Cryptanalysis of symmetric-key primitives and its application to backdoor design

Wang, Haoyang

2021

Wang, H. (2021). Cryptanalysis of symmetric-key primitives and its application to backdoor design. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/146046>

<https://doi.org/10.32657/10356/146046>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 20:30:39 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**CRYPTANALYSIS OF SYMMETRIC-KEY
PRIMITIVES AND ITS APPLICATION TO
BACKDOOR DESIGN**

HAOYANG WANG

SCHOOL OF PHYSICAL AND MATHEMATICAL SCIENCES

2021

CRYPTANALYSIS OF SYMMETRIC-KEY PRIMITIVES AND ITS APPLICATION TO BACKDOOR DESIGN

HAOYANG WANG

SCHOOL OF PHYSICAL AND MATHEMATICAL SCIENCES

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research done by me except where otherwise stated in this thesis. The thesis work has not been submitted for a degree or professional qualification to any other university or institution. I declare that this thesis is written by myself and is free of plagiarism and of sufficient grammatical clarity to be examined. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

29/12/2020

Haoyang Wang

.....

Date

.....

Name

.....

Signature

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it of sufficient grammatical clarity to be examined. To the best of my knowledge, the thesis is free of plagiarism and the research and writing are those of the candidate's except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

29/12/2020

Thomas Peyrin

.....

Date

.....

Name

.....

Signature

Authorship Attribution Statement

This thesis contains material from 3 papers published in the following peer-reviewed journals and accepted at conferences in which I am listed as an author and 1 paper under submission.

Chapter 2 is published as Wang, H., and Peyrin, T. (2019). Boomerang Switch in Multiple Rounds. Application to AES Variants and Deoxys. *IACR Transactions on Symmetric Cryptology*, 2019(1), 142-169. <https://doi.org/10.13154/tosc.v2019.i1.142-169>

The contributions of the co-authors are as follows:

- Prof. Thomas Peyrin provided helpful suggestions and revised the manuscript.
- I did the main work including finding the results and writing the manuscript.

Chapter 3 is currently under submission. The contributions of the co-authors are as follows:

- Prof. Jian Guo initiated the project and played an active role in this project.
- I did the main work including finding the results and writing the manuscript.
- Dr. Ling Song played an active role in this project.

Chapter 4 is published as Bao, Z., Ding, L., Guo, J., Wang, H., and Zhang, W. (2020). Improved Meet-in-the-Middle Preimage Attacks against AES Hashing Modes. *IACR Transactions on Symmetric Cryptology*, 2019(4), 318-347. <https://doi.org/10.13154/tosc.v2019.i4.318-347>

The contributions of the co-authors are as follows:

- Prof. Jian Guo initiated the project and provided helpful suggestions.
- I played an active role in finding the results and wrote the part of the manuscript related to 8-round attacks.

- Dr. Zhenzhen Bao and the other co-authors played active roles in this project.

Chapter 5 is published as Peyrin, T., and Wang, H. (2020). The MALICIOUS Framework: Embedding Backdoors into Tweakable Block Ciphers. *Advances in Cryptology, CRYPTO*, 2020, 249–278. http://dx.doi.org/10.1007/978-3-030-56877-1_9

The contributions of the co-authors are as follows:

- Prof. Thomas Peyrin initiated the project and provided helpful suggestions on finding the result. He also benchmarked the implementations and revised the manuscript.
- I did the main work including finding the results and writing the manuscript.

29/12/2020

Haoyang Wang

.....

Date

.....

Name

.....

Signature

Abstract

This thesis focuses on the cryptanalysis of symmetric-key cryptography, one of the most important branches of modern cryptography. The research spans two aspects of cryptanalysis.

Studying cryptanalysis provides better evaluation on the security of symmetric-key primitives. At first, we develop a new tool of boomerang attack and apply it to the analysis of the block cipher **AES** and the tweakable block cipher **Deoxys-BC**. We then introduce the technique of structure to key materials for the first time, with the help of which the currently best related-key attack on full-round **AES-256** can be improved. Lastly, we improve the meet-in-the-middle preimage attacks on **AES** hashing modes.

Aside from providing security evaluation, cryptanalysis can also be used intentionally to create weakness in a cipher, so-called backdoor. We propose the **MALICIOUS** framework, a new method to build tweakable block ciphers that have backdoors hidden, based on differential cryptanalysis. A concrete instantiation of this framework named **LowMC-M** is then proposed.

Acknowledgements

It was a great and unforgettable experience as a PhD student in the last four years. I am very glad to express my gratitude to all those who helped me during my PhD studies.

First of all, my deepest gratitude goes to my supervisor Prof. Thomas Peyrin for his inspired guidance and utmost support. His enthusiasm, dedication and rigorous attitude to cryptographic research left a deep impression on me. Whenever I was lost, I could always get insightful advice and encouragement from him. My research work would not be half as meaningful without his help. I would also like to thank him for organizing many recreational activities and supporting me to attend overseas conferences in the last four years.

I am also grateful to my co-supervisor Prof. Jian Guo. Without him, I would not have the chance to study in NTU. His creativity in research greatly inspired me. I am very thankful for his insightful advice and help in academic and other aspects.

I would like to express my thanks to Prof. Meiqin Wang for leading me to cryptography when I was a undergraduate student.

Many thanks to Prof. Khoong Ming Khoo and Prof. Hongjun Wu for being my TAC members.

It is my honor to have the chance to collaborate with many brilliant researchers: Zhenzhen Bao, Patrick Derbez, Lin Ding, Jian Guo, Tetsu Iwata, Thomas Peyrin, Ling Song, Ling Sun, Siwei Sun, Yosuke Todo, Meiqin Wang and Wenying Zhang.

I am so glad to have many nice colleagues sharing the lab SPMS-MAS-04-01: Alexandre Adomnicai, Adrien Benamira, Zhenzhen Bao, Subhadeep Banik, David Gerault, Mustafa Khairallah, Guozhen Liu, Zakaria Najm, Sumit Kumar Pandey, Siang Meng Sim, Ling Song, Quan Quan Tan and Vesselin Velichkov.

I would also like to thank my friends Nankun Hong, Zhe Li, Chen Li, Lunan Peng, Junjie Yang and Yi Tu for all the interesting things we did together: eating delicious food, playing basketball, playing badminton, playing video games, traveling, etc.

Finally, my special gratitude to my parents and my girlfriend for their unconditional support.

Contents

Statement of Originality	i
Supervisor Declaration Statement	iii
Authorship Attribution Statement	v
Abstract	vii
Acknowledgements	ix
List of Symbols	xiii
List of Abbreviations	xv
1 Introduction	1
1.1 Symmetric-Key Ciphers	1
1.1.1 Block Ciphers	1
1.1.2 Tweakable Block Ciphers	3
1.1.3 The Advanced Encryption Standard (AES)	4
1.1.4 AES-like Ciphers	5
1.2 Hash Functions	6
1.2.1 Hash Functions Based on Block Ciphers	7
1.2.2 Extendable Output Functions	8
1.3 Cryptanalysis of Symmetric-Key Primitives	8
1.3.1 Attack Models	9
1.3.2 Differential Cryptanalysis	10
1.3.3 Boomerang Attack	11
1.3.4 Meet-in-the-Middle Attack	12
1.4 Cryptographic Backdoors	13
1.5 Thesis Outline	13
2 Boomerang Switch in Multiple Rounds	15
2.1 Boomerang Switch	17
2.1.1 Boomerang Connectivity Table	17
2.1.2 Determining the Number of Rounds in E_m	18
2.1.3 Incompatibility in Multiple Rounds	20
2.1.4 Generalized Switching Effect in Multiple Rounds	20
2.1.5 Boomerang Switch in Two Rounds	23

2.2	Attack on 10-Round AES-256	26
2.2.1	The Boomerang Distinguisher	26
2.2.2	Key Recovery	28
2.3	Application to AES-192	30
2.3.1	Improvement of the Attack from Biryukov <i>et al.</i>	30
2.3.2	Evaluation and Improvement of the Attack from Biryukov <i>et al.</i>	31
2.4	Application to Deoxys-BC	33
2.4.1	Improved 10-Round Boomerang Attack	33
2.5	Conclusion	34
2.A	The Round Function of SKINNY	35
2.B	Differential Paths and Boomerang Distinguishers	35
3	Improved Related-Key Attack on the Full AES-256	45
3.1	Key Structures	47
3.1.1	Structure from Plaintext	48
3.1.2	Structure from Key	48
3.1.3	The Use of Key Structure	48
3.2	Improved Related-Key Attack on AES-256	49
3.2.1	Construction of the Key Structure	50
3.2.2	Boomerang Distinguisher	53
3.2.3	A Detailed Description of the Attack	53
3.3	Conclusion	59
4	Preimage Attacks on AES Hashing Modes	61
4.1	The MITM Preimage Attack	64
4.1.1	Application to Pseudo-Preimage Attacks	64
4.1.2	Conversion from Pseudo-Preimages to Preimages	67
4.1.3	Converting Block cipher to Compression Function	68
4.2	Techniques of MITM Attack on AES Hashing Modes	69
4.2.1	The 7-Round Attack by Sasaki and Its Improvement by Wu <i>et al.</i>	69
4.2.2	Introducing Neutral Bytes in Key	72
4.3	Reducing the Complexities of 7-Round Attacks	74
4.3.1	Improved Attack on 7-Round AES-128 Hashing Mode	74
4.3.2	Application to 7-Round AES-192 Hashing Mode	77
4.3.3	Application to 7-Round Kiasu-BC Hashing Mode	78
4.4	Extension to 8-Round Attacks	79
4.4.1	Techniques for Attacking 8 Rounds	80
4.4.2	The 8-Round Attack on AES-256	81
4.4.3	The 8-Round Attack on AES-192	83
4.4.4	The 8-Round Attack on Kiasu-BC	84
4.5	Discussion	85
4.6	Conclusion	87
5	Cryptographic Backdoor Design	89
5.1	Basics of Backdoor	91

5.1.1	Attacking Scenario	91
5.1.2	Security Notions and More	91
5.2	The MALICIOUS Framework	92
5.2.1	Block Ciphers with Partial Non-Linear Layers	92
5.2.2	The MALICIOUS Construction	93
5.2.3	The Backdoor Security	95
5.2.4	Rationale Underlying the MALICIOUS Construction	97
5.3	Instantiating the MALICIOUS Framework with LowMC	100
5.3.1	LowMC	100
5.3.2	Equivalent Representation of LowMC	100
5.3.3	LowMC-M	101
5.3.4	Embedding a Backdoor into LowMC-M	102
5.3.5	Parameters	105
5.4	Backdoor Security	106
5.4.1	Undetectability	106
5.4.2	Undiscoverability	109
5.4.3	Untraceability and Practicability	110
5.5	Cipher Security	110
5.5.1	Attacks Based on the Tweak	110
5.5.2	Attacks without Using the Tweak	111
5.6	Conclusion	112
6	Conclusion	115
	Bibliography	117

List of Symbols

$\oplus$	Exclusive-OR (XOR)
$\parallel$	Concatenation
Δ	Bitwise XOR-difference between a pair of binary strings
$\mathbb{F}_2$	The finite field of two elements
$\mathbb{F}_2^n$	The set of bitstrings of length n
M^T	Transpose of a matrix M
$\#E$	The number of elements in a set E
$\Pr[e]$	The probability of an event e
$O()$	Big O notation

List of Abbreviations

BCT	Boomerang Connectivity Table
BDT	Boomerang Difference Table
DDT	Difference Distribution Table
LSB	Least Significant Bit
MDS	Maximum Distance Separable
MILP	Mixed-Integer Linear Programming
MITM	Meet-in-the-Middle
NIST	National Institute of Standards and Technology
SPN	Substitution-Permutation Network
TBC	Tweakable Block Cipher
XOF	Extendable-Output Function
XOR	Exclusive-OR

Chapter 1

Introduction

Cryptography is the science of studying secure communication in the presence of adversaries. Its origin can be traced back to over 4000 years ago, one of the most famous practices in ancient times is the “Caesar cipher” which encrypts a text by a simple rotation of the alphabet. In modern times, cryptography has evolved a lot and became an almost new discipline, it lies at the intersection of mathematics and computer science. Over the past decades, the field of cryptography has grown to an extraordinary degree. Customarily, cryptography is classified into two main areas: *symmetric-key cryptography* and *public-key cryptography*, depending on how key material is used. In this thesis, we will restrict our attention to symmetric-key cryptography, more specifically, symmetric-key ciphers and hash functions.

1.1 Symmetric-Key Ciphers

Symmetric-key ciphers use the same key material for both *encryption* and *decryption*. If a *plaintext* is encrypted with a secret key, one can decrypt the *ciphertext* using the same key. There are two types of encryption algorithms in this category: *block ciphers* and *stream ciphers*. In this section, we introduce only block ciphers and its extension – *tweakable block ciphers* that are relevant to this thesis. A high level view of each of them is given in Figure 1.1 and 1.4 respectively.

1.1.1 Block Ciphers

A block cipher is a way of generating a family of permutations indexed by a key of k bits which operates on blocks of a fixed length n bits. The encryption can be described as $E(K, P) : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, which takes as input a key K and a plaintext P , and outputs the ciphertext. The value of k is usually the security claim of the cipher. More precisely, it represents the resistance of the cipher to *brute-force* attack, which simply enumerates all possible keys until the correct one is found.

The most common strategy for designing a block cipher is to iteratively update the *internal state*, initialized by the plaintext, by applying a simple

transformation, so-called *round function*, to the internal state multiple times. Meanwhile, a *key schedule* is applied to produce *subkeys* from the original secret key which are added to the internal state so that the round functions are dependent on the key. There are two main frameworks for designing a round function: Substitution-Permutation Network (SPN) and Feistel Network.

Figure 1.1: A block cipher.

Substitution-Permutation Network

An SPN round consists of a linear layer and a non-linear layer. The non-linear layer usually consists of several small permutations, called *S-boxes*, operating in parallel on a subset of the internal state. The linear layer consists of a linear operation to mix the outputs of the S-boxes, such as a matrix multiplication. The subkey is simply XORed to the internal state. Figure 1.2 illustrates the round function, where S denotes the S-box, L denotes the linear operation and K_i denotes the subkey.

Many block ciphers adopt this design structure. The most prominent one is the current NIST standard: Advanced Encryption Standard (AES) [45], a large part of this thesis is related to the security analysis of AES. The LowMC cipher [3] mentioned in Chapter 5 belongs to a subset of SPN where the non-linear layer is only applied to a part of the internal state.

Figure 1.2: An SPN round.

Feistel Network

In the case of a Feistel network, the round function uses a keyed function F_{K_i} and split the internal state into two branches L_i and R_i , it can be defined as:

$$\begin{cases} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus F_{K_i}(R_i), \end{cases}$$

a graphical illustration is given in Figure 1.3. Such round function is easy to invert, resulting that both encryption and decryption use the function F_{K_i} in the same forward direction, meaning that the function F_{K_i} does not have to be a permutation. The most famous block cipher using this structure is the previous NIST standard: Data Encryption Standard (DES) [97].

Figure 1.3: A Feistel round.

1.1.2 Tweakable Block Ciphers

The first formal concept of tweakable block cipher (TBC) was introduced by Liskov, Rivest and Wagner in [82]. Compared to a block cipher, a tweakable block cipher accepts an additional t -bit public input called *tweak*, its signature is $\tilde{E}(K, T, P) : \{0, 1\}^k \times \{0, 1\}^t \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, denoted by T the tweak. The introduction of a tweak input provides the ability for the user to select a permutation among a family of permutations even when the key is fixed.

Figure 1.4: A tweakable block cipher.

Due to this extra degree of freedom that can potentially be leveraged by the attacker, designing a TBC is not straightforward. Block cipher-based TBC constructions have been studied [78, 86, 103], but come with a non-negligible efficiency penalty. We can mention the TWEAKEY framework, a recent design strategy to build ad-hoc TBCs, that was proposed by Jean *et al.* [70]. In this framework, the key and tweak inputs are treated equivalently in terms of design and this material is called *tweakey*: the tweakey input can be used as key or tweak value, which is up to the choice of the user.

Unlike the key input, the tweak does not need to be kept secret and therefore one could assume that an adversary has full control over it.

1.1.3 The Advanced Encryption Standard (AES)

AES [45] is an iterated block cipher which encrypts 128-bit plaintext with secret key of sizes 128, 192, and 256 bits. AES with 128-bit (192, 256) master keys is denoted by AES-128 (192, 256). AES-128, AES-192, and AES-256 share the same round function with different number of rounds: 10, 12, and 14, respectively. The rounds are numbered $0, \dots, N_r - 1$, where $N_r \in \{10, 12, 14\}$ is the number of rounds. Its internal state can be represented as a 4×4 matrix whose elements are byte value (8 bits) in a finite field of $\text{GF}(2^8)$. The round function consists of four basic transformations in the following order:

- **SubBytes** (SB) is a nonlinear substitution that applies the same S-box to each byte of the internal state.
- **ShiftRows** (SR) is a cyclic rotation of i -th row by i bytes to the left, for $i = 0, 1, 2, 3$.
- **MixColumns** (MC) is a multiplication of each column with a Maximum Distance Separable (MDS) matrix over $\text{GF}(2^8)$.
- **AddRoundKey** (AK) is an exclusive-or with the round-dependent key.

MDS guarantees that the sum of active bytes (*i.e.*, bytes for which the difference is non-zero) in the input and output of the **MixColumns** operation is at least 5 unless all bytes are non-active (*i.e.*, bytes containing no difference). The matrices for the encryption and decryption are shown below. Note that $X[j]$ is the input value and $Y[j]$ is the updated value, and values are represented in hexadecimal notation.

$$\begin{pmatrix} Y[0] \\ Y[1] \\ Y[2] \\ Y[3] \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{pmatrix}, \quad \begin{pmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{pmatrix} = \begin{pmatrix} \text{e} & \text{b} & \text{d} & 9 \\ 9 & \text{e} & \text{b} & \text{d} \\ \text{d} & 9 & \text{e} & \text{b} \\ \text{b} & \text{d} & 9 & \text{e} \end{pmatrix} \begin{pmatrix} Y[0] \\ Y[1] \\ Y[2] \\ Y[3] \end{pmatrix}$$

One round of AES is depicted in Figure 1.5. The byte ordering of a state s can be represented by (i, j) , where i and j stand for the row and column indexes of the byte, respectively. Besides, we use s^r to denote the state at r -th round. Accordingly, $s_{i,j}^r$ denotes the byte of the r -th round state s^r located at i -th row and j -th column. Specially, we let x (resp. y) represent the state before (resp. after) **SubBytes**.

Figure 1.5: The AES round function.

At the very beginning of the encryption, an additional whitening key addition is performed, and the last round does not contain **MixColumns**.

The key schedule of AES transforms the master key into $N_r + 1$ subkeys of 128 bits each. The master key is divided into N_k 32-bit words ($W[0], W[1], \dots, W[N_k - 1]$), then $W[i]$ for $i = N_k, \dots, 4 \cdot N_r + 3$ is computed as

$$W[i] = \begin{cases} W[i - N_k] \oplus \text{SB}(\text{RotByte}(W[i - 1])) \oplus \text{Rcon}[i/N_k], & i \equiv 0 \bmod N_k, \\ W[i - 8] \oplus \text{SB}(W[i - 1]), & N_k = 8 \text{ and } i \equiv 4 \bmod 8, \\ W[i - N_k] \oplus W[i - 1], & \text{otherwise.} \end{cases}$$

Figure 1.6: Key schedule round functions for AES-128 (left), AES-192 (center), and AES-256 (right) [68].

RotByte is a cyclic shift by one byte to the left, and **Rcon** is the round constant, for which we refer to [45] for details. The i -th *round key*, denoted by RK^i , is the concatenation of 4 words $W[4i] \parallel W[4i + 1] \parallel W[4i + 2] \parallel W[4i + 3]$. We denote the i -th 128 (192, 256) -bit subkey of AES-128 (192, 256) by K^i , with K^0 being the master key. The graphic representation of the key schedule functions is depicted in Figure 1.6.

Differential properties of the AES S-box. The details of the AES S-box and its difference distribution table (DDT)¹ could be found in [45]. For any input difference $\Delta_{in} \neq 0$, there exists exactly one Δ_{out} such that $\text{DDT}(\Delta_{in}, \Delta_{out}) = 4$ (this results in the highest probability 2^{-6} for the AES S-box transition), 126 values of Δ_{out} such that $\text{DDT}(\Delta_{in}, \Delta_{out}) = 2$ (*i.e.*, probability 2^{-7}), and the rest 129 values of Δ_{out} with $\text{DDT}(\Delta_{in}, \Delta_{out}) = 0$. Those $(\Delta_{in}, \Delta_{out})$'s with $\text{DDT}(\Delta_{in}, \Delta_{out}) \neq 0$ are called *compatible*, and others are *incompatible*. These statistics will be used in our attack later.

1.1.4 AES-like Ciphers

Kiasu-BC

Kiasu-BC is the underlying TBC used in the authenticated encryption scheme Kiasu proposed by Jean *et al.* [69]. The TBC is almost identical to AES-128

¹Refer to Definition 1.

except for the additional input tweak, which renders it an attractive primitive for various modes of operation and applications requiring tweakable block ciphers. Therefore, studying how the additional tweak input affects the security strength compared to that of AES is highly valuable to gain trust for more adoptions.

Figure 1.7: The TBC Kiasu-BC.

The TBC Kiasu-BC takes three inputs: a 64-bit tweak T , a 128-bit key K and a 128-bit plaintext P , and it outputs a 128-bit ciphertext. As depicted in Figure 1.7, Kiasu-BC is exactly the AES-128 cipher, but with a 64-bit tweak value XORed to the first two rows of the internal state after each round key addition in the round function of the encryption, including after the pre-whitening key addition. There is no tweak schedule, *i.e.*, the same T is used every time in its original form. Kiasu-BC can actually be viewed as one of the simplest instances of the TWEAKEY framework based on AES.

Deoxys-BC

Deoxys [71] is an authenticated encryption scheme and was selected as one of the CAESAR winner of the security-in-depth portfolio [1]. Its internal primitive Deoxys-BC is a 128-bit AES-based TBC following the TWEAKEY framework [70]. It has a linear tweakey schedule and the same round function as AES. Deoxys-BC has two versions: Deoxys-BC-256 and Deoxys-BC-384. Both versions take three inputs: a plaintext, a key and a tweak. The concatenation of key and tweak is named as tweakey, whose size is 256 bits for Deoxys-BC-256 and 384 bits for Deoxys-BC-384.

1.2 Hash Functions

A cryptographic hash function H is a keyless algorithm that maps a string of arbitrary length to a string of a fixed length n . The output is called *hash value* or *digest*. A “good” cryptographic hash function must satisfy the following properties:

- *Collision resistance*: it is computationally difficult to find two different messages M_1 and M_2 such that $H(M_1) = H(M_2)$ in less than $2^{\frac{n}{2}}$ operations.
- *Preimage resistance*: for a given hash value h , it is computationally difficult to find a message M such that $H(M) = h$ in less than 2^n operations.

- *Second preimage resistance*: for a given message M_1 , it is computationally difficult to find another message $M_2 \neq M_1$ such that $H(M_1) = H(M_2)$ in less than 2^n operations.

These requirements are derived from practical applications, such as digital signature, message authentication codes, key derivation, etc.

Most hash functions use an iterative method, such as the Merkle-Damgård construction [47, 85] and the sponge construction [23], to process arbitrary-length messages by applying a fixed-length input/output function called *compression function* iteratively. This method is adopted by many famous hash functions, such as SHA-1 [89], SHA-2 [90] and SHA-3 [91].

1.2.1 Hash Functions Based on Block Ciphers

Hashing modes are ways to convert block ciphers to compression functions, and then to hash functions under some domain extensions. In [96], Preneel, Govaerts, and Vandewalle summarized twelve secure ways to convert a block cipher to a compression function, and these constructions are referred to as PGV modes nowadays after the name of the authors. These twelve PGV constructions can be expressed by the formulas given in Table 1.1, where H_{i-1} and H_i denote the chaining states before and after the update, M_i the message block, and X_i the XOR of H_{i-1} and M_i . These twelve PGV modes can be classified according to the material fed in through the key into three classes: Class 1 – chaining values are fed in through the key (1st row in Table 1.1); Class 2 – messages are fed in through the key (2nd row in Table 1.1); Class 3 – XOR sum of messages and the chaining values are fed in through the key (3rd row in Table 1.1). Out of them, there are three modes named Davies-Meyer (DM), Matyas-Meyer-Oseas (MMO), Miyaguchi-Preneel (MP) commonly used in practice (see Figure 1.8). For instance, all the hash functions in the MD-SHA hash family, including MD4 [102], MD5 [101], SHA-1 [89], and SHA-2 [90], can be viewed as DM modes.

Table 1.1: Twelve secure PGV constructions [96, 104].

	No.	Computation	No.	Computation	No.	Computation	No.	Computation
Class 1	1	$E(H_{i-1}, M_i) \oplus M_i$	2	$E(H_{i-1}, X_i) \oplus X_i$	3	$E(H_{i-1}, M_i) \oplus X_i$	4	$E(H_{i-1}, X_i) \oplus M_i$
Class 2	5	$E(M_i, H_{i-1}) \oplus H_{i-1}$	6	$E(M_i, X_i) \oplus X_i$	7	$E(M_i, H_{i-1}) \oplus X_i$	8	$E(M_i, X_i) \oplus H_{i-1}$
Class 3	9	$E(X_i, M_i) \oplus M_i$	10	$E(X_i, H_{i-1}) \oplus H_{i-1}$	11	$E(X_i, M_i) \oplus H_{i-1}$	12	$E(X_i, H_{i-1}) \oplus M_i$

X_i represents $H_{i-1} \oplus M_i$.

Compared with designs from scratch, hashing modes enjoy both inherited security and performance efficiencies from the underlying block ciphers. Security proofs of hashing modes, which deduce the security of the hash function (for security notions such as collision resistance) to the security of the underlying block cipher, ensure that no attack against the hash function could be possible before an attack against the underlying block cipher is found. This removes the hassle of intensive cryptanalysis required by hash

Figure 1.8: Illustrations for DM, MMO, and MP modes [68, 104].

functions designed from scratch, when the hashing mode is instantiated by a secure block cipher such as AES. For environments like resource-constrained hardware where a block cipher is probably already implemented, a hashing mode could be the most economical way to achieve a hash function for purposes like hash digest and signature, since in most of the cases implementing a hashing mode on top of an existing block cipher costs much lesser than that of a standalone hash function.

1.2.2 Extendable Output Functions

An extendable-output function (XOF) is a generalization of a hash function, where the output can be extended to any desired length. Similar to a hash function, it should be collision, preimage and second-preimage resistant. A XOF is a natural choice when an application requires a hash function to have non-standard digest length. Technically, it is also possible to use a XOF as a generic hash function by fixing the output length. Besides, it has some other applications, such as key derivation functions and stream ciphers.

Currently, there are many instances of XOF, such as SHAKE128 and SHAKE256 (defined in SHA-3 standard [91]) and the more efficient variant KangarooTwelve [22].

1.3 Cryptanalysis of Symmetric-Key Primitives

When designing a cipher, the designer normally follows the *Kerckhoffs' principle* which assumes that the adversary knows every detail of the cipher algorithm except the secret key. In this situation, a cipher should be secure as long as the secret key is unknown. In general, there are two main forms of attack against a block cipher.

- *Key recovery*: the adversary tries to recover the secret key so that he can easily decrypt all ciphertexts. Since a cipher is usually claimed to have k -bit security (key size), only those methods that can retrieve the secret key faster than brute-force are considered as a key recovery attacks.

- *Distinguishing attack*: given two black boxes, one containing the block cipher with an unknown key and the other containing a permutation chosen uniformly at random from the set of all permutations, the adversary tries to determine the identities of the black boxes by simply interacting with them via encryption and/or decryption queries. The algorithm used in the attack faster than brute-force is called *distinguisher*.

Although the distinguishing attack does not recover the secret key, it implies that the targeted block cipher is not perfect and presents some non-random behaviors. Moreover, it is usually possible to transform a distinguisher to a key recovery attack by adding some extra rounds to the distinguisher. This method will be extensively used in this thesis.

1.3.1 Attack Models

Beside knowing the cipher mechanism, an adversary may have different kinds of access to the encryption system. An attack is generally considered under *black-box context* when the adversary is unable to access the internal state of the cipher but can only ask for encryptions/decryptions queries, while the counterpart *white-box context* permits the adversary to have complete control of the cipher execution and internals. This thesis only covers the common black-box context. We can further divide the black-box context into several scenarios depending on the way of accessing input/output of the cipher.

- *Ciphertext-only attack*: the adversary has only access to the ciphertexts.
- *Known-plaintext attack*: the adversary has access to a limited number of pairs of plaintexts and ciphertexts.
- *Chosen-plaintext attack*: the adversary is allowed to choose a set of plaintexts determined in advance and obtain their corresponding ciphertexts.
 - *Adaptive chosen-plaintext attack*: in addition, the adversary is able to analyze the previous results before choosing the next plaintext to be queried.
- *Chosen-ciphertext attack*: the adversary is allowed to choose a set of ciphertexts determined in advance and obtain their corresponding plaintexts to be queried.
 - *Adaptive chosen-ciphertext attack*: in addition, the adversary is able to analyze the previous results before choosing the next ciphertext.

On top of the above models, there is a distinction that can be made with regards to key material.

- *Single-key attack*: the adversary is only allowed to make queries to the cipher with the original secret key K .
- *Related-key attack* [24]: the adversary is not only allowed to make queries to the cipher with the original secret key K , but also with a key K' related to the original one. Most of the time, the relation is simply the application of an XOR difference Δ to the original key K : $K' = K \oplus \Delta$.

Besides, the additional input tweak of a tweakable block cipher can also be used in an attack. Similar to a related-key attack, a *related-tweak attack* allows the adversary to choose a relation between tweaks. In particular, since the tweak value is publicly known, the adversary is actually able to choose any desired tweak value in an attack, this model is called *chosen-tweak attack*. Moreover, these notions related to key and tweak materials can be combined, thereby we can have related-key related-tweak attack and related-key chosen-tweak attack.

1.3.2 Differential Cryptanalysis

Differential cryptanalysis is a chosen-plaintext attack and was first introduced by Biham *et al.* [29]. It is a statistical cryptanalysis that exploits the probability of a *differential*, *i.e.*, a pair of XOR differences (α, β) such that for a given input difference α applied on the plaintext, the output difference on the ciphertext after multiple round functions is equal to β . For a random n -bit permutation, the expected probability of a differential is 2^{-n} . If this probability is higher than 2^{-n} for an encryption algorithm, then the differential can be used in an attack.

The differential attack usually starts by searching a *differential characteristic*, which is a sequence of differentials of a single round (the output difference of one round is the input difference of the subsequent round). The probability of a differential characteristic is then the product of the probabilities of the one-round differentials in the sequence, which assumes that the cipher rounds are independent from each other. In practice, this assumption is not true as the subkeys added to each round are related due to the key schedule. However, this computation is still a good estimation in reality.

In most cases, a differential (α, β) could contain several differential characteristics with the same initial and final differences while the differences in intermediate states might differ. All of these differential characteristics contribute to the overall probability of the differential. Usually, the adversary only needs to find the dominant differential characteristic with the highest probability.

One of most important tools in differential cryptanalysis is the *difference distribution table* (DDT), which is used to evaluate the differential transition of S-box for those ciphers based on predefined S-box. Its definition is as follows:

Definition 1. For an n -bit S -box S , its difference distribution table (DDT) is a $2^n \times 2^n$ table, in which the entry for any pair $(\Delta_{in}, \Delta_{out})$ is given by

$$DDT(\Delta_{in}, \Delta_{out}) = \#\{x \in \{0, 1\}^n | S(x) \oplus S(x \oplus \Delta_{in}) = \Delta_{out}\}.$$

With the help of DDT, the probability of the differential transition from the input difference Δ_{in} to the output difference Δ_{out} can be easily computed, being $DDT(\Delta_{in}, \Delta_{out})/2^n$.

For a key recovery attack against an r -round block cipher, a good differential on $r - 1$ rounds can be used to recover the subkey of the r -th round. First, the adversary queries the encryption oracle with pairs of plaintexts satisfying the input difference of the differential. Then, once the ciphertext pairs are received, the adversary can check whether the pairs satisfy the differential by making a guess on a subpart of the r -th round subkey and then computing back the ciphertext difference through the last round. Since the diffusion in a single round of the cipher will not be perfect, we have enough information to filter the wrong guesses. Finally, the correct subkey should be the guess that satisfies the most often the differential output expected at the $(r - 1)$ -th round.

1.3.3 Boomerang Attack

The *boomerang attack* [108] is an extension of the differential attack, where two differentials are combined in an elegant way to provide a distinguishing property of the cipher. More precisely, it regards the targeted cipher E as a cascade of two sub-ciphers, *i.e.*, $E = E_1 \circ E_0$ as depicted in Figure 1.9. Assume that there is a differential $\alpha \rightarrow \beta$ with probability p for E_0 and a differential $\gamma \rightarrow \delta$ with probability q for E_1 , the boomerang attack exploits the probability of the following event:

$$Pr[E^{-1}(E(x) \oplus \delta) \oplus E^{-1}(E(x \oplus \alpha) \oplus \delta) = \alpha] = p^2 q^2, \quad (1.1)$$

since both the two differentials are used twice for the both sides of the boomerang.

Figure 1.9: The boomerang attack.

The basic boomerang attack requires an adaptive chosen-plaintext/ciphertext scenario as the attacker needs to ensure the α difference on the encryption queries and the δ difference on the decryption queries. Later, the *amplified boomerang attack* [73] was proposed, which only requires a chosen-plaintext scenario and where a right quartet is obtained with probability $2^{-n}p^2q^2$. Furthermore, it was pointed out in [26, 27] that any value of β and γ is allowed as long as $\beta \neq \gamma$. As a result, the probability of the right quartet is increased to $2^{-n}\hat{p}^2\hat{q}^2$, where $\hat{p} = \sqrt{\sum_i \Pr^2(\alpha \rightarrow \beta_i)}$ and $\hat{q} = \sqrt{\sum_j \Pr^2(\gamma_j \rightarrow \delta)}$, which is known as the *rectangle attack*.

Related-Key Boomerang Attack. Boomerang attacks under the related-key setting were formulated in [28]. Let ΔK and ∇K be the key differences for sub-ciphers E_0 and E_1 , respectively. The attacker needs to access four related-key oracles with $K_1 \in \mathbb{K}$, where $\mathbb{K}$ denotes the key space: $K_2 = K_1 \oplus \Delta K$, $K_3 = K_1 \oplus \nabla K$ and $K_4 = K_1 \oplus \Delta K \oplus \nabla K$. The attack is then performed with the following process:

1. Choose a plaintext P_1 at random, compute another plaintext $P_2 = P_1 \oplus \alpha$.
2. Ask for the encryption of P_1 and P_2 with secret key K_1 and K_2 separately, denote the ciphertexts by C_1 and C_2 respectively.
3. Compute $C_3 = C_1 \oplus \delta$ and $C_4 = C_2 \oplus \delta$.
4. Ask for the decryption of C_3 and C_4 with K_3 and K_4 , and denote the new plaintexts by P_3 and P_4 respectively.
5. Check whether $P_3 \oplus P_4 = \alpha$.

