377. Springer, 2016.

[TM16b] Yosuke Todo and Masakatu Morii. Compact representation for division property. In Sara Foresti and Giuseppe Persiano, editors, *CANS*, volume 10052 of *LNCS*, pages 19–35. Springer, 2016.

[Tod15a] Yosuke Todo. Integral cryptanalysis on full MISTY1. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO Part I*, volume 9215 of *LNCS*, pages 413–432. Springer, 2015.

[Tod15b] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT Part I*, volume 9056 of *LNCS*, pages 287–314. Springer, 2015.

[TSSK08] Yukiyasu Tsunoo, Teruo Saito, Maki Shigeri, and Takeshi Kawabata. Higher order differential attacks on reduced-round MISTY1. In Pil Joong Lee and Jung Hee Cheon, editors, *ICISC*, volume 5461 of *LNCS*, pages 415–431. Springer, 2008.

[U.S01] U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology. *Specification for the ADVANCED ENCRYPTION STANDARD (AES)*, 2001. Federal Information Processing Standards Publication 197.

[WLV$^+$14] Qingju Wang, Zhiqiang Liu, Kerem Varici, Yu Sasaki, Vincent Rijmen, and Yosuke Todo. Cryptanalysis of reduced-round SIMON32 and SIMON48. In Willi Meier and Debdeep Mukhopadhyay, editors, *INDOCRYPT*, volume 8885 of *LNCS*, pages 143–160. Springer, 2014.

[WW13] Shengbao Wu and Mingsheng Wang. Integral attacks on reduced-round PRESENT. In Sihan Qing, Jianying Zhou, and Dongmei Liu, editors, *ICICS*, volume 8233 of *LNCS*, pages 331–345. Springer, 2013.

[WZ11] Wenling Wu and Lei Zhang. LBlock: A lightweight block cipher. In Javier Lopez and Gene Tsudik, editors, *ACNS*, volume 6715 of *LNCS*, pages 327–344. Springer, 2011.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016 Part I*, volume 10031 of *LNCS*, pages 648–678, 2016.

[YPK02] Yongjin Yeom, Sangwoo Park, and Iljun Kim. On the security of CAMELLIA against the Square attack. In Joan Daemen and Vincent Rijmen, editors, *FSE*, volume 2365 of *LNCS*, pages 89–99. Springer, 2002.

[YZS$^+$15] Gangqiang Yang, Bo Zhu, Valentin Suder, Mark D. Aagaard, and Guang Gong. The Simeck family of lightweight block ciphers. In Tim Güneysu and Helena Handschuh, editors, *CHES 2015*, volume 9293 of *LNCS*, pages 307–329. Springer, 2015.

[ZMI89] Yuliang Zheng, Tsutomu Matsumoto, and Hideki Imai. On the construction of block ciphers provably secure and not relying on any unproved hypotheses. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 461–480. Springer, 1989.

[ZRHD08] Muhammad Reza Z'aba, Håvard Raddum, Matthew Henricksen, and Ed Dawson. Bit-pattern based integral attack. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *LNCS*, pages 363–381. Springer, 2008.

[ZW14] Lei Zhang and Wenling Wu. Differential analysis of the extended generalized Feistel networks. *Inf. Process. Lett.*, 114(12):723–727, 2014.

[ZW15] Huiling Zhang and Wenling Wu. Structural evaluation for generalized Feistel structures and applications to LBlock and TWINE. In Alex Biryukov and Vipul Goyal, editors, *INDOCRYPT*, volume 9462 of *LNCS*, pages 218–237. Springer, 2015.

[ZWW15] Huiling Zhang, Wenling Wu, and Yanfeng Wang. Integral attack against bit-oriented block ciphers. In Soonhak Kwon and Aaram Yun, editors, *ICISC*, volume 9558 of *LNCS*, pages 102–118. Springer, 2015.

List of Publications

Journal and Transactions

1. Yosuke Todo. Integral Cryptanalysis on Full MISTY1. *Journal of Cryptology*, First Online http://rd.springer.com/article/10.1007/s00145-016-9240-x.

2. Sho Sakikoyama, Yosuke Todo, Kazumaro Aoki, and Masakatu Morii. Efficient implementations for practical linear cryptanalysis and its application to FEAL-8X. *IEICE Transactions*, 99-A(1):31–38, 2016.

3. Yosuke Todo. Impossible differential attack against 14-round *Piccolo*-80 without relying on full code book. *IEICE Transactions*, 99-A(1):154–157, 2016.

4. Yosuke Todo and Kazumaro Aoki. Fast fourier transform key recovery for integral attacks. *IEICE Transactions*, 98-A(9):1944–1952, 2015.

