Cryptanalysis of Symmetric Key Schemes using Classical and Quantum Techniques

Submitted to Indian Statistical Institute
in partial fulfillment of the thesis requirements for the Degree of
Doctor of Philosophy in Computer Science

Author: **Mostafizar Rahman**
Senior Research Fellow

Supervisor: **Goutam Paul**
Associate Professor

Cryptology and Security Research Unit
R. C. Bose Centre for Cryptology and Security
Indian Statistical Institute
Kolkata - 700108, India

January 2022

Dedicated to

My Parents and my Brother

DECLARATION OF AUTHORSHIP

I, Mostafizar Rahman, a student of Cryptology and Security Research Unit, of the Ph.D. program of Indian Statistical Institute, Kolkata, hereby declare that the research work presented in this thesis titled "Cryptanalysis of Symmetric Key Schemes using Classical and Quantum Techniques" is based on my works. To the best of my knowledge, the materials presented in this thesis have not previously been published or written by any other person, nor it has been submitted as a whole or as a part of any degree/diploma or any other academic award anywhere before.

Mostafizar Rahman

Cryptology and Security Research Unit

Indian Statistical Institute, Kolkata

203, Barrackpore Trunk Road

Kolkata 700108, INDIA.

LIST OF PUBLICATIONS/MANUSCRIPTS

1. **Mostafizar Rahman**, Dhiman Saha and Goutam Paul, "Cryptanalysis of FlexAEAD", Progress in Cryptology - AFRICACRYPT 2020. Lecture Notes in Computer Science, vol 12174. Springer, Cham,
   DOI: https://doi.org/10.1007/978-3-030-51938-4_8.

2. Dhiman Saha, **Mostafizar Rahman** and Goutam Paul, "New Yoyo Tricks with AES-based Permutations", IACR Transactions on Symmetric Cryptology, 2018(4), 102-127,
   DOI: https://doi.org/10.13154/tosc.v2018.i4.102-127.

3. **Mostafizar Rahman**, Dhiman Saha and Goutam Paul, "Boomeyong: Embedding Yoyo within Boomerang and its Application to Key Recovery Attacks on AES and Pholkos", IACR Transactions on Symmetric Cryptology, 2021(3), 137–169,
   DOI: https://doi.org/10.46586/tosc.v2021.i3.137-169.

4. **Mostafizar Rahman** and Goutam Paul, "Quantum Attacks on HCTR and Its Variants", IEEE Transactions on Quantum Engineering, vol. 1, pp. 1-8, 2020,
   DOI: https://doi.org/10.1109/TQE.2020.3041426.

5. **Mostafizar Rahman** and Goutam Paul, "Grover on Katan: Quantum Resource Estimation", Accepted in IEEE Transactions on Quantum Engineering on 21 December, 2021
   (DOI not yet available)

6. **Mostafizar Rahman** and Goutam Paul, "Grover on Present: Quantum Resource Estimation", communicated to: Journal of Cryptographic Engineering, submitted on 12^{th} November, 2021.

ACKNOWLEDGEMENT

I wish to express my genuine appreciation and warm gratitude to all the well-wishers without whom this thesis would not have been possible. First and foremost, it is my great pleasure to acknowledge my Ph.D. supervisor Dr. Goutam Paul, Associate Professor of Cryptology and Security Research Unit, Indian Statistical Institute, Kolkata, for introducing me to the exciting and fascinating research field of cryptography. His valuable comments, endless support and encouragement, motivating discussions and ideas, positive criticisms help me a lot to continue my research work. I shall be forever obliged to him for his support and encouragement.

I want to express my special gratitude to Dr.Dhiman Saha, Department of Electrical Engineering and Computer Science, IIT Bhilai, for guiding me in every step and motivating me with his insightful comments and suggestions. The hour-long technical and non-technical discussions with him have a profound impact on my life.

I would like to express my deepest appreciation to all of my colleagues, namely, Samir Kundu, Amit Jana, Nayana Das, Prabal Banerjee, Avishek Majumder, Laltu Sarder, Diptendu Chatterjee, Pritam Chattopdhayay, Soumya Das, Ram Govind Singh for helping me with their passionate discussions without any entitlement during this journey. I am lucky enough to be surrounded by all of them during this tedious and tough journey. The moments I have spent with them will be cherished by me forever. I am also thankful to Dr. Pabitra Pal and other members of de.ci.phe.red lab, IIT Bhilai, for their support during my visit to IIT, Bhilai.

I greatly appreciate my parents for their continuous support and the sacrifices that they have made on my behalf. I express my deep gratitude to my beloved brother Masud for his constant encouragement and unconditional support. Special thanks to my friend Bidar for being there to help me out during the rough patches of my life. Thank you, Tamanna, for all the love and support you give me since we met.

ABSTRACT

Symmetric key cryptography refers to the encryption methods in which the same key is used by both the sender and the receiver. Cryptanalysis is a process of finding vulnerabilities in cryptographic algorithms in order to distinguish the algorithm, or to retrieve the plaintext from ciphertext without the knowledge of the secret key, or sometimes to recover the secret key also. In this work, in addition to using existing cryptanalysis techniques to analyze some recent ciphers, we also develop novel cryptanalysis techniques.

The cryptanalysis techniques that are involved here are based on both classical and quantum computing models. In classical cryptanalysis, first of all, we break the authenticated encryption scheme FlexAEAD by mounting forgery using the devised iterated truncated differentials. Further, we mount key recover attacks on the underlying keyed permutation of FlexAEAD. We develop new techniques of cryptanalysis by augmenting yoyo game with classical, impossible and improbable differentials and its impact is shown by applying it on public permutation AESQ and `AES` in the known-key setting. Another new technique is developed by embedding a boomerang attack within a yoyo game, which is shown to be effective by breaking the claimed security of AES-like block ciphers. In quantum cryptanalysis, we analyze several symmetric key schemes by using Simon's algorithm or by combining Simon's with Grover's algorithms. We also provide cost estimation for mounting Grover's attack on lightweight block ciphers KATAN and Present. To strengthen the validity of our results, all practical attacks are experimentally verified.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACROYNMS AND ABBREVIATIONS

Expansion	Acronyms/ Abbreviations
Authenticated Encryption with Associated Data	AEAD
Advanced Encryption Standard	AES
AddRoundKey	AK
AddRoundTweakey	ATK
Cipher Block Chaining	CBC
Difference Distribution Table	DDT
Decryption Queries	Decs
Electronic Codebook	ECB
Elliptic-curve Diffie–Hellman	ECDH
Elliptic Curve Digital Signature Algorithm	ECDSA
Encryption Queries	Encs
Initialization Vector	IV
Hash-Counter-Hash	HCH
Hash Counter	HCTR
Key Scheduling Algorithm	KSA
Lightweight Cryptography	LWC
Memory Access	MA
MixColumns	MC
MegaMixColumns	MMC
National Institute of Standards and Technology	NIST
Post Quantum Cryptography	PQC
Rivest, Shamir, Adleman	RSA
Substitution Box	S-Box
SubBytes	SB
Secure Hash Algorithm	SHA
Substitution-Permutation Network	SPN
Signal-to-Noise Ratio	S/N
ShiftRows	SR
Exclusive-OR	XOR
Zero Difference Pattern	ZDP
That is	i.e.

LIST OF SYMBOLS

Throughout the thesis, we use some notations and we describe those common notations here.

- $\Pr(A)$: Probability of occurrence of an event A.
- $\Pr(A|B)$: Probability of occurrence of an event A given that the event B has already occurred.
- $x \leftarrow y$: x gets the value of y
- $\alpha \rightarrow \beta$: The transition from α to β
- $\alpha \nrightarrow \beta$: α does not transit to β
- $wt(x)$: Weight of a vector x
- x_i: i^{th} component of vector x
- $\texttt{AESQ}_{i \rightarrow j}$: `AESQ` permutation from round i to round j
- $\subset_\phi$: Non-null proper subset
- $|+\rangle : \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.
- $U^\dagger$: Conjugate transpose of U.
- $H = \frac{1}{\sqrt{2}}(\sigma_x + \sigma_z)$: The Hadamard operator.

1 INTRODUCTION

Contents

Cryptography is the science of protecting information from potential adversaries. As of documented sources, the history of cryptography starts with the introduction of *scytale*, a Spartan cryptographic device based on transposition technique which was used by the military in the fifth century BC. Caesar cipher is one of the famous ancient cryptographic schemes based on substitution techniques. Although, these techniques were quite secure at that time, with the advent of modern systems, these crypto-systems become vulnerable. In earlier times, cryptography is practiced primarily as a form of an art. The introduction of the communication theory of secrecy systems by Shannon [185] helped in transiting the subject from an art to a science. The main goal of cryptography is to provide confidentiality, integrity, authenticity, anonymity, etc.

In terms of security, cryptography can be divided into two types: information-theoretic security and computational security. Information-theoretic security, also

called perfect security or unconditional security, defines the security of a crypto-system based on the theoretical impossibility of breaking the system, even if unlimited computing power is available to the adversary; whereas computational security defines the security of crypto-systems based on the hardness assumption of breaking the system using limited computing power. Although, information-theoretic secure systems are more secure in comparison to the computational secure systems; however, such systems are quite impractical to implement. One such information-theoretic secure cryptosystem is *one-time pad* which requires a key of length same as the plaintext to be encrypted and the key has to be different for each encryption. However, with respect to information theoretic security, two relaxations are considered for computational security. They are as follows-

1. **Time Boundedness.** It is assumed that the adversary will run only for a feasible amount of time. If unlimited time is available to the adversary, then the scheme can be broken.

2. **Low Success Probability.** The adversary can break the scheme with a very low probability. This probability should have a negligible effect on the security of the scheme.

To precisely define these relaxations, two approaches are considered: the *concrete approach* and the *asymptotic approach*.

Concrete Approach. In a concrete approach, the security of a cryptographic scheme is quantified. The computational effort of the adversary is specified and the maximum success probability of the adversary explicitly bounds the security of the scheme.

Asymptotic Approach. In the asymptotic approach, the security of cryptographic schemes are parametrized by a security parameter. Generally, the length of the key is considered as the security parameter and it is assumed that it is known to the adversaries.

1.1 Motivation

Due to the impracticability of information-theoretic security, in the modern age crypto-systems are designed mainly based on computational security. In terms of application, cryptography covers a broad area which includes banking, communication, digital authentication, healthcare, etc. Computational security provides a way to measure the strength of security and requires rigorous proof. However, such security proofs do not guarantee "absolute security" as bypassing the underlying security assumptions makes the system vulnerable. In addition, there are many security schemes which relies on the rigorous third-party analysis rather than security proofs [21], like `AES` [88, 6]. With the advent of quantum computers and the recent progress towards the design of new quantum algorithms, the computational secure systems are also at the edge of vulnerability with respect to newly evolved threat models.

1.1.1 Symmetric Key Algorithms

Broadly, there are two types of cryptographic algorithms, namely, symmetric key algorithms and asymmetric key algorithms. Symmetric key, also known as secret key or private key cryptography, uses the same key for both encryption and decryption. In asymmetric key cryptography two different keys are used- a public key is used for encryption and a private key is used for decryption and thus it is also known as public key cryptography. Symmetric key schemes are faster and more efficient to use whereas asymmetric key schemes are more secure to use. However, in *end-to-end* encryption both symmetric and asymmetric key algorithms are used.

The symmetric key algorithms can be categorized into several variants: stream cipher, block ciphers, message authentication code (MAC) and authenticated encryption (AE). Stream ciphers and block ciphers are used for achieving data confidentiality. In general, the block ciphers use the secret key to encrypt a block of plaintext to ciphertext whereas the stream ciphers encrypt the plaintext by XOR-ing the plaintext with a keystream that is generated using the secret key. MAC are used to provide data integrity by generating a tag on the sender's end and verifying it at the receiver's

end. To encrypt messages arbitrary length using block ciphers, encryption modes are used. AE algorithms serve data confidentiality and data integrity by encrypting arbitrary length messages as well as generating tags. In recent times, owing to the application of cryptography in resource constraint devices, the notion of lightweight cryptography has emerged.

1.1.2 Cryptanalysis

Cryptanalysis is the study of finding the weaknesses in ciphers. According to Kerckhoff's principle (Kerckhoffs's desideratum, assumption, axiom), a cipher should be secure even if the cipher's construction and structure, except the secret key, is public. Owing to this principle, a cryptographic algorithm should be designed in a way such that no attacker can break the scheme without the knowledge of the key even though the complete structure of the scheme is known to the attacker.

This model is known as the open cryptographic model and based on this model, several cryptographic algorithms are standardized, like- Authenticated Encryption Standard (`AES`) [169], Secure Hash Algorithm 3 (SHA3) [5], CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness [1], New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project [3], eStream [2], Lightweight Cryptography (LWC) competition [4], etc. Several algorithms that are submitted in such standardization competitions are made public and receive rigorous analysis for several years. After public scrutiny, one algorithm or a suite of algorithms are standardized based on the requirement of the competition. Thus designing a cryptographic schemes does not complete the requirements of obtaining a secure system, rigorous analysis of such systems are necessary to build confidence in such schemes to use them for practical purposes.

1.1.3 Speeding up the Attacks using Quantum Algorithms

Exploting the quantum-mechanical phenomena to solve computationally hard problems is the focus of researchers in the recent time which has lead to the development of

Grover's search algorithm [120], Simon's algorithm [189], Shor's algorithm [188, 187], etc. The introduction of such algorithms threatens the security of cryptographic schemes. The most notable of these is the Shor's algorithm, whose ability to solve the factorization problem and compute discrete logarithms in polynomial-time has unveiled the vulnerability of several public key cryptographic schemes, like, RSA, ECDSA and ECDH. Private key schemes are vulnerable to generic key recovery attacks due to implications of Grover's search algorithm on block ciphers [211]. Recently, the vulnerabilities posed by Simon's algorithm on some specific symmetric key schemes have been studied [145, 146, 134, 66, 175, 112]. Thus, the security of cryptographic algorithms are on the verge of being compromised due to the inevitability of the introduction of quantum computers. Owing to such conditions, the National Institute of Standards and Technology (NIST) has called for proposals for post-quantum cryptography standardization with goals for standardizing new cryptographic algorithms that are secure against classical as well as quantum attacks [171]. Thus studying the security of existing cryptographic algorithms in quantum computing models provides insights regarding the validity of such algorithms in the post-quantum world.

1.2 Outline and Contribution

The thesis discusses the cryptanalysis of symmetric key schemes using classical and quantum techniques. New cryptanalytic techniques are designed to mount attacks on several new and old ciphers. First, a brief introduction is provided and then in Chapter 2, necessary backgrounds required for the rest of the thesis are discussed. The main work of the thesis is divided as follows.

- In Chapter 2, we provide a brief introduction to block cipher and mode of operation. The block cipher and permutations used in the thesis are discussed here. We briefly describe about several classical cryptanalysis techniques and introduce some quantum algorithms that are considered in the later part of the thesis.

- In Chapter 3, we report an iterated truncated differential for all the variants of internal keyed permutation of FlexAEAD (PF_k) using the property of `AES` Difference Distribution Table (DDT) where the output difference of a byte is *confined to either upper or lower nibble*. The probability of the truncated differential for one round is 2^{-7}. Its iterative nature makes it possible to penetrate more number of rounds for all Flex-x. These differentials are further exploited to devise key-recovery attacks on all the variants.

  Further, we have used the iterated truncated differentials to mount forgery attacks on FlexAEAD similar to the ones reported by Eichlseder *et al*. [103, 104]. Finally, to measure the effectiveness of all distinguishers reported in this work, their theoretical success probabilities are estimated by following the approach given in [174]. The success probabilities are estimated to be high and some of them with practical complexities are experimentally verified.

  All the attacks presented in this chapter exploit the vulnerability that merely divides the bytes into nibbles while using `AES` s-box is susceptible to differential attacks as diffusion may be slow in some scenarios. Although, FlexAEAD is out of NIST lightweight cryptography competition, this particular vulnerability has a far-reaching impact on designing ciphers using `AES` s-box. Hence, it forms the basis of continued motivation for this work.

- In Chapter 4, we explore the application of the yoyo property on PF_k which has been introduced by Rønjom *et al*. [176] on generic 2-round Substitution Permutation Networks and further extended on `AES`-based permutations and block ciphers [180, 27]. We have been able to devise deterministic yoyo distinguishers for 4, 6 and 8 rounds of Flex-64, Flex-128 and Flex-256 respectively which are further extended by one more round to mount key recovery attacks.