Boomerang Switching Effect. The probability computed from Formula 1.1 is under the assumption that the two sub-ciphers E_0 and E_1 are independent. However, in practice, E_0 and E_1 are probably dependent at the switching position in one or more rounds, which would have positive or negative effect on the boomerang attack. Chapter 2 will discuss the switching effect in detail.

1.3.4 Meet-in-the-Middle Attack

In its early stages of development, the meet-in-the-middle approach proposed by Diffie and Hellman in [50] was mainly used as a generic time-memory trade-off technique to attack encryption schemes with clear separations, *e.g.*, Double DES [84]. That is because it is straightforward to divide the whole computation into two or multiple independent computational chunks. Thus, the whole “for” loop in a brute-force attack can be separated into two or multiple independent “for” loops, which have a quite smaller and mutually balanced amount of computations, and which independently generate lists of

candidates (of partial solutions). The independence between the smaller “for” loops makes each element in one list able to make a pair with any element in other lists to form a candidate solution. Such an effect of taking cartesian product between two sets enlarges the number of candidates for the correct computation dramatically. Then, according to the birthday paradox, say for two lists of 2^ℓ entries of n -bit values, to find a match with high probability, it is required that $2^{(\ell+\ell)} \geq 2^n$, i.e., $\ell \geq n/2$. Thus, the minimum time complexity of a simple MITM attack is $2^{n/2}$, together with $2^{n/2}$ memory.

1.4 Cryptographic Backdoors

A backdoor in an encryption algorithm enables an entity who knows it to circumvent the security guarantees so that he can obtain the secret information more efficiently than with a generic black-box attack. There are two categories of backdoors. The first one is the backdoor implemented in a security product at the protocol or key-management level, which is generally considered in practice. The second one is *cryptographic backdoor*. A cryptographic backdoor is embedded directly during the design phase of a cryptographic primitive and renders the cipher susceptible to some dedicated cryptanalysis.

Cryptographic backdoors have been extensively studied by Young and Yung, introducing the term “Kleptography” [116, 119]. However, despite some interest from the academic community about this topic, there are very few publicly known backdoored primitives. A concrete example is the pseudorandom number generator `Dual_EC_DRBG` [17] designed by NSA, whose backdoor was revealed by Edward Snowden in 2013 and also in some research works [21, 107]. Some public block ciphers might contain backdoors not claimed by the designers. In particular, primitives whose detailed design rationale is not provided are naturally more suspicious, especially when the ciphers have been designed by governmental agencies (as can be seen by the difficulties encountered by the NSA lightweight block ciphers `SIMON` and `SPECK` [18] to become ISO standards).

1.5 Thesis Outline

This thesis presents some new and improved cryptanalytic techniques and their applications to `AES`, `Deoxys-BC` and `Kiasu-BC`, as well as a new backdoor design framework together with one instantiation based on `LowMC`. The outline of the rest of this thesis is as follows:

Chapter 2. We revisit the issue of boomerang switching effect, and exploit it in the case where multiple rounds are involved. To support our analysis, we propose a tool called Boomerang Difference Table (BDT), which allows a systematic evaluation of the boomerang switch through multiple rounds. In order to illustrate the power of this technique, we propose a new related-key

attack on 10-round AES-256, and also provide improved attacks against full AES-192 and reduced-round Deoxys-BC. This work was published at [109].

Chapter 3. We explore the possibility of introducing structure to key material in the related-key setting, while a similar idea has long been applied to plaintexts/ciphertexts. We use this simple idea to improve the related-key boomerang attack against AES-256 by Biryukov and Khovratovich in 2009 [32]. It turns out to be effective, *i.e.*, both data and time complexities are reduced, at the cost of an increased amount of required related keys. We also propose some trade-offs between the data/time complexity and the number of keys. This work is under submission.

Chapter 4. We introduce neutral bits [10] from key to gain extra degrees of freedom for preimage attack against AES-based hashing modes. As an immediate result, the current best 7-round pseudo-preimage attacks against AES-128, AES-192 and AES-256 are improved. Besides, by carefully choosing the neutral bits from the key to cancel those from the internal state, we propose the first preimage attacks against 8-round AES and Kiasu-BC hashing modes. This work was published at [14].

Chapter 5. We propose a general framework for embedding backdoors into tweakable block ciphers. We explain how in our construction any entity knowing the backdoor can practically recover the secret key of a user and we also argue why even knowing the presence of the backdoor and the workings of the cipher will not permit to retrieve the backdoor for an external user. We analyze the security of our scheme in the classical black-box context and we show that retrieving the backdoor is very difficult. An instantiation based on the cipher LowMC is proposed to support our framework. This work was published at [95].

Chapter 6. We conclude this thesis and discuss some possible directions for future work.

Chapter 2

Boomerang Switch in Multiple Rounds

In boomerang-style attacks, the probability of obtaining a right quartet is computed based on the assumption that the two sub-ciphers E_0 and E_1 are independent. However, Murphy [88] pointed out that the independence assumption is not always legitimate. He provided some counterexamples where two independently chosen differential characteristics may be *incompatible*, which causes a zero probability of the boomerang distinguisher. Interestingly, he also showed that some other cases lead in contrary to a much higher probability. Besides, many improvements taking advantages of the dependency between the two differential characteristics have been proposed, such as the *middle round S-box trick* [30], *ladder switch*, *S-box switch* and *Feistel switch* [32].

Figure 2.1: The sandwich attack.

Those observations can be captured in the framework of *sandwich attack* [55], which decomposes the cipher as $E = E_1 \circ E_m \circ E_0$, with the

middle part E_m consisting of relatively short transformations (as depicted in Figure 2.1). The differential characteristics of E_0 and E_1 specify the input difference β and output difference γ to E_m separately. Then, a small boomerang distinguisher is applied on E_m with probability r computed by:

$$r = \Pr[E_m^{-1}(E_m(x) \oplus \gamma) \oplus E_m^{-1}(E_m(x \oplus \beta) \oplus \gamma) = \beta] \quad (2.1)$$

The entire boomerang distinguisher succeeds only if a right quartet of E_m is generated, thus the overall success probability becomes p^2q^2r . Then, the boomerang switching effects can be unified as the dependency between the two differential characteristics lie now in E_m . We call the interaction between the two differential characteristics *boomerang switch*, just as the name chosen in [32].

At Eurocrypt 2018, Cid *et al.* [42] proposed an efficient and systematic method to evaluate r . It narrows the switching effect of a state to a single S-box with a newly created table, called *Boomerang Connectivity Table (BCT)*. The entries in the BCT record the probability of generating a right quartet for each input/output difference at the S-box level, such that the boomerang incompatibility, ladder switch and S-box switch can easily be detected. Moreover, the BCT provides a potentially stronger switching effect than those in the previous observations. Later, a follow-up work on the uniformity of the BCT was produced in [38].

Note that all the switching effects and the BCT tool are derived under the scenario where E_m is a simple operation, such as one S-box layer. However, practical experiments from [41] show that additional improvements might be obtained when E_m contains two rounds, and the analysis of **SKINNY** cipher [19] in [42] also highlights that boomerang switching effect in 2-round E_m is possible. Then, from the research results summarized above, the following questions naturally arise:

- What is the maximum number of rounds that can compose E_m , such that the boomerang switching effect still exists?
- Can we directly apply the current switching techniques to E_m when it is composed of multiple rounds? If not, are there other switching techniques that can provide a systematic evaluation?

Contributions of this Chapter. In this chapter, we provide a systematic analysis of the boomerang switching effect in multiple rounds for S-box based ciphers. First, we exploit the principle behind the ladder switch and show that the boomerang switch is applicable to multiple rounds, *e.g.*, a 4-round boomerang distinguisher of **SKINNY** can be constructed with probability 1. Then, the applicability of the BCT in multiple rounds is shown to be defective with an incompatibility example on two rounds of **AES**. In order to capture the switching effect in multiple rounds, we propose a new tool, so-called *Boomerang Difference Table (BDT)*. The BDT is created at the S-box level and is a combination of the BCT and the DDT (Difference Distribution

Table). It represents the differential propagation and the boomerang switch in a unified manner. With the help of the BDT, we present a general evaluation of the 2-round boomerang switch for the first time.

To illustrate the power of the boomerang switch in multiple rounds, we exhibit a full key recovery related-key attack on 10-round AES-256 with only 2^{75} computations and 2 related-keys. Previously, the best attack [31] on this reduced-round variant of AES-256 required 2^{44} computations, but only recovered 35 subkey bits (the rest of the key having to be brute-forced). Moreover, it has to be conducted under a much less realistic related-subkey scenario, where the attacker is allowed to choose differences on subkeys and not directly on the main secret key. Known attacks [28, 74] on 10-round AES-256 in the classical related-key scenario require much more related keys (64 or 256) and computations effort (2^{172}).

Besides, we further showcase the practical usage of the BDT by extending from one round to two rounds the boomerang switch in the attacks against full AES-192 [32, 34] (the attack from [34] remained the best publicly known attack on full-round AES-192).

Finally, we describe how to improve the boomerang attack [41] against 10-round Deoxys-BC [71] by accurately detecting from which parts of the attack the differential probabilities are coming from. This work was published at [109].

Organization. In Section 2.1, we discuss the boomerang switch in the case of multiple rounds, and introduce the boomerang difference table (BDT) as a method to evaluate the switching effect and its applications on 2-round boomerang switch. In Section 2.2, the boomerang attack on 10-round AES-256 is proposed by applying the 2-round boomerang switch. The applications on AES-192 and Deoxys-BC with the 2-round boomerang switch are presented in Sections 2.3 and 2.4. Section 2.5 summarizes this chapter.

2.1 Boomerang Switch

2.1.1 Boomerang Connectivity Table

The boomerang switch, proposed in [32], was used to obtain free rounds in the middle of the cipher in the attacks against full AES-192 and AES-256. The idea was to optimize the transition between the sub-paths of E_0 and E_1 in order to minimize the overall complexity of the distinguisher. In [32], two S-box based switches were introduced: the ladder switch, and the S-box switch. The idea of the ladder switch is to realize that instead of necessarily decomposing the cipher into rounds, one can decompose it into smaller parallel transformations and this may lead to better distinguishers. The idea of the S-box switch is that when a same S-box is activated (*i.e.*, differentially active) in both E_0 and E_1 , and when the output difference in E_0 is identical to the input difference in E_1 , then the differential transition through the

S-box is free in one of the two directions.

These switches were further generalized with the *boomerang connectivity table* [42] and we provide here the definition.

Definition 2. [42] Let S be an invertible function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$, and $\Delta_0, \nabla_0 \in \mathbb{F}_2^n$. The boomerang connectivity table (BCT) of S is defined by a $2^n \times 2^n$ table, in which the entry for (Δ_0, ∇_0) is given by:

$$BCT(\Delta_0, \nabla_0) = \#\{x \in \{0, 1\}^n \mid S^{-1}(S(x) \oplus \nabla_0) \oplus S^{-1}(S(x \oplus \Delta_0) \oplus \nabla_0) = \Delta_0\}.$$

The generation of the BCT can be visualized in Figure 2.2. The ladder switch is captured by the BCT in the case where at least one of the index equals to zero. The S-box switch is captured by the BCT in the case where ∇_0 equals Δ_1 . Moreover, the incompatibility pointed out by Murphy [88] simply corresponds to zero entries in the BCT.

Figure 2.2: Generation of a right quartet at the S-box level.

2.1.2 Determining the Number of Rounds in E_m

Throughout this chapter, we will analyze the boomerang switching effect in E_m . The notations in Figure 2.1 and 2.2 are adopted. We will use Δ and ∇ to denote the differences for sub-cipher E_0 and E_1 , respectively. Note that the linear layers at the head and at the tail of the round operations in E_m are not considered, because linear layers make no difference on the switching effect.

At FSE 2018 [41], Cid *et al.* proposed a Mixed-Integer Linear Programming (MILP) based method to exploit the switching effect in two rounds of Deoxys-BC [71]. Their method adopts the idea of ladder switch where the round function can be divided into two independent parts. Suppose one part is only active in E_0 and the other part is only active in E_1 , then by assigning the former as a part of E_1 and the latter as a part of E_0 , the differential probability of all the S-boxes becomes 1. Since the round function of Deoxys-BC can operate on two independent parts for two rounds, Cid *et al.* set the length of E_m to 2 rounds in their search tool.

Recalling the procedure of boomerang attack, the differential characteristic of E_0 is used in the forward direction for the pair (P_1, P_2) and the backward direction for the pair (P_3, P_4) , and the differential characteristic of E_1 is used

in the backward direction for both pairs (C_1, C_3) and (C_2, C_4) . The principle behind the ladder switch in E_m is that the backward diffusion of the active cells in γ has no interaction with the forward diffusion of the active cells in β , thus the difference between y_3 and y_4 can backtrack to β with probability 1. We summarize it as follows:

Property 1. *In E_m , if the forward diffusion of the active cells in β has no interaction with the backward diffusion of the active cells in γ , a right quartet of E_m can be generated with probability 1.*

The effect of Property 1 is not exactly the same as the ladder switch and we use the following example to highlight the difference.

Example 1. *Let SKINNY [19] be the block cipher considered¹. By applying Property 1, we find that a right quartet of E_m can be generated with probability 1 up to four rounds with nonzero β and γ . One example is depicted in Figure 2.3, which is composed of two truncated differential characteristics. With any instantiated values of β and γ , the boomerang can always return to β with probability 1, i.e., $r = 1$. However, by the original definition of ladder switch the round function of SKINNY cannot be divided into two independent parts for more than two rounds as the diffusion of the two parts will interact with each other.*

Figure 2.3: An example of 4-round E_m for SKINNY, the two differential characteristics are given in truncated notation (gray denotes an active nibble, white denotes an inactive nibble), the above one is used for encryption and the below one is used for decryption.

Thus, using Property 1, the maximal number of rounds for E_m can be easily determined for a specific cipher. Briefly, start from a 1-round E_m , iterate the truncated difference of β and γ to meet the condition in Property 1, if it is satisfied, increase the number of rounds in E_m by 1 and repeat the previous process until the condition cannot be met, then the maximal number of rounds for E_m is the one of the last E_m that meets the condition. In particular, E_m can contain more rounds for the ciphers with a sparser diffusion layer. Besides, we note that the above E_m is actually a 4-round distinguisher of SKINNY with probability 1, which is an interesting observation by it self.

¹Its round function is given in Appendix 2.A

2.1.3 Incompatibility in Multiple Rounds

As stated in [42], the BCT provides a unified representation of the existing observations on the boomerang switch. Nevertheless, it was proposed to evaluate E_m when composed of a single S-box layer. The applicability of the BCT to the boomerang switch in multiple rounds remained unknown so far.

We give an example in Figure 2.4 and show that the BCT is actually incapable of detecting incompatibilities when E_m consists of two rounds.

Figure 2.4: A 2-round E_m example for AES, with the actual β and γ values, which will never generate a right quartet. The differences are represented in hexadecimal notation.

In Figure 2.4, both differential characteristics are established with probability 2^{-7} . In the first S-box layer, the upper differential characteristic fixes the Δ_0 of the active S-box to $0xdf$ and the lower differential characteristic fixes $\nabla_0 = 0xa9$, for which the BCT entry is 2. At the second S-box layer, for the targeted S-box, the BCT entry is 2 for $\Delta_0 = 0xf9$ and $\nabla_0 = 0xc6$. All the other S-boxes in the two rounds will be bypassed with probability 1 (entries of the BCT are 2^8). Thus, according to the BCT, the difference pairs (Δ_0, ∇_0) of all S-boxes in the two S-box layers are compatible, which implies that the two differential characteristics of E_m are compatible in a boomerang distinguisher. Let us focus on the S-box at position $(0, 0)$ in the first S-box layer, a right quartet requires that the DDT entry for $(0xdf, 0xf1)$ and the BCT entry for $(0xdf, 0xa9)$ are nonzero simultaneously. However, a brute-force search of $\mathbb{F}_2^n$ shows that there is no value satisfying this condition. Hence, a right quartet will never be generated for the E_m , which contradicts the conclusion derived by the BCT.

2.1.4 Generalized Switching Effect in Multiple Rounds

The above incompatible example implies that the BCT has limitations on the evaluation of boomerang switch in multiple rounds. In the definition of the BCT, it takes into account Δ_0 and ∇_0 to evaluate the switching probability of a single S-box. Δ_0 and ∇_0 correspond to β and γ respectively when E_m only consists in one S-box layer, and only β and γ matter in computing the switching probability (as shown in Formula 2.1). Thus, the

BCT measures the switching effect nicely for one S-box layer. However, when it comes to multiple S-box layers, this correspondence no longer exists. Consecutive multiple S-box layers are highly related. Such as the S-box with input difference $\Delta_0 = 0\text{xdf}$ in the first round in Figure 2.4, the first round switch requires $\nabla_0 = 0\text{xa9}$, while the second round switch requires the output difference Δ_1 to be 0xf1 , which causes an incompatibility. This fact reveals that several independent compatible boomerang switches could lead to incompatibilities when they are combined together. Therefore, it is necessary to find a way to consider consecutive rounds in a unified manner.

The above example shows a contradiction of the 3-tuple $(\Delta_0, \Delta_1, \nabla_0)$ for a S-box. Hence, it is crucial to analyze the relation among them. In the following, we provide two lemmas which are applicable to any S-box.

Lemma 1. *For any fixed Δ_0 and Δ_1 , for which the DDT entry is $2l$, l being a nonzero integer, the maximum number of nontrivial values of ∇_0 , for which a right quartet could be generated, is $2\binom{l}{2} + 1$.*

Proof. The straightforward case is when ∇_0 is equal to Δ_1 . Apart from that case, when $l > 1$, there are l paired values satisfying the differential propagation, then each two paired values can construct a quartet. In total, $\binom{l}{2}$ distinct quartets can be built. For each quartet, we denote the two paired values of S-box outputs by $\{(y_1, y_2), (y_3, y_4)\}$. There are two new ways to define ∇_0 such that a boomerang quartet could be generated: $y_1 \oplus y_3$ and $y_1 \oplus y_4$. Hence, there are $2\binom{l}{2} + 1$ nontrivial choices of ∇_0 . $\square$

Lemma 2. *For any fixed Δ_0 and ∇_0 , for which the BCT entry is $2l$ and the DDT entry is $2l'$, l and l' being nonzero integers, the maximum number of choices of Δ_1 , for which a right quartet could be generated, is $1 + (2l - 2l')/4$.*

Proof. The straightforward case is when Δ_1 is equal to ∇_0 , which corresponds to a part of the BCT entry ($2l'$ out of $2l$). For any other possible choices of Δ_1 , there will be two pairs of S-box outputs $\{(y_1, y_2), (y_3, y_4)\}$ satisfying $y_1 \oplus y_2 = y_3 \oplus y_4 = \Delta_1$ and $y_1 \oplus y_3 = \nabla_0$ (or $y_1 \oplus y_4 = \nabla_0$), which contributes 4 to the remaining value $(2l - 2l')$ of the BCT entry. Thus, the number of choices of Δ_1 is $1 + (2l - 2l')/4$. $\square$

In order to capture all these observations and to easily analyze the switching effect, we propose a new tool: the *Boomerang Difference Table (BDT)*.

Definition 3 (Boomerang Difference Table (BDT)). *Let S be an invertible function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$, and $(\Delta_0, \Delta_1, \nabla_0) \in \mathbb{F}_2^n$. The boomerang difference table (BDT) of S is a three-dimensional table, in which the entry for $(\Delta_0, \Delta_1, \nabla_0)$ is given by:*

$$BDT(\Delta_0, \Delta_1, \nabla_0) = \#\{x \in \{0, 1\}^n \mid S^{-1}(S(x) \oplus \nabla_0) \oplus S^{-1}(S(x \oplus \Delta_0) \oplus \nabla_0) = \Delta_0, \\ S(x) \oplus S(x \oplus \Delta_0) = \Delta_1\}.$$

The BDT reveals the probability of generating a boomerang quartet with a certain differential trail at the S-box level. Inspired by the efficient construction of the BCT proposed by Dunkelman [54], we show that the time complexity for generating the BDT for an n -bit S-box is $O(2^{2n})$, the algorithm being depicted in Algorithm 1.

Algorithm 1: The algorithm for constructing the BDT.

```

for all values of  $\nabla_0$  do
    Initialize an empty table  $T$  of  $2^n$  lists.
    for all values of  $x$  do
        Compute  $y = x \oplus S^{-1}(S(x) \oplus \nabla_0)$ .
        Concatenate  $x$  to  $T[y]$ .
    end
    Initialize a two-dimensional array  $Slice$  of  $2^n \times 2^n$  entries.
    for all entries in  $T$  do
        if the entry is not empty then
            for all pairs of values  $(x_i, x_j)$  in the entry do
                Increment  $Slice[x_i \oplus x_j, S(x_i) \oplus S(x_j)]$  by 1.
            end
        end
    end
    The entries of  $Slice$  corresponds to the entries of the BDT with
    the fixed  $\nabla_0$ .
end

```

It is easy to see that the BDT is a combination of the BCT and the DDT. Their relations and a property of the BDT are listed below:

$$DDT(\Delta_0, \Delta_1) = BDT(\Delta_0, \Delta_1, 0) = BDT(\Delta_0, \Delta_1, \Delta_1) \quad (2.2)$$

$$BCT(\Delta_0, \nabla_0) = \sum_{\Delta_1=0}^{2^n} BDT(\Delta_0, \Delta_1, \nabla_0) \quad (2.3)$$

$$BDT(0, 0, \nabla_0) = 2^n \quad (2.4)$$

As we can see from Equation (2.3), the BDT limits the switching effect offered by the BCT with the additional requirement of Δ_1 . To illustrate this point, we provide the following example.

Example 2. *The S-box of AES is used as illustration here. Suppose that the DDT entry is 4 for a fixed difference pair (Δ_0, Δ_1) , then a right quartet can be generated by two new choices of ∇_0 according to Property 1. Take ∇'_0 as one of them for example, the BCT entry for (Δ_0, ∇'_0) would be 6 (for any fixed Δ_0 in AES, the value of two positions in the BCT, for which the DDT value is 2, will increase by 4). However, among the three paired values, only two pairs lead to output difference of the S-box to Δ_1 , and the other pair leads to ∇'_0 . Thus, the BDT entry for $(\Delta_0, \Delta_1, \nabla'_0)$ is 4, and the entry for $(\Delta_0, \nabla'_0, \nabla'_0)$ is 2, both are lower than the BCT entry for (Δ_0, ∇'_0) . If the*

output difference is required to be either Δ_1 or ∇'_0 , the switching probability from Δ_0 to ∇'_0 for the S-box is lower than the probability $6/2^8$ evaluated by the BCT.

The above observation can be incorporated into the following lemma. Note that we define the *uniformity* of the BDT as the maximal value in the table, except the values where Δ_0 and Δ_1 are zero.

Lemma 3. *For any S-box, the uniformity of the BDT equals the uniformity of the DDT.*

Proof. Firstly, the BDT entry for $(\Delta_0, \Delta_1, \nabla_0)$ is no greater than $\text{DDT}(\Delta_0, \Delta_1)$ and $\text{BCT}(\Delta_0, \nabla_0)$. Secondly, for any choices of (Δ_0, Δ_1) , the entry in the DDT is smaller than or equal to the one in the BCT². Thirdly, Equation (2.2) always holds. The lemma is accordingly proved by the three observations. $\square$

Incompatibility. A 3-tuple $(\Delta_0, \Delta_1, \nabla)$ is incompatible if the corresponding entry of the BDT is 0. The incompatible difference pair (Δ_0, ∇) detected by the BCT can also be detected by the BDT through Equation (2.3). Moreover, the BDT is able to detect new incompatibilities even if the corresponding values of the BCT and the DDT are nonzero.

With the help of the BDT, the incompatibility exhibited in Section 2.1.3 can be easily detected: the BDT entry for $(0\text{xdf}, 0\text{xf1}, 0\text{xa9})$ is 0.

Variants of BDT. When the boomerang returns back in the decryption direction, the ∇_1 of the S-box in the last S-box layer of E_m determines the differential characteristic in the backward rounds. Thus, naturally a variant BDT' that takes into account $(\nabla_0, \nabla_1, \Delta_0)$ can be proposed to evaluate the last S-box layer:

$$\text{BDT}'(\nabla_0, \nabla_1, \Delta_0) = \#\{x \in \{0, 1\}^n \mid S(S^{-1}(x) \oplus \Delta_0) \oplus S(S^{-1}(x \oplus \nabla_0) \oplus \Delta_0) = \nabla_0, \\ S^{-1}(x) \oplus S^{-1}(x \oplus \nabla_0) = \nabla_1\}.$$

According to the symmetry of the boomerang, all the previous analysis of the BDT also apply to the BDT' . Furthermore, another variant of the BDT to capture all the four factors $(\Delta_0, \nabla_0, \Delta_1, \nabla_1)$ is needed when analyzing the middle S-box layers in the case where E_m consists of more than two rounds. The analysis is also similar, we ignore it to avoid redundancy.

2.1.5 Boomerang Switch in Two Rounds

In this section, we discuss the application of the BDT (BDT') to the boomerang switch in two rounds. E_m consists of two S-box layers and one linear layer in between — as depicted in Figure 2.5, the S-box layer being

²Lemma 1 in [42].

denoted by SL and the linear layer by R . The switching probability of the first S-box layer is denoted by p_1 , and the switching probability of the second S-box layer is denoted by p_2 , the whole probability of E_m is then computed by $r = p_1 p_2$.

Figure 2.5: A 2-round E_m which consists of two S-box layers and one linear layer.

In the analysis of the boomerang switch, the BDT is applied at the first S-box layer, and the BDT' is applied at the second S-box layer. In general, the probability p_1 and p_2 are the product of the switching probability of each S-box in the internal state, and are computed by:

$$p_1 = \prod_{(\Delta_0, \Delta_1, \nabla_0) \in L_1} BDT(\Delta_0, \Delta_1, \nabla_0)/2^n, \quad (2.5)$$

$$p_2 = \prod_{(\nabla_0, \nabla_1, \Delta_0) \in L_2} BDT'(\nabla_0, \nabla_1, \Delta_0)/2^n, \quad (2.6)$$

where L_1 contains the 3-tuple difference of the S-box in $(\beta, \beta', \gamma'')$, and L_2 contains the same in $(\gamma, \gamma', \beta'')$.

Notice that given E_m with the fixed β and γ , there might exist several differential characteristics satisfying this differential, which could contribute to increase the probability.

In practice, the 2-round boomerang switch can be classified into three cases:

- (a) The two differential characteristics have no same active S-box in both S-box layers.
- (b) The two differential characteristics activate the same S-boxes only in one of the two S-box layers.
- (c) The two differential characteristics activate the same S-boxes in both S-box layers.

Examples of the three cases are presented in Figure 2.6. In the following, we explain the three cases and provide simplified analysis for the first two cases.

Figure 2.6: Examples of three cases for the boomerang switch in two rounds

Analysis of the Three Cases

Due to the relation among the BDT, the BCT and the DDT, in the BDT-based analysis, we are able to switch the BDT to the BCT when there is no requirement for Δ_1 (Equation (2.3)), and to switch the BDT to the DDT when ∇_0 equals to zero (Equation (2.2)). And the BDT' can be handled similarly.

Case (a). If each S-box layer is evaluated independently, no S-box differential probability has to be paid due to the ladder switch. However, the result is different when they are combined together.

We call that the BDT entry is 2^n when both Δ_0 and Δ_1 equal to zero, so the S-box with zero input/output difference can be switched with probability 1. Then, regarding the first S-box layer, the switching probability only comes from the active S-boxes corresponding to β (β'). Since the difference ∇_0 equals to zero for those S-boxes, we can utilize the DDT to evaluate their switching probabilities instead of the BDT. Thus, p_1 equals to the probability of the differential propagation from β to β' , *i.e.*, $p_1 = Pr(\beta \xrightarrow{SL} \beta')$. In particular, the probability is 1 if no specific value of β' is required.

The analysis is similar for the second S-box layer with the help of the BDT', and $p_2 = Pr(\gamma \xrightarrow{SL^{-1}} \gamma')$. Also, p_2 equals to 1 if there is no requirement for the value of γ' .

Case (b). If the same active S-boxes are present in the first S-box layer, the analysis for the second S-box layer is similar to Case (a), *i.e.*, $p_2 = Pr(\gamma \xrightarrow{SL^{-1}} \gamma')$.

As for the first S-box layer, the switching probability p_1 can be directly computed from the analysis of the BCT if there is no requirement for β' .

Otherwise, the 3-tuple $(\Delta_0, \Delta_1, \nabla_0)$ has to be evaluated in the BDT, and the switching probability should be computed by Equation 2.5.

Finally, the case where the same active S-boxes are located in the second S-box layer can be analyzed similarly because of the symmetry of the boomerang attack.

Case (c). The BDT has to be applied to both S-box layers. Since specific values of all the intermediate states (β' , β'' , γ' and γ'') are required, there is a high chance of incompatibility.

2.2 Attack on 10-Round AES-256

In this section, we propose a related-key attack on 10-round AES-256 by applying the 2-round boomerang switch. The best previously published attack on this variant required 2^{45} time to partially obtain 35 subkey bits [31] (the rest of the key having to be brute-forced for full recovery), and it used a strong type of related-subkey oracles. The related-subkey setting requires a complex key access scheme [20] when compared to the related-key setting, and it could be too contrived for academic interest according to [46]. Our boomerang attack is mounted under a much more realistic scenario: a simple related-key setting with only two related keys, instead of four (or more) for most related-key boomerang attacks. We are able to recover the full 256-bit key with 2^{75} computations. This also compares favourably to previously known attacks in the same setting [28, 74], which require more related-keys and much more computations. We summarize the current attacks in Table 2.1.

Table 2.1: Summary of attacks on 10-round AES-256

Scenario	# keys	Time	Data	Result	Reference
Key Diff.	64/256	2^{172}	2^{114}	Full key	[74]/ [28]
Subkey Diff.	2	2^{45}	2^{44}	35 subkey bits	[31]
Key Diff.	2	2^{75}	2^{75}	Full key	this section

2.2.1 The Boomerang Distinguisher

The boomerang distinguisher³ is given in Figure 2.12. The trail used in E_0 is a 9-round related-key differential characteristic which is produced with the local collision strategy introduced in [33]. There are five active S-boxes in rounds 1-7, and we require all of them to use the optimal S-box

³As we use colors in the diagrams of the trails, we advise the reader to refer to the color scheme defined in Appendix 2.B.

differential transition $0x02 \rightarrow 0x14$. The key relation $\Delta K = K_A \oplus K_B$ is chosen as shown in Figure 2.12. The trail used in E_1 is a single-key differential characteristic that covers rounds 8-10, while round 10 is used for key recovery. The switching state is located at rounds 8 and 9.

Let us now compute the probability of this 9-round boomerang distinguisher. There are five active S-boxes in rounds 1-7, each of them is passed with differential probability 2^{-6} . Thus the overall differential probability for the first seven rounds is 2^{-30} .

Since rounds 8 and 9 are for boomerang switch and round 10 is for key recovery, the 9-round distinguisher is actually composed of E_0 and E_m , without E_1 . The 2-round boomerang switch is depicted in Figure 2.7, and the actual values are specified in Table 2.3. The value of $\Delta y_{0,0}^8$ is fixed to $0x14$ so that the three S-boxes in round 9 are inactive. In order to generate a right quartet, $\nabla y_{0,0}^8$ can take three values according to Lemma 1, and these values are $0x14$, $0x8c$ and $0x98$. Hence, we can obtain three trails for the boomerang switch. Since the active S-boxes marked with ■ are not concerned, the switching probability in round 8 only comes from the first S-box, and the BDT entries for $(0x02, 0x14, 0x14)$, $(0x02, 0x14, 0x8c)$ and $(0x02, 0x14, 0x98)$ are all 4 so that the switching probability is 2^{-6} for all the three trails. Regarding the boomerang switch in round 9, only the first S-box is concerned, and the BDT entries for the three trails are 2, 2 and 4 respectively. Finally the switching probability in the two rounds are $2^{-6} \cdot 2^{-7} + 2^{-6} \cdot 2^{-7} + 2^{-6} \cdot 2^{-6} = 2^{-11}$. In the end, the probability of the 9-round boomerang distinguisher is $2^{-30-30-11} = 2^{-71}$.

Figure 2.7: 2-round boomerang switch for the 9-round distinguisher against AES-256.

Experimental Verification. In order to verify the switching probability in round 8 and 9, we mounted an experiment. The amount of data used in each test is 2^{20} , and the test is iterated for 1000 randomly chosen key quartets satisfying the required key difference. In the end, the average number of right quartets obtained was 510, so the switching probability is $510/2^{20} = 2^{-11}$, which matches our evaluation.

2.2.2 Key Recovery

In the key recovery attack, round 10 is added at the end of the distinguisher. The attack works in the reverse direction of the normal boomerang attack, *i.e.*, we first choose the ciphertexts then choose the plaintexts. The algorithm is as follows.

1. Prepare a structure F of 2^{32} ciphertexts, by exhausting all possible values of the dark bytes while keeping the other bytes fixed.
2. Ask for decryption of F with K_A , and denote by G the set of plaintexts obtained.
3. For each plaintext $P \in G$, compute $P' = P \oplus \alpha$, and denote by H the new set of values P' .
4. Ask for encryption of H with K_B and insert the obtained ciphertexts into a hash table according to the 12 bytes which are inactive in .
5. In case of a collision of ciphertexts (C_3, C_4) in the hash table: we search the corresponding ciphertext pair (C_1, C_2) in F . Then, save all the candidates of the 32-bit information of RK_A^{10} and RK_B^{10} for which $D_{bK_A}(C_1) \oplus D_{bK_A}(C_2) = D_{bK_B}(C_3) \oplus D_{bK_B}(C_4) = \delta$. By D_b we denote the last round of decryption.
6. Repeat steps 1-5 $2^{41.5}$ times.

From each structure we can compose 2^{63} unordered pairs, $2^{63-32} = 2^{31}$ of them pass the last round. Thus, we expect $2^{31-71} = 2^{-40}$ right quartet per structure, and 3 right quartets out of $2^{41.5}$ structures.

Let us now compute the number of candidate quartets. The ciphertext has a 96-bit filter, so we are left with $2^{63+41.5-96} = 2^{8.5}$ candidate quartets. On average, each remaining quartet suggests 2^4 guesses for the 32-bit of RK_A^{10} and RK_B^{10} , respectively, or 2^8 in total. Thus, the $2^{8.5}$ candidate quartets would propose $2^{8+8.5} = 2^{16.5}$ key guesses for the 64 subkey bits, while 3 right quartets would all vote for the correct 64-bit subkey. No wrong key guesses survive and we can get the 32-bit subkeys $k_{0,0}^{10}, k_{3,1}^{10}, k_{2,2}^{10}, k_{1,3}^{10}$ of K_A and K_B as a result.