5. Yosuke Todo. Upper bounds for the security of several Feistel networks. *IEICE Transactions*, 98-A(1):39–48, 2015.

6. Yosuke Todo, Yuki Ozawa, Toshihiro Ohigashi, and Masakatu Morii. Falsification attacks against WPA-TKIP in a realistic environment. *IEICE Transactions*, 95-D(2):588–595, 2012.

7. Masakatu Morii and Yosuke Todo. Cryptanalysis for RC4 and breaking WEP/WPA-TKIP. *IEICE Transactions*, 94-D(11):2087–2094, 2011.

International Conference

1. Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear invariant attack - Practical attack on Full SCREAM, iSCREAM, and Midori64. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2016.

2. Yosuke Todo and Masakatu Morii. Compact representation for division property. In Sara Foresti and Giuseppe Persiano, editors, *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, volume 10052 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2016.

3. Yu Sasaki and Yosuke Todo. New differential bounds and division property of LILLIPUT: block cipher with extended generalized Feistel network. *Selected Areas in Cryptography - SAC 2016. Proceedings* (in press).

4. Yosuke Todo and Kazumaro Aoki. Wide trail design strategy for binary MixColumns - enhancing lower bound of number of active S-boxes. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, volume 9696 of *Lecture Notes in Computer Science*, pages 467–484. Springer, 2016.

5. Yosuke Todo and Masakatu Morii. Bit-based division property and application to Simon family. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 357–377. Springer, 2016.

6. Yosuke Todo. Integral cryptanalysis on full MISTY1. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 413–432. Springer, 2015.

7. Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 287–314. Springer, 2015.

8. Sho Sakikoyama, Yosuke Todo, Kazumaro Aoki, and Masakatu Morii. How much can complexity of linear cryptanalysis be reduced? In Jooyoung Lee and Jongsung Kim, editors, *Information Security and Cryptology - ICISC 2014 - 17th International Conference, Seoul, Korea, December 3-5, 2014, Revised Selected Papers*, volume 8949 of *Lecture Notes in Computer Science*, pages 117–131. Springer, 2014.

9. Yosuke Todo and Kazumaro Aoki. FFT key recovery for integral attack. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings*, volume 8813 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.

10. Qingju Wang, Zhiqiang Liu, Kerem Varici, Yu Sasaki, Vincent Rijmen, and Yosuke Todo. Cryptanalysis of reduced-round SIMON32 and SIMON48. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 143–160. Springer, 2014.

11. Martin R. Albrecht, Jean-Charles Faugère, Robert Fitzpatrick, Ludovic Perret, Yosuke Todo, and Keita Xagawa. Practical cryptanalysis of a public-key encryption scheme based on new multivariate quadratic assumptions. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 446–464. Springer, 2014.

12. Yosuke Todo and Yu Sasaki. New property of diffusion switching mechanism on CLEFIA and its application to DFA. In Kazuo Sakiyama and Masayuki Terada, editors, *Advances in Information and Computer Security - 8th International Workshop on Security, IWSEC 2013, Okinawa, Japan, November 18-20, 2013, Proceedings*, volume 8231 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2013.

13. Yosuke Todo. Upper bounds for the security of several Feistel networks. In Colin Boyd and Leonie Simpson, editors, *Information Security and Privacy - 18th Australasian Conference, ACISP 2013, Brisbane, Australia, July 1-3, 2013. Proceedings*, volume 7959 of *Lecture Notes in Computer Science*, pages 302–317. Springer, 2013.

14. Tsubasa Tsukaune, Yosuke Todo, and Masakatu Morii. Proposal of a secure WEP operation against existing key recovery attacks and its evaluation. In *Seventh Asia Joint Conference on Information Security, AsiaJCIS 2012, Kaohsiung, Taiwan, August 9-10, 2012*, pages 25–30. IEEE, 2012.

15. Yosuke Todo, Toshihiro Ohigashi, and Masakatu Morii. Effective falsification attack on WPA-TKIP by modifying any packet to QoS packet. In *The Fifth Joint Workshop on Information Security, JWIS 2010, Guangzhou, P.R.China, August 5-6, 2010*, pages 118–132, 2010.

Invited and Keynote Talks

1. Yosuke Todo. Division Property: Efficient Method to Estimate Upper Bound of Algebraic Degree. Mycrypt 2016, Insight Talk, December 1, 2016.
2. Yosuke Todo. Nonlinear Invariant Attack. ASK 2016, Invited Talk, September 28, 2016.
3. Yosuke Todo. Division Property: Efficient Method to Estimate the Algebraic Degree. IWSEC 2016, Keynote Talk, September 13, 2016.
4. Yosuke Todo. Design for Involutive Symmetric-Key Primitive. IWSEC 2015, SCIS/CSS Invited Session, August 28, 2015.

Doctor Thesis, Kobe University
"Integral Cryptanalysis against Symmetric-Key Cryptosystems",164 pages
Submitted on January, 17, 2017
The date of publication is printed in cover of repository version published in Kobe University Repository Kernel.