- In Chapter 5, we explore the yoyo idea in distinguishing public permutations for the first time. We introduce the notion of nested zero difference pattern, which extends the yoyo idea and helps compose it using improbable and impos-

sible differential strategies to penetrate a higher number of rounds. We devise a novel inside-out application of yoyo which enables us to start the yoyo game from an internal round. We devise a novel inside-out application of yoyo which enables us to start the yoyo game from an internal round. As an application, we investigate the AES-based public permutation AESQ used inside the authenticated cipher PAEQ. We achieve the first deterministic distinguisher of AESQ up to 8 rounds and the first 9-round distinguisher of AESQ that start from the first round with a practical complexity of around 2^{26}. We manage to augment yoyo with improbable and impossible differentials leading to distinguishers on $9, 10, 12$ rounds with complexities of about $2^2, 2^{28}, 2^{126}$ respectively. Further, with impossible differentials and a bi-directional yoyo strategy, we obtain a 16-round impossible differential distinguisher with a complexity of 2^{126}. Our results outperform all previous records on AESQ by a substantial margin. As another application, we apply the proposed strategies on AES in the *known-key* setting leading to one of the best 8-round known-key distinguisher with a complexity of 2^{30}. Finally, this work amplifies the scope of the yoyo technique as a generic cryptanalysis tool.

- Chapter 6 investigates a generic way of combining two very effective and well-studied cryptanalytic tools, proposed almost 18 years apart, namely the boomerang attack introduced by Wagner in FSE 1999 and the yoyo attack Ronjom *et al.* in Asiacrypt 2017. In doing so, the *s-box switch* and *ladder switch* techniques are leveraged to embed a yoyo trail inside a boomerang trail. As an immediate application, a 6-round key recovery attack on AES-128 is mounted with a time complexity of 2^{78}. A 10-round key recovery attack on recently introduced AES-based tweakable block cipher Pholkos is also furnished to demonstrate the applicability of the new technique on AES-like constructions. The results on AES are experimentally verified by applying and implementing them on a small-scale variant of AES. We provide arguments that relate the proposed strategy with the retracing boomerang attack devised in Eurocrypt 2020. To the best

of our knowledge, this is the first attempt to merge the yoyo and boomerang techniques to analyze SPN ciphers and warrants further attention as it has the potential of becoming a vital cryptanalysis tool.

- In Chapter 7, we use similar approach to the one proposed by Bonnetain *et al*. in Asiacrypt 2019 to mount new attacks on HCTR and HCH construction. In addition, we mount attacks on HCTR, Tweakable-HCTR and HCH using the superposition queries to the encryption oracle using strategies proposed by Leander and May in Asiacrypt 2017 and Kaplan *et al*. in Crypto 2016.

- Chapter 8 presents the cost analysis of mounting Grover's key search attack on the family of KATAN block cipher. Several designs of the reversible quantum circuit of KATAN are proposed. Due to NIST's proposal for Post Quantum Cryptography standardization, the circuits are designed to minimize the overall depth. We observe that the reversible quantum circuits designed using `AND` gates and T-depth one toffoli gate give more shallow circuits. Grover oracle for KATAN is designed based on those designs which are used further to mount Grover's key search attack on KATAN. The designs are implemented using the software framework ProjectQ which provides a resource estimation tool to perform an appropriate cost analysis in an automated way. While estimating the resources, NIST's depth restrictions are also respected. We also provide a similar kind of analysis for the Present block cipher.

- In Chapter 9, all the results in this thesis are summarized and we furnished concluding remarks.

2 BACKGROUND

Contents

In general, symmetric key schemes includes block ciphers, stream ciphers, mode of encryptions, message authentication codes, authenticated encryptions and hash functions. However, as this thesis is focused on the cryptanalysis of block ciphers and mode of encryptions, these two schemes are briefly discussed in this chapter. The constructions of some block ciphers and permutations are also provided. The details regarding the adversarial models, attack costs and attack goals are also discussed.

2.1 Block Cipher Primitives

A block cipher is a family of functions and inverse functions that provides confidentiality by mapping fixed-length bit strings (input block) to the same length bit strings (output block). This family of functions are parametrized by a secret key and for a fixed key it acts as a permutation. More commonly, the input and output blocks are called plaintexts and ciphertexts respectively. Ideally, for a block cipher, the relation between the plaintext and the ciphertext should be completely random.

Definition 1. *Let, n be the block length and k be the key length. Then the keyed permutation $E_k : \{0,1\}^n \times \{0,1\}^k \mapsto \{0,1\}^n$ is called as a block cipher. E_k^{-1} is defined as the inverse of E_k.*

A block cipher can also be considered as a public permutation by making the secret key public. Now, a brief discussion about some of the block ciphers and permutations analyzed in the thesis are provided.

2.1.1 `AES`: The Advanced Encryption Standard

`AES`, designed by Joan Daemen and Vincent Rijmen, is an iterated block cipher with 128-bit data blocks [8, 88]. Depending on the key length, it has three variants- i) `AES`-128- it uses a 128-bit key, ii) `AES`-192- it uses a 192-bit key and iii) `AES`-256- it uses a 256-bit key. The number of rounds in `AES`-128, `AES`-192 and `AES`-256 are 10, 12 and 14 respectively.

128-bit plaintext in `AES` is represented by a 4×4 byte matrix called state. The rows and columns of the state are both numbered from 0 to 3. In each round, four transformations are applied to an `AES` state. They are-

- SubBytes (SB)- A non-linear substitution operation applied to each byte of `AES` state in parallel.

- ShiftRows (SR)- It cyclically shifts left different rows of the state by different offsets. In general, for $0 \leq i \leq 3$, i-th row is cyclically shifted left by i bytes.

- MixColumns (MC)- It is column-mix operation. For applying this operation, a 4×4 constant maximum distance separable (MDS) matrix is used. Note that, in the context of differential cryptanalysis, in the input and output of mixcolumns, out of 8 bytes at least 5 bytes should be active. So, if there are 4 active bytes in the input of MC, then there must be at least one active byte in the output. In the rest of the thesis, a term *4-to-1* property is used, which denotes the transition from 4 active bytes to 1 active byte via MC. *4-to-1* property occurs with probability $4 \times 2^{-24} = 2^{-22}$.

- AddRoundKey (AK)- This operation is the XOR-ing of subkey with the `AES` state. The subkeys for each round are generated by key scheduling algorithm.

All the operations discussed above are invertible. In the last round, MC is omitted and before the start of the first round, AK is applied to the state. In this thesis, a special construction named Super-Sbox [89] is used for applying the attacks.

Note that, the analysis of `AES` permutation is also a part of this thesis where it is assumed that the key is known to the attacker.

2.1.2 Internal *keyed* Permutation (PF_k) of FlexAEAD

The design strategy of PF_k follows the Feistel construction. Let m be the number of bytes in a Flex-x state ($m = x/8$). The state of Flex-x is denoted by B and is divided into two equal halves: the bytes in the left half being numbered from $B[0]$ to $B[\frac{m}{2}-1]$, and the ones on the right half from $B[\frac{m}{2}]$ to $B[m-1]$. Each byte is divided into two parts representing the two nibbles, with the upper half (upper nibble) being the most significant one. The other nibble is called the lower nibble. After the `BlockShuffle` operation, m nibbles from $B[0]$ to $B[\frac{m}{2}-1]$ constitute the upper nibbles of each bytes whereas the nibbles from $B[\frac{m}{2}]$ to $B[m-1]$ constitute the lower ones. The bytes at position $B[i]$ and $B[i+\frac{m}{2}]$ are referred to as a "pair of symmetric bytes". Application of `BlockShuffle` operation on state s in r-th round is denoted by $BS^r(s)$. Fig. 2-1 shows the byte representation in Flex-128 state.

Fig. 2-2 shows the round function of Flex-128. Each round of Flex-x starts with

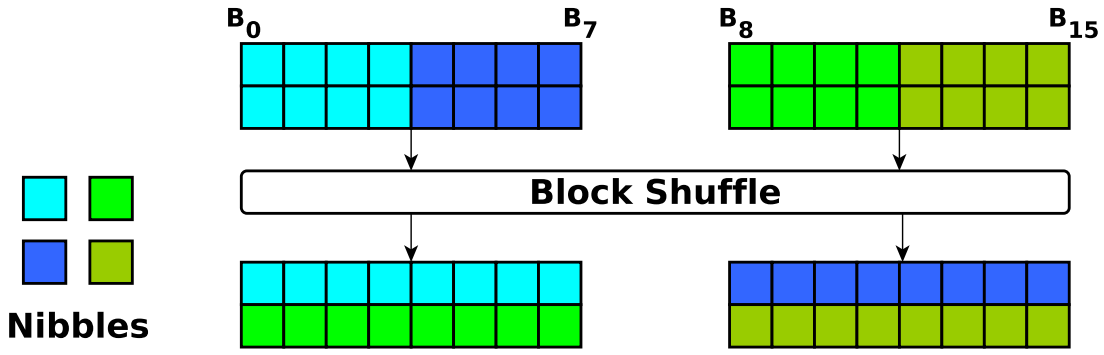


Figure 2-1: Byte Representation of Flex-128 Block Cipher

the `BlockShuffle` operation. Then the state is bifurcated and the right half goes through subbytes operation. `AES` s-box is used for byte substitution. The left half is modified by XOR-ing it with the right half and applying the subbytes operation. The modified values of the left half are XOR-ed with the right half values and subbytes is applied to get new values of the right half. Then the left and right half are combined to form the new state and the next round follows. In Flex-x there are no round keys; there are only two subkeys K_α, K_β which are used at the beginning and the end of round functions, respectively. The total number of rounds for Flex-64, Flex-128 and Flex-256 are 5, 6 and 7 respectively [96]. In authenticated encryption modes, three PF_k are used sequentially for encrypting a block of plaintext, which makes the effective number of rounds 15, 18 and 21 in FlexAEAD-64, FlexAEAD-128 and FlexAEAD-256, respectively.

Key Generation. Key generation in Flex-x uses the PF_k where the master key K is divided into two parts and used as two subkeys. The state is initialized with $0^{|K|/2}$ and three times PF_k is applied to generate part of the subkey to be used for encryption of the plaintext. This process is repeated several times till the required number of subkeys is obtained. Apart from the first round, each time, the state is initialized with the output of the previous round. The key generation algorithm makes it difficult to recover the master key from a known subkey.

2.1.3 AESQ Permutation

PAEQ is an authenticated encryption scheme. At its core, PAEQ uses the 512-bit `AESQ` permutation. This can be viewed as four 128-bit registers with each register running two rounds of `AES` where XOR-ing the subkey operation is replaced with XOR-ing a round constant. In `AESQ`, a state is of 64 bytes. There are four groups of 16 bytes each. We called each of them a register and named them A,B,C and D from

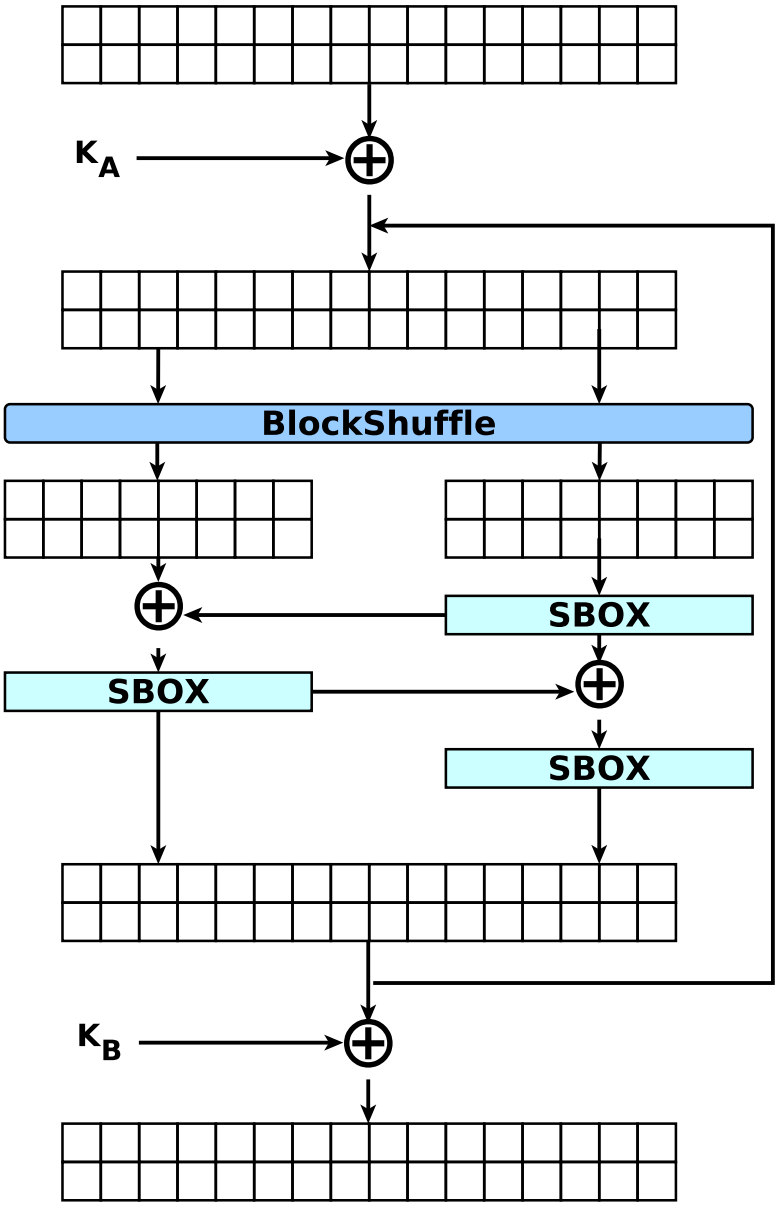


Figure 2-2: Round Function of Flex-128 Block Cipher

left to right. Each column of the registers is 32-bit words and is numbered from 0 to 3. The first and last column of register A is A[0] and A[3], respectively. Two rounds `AES` is run for each of the registers. Then a shuffling is done among all the registers. Shuffle mapping is shown in Table 2.1. In original `AESQ`, this operation is repeated 10 times. So, `AESQ` permutation consists of 20 `AES` rounds. Fig. 2-3 shows a 2-Round `AESQ` Permutation.