Notice that the trail in E_1 can follow two other truncated differential characteristics, see Figure 2.8. For either of the two, we can produce the actual differences similarly to the previous one, and the final probability of the two new distinguishers will remain unchanged. When we run the above algorithm, a right quartet is captured as long as it belongs to one of the three distinguishers. On average, we expect one right quartet per 2^{40} structures for each distinguisher, then 2^{40} structures are enough in order to have 3 right quartets.

Figure 2.8: Two other truncated differential characteristics for E_1 on AES.

Recovering Other Key Bytes. We can change the trail in E_1 to derive other subkey bytes of RK^{10} . We can construct three differential characteristics (see Figure 2.9) to recover the left 12 bytes of RK^{10} . The algorithm and complexity analysis are similar to the previous case and this allows to derive the full RK^{10} round key. Eventually, from the knowledge of this round key, we can now reduce the cipher to 9 rounds and recover the full key with various approaches with lower costs (for example, one can use the attack from [31] that requires only 2^{39} computations to recover the key on 9 round of AES-256).

Figure 2.9: Recovery of other subkey bytes in the boomerang attack against 10-round AES-256.

Complexity Analysis. The memory complexity is dominated by the hash table whose size is 2^{32} . As for the time and data complexity, step 2 and 4 need 2^{33} encryption/decryption oracle accesses for each structure, step 3 and 5 are negligible, thus we need $2^{33+40+2} = 2^{75}$ time and data to complete the attack.

2.3 Application to AES-192

In [32] Biryukov and Khovratovich proposed the first related-subkey amplified boomerang attack against full AES-192. Later, improvements were made [34] by slightly modifying the trail of E_0 , and this attack remains the best one as of today. In this section, by exploiting the boomerang switch in two rounds, we show an improvement of the attack in [34] and that the actual probability in [32] is much higher than expected. Table 2.2 summarizes the attacks and the improvements. Note that both attacks use the optimal S-box differential: $0x01 \rightarrow 0x1f$ in the boomerang trail.

Table 2.2: Summary of existing attacks against full-round AES-192 and the corresponding improvements with the help of the 2-round boomerang switch.

Attack	Time	Data	Evaluated by
[34]	2^{169}	2^{116}	[34]
	$2^{167.7}$	$2^{114.7}$	Section 2.3.1
[32]	2^{176}	2^{123}	[32]
	$2^{171.2}$	$2^{118.2}$	Section 2.3.2

2.3.1 Improvement of the Attack from Biryukov *et al.*

The attack in [34] uses 6-round differential characteristics for both E_0 and E_1 , and the first and last round is used for key extraction. The ladder switch is applied to round 6. Since the boomerang switching effect could exist in two rounds for AES, we tried to analyze their distinguisher with the method we developed. However, we found that it is hard to apply the 2-round boomerang switch to our benefit in their attack, due to the small dependency between the two differential characteristics. Hence, we should create a higher dependency in order to gain additional advantages from the 2-round boomerang switch. We eventually managed to produce a new differential characteristic for E_0 , while keeping the same differential characteristic for E_1 as in [34]. The distinguisher is shown in Figure 2.13 in Appendix 2.B.

We switch the boomerang at rounds 6 and 7, and the switching states are depicted in Figure 2.10. For the boomerang switch in round 6, we do not pay the S-boxes in the first three rows, and since the values of ■ are not concerned, the three S-boxes in the last row are free due to the analysis from the BDT. Thus, only the S-box at position (3, 1) should be counted. For the boomerang switch in round 7, only the S-box at position (0, 2) has to be counted due to the analysis of the BDT'. The choices of input/output differences of these two S-boxes might lead to different cases of 2-round boomerang switch described in Section 2.1.5.

Figure 2.10: The middle two rounds of the boomerang attack against full AES-192 in Section 2.3.1. The differences of the hashed bytes are not fixed.

After we iterated all possible input/output differences, four trails were found and they are given in Table 2.4. Take the first trail for example, it belongs to case (a). Although there is no overlapped S-box in the two rounds, a cost has to be paid. At the first S-box layer, the value of $\Delta y_{3,1}^6$ is fixed to $0x1f$, then the 3-tuple $(\Delta_0, \Delta_1, \nabla_0)$ of the S-box at $(3, 1)$ equals to $(0x01, 0x1f, 0x00)$, for which the BDT entry is 4. At the second S-box layer, the value of $\nabla x_{0,2}^7$ is fixed to $0x01$, then the 3-tuple $(\nabla_0, \nabla_1, \Delta_0)$ equals to $(0x1f, 0x01, 0x00)$, for which the BDT' entry is 4. Thus the 2-round boomerang switch has probability $4/2^8 \cdot 4/2^8 = 2^{-12}$ for the first trail. As for the other three trails, whose differential probabilities can be computed by a similar analysis of the BDT (BDT'), they have the same probability 2^{-13} . To sum up, the switching probability for rounds 6 and 7, for the given β and γ , is $2^{12} + 3 \cdot 2^{-13} = 2^{-10.7}$. Compared to the cost of the two rounds in [34], which is 2^{-12} , we get a speed-up factor of $2^{1.3}$, and the other parts of the attack remain the same: the new boomerang attack requires $2^{114.7}$ data and $2^{167.7}$ time.

Experimental Verification. We mounted an experiment to search for right quartets for the 2-round E_m in Figure 2.10. In the experiment, we set the data amount to 2^{20} and iterated the test for 1000 randomly chosen key quartets satisfying the required key difference. Finally the average number of right quartets obtained was 640. Hence, the success probability is $640/2^{20} = 2^{-10.7}$, which matches our analysis.

2.3.2 Evaluation and Improvement of the Attack from Biryukov *et al.*

The attack in [32] is the first published attack against the full AES-192. Here, we evaluate the boomerang distinguisher from their attack, and point out some mistakes in their evaluation. It turns out that their attack can actually be even more powerful. The boomerang trail is depicted in Figure 2.13 in

the Appendix 2.B.

Their attack uses 6-round differential characteristics for both E_0 and E_1 , and applies the ladder switch in round 7. In their evaluation, they fix the value of $\Delta y_{0,2}^6$ to $0x1f$ with probability 2^{-6} in the upper trail so that they can get a ladder switch in round 7. They claim that the output difference of the other two active S-boxes ($\Delta y_{0,1}^6, \Delta y_{0,3}^6$) can be any value such that it is the same as in the second related-key pair. Thus, in their analysis round 7 is free, and round 6 requires probability $2^{(-6-2 \cdot 3.5) \cdot 2} = 2^{-26}$ for the both sides of the boomerang.

However, when considering the switching effect in round 6 and 7, we found out that the values of $\Delta y_{0,1}^6$ and $\Delta y_{0,3}^6$ should be fixed as shown in Figure 2.11. For the two S-boxes in round 6, the value of Δ_1 has no effect on the boomerang switch in the round 7, which means that Δ_1 could be any value. However, Δ_0 and ∇_0 are fixed to $0x01$ and $0x1f$, so the number of choices of Δ_1 is only 1 according to Lemma 2: since the entries of the DDT and the BCT for $(0x01, 0x1f)$ are both 4, Δ_1 can only take the value $0x1f$. The corresponding BDT entry is 4, hence the switching probability of the two S-boxes is $2^{-6-6} = 2^{-12}$. Now, we are only left with two S-boxes: the one at $(0, 2)$ in round 6 and the other one at $(0, 2)$ in round 7. In order to get non-zero entries in the BDT (BDT') on the two S-boxes at the same time, we iterated all the values of $\Delta y_{0,2}^6$ and obtained several valid trails which are recorded in Table 2.5. The probability of each trail can be computed from the analysis of the BDT (BDT'). To sum up, the probability of the 2-round boomerang switch is $2^{-22.4}$. Compared to the previous evaluation, we get a speed-up factor of $2^{3.6}$.

Figure 2.11: The middle two rounds of the boomerang attack against full AES-192 in Section 2.3.2. The differences of the hashed bytes are not fixed.

Experimental Verification. We mounted a test on this reduced 2-round boomerang distinguisher using 2^{28} data, and iterated it for 100 randomly chosen key quartets. The result shows that the average success probability is $2^{-22.45}$, which demonstrates the validity of our analysis.

Further Improvement and Summary. Recall that the maximum switching probability for AES S-box is $6/2^8 = 2^{-5.4}$. Moreover, note that we can directly apply the BCT to the two S-boxes at $(0, 1)$ and $(0, 3)$ in round 6, because the output difference Δ_1 is not required to be a specific value. Hence, we can choose $\nabla_0 = 0x06$ with which the entry of the BCT is 6 so that the switching probability for each of the two S-boxes is $2^{-5.4}$, instead of 2^{-6} . We can simply replace the difference of the truncated characteristic for E_1 by using the optimal S-box differential transition: $0xbc \rightarrow 0x06$ instead. The probability of other parts remains the same, so the attack can be further improved by a factor of $2^{1.2}$.

Although the improvement fails to beat the result in Section 2.3.1, our point is that it indicates that the boomerang switch in multiple rounds do exist in a boomerang attack. The previous boomerang attacks that ignore the switching effect are thus unlikely to be evaluated properly.

2.4 Application to Deoxys-BC

In this section, we apply our method to improve the boomerang switch in the related-tweakey boomerang attacks against Deoxys-BC [41].

2.4.1 Improved 10-Round Boomerang Attack

At FSE 2018, Cid *et al.* [41] proposed a boomerang attack against 10-round Deoxys-BC-256 including a 9-round boomerang distinguisher with probability 2^{-122} , as shown in Table 2.6. The distinguisher is produced by taking into account the ladder switch in two rounds, which are rounds 5 and 6. However, they did not provide a concrete analysis of the two rounds. Here, our goal is to provide a specific BDT-based analysis, then search for new differential characteristics with higher probability.

As we can see in Table 2.6, both differential characteristics for E_0 and E_1 have an active S-box at position $(1, 1)$ in round 6, and the S-box is regarded as a part of E_1 by the ladder switch. The characteristic for E_1 specifies the input and output difference (from $0x32$ to $0x2f$), which has differential probability 2^{-7} , and the differential characteristics for E_0 and E_1 have no same active S-box in round 5, so the probability for rounds 5 and 6 is 2^{-14} in the boomerang distinguisher. In their analysis, the differential for the S-box at position $(1, 1)$ in round 6 of E_0 is denoted with $**$, which means its value is not crucial to the distinguisher. However, we show that the differential in E_0 is crucial by the analysis on the BDT'. For the targeted S-box, the characteristic for E_1 fixes the ∇_0 and ∇_1 to $0x2f$ and $0x32$, respectively. We confirmed that there exists only one choice of Δ_0 such that the BDT' entry for $(0x2f, 0x32, \Delta_0)$ is nonzero. The value is $0x32$ ($0x00$ is excluded due to the characteristics for E_0), and the BDT entry is 2. Then, we can uniquely obtain the value $\Delta y_{1,2}^5 = 0x19$ in the characteristic for E_0 through the inverse MixColumns. For the S-box at position $(1, 2)$ in round 5, the

differential characteristic for E_0 fixes the $\Delta_0 = 0x80$ and $\Delta_1 = 0x19$, and the characteristic for E_1 fixes $\nabla_1 = 0$. The BDT entry for $(0x80, 0x19, 0x00)$ is 2. Therefore, the probability 2^{-14} of the two rounds comes from the two bytes in round 5 and round 6, respectively.

Since the switching probability is determined, the possible direction to improve the attack becomes clear. Recall that the optimal switching probability is $2^{-5.4}$ for the AES S-box, thus a potential direction is to try to improve the switching probability in round 5 or round 6 or both. Due to the careful optimization done in [41], we will use the same truncated differential characteristics and only replace the difference. We borrowed the strategy of using an automated search tool from [41], and obtained a new differential characteristic for E_1 , as shown in Table 2.7. At position $(1, 2)$ in round 5, the BDT entry for $(0x80, 0xae, 0x00)$ is 4 which gives probability 2^{-6} , and the switching probability at position $(1, 1)$ in round 6 is 2^{-7} because the BDT' entry is 2 for $(0xe1, 0x47, 0x47)$. Moreover, there exists another differential characteristic with the same differential and has probability 2^{-14} , as shown in Table 2.8. In the end, the switching probability in rounds 5 and 6 is $2^{-13} + 2^{-14} = 2^{-12.4}$. We get a speed-up factor of $2^{1.6}$ and the probability of the new boomerang distinguisher is $2^{-120.4}$.

Experimental Verification. In the experiment, we used 2^{20} data, and iterated it for 1000 randomly chosen key quartets. The result shows that the average success probability is $2^{-12.4}$, which confirms our improvement evaluation.

2.5 Conclusion

In this chapter, we performed an extensive analysis of the switching effect between the two differential characteristics of a boomerang distinguisher. Specifically, we exploited the principle behind the ladder switch, and showed that the sparser the diffusion layer is, the more rounds the switching effect exists. Moreover, we introduced the BDT as a generalized method to easily evaluate the boomerang switching probability in multiple rounds. The extended switching effect was demonstrated by several applications, which includes the currently best related-key attack on 10-round AES-256, two improved attacks on full AES-192 and an improved attack on Deoxys-BC.

Finally, we would like to emphasize that the boomerang switching effect can exist in multiple rounds, regardless of the attacker creating it deliberately or not. This phenomenon was so far widely ignored by the cryptanalysis community.

Appendices

2.A The Round Function of SKINNY

The internal state of SKINNY [19] is represented by a 4×4 array, the round function consisting of five operations:

1. *Subcells* - The S-box is applied to all cells (4-bit S-box for SKINNY-64, 8-bit S-box for SKINNY-128).
2. *AddRoundConstants* - Three round constants are added to the first column of the internal state.
3. *AddRoundTweakey* - The round tweakey is XORed to the first two rows of the internal state.
4. *ShiftRows* - A cyclic shift of the i -th row by i positions is applied to the right, for $i = 0, 1, 2, 3$.
5. *MixColumns* - Each array column is multiplied by a binary matrix M . The matrix M^{-1} of the inverse *MixColumns* is also shown below.

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad M^{-1} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

2.B Differential Paths and Boomerang Distinguishers

We give all the details of the differential characteristics that are used in this chapter. By ΔA we denote the upper trail difference in the internal state, and by ∇A the same for the lower trail. The differences are represented in hexadecimal and differences that are not crucial to the distinguishers are denoted with “**”. The S-boxes that are crucial for the boomerang switch are highlighted in red.

Color Scheme. We extensively use colors in our figures in order to provide better understanding. Here is the color scheme utilized.

- ■ Fixed S-box input difference.
- ■ Fixed S-box output difference.
- ■ Arbitrary difference.

- Unfixed difference, depending on different characteristics.

- MixColumns expansion of .

Table 2.3: Internal state difference for rounds 8 and 9 in the 9-round distinguisher of AES-256. The trails denoted with $\dagger$ have the same ΔA as trail 1.

Trail		Round 8		Round 9		Prob.
		Before SB	After SB	Before SB	After SB	
1	ΔA	02 02 02 02	14 ** ** **	00 ** ** **	00 ** ** **	$2^{-6} \cdot 2^{-7}$
		00 00 00 00	00 00 00 00	00 ** ** **	00 ** ** **	
		00 00 00 00	00 00 00 00	00 ** ** **	00 ** ** **	
		00 00 00 00	00 00 00 00	** ** ** **	** ** ** **	
	∇A	** 00 00 00	14 00 00 00	6b 00 00 00	c0 00 00 00	
		00 ** 00 00	00 1e 00 00	00 00 00 00	00 00 00 00	
		00 00 ** 00	00 00 a9 00	00 00 00 00	00 00 00 00	
		00 00 00 **	00 00 00 c8	00 00 00 00	00 00 00 00	
$2^\dagger$	∇A	** 00 00 00	8c 00 00 00	1a 00 00 00	c0 00 00 00	$2^{-6} \cdot 2^{-7}$
		00 ** 00 00	00 ca 00 00	00 00 00 00	00 00 00 00	
		00 00 ** 00	00 00 a2 00	00 00 00 00	00 00 00 00	
		00 00 00 **	00 00 00 fe	00 00 00 00	00 00 00 00	
$3^\dagger$	∇A	** 00 00 00	98 00 00 00	71 00 00 00	c0 00 00 00	$2^{-6} \cdot 2^{-6}$
		00 ** 00 00	00 d4 00 00	00 00 00 00	00 00 00 00	
		00 00 ** 00	00 00 0b 00	00 00 00 00	00 00 00 00	
		00 00 00 **	00 00 00 36	00 00 00 00	00 00 00 00	

Table 2.4: Differential characteristic of the middle two rounds that is used in the boomerang attack against AES-192 in Section 2.3.1.

Trail		Round 6				Round 7				Prob.
		Before SB		After SB		Before SB		After SB		
1	ΔA	00 00 00 00	00 00 00 00	** ** 00 **	** ** 00 **	$2^{-6} \cdot 2^{-6}$				
		00 00 00 00	00 00 00 00	** ** 00 **	** ** 00 **					
		00 00 00 00	00 00 00 00	** ** ** **	** ** ** **					
		01 01 01 01	** 1f ** **	** ** 01 **	** ** ** **					
	∇A	** ** ** **	1f 1f 1f 1f	00 00 01 00	00 00 1f 00					
		00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00					
2	ΔA	00 00 00 00	00 00 00 00	** ** 00 **	** ** 00 **	$2^{-6} \cdot 2^{-7}$				
		00 00 00 00	00 00 00 00	** ** 00 **	** ** 00 **					
		00 00 00 00	00 00 00 00	** ** ** **	** ** ** **					
		01 01 01 01	** 1f ** **	** ** 01 **	** ** ** **					
	∇A	** ** ** **	1f 1f ce 1f	00 00 99 00	00 00 1f 00					
		00 00 00 **	00 00 00 34	00 00 00 00	00 00 00 00					
3	ΔA	00 00 00 00	00 00 00 00	** ** bc **	** ** 06 **	$2^{-7} \cdot 2^{-6}$				
		00 00 00 00	00 00 00 00	** ** bc **	** ** ** **					
		00 00 00 00	00 00 00 00	** ** ** **	** ** ** **					
		01 01 01 01	** a3 ** **	** ** 62 **	** ** ** **					
	∇A	** ** ** **	1f 1f 1f 1f	00 00 01 00	00 00 1f 00					
		00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00					
4	ΔA	00 00 00 00	00 00 00 00	** ** 01 **	** ** 1f **	$2^{-7} \cdot 2^{-6}$				
		00 00 00 00	00 00 00 00	** ** 01 **	** ** ** **					
		00 00 00 00	00 00 00 00	** ** ** **	** ** ** **					
		01 01 01 01	** 1e ** **	** ** 03 **	** ** ** **					
	∇A	** ** ** **	1f 1f 1f 1f	00 00 01 00	00 00 1f 00					
		00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00					

Table 2.5: Differential characteristic of the middle two rounds that is used in the boomerang attack against AES-192 in Section 2.3.2. The trails denoted with $\dagger$ have the same ΔA as trail 1.

Trail		Round 6		Round 7		Prob.
		Before SB	After SB	Before SB	After SB	
1	ΔA	00 01 01 01	00 1f 1f 1f	00 00 00 **	00 00 00 **	$2^{-18} \cdot 2^{-6}$
		00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
		00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
		00 00 00 00	00 00 00 00	** ** ** *	** ** ** *	
	∇A	** 01 01 01	1f 1f 1f 1f	00 00 01 00	00 00 1f 00	
		00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
		00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
		00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
$2^\dagger$	∇A	** 01 00 01	1f 1f 00 1f	00 00 37 00	00 00 1f 00	$2^{-18} \cdot 2^{-7}$
		00 00 00 **	00 00 00 9d	00 00 00 00	00 00 00 00	
		** 00 00 00	45 00 00 00	00 00 00 00	00 00 00 00	
		00 ** 00 00	00 f1 00 00	00 00 00 00	00 00 00 00	
$3^\dagger$	∇A	** 01 bc 01	1f 1f 06 1f	00 00 5f 00	00 00 1f 00	$2^{-18} \cdot 2^{-7}$
		00 00 00 **	00 00 00 98	00 00 00 00	00 00 00 00	
		** 00 00 00	fb 00 00 00	00 00 00 00	00 00 00 00	
		00 ** 00 00	00 2c 00 00	00 00 00 00	00 00 00 00	
$4^\dagger$	∇A	** 01 bd 01	1f 1f 19 1f	00 00 69 00	00 00 1f 00	$2^{-18} \cdot 2^{-7}$
		00 00 00 **	00 00 00 05	00 00 00 00	00 00 00 00	
		** 00 00 00	be 00 00 00	00 00 00 00	00 00 00 00	
		00 ** 00 00	00 d5 00 00	00 00 00 00	00 00 00 00	
5	ΔA	00 01 01 01	00 1f 89 1f	00 00 37 **	00 00 1f **	$2^{-19} \cdot 2^{-7}$
		00 00 00 00	00 00 00 00	00 00 96 00	00 00 ** 00	
		00 00 00 00	00 00 00 00	00 00 96 00	00 00 ** 00	
		00 00 00 00	00 00 00 00	** ** ** *	** ** ** *	
	∇A	** 01 00 01	1f 1f 00 1f	00 00 37 00	00 00 1f 00	
		00 00 00 **	00 00 00 9d	00 00 00 00	00 00 00 00	
		** 00 00 00	45 00 00 00	00 00 00 00	00 00 00 00	
		00 ** 00 00	00 f1 00 00	00 00 00 00	00 00 00 00	
6	ΔA	00 01 01 01	00 1f b2 1f	00 00 41 **	00 00 1f **	$2^{-19} \cdot 2^{-7}$
		00 00 00 00	00 00 00 00	00 00 ad 00	00 00 ** 00	
		00 00 00 00	00 00 00 00	00 00 ad 00	00 00 ** 00	
		00 00 00 00	00 00 00 00	** ** ** *	** ** ** *	
	∇A	** 01 01 01	** 01 b2 01	00 00 41 00	00 00 1f 00	
		00 00 00 **	00 00 00 76	00 00 00 00	00 00 00 00	
		** 00 00 00	6d 00 00 00	00 00 00 00	00 00 00 00	
		00 ** 00 00	00 f6 00 00	00 00 00 00	00 00 00 00	

Table 2.6: The 9-round distinguisher of Deoxys-BC-256 from [41]

Round	Initial Δ	Tweakey Δ	Before SB	After SR	Prob.
1	00 00 7b 00 b0 c0 00 00 00 00 af 00 00 00 00 c2	00 00 7b 00 b0 c0 00 00 00 00 af 00 00 00 00 c2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	e0 80 00 00 00 4d 00 00 00 00 00 00 00 00 00 ea	e0 80 00 00 00 4d 00 00 00 00 00 00 00 00 00 ea	b4 c9 00 00 21 00 00 00 00 00 00 00 73 00 00 00	2^{-28}
3	63 89 00 00 85 c9 00 00 00 c9 00 00 00 40 00 00	00 89 00 00 85 00 00 00 00 c9 00 00 00 40 00 00	63 00 00 00 00 c9 00 00 00 00 00 00 00 00 00 00	8d 00 00 00 8c 00 00 00 00 00 00 00 00 00 00 00	2^{-14}
4	8e 00 00 00 8e 00 00 00 01 00 00 00 00 00 00 00	8e 00 00 00 8e 00 00 00 01 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 80 03 13 00 00 00 00 98 00 00	00 00 00 00 00 00 80 03 13 00 00 00 00 98 00 00	00 00 00 00 00 ** ** 00 00 00 ** 00 00 00 ** 00	1
6	00 ** ** 00 00 ** ** 00 00 ** ** 00 00 ** ** 00	00 00 81 07 00 00 00 35 00 00 00 b4 00 1d 00 00	00 ** ** 07 00 ** ** 35 00 ** ** b4 00 ** ** 00	00 ** ** ** ** ** ** 00 ** ** 00 ** 00 00 ** **	1
5	** ** 00 ** ** ** 00 55 00 ** ** ** ** 00 ** **	00 00 00 00 00 00 00 55 00 00 00 00 00 00 00 84	** ** 00 ** ** ** 00 00 00 ** ** ** ** 00 ** **	** e4 00 ** ** 00 00 ** ** 8f 00 ** ** 5c 00 **	1
6	** 00 00 49 00 32 00 00 00 05 00 00 00 00 00 **	00 00 00 49 00 00 00 00 00 05 00 00 00 00 00 00	** 00 00 00 00 32 00 00 00 00 00 00 00 00 00 **	ee 00 00 00 2f 00 00 00 00 00 00 00 b6 00 00 00	2^{-7}
7	00 00 00 00 06 00 00 00 00 00 00 00 71 00 00 00	00 00 00 00 06 00 00 00 00 00 00 00 71 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
9	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 e3 00 00 00 00 00 00 00 0c 00 00	00 00 00 00 00 e3 00 00 00 00 00 00 00 0c 00 00	00 00 00 00 72 00 00 00 00 00 00 00 00 00 9d 00	2^{-12}

Table 2.7: Improved 9-round distinguisher for Deoxys-BC-256. The probabilities marked with † are only counted once due to the 2-round boomerang switch.

Round	Initial Δ	Tweakey Δ	Before SB	After SR	Prob.
5	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	$2^{-6} \uparrow$
	00 00 00 00	00 00 80 03	00 00 80 03	00 ae ** 00	
	00 00 00 00	13 00 00 00	13 00 00 00	00 00 ** 00	
	00 00 00 00	00 98 00 00	00 98 00 00	00 00 ** 00	
6	00 e9 ** 00	00 00 81 07	00 e9 ** 07	00 ** ** *	1
	00 47 ** 00	00 00 00 35	00 47 ** 35	** ** ** 00	
	00 ae ** 00	00 00 00 b4	00 ae ** b4	** ** 00 **	
	00 ae ** 00	00 1d 00 00	00 b3 ** 00	00 00 ** **	
5	** ** 00 **	00 00 00 00	** ** 00 **	** ab 00 **	1
	** ** 00 2a	00 00 00 2a	** ** 00 00	** 00 00 **	
	00 ** ** *	00 00 00 00	00 ** ** *	** dd 00 **	
	** 00 ** *	00 00 00 11	** 00 ** *	** 90 00 **	
6	** 00 00 24	00 00 00 24	** 00 00 00	e9 00 00 00	$2^{-7} \uparrow$
	00 47 00 00	00 00 00 00	00 47 00 00	e1 00 00 00	
	00 a1 00 00	00 a1 00 00	00 00 00 00	00 00 00 00	
	00 00 00 **	00 00 00 00	00 00 00 **	f1 00 00 00	
7	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	1
	c1 00 00 00	c1 00 00 00	00 00 00 00	00 00 00 00	
	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
	38 00 00 00	38 00 00 00	00 00 00 00	00 00 00 00	
8	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	1
	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
9	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	2^{-12}
	00 00 00 00	00 71 00 00	00 71 00 00	c0 00 00 00	
	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
	00 00 00 00	00 83 00 00	00 83 00 00	00 00 8f 00	
Master tweakey difference ∇K					
00 00 00 00 00 d0 00 7e 00 00 00 00 00 00 00 00					
00 00 00 00 00 df 00 66 00 00 00 00 00 00 00 00					

2.B. DIFFERENTIAL PATHS AND BOOMERANG DISTINGUISHERS41

Table 2.8: Another 2-round differential characteristic of the boomerang distinguisher in Table 2.7. The probabilities marked with † are only counted once due to the 2-round boomerang switch.

Round	Initial Δ	Tweakey Δ	Before SB	After SR	Prob.
5	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	$2^{-7} \dagger$
	00 00 00 00	00 00 80 03	00 00 80 03	00 96 ** 00	
	00 00 00 00	13 00 00 00	13 00 00 00	00 00 ** 00	
	00 00 00 00	00 98 00 00	00 98 00 00	00 00 ** 00	
6	00 a1 ** 00	00 00 81 07	00 a1 ** 07	00 ** ** **	1
	00 37 ** 00	00 00 00 35	00 37 ** 35	** ** ** 00	
	00 96 ** 00	00 00 00 b4	00 96 ** b4	** ** 00 **	
	00 96 ** 00	00 1d 00 00	00 8b ** 00	00 00 ** **	
5	** ** 00 **	00 00 00 00	** ** 00 **	** 96 00 **	1
	** ** 00 2a	00 00 00 2a	** ** ** 00	** 96 00 **	
	00 ** ** **	00 00 00 00	00 ** ** 00	** 00 00 **	
	** 00 ** **	00 00 00 11	** 00 ** **	** 96 00 **	
6	** 00 00 24	00 00 00 24	** 00 00 00	e9 00 00 00	$2^{-7} \dagger$
	00 37 00 00	00 00 00 00	00 37 00 00	e1 00 00 00	
	00 a1 00 00	00 a1 00 00	00 00 00 00	00 00 00 00	
	00 00 00 **	00 00 00 00	00 00 00 **	f1 00 00 00

Figure 2.12: Boomerang attack on 10-round AES-256. The red rectangles depict the locations of 2-round boomerang switch.

Figure 2.13: Two boomerang attacks on full-round AES-192. For E_0 , the left trail is the one used in [32] and is evaluated in Section 2.3.2, the middle trail is the one used in Section 2.3.1. Both attacks use the same trail for E_1 as shown on the right-hand side. The red rectangles depict the locations of 2-round boomerang switch.

Chapter 3

Improved Related-Key Attack on the Full AES-256

Since the design of Rijndael, AES has attracted tremendous efforts from the research community in security analysis. One of the most important security features of AES is its proven resistance against differential and linear cryptanalysis, which were applied to its predecessor — DES. It achieved this by the so-called *wide trail strategy* [44], *e.g.*, the minimum number of active S-boxes (those with non-zero differences) in 4 consecutive AES rounds for any differential characteristic in the single-key setting is 25. It has been analyzed by many cryptanalysis techniques¹, just to name a few here. As mentioned in the previous chapter, Biryukov, Khovratovich, and Nikolić gave the first related-key attack against AES-256 by using differentials [33] in 2009, the complexity of which was later improved in [32] with related-key boomerang attack. In [83], Lu *et al.* gave 7-round attacks against AES-128 and AES-192, and 8-round against AES-256 by using impossible differentials. Leveraging integral cryptanalysis, Ferguson *et al.* [58] gave a practical attack against 6-round AES-128 and then the first attack against 7-round AES-128. In terms of complexities of the key-recovery attacks in the single-key setting, the best attacks up to date are due to the the Demirci-Selçuk meet-in-the-middle attack [48, 49, 56]. The meet-in-the-middle attack, which was previously known to be powerful for finding preimages of hash functions, led to attacks against 7-round AES-128 in some hashing modes [104]. Many more attacks were found under other cryptanalysis settings *e.g.*, single-key, related-key, hashing modes, for various security aspects, *e.g.*, distinguisher, key-recovery, collision/preimage finding in hashing modes, etc. Up to date, the most successful key-recovery² attack against AES-128/192/256 is for 7, 9, and 9 out of 10, 12, and 14 rounds respectively in the single-key setting, due to Derbez *et al.* [49] and an improvement by Li *et al.* [81].

To the best of our knowledge, the best attack against AES, in terms of number of attacked rounds, is due to Biryukov *et al.* [32, 33] which dates back to 2009, where the key of the full version of AES-256 and AES-192 can be

¹Only a few papers are cited here as examples since there are simply too many results.

²Besides those optimized brute-force style attacks, such as [35].

recovered under the related (sub-)key setting. While the bound given by the wide trail strategy could not be overcome in the differential attack under the single-key setting, differential characteristics with much higher probability exist in the related-key setting, where the differences from round keys and the data path can be cancelled out. Also, the boomerang attack is able to utilize two high-probability differential characteristics for small number of rounds. The related-key attack is successful due to these two properties.

Contributions of this Chapter. Biryukov *et al.*'s works [32] remain as the best publicly known key-recovery attack against the full AES-256 under the related-key model, and there exists no essential improvement since 2009. In this chapter, we try to improve their attack, in terms of data and time complexities, under the same related-subkey boomerang attack framework. The core idea comes from the observation that, while structures have been used in plaintext to enjoy the birthday effect and improve cryptanalysis³, the same could potentially be applied to key material as well, even though this has not been tried yet. It is necessary to note that similar expressions to “key structure” already existed in some papers before, such as [28]. Their purpose is to generate some required subkey differences from key structure, due to non-linear key schedule of their targeted ciphers. However, our aim in this chapter is to make further improvement by enjoying birthday effect of a key structure. There are some technical difficulties to overcome in the key structure boomerang attack framework, before the idea can eventually work out.

- Firstly, when a structure rather than a fixed or chosen difference is introduced in the key material, one has to ensure that the uncontrollable difference in the key will not affect the validity of the two differential characteristics in the data path of the boomerang attack.
- Secondly, when a structure is applied to the key, one has to ensure that the two differentials are neutral to each other in two halves of the key schedule, i.e., the differential characteristics in the key schedule in one half will not affect the other half regardless of the actual key difference chosen from the structure.
- Thirdly, one has to ensure that the key difference will not affect the probability of the differential characteristics. Note that Biryukov *et al.* chose the high probability difference transition of the S-box (those with 2^{-6} , rather than the 2^{-7} ones for the AES S-box) in order to increase the overall probability of the differential characteristics. In our case of structure we only utilize those 2^{-6} difference pairs to achieve the same optimization. Although this only happens once in every 2^8 difference pairs (so we lose most of the pairs), we gain back by enforcing the

³It is hard to attribute this technique to a certain entity, so we refer to those works adopted this technique, such as [31, 32, 33].

high probability 2^{-6} transition once, and re-use it multiple times in the differential characteristics.

The final result, presented in Section 3.2, turns out to improve both the time and data complexities of Biyukov *et al.* attack by a factor of about 2^8 , at the expense of the number of required keys being increased from 4 to 2^{19} . A detailed comparison is provided in Table 3.1. This work is currently under submission.

Table 3.1: Comparison with previous key-recovery attacks on the full AES-256 under related-key model

Attack	Time	Data	Memory	# keys	Reference
Related-Key Differential	2^{131}	2^{131}	2^{65}	2^{35}	[33]
Related-Key Boomerang	$2^{99.5}$	$2^{99.5}$	2^{77}	4	[32]
Key-Structure	$2^{92.5+s}$	2^{92+s}	2^{89-s}	2^{17-s}	Ours
Boomerang	2^{92+s}	2^{91+s}	2^{89-s}	2^{19-s}	

Note: $0 \leq s \leq 7.5$

Organization. The rest of this chapter is organized as follows. Section 3.1 explains the ideas behind key structures. The details of the improved attack are given in Section 3.2. Finally, Section 3.3 concludes the chapter.

3.1 Key Structures

As explained in Chapter 1, in differential cryptanalysis, an attacker tries to find a distinguisher of a cipher so that he can distinguish the cipher from a random permutation. After that, key recovery attacks can be mounted based on the distinguisher directly, or with additional rounds added before and/or after the distinguisher. In this chapter, we focus on the latter one.

We assume that there is a distinguisher which consists of a differential $\alpha \rightarrow \beta$ with probability p covering the last $(r - r_0)$ rounds of the target cipher (r rounds in total). In order to launch a full-round attack, r_0 rounds should be prepended to the distinguisher. The aim of the attacker is to obtain enough ciphertext pairs with the difference β by querying the encryption oracle with pre-chosen plaintexts. We define V to be the space spanned by all the plaintext differences that may lead to the difference α after the first r_0 rounds, and let $m = \log_2|V|$.

3.1.1 Structure from Plaintext

The first step of the attack is to generate pairs of plaintexts whose output differences after the first r_0 rounds are the expected input difference α of the differential. The way to improve the efficiency of this step is to build a structure of plaintexts which consists of $P \oplus v_i$, where P is chosen randomly and $v_i \in V$. The XOR difference between any two elements of the structure belongs to V . In this way, at most 2^{2m-1} unordered plaintext pairs (P_i, P_j) (the order of i, j does not matter) can be composed from a single structure, while the data and time complexity to prepare this structure is only 2^m . We refer to the ratio between the number of pairs generated and the size of structure as *gain* (this is also called birthday effect in other places), quantitatively it is $2^{(2m-1)-m} = 2^{m-1}$ here. If more plaintext pairs are needed for the attack, another new structure can be constructed in the same way by selecting another random value of P . However, this would not increase the overall gain. As can be seen, the gain only depends on the structure size. Thus, if the structure size could be increased, the attack complexity would be reduced accordingly.

3.1.2 Structure from Key

In the related-key setting, the attacker is allowed to choose a desired relation between keys. The most common form is: for an unknown key K_1 , the attacker uses a XOR difference D to produce another key $K_2 = K_1 \oplus D$. Then, the subkey additions in round functions can be used to cancel some differences in the differential attack in order to obtain better differential characteristics. Compared to the single-key setting, the related-key setting provides additional freedom in choosing the key difference D . In our case, this fact enlightens us that a key structure utilizing the key difference could be used to improve the attack.

For a related-key differential characteristic with key difference D , we can build a key structure from the original secret key K_1 , and let $\{K_1 \oplus v_i \mid v_i \in V_D\}$ be the set of keys inside the structure, where V_D is the space consisting of the differences that have the same truncated difference as that of D . Similarly, we define $m_k = \log_2 |V_D|$. Together with a plaintext structure, at most a total of $2^{2(m+m_k)-1}$ unordered pairs of $((P_i, K_i), (P_j, K_j))$ can be obtained, while only 2^{m+m_k} data/queries are used. Thus, the gain increases to $2^{2(m+m_k)-1-(m+m_k)} = 2^{m+m_k-1}$, compared to the use of plaintext structure alone.

3.1.3 The Use of Key Structure

Key structure should be applied together with plaintext structure to provide additional advantage. However, compared to plaintext structure, key structure has more constraints in its application. Due to the fact that each plaintext in the plaintext structure will be encrypted with each key in the key structure during an attack, the difference between the pairs of keys will

not be fixed, thus the subkey differences in the whole rounds are difficult to control and are unlikely to match the exact differential characteristic $\alpha \rightarrow \beta$.

Hence, in order to have as many valid key pairs as possible, the key schedule is better to be linear or the proportion of the non-linear part is small, and the key difference should not have strong impact on the truncated differential characteristic, which means that the truncated differential characteristic should be able to be instantiated with many differences. Otherwise, if the differential characteristic is valid with only a small proportion of key pairs, smaller than 2^{-m_k} , the use of key structure will only weaken the attack. On the other hand, the distinguisher obtained with the key structure might not be as good as the original one without key structure, because the probability of the differential characteristic will vary according to key difference since the propagation of non-linear part in the internal state will be different. All in all, in order to make good use of key structure, the extra data and time consumption of it should be lower than the gain it offers.

Last but not least, it is necessary to mention that at most one key structure can be constructed in an attack as the key structure is created from the original secret key which is fixed.

3.2 Improved Related-Key Attack on AES-256

In this section, we apply key structure to the related-key boomerang attack on AES-256, which is based on the attack in [32]. We refer the specifications of AES in Section 1.1.3. We will first give an overview of the boomerang distinguisher, then describe the construction of the key structure, and finally explain the details of the attack.

The differential characteristics used in our boomerang attack are depicted in Figure 3.1. The differential characteristic of E_1 is fixed, while the differential characteristic of E_0 has a lot of candidates. In Figure 3.1, different colors refer to different values. The differentials for all the active S-boxes of the differential characteristic of E_1 are set to be $(0x01, 0x1f)$, which holds with probability 2^{-6} . For the differential characteristic of E_0 , the red and blue hashed cells are not fixed but always pass the S-box differential with the maximal probability 2^{-6} , and the green cells are unknown. The switching position of the boomerang distinguisher is pointed by the green ovals.

The differences in the key schedule are given in Table 3.2. Since the differential characteristic of E_0 is not fixed, we use the variables R , B and G_i to represent its truncated pattern, $i = 1, 2, 3, 4$. Given the value of R , B is then derived from the table DDT with the requirement that $DDT(R, B) = 4$, lastly the value of G_i is uniquely determined by B through the MixColumns transformation:

$$\begin{pmatrix} 0 \\ B \\ 0 \\ 0 \end{pmatrix} \xrightarrow{\text{MixColumns}} \begin{pmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{pmatrix}.$$

Table 3.2: Key schedule difference for the boomerang attack on AES-256. The values are given in hexadecimal notation.

ΔK^i																											
0	?	00	00	00	G ₁	G ₁	G ₁	G ₁	1	00	00	00	00	G ₁	00	G ₁	00	2	00	00	00	00	G ₁	G ₁	00	00	
	?	R	R	R	?	G ₂	G ₂	G ₂		00	R	00	R	G ₂	00	G ₂	00		00	00	R	R	00	G ₂	G ₂	00	00
	?	00	00	00	G ₃	G ₃	G ₃	G ₃		00	00	00	00	G ₃	00	G ₃	00		00	00	00	00	00	G ₃	G ₃	00	00
	?	00	00	00	G ₄	G ₄	G ₄	G ₄		00	00	00	00	G ₄	00	G ₄	00		00	00	00	00	00	G ₄	G ₄	00	00
3	00	00	00	00	G ₁	00	00	00	4	00	00	00	00	G ₁	G ₁	G ₁	G ₁										
	00	R	00	00	G ₂	00	00	00		00	R	R	R	?	?	?	?										
	00	00	00	00	G ₃	00	00	00		00	00	00	00	G ₃	G ₃	G ₃	G ₃										
	00	00	00	00	G ₄	00	00	00		00	00	00	00	G ₄	G ₄	G ₄	G ₄										
∇K^i																											
0	?	?	?	?	?	00	?	00	1	?	01	?	00	?	?	00	00	2	?	?	00	00	?	00	00	00	
	X	X	X	X	1f	00	1f	00		X	00	X	00	1f	1f	00	00		X	X	00	00	1f	00	00	00	
	?	?	?	?	1f	00	1f	00		?	00	?	00	1f	1f	00	00		?	?	00	00	1f	00	00	00	
	?	?	?	?	21	00	21	00		?	00	?	00	21	21	00	00		?	?	00	00	21	00	00	00	
3	?	01	01	01	3e	3e	3e	3e	4	01	00	01	00	3e	00	3e	00	5	01	01	00	00	3e	3e	00	00	
	X	00	00	00	1f	1f	1f	1f		00	00	00	00	1f	00	1f	00		00	00	00	00	1f	1f	00	00	
	?	00	00	00	1f	1f	1f	1f		00	00	00	00	1f	00	1f	00		00	00	00	00	1f	1f	00	00	
	?	00	00	00	21	21	21	21		00	00	00	00	21	00	21	00		00	00	00	00	21	21	00	00	
6	01	00	00	00	3e	00	00	00	7	01	01	01	01	?	?	?	?										
	00	00	00	00	1f	00	00	00		00	00	00	00	1f	1f	1f	1f										
	00	00	00	00	1f	00	00	00		00	00	00	00	1f	1f	1f	1f										
	00	00	00	00	21	00	00	00		00	00	00	00	21	21	21	21										

3.2.1 Construction of the Key Structure

Now, we describe how to construct the key structure, denoted by S_k . The key structure is generated from the second subkey K^1 . One can observe in Figure 3.1 that the difference of the second 256-bit subkey has 10 active bytes, but the difference cannot be chosen randomly. The differential characteristic of E_0 , as well as the one of E_1 , is constructed following the idea of local collision [40], that is, once a subkey difference is added to the internal state, the next subkey difference will try to cancel it in the next round. Therefore, in order to generate such a differential characteristic for E_0 , the two active bytes in the first half of ΔK^1 must take the same difference value, thus it can choose 2^8 values at most. Besides, for the second half of ΔK^1 , the differences of the two active columns are also required to be the same value, because the two active columns are supposed to cancel the two active columns in the internal state according to the differential characteristic of E_0 , and each

Figure 3.1: The differential characteristics of the related-key boomerang attack against AES-256.

active column in the internal state is computed from a single active byte through `MixColumns` and the two active bytes are equal. Furthermore, this relation also implies that the two active columns of ΔK^1 can only choose 2^8 values at most. To sum up, there are 2^{16} valid values for ΔK^1 , each are denoted by ΔK_i^1 , $1 \leq i \leq 2^{16}$.

For a secret key K_0 , the key structure S_k is generated by adding the non-zero difference ΔK_i^1 to the second subkey of K_0 , from which a new secret key K_i can be uniquely determined, see Figure 3.2a. Finally, the key structure consists of 2^{16} keys, from which 2^{31} unordered key pairs can be composed.

Note that the keys in the key structure are used in the encryption side of the boomerang attack. For the key K'_i used in the decryption side, they are computed by adding the fixed difference ∇K to the combination of the second half of K_i^3 and the first half of K_i^4 , then the full K'_i can be uniquely determined by the obtained eight consecutive columns, see Figure 3.2b. By doing so, the differential characteristic of E_1 will be fixed. The actual value of ∇K can be found at Table 3.2, it will make sure that the differential characteristic of E_1 is the optimal one *i.e.*, all S-box transitions happen with probability 2^{-6} .

Given a pair of keys (K_A, K_B) chosen from the key structure S_k , together with the corresponding key pair (K'_A, K'_B) used in the decryption side, the four keys form a key quartet. For a key quartet, the differences in the key schedule for both the differential characteristics of E_0 and E_1 can be found in Table 3.2. In particular, for the differential characteristic of E_1 (where the key pair (K_A, K'_A) or (K_B, K'_B) is applied), some byte in ∇K^i for $i = 1, 2, 3$ can even be determined due to the slow diffusion of the key schedule. These values will play an important role in the following key recovery attack. Last but not least, we note that only one key structure is used in our attack.

(a) Generating key structure S_k (b) Computing K'_i from K_i

Figure 3.2: Key generation.

3.2.2 Boomerang Distinguisher

Let us compute the probability of the boomerang distinguisher covering rounds 2–14. For the differential characteristic of E_0 which covers rounds 2–8, there are 5 active S-boxes and the differentials $(\Delta_{in}, \Delta_{out})$ for all of them are the same. Because of the use of key structure, the values for both Δ_{in} and Δ_{out} are not fixed, but they are directly related to the subkey differences, which are determined by the key pair used in the differential characteristic. Among the total of 2^{31} key pairs that can be composed from the key structure, $(2^7 - 2)/2^8$ of it will make the differential $(\Delta_{in}, \Delta_{out})$ happen with probability 2^{-7} , while a proportion of $1/2^8$ will lead the probability to 2^{-6} . Accordingly, the 5 active S-boxes in rounds 2–8 are passed with probability $2^{-7 \times 5} = 2^{-35}$ for the first case and $2^{-6 \times 5} = 2^{-30}$ for the second case.

The boomerang is switched in round 9. Although the differential characteristic of E_0 is not fixed, its truncated pattern is uniquely determined. Accordingly, we can ensure that there is no overlapped active S-box in round 9 between the differential characteristics of E_0 and E_1 . Thus, according to the BCT, the two differential characteristics are compatible for the boomerang attack and the switching probability is 1. Besides, due to the boomerang switching effect in multiple rounds discussed in Chapter 2, we also verified the switching effect in rounds 8–10, and it matches our evaluation.

For the differential characteristic of E_1 , there are 3 active S-boxes in rounds 10–14. Note that only one differential characteristic is used for E_1 and the differentials for all the active S-boxes are optimal with probability 2^{-6} , thus the probability of the differential characteristic of E_1 of round 10–14 is $2^{-6 \times 3} = 2^{-18}$.

Finally, the probability of the boomerang distinguisher is either $2^{2 \times (-35-18)} = 2^{-106}$ or $2^{2 \times (-30-18)} = 2^{-96}$, depending on the key pair from the key structure.

3.2.3 A Detailed Description of the Attack

One round is added at the beginning of the boomerang distinguisher to launch the full-round attack. The plaintext difference pattern, as show in Figure 3.1, is deduced from both the first subkey difference and the internal state difference in the second round. The attack procedure is described in Algorithm 2.

For each key pair, we can compose 2^{144} plaintext pairs from 1 plaintext structure, out of which $2^{144-72} = 2^{72}$ will pass through the first round with the desired input difference of the boomerang distinguisher. In total, 2^{103} pairs pass the first round for all the 2^{31} key pairs. The probability of the boomerang distinguisher is 2^{-106} for a proportion of $(2^7 - 2)/2^8$ key pairs, thus around $2^{103} \cdot (2^7 - 2)/2^8 \cdot 2^{-106} \approx 2^{-4}$ right quartets are expected. On the other hand, $2^{103} \cdot 1/2^8 \cdot 2^{-96} = 2^{-1}$ right quartets are expected when the probability of the boomerang distinguisher is 2^{-96} for $1/2^8$ of the key pairs. Compared to the first case, the boomerang distinguisher in the second case is much better, so we will only adopt the second one in our attack. Therefore, in order to obtain 4 right quartets, 2^3 plaintext structures are required. We

Algorithm 2: Related-key attack on AES-256 using key structure.

Prepare a plaintext structure consisting of 2^{72} plaintexts, which

traverses all values of the 9 gray cells in , and takes arbitrary constants in the others.

Create a hash table H of size 2^{88} .

for each of the 2^{16} keys K in the key structure S_K **do**

for each of the 2^{72} plaintexts P in the plaintext structure **do**

 Encrypt P under key K , denote the ciphertext as C .

 Compute $C' = C \oplus \Delta C$.

 Decrypt C' with K' , K' being computed from the corresponding K , and denote the new plaintext by P' .

 Insert the plaintext pair (P, P') into the hash table H , indexed by the 7 bytes of P' where constants of the plaintext structure fall and 2 bytes of $P \oplus P'$ at positions $(2, 0)$ and $(3, 0)$.

end

end

need to repeat Algorithm 2 2^3 times with different plaintext structures and the same key structure.

In the following, we will explain how to gradually filter out wrong quartets and recover key bits. Bytes of a subkey are denoted by $k_{i,j}^l$, l standing for the number of the subkey, and bytes of the plaintext being denoted by $p_{i,j}$. Let us compute the number of quartet candidates after Algorithm 2. Firstly, there is a 56-bit filter at the output of the boomerang. Then, observe in Table 3.2 that $\nabla k_{i,7}^0 = 0$ for $i > 1$, so $\Delta k_{i,0}^0$ should be equal for both pairs (K_A, K_B) and (K'_A, K'_B) in a key quartet, which implies that $\Delta p_{i,0}$ should be equal for both plaintext pairs (P_1, P_1) and (P'_1, P'_2) for the right quartet as well, because $\Delta k_{i,0}^0$ is equal to $\Delta p_{i,0}$ according to the differential characteristic of E_0 . This is a 16-bit filter. So there are on average $2^{72+16-56-16} = 2^{16}$ collisions for each index of the hash table H , from which 2^{31} quartets can be composed. In total, $2^{31+72} = 2^{103}$ quartet candidates are left for all the 2^{72} indexes of the hash table, and thus 2^{106} quartet candidates for all the 2^3 plaintext structures. These candidates are further filtered by the following steps. The key bytes that can be recovered are listed in Figure 3.3.

7							5
3	4	2	2	4D			7
D		4					5
D			4				5

Figure 3.3: The AES-256 key state with key information obtained at each step. Digits stand for the sub-steps in step II, “D” means difference.

Step I. Note that the key pairs of a right quartet must meet the requirement that the active S-boxes in the differential characteristic of E_0 are passed with probability 2^{-6} . The requirement is satisfied with probability 2^{-8} , thus $2^{106-8} = 2^{98}$ quartet candidates are eligible. Now we explain how to obtain the 2^{98} quartet candidates. The differential of the active S-box can be deduced by the difference in the key schedule. For example, $\Delta k_{1,1}^1$ is the input difference of the S-box at the position (1, 1) in the third round, and $\Delta k_{2,4}^1$ is the corresponding output difference due to the **ShiftRows** and **MixColumns**. Thus, we can simply check whether the differential $(\Delta k_{1,1}^1, \Delta k_{2,4}^1)$ is optimal for the AES S-box. Once it is confirmed, the differentials of the 5 active S-boxes will also be determined. Note that the differential characteristics of E_0 used in both sides of the boomerang are the same, thus we only need to check the encryption side, the details are given in Algorithm 3.

Algorithm 3: Filtration in Step I.

```

for each index of the hash table  $H$  do
    Insert the  $2^{16}$  collisions into a new hash table  $H'$  indexed by the
    difference of  $k_{1,1}^1$  between the current key and the original key
     $K_0$ .
    for each index  $i$  of  $H'$  do
        Insert the  $2^{16-8} = 2^8$  collisions (on average) into a new hash
        table  $H''_i$  indexed by the difference of  $k_{2,4}^1$  between the
        current key and the original key  $K_0$ .
    end
    for index  $i$  of  $H'$  from 0 to  $2^8 - 2$  do
        for index  $j$  of  $H'$  from  $i+1$  to  $2^8 - 1$  do
            Compute  $\Delta k_{1,1}^1 = i \oplus j$ , and find the value of  $\Delta k_{2,4}^1$  such
            that  $\text{DDT}(\Delta k_{1,1}^1, \Delta k_{2,4}^1) = 4$ .
            for each index  $s$  of  $H''_i$  do
                Compute  $t = s \oplus \Delta k_{2,4}^1$ . Check whether  $t$  is in  $H''_j$ . If
                yes, the pairs of  $H''_i$  and  $H''_j$  compose quartet
                candidates.
            end
        end
    end
end

```

Step II. There are $2^{31-8} = 2^{23}$ key quartets remaining after Step I, and each has on average $2^{98-23} = 2^{75}$ quartet candidates. In the following steps, we will proceed with each key quartet independently, and use K_A , K_B , K'_A and K'_B to denote the four keys.

- 1) There is a 2-bit filter at $\Delta p_{1,2}$ and $\Delta p_{1,3}$ due to the S-box compatibility, thus 4-bit at both sides of the boomerang in total. Besides, there is also a 2-bit filter at $\Delta p_{2,0}$ and $\Delta p_{3,0}$ due to the S-box compatibility in the key schedule. Thus, the number of quartets is reduced to $2^{75-6} = 2^{69}$.

- 2) Each quartet proposes 2^2 candidates of $k_{1,2}^0$ and $k_{1,3}^0$ for K_A and K'_A each, thus there are in total 2^4 candidates. As can be seen from Table 3.2, the four “ X ” of ∇K^0 are equal and take only 2^7 values. Hence, the differences $\Delta k_{1,2}^0$ and $\Delta k_{1,3}^0$ between K_A and K'_A have to be equal to X , which is a 16-bit filter for the key candidates. Thus, $2^{69+4+7-16} = 2^{64}$ quartet candidates are left, and the values of $k_{1,2}^0$ and $k_{1,3}^0$ are suggested.
- 3) Observe that the value of $\Delta k_{1,0}^0$ is determined by $k_{2,7}^0$ and $\nabla k_{2,7}^0 = 0$ from Table 3.2, the values of $\Delta k_{1,0}^0$ should be the same for both the key pairs (K_A, K_B) and (K'_A, K'_B) . Since $\Delta k_{2,7}^0$ is known, $\Delta k_{1,0}^0$ can take 2^7 values. For each guess of $\Delta k_{1,0}^0$, it has to be compatible with $\Delta p_{1,0}$ and $\Delta y_{1,0}^0$ through the S-box, which is a 2-bit filter for both sides of the boomerang. After that, each quartet proposes two candidates of $k_{1,0}^0$ for K_A and K'_A , respectively. Moreover, there is an 8-bit filter because the difference $\nabla k_{1,0}^0$ between K_A and K'_A should be equal to X . In the end, the number of quartet candidates is reduced to $2^{64+7-2+2-8} = 2^{63}$, and the value of $k_{1,0}^0$ is suggested.
- 4) Notice that $\nabla k_{1,3}^1 = 0$, a reasoning similar to the one above can be applied to $\Delta k_{1,4}^0$, which can take 2^7 values. For each guess of $\Delta k_{1,4}^0$, the values of $\Delta y_{0,0}^0, \Delta y_{1,1}^0, \Delta y_{2,2}^0, \Delta y_{3,3}^0$ can be uniquely computed by inverting the MixColumns transformation. There is a 1-bit filter on $\Delta y_{1,1}^0, \Delta y_{2,2}^0, \Delta y_{3,3}^0$ each due to the S-box compatibility, then 6-bit filter in total on both sides of the boomerang. Each quartet proposes two candidates of $k_{1,1}^0, k_{2,2}^0, k_{3,3}^0$ for K_A and K'_A , respectively. However, the difference $\nabla k_{1,1}^0$ between K_A and K'_A is restricted to X , which results in an 8-bit filter. To summarize, the number of quartets is reduced to $2^{63+7-6+6-8} = 2^{62}$ and the values of $k_{1,1}^0, k_{2,2}^0, k_{3,3}^0$ as well as $\Delta k_{1,4}^0$ are suggested.
- 5) Since $\Delta k_{1,0}^0, \Delta k_{2,0}^0$ and $\Delta k_{3,0}^0$ are known, it will provide 2 guesses for each of $k_{2,7}^0, k_{3,7}^0$ and $k_{0,7}^0$. However, these guesses for K_A and K'_A are the same because the differences $\nabla k_{2,7}^0, \nabla k_{3,7}^0$ and $\nabla k_{0,7}^0$ between K_A and K'_A are all 0. Thus, in this step, the number of key candidates is increased to $2^{62+3} = 2^{65}$, and the values of $k_{2,7}^0, k_{3,7}^0$ and $k_{0,7}^0$ are suggested.
- 6) Note that the key bytes $k_{2,7}^0, k_{1,0}^0, k_{1,1}^0, k_{1,2}^0, k_{1,3}^0$ and the difference $\Delta k_{1,4}^0$ have been derived in the above steps. On the other hand, we notice that $\Delta k_{1,4}^0$ can be computed from $k_{2,7}^0, k_{1,0}^0, k_{1,1}^0, k_{1,2}^0$ and $k_{1,3}^0$ according to the key schedule. This constraint can provide an 8-bit filter, and thus the number of key proposals is reduced to 2^{57} .
- 7) Make a guess of $k_{1,7}^0$ of K_A , which has 2^8 choices, then $k_{1,7}^0$ will be known for all the four keys and $\Delta k_{0,0}^0$ can be computed. After that, for each side of the boomerang there is a 1-bit filter on $\Delta k_{0,0}^0$ due to the S-box compatibility, then each quartet will propose 2 candidates of $k_{0,0}^0$

for K_A and K'_A , respectively. Thus $2^{57+8-2+2} = 2^{65}$ key proposals are obtained.

In the end, 2^{65} key candidates are proposed and 11 key bytes for each of K_A , K_B , K'_A and K'_B are suggested. However, many bytes are strongly related according to Table 3.2. Among them, at least $k_{0,0}^0$, $k_{1,1}^0$, $k_{2,2}^0$ and $k_{3,3}^0$ of K_A and K'_A are independent, so we can recover 15 bytes with 2^{65} proposals for each key quartet, and thus 2^{88} proposals for all the 2^{23} key quartets. Additionally, differences of 3 bytes are recovered: $\Delta k_{1,4}^0, \Delta k_{2,0}^0, \Delta k_{3,0}^0$, where $\Delta k_{2,0}^0$ and $\Delta k_{3,0}^0$ can be directly obtained from plaintext difference, but they can not be used to derive the corresponding key bytes.

Recover the Key. Recall that 4 right quartets are expected for the attack, which are supposed to be distinguishable from other wrong quartets. However, the 4 right quartets are very likely to be combined with different key pairs due to the key structure, thus the correct key bytes proposed by them will belong to different keys, which is hard to be distinguished. So we have to deduce all the proposed key bytes to the original keys K_0 and K'_0 . Looking at Table 3.2 we can see that most bytes of the first subkey difference between K_0 and all the other keys are known, except $\Delta k_{1,4}^0$ and $\Delta k_{i,0}^0$ where $0 \leq i \leq 3$. Hence, for those key bytes whose differences are known, all the proposals can be deduced to K_0 . The same reasoning applies K'_0 . As for the two unknown differences $\Delta k_{0,0}^0$ and $\Delta k_{1,0}^0$, since the value of $k_{1,7}^0$ was derived and the difference of $\Delta k_{1,7}^0$ is known, the difference $\Delta k_{0,0}^0$ can be easily computed through the key schedule. Same trick holds for $\Delta k_{1,0}^0$, it can be computed from $k_{2,7}^0$ and $\Delta k_{2,7}^0$. Then, we can deduce the proposals of the two bytes to K_0 and K'_0 .

In the end, all the proposed key bytes can be deduced to K_0 and K'_0 . We have 2^{88} proposals for 120 key bits, and the correct proposal is supposed to appear 4 times. The probability that a wrong key is suggested 4 times is $\binom{2^{88}}{4} \cdot (2^{-120})^4 \cdot (1 - 2^{-120})^{2^{88}-4} \approx 2^{-138.5}$, thus the expect number of such a key is $2^{120-138.5} = 2^{-18.5}$, while the 4 right quartets would always vote for the correct one. Therefore, no wrong key will survive and the correct 120 key bits will be recovered. With the knowledge of the recovered key bits, the remaining part of the key can be found with many approaches, which will not dominate the cost of the whole attack.

Complexity. In our attack, a total of 2^{92} plaintexts and ciphertexts are generated, thus the data complexity is 2^{92} . In Algorithm 2, there are 2^{88} encryption oracle calls, 2^{88} XOR operations, and 2^{88} decryption oracle calls. As the plaintexts are added into the hash table, each plaintext requires one memory access, thus 2^{88} memory accesses in total. Thus, for the 2^3 plaintext structures, the total time complexity of Algorithm 2 is 2^{92} encryption/decryption oracle calls and 2^{91} memory accesses.

In Algorithm 3, the plaintext pairs (P, P') in H are added into many new hash tables, which requires $2^{72} \times (2^{16} + 2^{16}) = 2^{89}$ memory accesses. The

lookups in DDT require $2^{72+15} = 2^{87}$ memory accesses. The lookups in the hash table H_j'' require $2^{72+23} = 2^{95}$ memory accesses, which dominates the algorithm. In sum, for the entire 2^3 plaintext structures, Step I requires $2^{95+3} = 2^{98}$ memory accesses.

For Step II 1), it requires 1 memory access for each quartet to check whether these bytes are compatible. Therefore, the time complexity of this step is 2^{98} memory accesses.

After Step II 1), the number of remaining quartets is 2^{92} , and the number is continuously decreasing in the following steps, thus the following computation will not dominate the cost of the whole attack.

Following the idea from [27, 31, 58], where memory access can be converted to equivalent amount of encryption/decryption oracle calls, one AES-256 encryption/decryption is equivalent to roughly 2^8 memory accesses by counting the number of S-box lookups. We conclude that the whole attack requires 2^{92} plaintexts and ciphertexts, the time complexity being equivalent to $2^{92.5}$ encryptions (2^{92} encryptions/decryptions and 2^{99} memory accesses $\approx 2^{91}$ encryptions), and the memory complexity is 2^{89} .

Further Improvement. In the attack, the key structure is only added in K_A and K_B in E_0 , the same idea could be extended to the K'_A and K'_B in E_1 . However, this attempt could not work. For a right quartet, the differentials of E_0 in both sides of the boomerang should be the same, which determines that the differential characteristic in the key schedule of (K'_A, K'_B) must be the same as that of (K_A, K_B) . In order to meet this condition, the switch in the key schedule should be the same for a quartet, *i.e.*, the differences ∇K added to K_A and K_B at the middle of the fourth and fifth subkey should be the same. Even if we use a structure of ∇K , we still need to find right quartets for each value of ∇K separately. Thus, this method does not provide additional improvement to the attack.

Although the potential key structure is a failure, it can also be used to improve the attack slightly. As discussed above, the functionality of the key structure is equivalent to plaintext structure: using a different ∇K leads to a different set of (P'_1, P'_2) . In our attack, 2^3 plaintext structures are required to produce 4 right quartets. Instead of choosing 2^3 plaintext structures, we could also choose $2^3 \nabla K$ (There are $2^8 - 1$ values of ∇K to produce the optimal differential characteristic of E_1). In this way, 2^{88} encryptions and $2^{88+3} = 2^{91}$ decryptions are needed, thus the number of decryptions dominates the complexity. The time and data complexity is reduced to 2^{92} and 2^{91} , respectively. The number of keys will increase to $2^{16+3} = 2^{19}$.

More Trade-offs. One key structure consists of 2^{16} , and hence the total number of keys required in this attack is 2^{17} (2^{16} for the encryption and decryption oracles each). There is a trade-off between the number of keys required, and the time and data complexities, by reducing the size of the key structure, *i.e.*, with a 2^{16-s} key structures, the resulted complexities of the attack will be: Time 2^{92+s} , Data 2^{91+s} , and # Keys 2^{17-s} , for $0 \leq s \leq 16$.

Further to note, when $s \geq 7.5$ the time complexity becomes higher than that in [32], and our attack offers no more advantage, so the trade-off makes sense only for $0 \leq s \leq 7.5$.

3.3 Conclusion

In this chapter, we brought the idea of structures to key materials, and successfully applied it to the related-key boomerang attack against AES-256. This improved the best known attack against AES-256 by reducing the data/time complexities by a factor of about 2^8 , at the cost of more required keys. While the general principle is simple, its deployment contains many details and it is important to ensure that the introduction of key structure will not invalidate or significantly reduce the probability of the differential characteristics. More trade-offs are provided between time/data complexity and the number of required keys.

Other Potential Applications. We note that our structure technique was applied to key material, and hence increases the number of required keys for the attack to succeed. However, this may be avoided when the attack is applied to AES-based tweakable block ciphers so that the structure is applied to tweak input, rather than keys. There are two such cases: TAES [15] and the TBCs following TWEAKEY framework [70]. TAES is basically AES-256 with the concatenation of a 128-bit secret key and a 128-bit tweak as the 256-bit key input. For the TWEAKEY framework, users will have the choice to decide which are the bits to be used as key or tweak material. The potential application of our technique is that, when the structure is applied to the tweak of either TAES or AES-based TWEAKEY designs, the increased requirement applies to the tweak only, and that of keys remains un-affected.

Inapplicability to AES-192 [32] and Differential Attack [33]. The boomerang attack was applied AES-192 as well in [32], so the idea of key structure naturally applies. However, looking into the details, the key bytes recovered in the AES-192 attack falls in two different locations, in both the pre- and post-whitening keys. Note that in our improved attack on AES-256, we are able to deduce the count of key suggestions of all keys to the original key K_0 and K'_0 , however this becomes impossible for both pre- and post-whitening keys simultaneously in case of AES-192. The direct application of the idea of key structure to the differential attack in [33] seems difficult, as the probability of the differentials will drop significantly, which overrules the potential gain key structures might bring. It will be interesting to see if these technical difficulties could be overcome and find more applications of key structures.

Chapter 4

Preimage Attacks on AES Hashing Modes

As mentioned in the introduction of this thesis, the Matyas-Meyer-Oseas (MMO) mode can be used to build an n -bit compression function out of an n -bit block cipher. This is one of the ISO/IEC standardized method to do so. A hash function can then be defined from that compression function with an operating mode such as the famous Merkle-Damgård construction [47, 85]. In [66], such a block cipher-based compression function (the round function in Hash function 1) is instantiated using AES as an example. Besides, the instantiation of Davis-Meyer (DM) mode with AES was once submitted to the NIST as a formal proposal under the name AES-Hash [43]. Furthermore, there are cryptographic protocols with rich functionality using hash functions as building blocks, e.g., Symmetric Searchable Encryption (SSE), which is designed to achieve the best trade-offs between security, efficiency, and functionality. In [36, 37], secure SSE schemes are proposed, using hash functions requiring preimage resistance. Besides, in the implementation of the proposed SSE scheme named Diana, the hash function is the AES block cipher used in the MMO mode [37].¹ Due to the standardization efforts, the well-studied security of PGV modes, and the AES-NI instructions in modern CPUs, hash functions built based on AES PGV modes may have also attracted other real-life cryptographic applications that require loose but finely controllable security. A hash function with a 128-bit internal state has potential applications in cases where it requires (second-)preimage resistance but explicitly does not require collision resistance (e.g., hashing with short-input for signature schemes [76]).

As explained in introduction, a preimage attack against a hash function tries to find a message whose digest equals to a certain challenge value. The most naive way to find a preimage is to randomly select a message, evaluate

¹Corresponding open-source libraries are released, and one can find the implementation of the hash function built using AES MMO-mode via https://gitlab.com/sse/crypto/blob/master/src/block_hash.cpp, and https://github.com/jgharehchamani/SSE/blob/master/third_party/crypto/src/block_hash.cpp. We found that in the newest repository moved to GitHub, the hash function used has been replaced with BLAKE2b and SHA-512, see <https://github.com/OpenSSE/crypto-tk/blob/master/src/hash.cpp>.

its digest and check against the challenge value. This brute-force method costs 2^n hash evaluations for an n -bit hash function. A method running faster than 2^n is considered as an attack. Sasaki and Aoki introduced the Meet-In-The-Middle (MITM) preimage attack in 2008 [105], and the technique was extended and used to break the theoretical preimage security claims of MD4 [61], MD5 [106], Tiger [61, 110], HAVAL [63, 105] and round-reduced variants of many other hash functions such as SHA-0 and SHA-1 [9, 57, 75], SHA-2 [8], BLAKE [57], HAS-160 [64], RIPEMD and RIPEMD-160 [111], Stribog [4], and Whirlwind [5]. It is interesting to see that the idea of MITM preimage attacks also leads to progress for collision attacks against reduced SHA-2 [80]. These preimage attacks are generally converted from a MITM pseudo-preimage attacks by using a generic algorithm, where pseudo-preimage attacks are attacks with unconstrained IVs (one allows free choice of IVs). Thus, formally, denote the hash function by H , for pseudo-preimage attacks, given a target T , the goal is to find an initialization vector or chaining value V and a message M such that $H(V, M) = T$.