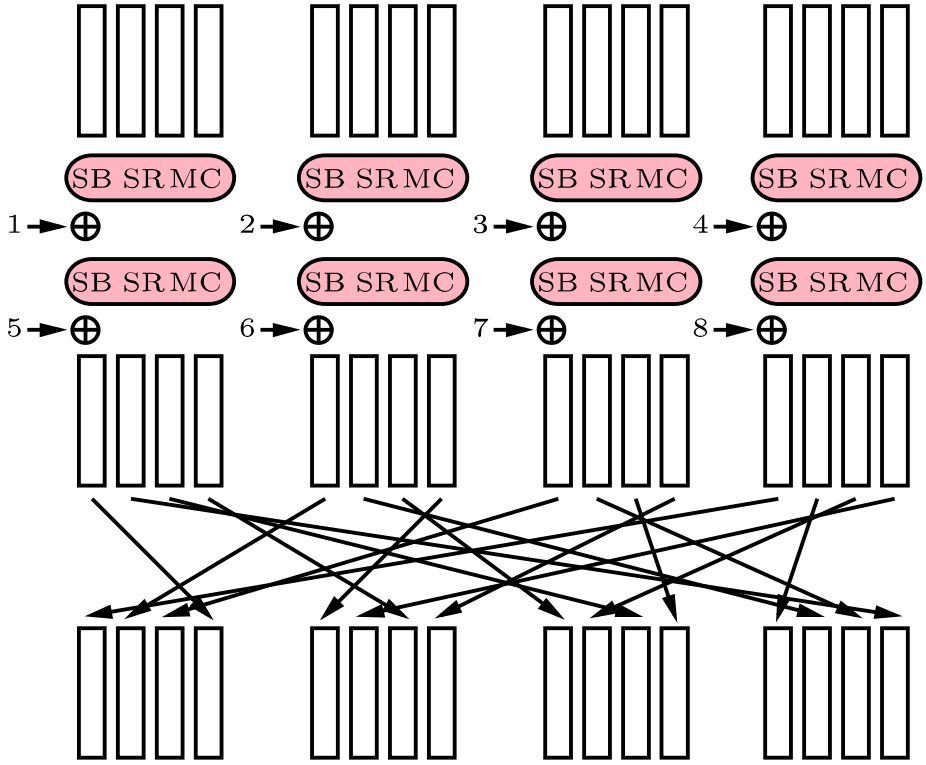


Figure 2-3: 2-Round `AESQ` Permutation

2.1.4 Katan Block Cipher

Katan is a family of lightweight hardware oriented block ciphers proposed by Cannière *et al*. in 2009 [91]. Depending on the block-length, there are three variants-

1. KATAN32- It has a block length of 32 bits.

2. KATAN48- It has a block length of 48 bits.

From	A[0]	A[1]	A[2]	A[3]
To	A[3]	D[3]	C[2]	B[2]
From	B[0]	B[1]	B[2]	B[3]
To	A[1]	D[1]	C[0]	B[0]
From	C[0]	C[1]	C[2]	C[3]
To	A[2]	D[2]	C[3]	B[3]
From	D[0]	D[1]	D[2]	D[3]
To	A[0]	D[0]	C[1]	B[1]

Table 2.1: Shuffle Table of `AESQ`

3. KATAN64- It has a block length of 64 bits.

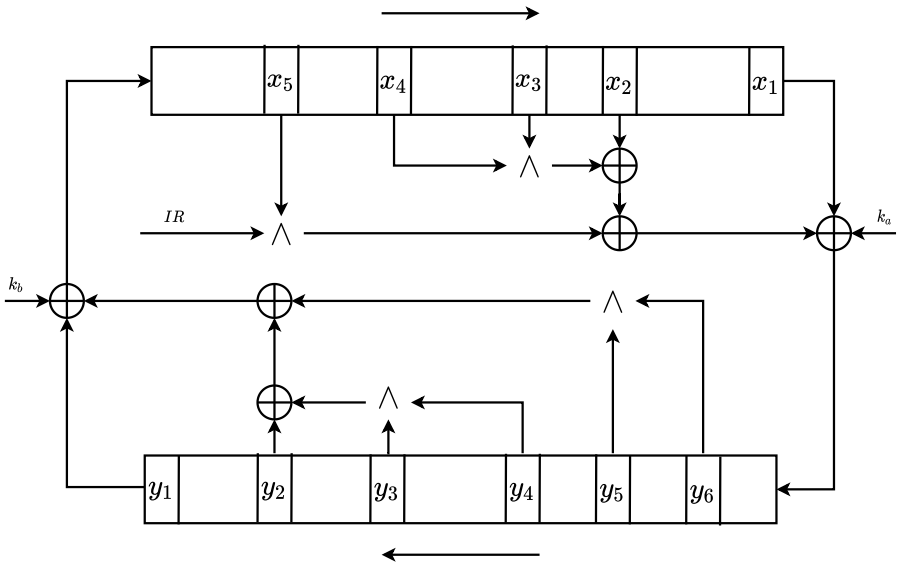


Figure 2-4: Schematic Diagram of a Round Function of KATAN.

The key size is 80 bits for all variants. Initially the plaintext is loaded into two registers, L_1 and L_2. The length of L_1 and L_2 is different across the variants and listed in Table 2.2. The least significant bit (LSB) and the most significant bit (MSB) of the plaintext are loaded to $L_2[0]$ and $L_1[|L_1| - 1]$ respectively where $L_i[p]$ denotes the p-th bit in register L_i ($i \in \{1, 2\}$). In each round, the values in L_1 and L_2 are updated using two non-linear feedback functions $f_a(\cdot)$ and $f_b(\cdot)$. These two functions

are defined as follows:

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

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

where k_a, k_b are two subkey bits and IR is *irregular* update rule. Let a KATAN variant uses r rounds in total. Then the 80-bit key is expanded using a key scheduling algorithm to $2r$ bits and in round i, key bit at position $2i$ and $2i+1$ are used as k_a and k_b respectively. The taps ($\{x_j\}$ and $\{y_j\}$) of L_1 and L_2 are different for the variants and their values are listed in Table 2.2. The *irregular* update rule IR controls the XOR-ing of $L_1[x_5]$ and its values depends on another linear feedback shift register (LFSR) (For more details regarding IR and its values, refer to [91, Table 3]). Fig. 2-4 shows a brief outline about the round function of KATAN.

There are in total 254 rounds for all variants. In a single round, f_a and f_b are applied one, two and three times for KATAN32, KATAN48 and KATAN64 respectively. The key bits that are used in f_a and f_b remains the same even if the functions are applied more than once in the same round. In each round, bit at position i moves to $i+1$ in both L_1, L_2 and the MSBs are discarded. $L_1[0]$ and $L_2[0]$ are updated using the value of $f_b(L_2)$ and $f_a(L_1)$ respectively. Now, the details regarding key scheduling algorithm are provided.

Table 2.2: Parameters of the KATAN Variants.

Variant	$\|L_1\|$	$\|L_2\|$	x_1	x_2	x_3	x_4	x_5
KATAN32	13	19	12	7	8	5	3
KATAN48	19	29	18	12	15	7	6
KATAN64	25	39	24	15	20	11	9

Variant	y_1	y_2	y_3	y_4	y_5	y_6
KATAN32	18	7	12	10	8	3
KATAN48	28	19	21	13	15	6
KATAN64	38	25	33	21	14	9

Key Scheduling Algortihm Consider an 80-bit key K and K_j denotes the j-th bit of K. Initially, K is loaded to an LFSR where the LSB of K is loaded to position 0 of the LFSR. As discussed earlier, in round i, key bits at position $2i$ and $2i+1$ are used from the expanded round key. The 80-bit key K is expanded in the following way:

$$k_i = \begin{cases} K_i & \text{for } 0 \leq i \leq 79 \\ k_{i-80} \oplus k_{i-61} \oplus k_{i-50} \oplus k_{i-13} & \text{Otherwise} \end{cases}$$

The key scheduling algorithm is same for all variants.

2.1.5 Present

Present [62] is a Substitution-Permutation network [165] based block cipher which has a block length of 64 bits. In terms of key size there are two variants- 80-bit and 128-bit. Present contains 31 rounds and the round function is comprised of adding the round key (AddRoundKey), a linear bitwise permutation (pLayer) and a non-linear substitution layer (sBoxLayer).

- **AddRoundKey.** There are in total 32 round keys are used in Present. 31 round keys are used in 31 different rounds and the last one is used for post-whitening. Consider a round key $RK^i = k^i_{63} \cdots k^i_0$ for $1 \leq i \leq 32$ and state bits $a_{63} \cdots a_0$, then the *AddRoundKey* operation is defined as

  $$a_j \leftarrow a_j \oplus k^i_j,$$

  where $0 \leq j \leq 63$.

- **pLayer.** In this layer bits are permutated as shown in Table 2.3. A bit in position i is moved to a new position $P(i)$.

- **sBoxLayer.** Present uses a 4-bit to 4-bit s-box which is applied in parallel 16 times to the Present-state. The input to the s-box is 4 consecutive bits starting from the least significant bit. The input and output to the s-box is shown in

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

Table 2.3: Bit Permutation of Present

Table 2.4.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 2.4: S-box of Present

Key Scheduling Algorithm (KSA).

There are two variants of Present on the basis of key- 80-bit and 128-bit variant. The two key scheduling algorithms are quite similar. Here, the a brief description regarding the two variants are provided.

KSA of 80-bit Key. Initially the key register is loaded with 80-bit supplied key. Let's consider the contents of the key register K is $\kappa_{79}\kappa_{78}\cdots\kappa_1\kappa_0$. In each round the key register is updated in the following way-

1. The key register is left rotated by 61 bits, i. e.
   $[\kappa_{79}\kappa_{78}\cdots\kappa_1\kappa_0] = [\kappa_{18}\kappa_{17}\cdots\kappa_{20}\kappa_{19}]$.
2. S-box is applied on the leftmost 4 bits, i. e. $[\kappa_{79}\kappa_{78}\kappa_{77}\kappa_{76}] = S[\kappa_{79}\kappa_{78}\kappa_{77}\kappa_{76}]$.
3. Round Counter is XOR-ed with κ_{19}, κ_{18}, κ_{17}, κ_{16} and κ_{15}.

In each round, the key bits $\kappa_{79}\kappa_{78}\cdots\kappa_{16}$ are used as round key bits.

KSA of 128-bit Key. Initially the key register is loaded with 128-bit supplied key. Let's consider the contents of the key register K is $\kappa_{127}\kappa_{126}\cdots\kappa_1\kappa_0$. The key bits $\kappa_{127}\kappa_{126}\cdots\kappa_{64}$ constitutes the round key for each round. In each round, the key register is updated in the following way-

1. The key register is left rotated by 61 bits, i. e.

$$[\kappa_{127}\kappa_{126}\cdots\kappa_1\kappa_0] = [\kappa_{66}\kappa_{65}\cdots\kappa_{68}\kappa_{67}].$$

2. Two s-boxes are applied on the leftmost 8 bits, i. e.

   (a) $[\kappa_{127}\kappa_{126}\kappa_{125}\kappa_{124}] = S[\kappa_{127}\kappa_{126}\kappa_{125}\kappa_{124}]$

   (b) $[\kappa_{123}\kappa_{122}\kappa_{121}\kappa_{120}] = S[\kappa_{123}\kappa_{122}\kappa_{121}\kappa_{120}]$

3. Round Counter is XOR-ed with κ_{66}, κ_{65}, κ_{64}, κ_{63} and κ_{62}.

2.2 Some Interesting Underlying Constructions in Block Ciphers and Permutations

In general, a round function is iterated multiple number of times in block ciphers and permutations. However, there are some underlying constructions, due to which a part of the state is dependent on another part of the state over several number of rounds. Such constructions are identified as Super-Sbox [89] and `MegaSbox` (cf. [86]).

2.2.1 Super-Sbox

Here, we briefly describe about the Super-Sbox of `AES` and `AESQ`.

Super-Sbox of `AES`

Super-Sbox [89] was introduced and first studied by Daemen and Rijmen in 2006. Refer to Fig. 2-5 for Super-Sbox construction in `AES`. Consider the diagonal in A (four red-colored bytes). After applying AK, SB and SR, those four bytes aligns in

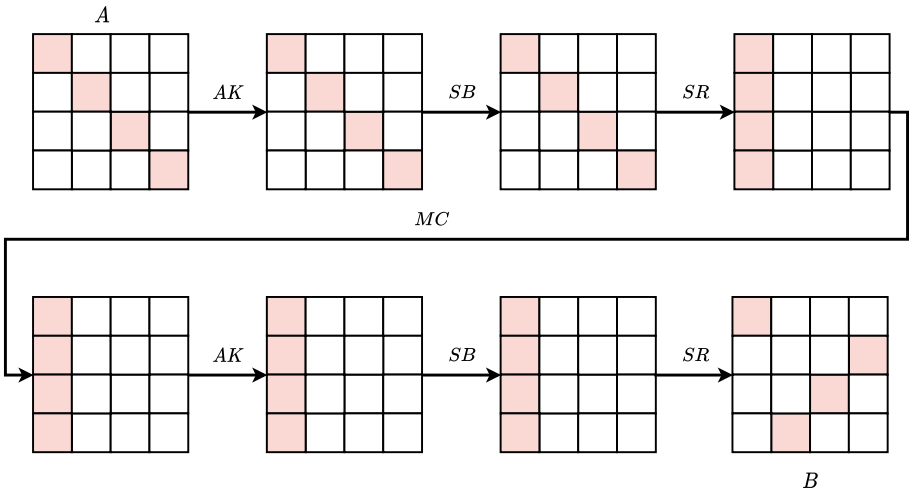


Figure 2-5: `AES` Super-Sbox

a column. The following MC affects only that column. As AK and SB are byte-wise operations, those four bytes remain independent of the other 12 bytes. The last SR operation aligns the bytes to an inverse diagonal in B. These four bytes in B are dependent on the four bytes in A only through the 1.5 rounds. This is conceptualized as Super-Sbox with 32-bit input and 32-bit output. In general, an inverse diagonal in B is uniquely determined by a diagonal in A. There are four Super-Sbox in `AES` state. Fig. 2-6 depicts the four parallel Super-Sbox.

Super-Sbox of `AESQ`

Let us consider round 2 and round 3 (before MixColumns) of `AESQ` permutation. We can consider the input to round 2 as 16 diagonals of 4 bytes each. In round 2, after SubBytes and ShiftRows operation, each of the 16 diagonals aligns in a single column. Effect of MixColumns and shuffle operation is confined within the column. SubBytes

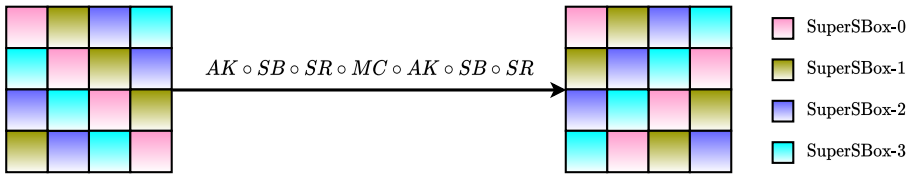


Figure 2-6: 4 Parallel `AES` Super-Sbox

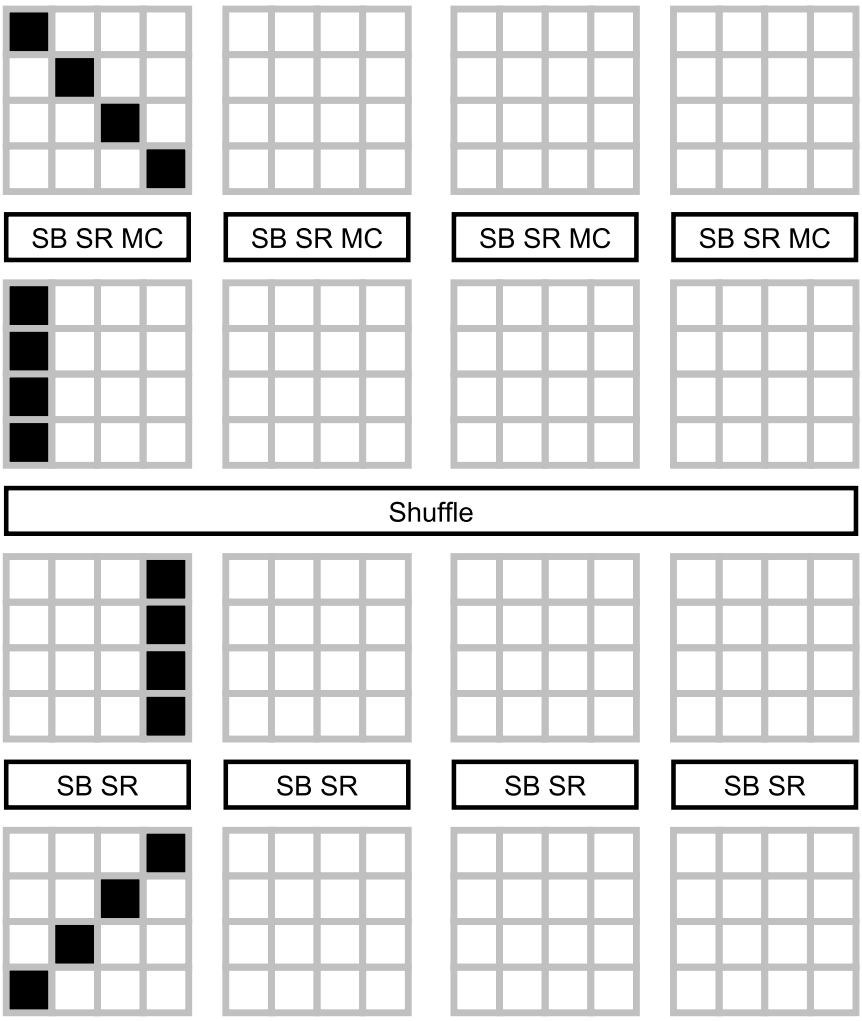


Figure 2-7: Super-Sbox of AESQ

and ShiftRows operation of round 3 dealigns the column into an inverse diagonal. From the above analysis we observe that a group of 4 bytes (a diagonal) in the input to round 2 affects only a group of 4 bytes (inverse diagonal) in the output of round 3 (before MixColumns). These operations can be grouped into a single 32-bit s-box called as Super-Sbox. Therefore, round 2 and round 3 of `AESQ` permutation can be viewed as a single round with 16 parallel Super-Sboxes.

2.2.2 MegaSbox

Now, a brief description about the MegaSbox of `AESQ` is discussed.