AES hashing modes refer to the compression function instantiated by the AES block cipher. In 2011, Sasaki [104] introduced the first attack against AES hashing modes with the underlying AES reduced to 7 rounds and the last round without the MixColumns operation. The complexity of the attack was slightly improved by Wu *et al.* [114] in 2012. To the best of our knowledge, there were no more public progress on this topic since then.

The general idea of MITM preimage attack is to split the cipher (or compression function) into two independent chunks, which can be computed independently from each other with respect to some *neutral bits*. Technically, the source of neutral bits of most previous works is the key bits of the block cipher or the message bits of the compression function. However, in [104], the neutral bits are chosen from the state while the key bits are not used and fixed to some random constants. The key bits are not used because finding neutral bits in the key is difficult due to the key schedule which diffuses all key bits quickly. It is then natural to ask whether it is possible to eventually find neutral bits from the key to either improve the attack complexity or to extend the number of attacked rounds. In this chapter, we achieve both.

Contributions of this Chapter. On one hand, additional neutral bits from the key improves the attack in two directions, *i.e.*, improving the time complexity directly due to more neutral bits and extending the attack to more rounds since local collisions of neutral bits from the encryption state and the key state are possible. On the other hand, to avoid dealing with the quick diffusion of the key schedule of AES, we choose neutral bits from the key state for one chunk v.s. for both the forward and backward chunks. This is possible thanks to the improvement of the attack in [114], where the unbalanced 8 and 32 bits neutral bits are found for the two chunks. The additional neutral bits from key make it closer to the balanced 32 and 32 bits, which improves the time complexity of the final attack by a factor of at most $2^{32-8} = 2^{24}$.

Target	#Rounds	Time-1	Time-2	Memory	(d_1, d_2, m)	Source
AES-128	7	2^{120}	$2^{125} *$	2^8	$(8, 8, 32)$	[104]
	7	$2^{120-\min(t,24)}$	$2^{123} *$	$2^{8+\min(t,24)}$	$(8, 32, 32)$	[114]
	7	2^{104}	2^{117}	2^{24}	$(24, 32, 24)$	Section 4.3.1
AES-192	7	2^{120}	$2^{125} *$	2^8	$(8, 8, 32)$	[104]
	7	2^{96}	2^{113}	2^{32}	$(32, 32, 32)$	Section 4.3.2
	8	$2^{112-\min(t,16)}$	2^{116}	$2^{16+\min(t,16)}$	$(16, 32, 32)$	Section 4.4.3
AES-256	7	2^{120}	$2^{125} *$	2^8	$(8, 8, 32)$	[104]
	8	2^{96}	2^{113}	2^{32}	$(32, 32, 32)$	Section 4.4.2
Kiasu-BC	7	$2^{104-\min(t,8)}$	2^{117}	$2^{24+\min(t,8)}$	$(24, 32, 32)$	Section 4.3.3
	8	$2^{120-\min(t,24)}$	2^{123}	$2^{8+\min(t,24)}$	$(8, 32, 32)$	Section 4.4.4

Table 4.1: Summary of our improved pseudo-preimage attacks against the round-reduced compression function of AES hashing modes, compared with previous works.

- Time-1 is the time complexity of pseudo-preimage. Here, 2^t is the number of available targets for multi-target pseudo-preimage attacks. In the full (second-)preimage attacks, that can be the number of available nodes in the unbalanced-tree-based conversion from multi-target pseudo-preimage to preimage attacks (refer to Section 4.1.2), or the number of blocks of given message for second-preimage attacks.
- Time-2 is the complexity of using the (multi-target) pseudo-preimage attacks to do (second-)preimage attacks when requiring an upper layer of meet-in-the-middle procedure of conversion (refer to Section 4.1.2) for some PGV modes, such as DM-mode, and here a single target is given.
- * Note that for some PGV modes, such as MMO and MP modes, converting the pseudo-preimage attacks in [104, 114] to second-preimage attacks will not require the upper-layer meet-in-the-middle procedure, and the time complexities will be the same as that of the pseudo-preimage attacks (refer to [104] for details). Whereas, the presented attacks in this chapter always require an upper-layer meet-in-the-middle procedure to be converted into (second-)preimage attacks, because they require that both inputs (to the encryption and to the key schedule) to be unfixed.

Larger key sizes allow more degrees of freedom for the choices of neutral bits, and also AES with a larger key size comes with a slower key diffusion. These factors lead us to a higher number of attacked rounds and lower time complexities for AES-192 and AES-256, compared with the previous attacks against AES-128 in [104, 114]. The details of the attacks, including the number of attacked rounds and time/memory complexities, compared with previous works, are summarized in Table 4.1. This work was published at [14].

Organization. The rest of the chapter is organized as follows. Section 4.1 gives a general description of the MITM preimage attack. The main techniques of MITM attack on AES hashing modes are summarized in Section 4.2,

including those used in previous attacks and in our attacks. Results of 7 and 8 rounds are given in Section 4.3 and 4.4, respectively. The applicability of the presented pseudo-preimage attacks on the PGV modes and on conversion into full (second-)preimage attacks are discussed in Section 4.5. Section 4.6 concludes the chapter.

4.1 The MITM Preimage Attack

4.1.1 Application to Pseudo-Preimage Attacks

Using the MITM approach in preimage finding on hash function can be seen in [11, 65, 79]. Aoki and Sasaki combined in [105] for the first time the MITM and local-collision approaches to devise preimage attacks on hash function HAVAL. Before that, local-collision approach is mainly used in collision attacks on hash functions. Based on these primary works, the MITM-based preimage attack on hash functions developed in a series of papers and advanced further.

Techniques Developed for MITM Preimage Attacks

Techniques invented along the development of the attacks are as follows.

Splice-and-Cut and Neutral Words. Aoki and Sasaki in [10] invented the splice-and-cut MITM attack and proposed the concept of “neutral word”. In the splice-and-cut MITM attack as depicted in Figure 4.1, the first and last steps of the attack target can be spliced to be consecutive steps by feed-forward mechanisms in the compression function of hashing modes (*e.g.*, DM-mode) or by querying the decryption in encryption schemes. The chain of computational steps of the attack target is cut starting from an internal step (named starting point), such that the chain is divided into two chunks of steps. Conventionally, the chunk of steps, which need to be computed forward (resp. backward) to reach the matching point, is named forward chunk (resp. backward chunk). The starting point is chosen so that each chunk includes at least one message (or key) word that is independent of the other chunk, where such message (or key) words are called “neutral words”.

Initial-Structure Technique [106]. Initial-structure is a generalization of the local-collision technique that enables one to skip several steps at the beginning of chunks. Initial structure includes at least two special defined neutral words m^{1st} and m^{2nd} which could be used to divide the operations of a few steps such that the operations on m^{1st} before the initial structure are independent from m^{2nd} and the operations on m^{2nd} after the initial structure are independent from m^{1st} .

Figure 4.1: The advanced MITM pseudo-preimage attack on DM-mode [104]. Let the space for both neutral words m_a and m_b be 2^ℓ , the time complexity is $2^{n-\ell}$, and memory complexity is 2^ℓ .

Partial Matching [10, 106]. In the primary MITM approach, the final phase of the attack involves full-state matching. Whereas, by executing only one word (or several words) matching instead of all words matching, the required independent computations can be expanded by more steps, thus enables to attack more steps of the target.

Partial-Fixing [10, 106]. Fixing a part of the neutral words (*e.g.*, lower x bits of a neutral word) enables one to partially compute more steps within a chunk even if a neutral word for the other chunk is involved.

Multi-Target [61]. When incorporating multi-target scenarios into the MITM framework, the multiple available targets can directly provide additional freedoms to one computation chunk without influencing the other.

The Attack Framework

As depicted in Figure 4.1, the attack framework of the splice-and-cut MITM attack using the initial structure and the partial match is as follows. Before the execution of the attack procedure, the configuration of the attack should be set up, which involves:

1. **Chunk separation:** by splicing and cutting, decide which part of the cipher will be the starting point of the forward/backward computation, and at which internal state will be the matching point. The principle of the chunk separation is to find the best balance between freedom degrees and the size of the matching point so that the freedom degrees are fully used. That requires one to decide:
2. **The neutral bytes for each chunk** – the selection on the neutral bytes will determine the freedom degrees.

3. The bytes for matching – the derivation on the bytes for match also depends on the selection of neutral bytes and the computation rule of the attack target.

With the above configurations decided, the attack procedure goes as follows (Figure 4.1 illustrates the MITM pseudo-preimage attack integrating these advanced techniques on Davies-Meyer mode). Denote the neutral words for the forward chunk and backward chunk by N^f and N^b , respectively:

1. Fix all other words except the neutral words N^f and N^b in the initial structure to arbitrary values.
2. For all possible values of N^f , forward compute from the starting point to the matching point at the final state of the forward chunk to get a list L^f of candidate values indexed by the value of N^f .
3. For all possible values of N^b , backward compute from the starting point to the matching point at the final state of the backward chunk to get a list L^b of candidate values indexed by the value of N^b .
4. Sorting the two lists L^f and L^b using hash tables, check whether there is a match/partial-match between them.
5. In case of partial-matching used in the above step, for the surviving pairs, check for a full match.
6. Repeat the whole procedure to find full-state matches by changing the values of fixed words.

The Complexity Analysis

Denote the size of the internal state by n , the freedom degrees in the forward and backward directions by d_1 and d_2 respectively, and the number of bits for the match by m .

1. Forward computing to get a list L^f of size 2^{d_1} requires 2^{d_1} computations of the forward chunk.
2. Backward computing to get a list L^b of size 2^{d_2} requires 2^{d_2} computations of the backward chunk.
3. Matching between L^f and L^b requires $2^{\max(d_1, d_2)}$ memory accesses which are usually ignored, compared to the $2^{\max(d_1, d_2)}$ computations of the entire compression function of the target in above steps.
4. $2^{d_1+d_2-m}$ pairs are left after partial matching, hence the same complexity is required for full-match checking.
5. Finding a full match requires $2^{m-(d_1+d_2)} \times 2^{n-m} = 2^{n-(d_1+d_2)}$ repetitions.

When d_1, d_2 are different, *i.e.*, unbalanced, we use $\max(2^{d_1}, 2^{d_2})$ to denote the sum of complexities for computing the two chunks. Note, when $d_1 = d_2$, the computation complexity is 2^{d_1} full targets (= forward chunk + backward chunk). Hence, $\max(2^{d_1}, 2^{d_2})$ is used for all cases. The above complexity analysis gives

$$2^{n-(d_1+d_2)} \cdot (2^{\max(d_1, d_2)} + 2^{d_1+d_2-m}) = 2^{n-\min(d_1, d_2)} + 2^{n-m} \simeq 2^{n-\min(d_1, d_2, m)}. \quad (4.1)$$

From this formula, it can be seen that the critical point for the attack being optimized is to reach a balance between the freedom degrees for the forward chunk and the backward chunk. Because in the end it will only depend on the minimum between the two freedom degrees. Besides, the memory complexity can be $2^{\min(d_1, d_2)}$ by only storing the list computed by the direction with lesser freedom degrees and make a match once a candidate for the other direction is available. Accordingly, the larger the $\min(d_1, d_2)$ the lesser the time complexity and at the same time the larger the memory complexity. However, generally, in such MITM preimage attack on hash function, the $\min(d_1, d_2)$ is smaller than $n/2$, thus, compared with memory complexity, time complexity is the bottleneck. Therefore, generally, the larger the $\min(d_1, d_2)$ the better the attack. The second term 2^{n-m} is usually minor when $m > \min(d_1, d_2)$, *i.e.*, the number of matching bits is more than that of neutral bits.

4.1.2 Conversion from Pseudo-Preimages to Preimages

Note that the above MITM attack only gives pseudo-preimage for the compression function. Below we consider the methods to convert pseudo-preimages on the compression function to preimages on the hash function.

Converting Pseudo-Preimages to A Preimage [84, Fact 9.99]. For an n -bit narrow-pipe (*i.e.*, the chaining variables are n -bit for an n -bit hash function) iterated hash function, using the unbalanced MITM approach, a pseudo-preimage attack with a computational complexity of 2^ℓ ($\ell < n - 2$) can be converted into a preimage attack with computational complexity of $2^{(n+\ell)/2+1}$. That is done by first finding $2^{(n-\ell)/2}$ pseudo-preimages (mapping $2^{(n-\ell)/2}$ random starting chaining values to the given target T), and then starting from the real initialization value vector IV of the hash function, using $2^{(n+\ell)/2+1}$ random message blocks to get $2^{(n+\ell)/2+1}$ random chaining values. Among the $2^{(n+\ell)/2+1}$ chaining values mapped from the real IV and the $2^{(n-\ell)/2}$ starting chaining values mapped by the pseudo-preimages to the given target T , we can expect to find one match due to the birthday paradox, which leads a valid preimage. Figure 4.2a visualizes the above generic MITM procedure.

An improvement over the above MITM procedure of converting pseudo-preimage attacks to preimage attacks was made possible by Leurent in [79], where he proposed to use multi-target pseudo-preimage attacks which form

an unbalanced tree, and require the assistance of the expandable message technique to overcome the length padding (see Figure 4.2b). The main idea is as follows. Like in the above MITM procedure, one has to first launch many procedures of pseudo-preimage attack. However, instead of using the starting chaining values of the pseudo-preimage attacks until the matching phase, those starting chaining values determined by previous procedures of pseudo-preimage attack are immediately used as targets for later procedures of pseudo-preimage attack. Suppose we have already launched i procedures of pseudo-preimage attack, there are then i targets for the $(i + 1)$ -th procedure of pseudo-preimage attack. Given i targets, the complexity of a pseudo-preimage attack is possible to be reduced from 2^ℓ to $2^\ell/i$.² As a result, to obtain 2^k starting chaining values in total by launching 2^k procedures of pseudo-preimage attack, it takes $2^\ell/1 + 2^\ell/2 + 2^\ell/3 + \dots + 2^\ell/2^k \simeq k \ln(2) \cdot 2^\ell$. Then, by generating 2^{n-k} chaining values mapped by the expandable messages from the real IV , one match is expected, which leads to a real preimage. Taking the optimal $k = n - \ell$, the overall time complexity of this improved method is

$$((n - \ell) \cdot \ln(2) + 1) \cdot 2^\ell = (\min(d_1, d_2) \cdot \ln(2) + 1) \cdot 2^{n-\min(d_1, d_2)}, \quad (4.2)$$

where a pseudo-preimage attack costs 2^ℓ computations given a single target ($\ell = n - \min(d_1, d_2)$ in our notation), and $2^{\ell-t}$ computations given 2^t targets.

The assumption that 2^t -target pseudo-preimage attack costs $2^{\ell-t}$ computations is possible in our attacks when the target can be used as the additional source of neutral words. Without loss of generality, let us assume targets can be used as part of the forward chunk, hence the size of L^f list is increased from 2^{d_1} to 2^{d_1+t} , then Equation (4.1) gives complexities

$$\text{Time: } 2^{n-\min(d_1+t, d_2, m)}; \text{ Memory: } 2^{\min(d_1+t, d_2, m)}. \quad (4.3)$$

In other words, the complexity of pseudo-preimage attack reduces linearly with respect to the number of targets available, as long as $d_1 + t < d_2$ and $m > \min(d_1 + t, d_2)$.

When these conditions could not be met, the complexity of the pseudo-preimage attack remains $2^{n-\min(d_1, d_2)} + 2^{n-m} \simeq 2^{n-\min(d_1, d_2, m)}$ for $d_1 + t \geq d_2$ (i.e., $t \geq d_2 - d_1$), for which the time complexity of preimage conversion will follow the original unbalanced MITM

$$2^{(n+\ell)/2+1} \simeq 2^{n+1-\min(d_1, d_2, m)/2}. \quad (4.4)$$

4.1.3 Converting Block cipher to Compression Function

In the original pseudo-preimage attack on AES hashing mode by Sasaki [104], the key is preset to random constants, which can be used by the attacker, and

²In our attacks, this is possible when there is a gap i between the freedom provided by neutral words of the forward computation and that of the backward computation, which can be filled by exploiting freedom provided by the multiple targets, as will be shown later.

Figure 4.2: Converting pseudo-preimages to preimages: circle denotes state, arrow denotes message block [61].

the input state is determined by the attack, hence such pseudo-preimage can be converted into preimages for modes in Class 1 (defined in Section 1.2.1) such as MMO and MP with the same complexity as pseudo-preimage. Other modes in Class 2 and 3 require pseudo-preimage to preimage conversion, and hence result in higher complexities for preimages as discussed in the subsection above.

Converting TBC to Compression Function. As we will discuss the TBC Kiasu-BC in this chapter, we consider here converting a TBC to a block cipher, then to a compression function through the PGV modes. The tweakable block cipher, denoted here as $\tilde{E}(K, T, P)$, has an additional input tweak T , compared with the block cipher $E(K, P)$. The TBC-to-BC conversion can be divided into 3 types:

- Type-I: $E(K, P) = \tilde{E}(K = K, T = C, P)$, where C is a constant.
- Type-II: $E(K, P) = \tilde{E}(K = C, T = K, P)$, where C is a constant.
- Type-III: $E(K, P) = \tilde{E}(K = K_1, T = K_2, P)$, where $K = K_1 || K_2$, *i.e.*, both key and tweak of $\tilde{E}$ are used as the key of the block cipher.

From the cryptanalyst's point of view, Type-III gives additional input to the attacker; hence it is likely more rounds could be attacked for Type-III, compared with the other two types.

4.2 Techniques of MITM Attack on AES Hashing Modes

4.2.1 The 7-Round Attack by Sasaki and Its Improvement by Wu *et al.*

Following the framework of splice-and-cut MITM attack depicted in Figure 4.1, and based on an important observation on the slow diffusion of a 4-round AES with the omission of MixColumns in the second round, Sasaki [104] invented a MITM preimage attack on a AES hashing mode. In [104], a basic splice-and-cut MITM attack on 4-round AES (with the omission of

MixColumns in the last round as in the full version of AES) hashing mode was first proposed, which was then extended to 7-round using two main techniques that will be introduced next. In the basic 4-round attack, the computation flow is spliced using the feed-forward mechanism in the hashing mode, and it is cut exactly in the middle (at the beginning of the third round). In doing this, the round missing **MixColumns** becomes the second round viewing from the starting point at where the computation was cut. Due to the slow diffusion observed, importing two neutral bytes at the starting point, the influence of a neutral byte penetrating forwardly and undisturbedly through 3-round (across the sliced point) and the influence of a neutral byte penetrating backwardly and undisturbedly through 1 round can be matched at the end of the first round.

Different from previous MITM preimage attacks on hash functions such as MD5 and Tiger, the neutral words in the attack on the AES hashing mode in [104] are not chosen from the message. Instead, they are chosen from the internal state values of the compression function. This choice is mainly due to the fast diffusion of the AES key schedule, *i.e.*, the key schedule quickly diffuses any choice of a key byte (as the neutral word) to all other key bytes in just a few rounds. As a result, in [104] the key-input/message-input is fixed to arbitrary constant values. Although the number of rounds of AES is much smaller than that of many dedicated hash functions such as MD5 and SHA-1, the round function is relatively heavier and the computation has faster diffusion, which prevents the MITM attacks from penetrating many rounds. Thus, more techniques were invented in [104] for attacking more rounds (up to 7 rounds, yet the results are still not for the full rounds, and currently leaves comfortable margins).

There are two main techniques used for attacking more rounds in the MITM pseudo-preimage attack in [104]. In a nutshell, the first is a carefully crafted initial structure, and the second is matching through indirect but efficiently computable **MixColumns** relations.

Initial Structure (IS). Concretely, when constructing the initial structure, by restricting the values of neutral bytes to a special set, the initial structure can extend the number of rounds in each chunk by one round. These special values of the neutral bytes are chosen in such a way so that their impacts on particular output bytes of the **MixColumns** (or **InverseMixColumns**) are known constants, *i.e.*, have no impact. This reduces the number of unknown bytes after (resp. before) the first **MixColumns** (resp. **InverseMixColumns**) in each chunk, which is the starting point of each chunk. For example, when 4 bytes $B^b[0, 1, 2, 3]$, *i.e.*, the bytes in the first column of the state B^b , are chosen as the neutral bytes for the forward chunk, there are 2^{32} possible values. Among them, we can choose a small subset of size 2^8 in the following way: first, fix an arbitrary constant C^{neut} of 3 bytes, then, impose the restriction of $B^f = \text{MC}^{-1}(B^b[0, 1, 2, 3])^3$ and $B^f[1, 2, 3] = C^{\text{neut}}$. Since

³In the sequel, we informally use $\text{MC}(X)$ (resp. $\text{MC}^{-1}(X)$) to represent the multiplication of a column X with the matrix (resp. its inverse) used in **MixColumns**.

C^{neut} is of 24 bits and $B^b[0, 1, 2, 3]$ is of 32 bits, there will be 2^8 solutions for $B^b[0, 1, 2, 3]$, which can be obtained by enumerating all possible values of $B^b[0]$ and for each computing the unique value of the other 3 bytes according to the restriction imposed through C^{neut} .

Matching through MixColumns (MTM). In the second technique, when matching at the meeting point, properties of the **MixColumns** are used again: instead of directly matching values of the same bytes in a state, matching via deterministic relations between bytes in different states (specifically, the states immediately before and after the **MixColumns**). This can extend the attack by one more round. Concretely, the **AES MixColumns** has the following feature: knowledge of any 4 out of the 8 input/output bytes to the MC will determine all other bytes, and knowledge of any additional byte(s) (*i.e.*, more than 4 bytes) can be used as an $8 \cdot (x - 4)$ -bit filter when a total of x bytes are known for $4 < x \leq 8$. With an example below, we demonstrate how this can be done in the way of MITM.

To efficiently check whether the paired values in the two lists match through the **MixColumns** operation, one can test the consistency between the bytes column-wise, individually for each column in sequence. Suppose the two states before and after the **MixColumns** are B^f and B^b , of which some bytes have been obtained independently via the forward and backward chunks, and the candidate values have been stored in two lists, *e.g.*, $B^f[0, 2]$ and $B^b[1, 2, 3]$ in the first column. These bytes are related through MC as follows:

$$\begin{aligned} B^f[0] &= (0xe, 0xb, 0xd, 0x9) \cdot (B^b[0], B^b[1], B^b[2], B^b[3])^T, \\ B^f[2] &= (0xd, 0x9, 0xe, 0xb) \cdot (B^b[0], B^b[1], B^b[2], B^b[3])^T. \end{aligned}$$

Let us denote by $g'(B^b[1, 2, 3]) = (0xb, 0xd, 0x9) \cdot (B^b[1], B^b[2], B^b[3])^T$ a linear mapping from 3 bytes to 1 byte, and by $g''(B^b[1, 2, 3]) = (0x9, 0xe, 0xb) \cdot (B^b[1], B^b[2], B^b[3])^T$ another linear mapping. Then

$$B^f[0] = 0xe \cdot B^b[0] \oplus g'(B^b[1, 2, 3]) \quad \text{and} \quad B^f[2] = 0xd \cdot B^b[0] \oplus g''(B^b[1, 2, 3]).$$

Canceling out $B^b[0]$ implies

$$0xd \cdot B^f[0] \oplus 0xe \cdot B^f[2] = 0xd \cdot g'(B^b[1, 2, 3]) \oplus 0xe \cdot g''(B^b[1, 2, 3]).$$

For the sake of simplicity, denote by $M^f = f(B^f[0, 2]) = 0xd \cdot B^f[0] \oplus 0xe \cdot B^f[2]$ a linear mapping from 2 bytes to 1 byte, and by $M^b = g(B^b[1, 2, 3]) = 0xd \cdot g'(B^b[1, 2, 3]) \oplus 0xe \cdot g''(B^b[1, 2, 3])$ another linear mapping from 3 bytes to 1 byte. Then, find matches with $M^f = M^b$, which can be computed independently and stored in hash tables, then matched in the way of MITM. We give below a more general statement of this.

Property 2. *When x out of the 8 input and output bytes of the **MixColumns** are known, there is a filter of $(x - 4)$ bytes ($= 8 \cdot (x - 4)$ bits), and such filtering can be done in the way of meet-in-the-middle.*

This property is used in the matching of all attacks to be given in the rest of this chapter, and for the sake of simplicity, the explicit expressions of M^f and M^b , *i.e.*, the derived matching functions f and g from the input B^f and B^b , are omitted. The total bits of filtering m is then the sum of 4 independent columns.

Attacks to be given in the rest of this chapter inherit both those techniques and the splice-and-cut framework of the attack in [104].

Exploiting the Freedom Lying in Multiple Targets (MT). In [114], authors observed that there is actually quite a great amount of additional degrees of freedom for backward chunk in the attack in [104], which are not utilized (set as constant). That is because the bottleneck for the attack in [104] lies in the lack of freedom for the forward chunk. That is, for the forward chunk, the degrees of freedom is much less than that of the backward chunk (*i.e.*, $d_1 \ll d_2$), which determines the time complexity. However, authors in [114] observed that in the setting of multi-target attacks, the freedom that lies in the target can be integrated into the forward chunk and thus increase the amount of degrees of freedom for the forward chunk. In doing that, those additional neutral bytes for the backward chunk can be utilized more fully. As a result, the attack in [104] was improved to be a more efficient multi-target pseudo-preimage attack, and the complexity of the overall preimage attack could be reduced.

4.2.2 Introducing Neutral Bytes in Key

Sasaki in [104] has already discussed the possibility to improve the attack described in Section 4.2.1 in the chosen-key setting, *i.e.*, introduce neutral bytes from key values. However, the conclusion is negative in consideration of the difficulties in adopting the splice-and-cut technique in the key schedule function.

However, in this chapter, we show that neutral bytes can be introduced from the key state to improve the computational complexity or increase the number of rounds that can be attacked. In our attacks, neutral bytes in the key are used as part of the forward chunk only, *i.e.*, none is used for the backward chunk. This enables us to avoid separating the key bytes in every round key generated by the key schedule function, and in the meantime, to exploit freedom from the key to balance the computations of forward and backward.

Explicitly, we introduce neutral bytes in the key state in addition to the original neutral bytes in the encryption state. Those neutral bytes are all for the forward chunk, while the effect of neutral bytes in the encryption state and those in the key state are different: neutral bytes in the encryption state affect the candidate values at the matching point through the AES round function in each direction, and neutral bytes in the key state affect through all subsequent round keys (via the key schedule). Thus, although they both affect the same bytes at the matching point, they both provide

possible candidate values for the matching bytes. Moreover, because the AES key schedule is computed independently of the AES round function, even the values that are chosen for the neutral bytes in encryption state and that in key state are within the same algebraic structure, *e.g.*, the same linear subspace, their effect on the matching values can be seen as independent.

At a high level, introducing neutral bytes in key state can play roles in various ways:

1. XORing neutral bytes in key state to neutral bytes in encryption state without any restriction on the values. In doing so, the lacking degrees of freedom can be increased, and the amount of computation cost by the forward chunk and by the backward chunk can be balanced, resulting in an improvement of the overall complexity. Abstractly, we represent this case as $\mathbf{A} \oplus \mathbf{A} = \mathbf{A}$, where $\mathbf{A}$ (the abbr. of “All”) represents an attribute of being fully free (that can take any value) of neutral bytes.
2. XORing neutral bytes in key state to neutral bytes in encryption state with a restriction that the XOR sum is constant. In doing so, the impact caused by neutral bytes in one computation chunk on the other computation chunk can be canceled, providing an opportunity to extend the attack to more rounds. Abstractly, we represent this case as $\mathbf{A} \oplus \bar{\mathbf{A}} = \mathbf{C}$, where $\bar{\mathbf{A}}$ represents an attribute of being restricted by the value of other fully free neutral bytes, and $\mathbf{C}$ represents an attribute of being constant.
3. Combining neutral bytes in key state with neutral bytes in encryption state by linear equations (related with the linear diffusion layer of the cipher) such that their XOR sum can take any value, but after being operated by the linear layer of the cipher, part of the result is constant. In doing so, the impact caused by neutral bytes in one computation chunk on the other computation chunk can be partially canceled, at the same time additional freedom is imported, providing an opportunity to improve the overall complexity and/or extend the attack to more rounds. Abstractly, we represent this case as $\mathcal{L}(\mathbf{A}, \mathbf{A}) = \mathbf{A} \parallel \mathbf{C}$, where $\mathcal{L}(\mathbf{A}, \mathbf{A})$ represents an attribute of several fully free neutral bytes being restricted by linear equations.

In the next two sections, we first present our 7-round pseudo-preimage attacks on AES-128, AES-192 and Kiasu-BC based hashing modes improved in terms of attack complexity, and then present our 8-round attacks on AES-192, AES-256, and Kiasu-BC hashing mode improved in terms of the number of attacked rounds, which are all enabled by introducing neutral bytes in the key.

4.3 Reducing the Complexities of 7-Round Attacks

4.3.1 Improved Attack on 7-Round AES-128 Hashing Mode

As mentioned above, in [114], authors identified a great amount of freedom that can be imported by activating bytes that were set as constant previously in [104] as neutral bytes for the backward chunk. The problem is then the lack of freedom for the forward chunk. In [114], the authors exploited the freedom that lies in the target in the setting where multiple targets are available. In our attack, we import freedom for the forward chunk by activating particular bytes that were set as constant previously in [104] both in the encryption state and in the key state as neutral bytes. The introduced neutral bytes from the key can cancel out the diffused impact caused by (inverse) `MixColumns` operated on a neutral byte in the forward chunk, at the same time provide an additional degree of freedom.

Details of our improved pseudo-preimage attack are as follows (illustrated in Figure 4.3, $\mathcal{S}$ denotes the state).

Attack Configuration

1. Initial structure: from $\mathcal{S}12 - \mathcal{S}19$ including $k_3 - k_4$

- Neutral bytes for forward:
 - In encryption state: $\mathcal{S}12[0, 5]$
 - In key state: $k_3[4, 5, 6, 7]$, such that

$$\begin{cases} \text{MC}^{-1}((0, \mathcal{S}12[5], 0, 0) \oplus k_3[4, 5, 6, 7])[1, 2, 3] = C_1^{\text{neut}} \\ k_3[0, 1, 2, 3] \oplus k_3[4, 5, 6, 7] = C_1^{\text{key}}. \end{cases}$$

In the first equation, C_1^{neut} is a 3-byte constant, $\mathcal{S}12[5]$, $k_3[4]$, $k_3[5]$, $k_3[6]$, $k_3[7]$ are 1-byte variables. That is, the first equation essentially consists of three linear equations on five variables, *i.e.*,

$$\begin{bmatrix} 9 & e & b & d & e \\ d & 9 & e & b & 9 \\ b & d & 9 & e & d \end{bmatrix} \times \begin{bmatrix} k_3[4] \\ k_3[5] \\ k_3[6] \\ k_3[7] \\ \mathcal{S}12[5] \end{bmatrix} = \begin{bmatrix} C_{1,0}^{\text{neut}} \\ C_{1,1}^{\text{neut}} \\ C_{1,2}^{\text{neut}} \end{bmatrix}.$$

Because the matrix used in AES `MixColumns` is MDS, the rows are pair-wise independent. The rank of the above coefficient matrix is three. Thus, the number of solutions for these equations is $2^{(5-3) \times 8} = 2^{16}$. By solving the linear equations, one can efficiently

obtain all the values of the neutral bytes fulfilling the constraints⁴. In short, the first equation brings 2 bytes degree of freedom for the forward computation, at the same time makes the impact of those neutral bytes on the backward computation of $\mathcal{S}11[5, 6, 7]$ be predefined constant C_1^{neut} .

Fix all other bytes in k_3 , *i.e.*, $k_3[8, \dots, 15]$ as constant. By the second equation and according to the key schedule of AES-128, $k_4[0, 1, 2, 3] \oplus k_3[0, 1, 2, 3]$ and all other bytes in k_4 are constant. Denotes all constant materials in the key state as C^{key} .

As a result, there is 3-byte degree of freedom brought by neutral bytes for the forward chunk, *i.e.*, $d_1 = 24$.

- Neutral bytes for backward:
 - In encryption state: $\mathcal{S}19[1, 2, 3; 4, 5, 6; 8, 9, 11; 12, 14, 15]$, such that

$$\begin{cases} \text{MC}((0, \mathcal{S}19[1], \mathcal{S}19[2], \mathcal{S}19[3])) &= C_{2,0}^{\text{neut}} \\ \text{MC}((\mathcal{S}19[4], \mathcal{S}19[5], \mathcal{S}19[6], 0)) &= C_{2,1}^{\text{neut}} \\ \text{MC}((\mathcal{S}19[8], \mathcal{S}19[9], 0, \mathcal{S}19[11])) &= C_{2,2}^{\text{neut}} \\ \text{MC}((\mathcal{S}19[12], 0, \mathcal{S}19[14], \mathcal{S}19[15])) &= C_{2,3}^{\text{neut}}. \end{cases}$$

That is, their impact on the forward computation of $\mathcal{S}20[0, 2; 5, 7; 8, 10; 13, 15]$ equal to predefined constants $C_2^{\text{neut}} = C_{2,0}^{\text{neut}} \| C_{2,1}^{\text{neut}} \| C_{2,2}^{\text{neut}} \| C_{2,3}^{\text{neut}}$, where $C_{2,i}^{\text{neut}}$ for $0 \leq i \leq 3$ are 2-byte constants.

As a result, there is 4-byte degree of freedom brought by neutral bytes for backward computation, *i.e.*, $d_2 = 32$.

2. Chunk separation:

- Forward chunk: the computation following $\mathcal{S}20 - \mathcal{S}28 - \mathcal{S}0 - \mathcal{S}7$
 - Backward chunk: the computation following $\mathcal{S}11 - \mathcal{S}8$
3. Bytes for matching: denote the equivalent subkey of k_2 by uk_2 , which can be computed via $\text{MC}^{-1}(k_2)$:

- $B^{\text{f}} = (\mathcal{S}7 \oplus uk_2)[0, 2; 8, 10; 13, 15]$, note the second column is not utilized.
- $B^{\text{b}} = \mathcal{S}8[1, 2, 3; 8, 9, 11; 12, 13, 14]$, note the second column is not utilized.

As a result, there are equivalently 3 bytes for the match, *i.e.*, $m = 24$.

⁴Please refer to the example codes for obtaining the values of the neutral bytes via <https://doi.org/10.21979/N9/JYECIO>.

Attack Procedure

1. Fix constants:
 - C^{key} : constant materials in the key state, including C_1^{key} .
 - C_0^{neut} : 2 bytes values for $\mathcal{S}12[10, 15]$.
 - C_1^{neut} : 3 bytes values for impacts from $\mathcal{S}12[5]$ and $k_3[4, 5, 6, 7]$ on $\mathcal{S}11[5, 6, 7]$.
 - C_2^{neut} : 8 bytes values for impacts from $\mathcal{S}19[1, 2, 3; 4, 5, 6; 8, 9, 11; 12, 14, 15]$ on $\mathcal{S}20[0, 2; 5, 7; 8, 10; 13, 15]$.
2. Forward computation: for each of the $2^{d_1=24}$ values of the neutral bytes for forward – $\mathcal{S}12[0, 5]$ and $k_3[4, 5, 6, 7]$ s.t. their impact on $\mathcal{S}11[5, 6, 7] = C_1^{\text{neut}}$:
 - Compute all required subkey bytes from $k_3[4, 5, 6, 7]$ and C^{key} , note that $k_3[0, 1, 2, 3] \oplus k_3[4, 5, 6, 7]$, $k_4[0, 1, 2, 3] \oplus k_3[4, 5, 6, 7]$, and all other bytes in k_3 and k_4 are constants.
 - Compute in forward from $\mathcal{S}12 - \mathcal{S}28$ (blue cells in Figure 4.3)
 - xor state $\mathcal{S}28$ with T , and compute $\mathcal{S}0 - \mathcal{S}7$ (blue cells in Figure 4.3).
 - compute M^f from B^f , store results in L^f .
3. Backward computation: for each of the $2^{d_2=32}$ values of the neutral bytes for backward – $\mathcal{S}19[1, 2, 3; 4, 5, 6; 8, 9, 11; 12, 14, 15]$ s.t. their impact on bytes $\mathcal{S}20[0, 2; 5, 7; 8, 10; 13, 15] = C_2^{\text{neut}}$:
 - Compute in backward from $\mathcal{S}19 - \mathcal{S}8$ (red cells in Figure 4.3).
 - Compute M^b from B^b , store the results in L^b .
4. Matching: find matches between L^f and L^b , for each such match of partial state, all bytes at the initial structure are fixed, and hence the full states of $\mathcal{S}7$ and $\mathcal{S}8$ can be tested for full-state matching. Output (H_{N-1}, M_N) if a full match is found, otherwise, go back to step 1 with some other values for the fixed constants at initial structure and repeat.

Attack Complexity. By directly applying the complexity analysis Equation (4.1) and plugging $d_1 = 24$, $d_2 = 32$, and $m = 24$, we can immediately conclude that the time complexity of this attack is $2^{128 - \min(24, 32, 24)} = 2^{104}$ and the memory complexity is 2^{24} .

In summary, in this attack, we introduce neutral bytes from the key in addition to two neutral bytes in the encryption state, and make them play a role in the third way, *i.e.*, $\mathcal{L}(\mathbf{A}, \mathbf{A}) = \mathbf{A} \parallel \mathbf{C}$. In doing that, the degree of freedom for the forward chunk is increased with a limited impact on the backward chunk. Note that the limited impact is the loss of 8 bits for the match. Nevertheless, the computations of the forward chunk and that of

Figure 4.3: Introducing free bytes in the key to improve the 7-round preimage attack on AES-128 hashing mode.

the backward chunk are more balanced. As a result, the computational complexity can be reduced from 2^{120} to 2^{104} given a single target compared with the attack in [104], and changed from $2^{120-\min(t,24)}$ to 2^{104} given 2^t targets compared with the attack in [114].

4.3.2 Application to 7-Round AES-192 Hashing Mode

Unlike the 7-round attacks in [104, 114], the attack presented in above Section 4.3.1 is not quite general due to its dependence on key schedule algorithms. Thus, it cannot be applied without modification for AES-192 and AES-256. However, the idea of introducing neutral bytes in the key can nevertheless be applied.

Compared to AES-128, the key schedule of AES-192 has a relatively slow diffusion. Consequently, it is possible to select more neutral bytes from the key to completely cancel out the impact caused by neutral bytes in the

forward chunk on the backward chunk. Concretely, for AES-192, according to the key schedule, the full key state k_3 can be fixed without dependence on neutral bytes in the first column of k_4 (as depicted in Figure 4.4b). Thus, the propagation of the influence brought by the neutral bytes in k_4 will not impact the backward computation which is relatively short.

An attack follows the same framework as previous attacks but with improved complexity that can then be devised. The main different part is depicted in Figure 4.4. As can be seen from Figure 4.4, for the forward chunk, when introducing neutral bytes $\mathcal{S}17[0, 1, 2, 3]$ in encryption state and $k_4[0, 1, 2, 3]$ in key state and restricting their XOR sum to be constant, their impact on the backward chunk is completely canceled, and at the same time, both the degrees of freedom for the forward chunk and that for the backward chunk reach an amount of 32, *i.e.*, $d_1 = d_2 = 32$. The number of bits for match becomes even more, *i.e.*, $m = 64$. Applying Equation (4.1) to the complexity analysis, we conclude that, for this attack on 7-round AES-192 hashing mode, the total computational complexity is $2^{128 - \min(32, 32, 64)} = 2^{96}$ computations of 7-round AES. The memory complexity is 2^{32} .

Note that compared to the key schedule of AES-192, the diffusion of the key schedule of AES-256 is slower. Again, the full key state k_3 can be fixed as constant without being impacted by the neutral bytes in the first column of k_4 . Thus, this attack directly applies to AES-256 hashing mode as well.

In summary, in this attack, we introduce neutral bytes from the key and make them play a role in the second way, *i.e.*, $\mathbf{A} \oplus \bar{\mathbf{A}} = \mathbf{C}$. Due to the relatively slow diffusion of the key schedule, the introduced large number of neutral bytes from the key can completely cancel out the impact caused by the forward neutral bytes in the encryption state on the backward chunk, at the same time, leave an adequate degree of freedom for the forward chunk.

4.3.3 Application to 7-Round Kiasu-BC Hashing Mode

In the scenario where a tweakable block cipher is used in the PGV hashing mode and the tweaks can accept chosen inputs, freedom from this additional input might be exploited in similar attacks to the above ones. We apply the same technique to the hashing mode based on Kiasu-BC in this section as its encryption algorithm and key schedule are exactly that of AES-128 (even though it was never advised by the authors that Kiasu-BC should be used in a compression function).

The attack on 7-round AES hashing mode in [114] can be directly improved by introducing two additional neutral bytes for the forward computation from the tweak without any impact on the backward computation. Specifically, we choose $tk[0, 1]$ as the neutral bytes for the forward chunk, which can take 2^{16} values and which has limited impact on the backward chunk (refer to Figure 4.5). In this way, together with the 2^8 freedom from neutral byte $\mathcal{S}12[0]$, we can finally obtain $2^{d_1=8+16=24}$ candidates in M^f without any cost (which is unlike the attack in Section 4.3.1 that reduces the number of bits for the match from 32 to 24). Corresponding to the complexity analysis equation

Figure 4.4: Introducing free bytes in the key to improve the 7-round preimage attack on AES-192 hashing mode.

Equation (4.3) and compared with that of the attack in [114], the parameter d_1 increases from 8 to 24 and leaves all other parameters unchanged. Note that the number of bits for the match is still 32 which is unlike the attack in Section 4.3.1. Thus, additional freedom in the target can still be utilized in the setting of a multi-target attack.

Consequently, for 7-round Kiasu-BC hashing mode, by introducing two neutral bytes from the tweak in addition to one neutral byte in the encryption state, and in the setting given 2^t targets, the attack will have a computational complexity $2^{128-\min(24+t,32)} = 2^{104-\min(t,8)}$ and a memory complexity $2^{24+\min(t,8)}$ (recall Equation 4.3). Note that both the key and the tweak values are used here, so Type-III TBC-to-BC conversion is assumed here.

In summary, in this attack, we introduce neutral bytes from the tweak and make them play a role in the first way, *i.e.*, $\mathbf{A} \oplus \mathbf{A} = \mathbf{A}$. By adding neutral bytes from the tweak directly to neutral bytes in the encryption state without any restriction on the XOR sum, the gain of degree of freedom is completely free in the sense that the neutral bytes from the tweak brings no additional impact on the backward chunk (on top of the impact caused by the neutral bytes in the encryption state).

4.4 Extension to 8-Round Attacks

In this section, we extend the 7-round attacks of the previous section to 8-round attacks by introducing freedom from the key. We first explain the technique used in our extension, then provide its application to AES-256, AES-192 and Kiasu-BC.

Figure 4.5: Introducing free bytes in the key to improve the 7-round preimage attack on Kiasu-BC hashing mode.

4.4.1 Techniques for Attacking 8 Rounds

Figure 4.6: Basic idea of the extended 8-round attack, k_3 and k_5 can have at most one column of neutral bytes each.

We add one more round to the backward chunk based on the previous 7-round attacks, and the new chunk separation is shown in Figure 4.6 (for a comparison, see Figure 4.3, 4.4, and 4.5). Notice that $S_{17}[0]$ is a neutral byte for the forward computation and can have 2^8 values, and it is denoted by x . We regard the whole key schedule as a part of the forward chunk and set the value of $k_4[0]$ to be $x \oplus c$, where c is a constant value, and the left 15 bytes of k_4 is equal to predefined constant values. In this way, the forward chunk

Figure 4.7: Introducing free bytes in the key to launch an 8-round preimage attack on AES-256 hashing mode.

is computed with the knowledge of neutral bytes from both the key and the internal states. While for the backward computation, the value of $\mathcal{S}16[0]$ is fixed to be c , thus states $\mathcal{S}13$ to $\mathcal{S}15$ can be computed deterministically. As for k_3 , at most one column can be the forward neutral bytes, otherwise, $\mathcal{S}8$ would have more than one unknown bytes in each column, which invalidates the matching step. And the neutral bytes in k_5 should not overlap with the neutral bytes for the backward chunk in state $\mathcal{S}20$. The value of k_2 does not have any impact on the backward computation since the `MixColumns` is linear and we can instead find the match between $\mathcal{S}7 \oplus \text{MC}^{-1}(k_2)$ and $\mathcal{S}8$ through the `MixColumns`.

4.4.2 The 8-Round Attack on AES-256

Due to the slower diffusion of the key schedule, we can apply the technique of Section 4.4.1 to the attack on the 8-round AES-256 hashing mode. The key path is initialized from the 256-bit state k_4 and k_5 in the way that the freedom in k_4 could neutralize the randomness in the backward chunk, and additional 2^{32} freedom degrees could be introduced in k_5 , which is shown in Figure 4.7. We give below the details of the attack.

Attack Configuration

1. Initial structure:

- Neutral bytes for forward (in encryption state): 2^8 possible values of $\mathcal{S}20[12,13,14,15]$, s.t., $\mathcal{S}19[12,14,15]$ equals predefined constant

$$C_0^{\text{neut}},$$

- Neutral bytes for forward (in key state): 2^{32} possible values of $k_5[12, 13, 14, 15]$, s.t., their impacts on $\mathcal{S}19[12, 14, 15]$ equals predefined constant C_1^{neut} .

The value of $k_4[1]$ is chosen in the way so that its XOR difference with $\mathcal{S}17[1]$ equals to a predefined constant C^{byte} , i.e., $k_4[1] = \mathcal{S}17[1] \oplus C^{\text{byte}}$.

As a result, for the backward chunk, states $\mathcal{S}13$ to $\mathcal{S}16$ can be computed deterministically.

Other bytes in k_4 and k_5 equal to predefined constant C^{key} .

- Neutral bytes for backward: $(2^8)^4 = 2^{32}$ possible values of $\mathcal{S}23[0, 1, 2], \mathcal{S}23[4, 5, 7], \mathcal{S}23[8, 10, 11], \mathcal{S}23[13, 14, 15]$, s.t., their impacts on $\mathcal{S}24[0, 2], \mathcal{S}24[5, 7], \mathcal{S}24[8, 10], \mathcal{S}24[13, 15]$ equal predefined constant $C_2^{\text{neut}} = C_{2,0}^{\text{neut}} \parallel C_{2,1}^{\text{neut}} \parallel C_{2,2}^{\text{neut}} \parallel C_{2,3}^{\text{neut}}$.

2. Chunk separation:

- Forward chunk: the computation following $\mathcal{S}24 - \mathcal{S}32 - \mathcal{S}0 - \mathcal{S}7$.
- Backward chunk: the computation following $\mathcal{S}19 - \mathcal{S}8$.

3. Bytes for matching: Denote the equivalent subkey of k_2 by uk_2 , which can be computed via $\text{MC}^{-1}(k_2)$:

- $B^f = (\mathcal{S}7 \oplus uk_2)[0, 2; 5, 7; 8, 10; 13, 15]$.
- $B^b = \mathcal{S}8[0, 2, 3; 4, 5, 7; 8, 9, 10; 13, 14, 15]$.

Attack Procedure

1. Fix constants:

- C_0^{neut} : a value for 3-byte impact from neutral bytes in $\mathcal{S}20$ on $\mathcal{S}19[12, 14, 15]$.
- C_1^{neut} : a value for 3-byte impact from $k_5[12, 13, 14, 15]$ on $\mathcal{S}19[12, 14, 15]$.
- C^{byte} : a value for byte $\mathcal{S}16[1]$.
- C^{key} : a 27-byte constant value in k_4 and k_5 .
- C_2^{neut} : a value for 8-byte impact from neutral bytes in $\mathcal{S}23$ on $\mathcal{S}24[0, 2, 5, 7, 8, 10, 13, 15]$.

2. Forward computation: for 2^8 values of $\mathcal{S}20[12, 13, 14, 15]$ s.t. their impact on $\mathcal{S}19[12, 14, 15]$ equals to C_0^{neut} , and for 2^{24} out of 2^{32} values of $k_5[12, 13, 14, 15]$ s.t. their impact on $\mathcal{S}19[12, 14, 15]$ equals to C_1^{neut} :

- Compute the value of $\mathcal{S}17[1]$.
- Determine the value of $k_4[1]$: $k_4[1] = \mathcal{S}17[1] \oplus C^{\text{byte}}$.
- Compute all subkeys from k_4 and k_5 through the key schedule.

- Compute the forward chunk from $\mathcal{S}24$ to $\mathcal{S}32$, and from $\mathcal{S}0$ to $\mathcal{S}7$.
 - Compute M^f from B^f , and store the results in a table L^f .
3. Backward computation. For 2^{32} values of the neutral bytes $\mathcal{S}23[0, 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 15]$ s.t. their impact on bytes $\mathcal{S}24[0, 2, 5, 7, 8, 10, 13, 15]$ equals to C_2^{neut} .
- Compute the backward chunk from $\mathcal{S}19$ to $\mathcal{S}8$. The involved subkey bytes are constant in the backward computation.
 - Compute M^b from B^b , and store the results in a table L^b .
4. Matching: follow the same idea of the attack in Section 4.3.1.

Attack Complexity . The freedom degrees for both of the forward and backward chunk are 32, *i.e.*, $d_1 = d_2 = 32$, and the number of bits for matching comes from 32 bits ($m = 32$). Thus according to Equation (4.1), the time complexity is 2^{96} and the memory complexity is 2^{32} .

In summary, in this attack, we introduce neutral bytes from the key and make them play the role in the second way, *i.e.*, $\mathbf{A} \oplus \bar{\mathbf{A}} = \mathbf{C}$.

4.4.3 The 8-Round Attack on AES-192

Figure 4.8: Introducing free bytes in the key to launch an 8-round preimage attack on AES-192 hashing mode.

The main part of the attack is illustrated in Figure 4.8a. The data path in the internal state is similar to the attack on AES-256, the freedom for

backward chunk remains $2^{d_2=32}$, but the neutral bytes for the forward in state $\mathcal{S}20$ is changed to the second column. The way of introducing neutral bytes in the key is similar to the attack on 7-round AES-128: we introduce $\mathcal{S}16[3]$ as the neutral byte in encryption state and $k_4[0, 1, 2, 3]$ as neutral bytes in the key state for the forward chunk, with the following constraints ⁵

$$\begin{cases} \text{MC}^{-1}((0, 0, 0, \mathcal{S}16[3]) \oplus k_4[0, 1, 2, 3])[0, 1, 2] = C_1^{\text{neut}} \\ k_3[12, 13, 14, 15] \oplus k_4[0, 1, 2, 3] = C_1^{\text{key}}. \end{cases}$$

Besides, all remaining bytes in the 192-bit subkey state (including the whole state of k_3 and the first two columns of k_4) are fixed as constant. According to the key schedule of AES-192, we have $k_5[4, 5, 6, 7] = k_3[12, 13, 14, 15] \oplus \text{constant}$. In the constraints, the first equation ensures the impact caused by neutral bytes in the forward chunk on the backward chunk to be limited to a single byte (is constant in three bytes); the second equation together with the property of the key schedule of AES-192 ensure that all the active bytes of the 192-bit subkey state are determined by predefined constants and the four neutral bytes without bringing further impact on backward chunk. Combining the neutral byte $\mathcal{S}16[3]$ in encryption state and neutral bytes $k_4[0, 1, 2, 3]$ in the key state using the first linear equation, we obtain $d_1 = (5 - 3) \cdot 8 = 16$ bits of freedom for forward chunk.

Note that the number of bits for matching is $m = 32$ as can be seen in Figure 4.8a, which is larger than d_1 . Thus, in a multi-target scenario, we can use freedom from the target to further balance the computation cost by the forward chunk and that by the backward chunk. Applying Equation 4.3 to the complexity analysis, we conclude that this attack requires $2^{128 - \min(16+t, 32, 32)} = 2^{112 - \min(t, 16)}$ computations of 8-round AES and $2^{\min(16+t, 32, 32)} = 2^{16 + \min(t, 16)}$ memory.

In summary, in this attack, we introduce neutral bytes from the key and make them play a role in the third way, *i.e.*, $\mathcal{L}(\mathbf{A}, \mathbf{A}) = \mathbf{A}||\mathbf{C}$. By introducing neutral bytes from the key, the impact caused by the forward neutral bytes on the backward chunk is partially canceled, and at the same time the freedom for the forward chunk is increased, which enables to extend the attack to more rounds with an improved complexity.

4.4.4 The 8-Round Attack on Kiasu-BC

Since Kiasu-BC adds the same tweak to the first two rows of internal state for each round, we can introduce neutral bytes from tweak instead of key values, and the attack configuration in Figure 4.6 can be directly adopted with the only modification in the key and the tweak: the key states are set to be constant values, and the tweak is initialized in such a way that $tk[0] = \mathcal{S}17[0] \oplus \text{constant}$ and the left 7 bytes are constant values. This

⁵Again, to obtain the values of neutral bytes fulfilling the constraints, one can efficiently solve the linear equations using linear algebra. Please refer to the example codes via <https://doi.org/10.21979/N9/JYECIO>.

attack follows the same framework as previous attacks, the degree for the forward chunk is $d_1 = 8$, for the backward chunk is $d_2 = 32$, and the number of bits for matching is $m = 32$, the freedom that lies in the target in multi-target setting can be utilized. Applying Equation 4.3, we have that the time complexity is $2^{120-\min(t,24)}$ and the memory complexity is $2^{8+\min(t,24)}$. Note the tweak values are used here, but not the key values, so both Type-II and Type-III TBC-to-BC conversions fit the attack setting here. In summary, this attack is enabled by introducing neutral bytes from the tweak and make them play the role in the second way, *i.e.*, $\mathbf{A} \oplus \bar{\mathbf{A}} = \mathbf{C}$, so that extending to one more round is quite straightforward.

4.5 Discussion

Applying to Other Secure PGV Modes. As mentioned in Section 4.1.3, there are twelve secure PGV mode-of-operations to build a compression function from a block cipher. We note that for those modes in which the chaining state is fed into the key schedule (*i.e.*, PGV modes in Class 1 and Class 3, refer to Table 1.1), because of the incompatibility between key size (192 or 256 bits) and state size (128 bits), AES-192/AES-256 are not applicable. Apart from these incompatible instantiations of PGV modes by AES-192/AES-256, we conclude that the presented pseudo-preimage attacks are applicable to all the twelve PGV modes instantiated by AES.

Specifically, when H_{i-1} and M_i are of the same size, for our attacks, those PGV modes are essentially equivalent up to the exchange between H_{i-1} and M_i , and/or up to the XORing of whitening key (the material fed into the key schedule of the block cipher), and/or up to replacing either H_{i-1} or M_i with $H_{i-1} \oplus M_i$. When applying the presented attacks, the output of the encryption, denoted by Z , and the input to the key schedule, denoted by K , are firstly determined following the same MITM procedure for all PGV modes. The freedom required to find the correct Z and K are provided by H_{i-1} and M_i . The difference between the attacks on difference PGV modes lies in where the required freedom comes from. Thus, the equivalence up to the exchange between H_{i-1} and M_i is relatively easy to be understood, considering that for pseudo-preimage attacks, both H_{i-1} and M_i are unknown variables that are determined at the end of the attack procedure such that $\mathcal{H}(H_{i-1}, M_i) = T$ for the given target T . Examples for this case of equivalence are PGV No.1 $\equiv$ No.5, No.2 $\equiv$ No.6, No.3 $\equiv$ No.7, No.4 $\equiv$ No.8, No.9 $\equiv$ No.10, No.11 $\equiv$ No.12 (refer to Figure 4.9).

For the equivalence up to the XORing of the whitening key, the explanation is as follows. Recall that, in AES, the whitening key is the master key and is XORed at the very beginning of the encryption, and recall that our attacks use the splice-and-cut method. For PGV modes in which both of the material fed into the key schedule and the material fed into the encryption are XORed at the end of the encryption, splicing in our attacks will result in a cancellation effect on the whitening key at the beginning of the encryption.

Figure 4.9: Equivalence among difference PGV modes for our attacks.

However, as can be seen in Figure 4.3, whether the master key is XORed or is not XORed at the beginning of the encryption has no essential influence on our attacks. Thus, there is no essential difference for our attacks applying to those PGV modes. Examples for this case of equivalence are PGV No.2 $\equiv$ No.3, No.6 $\equiv$ No.7, No.1 $\equiv$ No.4, No.5 $\equiv$ No.8, No.9 $\equiv$ No.12, No.10 $\equiv$ No.11 (refer to Figure 4.9).

At last, it is easy to see that PGV No.1 and No.2 are essentially equivalent up to taking $H_{i-1} \oplus M_i$ as the variable providing the freedom required for finding correct Z , PGV No.1 and No.9 are essentially equivalent up to taking $H_{i-1} \oplus M_i$ as the variable providing the freedom required for finding correct K (refer to Figure 4.9).

In conclusion, for our attacks, the PGV modes are essentially equivalent and our attacks are applicable to all of them (apart from the above mentioned incompatible instances).

Converting to (Second-) Preimage Attacks. Whether or not the presented pseudo-preimage attacks can be converted to full preimage attacks depends on the padding rule of the hashing mode. That is because all our pseudo-preimage attacks require that the message block can be freely chosen (for providing the freedom required by the input to the key schedule or by the input to the encryption). When the padding rule imposes special format to the last message block such that a valid padded message block cannot take arbitrary value (*i.e.*, the padding rule makes the domain of valid padded block smaller than that of normal blocks), converting our attacks to full preimage attacks requires additional computations (repeating the procedure of the pseudo-preimage attack for the last block so that the recovered block happens to be a valid one, *e.g.*, when using the padding rule in **SHA-3** that

reduces the domain by a factor of 2^{-1} , we expected to repeat two times; when using the Merkel-Damgård strengthening, we may need to repeat many times such that the overall complexity is higher than that of a brute-force attack).

Nevertheless, all our pseudo-preimage attacks can be converted into second-preimage attacks, in which the crafted messages reuse the last block of the given message, and the target is the second to the last chaining value in the processing of the given message. The complexity analysis on converting pseudo-preimage attack to preimage attack provided in Section 4.1.2 applies directly without any modification to the case of converting pseudo-preimage attack into a second-preimage attack.

4.6 Conclusion

Under the general framework of MITM preimage attack against AES hashing modes introduced by Sasaki in 2011 and improved by Wu *et al.* in 2012, we made two observations: the key bits are not used, and the neutral bits in the two chunks are not balanced in Wu *et al.*'s improvement. In this chapter, we introduced neutral bits from the key, and to avoid dealing with the fast diffusion of the AES key schedule, neutral bits from the key are introduced for one chunk only. By carefully choosing the key neutral bits, we found it was indeed possible while keeping the computation of the other chunk unaffected. Then the additional degrees of freedom are used in three ways to play two roles, *i.e.*, to reduce time complexities and to extend the attack to more rounds. As a result, we improved the MITM preimage attack complexities for 7-round AES hashing modes under all 3 versions of AES, and extended the attack to 8 rounds under AES-192 and AES-256. The same was applied to Kiasu-BC.

Chapter 5

Cryptographic Backdoor Design

Inserting backdoors in encryption algorithms has long seemed like a very interesting, yet difficult problem. Most attempts have been unsuccessful for symmetric-key primitives so far and it remains an open problem how to build such ciphers. Especially, embedding backdoors into block ciphers is a challenging problem since block ciphers are deterministic and thus it is complex to exploit randomness in computations. Young and Yung have designed several backdoors in secret block ciphers [117, 118, 120], where it is assumed that the cipher specifications are unknown to adversaries. However, it would be more interesting and desirable to design backdoors in public block ciphers, and it is the topic of this chapter.

There have been only limited works focusing on backdooring public block ciphers and to the best of our knowledge none of them provides a secure backdoor. In 1997, Rijmen and Preneel proposed a special S-box design strategy which was used to hide a high-probability linear approximation in an S-box [100]. The knowledge of this backdoor leads to an efficient key recovery attack based on linear cryptanalysis, but only a part of the key information can be obtained. They presented concrete instantiations by applying the S-box design to CAST and LOKI91 ciphers and claimed that the embedded backdoors are undetectable even if the general form of the backdoor is known. However, this design was broken subsequently in 1998 [113] by Wu *et al.* who found a way to easily recover the backdoor and showed that the security and practicability of the backdoor can't be guaranteed at the same time. Later in 1999, Paterson suggested that if the group generated by round functions acts imprimitively on the message group, then it is possible to create a backdoor in the cipher [94]. Built upon this mechanism, he introduced a DES-like cipher which allows an entity knowing the backdoor to retrieve the key with 2^{41} computations. However, as mentioned by the author, the backdoor is detectable and the cipher is vulnerable to differential attacks. Following on this idea, a backdoor based on partitioning cryptanalysis was studied in [12] and a concrete instance of an AES-like cipher called BEA-1 was later proposed in [13], but no explicit backdoor security was provided. One can also mention

the work from Patarin and Goubin [92, 93] who proposed “2R-schemes”, basically S-box-based asymmetric schemes secretly consisting of a 2-round secret Substitution-Permutation Network (SPN) but publicly represented as its corresponding algebraic equations. However, this research direction also suffered from attacks [25, 115]. Two more backdoor designs [7, 39] have been introduced, but neither of them provide solid proof for the backdoor security and even the security of the cipher itself is questionable. Lastly, in a different setting, a backdoored version of the **SHA-1** hash function was proposed in [2], where the attacker is allowed to pre-choose the constants used in the design, so he can prepare in advance some specific collision messages for that particular instance.

Contributions of this Chapter. We propose a new method to generate backdoor encryption algorithms. We bring together tweakable block ciphers (TBC) and Extendable-Output Function (XOF) in a common framework called **MALICIOUS**, which enables the designer to embed backdoors into the TBC. The general representation of our construction is similar to that of the **TWEAKEY** framework, but the tweak is handled separately by a XOF and the round function has to be partially non-linear.

Our backdoor is based on differential cryptanalysis: due to the partial non-linear layer, the designer can embed related-tweak differential characteristics with probability 1 over many rounds. In particular, the subtweak difference employed in an embedded differential characteristic is generated from a specific tweak pair that is chosen in advance by the designer. This malicious tweak pair is the backdoor, and the XOF applied in the tweak schedule is used to protect the malicious tweak pair: even knowing the high-probability related-tweak differential characteristic, it will remain computationally difficult to find a tweak pair that triggers it. More importantly, the backdoor security is ensured by the target-difference resistance ability of the chosen XOF. An attacker with the knowledge of the backdoor is able to retrieve the full key with negligible effort under the chosen-tweak scenario.

Based on the **MALICIOUS** framework, we also propose a concrete instantiation that we call **LowMC-M**. Our family of TBC **LowMC-M** is created based on some instances of the block cipher **LowMC** [3]. Compared to **LowMC**, our proposal **LowMC-M** has an additional subtweak addition in each round and the tweak schedule is a XOF, but the other parts of the round function and the number of rounds remain unchanged. Apart from its backdoor security that is naturally inherited from the **MALICIOUS** framework, we claim that its classical black-box security against state-of-the-art cryptanalysis is the same as the corresponding **LowMC** variants. This work was published at [95].

Organizations. In Section 5.1, we present the attacking scenario and some security notions for backdooring cryptographic primitives. In Section 5.2 the **MALICIOUS** framework is described and its backdoor security and design rationale are explained. We introduce a concrete instantiation of **MALICIOUS** (so-called **LowMC-M**) in Section 5.3. We then analyze **LowMC-M** with respect

to the backdoor security and the classical black-box security in Section 5.4 and Section 5.5, respectively. Finally, we present our conclusions in Section 5.6.

5.1 Basics of Backdoor

5.1.1 Attacking Scenario

For classical (tweakable) block ciphers, the attacking scenario considers only two entities: the *user* (or pair of users) who owns the secret key and the *attacker* who tries to break the cryptosystem, *i.e.*, to find out the secret key. For (tweakable) block ciphers with a backdoor, another entity has to be involved in the attacking scenario: the *designer*, who inserts the backdoor into the primitive. Thus, we have in total three entities: the designer (knows the backdoor, but not the secret key), the user (knows the secret key, but not the backdoor) and the attacker (neither backdoor nor key is known).

One can see that both the user and the attacker have some motivation to find out what is the backdoor. More importantly, in our model the backdoor is independent of the secret key, and therefore the user and the attacker possess the same capability in trying to uncover the backdoor (the cipher specifications are public known, so they can test the cipher with any chosen key they want). For the rest of this chapter, when considering the recovery of the backdoor, we will simply refer to both of them as the attacker.

5.1.2 Security Notions and More

We introduce below various notions regarding the security and the practicality of a backdoor:

- *Undetectability*: this security notion represents the inability for an external entity to realize the existence of the hidden backdoor.
- *Undiscoverability*: it represents the inability for an attacker to find the hidden backdoor, even if the general form of the backdoor is known.
- *Untraceability*: it states that an attack based on the backdoor should not reveal any information about the backdoor itself.
- *Practicability*: this usability notion stipulates that the backdoor is practical, in the sense that it is easy to recover the secret key once the backdoor is known.

If a cipher is publicly claimed as potentially backdoored, it will naturally increase the watchfulness of users, even if they do not know whether there is indeed backdoored or not embedded in the primitive. In this scenario, the undetectability notion models the incapacity of a user to find any hard evidence that a backdoor indeed exists.

5.2 The MALICIOUS Framework

In this section, we introduce the MALICIOUS framework which allows to generate tweakable block ciphers that are embedded with hidden high-probability differential characteristics. This framework is based on partial non-linear layers for the internal state transformation and a tweak schedule based on an extendable-output function (XOF).

5.2.1 Block Ciphers with Partial Non-Linear Layers

SPN-based block ciphers are usually designed to apply linear layers (L_i) and non-linear layers (S_i) to the entire state at every round i . In 2013, an irregular design was suggested by Gérard *et al.* [59], where the non-linear layer is only applied to a subpart of the state at each round. We consider such design with block size n bits and partial non-linear layers of size s ($< n$) bits. Assume, without loss of generality, that the non-linear layer is always applied before the linear layer at every round. Then, we can write $f_i(x) = L_i(S_i(x^{(0)} || x^{(1)}))$ the round function f_i that transforms the state x at round i , the state being partitioned into two parts where the non-linear layer only operates on the part $x^{(0)}$ and not on the part $x^{(1)}$.

Such design allows efficient masking and thus can improve security against side-channel attacks. A concrete instantiation of this methodology named ZORRO was then proposed [59]. Even though ZORRO was rapidly broken [16, 62, 98, 112], the general design strategy continued to attract interest from the research community: in 2015, another such design LowMC was proposed [3]. Its aim was to minimize the multiplicative complexity and depth of the cipher in order to have performance advantages in certain applications, including multi-party computation (MPC), fully homomorphic encryption (FHE) and zero-knowledge proofs (ZK). After a few tweaks due to security concerns, the current version v3 of LowMC remains solid after the several third party analysis [52, 53, 99].

Compared to a full non-linear layer, a partial non-linear layer inevitably weakens the security of a cipher. One notable property is that there will exist non-trivial differential characteristics that will not activate any S-boxes over one or more rounds of the cipher. In a single round, by setting the difference on $x^{(0)}$ to be 0, there are 2^{n-s} differences of x that do not differentially activate any S-boxes. Assuming a well designed linear layer with good mixing properties, one can still expect around 2^{n-2s} differences that will also not differentially activate any S-boxes in the second round. This reasoning can be continued until no difference survives and thus the maximal expected number of rounds that a deterministic differential characteristic can cover is $\lfloor \frac{n}{s} \rfloor$. Note that this number would of course vary depending on the specificities of the linear layers.

5.2.2 The MALICIOUS Construction

Motivation. Differential and linear cryptanalysis are among the most efficient and well-understood attacks against block ciphers, both in theory and in practice. Thus, it seems natural to try creating backdoors using these techniques. Yet, there have been only a few works focusing on this research direction. For example, [6] and [87] explored backdoors in hash functions based on differential cryptanalysis. As for block ciphers, to the best of our knowledge, there is only one work from 1997 [100] using linear cryptanalysis. In that paper, special S-boxes are designed to hide high-probability linear approximations, which then enable a practical linear cryptanalysis. However, this construction was easily broken by Wu *et al.* in the subsequent year [113]. The attack against this cipher shows that the higher the probability of the embedded linear approximation, the weaker the backdoor security. Consequently, the authors claimed that it is infeasible for such a cipher to build a practical backdoor while keeping acceptable backdoor security. They further noted that “*it seems that hiding differentials is more difficult than hiding linear relations*”.

Even though other block ciphers embedding backdoors have been proposed [7, 12, 13, 39, 94], their design methodologies are usually very dedicated. On the other hand, as the topic of backdoor ciphers has not drawn much attention from the cryptography community, the backdoor security of these ciphers has not been well analyzed yet.

Considering the above facts, we introduce the MALICIOUS framework which allows to build efficient backdoors based on differential cryptanalysis. Moreover, we will show that the backdoor security can be reduced to a variation of the collision resistance notion of the XOF used in the tweak schedule.

The Construction. MALICIOUS is a framework to build a tweakable block cipher with n -bit blocksize, k -bit key and tweak of arbitrary size. It consists of three components:

- a round function f_i with partial non-linear layer, which can be expressed as $f_i(x) = L_i(S_i(x^{(0)} || x^{(1)}),$
- a tweak schedule based on a XOF,
- a key schedule.

The subtweak and subkey values are XORed only to the non-linear part of the state, but are XORed to full state at the whitening stage¹. The cipher is composed of r consecutive rounds. The framework is depicted in Figure 5.1.