MegaSbox of AESQ

The notion of Super-Sbox can be further extended and four round AESQ permutation can be viewed as a single round with 128-bit MegaSboxes [86, 9]. We are now analysing round 2 to round 5 of AESQ permutation. AESQ state consists of 4 registers of 128 bits each. Consider four diagonals each from all the registers. After the SubBytes and ShiftRows operation each of them transforms into a column. MixColumns and adding a constant operation does not influence the other columns. Shuffling accumulates all the four columns into a single register where each of the registers undergoes two rounds of AES-like operation (round 3 and 4) which again does not influence the other registers. Shuffling disperse the columns from a single register to four registers. Round 5 SubBytes and ShiftRows operation dealigns the columns into inverse diagonals. These operations can be grouped into a single 128-bit MegaSbox and round 2 to 5 (before MixColumns) can be viewed as a single round with 4 parallel MegaSboxes. The following MixColumns operation can be considered as mega-linear transformation on AESQ state (512-bits) and called as MegaMixColumns (MMC) operation. Fig. 2-7 and Fig. 2-8 respectively shows how two rounds and four rounds of AESQ permutation exhibits the properties of Super-Sbox and MegaSbox.

2.3 Mode of Operations

Block ciphers can only encrypts a fixed length plaintext to the same length ciphertext. However, to encrypt a message with arbitrary length, modes of operations are used. Here, a brief discussion about major modes of operations are provided.

1. **Electronic Codebook Mode (ECB Mode).** In this mode, each plaintext block is independently encrypted by the block cipher using the same. This mode can operate in parallel. However, by using this mode, identical plaintext blocks are encrypted identical ciphertext blocks. Fig. 2-9 and Fig. 2-10 shows the encryption mode and decryption of ECB respectively.

2. **Cipher Block Chaining Mode (CBC Mode).** In this mode, before encryp-

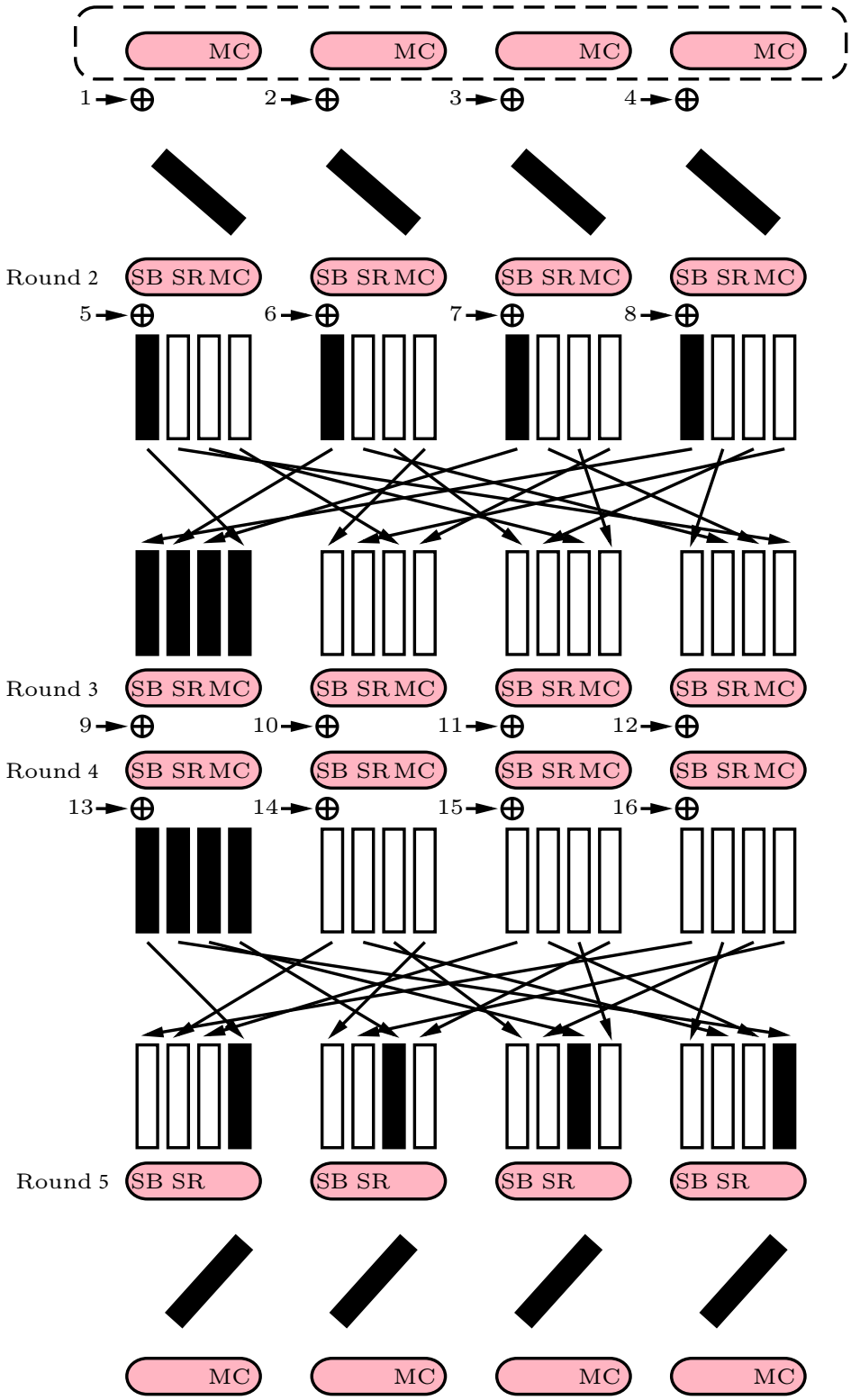


Figure 2-8: MegaSbox of AESQ [9]

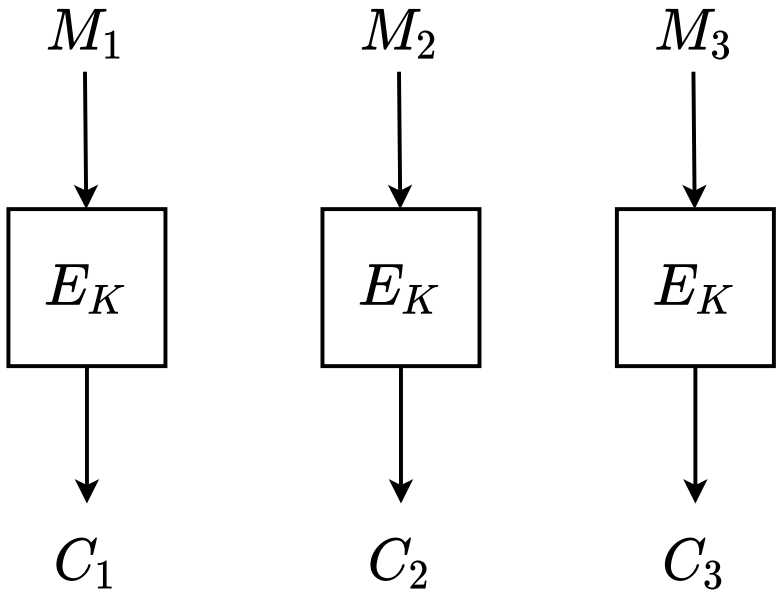


Figure 2-9: ECB Mode Encryption

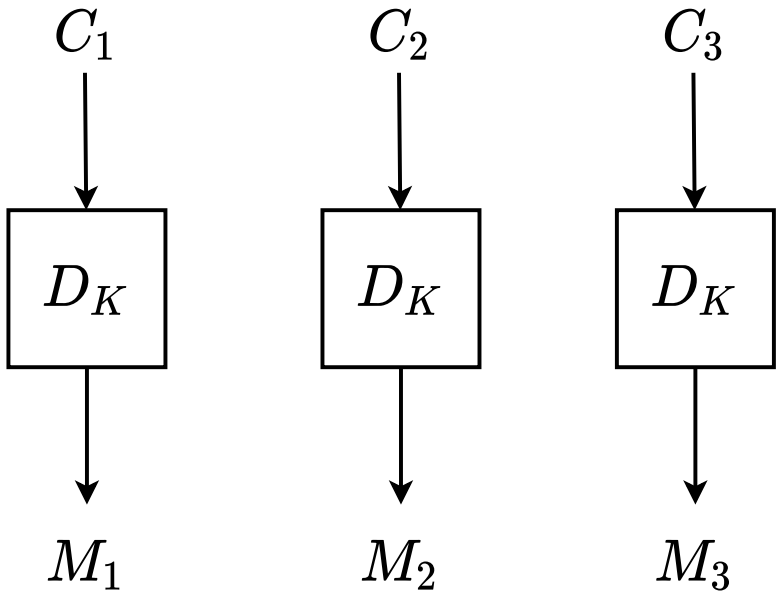


Figure 2-10: ECB Mode Decryption

tion, a plaintext block is XOR-ed with the previous ciphertext block. For the first block of plaintext, an *initialization vector* (IV) is used for XOR-ing. While sending the ciphertext, the IV is also sent for using in the decryption function.

As the encryption of each block is dependent on the previous block, CBC mode can only operate serially during the encryption process. However, as during

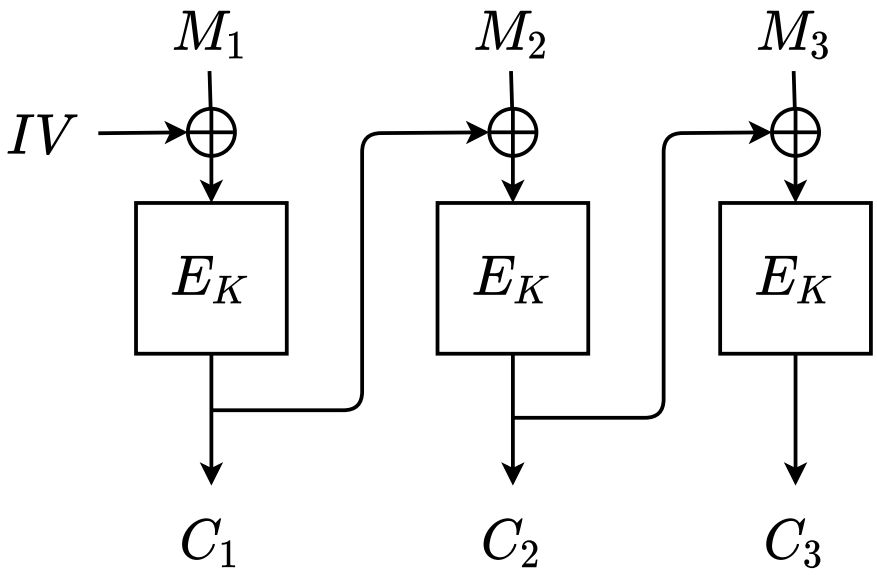


Figure 2-11: CBC Mode Encryption

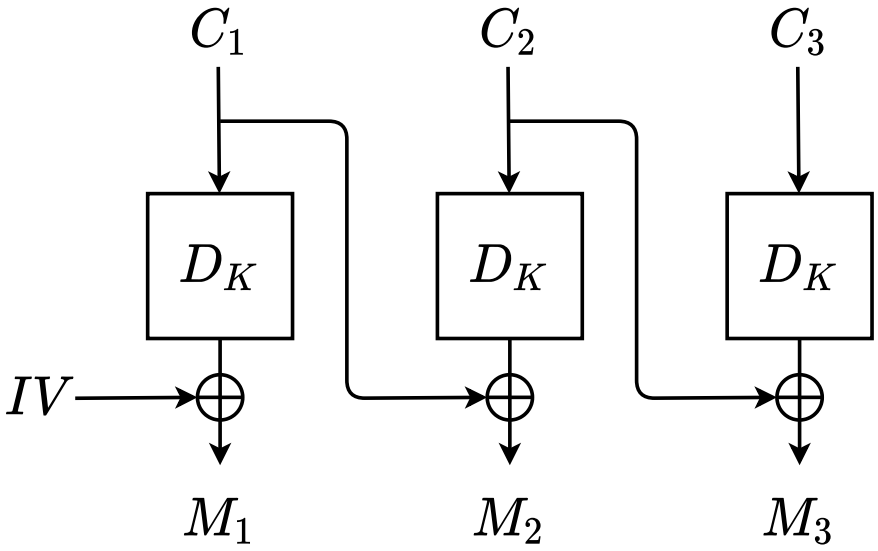


Figure 2-12: CBC Mode Decryption

the decryption all the ciphertext blocks are available, it can be run in parallel. Note that, during the encryption process changing a plaintext block affects the subsequent ciphertext blocks. Fig. 2-11 and Fig. 2-12 shows the encryption and decryption of CBC respectively.

3. **Counter Mode.** In this mode, first a counter is initialized. Then for each block of plaintext, the value of the counter is incremented and encrypted uing the block cipher. The plaintext is XOR-ed with the output of the block cipher

to generate the ciphertext blocks. This mode can be run in parallel. Fig. 2-13 and Fig. 2-14 shows the encryption and decryption of counter mode respectively.

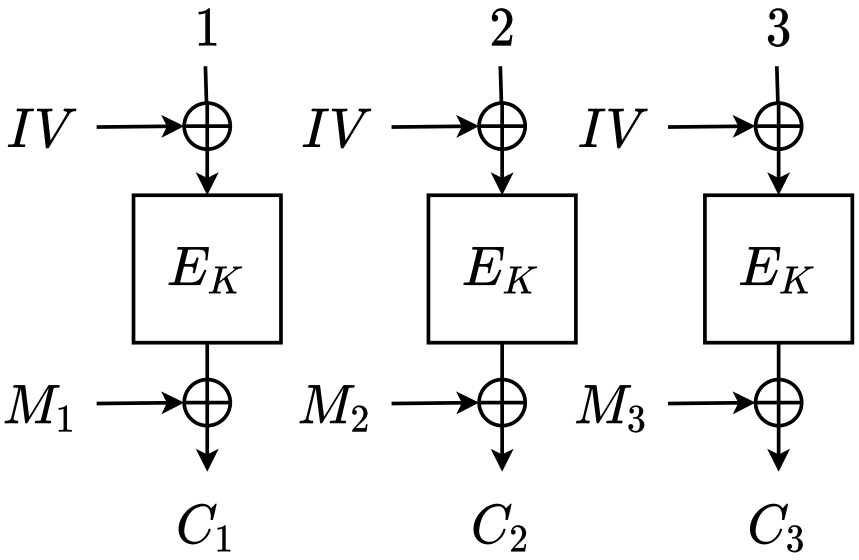


Figure 2-13: Counter Mode Encryption

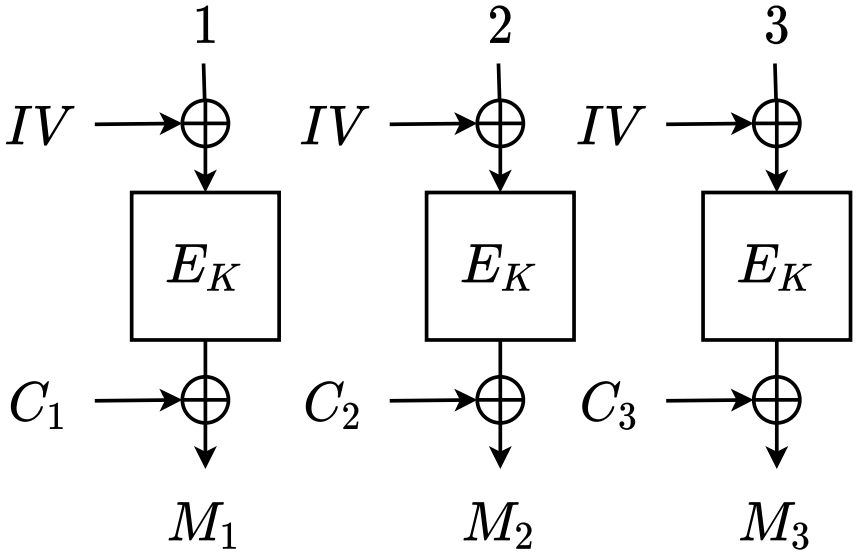


Figure 2-14: Counter Mode Decryption

2.4 Cryptanalysis Basics

Cryptanalysis is the analysis of cryptographic schemes with the aim of finding weaknesses in those schemes. However, for block cipher what should be considered as a weakness needs to be specified. In addition, assumption on the power of the adversary and the way of modeling the cost of an attack are also discussed. Attack scenarios in the quantum computing model are also briefly described.

2.4.1 Attack Goals

For a block cipher, compromising the secrecy of the private key breaks the security. Apart from that, if a probabilistic-polynomial time adversary distinguishes a block cipher from a uniform random permutation, then it is also considered as a valid attack. Based on this, there are mainly two kind of attacks on a block cipher.