The backdoor introduced by MALICIOUS are related-tweak differential characteristics with probability 1 (deterministic). With the knowledge of this backdoor, a key recovery attack can be performed using various methods

¹This is equivalent to a full state addition for all rounds, see Section 5.3.2 for details.

Figure 5.1: The MALICIOUS framework.

of differential cryptanalysis. It is to be noted that the attack is under the chosen-tweak model: both the designer and the attacker have complete freedom over the tweak values. This model is classical for TBC and realistic in practice.

We now describe how the backdoor can be embedded in the cipher. The core idea is that the subtweet difference of the backdoor chosen tweaks is used to cancel the difference of the non-linear part of the state in each round, so that the resulting differential characteristics will have no differentially active S-box (as illustrated in Figure 5.2). In Algorithm 4, we present the general steps to construct a MALICIOUS instance, in which a deterministic differential characteristic over $r_0 (\leq r)$ rounds is embedded.

Figure 5.2: Transitions of state difference in the embedded related-tweak differential characteristic. The differences of the hashed blocks can be zero or non-zero, while the differences of the white blocks are necessarily zero.

The key of the backdoor is the tweak pair generating these particular subtweet differences and the plaintext difference used in the embedded differential characteristic. We will use the prefix *malicious* to denote them.

Algorithm 4: Constructing a MALICIOUS instance with an embedded deterministic differential characteristic over r_0 rounds.

Select a XOF as the tweak schedule.

Choose uniformly at random a pair of tweak values (T_1, T_2) of arbitrary length.

Compute $t_0^1 || \dots || t_r^1 \leftarrow \text{XOF}(T_1)$ and $t_0^2 || \dots || t_r^2 \leftarrow \text{XOF}(T_2)$.

Evaluate the differences $\Delta t_i = t_i^1 \oplus t_i^2$ for all $i \in \{0, \dots, r\}$.

Randomly select a plaintext difference $\Delta P = \Delta x_0$ for the linear part $x_0^{(1)}$ and set $\Delta x_0^{(0)} = \Delta t_0^{(0)}$.

for i from 1 to r **do**

Determine a round function f_i with partial non-linear layers such that:

if $i < r_0$ **then**

Given the input difference $(\Delta x_{i-1} \oplus \Delta t_{i-1})$, the output difference after f_i has to satisfy $\Delta x_i^{(0)} = \Delta t_i$.

end

end

Output the cipher description and the r_0 -round related-tweak differential characteristic that is embedded into it (with related tweaks T_1 and T_2).

We also note that it is possible to embed multiple differential characteristics simultaneously. Then, the key recovery complexity will depend on the number of embedded differential characteristics and the cryptanalysis method.

We emphasize that the framework only focuses on the requirements of the cipher to embed a backdoor. However, a concrete instantiation would also have to take into account many other design principles so that the cipher could resist all state-of-the-art cryptanalysis as well as the attack against the backdoor described in the following section.

5.2.3 The Backdoor Security

In this section, we will evaluate two particular aspects of the backdoor security: (1) the complexity for the attacker to find the embedded differential characteristics, (2) whether additional backdoors exist in the resulting primitives, and if so, what is the complexity to find them.

Firstly, we will discuss the relation between the malicious tweak pair and its corresponding plaintext difference. We consider in this chapter that the number of rounds for the embedded differential characteristic is publicly known. On the one hand, if the malicious tweak pair is known to the attacker, then the corresponding subtweak differences can of course be computed. From these subtweak differences, he can obtain partial information about the state differences expected during the differential characteristic. Note that the embedded differential characteristic being deterministic indicates

that the transformations of state differences are linear. Hence, by reversing the linear transformations, the malicious plaintext difference can eventually be recovered. That is, the leakage of the malicious tweak pair reveals the malicious plaintext difference.

On the other hand, if the malicious plaintext difference is known to the attacker, he can compute its transformation through the linear layer and obtain the required value for the sub tweak difference such that it cancels the non-linear layer difference (since the sub tweak is only XORed to the non-linear part, there is only one such candidate), and continue this process in the following rounds. Eventually, the embedded differential characteristic will be revealed. However, it remains difficult to recover the actual malicious tweak pair due to the XOF-based tweak schedule: given the embedded related-tweak differential characteristic, finding a tweak pair that leads to it through the XOF will be difficult. We define this new security notion as *target-difference resistance*:

Definition 4 (Target-Difference Resistance). *A hash function H is target-difference resistant if it is hard to find two inputs x and y such that $H(x) \oplus H(y) = \Delta$, where Δ is a given non-zero constant.*

To better understand target-difference resistance, we introduce the limited-birthday problem, which was first proposed in [60]:

Definition 5 (The Limited-Birthday Problem [67]). *Let H be an n -bit output hash function that can be randomized by some input (IV or tweak or etc.) and that processes any input message of fixed size m bits, where $m > n$. Let IN be a set of admissible input differences and OUT be a set of admissible output differences, with the property that IN and OUT are closed sets with respect to $\oplus$ operation. Then, for the limited-birthday problem, the goal of the adversary is to generate a message pair (x, y) such that $x \oplus y \in IN$ and $H(x) \oplus H(y) \in OUT$ for a randomly chosen instance of H .*

Let 2^I and 2^O denote the sizes of IN and OUT respectively. The lower bound on the time complexity to find a solution for the limited-birthday problem is $\max(2^{\frac{n-O+1}{2}}, 2^{n-I-O+1})^2$. If I is small, the complexity is $2^{n-I-O+1}$. However, even if I is very big, the complexity cannot be below $2^{\frac{n-O+1}{2}}$.

Target-difference resistance can be seen as a special case of the limited-birthday problem (as well as a generalisation of the classical collision resistance) where OUT is limited to a single value ($2^O = 1$) and IN is the full input space. Therefore, target-difference resistance has the same generic complexity as the classical collision resistance notion, that is the birthday bound $O(2^{n/2})$.

More generally, instead of the exact malicious tweak pair, the attacker could try to find another tweak pair whose sub tweak differences are also the desired ones for the embedded differential characteristic. Yet, its complexity is still covered by the expected target-difference resistance of the XOF.

²The success probability here is about 0.63.

The above attack can possibly be applied to other plaintext differences. According to the construction of the MALICIOUS framework, the size of the input (tweak) to the XOF can be arbitrary long and thus any output of the XOF can potentially be obtained. For instance, if SHAKE128 is used as XOF, it can produce at most 2^b output streams (b being the state size between absorbing and squeezing phases in the sponge construction). Hence the number of possible sub tweaks values is bounded by 2^b , no matter how many rounds it covers, and the number of sub tweak differences is accordingly bounded by a greater value N ($\geq 2^b$). Thus, given a random plaintext difference and a certain number of rounds, if the size of the required sub tweak differences for the deterministic related-tweak differential characteristic does not exceed $\log N$, then there will be a tweak pair matching the differential characteristic. We summarize this finding as follows:

Property 3. *In addition to the embedded differential characteristics, there might exist other deterministic differential characteristics that would threaten the cipher security.*

Consequently, we have to evaluate the security of the cipher with respect to all the potential deterministic differential characteristics, not only the planned ones. We consider a MALICIOUS instance that has a key size of 128 bits and employs SHAKE128 as tweak schedule. The security strength of SHAKE128 against collision attack is $\min(l/2, 128)$ bits, where l is the output length (or the length of the colliding part). In order to recover an r_0 -round deterministic differential characteristic, the attacker has to find a tweak pair whose sub tweak differences are the desired ones. The total size of these sub tweak differences is $n + s \cdot (r_0 - 1)$ bits and thus the generic attack complexity is $2^{\min((n+s \cdot (r_0-1))/2, 128)}$, which becomes 2^{128} when $(n + s \cdot (r_0 - 1))/2 \geq 128$. The analysis is similar for the case where the key size is 256 bits and SHAKE256 is employed. We define r' to represent the value of r_0 that turns this inequality into an equality:

$$(n + s \cdot (r' - 1))/2 = k \quad (5.1)$$

All the deterministic related-tweak differential characteristics smaller than r' rounds can be recovered with a complexity smaller than the actual key size. Therefore, in order to prevent these differential characteristics to weaken the cipher, r' must be taken into consideration when determining the number of rounds of the MALICIOUS instance. Actually, these related-tweak differential characteristics will decay exponentially in the remaining rounds as the corresponding sub tweak differences are basically random.

5.2.4 Rationale Underlying the MALICIOUS Construction

When designing a backdoor for block ciphers, the first question that comes into mind is probably *what type of backdoor should be used ?* While some

existing backdoor designs directly insert a backdoor inside S-boxes or some other parts of the round function, we found out that the additional input tweak capability of a tweakable block cipher could be a perfect carrier of the backdoor. Suppose that a tweakable block cipher has a special property only when it is initiated with very specific tweak values, while it performs normally for all the other tweak values, then this property could be used as a backdoor. Moreover, if the tweak size is large enough, finding these special tweak values could be as hard as finding the secret key in the ideal case. One straightforward example of the special property is to build related-tweak differential characteristics using these tweaks. In the following, we provide more in-depth explanations on the design choices in **MALICIOUS**.

Components Rationale

When instantiating the **MALICIOUS** framework, some (security) notions have to be taken into account. The first and most important one is the undiscoverability: an entity who does not know the backdoor should not have increased chances to break the cipher. This requires that the backdoor security has to be as high as the cipher security. Thus, the **MALICIOUS** framework should provide a valid and solid security evaluation for the backdoor.

Another important notion is the practicability of the backdoor, and we will aim to make it as efficient as possible.

We detail in the following how the components of the **MALICIOUS** framework do follow these principles.

Tweak Schedule Based on XOF. As the malicious tweak values are the backdoor, the main task of the tweak schedule is to protect the malicious tweaks. According to the security analysis from Section 5.2.3, the backdoor security relies on the target-difference problem, where the attacker tries to find a tweak pair whose sub tweak differences are the desired ones. This notion is simply a variation of the classical collision resistance for a hash function, so we expect a good cryptographic hash function to naturally provide this resistance.

Since **MALICIOUS** is a generalized framework, the total number of rounds will vary according to the different instantiations, so does the length of the sub tweaks. Hence, the output length of the tweak schedule is expected to be flexible. Besides, if the tweak schedule was designed specifically for each **MALICIOUS** instantiation, it will render the backdoor evaluation much more difficult. Thus, for sake of simplicity of the analysis, it seems a better idea to make the tweak schedule uniform in the framework.

For all these reasons, a XOF seemed to be the best choice for our tweak schedule. The security of actual XOF functions such as **SHAKE128** or **SHAKE256** is rather well-analyzed and it can provide many choices in terms of security level.

Partial Non-Linear Layers. The probability of a differential characteristic is determined by the number of differentially active S-boxes. Hence, in order to embed an efficient backdoor based on a differential characteristic, the best case is that the differential characteristic activates no S-box at all. This is obviously very unlikely to happen in the MALICIOUS framework if the round functions are fully non-linear layers. Indeed, unless the related-key model is considered, a non-zero difference inserted in the plaintext would have to be cancelled by the first sub tweak difference. However, when inserting differences in the tweak input, as the sub tweak differences produced by the XOF will be random, they will force many active S-boxes in the subsequent rounds. Thus, it is unlikely for the MALICIOUS framework to be able to embed a deterministic related-tweak differential characteristic that covers more than a few rounds if full non-linear layers are utilized. Of course, it is possible to construct a differential characteristic with limited number of active S-boxes, but this is not the efficiency we are targeting.

We have also tried to modify the framework such that the sub tweak addition is not performed every round. For example, an r rounds deterministic related-tweak differential characteristic can be realized by applying the tweak addition only once at the beginning, see Figure 5.3. This way, the sub tweak difference Δt_0 could neutralize the plaintext input difference Δx_0 and the resulting zero difference would get through the r rounds with probability 1. However, this candidate has an obvious fatal flaw: for any tweak pair the attacker can always set the plaintext input difference to be equal to Δt_0 .

Figure 5.3: A defective variant of the MALICIOUS framework. Key addition is omitted.

The above analysis shows that full non-linear layers seem not suitable for the MALICIOUS framework. On the contrary, partial non-linear layers satisfy our requirements. As in that case the S-box only applies to a part of the internal state, the round function is able to map a non-zero input difference to a non-zero output difference while no active S-box is activated. In term of building deterministic differential characteristics, we only have to set the difference of the non-linear part of the internal state to be zero rather than the full state. This allows to choose the linear transformation so that the output difference could satisfy the requirements from Algorithm 4.

5.3 Instantiating the MALICIOUS Framework with LowMC

In this section, we introduce a concrete instantiation of the MALICIOUS framework, called LowMC-M, which is based on the family of block ciphers LowMC.

5.3.1 LowMC

LowMC [3] is a family of block ciphers based on SPN structure with partial non-linear layers. The parameters are flexible and we denote the block size by n , the key size by k , the number of S-boxes applied each round by m and the maximum allowed data complexity by d (d is the $\log_2$ of the allowable data complexity up to which the cipher is expected to give the claimed security). In order to reach the security claims, the number of rounds r is then derived from all these parameters using a round formula, the latest version being given in [99].

At the beginning of the encryption process, a key whitening is performed. The round function at round i consists of four operations in the following order:

- **SboxLayer**. A 3-bit S-box is applied in parallel on the $s = 3m$ LSBs of the state, while the transformation for the remaining $n - s$ bits is the identity.
- **LinearLayer**(i). The state is multiplied in $\text{GF}(2)$ with an invertible $n \times n$ binary matrix L_i which is chosen independently and uniformly at random.
- **ConstantAddition**(i). The state is XORed with an n -bit round constant C_i which is chosen independently and uniformly at random.
- **KeyAddition**(i). The state is XORed with an n -bit round key k_i . To generate k_i , the master key K is multiplied in $\text{GF}(2)$ with an $n \times k$ binary matrix KL_i . This matrix is chosen independently and uniformly at random with rank $\min(n, k)$.

5.3.2 Equivalent Representation of LowMC

As discussed in [51, 72, 99], round keys and constants in LowMC can be compressed due to the fact that the non-linear layer is partial.

In the round function, it is possible to exchange the order of consecutive linear operations. We swap the order of **LinearLayer** and **KeyAddition** operations while keeping **ConstantAddition** as the last step in round i . Then, the equivalent round key can be written as $k'_i = L_i^{-1}(k_i)$. We observe that the S-box only operates on the first s bits of the state and does not change the rest of the $n - s$ bits. Thus, we split k'_i into $k_i^{(0)}$ and $k_i^{(1)}$, and we

5.3. INSTANTIATING THE MALICIOUS FRAMEWORK WITH LOWMC101

can move the addition of $k_i'^{(1)}$ to the beginning of the round. Next, we observe that $k_i'^{(1)}$ can move further up to be combined with k_{i-1} in the previous round. The procedure is illustrated in Figure 5.4. In general, if we start from the last round and iterate this procedure recursively until all the additions to the linear part have been moved to the beginning of the algorithm, we will end up with an equivalent representation where all the round keys are reduced to s bits apart from the whitening key. We remark that the same reasoning can be applied to the round constants.

This optimized representation can also reduce the implementation cost of the key schedule. Since all transformations performed during the optimization are linear and since the key schedule is itself linear, these transformations can be composed with the key schedule in order to compute the new $3m$ -bit round keys directly. We refer to [51] for details.

Figure 5.4: Simplified representation of LowMC.

5.3.3 LowMC-M

We will directly use the simplified representation of LowMC as a starting point in our design, with a further modification: we move **LinearLayer** behind **SboxLayer** in every round³.

LowMC-M is a family of tweakable block ciphers built upon LowMC with an additional transformation in each round:

- **TweakAddition**(i). The non-linear part of the state is XORed with an s -bit sub tweak t_i just after **KeyAddition**. t_i is generated from a XOF whose input is the original tweak value T .

The XOF is based on SHAKE128 or SHAKE256, depending on the key size. All the other transformations of the round function are the same as for LowMC. The round function is finally composed of the following operations:

$$\text{TweakAddition}(i) \circ \text{KeyAddition}(i) \circ \text{ConstantAddition}(i) \circ \text{LinearLayer}(i) \circ \text{SboxLayer}$$

The encryption starts with a key and tweak whitening and the sizes of k_0 and t_0 are both n . The derivation formula for the number of rounds r is the same as for LowMC.

³The resulting primitive is an equivalent representation of a LowMC instantiation with different linear layers, key schedule and round constants, because these components are chosen randomly.

Figure 5.5: A single round of LowMC-M.

Notations for LowMC-M

Given a bitstring x , we denote by $x[i]$ its i -th bit, counting from the least significant bit (LSB). During a differential cryptanalysis, we denote by X_i the i -th round state difference before the **LinearLayer** transformation. Given a matrix L_i , we denote its j -th row by $L_i[j, *]$, and partition L_i into four sub-matrices:

$$L_i = \left[\begin{array}{c|c} L_i^{00} & L_i^{01} \\ \hline L_i^{10} & L_i^{11} \end{array} \right]$$

where $L_i^{00} \in \text{GF}(2)^{s \times s}$, $L_i^{01} \in \text{GF}(2)^{s \times (n-s)}$, $L_i^{10} \in \text{GF}(2)^{(n-s) \times s}$, $L_i^{11} \in \text{GF}(2)^{(n-s) \times (n-s)}$. With this notation, L_i^{00} and L_i^{01} will map $X_i^{(0)}$ and $X_i^{(1)}$ to the non-linear part of the state, respectively. And L_i^{10} and L_i^{11} will map $X_i^{(0)}$ and $X_i^{(1)}$ to the linear part of the state, respectively.

5.3.4 Embedding a Backdoor into LowMC-M

There are many forms of differential cryptanalysis that can perform a key recovery attack, such as the impossible differential attack, the boomerang attack, etc. For LowMC-M, we use the plain version where the attacker can deduce full or partial information about the r -th round key from a differential characteristic over $r - 1$ rounds.

Since an $(r - 1)$ -round deterministic differential characteristic can only reveal the s -bit subkey k_r of the r -th round, more deterministic differential characteristics should be added in order to eventually recover the full key. After k_r has been retrieved, the cipher can be reduced to $r - 1$ rounds and thus another s -bit subkey k_{r-1} can be recovered from an $(r - 2)$ -round deterministic differential characteristic. Finally, assume that there are a total of a such deterministic differential characteristics embedded in LowMC-M (one on $r - 1$ rounds, one on $r - 2$ rounds, etc., see Figure 5.6), then $a \cdot s$

5.3. INSTANTIATING THE MALICIOUS FRAMEWORK WITH LOWMC103

subkey bits can be recovered. As the key schedule is fully linear and each matrix inside the key schedule is generated independently and uniformly at random, it implies that one will recover $a \cdot s$ bits of information about the key by solving a system of linear equations. Therefore, at most $a = \lceil k/s \rceil$ deterministic differential characteristics are needed to recover the full key.

Figure 5.6: The deterministic differential characteristics embedded into LowMC-M.

Now, we explain how to embed such differential characteristics into an instantiation of **LowMC-M**. The general procedure is given in Algorithm 4. The a malicious tweak pairs are chosen by the designer at the very beginning and the corresponding subtweak differences are computed. Then, the linear layer matrix L_i is generated along with the generation of the deterministic differential characteristics, **round by round**.

Firstly, we explain how to generate the linear layer matrices. Note that in order to have a deterministic differential characteristic over i rounds, only the linear layer matrices of the first $i - 1$ rounds have to be specifically designed as the matrix L_i has no impact on the differences of the i -th round S-boxes. Assuming we have already embedded a deterministic differential characteristics over i rounds, then all the linear layer matrices of the first $i - 1$ rounds of **LowMC-M** have been fixed accordingly. If we plan to extend b ($b \leq a$) of the a deterministic differential characteristics by one more round, the matrix L_i should be specified. Denote by SX_i the set of $X_i^{(1)}$ of those deterministic differential characteristics that will be extended in the next round. Here, SX_i refers to the b differential characteristics. Since the non-linear state difference $X_i^{(0)}$ equals to zero for all the b differential characteristics, the set SX_i will determine the differential in the following round. Given the difference set SX_i , the output differences after the multiplication by the matrix L_i^{01} should cancel the following subtweak differences so that the b differential characteristics will activate no S-box in round $i + 1$. We detail the generation of L_i in Algorithm 5.

Denote the $b \times (n - s)$ matrix in Equation (5.2) by MX_i . We emphasize that the rank of MX_i should be $\min(b, n - s)$, otherwise Equation (5.2) is likely to have no solution. In practice, b is always smaller than $n - s$ for a normal parameters set of **LowMC-M**. Thus, this requirement also means that the binary vectors of $X_i^{(1)}$ in SX_i should be linearly independent.

The whole process of generating an instance of **LowMC-M** is given here:

Algorithm 5: Generate linear layer matrix L_i .

Input : The set $SX_i = (X_{i,1}^{(1)}, X_{i,2}^{(1)}, \dots, X_{i,b}^{(1)})$ and the subweak differences $(\Delta t_i^1, \Delta t_i^2, \dots, \Delta t_i^b)$ for the b differential characteristics.

Output : Matrix L_i

while *True* **do**

for j *from* 1 *to* s **do**

 Solve the following system of linear equations and randomly pick one solution of $x = (x_1, x_2, \dots, x_n)$ as $L_i^{01}[j, *]$.

$$\begin{pmatrix} X_{i,1}^{(1)}[1] & X_{i,1}^{(1)}[2] & \dots & X_{i,1}^{(1)}[n-s] \\ X_{i,2}^{(1)}[1] & X_{i,2}^{(1)}[2] & \dots & X_{i,2}^{(1)}[n-s] \\ & & \ddots & \\ X_{i,b}^{(1)}[1] & X_{i,b}^{(1)}[2] & \dots & X_{i,b}^{(1)}[n-s] \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-s} \end{pmatrix} = \begin{pmatrix} \Delta t_i^1[j] \\ \Delta t_i^2[j] \\ \vdots \\ \Delta t_i^b[j] \end{pmatrix} \quad (5.2)$$

end

 Randomly select the sub-matrices L_i^{00}, L_i^{10} and L_i^{11} .

if L_i *is full rank* **then**

 | return L_i

end

end

1. Select a different pairs of tweaks of any desired length and compute the corresponding subweak differences in all rounds for each pair of tweaks.
2. For each tweak pair, choose an n -bit value of the plaintext difference ΔP as the input difference for the embedded differential characteristic, while setting the first s bits of ΔP to be equal to $\Delta t_0^{(0)}$.
3. For the a differential characteristics, compute $X_1^{(1)} = \Delta P^{(1)} \oplus \Delta t_0^{(1)}$ and if the binary vectors of SX_1 are not linearly independent, then go back to step 2.
4. For round i from 1 to $r - 2$:
 - Generate the matrix L_i using Algorithm 5 with SX_i and the corresponding subweak differences as inputs⁴.
 - Except for the last loop, compute the set of SX_{i+1} through the matrix multiplication of L_i . If the binary vectors of SX_{i+1} are not linearly independent, repeat this loop.
5. Choose L_{r-1} and L_r independently and uniformly at random from all invertible $n \times n$ binary matrices.

⁴Starting from round $r - a + 1$, the number of deterministic differential characteristics decrements by 1 at every loop.

6. For all rounds i , choose KL_i independently and uniformly at random from all $n \times k$ binary matrices of rank $\min(n, k)$ and the round constants C_i as well.

Recovering the Secret Key With the Backdoor. The backdoor is the a malicious tweak pairs and the corresponding plaintext differences. With the knowledge of these related-tweak differential characteristics, the designer can recover the full key in a very short time. To create the a plaintext differences, the designer can firstly choose a random P , then compute $P_i = P \oplus \Delta P_i$ for $i \in \{1, \dots, a\}$. We note the fact that for any non-zero probability differential (Δ_1, Δ_2) of LowMC-M S-box, where $\Delta_1 \neq 0$ and $\Delta_2 \neq 0$, there is only one unordered pair of inputs/outputs of the S-box satisfying the differential. If each plaintext difference is used only once in the attack, then two subkey candidates will remain for each S-box as we cannot determine which order of the input/output pair of the targeted S-box should be in the attack. The wrong subkey candidate can be filtered by repeating the attack with another pair of plaintexts of the same difference. By doing so, $a \cdot s$ bits of information of the key can be retrieved in the end. Later, the remaining $(k - a \cdot s)$ key bits, if they exist, can be brute-forced. Finally, the key recovery requires $2(a + 1) + \max(k - a \cdot s, 0)$ encryptions and the data complexity is $2(a + 1)$.

Note that the bit length of $X_i^{(1)}$ is $n - s$. In order to ensure that Equation (5.2) is solvable, the number of differential characteristics that are embedded in LowMC-M should not be higher than $n - s$. Generally, this bound is much higher than the number of differential characteristics that is actually needed in a concrete instantiation. Last but not least, one may wonder why we chose different malicious tweak pairs for the a related-tweak differential characteristics (indeed using a single malicious tweak pair would work), but we recommend doing so for security reasons as we will explain in Section 5.4.1.

5.3.5 Parameters

The design goal of LowMC-M is to keep the backdoor and the cipher secure, but also to ensure the efficiency of the key recovery using the backdoor. Based on these principles, we selected some instantiation parameters⁵ and we present them in Table 5.1. The security analysis is given in Section 5.4 and Section 5.5.

Regarding the performances, we evaluated the corresponding LowMC used in the LowMC-M instances. The LowMC implementations we benchmarked are optimized for AVX2 instructions. Measurements were performed on an AMD EPYC 7401 running Ubuntu 18.04. We tested several instances and we observed that a single encryption generally costs around 10000 to 30000 cycles depending on the parameters, the block size (= key size) ranging from 128 to 256 bits.

⁵The reference code of LowMC-M generation can be found at <https://doi.org/10.21979/N9/X0P4YG>.

Table 5.1: A range of different parameters sets of **LowMC-M** instantiations. For each instantiation, the malicious tweak pair that triggers each embedded differential characteristic is unique. d is the $\log_2$ of the allowed data complexity, a is the number of differential characteristics embedded.

Block size n	Non-linear s	Key size k	Data d	Rounds r	#Differentials a	XOF
128	3	128	64	208	43	SHAKE128
	6	128	64	104	21	SHAKE128
	9	128	64	70	14	SHAKE128
	30	128	64	23	5	SHAKE128
	90	128	64	14	2	SHAKE128
256	3	256	64	384	85	SHAKE256
	9	256	64	129	28	SHAKE256
	60	256	64	21	5	SHAKE256
	120	256	64	14	3	SHAKE256

5.4 Backdoor Security

In this section, we will discuss the backdoor security of **LowMC-M** with respect to the notions mentioned in Section 5.1.2: undetectability, undiscoverability, untraceability and practicability.

5.4.1 Undetectability

In this subsection, we discuss whether a **LowMC-M** instance containing a backdoor is distinguishable from a random **LowMC-M** with no backdoor embedded. Since the only difference between these two cases lies in the way the linear layer matrices are generated, we will investigate the properties of these matrices.

We now would like to show that all embedded differential characteristics must use distinct tweak pairs in order to maintain undetectability. Assuming there is a backdoored **LowMC-M** instance that is generated following the steps described in Section 5.3.4 and a total of a deterministic related-tweak differential characteristics are embedded, while only a' ($< a$) different tweak pairs are used during the generation phase. Let c_j denote the number of embedded differential characteristics triggered by the same tweak pair, with $j \in \{1, \dots, a'\}$. We will show that some dependency will exist in the linear layer matrices for the first i ($\leq r - a$) rounds, consequently some additional deterministic related-tweak differential characteristics over the first i rounds can be recovered.

Definition 6. For a **LowMC-M** instance, A_i is the matrix of dimension

$(i \cdot s) \times (n - s)$ defined as:

$$\begin{pmatrix} L_1^{01} \\ L_2^{01} \cdot L_1^{11} \\ L_3^{01} \cdot L_2^{11} \cdot L_1^{11} \\ \vdots \\ L_i^{01} \cdot L_{i-1}^{11} \cdot \dots \cdot L_1^{11} \end{pmatrix}$$

We remark that a malicious plaintext difference ΔP can be retrieved if the corresponding malicious tweak pair is provided: in order to have a deterministic differential characteristic all S-boxes must be differentially inactive (*i.e.*, the input difference of each S-box should be zero) and thus for a malicious tweak pair that takes any of the a' different values, recovering $\Delta P^{(0)}$ (the non-linear part of ΔP) is straightforward as it is equal to the sub tweak difference $\Delta t_0^{(0)}$. After that, one just needs to retrieve the remaining part $\Delta P^{(1)}$. In order to have a deterministic differential characteristic over the first two rounds, $L_1^{01}(X_1^{(1)})$ should be equal to Δt_1 , where $X_1^{(1)} = \Delta P^{(1)} \oplus \Delta t_0^{(1)}$. To extend the differential characteristic to the third round, $L_2^{01} \cdot L_1^{11}(X_1^{(1)})$ should be equal to Δt_2 . Continuing this process until the i -th round, we can create a system of linear equations with $n - s$ binary variables:

$$\begin{pmatrix} L_1^{01} \\ L_2^{01} \cdot L_1^{11} \\ L_3^{01} \cdot L_2^{11} \cdot L_1^{11} \\ \vdots \\ L_{i-1}^{01} \cdot L_{i-2}^{11} \cdot \dots \cdot L_1^{11} \end{pmatrix} \cdot (X_1^{(1)}) = A_{i-1} \cdot (X_1^{(1)}) = \begin{pmatrix} \Delta t_1 \\ \Delta t_2 \\ \Delta t_3 \\ \vdots \\ \Delta t_{i-1} \end{pmatrix} \quad (5.3)$$

Solving Equation (5.3) will output the solution of $X_1^{(1)}$, then the remaining part $\Delta P^{(1)}$ can be recovered naturally. However, there may be more solutions as the number of solutions is determined by the rank of A_{i-1} .

In cases where the number of rounds i is large enough such that $(i-1) \cdot s \gg (n-s)$, if all the linear layer matrices are chosen independently and uniformly at random, the rank of A_{i-1} will be $n-s$ with very high probability. However, for a **LowMC-M** instance with backdoor embedded, since the linear layer matrices are specially designed, the rank of A_{i-1} can not be determined similarly.

Determining the Rank of A_{i-1}

We first introduce the following definition.

Definition 7. If M is an $n \times m$ binary matrix and v is an n -bit vector, the solution space $\text{sol}(M, v)$ is defined as: $\text{sol}(M, v) = \{x^T \in \{0, 1\}^m : Mx = v\}$.

Assume that a special **LowMC-M** instance is generated with c related-tweak deterministic differential characteristics over i rounds while only one malicious tweak pair is used. During the generation of L_j^{01} , $j \in \{1, \dots, i-1\}$, Equation (5.2) could be simplified as:

$$MX_j \cdot x = \vec{1} \text{ or } MX_j \cdot x = \vec{0}, \quad (5.4)$$

where $\vec{0}$ and $\vec{1}$ are c -bit vectors full of zeros and ones, respectively.

Denote by V the union of $\text{sol}(MX_1, \vec{1})$ and $\text{sol}(MX_1, \vec{0})$, the rows of L_1^{01} are chosen from V . Since the dimensions of $\text{sol}(MX_1, \vec{1})$ and $\text{sol}(MX_1, \vec{0})$ are both $n - s - c$, then the dimension of V is $n - s - c + 1$. When $j = 2$, Equations (5.4) can be represented by:

$$MX_1 \cdot (L_1^{11})^T \cdot x = \vec{1} \text{ or } MX_1 \cdot (L_1^{11})^T \cdot x = \vec{0},$$

because $X_2^{(1)} = L_1^{11} \cdot X_1^{(1)}$. The rows of L_2^{01} are chosen from $\text{sol}(MX_1 \cdot (L_1^{11})^T, \vec{1})$ or $\text{sol}(MX_1 \cdot (L_1^{11})^T, \vec{0})$. Before we continue, we will use the following lemma.

Lemma 4. *Let M_1 and M_2 be two binary matrices of dimension $(n \times m)$ and $(m \times m)$ respectively. If $x \in \text{sol}(M_1 \cdot M_2, v)$, then $x \cdot M_2^T \in \text{sol}(M_1, v)$ for any n -bit vector v .*

Proof. For any $x \in \text{sol}(M_1 \cdot M_2, v)$, we have $(M_1 \cdot M_2) \cdot x^T = v$. It can be represented by $M_1 \cdot (M_2 \cdot x^T) = v$, thus $(M_2 \cdot x^T)^T = x \cdot M_2^T \in \text{sol}(M_1, v)$. $\square$

According to Lemma 4, if $x \in \text{sol}(MX_1 \cdot (L_1^{11})^T, \vec{1})$, then $x \cdot L_1^{11} \in \text{sol}(MX_1, \vec{1})$ and also if $x \in \text{sol}(MX_1 \cdot (L_1^{11})^T, \vec{0})$, then $x \cdot L_1^{11} \in \text{sol}(MX_1, \vec{0})$. Thus, all the rows of $L_2^{01} \cdot L_1^{11}$ are in the space V . Similarly, we can get the same results for $L_3^{01} \cdot L_2^{11} \cdot L_1^{11}, \dots, L_{i-1}^{01} \cdot L_{i-2}^{11} \cdot \dots \cdot L_1^{11}$. To summarize, all the rows of A_{i-1} for this special **LowMC-M** instance are chosen from the space V of dimension $n - s - c + 1$. Thus, the rank of A_{i-1} is $n - s - c + 1$.

Let us return back to the previous **LowMC-M** instance mentioned at the beginning of this subsection. We can divide the a differential characteristics into a' sub-groups where each sub-group includes c_j differential characteristics that are triggered with the same tweak pair, $j \in \{1, \dots, a'\}$. Then, the space V will be the intersection of all the spaces that are determined by the a' sub-groups. We summarize the result as follows.

Proposition 1. *If there is a total of a' different malicious tweak pairs and each of them is used to build c_j deterministic differential characteristics over i rounds in an instance of **LowMC-M**, with $(i-1) \cdot s \gg (n-s)$, then the rank of A_{i-1} will be $n - s - \sum_{j=1}^{a'} (c_j - 1)$.*

As a result, the rank of A_{i-1} is $n - s - \sum_{j=1}^{a'} (c_j - 1)$ and a total of $2^{\sum_{j=1}^{a'} (c_j - 1)}$ deterministic differential characteristics for each of the a' tweak pairs can be recovered by the designer. Note that the rank of A_{i-1} can be easily computed by any entity. Compared to the full rank A_{i-1} for a random

LowMC-M with no backdoor embedded, the unusual property of A_{i-1} for the backdoored **LowMC-M** will uncover the existence of the backdoor if $a' < a$. However, if $a' = a$, that is, $c_j = 1$ for all $j \in \{1, \dots, a'\}$, then A_{i-1} will be full rank. Therefore, in order to keep the backdoor of **LowMC-M** undetectable, we recommend to not use the same tweak pair for building more than one differential characteristics in the generation phase.