1. **Key Recovery Attack.** If the secret key of a block cipher is retrieved by an adversary, then it is said that the cipher is susceptible to key recovery attack. As the public permutations have no secret key, the notion of key recovery attack on such schemes is invalid.

2. **Distinguishing Attack.** If an adversary is able to distinguish a block cipher from a uniform random permutation, then the cipher is considered to be susceptible to distinguishing attack.

2.4.2 Classical Attack Models

The attack models are defined based on how much information is available to the adversary. Generally, there are three kinds of attack models- black-box model, gray-box model and white-box model. In black-box model, an adversary can query the cipher and obtains a response without getting any additional information regarding the implementation of the cipher. In gray-box model, the adversary has access to the implementation of the cipher. The adversary receives information regarding power consumption, electromagnetic emanations of the cipher. In white-box model, the

adversary has full control over the cipher implementation and its execution environment. In this thesis, only the black-box model is considered. The attack scenarios in the black-box model that are considered while mounting cryptographic attacks are sorted below based on the increasing power of the adversary.

1. **Ciphertext Only Attack.** In this model, the adversary has access only to the ciphertexts. He has no knowledge about the corresponding plaintexts.

2. **Known Plaintext Attack.** The adversary has access to plaintext-ciphertext pairs. Although, the adversary can not chose the plaintexts, however, he has knowledge regarding the ciphertexts associated with the plaintexts.

3. **Chosen Plaintext Attack.** In this model, the adversary can obtain ciphertexts associated with the plaintexts chosen by him. The plaintexts should be chosen uniformly at random.

4. **Chosen Ciphertext Attack.** Adversary can choose random ciphertexts and query the decryption oracle to obtain corresponding plaintexts.

5. **Adaptive Chosen Plaintext Attack.** The adversary can query the encryption oracle by adaptively chosing plaintexts to obtain the corresponding ciphertexts.

6. **Adaptive Chosen Ciphertext Attack.** The adversary can query the decryption oracle by adaptively chosing ciphertexts to obtain the corresponding plaintexts.

7. **Related Key Attack.** In this model, the adversary can obtain several ciphertexts/plaintexts corresponding to a plaintext/ciphertext that are encrypted/decrypted with multiple keys. The relation between the keys that are used for encryption/decryption are known to the adversary. Note that, the adversary has no knowledge about the secret key.

2.4.3 Quantum Attack Models

The attacks that are discussed are valid only when classical computing models are considered. One of the major questions in the analysis of quantum attacks is what should be the adversarial model. In this regard, there are mainly two types of adversarial models, mainly Q_1 and Q_2 models, which are used extensively in the literature for mounting quantum attacks on cryptographic schemes [64, 65, 111, 90, 66, 110, 67, 135, 126]. In the quantum computing model, based on the access to a quantum computer for an adversary and on the computing model of the oracle, several attack scenarios are described. The oracle that only accepts only classical queries are referred to as classical oracle whereas the oracles that responds to superposition queries are referred to as quantum oracle. In [214], several adversarial models in the context of post-quantum world have been defined.

1. Q_0 Model. The adversary has access to a classical computer and can query to a classical oracle.

2. Q_1 Model. In addtion to the power in Q_0 model, the adversary has access to a quantum computer.

3. Q_2 Model. In addtion to the power in Q_1 model, the adversary can ask superposition queries to a quantum oracle.

2.4.4 Complexity of Cryptanalysis.

To measure the efficiency of an attack, following three quantities are measured by following the convention in [181].

1. **Data Complexity.** Total number of queries that are queried to the oracle is measured as the data complexity of an attack. Usually, it is measured in terms of the number of encryption/decryption queries in a corresponding attack model. It is also referred to as online computational cost.

2. **Time Complexity.** The offline computational cost of an attack is measured as the time complexity of the attack. Usually, it is measured in terms of the

one execution of the encryption/decryption algorithm. Note that, the computational cost of the oracle related to online queries are not considered as a part of the time complexity [181].

3. **Memory Complexity.** It measures the amount of memory required by an adversary to mount an attack. Usually, it is measured in terms of block length of the cipher on which the attack is mounted.

These three quanttities are together referred toas attack complexity.

2.5 Classical Cryptanalysis Techniques

One of the oldest and strongest generalized cryptanalytic technique is *differential cryptanalysis* (besides linear cryptanalysis [159]) introduced by Biham and Shamir [47, 48, 49, 50, 51, 52]. It was claimed that *differential cryptanalysis* is already known to IBM as "T-attack" and resistance against this was considered as one of the most important design criteria for cryptographic schemes [85]. Typically, in differential cryptanalysis, a cipher is analyzed by considering the propagation of an initial difference through the cipher. Let, E be a cipher and two plaintexts P^1 and $(P^1 \oplus \Delta P)$ are queried to E to obtain C^1 and C^2 respectively. The goal of differential cryptanalysis is to find any non-random relation between ΔP and $(C^1 \oplus C^2)$. Later on, many ciphers are analyzed using this technique [162]. However, differential cryptanalysis has its own limitations and thus often the whole cipher is not susceptible to such attacks. This limitation was overcome by boomerang attack [201] which uses two smaller differential trails to attack the complete cipher. In boomerang attack, an adversary can query the oracle by adaptively chosing plaintexts and ciphertexts (which is a stronger assumption). Rectangle attack [136, 42] is a variant of the boomerang attack which was introduced in the chosen plaintext model. Later, many cryptanalytic techniques are developed based on differential cryptanalysis, like, impossible differential [41], mixture differential cryptanalysis [116], yoyo cryptanalysis [38], retracing boomerang cryptanalysis [99], exchange attack [33], higher order differentials [141], truncated

differential attack [141, 69], extended truncated differential attack [29], related key attack [37, 137, 45], meet-in-the-middle (mitm) attack [40], etc. In such attacks, except the secret key, the underlying design of the cipher is considered publicly known. However, there are also attacks in which the underlying s-box is considered to be secret [199, 68, 192]. In the light of this thesis, a brief discussion on boomerang attack and yoyo attack is provided here.

2.5.1 Boomerang Attack

Boomerang attack, introduced by Wagner, is an extension of the differential cryptanalysis [201]. It attempts to construct a trail for a cipher by a quartet of plaintexts combining two differential trails. It decomposes the cipher into two parts- the upper and lower; likewise, the differential trails corresponding to these parts are called the upper trail and lower trail. E is decomposed into E_0 and E_1, with E_0 being the upper part. Let $Pr[\alpha \xrightarrow{E_0} \beta] = p$ and $Pr[\gamma \xrightarrow{E_1} \delta] = q$ and initially assume $p = q = 1$. The boomerang attack works in the following way.

1. Choose a pair of plaintext P^1, P^2 such that $P^1 \oplus P^2 = \alpha$. Encrypt them using E to obtain C^1, C^2 respectively.

2. Construct C^3, C^4 such that $C^1 \oplus \delta = C^3$ and $C^2 \oplus \delta = C^4$. Decrypt them using E to obtain P^3, P^4 respectively. The value $P^3 \oplus P^4$ should be α.

As $Pr[\alpha \xrightarrow{E_0} \beta] = 1$, $E_0(P^1) \oplus E_0(P^2) = \beta$. Also, $E_1^{-1}(C^1) \oplus E_1^{-1}(C^3) = E_1^{-1}(C^2) \oplus E_1^{-1}(C^4) = \gamma$ as $Pr[\gamma \xrightarrow{E_1} \delta] = 1$. Note that, $E_0(P^i) = E_1^{-1}(C^i)$.

Let E encrypts P^1, P^2, P^3 and P^4 to obtain C^1, C^2, C^3 and C^4. Q^0, Q^1, Q^2 and Q^3 are intermediate encrypted values of P^1, P^2, P^3 and P^4 $\left(Q^i = E_0(P^i) = E_1^{-1}(C^i)\right)$.

Then $Q^1 \oplus Q^2 = \beta$ and $Q^1 \oplus Q^3 = Q^2 \oplus Q^4 = \gamma$. Now,

$$\begin{aligned} Q^3 \oplus Q^4 &= E_0(P^3) \oplus E_0(P^4) \\ &= \Big(E_0(P^1) \oplus E_0(P^2)\Big) \oplus \Big(E_0(P^1) \oplus E_0(P^3)\Big) \oplus \Big(E_0(P^2) \oplus E_0(P^4)\Big) \\ &= \Big(Q^1 \oplus Q^2\Big) \oplus \Big(Q^1 \oplus Q^3\Big) \oplus \Big(Q^2 \oplus Q^4\Big) \\ &= \beta \end{aligned}$$

As $p = 1$, so $P^3 \oplus P^4 = \alpha$. Note that, for any arbitary p, q, $P^3 \oplus P^4 = \alpha$ with probability p^2q^2 under the assumption that the upper and lower trail are independent. Fig. 2-15 shows the framework for boomerang attack.

Later on, several other ciphers are analyzed by using this technique [54]. Biryukov and Khovratovich further improved the boomerang attack by introducing the concept of *S-box switch* and *ladder switch* [57, 55]. These notions add dependency between upper and lower trail.

Sbox Switch and Ladder Switch. Assume, that the last substitution layer in E_0 partitions the state into t parts, *i. e.*, $Q^1 = Q^1_0||Q^1_1 \cdots Q^1_{t-1}$ and $Q^2 = Q^2_0||Q^2_1 \cdots Q^2_{t-1}$. In similar way, β and γ can also be partitioned. Let the last substituion layer in E_0 be S and $S^{-1}(Q^1_i) \oplus S^{-1}(Q^2_i) = \phi$. Consider the i-th partition.

$$\begin{aligned} Q^3_i &= Q^1_i \oplus \gamma_i \\ Q^4_i &= Q^2_i \oplus \gamma_i \end{aligned}$$

For satisfying the E_0 trail, $S^{-1}(Q^3_i) \oplus S^{-1}(Q^4_i) = \phi$ must hold. If $Pr[\phi \xrightarrow{S} \beta_i] = q'$, then q'^2 probability needs to be paid for satisfying this trail. Now, analyze two special cases.

- Case 1 [$\gamma_i = \beta_i$]: In such cases, $Q^3_i = Q^2_i$ and $Q^4_i = Q^1_i$. Now,

$$S^{-1}(Q^3_i) \oplus S^{-1}(Q^4_i) = S^{-1}(Q^2_i) \oplus S^{-1}(Q^1_i) = \phi$$

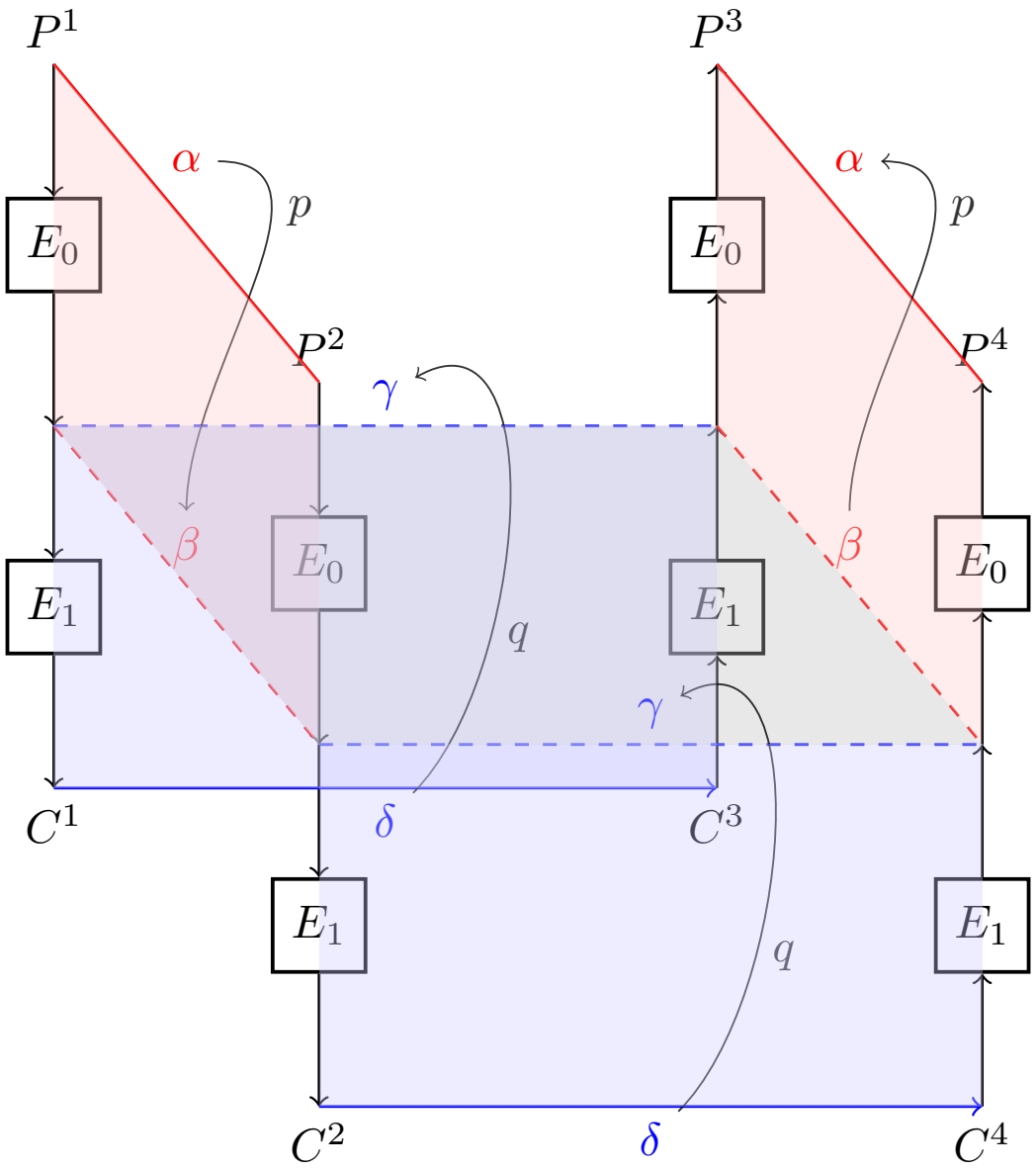


Figure 2-15: Boomerang Attack Framework [178]

Therefore, in such cases probability for one side needs to be paid and other side occurs deterministically; which improves the overall probability by a factor of q'. This is known as *S-box switch*.

- Case 2 [$\gamma_i = 0$]: In such cases, $Q_i^3 = Q_i^1$ and $Q_i^4 = Q_i^2$. Observe that, irrespective of the value of β_i, $S^{-1}(Q_i^3) \oplus S^{-1}(Q_i^4) = \phi$ always holds. The trail probability is improved by a factor of q'^2. This is referred to as *ladder switch*.

In [57, 55], these two notions were exploited to mount related key boomerang attacks on AES-192 and AES-256. Later on, Dunkelman *et al*. formalized these notions by dividing the cipher into E_0, E_m and E_1 [100]. The notion was introduced as sandwich attack which was used to attack 7 rounds out of 8 rounds of Kasumi block cipher [100]. Fig. 2-16 shows the framework for sandwich attack. The probability of a distingisher in this attack framework is p^2q^2r where

$$\begin{aligned} r =& Pr[E_0(P^3) \oplus E_0(P^4) = \beta | (E_0(P^1) \oplus E_0(P^2) = \beta) \\ & \wedge (E_1^{-1}(C^1) \wedge E_1^{-1}(C^3) = \gamma) \wedge (E_1^{-1}(C^2) \wedge E_1^{-1}(C^4) = \gamma)] \end{aligned} \tag{2.1}$$

Cid *et al*. considered the E_m as a S-box layer and developed a tool *Boomerang Connectivity Table* in order to unify *s-box switch* and *ladder switch* [83]. Further, to realize the switching effect on multiple rounds *Boomerang Difference Table* was proposed [203]. Several other papers have followed to extend this concept further [72, 92].