5.4.2 Undiscoverability

In this subsection, we discuss whether the backdoor from a **LowMC-M** instance can be efficiently recovered by an attacker. Recall that some unknown deterministic related-tweak differential characteristics potentially exist in **LowMC-M**, according to Property 3. Instead of considering the embedded backdoor exclusively, we evaluate the complexity of finding any useful deterministic related-tweak differential characteristics for an attacker. Basically, the complexity is based on the XOF security properties.

We simply adopt the security analysis for the general **MALICIOUS** framework in Section 5.2.3. For any **LowMC-M** instance, the bound r' derived from Formula 5.1 is much smaller compared to the total number of rounds, which poses no threat to the backdoor. We list the evaluation for some instances in Table 5.2.

We can examine the undiscoverability security from another perspective. Note that deterministic related-tweak differential characteristics can be derived as long as Equation (5.3) is solvable. The requirement for the equation to be solvable is that the ranks of the coefficient matrix A_{i-1} and the augmented matrix of Equation (5.3) are equal, which means that the vector on the right side of the equation, denoted as v , has to be a combination of the columns of A_{i-1} . Observe that the number of such combinations is 2^α , α being the rank of A_{i-1} and it can be computed according to Proposition 1. As for vector v , it is random due to the XOF and its size is $s \cdot (i - 1)$. In conclusion, Equation (5.3) is solvable with probability $2^{\alpha - s \cdot (i-1)}$, that is, the complexity of finding an i -round deterministic related-tweak differential characteristic is $2^{s \cdot (i-1) - \alpha}$. We define r'' to represent the value of i that turns the complexity to be equal to the key space size

$$r'' = \frac{k + \alpha}{s} + 1 \quad (5.5)$$

The maximal value is $r'' = \frac{k+n}{s}$ when A_{i-1} is full rank of $n - s$. Still, r'' is much smaller than the number of rounds of any **LowMC-M** instance, see examples in Table 5.2.

To summarize, the backdoor and the other potential deterministic related-tweak differential characteristics of the same length are fully protected by the XOF, and its recovery is as hard as brute-forcing the key.

Table 5.2: Backdoor security evaluation for **LowMC-M- n/s** with block size n , key size n , non-linear layer size s and $\log_2$ data complexity 64. r is the actual number of rounds of the instance, r' and r'' are defined in Formulas 5.1 and 5.5 respectively.

Parameters	r	r'	r''
LowMC-M-128/3	208	44	86
LowMC-M-128/6	104	23	43
LowMC-M-128/9	70	16	29
LowMC-M-128/30	23	6	9
LowMC-M-128/90	14	3	3
LowMC-M-256/3	384	87	170
LowMC-M-256/9	129	30	57
LowMC-M-256/60	21	6	9
LowMC-M-256/120	14	4	5

5.4.3 Untraceability and Practicability

As for practicability, only negligible data and computation are required to launch a full key recovery attack with the knowledge of the backdoor, as explained in Section 5.3.4. Thus, the full key can be recovered within seconds.

Since the usage of the backdoor requires chosen tweaks, the malicious tweaks can be detected by the user once the designer makes queries to attack him, which means the backdoor is traceable. Besides, as only a few queries are needed to launch an attack with the knowledge of the backdoor, the user is able to quickly brute-force the queries to find out the malicious tweak pairs.

5.5 Cipher Security

In this section, we study the security of **LowMC-M** as a tweakable block cipher.

5.5.1 Attacks Based on the Tweak

In comparison to **LowMC**, an additional tweak addition is introduced in **LowMC-M**. Theoretically, this feature will provide extra degrees of freedom for the attacker and might naturally weaken **LowMC-M** when compared to **LowMC**. However, since the tweak schedule is an XOF, the attacker cannot control its output. Even if the attacker could brute-force some structures on the subtweaks for a few rounds, this will result in the remaining rounds containing completely random structure, which consequently prevent

the attacker utilizing these remaining rounds for what should have been the best attack on **LowMC**. Hence, we believe that the extra degrees of freedom provided by the tweak is not easily usable and will not lead to any important improvement over classical attack, including the existing cryptanalysis [52, 53, 99] on **LowMC**.

5.5.2 Attacks without Using the Tweak

All the current attacks [52, 53, 99] on **LowMC** have been conducted under the assumption that the linear layer matrices of **LowMC** are chosen independently and uniformly at random. Except the tweak addition, **LowMC-M** has the equivalent specification to **LowMC**. The only difference lies in the way the linear layer matrices L_i are chosen during the generation phase. In order to prove that the security of **LowMC-M** is on par with that of **LowMC**, we need to show that the linear layer matrices of **LowMC-M** are indistinguishable from those of **LowMC** from the perspective of the attacker. We will evaluate this with respect to the randomness and independence.

Randomness of Linear Layer Matrices

The randomness of the linear layer matrix L_i is analyzed by scrutinizing its four sub-matrices one by one.

L_i^{00} and L_i^{10} . As described in Algorithm 5, the two sub-matrices L_i^{00} and L_i^{10} of L_i are chosen independently and uniformly at random for each round.

L_i^{11} . Even though L_i^{11} is chosen randomly in Algorithm 5, there is a supplementary requirement during the generation phase. That is, the binary vectors of SX_{i+1} have to be linearly independent, which adds an extra constraint to L_i^{11} since each binary vector of SX_{i+1} is obtained by:

$$X_{i+1}^{(1)} = L_i^{11} \cdot X_i^{(1)} \quad (5.6)$$

and thus the transformation of L_i^{11} should map a set of linearly independent vectors to another set of linearly independent vectors. Since L_i^{11} is chosen randomly and all the $X_i^{(1)}$ involved are linearly independent, every $X_{i+1}^{(1)}$ in SX_{i+1} produced by Formula 5.6 can be regarded as random binary vectors and are independent from each other. On the other hand, note that at most $a = \lceil k/s \rceil$ differential characteristics are embedded in **LowMC-M**, which means that the size of SX_{i+1} is $\lceil k/s \rceil$ at most. For any reasonable parameter set, we will have $\lceil k/s \rceil \ll (n - s)$. Based on Lemma 5 below, we can compute the probability that the set SX_{i+1} is linearly independent. As a result, the probability is almost 1, which is also verified from our experiments.

Hence, the constraint on L_i^{11} is very loose. The final selection of L_i^{11} will not introduce any special property.

Lemma 5. [77, adapted] For m random n -bit vectors over $\mathbb{F}_2$ ($m \leq n$), the probability that they are linearly independent is $p(m) = \prod_{k=0}^{m-1} (1 - 2^{k-n})$. In particular, $p(n) > 0.2887$.

L_i^{01} . L_i^{01} is the essential part for embedding backdoors, and thus it is the one specially designed. The row length of L_i^{01} is $n - s$ bits, while in the generation phase each row is chosen from a sub-space of dimension $n - s - b$ which is determined by the corresponding Equation (5.2), b being the size of SX_i . However, we will show that for the attacker this special chosen L_i^{01} is still indistinguishable from a randomly chosen one.

Observe that both MX_i and the subtweak difference vector in Equation (5.2) are unknown for the attacker, thus the solution space is unidentified. Moreover, the solution space for each row of L_i^{01} could be different due to the subtweak difference. Therefore, it is impossible for the attacker to trace some rows of L_i^{01} to the targeted hidden sub-space.

To summarize, the four sub-matrices are indistinguishable from random matrices for the attacker. The only connection between these four sub-matrices is that the combined matrix L_i should be invertible, which is also the same for **LowMC**, so it reveals no additional information. Hence, we conclude that for the attacker the matrix L_i is indistinguishable from a random matrix.

Independence between Linear Layer Matrices

The definition of A_r captures partial information of the matrices that includes L_i^{10} and L_i^{11} over r consecutive rounds. If the linear layer matrices are chosen independently and uniformly at random, the resultant A_r should be random, thus the rank of A_r will be $n - s$ when $r \cdot s \gg (n - s)$. If the rank for a **LowMC-M** instance is smaller than $n - s$, it will imply a connection between these matrices. As suggested in Proposition 1, the rank of A_r can be computed by $n - s - \sum_{i=1}^{a'} (c_i - 1)$. In order to eliminate the connections, each c_i should equal to 1, that is, different malicious tweak pairs should be used to build different differential characteristics during the generation phase.

The two sub-matrices L_i^{00} and L_i^{10} are chosen randomly and independently, so it will not impose any connection between the matrices.

We remark that even if there is some dependence existing between the linear layer matrices, the cipher security is still unlikely to be threatened. Yet, we conservatively recommend to avoid such dependency in a **LowMC-M** instance.

5.6 Conclusion

In this chapter, we proposed the **MALICIOUS** framework for embedding backdoors into tweakable block ciphers. The backdoor is a set of related-tweak differential characteristics with probability 1, from which the secret key can be recovered fully and efficiently. Besides, the backdoor security

of our proposal is reduced to the target-difference resistance (a variant of the classical collision resistance, with the same generic complexity) of the XOF employed in the cipher. We also proposed several concrete instances **LowMC-M**, which are directly inspired from the block cipher **LowMC**.

We have proved that it is possible to build a secure and efficient backdoor into tweakable block ciphers. Third party analysis is of course required to fully understand its security, but our proposal could be a new interesting direction towards building backdoors in symmetric-key primitives.

Not only this result will increase the community's awareness to potential backdoors in symmetric-key primitives, but it can also lead to new applications. It has been shown in [100] that a backdoored block cipher is equivalent to a public key encryption where the backdoor is regarded as the secret key. Even though our proposal does not yet reach the usability of a public encryption scheme, building public-key primitives out of symmetric-key ones has been a long standing open problem.

Chapter 6

Conclusion

In this thesis, we presented several improved cryptanalytic techniques and their applications to symmetric-key algorithms, including AES, Deoxys-BC and the hashing modes of AES and Kiasu-BC. As a result, we proposed a new related-key attack on 10-round AES-256 and optimized the evaluation on the previous attacks against AES-192 and Deoxys-BC. We also improved the full-round related-key attack on AES-256 by reducing time and data complexities. Lastly, we improved the 7-round preimage attacks on AES hashing modes and extended the attacks to 8-round for the first time. On the other hand, we proposed the framework MALICIOUS for building tweakable block ciphers embedded with a backdoor, which is differential in nature. Based on this framework, a concrete instantiation LowMC-M was constructed

As a continuation of the research in this thesis, there are some interesting directions for future works:

- On the study of boomerang switch, our method mainly deals with the switching effect in two rounds. Actually, the switching effect can affect much more rounds in some cases. In that situation, the analysis seems to become difficult. Thus, it will be desirable to develop a more efficient method to measure the switching effect in more rounds.
- We have applied the structure technique to the key, but this will increase the number of required keys in an attack, which is not a desirable feature in practice. As the tweak plays a similar role to key and is easier to control, it will be interesting to apply structure to the tweak and find corresponding applications.
- Because of the chosen-plaintext feature of differential cryptanalysis, our backdoor is somewhat traceable, it might not be secret anymore once used. It is an open problem how to avoid this issue. Besides, other cryptanalysis techniques than just a plain differential attack (such as impossible differential attacks, boomerang attacks, integral attacks, etc.) might also be used to build backdoors and could allow us to build more efficient or more usable designs. Moreover, the idea of our framework could possibly be applied to other primitives, such as hash functions and MACs.

Bibliography

- [1] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. <https://competitions.cr.yp.to/caesar-submissions.html>.
- [2] Ange Albertini, Jean-Philippe Aumasson, Maria Eichlseder, Florian Mendel, and Martin Schlffer. Malicious Hashing: Eve’s Variant of SHA-1. In *SAC 2014: 21st Annual International Workshop on Selected Areas in Cryptography*, volume 8781 of *LNCS*, pages 1–19. Springer, Heidelberg, Germany, 2014.
- [3] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, Germany, 2015.
- [4] Riham AlTawy and Amr M. Youssef. Preimage Attacks on Reduced-Round Stribog. In *AFRICACRYPT 14: 7th International Conference on Cryptology in Africa*, volume 8469 of *LNCS*, pages 109–125. Springer, Heidelberg, Germany, 2014.
- [5] Riham AlTawy and Amr M. Youssef. Second Preimage Analysis of Whirlwind. In *Information Security and Cryptology*, volume 8957 of *LNCS*, pages 311–328. Springer, Heidelberg, Germany, 2014.
- [6] Riham AlTawy and Amr M. Youssef. Watch your Constants: Malicious Streebog. Cryptology ePrint Archive, Report 2014/879, 2014. <https://eprint.iacr.org/2014/879>.
- [7] Vesela Angelova and Yuri Borissov. Plaintext Recovery in DES-like Cryptosystems Based on S-boxes with Embedded Parity Check. *Serdica Journal of Computing*, 7(3):257p–270p, 2013.
- [8] Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for Step-Reduced SHA-2. In *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 578–597. Springer, Heidelberg, Germany, 2009.
- [9] Kazumaro Aoki and Yu Sasaki. Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In *Advances in Cryptology –*

- CRYPTO 2009*, volume 5677 of *LNCS*, pages 70–89. Springer, Heidelberg, Germany, 2009.
- [10] Kazumaro Aoki and Yu Sasaki. Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In *SAC 2008: 15th Annual International Workshop on Selected Areas in Cryptography*, volume 5381 of *LNCS*, pages 103–119. Springer, Heidelberg, Germany, 2009.
 - [11] Jean-Philippe Aumasson, Willi Meier, and Florian Mendel. Preimage Attacks on 3-Pass HAVAL and Step-Reduced MD5. In *SAC 2008: 15th Annual International Workshop on Selected Areas in Cryptography*, volume 5381 of *LNCS*, pages 120–135. Springer, Heidelberg, Germany, 2009.
 - [12] Arnaud Bannier, Nicolas Bodin, and Eric Filiol. Partition-Based Trapdoor Ciphers. Cryptology ePrint Archive, Report 2016/493, 2016. <http://eprint.iacr.org/2016/493>.
 - [13] Arnaud Bannier and Eric Filiol. Mathematical Backdoors in Symmetric Encryption Systems-Proposal for a Backdoored AES-like Block Cipher. arXiv preprint arXiv:1702.06475, 2017. <https://arxiv.org/abs/1702.06475>.
 - [14] Zhenzhen Bao, Lin Ding, Jian Guo, Haoyang Wang, and Wenying Zhang. Improved Meet-in-the-Middle Preimage Attacks against AES Hashing Modes. *IACR Transactions on Symmetric Cryptology*, 2019(4):318–347, 2020.
 - [15] Zhenzhen Bao, Jian Guo, Tetsu Iwata, and Kazuhiko Minematsu. ZOCC and ZOTR: Tweakable Blockcipher Modes for Authenticated Encryption with Full Absorption. *IACR Transactions on Symmetric Cryptology*, 2019(2):1–54, 2019.
 - [16] Achiya Bar-On, Itai Dinur, Orr Dunkelman, Virginie Lallemand, Nathan Keller, and Boaz Tsaban. Cryptanalysis of SP Networks with Partial Non-Linear Layers. In *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 315–342. Springer, Heidelberg, Germany, 2015.
 - [17] Elaine B Barker and John Michael Kelsey. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators (revised)*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, Computer Security Division, Information Technology Laboratory, 2007.
 - [18] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. <http://eprint.iacr.org/2013/404>.

- [19] 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 *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 123–153. Springer, Heidelberg, Germany, 2016.
- [20] Mihir Bellare and Tadayoshi Kohno. A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, Heidelberg, Germany, 2003.
- [21] Daniel J Bernstein, Tanja Lange, and Ruben Niederhagen. *Dual EC: A Standardized Back Door*, pages 256–281. Springer, Berlin, Heidelberg, 2016.
- [22] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, Ronny Van Keer, and Benoît Viguier. KangarooTwelve: Fast Hashing Based on Keccak-p. In *ACNS 18: 16th International Conference on Applied Cryptography and Network Security*, volume 10892 of *LNCS*, pages 400–418. Springer, Heidelberg, Germany, 2018.
- [23] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT hash workshop*, volume 2007. Citeseer, 2007.
- [24] Eli Biham. New Types of Cryptanalytic Attacks Using Related Keys. *Journal of Cryptology*, 7(4):229–246, 1994.
- [25] Eli Biham. Cryptanalysis of Patarin’s 2-Round Public Key System with S Boxes (2R). In *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 408–416. Springer, Heidelberg, Germany, 2000.
- [26] Eli Biham, Orr Dunkelman, and Nathan Keller. The Rectangle Attack - Rectangling the Serpent. In *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 340–357. Springer, Heidelberg, Germany, 2001.
- [27] Eli Biham, Orr Dunkelman, and Nathan Keller. New Results on Boomerang and Rectangle Attacks. In *Fast Software Encryption – FSE 2002*, volume 2365 of *LNCS*, pages 1–16. Springer, Heidelberg, Germany, 2002.
- [28] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-Key Boomerang and Rectangle Attacks. In *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 507–525. Springer, Heidelberg, Germany, 2005.

- [29] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *Advances in Cryptology – CRYPTO’90*, volume 537 of *LNCS*, pages 2–21. Springer, Heidelberg, Germany, 1991.
- [30] Alex Biryukov, Christophe De Cannière, and Gustaf Dellkrantz. Cryptanalysis of SAFER++. In *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *LNCS*, pages 195–211. Springer, Heidelberg, Germany, 2003.
- [31] Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key Recovery Attacks of Practical Complexity on AES-256 Variants with up to 10 Rounds. In *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 299–319. Springer, Heidelberg, Germany, 2010.
- [32] Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 1–18. Springer, Heidelberg, Germany, 2009.
- [33] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and Related-Key Attack on the Full AES-256. In *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *LNCS*, pages 231–249. Springer, Heidelberg, Germany, 2009.
- [34] Alex Biryukov and Ivica Nikolic. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to AES, Camellia, Khazad and others. In *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 322–344. Springer, Heidelberg, Germany, 2010.
- [35] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique Cryptanalysis of the Full AES. In *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 344–371. Springer, Heidelberg, Germany, 2011.
- [36] Raphael Bost. Σοφος: Forward Secure Searchable Encryption. In *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1143–1154. ACM Press, 2016.
- [37] Raphaël Bost, Brice Minaud, and Olga Ohrimenko. Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives. In *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1465–1482. ACM Press, 2017.
- [38] Christina Boura and Anne Canteaut. On the Boomerang Uniformity of Cryptographic Sboxes. *IACR Transactions on Symmetric Cryptology*, 2018(3):290–310, 2018.

- [39] Marco Calderini and Massimiliano Sala. On Differential Uniformity of Maps that May Hide an Algebraic Trapdoor. In *Algebraic Informatics*, pages 70–78. Springer International Publishing, 2015.
- [40] Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In *Advances in Cryptology – CRYPTO’98*, volume 1462 of *LNCS*, pages 56–71. Springer, Heidelberg, Germany, 1998.
- [41] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. A Security Analysis of Deoxys and its Internal Tweakable Block Ciphers. *IACR Transactions on Symmetric Cryptology*, 2017(3):73–107, 2017.
- [42] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang Connectivity Table: A New Cryptanalysis Tool. In *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 683–714. Springer, Heidelberg, Germany, 2018.
- [43] Bram Cohen and Ben Laurie. AES-hash. *NIST: Modes of Operation for Symmetric Key Block Ciphers*, 2001.
- [44] Joan Daemen. *Cipher and Hash Function Design Strategies Based on Linear and Differential Cryptanalysis*. PhD thesis, Doctoral Dissertation, March 1995, KU Leuven, 1995.
- [45] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
- [46] Joan Daemen and Vincent Rijmen. On the related-key attacks against AES. *Proceedings of the Romanian Academy, Series A*, 13:395–400, 2010.
- [47] Ivan Bjerre Damgård. A design principle for hash functions. In *Advances in Cryptology — CRYPTO’ 89 Proceedings*, volume 435 of *LNCS*, pages 416–427. Springer, Heidelberg, Germany, 1989.
- [48] Hüseyin Demirci and Ali Aydin Selçuk. A Meet-in-the-Middle Attack on 8-Round AES. In *Fast Software Encryption – FSE 2008*, volume 5086 of *LNCS*, pages 116–126. Springer, Heidelberg, Germany, 2008.
- [49] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. In *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 371–387. Springer, Heidelberg, Germany, 2013.
- [50] Whitfield Diffie and Martin E. Hellman. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *IEEE Computer*, 10(6):74–84, 1977.

- [51] Itai Dinur, Daniel Kales, Angela Promitzer, Sebastian Ramacher, and Christian Rechberger. Linear Equivalence of Block Ciphers with Partial Non-Linear Layers: Application to LowMC. In *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 343–372. Springer, Heidelberg, Germany, 2019.
- [52] Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. Optimized Interpolation Attacks on LowMC. In *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 535–560. Springer, Heidelberg, Germany, 2015.
- [53] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Higher-Order Cryptanalysis of LowMC. In *ICISC 15: 18th International Conference on Information Security and Cryptology*, volume 9558 of *LNCS*, pages 87–101. Springer, Heidelberg, Germany, 2016.
- [54] Orr Dunkelman. Efficient Construction of the Boomerang Connection Table. Cryptology ePrint Archive, Report 2018/631, 2018. <http://eprint.iacr.org/2018/631>.
- [55] Orr Dunkelman, Nathan Keller, and Adi Shamir. A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. In *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *LNCS*, pages 393–410. Springer, Heidelberg, Germany, 2010.
- [56] Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 158–176. Springer, Heidelberg, Germany, 2010.
- [57] Thomas Espitau, Pierre-Alain Fouque, and Pierre Karpman. Higher-order differential meet-in-the-middle preimage attacks on SHA-1 and BLAKE. In *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 683–701. Springer, Heidelberg, Germany, 2015.
- [58] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved Cryptanalysis of Rijndael. In *Fast Software Encryption – FSE 2000*, volume 1978 of *LNCS*, pages 213–230. Springer, Heidelberg, Germany, 2001.
- [59] Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert. Block Ciphers That Are Easier to Mask: How Far Can We Go? In *Cryptographic Hardware and Embedded Systems – CHES 2013*, volume 8086 of *LNCS*, pages 383–399. Springer, Heidelberg, Germany, 2013.
- [60] Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In *Fast Software Encryption – FSE 2010*, volume 6147 of *LNCS*, pages 365–383. Springer, Heidelberg, Germany, 2010.

- [61] Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 56–75. Springer, Heidelberg, Germany, 2010.
- [62] Jian Guo, Ivica Nikolic, Thomas Peyrin, and Lei Wang. Cryptanalysis of Zorro. Cryptology ePrint Archive, Report 2013/713, 2013. <http://eprint.iacr.org/2013/713>.
- [63] Jian Guo, Chunhua Su, and Wun-She Yap. An improved preimage attack against HAVAL-3. *Inf. Process. Lett.*, 115(2):386–393, 2015.
- [64] Deukjo Hong, Bonwook Koo, and Yu Sasaki. Improved Preimage Attack for 68-Step HAS-160. In *ICISC 09: 12th International Conference on Information Security and Cryptology*, volume 5984 of *LNCS*, pages 332–348. Springer, Heidelberg, Germany, 2010.
- [65] Sebastiaan Indesteege and Bart Preneel. Preimages for Reduced-Round Tiger. In *Research in Cryptology*, volume 4945 of *LNCS*, pages 90–99. Springer, Heidelberg, Germany, 2007.
- [66] ISO/IEC. 10118-2:2010 Information technology — Security techniques – Hash-functions – Part 2: Hash-functions using an n -bit block cipher. 3rd ed., International Organization for Standardization, Geneva, Switzerland, October, 2010.
- [67] Mitsugu Iwamoto, Thomas Peyrin, and Yu Sasaki. Limited-Birthday Distinguishers for Hash Functions - Collisions beyond the Birthday Bound Can Be Meaningful. In *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 504–523. Springer, Heidelberg, Germany, 2013.
- [68] Jérémy Jean. TikZ for Cryptographers. <https://www.iacr.org/authors/tikz/>, 2016.
- [69] Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. KIASU v1. Additional first-round candidates of CAESAR compeition, <https://competitions.cr.yp.to/caesar-submissions.html>, 2014.
- [70] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 274–288. Springer, Heidelberg, Germany, 2014.
- [71] Jérémy Jean, Ivica Nikolic, Thomas Peyrin, and Yannick Seurin. Deoxys v1. 41. *Submitted to CAESAR*, 2016.

- [72] Daniel Kales, Léo Perrin, Angela Promitzer, Sebastian Ramacher, and Christian Rechberger. Improvements to the Linear Operations of LowMC: A Faster Picnic. Cryptology ePrint Archive, Report 2017/1148, 2017. <https://eprint.iacr.org/2017/1148>.
- [73] John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In *Fast Software Encryption – FSE 2000*, volume 1978 of *LNCS*, pages 75–93. Springer, Heidelberg, Germany, 2001.
- [74] Jongsung Kim, Seokhie Hong, and Bart Preneel. Related-Key Rectangle Attacks on Reduced AES-192 and AES-256. In *Fast Software Encryption – FSE 2007*, volume 4593 of *LNCS*, pages 225–241. Springer, Heidelberg, Germany, 2007.
- [75] Simon Knellwolf and Dmitry Khovratovich. New Preimage Attacks against Reduced SHA-1. In *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *LNCS*, pages 367–383. Springer, Heidelberg, Germany, 2012.
- [76] Stefan Kölbl, Martin M. Lauridsen, Florian Mendel, and Christian Rechberger. Haraka v2 - Efficient Short-Input Hashing for Post-Quantum Applications. *IACR Transactions on Symmetric Cryptology*, 2016(2):1–29, 2016.
- [77] V.F. Kolchin. *Random Graphs*. Cambridge University Press, 1999.
- [78] Will Landecker, Thomas Shrimpton, and R. Seth Terashima. Tweakable Blockciphers with Beyond Birthday-Bound Security. In *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *LNCS*, pages 14–30. Springer, Heidelberg, Germany, August 19–23, 2012.
- [79] Gaëtan Leurent. MD4 is Not One-Way. In *Fast Software Encryption – FSE 2008*, volume 5086 of *LNCS*, pages 412–428. Springer, Heidelberg, Germany, 2008.
- [80] Ji Li, Takanori Isobe, and Kyoji Shibutani. Converting Meet-In-The-Middle Preimage Attack into Pseudo Collision Attack: Application to SHA-2. In *Fast Software Encryption – FSE 2012*, volume 7549 of *LNCS*, pages 264–286. Springer, Heidelberg, Germany, 2012.
- [81] Leibo Li, Keting Jia, and Xiaoyun Wang. Improved Single-Key Attacks on 9-Round AES-192/256. In *Fast Software Encryption – FSE 2014*, volume 8540 of *LNCS*, pages 127–146. Springer, Heidelberg, Germany, 2015.
- [82] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. In *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, Heidelberg, Germany, 2002.

- [83] Jiqiang Lu, Orr Dunkelman, Nathan Keller, and Jongsung Kim. New Impossible Differential Attacks on AES. In *Progress in Cryptology - INDOCRYPT 2008: 9th International Conference in Cryptology in India*, volume 5365 of *LNCS*, pages 279–293. Springer, Heidelberg, Germany, 2008.
- [84] Alfred J Menezes, Jonathan Katz, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.
- [85] Ralph C Merkle. One way hash functions and DES. In *Advances in Cryptology — CRYPTO’ 89 Proceedings*, volume 435 of *LNCS*, pages 428–446. Springer, Heidelberg, Germany, 1989.
- [86] Kazuhiko Minematsu. Beyond-Birthday-Bound Security Based on Tweakable Block Cipher. In *Fast Software Encryption – FSE 2009*, volume 5665 of *LNCS*, pages 308–326. Springer, Heidelberg, Germany, February 22–25, 2009.
- [87] Pawel Morawiecki. Malicious Keccak. Cryptology ePrint Archive, Report 2015/1085, 2015. <https://eprint.iacr.org/2015/1085>.
- [88] Sean Murphy. The return of the cryptographic boomerang. *IEEE Transactions on Information Theory*, 57(4):2517–2521, 2011.
- [89] National Institute of Standards and Technology. FIPS 180-1: Secure Hash Standard. April 1995.
- [90] National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard. August 2002.
- [91] National Institute of Standards and Technology. FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. August 2015.
- [92] Jacques Patarin and Louis Goubin. Asymmetric Cryptography with S-Boxes. In *ICICS 97: 1st International Conference on Information and Communication Security*, volume 1334 of *LNCS*, pages 369–380. Springer, Heidelberg, Germany, 1997.
- [93] Jacques Patarin and Louis Goubin. Trapdoor One-Way Permutations and Multivariate Polynomials. In *ICICS 97: 1st International Conference on Information and Communication Security*, volume 1334 of *LNCS*, pages 356–368. Springer, Heidelberg, Germany, 1997.
- [94] Kenneth G. Paterson. Imprimitive Permutation Groups and Trapdoors in Iterated Block Ciphers. In *Fast Software Encryption – FSE’99*, volume 1636 of *LNCS*, pages 201–214. Springer, Heidelberg, Germany, 1999.

- [95] Thomas Peyrin and Haoyang Wang. The MALICIOUS Framework: Embedding Backdoors into Tweakable Block Ciphers. In *Advances in Cryptology – CRYPTO 2020*, volume 12172 of *LNCS*, pages 249–278. Springer, Heidelberg, Germany, 2020.
- [96] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In *Advances in Cryptology – CRYPTO’93*, volume 773 of *LNCS*, pages 368–378. Springer, Heidelberg, Germany, 1994.
- [97] NIST FIPS PUB. 46-3. Data Encryption Standard. *Federal Information Processing Standards, National Bureau of Standards, US Department of Commerce*, 1977.
- [98] Shahram Rasoolzadeh, Zahra Ahmadian, Mahmood Salmasizadeh, and Mohammad Reza Aref. Total Break of Zorro using Linear and Differential Attacks. Cryptology ePrint Archive, Report 2014/220, 2014. <http://eprint.iacr.org/2014/220>.
- [99] Christian Rechberger, Hadi Soleimany, and Tyge Tiessen. Cryptanalysis of Low-Data Instances of Full LowMCv2. *IACR Transactions on Symmetric Cryptology*, 2018(3):163–181, 2018.
- [100] Vincent Rijmen and Bart Preneel. A Family of Trapdoor Ciphers. In *Fast Software Encryption – FSE’97*, volume 1267 of *LNCS*, pages 139–148. Springer, Heidelberg, Germany, 1997.
- [101] Ronald Rivest and S Duse. The MD5 message-digest algorithm, 1992.
- [102] Ronald L Rivest. The MD4 message digest algorithm. In *Advances in Cryptology-CRYPTO’ 90*, volume 537 of *LNCS*, pages 303–311. Springer, Heidelberg, Germany, 1990.
- [103] Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, Heidelberg, Germany, December 5–9, 2004.
- [104] Yu Sasaki. Meet-in-the-Middle Preimage Attacks on AES Hashing Modes and an Application to Whirlpool. In *Fast Software Encryption – FSE 2011*, volume 6733 of *LNCS*, pages 378–396. Springer, Heidelberg, Germany, 2011.
- [105] Yu Sasaki and Kazumaro Aoki. Preimage Attacks on 3, 4, and 5-Pass HAVAL. In *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 253–271. Springer, Heidelberg, Germany, 2008.
- [106] Yu Sasaki and Kazumaro Aoki. Finding Preimages in Full MD5 Faster than Exhaustive Search. In *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 134–152. Springer, Heidelberg, Germany, 2009.

- [107] Dan Shumow and Niels Ferguson. On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng. In *Proc. Crypto*, volume 7, 2007.
- [108] David Wagner. The Boomerang Attack. In *Fast Software Encryption – FSE’99*, volume 1636 of *LNCS*, pages 156–170. Springer, Heidelberg, Germany, 1999.
- [109] Haoyang Wang and Thomas Peyrin. Boomerang Switch in Multiple Rounds. Application to AES Variants and Deoxys. *IACR Transactions on Symmetric Cryptology*, 2019(1):142–169, 2019.
- [110] Lei Wang and Yu Sasaki. Finding Preimages of Tiger Up to 23 Steps. In *Fast Software Encryption – FSE 2010*, volume 6147 of *LNCS*, pages 116–133. Springer, Heidelberg, Germany, 2010.
- [111] Lei Wang, Yu Sasaki, Wataru Komatsubara, Kazuo Ohta, and Kazuo Sakiyama. (Second) Preimage Attacks on Step-Reduced RIPEMD/RIPEMD-128 with a New Local-Collision Approach. In *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *LNCS*, pages 197–212. Springer, Heidelberg, Germany, 2011.
- [112] Yanfeng Wang, Wenling Wu, Zhiyuan Guo, and Xiaoli Yu. Differential Cryptanalysis and Linear Distinguisher of Full-Round Zorro. Cryptology ePrint Archive, Report 2013/775, 2013. <http://eprint.iacr.org/2013/775>.
- [113] Hongjun Wu, Feng Bao, Robert H. Deng, and Qin-Zhong Ye. Cryptanalysis of Rijmen-Preneel Trapdoor Ciphers. In *Advances in Cryptology – ASIACRYPT’98*, volume 1514 of *LNCS*, pages 126–132. Springer, Heidelberg, Germany, 1998.
- [114] Shuang Wu, Dengguo Feng, Wenling Wu, Jian Guo, Le Dong, and Jian Zou. (Pseudo) Preimage Attack on Round-Reduced Grøstl Hash Function and Others. In *Fast Software Encryption – FSE 2012*, volume 7549 of *LNCS*, pages 127–145. Springer, Heidelberg, Germany, 2012.
- [115] Dingfeng Ye, Kwok-Yan Lam, and Zong-Duo Dai. Cryptanalysis of "2 R" Schemes. In *Advances in Cryptology – CRYPTO’99*, volume 1666 of *LNCS*, pages 315–325. Springer, Heidelberg, Germany, 1999.
- [116] Adam Young and Moti Yung. The Dark Side of “Black-Box” Cryptography, or: Should We Trust Capstone? In *Advances in Cryptology – CRYPTO’96*, volume 1109 of *LNCS*, pages 89–103. Springer, Heidelberg, Germany, 1996.
- [117] Adam Young and Moti Yung. Monkey: Black-Box Symmetric Ciphers Designed for MONopolizing KEYS. In *Fast Software Encryption – FSE’98*, volume 1372 of *LNCS*, pages 122–133. Springer, Heidelberg, Germany, 1998.

- [118] Adam Young and Moti Yung. A Subliminal Channel in Secret Block Ciphers. In *SAC 2004: 11th Annual International Workshop on Selected Areas in Cryptography*, volume 3357 of *LNCS*, pages 198–211. Springer, Heidelberg, Germany, 2004.
- [119] Adam Young and Moti Yung. *Malicious Cryptography: Exposing Cryptovirology*. John Wiley & Sons, 2004.
- [120] Adam L. Young and Moti Yung. Backdoor Attacks on Black-Box Ciphers Exploiting Low-Entropy Plaintexts. In *ACISP 03: 8th Australasian Conference on Information Security and Privacy*, volume 2727 of *LNCS*, pages 297–311. Springer, Heidelberg, Germany, 2003.