2.5.2 Yoyo Attack

Another interesting cryptanalytic tool is the yoyo game which was recently shown to be very effective in devising distinguishers [176] on the block cipher standard AES. The yoyo strategy was first reported in crypto literature by Biham et al. who used it for the cryptanalysis of SKIPJACK [38]. In the yoyo game, new pairs of plaintexts and ciphertexts are made adaptively from the original pairs. While making new pairs a certain property is kept invariant. A common strategy is the use of zero difference in the pairs. Suppose a pair of plaintext/ciphertexts have certain zero difference after some rounds of a cipher. In the yoyo game, it is verified whether new pairs of plaintexts/ciphertexts that are formed by swapping bytes/words of the original pairs still hold the same zero difference after the same number of rounds of the cipher. Using the yoyo game Biryukov et al. have found a 7-round distinguisher for Feistel networks[58]. In Asiacrypt 2017, Rønjom et al. applied the yoyo game to generic Substitution-Permutation (SP) networks [176] and proposed a generic 2-SP round deterministic distinguisher. As a case-study they applied the strategy on

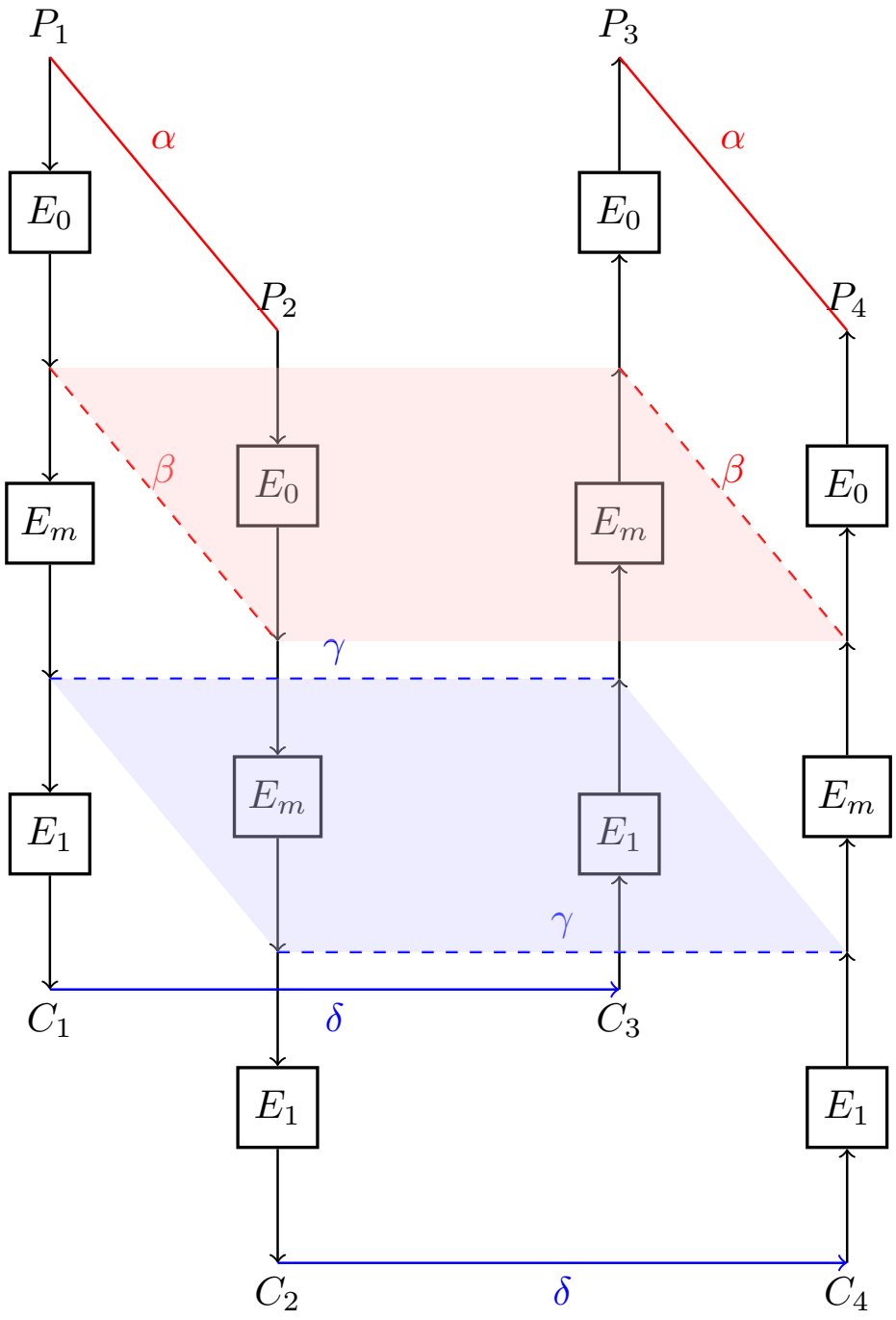


Figure 2-16: Sandwich Attack Framework [178]

variants of AES and found many practical distinguishers on up to 5 rounds of AES. They also reported a distinguisher for 6-round AES with data complexity $2^{122.83}$ and a key recovery attack on 5-round AES with complexity 2^{31} that requires $2^{11.3}$

plaintexts/ciphertexts pairs.

Here, we describe some of the notations used in [176]. A generic permutation is assumed to be of the form of $\mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$ where, $q = 2^k$ given by:

$$F(x) = S \circ L \circ S \circ L \circ S(x)$$

Here, S is considered as a large s-box to be visualized as a concatenation of smaller component s-boxes operating on $\mathbb{F}_q$. The linear layer over $\mathbb{F}_q^n$ is denoted by L. A *word* represents an element of $\mathbb{F}_q$ while the internal *state* is a vector of words $\alpha = (\alpha_0, \alpha_1, \cdots, \alpha_{n-1}) \in \mathbb{F}_q^n$. Based on this the authors in [176] define the Zero Difference Pattern (ZDP) as below:

Definition 2. ***Zero Difference Pattern.****[176] Let, $\alpha \in \mathbb{F}_q^n$ for $q = 2^k$. The Zero Difference Pattern for α is*

$$\nu(\alpha) = (z_0, z_1, ..., z_{n-1}),$$

where $\nu(\alpha)$ takes values in $\mathbb{F}_2^n$ and $z_i = 1$ if $\alpha_i = 0$ or $z_i = 0$ otherwise.

Interestingly, the zero-difference pattern does not consider the nature of separate words when they are non-zero and just classifies them into one category. Our aim is to look further into individual words i.e. we want to investigate the nature of α_i when $z_i = 0$. To facilitate this, we define a *unit* as the element on which the smallest s-box of the cipher is defined. For e.g. for `AES` a unit is a byte. It can be noted that considering the smallest s-box a word is 8-bit, while considering the Super-Sbox representation [89], a word is 32 bits. So, the `AES` state representation changes from $\mathbb{F}_{2^8}^{16}$ to $(\mathbb{F}_{2^8}^4)^4$. When a word uses multiple units, the zero difference pattern does not take into account the nature of these units and marks a word active even if at least one unit in the word is active. We want to study the activity of the units. So we introduce the notion of *Nested Zero Difference Pattern.*

Definition 3. ***Nested Zero Difference Pattern*** *Let, $\alpha \in (\mathbb{F}_q^{\frac{n}{m}})^m$ for $q = 2^k$ and $\alpha_i \in \mathbb{F}_q^{\frac{n}{m}}$ and $\alpha_i = (\beta_{i0}, \beta_{i1}, \cdots, \beta_{i(\frac{n}{m}-1)})$, where β_{ij} is the unit. The Nested Zero*

Difference Pattern (ν^2) defined for α is

$$\nu^2(\alpha) = \{\nu^2(\alpha_0), \nu^2(\alpha_1), \cdots, \nu^2(\alpha_{n-1})\}, \quad \nu^2(\alpha_i) = (y_0, y_1, ..., y_{\frac{n}{m}-1}),$$
$$wt(\nu^2(\alpha)) = \sum_{i=0}^{n-1} wt(\nu^2(\alpha_i)),$$

where $\nu^2(\alpha_i)$ takes values in $\mathbb{F}_2^n$ and $y_i = 1$ if $\beta_{ij} = 0$ or $y_i = 0$ otherwise[1].

The following example will make things clearer.

Example 1. *Here we show the different words of the* ***AES*** *state considering the inputs of the Super-Sbox. Note that the words will change based on whether we are observing the Super-Sbox input or output.*

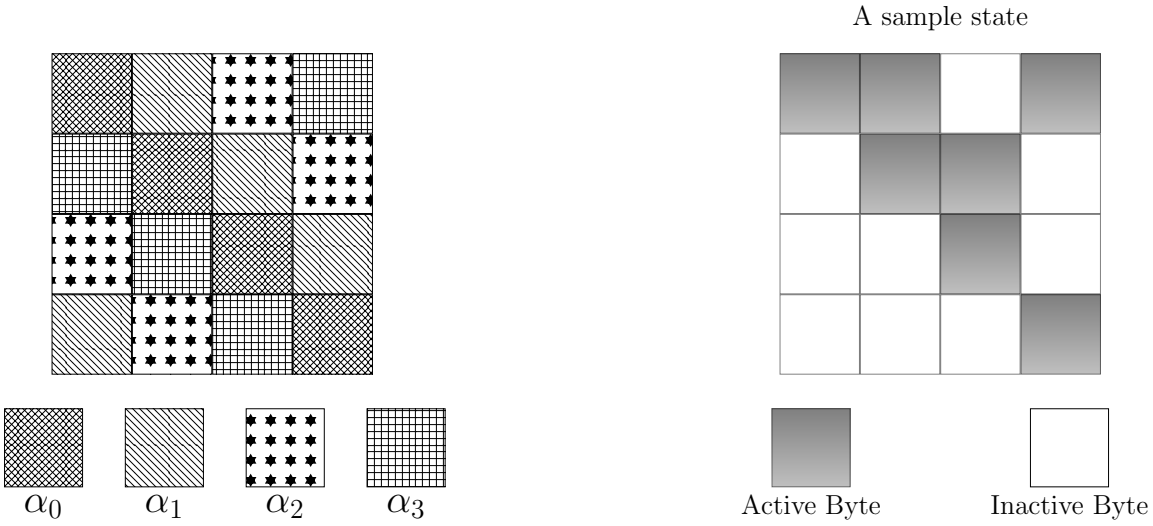


Figure 2-17: Different words and a sample state showing zero and non-zero bytes.

Let us consider the Zero Difference Pattern of the sample state (α) in Fig. 2-17: $\nu(\alpha) = (0, 0, 1, 0)$ *and* $wt(\nu(\alpha)) = 1$. *Thus ZDP considers only one word to be inactive. Let us now look at the Nested ZDP of the state α. It can be easily inferred that Nested ZDP gives more information pertaining to the active words. The idea of Nested ZDP will be useful when we will consider differentials over and above the yoyo game.*

[1]It is understood that $m|n$

$$\nu_1^2(\alpha_0) = (0,0,0,0), \quad \nu_2^2(\alpha_1) = (0,0,1,1), \quad \nu_3^2(\alpha_2) = (1,1,1,1), \quad \nu_4^2(\alpha_3) = (0,1,1,1), \quad wt(\nu^2(\alpha)) = 9.$$

2.5.3 Yoyo Analysis for Two Generic SP-Rounds

Rønjom et al. have carried out yoyo analysis for two generic SP-rounds [176]. Two generic SP-round is $G_2' = L \cdot S \cdot L \cdot S$, where L is the linear transform layer and S is the permutation layer. For simplicity, the final L layer is omitted and the modified two generic SP round is denoted as $G_2 = S \cdot L \cdot S$. They have presented a deterministic distinguisher for G_2. For the explanation of the distinguisher and how it works, we have to go through some definitions originally defined in their work. The next definition signifies how to swap between pairs of texts to form a new pair of texts.

Definition 4. *[176] Let, $\alpha, \beta \in \mathbb{F}_q^n$ be two states and $v \in \mathbb{F}_2^n$ be a vector, then $\rho^v(\alpha, \beta)$ is a new state in $\mathbb{F}_q^n$ created from α, β by swapping components among them. The i^{th} component of $\rho^v(\alpha, \beta)$ is defined as*

$$\rho^v(\alpha,\beta)_i = \begin{cases} \alpha_i, & \text{if } v_i = 1; \\ \beta_i, & \text{if } v_i = 0. \end{cases} \tag{2.2}$$

The following theorem describes the deterministic distinguisher for 2 generic SP-rounds.

Theorem 1. *[176] Let, $p^0, p^1 \in \mathbb{F}_q^n$, $c^0 = G_2(p^0)$ and $c^1 = G_2(p^1)$. For any vector $v \in \mathbb{F}_2^n$, $c'^0 = \rho^v(c^0, c^1)$ and $c'^1 = \rho^v(c^1, c^0)$. Then*

$$\nu(G_2^{-1}(c'^0) \oplus G_2^{-1}(c'^1)) = \nu(p'^0 \oplus p'^1) = \nu(p^0 \oplus p^1).$$

Fig. 2-18 depicts the application of yoyo attack on the $S \circ L \circ S$ construction. The trick is to choose any random pair of plaintexts with certain zero difference pattern and encrypt them using G_2. Then swap words/bytes between the produced ciphertexts and create a new pair of ciphertexts. Decrypt the new pair using G_2 and obtain a new pair of plaintexts. The zero difference pattern of these new pair of plaintexts will be same as the zero difference pattern of the original pairs of plaintexts.

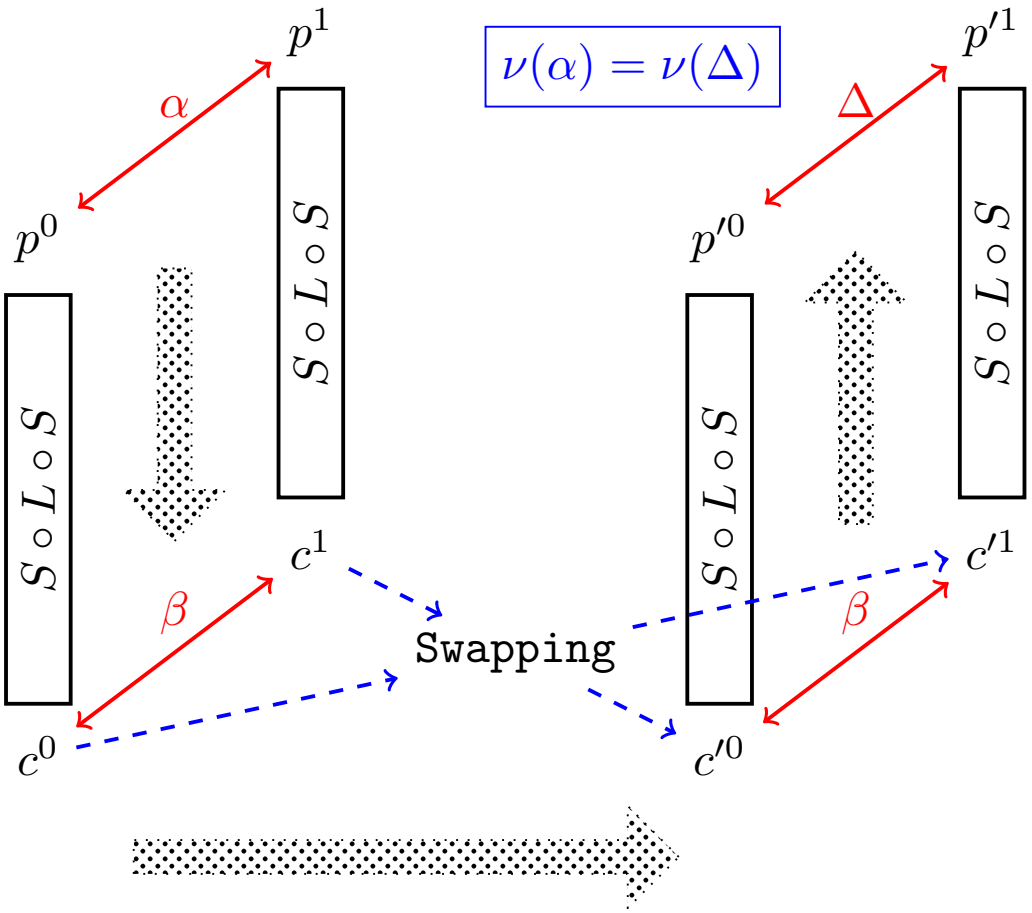


Figure 2-18: Yoyo Attack on $S \circ L \circ S$ Constructon [177]. Here, $p^0 \oplus p^1 = \alpha$, $c^0 \oplus c^1 = c'^0 \oplus c'^1 = \beta$ and $p'^0 \oplus p'^1 = \Delta$.

This event occurs with probability 1. This property of two generic SP-rounds can be exploited to distinguish it from a random construction.

2.6 Quantum Cryptanalysis Tools

Here, some quantum algorithms and how they have been used in cryptanalysis are discussed. First, a brief description of Simon's algorithm is given and how it was applied in [134] is discussed. Next, Grover's search algorithm is briefly mentioned. Finally, the results in [151] and [66] are illustrated upon.

2.6.1 Simon's Algorithm

In the discussion of Simon's algorithm [189], first of all, the problem that it solves needs to be defined. The problem is popularly known as Simon's Problem.

Problem 1: Simon's Problem: Given a boolean function f: $\{0,1\}^n \mapsto \{0,1\}^n$ and the promise that there exists $s \in \{0,1\}^n$ (Simon's promise) such that for any $(x,y) \in \{0,1\}^n$, $[f(x) = f(y)] \iff [x \oplus y \in \{0^n, s\}]$; the goal is to find s.

Classically, this problem can be solved in $\Theta(2^{n/2})$. Using Simon's algorithm this problem can be solved in $O(n)$ quantum complexity. The steps of Simon's algorithm are given in Algorithm 1.

Simon's Algorithm in Cryptography.

In cryptographic applications, it is not the case that always Simon's algorithm can be applied directly. The reason is that sometimes the function that needs to be analyzed has some partial period apart from having a full period. Kaplan *et al.*. have also shown the application of Simon's algorithm under such constraints. This particularly handles the conditions where $\exists t$ such that $f(x) = f(x \oplus t), t \notin \{0, s\}$. They have used $\epsilon(f,s)$ for computing the success probability of Simon's algorithm based on the rate of the collision, where

$$\epsilon(f,s) = \max_{t \in \{0,1\}^n \setminus \{0,s\}} Pr_x[f(x) = f(x \oplus t)].$$

The following theorems in [134] handles the conditions when Simon's promise does not hold precisely.

Theorem 2. ***(Simon's Algorithm with Approximate Promise).*** *[134] If $\epsilon(f,s) \leq p_0 \leq 1$, then with probability at least $1 - \left(2\left(\frac{1+p_0}{2}\right)^c\right)^n$ Simon's algorithm returns s at the expense of cn queries.*

If there is no bound on $\epsilon(f,s)$, then it is not possible to recover s always. But we can find a t such that $\Pr_x[f(x) = f(x \oplus t)]$ is very high. The following theorem

Algorithm 1 Simon's Algorithm

1. Let's consider a unitary map U_f given by $|x, y\rangle \mapsto |x, y \oplus f(x)\rangle$. Two registers are initialized with n-qubit state $|0\rangle$ each. Hadamard transform $H^{\otimes n}$ is applied to the first register to obtain quantum superposition
$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|0\rangle.$$
2. f is queried to U_f using these two registers to obtain
$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle.$$
3. The second register is measured. Measuring the value in the second register collapses the value in both the registers. If the value in the second register is $f(z)$, then the state in the first register should be a superposition state due to Simon's promise. The state in the first register is
$$\frac{1}{\sqrt{2}}(|z\rangle + |z \oplus s\rangle).$$
4. On first register, Hadamard transform $H^{\otimes n}$ is again applied to obtain
$$\frac{1}{\sqrt{2}} \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{y.z}(1 + (-1)^{y.s})|y\rangle.$$
5. Measuring the first register collapses it to a random vector y such that $y.s = 0$. The y vectors with $y.s = 1$ have 0 amplitude; so, the first register never collapses to such values.
6. Steps 1 to 5 are repeated $O(n)$ times, produceing $n-1$ random vectors orthogonal to s. These can be solved to retrieve the value of s.

dictates that.

Theorem 3. ***(Simon's Algorithm without Promise).*** *[134] After the execution of cn steps of Simon's algorithm, if t is orthogonal to all vectors u_i returned by each step of the algorithm, then $Pr_x[f(x) = f(x \oplus t)] \geq p_0$ with probability at least $1 - \left(2\left(\frac{1+p_0}{2}\right)^c\right)^n$.*

In both cases, if $c \geq \frac{3}{(1-p_0)}$ then the probabilities become high.

2.6.2 Grover's Algorithm

Grover's algorithm [120] is used for searching in a completely unstructured dataset. Consider a function $f : \{0,1\}^n \rightarrow \{0,1\}$ $(N = 2^n)$ and there exists a state w such that

$$f(x) = \begin{cases} 1, \text{ for } x = w \\ 0, \text{ for } x \neq w. \end{cases}$$

Suppose, f can be realized using a black-box reversible function B_f, where

$$B_f|x\rangle|a\rangle = |x\rangle|f(x) \oplus a\rangle, \forall x \in \{0,1\}^n \text{and } a \in \{0,1\}.$$

The problem is to find a x such that $f(x) = 1$. Any deterministic classical algorithm can solve the problem by querying the function f $O(2^n)$ times. However, leveraging on quantum computing techniques, Grover's algorithm can solve the problem by making $O(\sqrt{2^n})$ queries; which is a significant speed-up in comparison to the classical counterpart.

Before describing Grover's algorithm, consider two unitary maps on n qubits, $U_f : |x\rangle \mapsto (-1)^{f(x)}|x\rangle$ and $U : |x\rangle \mapsto (-1)^{g(x)}|x\rangle$, where the function $g : \{0,1\}^n \rightarrow \{0,1\}$ is defined as

$$g(x) = \begin{cases} 1, \text{ for } x = 0^n \\ 0, \text{ for } x \neq 0^n. \end{cases}$$

Fig. 2-19 shows the implementation of U_f using the black-box B_f and an ancillary qubit by employing the phase kick-back phenomenon. Classically, $g(x)$ can be realized by computing $(\overline{x}_0 \wedge \overline{x}_1 \cdots \wedge \overline{x}_{n-1})$, where x_i corresponds to $(i+1)^{\text{th}}$ bit of x. Thus, U can be implemented by replacing B_f with a reversible quantum circuit for the following map:

$$|x\rangle|a\rangle \mapsto |x\rangle|a \oplus (\overline{x}_0 \wedge \overline{x}_1 \cdots \wedge \overline{x}_{n-1})\rangle.$$

Now, Grover operator G is defined as $U_{\psi^\perp}U_f$ where $U_{\psi^\perp} = H^{\otimes n}UH^{\otimes n}$. Based on

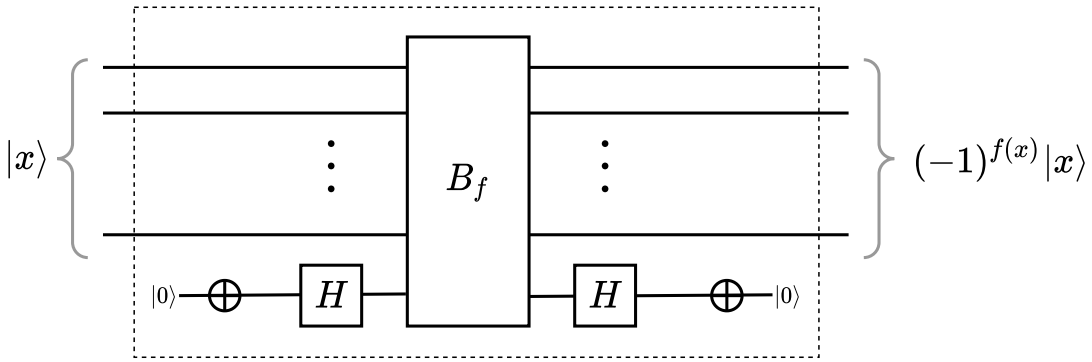


Figure 2-19: Implementation of U_f using B_f

this, the steps of Grover's algorithm is shown in Algorithm 2.

Algorithm 2 Grover's Algorithm

1. An n-qubit register Z is initialized. Hadamard transformation $H^{\otimes n}$ is applied on Z.
2. Grover operator G is applied $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ times to the register Z.
3. Z is measured and the result is output.

Consider instead of iterating $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ times, G is iterated k times in Step 2 of Algorithm 2. Two superposition states $|U\rangle$ and $|V\rangle$ are defined as

$$|U\rangle = \frac{1}{\sqrt{u}} \sum_{x \in U} |x\rangle$$

$$|V\rangle = \frac{1}{\sqrt{v}} \sum_{x \in V} |x\rangle$$

where $U = \{x \in \{0,1\}^n : f(x) = 1\}$, $u = |U|$ and $V = \{x \in \{0,1\}^n : f(x) \neq 1\}$, $u = |V|$. Based on this, following proposition can be stated.

Proposition 1 ([120, 207]). *As hadamard operation is applied in Step 1 of Algorithm 2, the state is in the superposition* $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$. *After application of* k *iterations in Step 2, the resulting superposition of the state is*

$$sin((2k+1)\theta)|U\rangle + cos((2k+1)\theta)|V\rangle,$$

where $\theta = sin^{-1}\sqrt{\frac{u}{N}}$.

Note that, in this case $u = 1$, as it is assumed that only when $x = w$, $f(x) = 1$. However, Proposition 1 holds for any arbitary value of u. From Proposition 1, it can be concluded that after the measurement in Step 3, a right/correct state is output by Grover's algorithm with probability $|sin((2k+1)\theta)|^2$. When $k = \lfloor \frac{\pi}{4}\sqrt{N/u} \rfloor$, a state from U is measured with probability at least $\frac{1}{2}$ [120, 76]. Now, the technique of mounting key recovery attack on block ciphers using Grover's search is discussed.

Key Recovery Attack on Block Cipher using Grover's Algorithm.

Yamamura and Ishizuka have shown that Grover's algorithm can be used to mount key recovery attack on a block cipher [211]. This attacks is generic as it does not rely on the construction of the block cipher. Consider a block cipher E which uses a k-bit key $\mathcal{K}$ and its block length is n. The encryption of a message m by block cipher E using the key $\mathcal{K}$ is denoted by $E_{\mathcal{K}}(m)$. Algorithm 3 shows the steps of mounting Grover's attack on E.

Algorithm 3 Grover's Attack on Block Cipher

1. A plaintext-ciphertext pair (P, C) is prepared where $C = E_{\mathcal{K}}(P)$. The function f is defined as
$$f(x) = \begin{cases} 1 \text{ if } E_x(P) = C \\ 0 \text{ if } E_x(P) \neq C \end{cases}$$
2. A k-qubit register Z is initialized. Hadamard transformation $H^{\otimes n}$ is applied on Z to obtain
$$\frac{1}{\sqrt{2^k}} \sum_{x \in \{0,1\}^k} |x\rangle.$$
3. Grover operator G is applied $\lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$ times to the register Z.
4. Z is measured and the right key $\mathcal{K}$ is obtained with probability at least $\frac{1}{2}$.

2.6.3 Simon's Algorithm with Asymmetric Queries

Leander *et al.* has combined Grover's search algorithm with Simon's algorithm to recover keys for FX construction [151]. This combination of algorithms for finding a period has a huge impact on cryptographic schemes and Bonnetain *et al.* have formally defined the problem as *Asymmetric Search of a Period* [66].

Problem 3: Asymmetric Search of a Period [66]: Consider a family of functions F indexed by $\{0,1\}^m$, denoted by $F(i,\cdot) = f_i(\cdot)$ and a function g; they are defined as

$$F : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^l$$
$$g : \{0,1\}^n \to \{0,1\}^l$$

The problem is to find an i_0 and a s such that $\forall x \in \{0,1\}^n$, $f_{i_0}(x) \oplus g(x) = f_{i_0}(x \oplus s) \oplus g(x \oplus s)$ for a certain s, under the following assumptions,

- Quantum oracle access to F is given.
- In $Q1$ model, classical oracle access to g is given whereas in $Q2$ setting g is accessed as quantum oracle.
- There is exactly one $i \in \{0,1\}^m$ such that $f_i \oplus g$ has a hidden period.

Bonnetain *et al.* have observed that while testing whether $f_i \oplus g$ have period or not; the function g always remains same. Leveraging on that the number of queries to g is reduced and the superposition

$$|\psi_g\rangle = \bigotimes^{cn} \Big(\sum_{x \in \{0,1\}^n} |x\rangle |g(x)\rangle \Big)$$

is used several times. In $Q2$ model, g is queried using superposition queries; whereas in $Q1$ only classical queries are allowed to make to g. From $|\psi_g\rangle$,

$$|\psi_{f_i \oplus g}\rangle = \bigotimes^{cn} \Big(\sum_{x \in \{0,1\}^n} |x\rangle |f_i(x) \oplus g(x)\rangle \Big)$$

is constructed by making quantum superposition queries to f_i.

In our work, we have used the existing techniques in [66] to attack encryption schemes. A brief overview of all algorithms and their corresponding complexities introduced in [66] to solve the problem of *Asymmetric Search of a Period* is given here (For details, refer to [66]).

- **Alg-PolyQ2-** Solves the problem of *Asymmetric Search of a Period* in $Q2$ model. It is allowed to make quantum superposition queries to g for online computations.

- **Alg-ExpQ1-** Solves the problem of *Asymmetric Search of a Period* in $Q1$ model. It is allowed to make classical queries to g for online computations.

During offline computations both Alg-PolyQ2 and Alg-ExpQ1 find an i using Grover's search algorithm, such that for that fixed i, $f_i \oplus g$ has a period. Note that, both algorithms never returns the actual period of $f_i \oplus g$. For finding the period, Simon's algorithm is applied to $f_i \oplus g$. In the $Q1$ model, for finding period Simon's algorithm is applied by making classical queries to the oracle. In regard to this **Alg-SimQ1** has been defined in [66].

Cost Estimation.

The attacks, presented in this work, make use of Alg-ExpQ1 and Alg-SimQ1. The following two propositions (proposed in [66]) are regarding the cost estimation when these algorithms are applied to mount attacks.

Proposition 2. ***(Proposition 3 in [66])*** *Let c be a sufficiently large constant, m*

is in $O(n)$ and $g \oplus f_{i_0}$ has a period for a good i_0. Assume that

$$\max_{\substack{t\in\{0,1\}^n\setminus\{0^n\},\\ i\in\{0,1\}^n\setminus\{i_0\},\\ x\in\{0,1\}^n}} \Pr[(f_i \oplus g)(x \oplus t) = (f_i \oplus g)(x)] \leq \frac{1}{2} \tag{2.3}$$

holds. Then a good $i \in \{0,1\}^m$ with probability $\Theta(1)$ is found by Alg-ExpQ1 by making classical and quantum queries to g and F respectively. The number of classical and quantum queries are $O(2^n)$ and $O(n2^{m/2})$ respectively. If for evaluating F once T_F is the required time, then Alg-ExpQ1 executes the offline computations in time $O\Big((n^3 + nT_F)2^{m/2}\Big)$. Note that, in offline computation the time required for preparing the state $|\psi_g\rangle$ is not included.

Proposition 3. ***(Proposition 4 in [66])*** *Suppose that, $f_{i_0} \oplus g$ has a period $s \neq 0$ and satisfies*

$$\max_{t\neq\{s,0^n\}} Pr_x[(f_i \oplus g)(x \oplus t) = (f_i \oplus g)(x)] \leq \frac{1}{2}. \tag{2.4}$$

Then Alg-SimQ1 makes $O(2^n)$ classical queries to g and cn queries to f_{i_0} and returns the period s with a probability at least $1 - 2^n.(3/4)^{cn}$. If T_f is the required time for evaluating f_{i_0} once, then the offline computation of Alg-SimQ1 runs in time $O(n^3 + nT_f)$.

For performing attacks in $Q1$ model, to form $|\psi_g\rangle$ whole codebook of g should be queried. In order to reduce the number of queries to g, a trade-off between online classical queries to g (Data complexity) and offline quantum computations (Time complexity) exists. In Chapter 7, number of online classical queries is denoted by D and number of offline computations is denoted by T.

2.7 Other Tools

Here, details regarding data complexity, success probability, signal-to-noise ratio and ranking test are discussed.

2.7.1 Data Complexity and Success Probability

For a distinguishing event, the data complexity and the success probability depend on the probabilities p and $p_0 = p(1+q)$ of the same event respectively in the random case and in the case of the algorithm under consideration. In [174], detailed analysis of various relations between data complexity of the distinguisher and the corresponding success probability is presented. We use the most general result from [174, Theorem 2], which involves no crude approximation such as ignoring the constant terms or assuming p, q to be small.

Theorem 4. *[174, Theorem 2] Suppose, the event e happens in uniform random bitstream with probability p and in keystream of a stream cipher with probability $p(1+q)$. Then the data complexity of the distinguisher with false positive and false negative rates α and β is given by*

$$n > \frac{\left(\kappa_1\sqrt{1-p} + \kappa_2\sqrt{\big(1+q\big)\big(1-p(1+q)\big)}\right)^2}{pq^2} \tag{2.5}$$

where $\Phi(-\kappa_1) = \alpha$ and $\Phi(\kappa_2) = 1-\beta$.

For computing success probability, we consider $\kappa_1 = \kappa_2$ in Theorem 4, which gives us $\alpha = \beta$. Then the success probability is given by $(1-\beta)$. Note that, in the theorem data complexity essentially refers to sample complexity.

Experimental Verification of Success Probabilities.

For experimental verification of success probabilities of distinguishers, the strategy from [180] has been followed. First, consider a blackbox which can act as either a cipher $\mathcal{C}$ or a uniform discrete random permutation $\mathcal{R}$. Then the experiment is run two times in the following ways:

1. Consider the blackbox as $\mathcal{C}$ and repeat the experiment a_c times.

2. Consider the blackbox as $\mathcal{R}$ and repeat the experiment a_r times.

Let out of (a_c+a_r) experiments, distinguisher decides it as $\mathcal{C}$ o_c times and as $\mathcal{R}$ o_r times. n_{FP} and n_{FN} denotes the number of false positives and false negatives respectively. Based on this parameters, the confusion matrix is shown in Table 2.5.

Table 2.5: Confusion Matrix of $\mathcal{C}$ and $\mathcal{R}$

Actual \ Observed	$\mathcal{C}$	$\mathcal{R}$
$\mathcal{C}$	$o_c - n_{FP}$	n_{FN}
$\mathcal{R}$	n_{FP}	$o_r - n_{FN}$

Then the success probability is calculated by:

$$\begin{aligned} Pr[Success] &= \frac{(o_c - n_{FP}) + (o_r - n_{FN})}{o_c + o_r} \\ &= \frac{(o_c - n_{FP}) + (o_r - n_{FN})}{a_c + a_r}. \end{aligned}$$

2.7.2 Signal-to-Noise Ratio and Ranking Test

While mounting a key recovery attack, situations may arrive when it is not possible to distinguish the right pair from the wrong ones. In such cases, the notion of *signal-to-noise* ratio is used. *Signal-to-noise* ratio (S/N) is used to determine the number of right pairs required to recover the right key. The right key can be suggested by both right pairs or wrong pairs. The ones that are suggested by the right pair are called signal whereas the ones that are suggested by the wrong pair are called noise. Let M be the number of pairs queried by the adversary and p be the probability of the characteristic. Then the number of right pair is Mp. As each right pair suggests the right key one time, hence the amount of signal is Mp. Now the number of wrong pairs is $M(1-p)$. Let a filtering technique is used and a wrong pair survive the filtering with probability β. Therefore, after filtering the remaining number of wrong keys is $M(1-p)\beta$. Consider η be the average number of key candidates suggested by the wrong pair. Note that, such suggestions consist of both right and wrong key candidates. So, the total number of keys suggested by wrong pairs is

$M(1-p)\beta\eta$. Under the assumption that the keys suggested by wrong pairs are uniformly distributed, the amount of noise is $M(1-p)\beta\eta 2^{-k}$, where k is the length of the guessed key in bits. Therefore *signal-to-noise* ratio is $\frac{Mp}{M(1-p)\beta\eta 2^{-k}} = \frac{2^k p}{(1-p)\beta\eta}$.

A counter is maintained for each key suggested by either right or wrong pairs. The value of the counter for each key depends on the *signal-to-noise* ratio. If $S/N > 1$ then the right key is suggested more than the other keys whereas for $S/N < 1$ the right key is suggested fewer times than the wrong ones. By analysing the counters, the right key can be detected. More details regarding the *signal-to-noise* ratio is provided in [143, 181]. For $S/N > 1$, generally the candidate key with highest counter value is considered as the right candidate. But cases may arrive when the counter value for the right candidate is not maximum. Thus, as stated in [181], several key candidates whose counter value is close the highest one is considered as the candidate key. This method is known as ranking test.

In the context of differential cryptanalysis, the relation between the number of right pairs required to identify the unique key, the number of key candidates whose counter value is close to the highest one and the success probability was given by Selçuk in [182]. Let M be the number of pairs queried to the oracle, p be the probability of the characteristic and let k be the length of the guessed key in bits. Without loss of generality, let the right key be denoted by $\mathcal{K}_0$ and $\mathcal{K}_1, \cdots, \mathcal{K}_{2^k-1}$ denote the wrong keys. A plaintext pair suggests $\mathcal{K}_i$ as key candidate with probability p_i and the counter value for each $\mathcal{K}_i$ is T_i. Under the assumption that T_i's are independent and identically distributed (i.i.d.), the probability of any of the wrong keys being suggested as the right one is the same and is denoted by p_w. For $1 \leq i \leq 2^k - 1$, T_i follows the binomial distribution $\mathcal{B}(M, p_w)$ and T_0 follows the binomial distribution $\mathcal{B}(M, p_0)$. For large values of M, these binomial distributions can be approximated

by normal distribution $\mathcal{N}(\mu_w, \sigma_w^2)$ and $\mathcal{N}(\mu_0, \sigma_0^2)$ where

$$\mu_0 = p_0 M, \qquad \sigma_0^2 = p_0(1-p_0)M \approx p_0 M$$
$$\mu_w = p_w M, \qquad \sigma_w^2 = p_w(1-p_w)M \approx p_w M.$$

The right keys are deterministically suggested by the right pairs and probabilistically suggested by the wrong pairs; whereas wrong keys are probabilistically suggested by both the right and the wrong pairs. If a certain key is suggested as the right key candidate by a random pair with probability p_r, then

$$p_0 = p + (1-p)p_r \approx p + p_r$$
$$p_w = p_r.$$

The attack is successfully performed if $\mathcal{K}_0$ is ranked among the top r candidates on the basis of the counter values. Let ϕ be the probability density function and Φ be the cumulative distribution function. Then the success probability P_s can be given by

$$P_s = \int_{-\frac{\mu_0-\mu_q}{\sqrt{\sigma_0^2+\sigma_q^2}}}^{\infty} \phi(x)dx,$$

where $\sigma_q = \frac{\sigma_w}{\phi(\Phi^{-1}(1-2^{log_2 r-k}))} 2^{-\frac{2k-log_2 r}{2}}$ and $\mu_q = \mu_w + \sigma_w \Phi^{-1}(1-2^{log_2 r-k})$ [182]. Based on this, the following propositions connect success probability, data complexity and the number of top ranked values that should be considered as right key candidate.

Proposition 4. *[182] Let the correct key $\mathcal{K}_0$ of length k is among the top r values of key counters with probability P_s when a differential attack with characteristic probability p is mounted using M plaintext-ciphertext pairs and signal-to-noise ratio of S_N. Under the assumptions that the counters corresponding to the wrong keys are independent and follows an identical distribution and the value of k and M is too large, then P_s can be expressed as a function of the other variables by the following equation:*

$$P_s = \Phi\left(\frac{\sqrt{pMS_N} - \Phi^{-1}(1-2^{log_2 r-k})}{\sqrt{S_N+1}}\right)$$

Proposition 5. *[182] Let the correct key $\mathcal{K}_0$ of length k is among the top r values of key counters with probability P_s when a differential attack with characteristic probability p is mounted using M plaintext-ciphertext pairs and signal-to-noise ratio of S_N. Under the assumptions that the counters corresponding to the wrong keys are independent and follows an identical distribution, the value of k and M is too large, then M can be expressed as a function of the other variables by the following equation:*

$$M = \frac{(\sqrt{S_N + 1}\Phi^{-1}(P_s) + \Phi^{-1}(1 - 2^{log_2 r - k}))^2}{S_N} p^{-1}.$$

The rest of the thesis is described by following the conventions and methodology discussed here. The main part of the thesis is based on the tools and techniques provided in this chapter.

3 DIFFERENTIAL ATTACKS ON FlexAEAD

Contents

In the modern era, the aim is to connect each of the physical devices, even the miniature ones, with the internet so that they can be monitored and controlled remotely for maximum utilization. These devices are powered with the ability of communicating among themselves. Such a huge interconnected system, consisting of numerous tiny devices, is not free from vulnerabilities. Moreover, a security breach in such systems can be catastrophic. So, a major concern in the world of *internet-of-things* is how to provide security and privacy to each system with the constraints of limited power and area. SKINNY [34], PRESENT [62], QARMA [23], KATAN and KTANTAN [91], GIFT [28] are some of the block ciphers which are designed for such constrained environments. Until recently, no standardization process has been introduced (like AES Development [6], SHA-3 Project [5], CAESAR Competition [1]) for cryptographic schemes in lightweight environments. NIST LightWeight Cryptography (LWC) competition [4] is a major step towards addressing these issues. There are a total of 57 submissions in this competition. Apart from authenticated encryption algorithms in lightweight environment, some of the designs also comprise of hash functions. Some of them have also provided new primitives for block cipher design.

FlexAEAD is one of the round-1 candidates proposed by Nascimento and Xexéo in NIST LWC competition [96]. It is a family of lightweight authenticated encryption schemes with associated data. In this version, the processing of Associate Data (AD) has been added to the original variants [97, 95, 94]. There are mainly three variants of FlexAEAD that have been listed with block sizes of 64, 128 and 256 bits. In general, a FlexAEAD scheme is denoted by FlexAEAD-b, with b being the block size. The size of nonce and tag is the same as block size across all variants. The length of key is 128 bits for FlexAEAD-64 and FlexAEAD-128 whereas it is 256 bits for FlexAEAD-256. The nonce in FlexAEAD is used to generate sequence numbers which are eventually XOR-ed with associated data, plaintext and intermediate-state to produce ciphertext-tag pair. The lightweight of FlexAEAD essentially comes from the fact that for computational purposes it uses XOR operations, a look-up table for substitution layer and bit reorganizations for `BlockShuffle` layer. FlexAEAD has an underlying block cipher; internal *keyed* permutation (PF_k) of 64, 128 and 256 bits. We have analyzed the PF_k function and reported several results. A brief description of PF_k has been provided in Section 2.1.2. The PF_k with x-bit state is referred to as Flex-x.

Existing Security Claims.

The designers have claimed that mounting an attack on Flex-x based on differential and linear characteristics is more difficult than the brute force attack. According to their analysis, the probability of best differential characteristic for Flex-64, Flex-128 and Flex-256 is 2^{-168}, 2^{-204} and 2^{-240} respectively. The number of chosen plaintext pairs required for a linear trail in Flex-64, Flex-128 and Flex-256 are 2^{272}, 2^{326} and 2^{380} respectively [96]. Eichlseder *et al.* have claimed several forgery attacks [103, 104] on FlexAEAD. They have followed several different approaches: like changing associated data, truncating ciphertexts and reordering ciphertexts. They have reported differential characteristics for 5-round Flex-64, 6-round Flex-128 and 7-round Flex-256 with probability 2^{-66}, 2^{-79} and 2^{-108} respectively. Length extension attacks based

Table 3.1: Comparison of trail probabilities of internal *keyed* permutation of Flex-AEAD. #R denotes the number of rounds.

Block Size	#R	Trail Probability	Technique	Reference
64	5	2^{-66}	Differential Characteristics	[103]
	5	2^{-46}	Clustered Characteristics	[103]
	5	2^{-21}	Iterated Truncated Differential	**Section 3.1**
	5	2^{-13}	Yoyo Game	**Section 4.1.3**
128	6	2^{-79}	Differential Characteristics	[103]
	6	2^{-54}	Clustered Characteristics	[103]
	6	2^{-21}	Iterated Truncated Differential	**Section 3.1**
	6	1	Yoyo Game	**Section 4.1.2**
256	7	2^{-108}	Differential Characteristics	[103]
	7	2^{-70}	Clustered Characteristics	[103]
	7	2^{-21}	Iterated Truncated Differential	**Section 3.1**
	9	2^{-11}	Yoyo Game	**Section 4.1.3**

on associated data have also been shown [161]. Table 3.1 shows the comparison of different trail probabilities reported till date with the ones furnished in the current work. For uniformity, we have enlisted trail probabilities for same number of rounds.

3.1 Iterated Truncated Differential Attacks on PF_k

Differential of iterative characteristics can be easily exploited to penetrate full rounds of a cipher. The fundamental strategy behind devising an iterated differential is to choose the output differential in a way such that after some operations the input differential can be produced easily. Alkhzaimi *et al.* have reported such differentials for SIMON family of block ciphers [14]. In this work, iterated differentials in truncated form have been considered. First of all, a particular property of `AES` s-box which has

Table 3.2: List of iterated truncated differential based key recovery attacks on PF_k reported in this work. Encs, Decs, MAs refers to encryption queries, decryption queries and Memory Accesses respectively. For uniformity, memory accesses and memory complexity has been provided in terms of Flex-128 state. 1 MA for Flex-128 corresponds to 2 MA in Flex-64 and 0.5 MA in Flex-256. Memory complexity is also normalized by the same ratio. #R denotes the number of rounds.

Block Size	#R	Data Complexity		Time Complexity	Memory Complexity	Attack Type	Section No. of Current Work
		Encs	Decs	MAs			
64	7	$2^{30.5}$	–	$2^{34.5}$	$2^{18.5}$	Iterated Truncated Differential	3.1.2
128	16	$2^{93.5}$	–	$2^{108.5}$	$2^{20.5}$	Iterated Truncated Differential	3.1.2
256	21	$2^{109.5}$	–	$2^{125.5}$	$2^{22.5}$	Iterated Truncated Differential	3.1.2

been exploited needs to be discussed.

Property of AES DDT Table.

From AES DDT table it has been observed that the number of randomly chosen input differences that map to output differences, such that the non-zero bits in each output difference are confined to the upper nibble is 4096. Same is true if they are confined to the lower nibble. In other words,

$$\left|\left\{(x_1,x_2)|\Big(S(x_1)\oplus S(x_2)\Big) \;\&\; \texttt{0xf0} = 0, \forall x_1,x_2 \in \mathbb{F}_{2^8}\right\}\right| = 4096,$$
$$\left|\left\{(x_1,x_2)|\Big(S(x_1)\oplus S(x_2)\Big) \;\&\; \texttt{0x0f} = 0, \forall x_1,x_2 \in \mathbb{F}_{2^8}\right\}\right| = 4096,$$

where S is the `AES` s-box. Therefore, with probability $\frac{4096}{2^{16}} = 2^{-4}$ a random input difference transits to upper nibble in the output difference. With same probability, random input difference transits to lower nibble. The way this property is exploited to devise iterated truncated differential is provided in the next subsection.

3.1.1 One Round Probabilistic Iterated Truncated Differential

Refer to Fig. 3-1 for the iterated differential of Flex-128. In X_1, keeping the difference in $B[0]$ ensures that in Y_1 difference are in $B[0]$ and $B[8]$. With probability 2^{-7}

both differences are confined in either upper nibble or lower nibble in those bytes. Therefore, after `BlockShuffle` only one byte is active in X_2. In X_2 the active byte can be either $B[0]$ or $B[1]$, depending on whether the upper or lower nibbles in Y_1 are active. The iterative nature of the differential comes from the fact that in X_2 only one byte is active at the cost of 2^{-7} probability under the constraints that only one byte is active in X_1, and this particular event can be repeated an infinite number of times. Similar kinds of iterated truncated differential with the same probability exists for Flex-64 and Flex-256. Now, how these one round differentials are exploited to penetrate more number of rounds is discussed.

Application to Variants of PF$_k$.

The one round iterated truncated differential can be applied to all the versions of PF$_k$. The iterated differential occurs with probability 2^{-7}. Depending on the blocksize, last few rounds can be made free as no byte to nibble transition is needed for those rounds. Let the iterated truncated differential is kept free for last f rounds for Flex-x. Then the probability of the trail is $2^{-7\times(r-f)}$. For uniform random discrete distribution, the same event will occur with probability $2^{-8\times(\frac{x}{8}-2^f)} = 2^{-(x-8\times 2^f)}$. For devising a distinguisher for x-bit flex,

$$\begin{aligned} 2^{-7\times(r-f)} &> 2^{-(x-8\times 2^f)} \\ \implies r &< \frac{(x-8\times 2^f)}{7} + f. \end{aligned} \tag{3.1}$$

Then, the probability of the iterated truncated differential trail for r-round Flex-x is $2^{-7\times(r-f)}$. Table 3.3 shows the trail probabilities for different Flex-x. r_{max} denotes the maximum number of rounds reachable under the constraints of fixed f. Table 3.4 compares the differential probabilities claim of the designers with our claim using the iterated differential. $\mathcal{P}_D$ denotes the designers' claim whereas $\mathcal{Q}_D$ denotes our claim.

Another aspect of such kind of trails is the position of active byte in each round. As mentioned in 3.1.1, if $B[0]$ is active in X_0, then either $B[0]$ or $B[1]$ is active in X_2. If $B[1]$ is active in X_2, then either $B[2]$ or $B[3]$ is active in X_3. In general, for

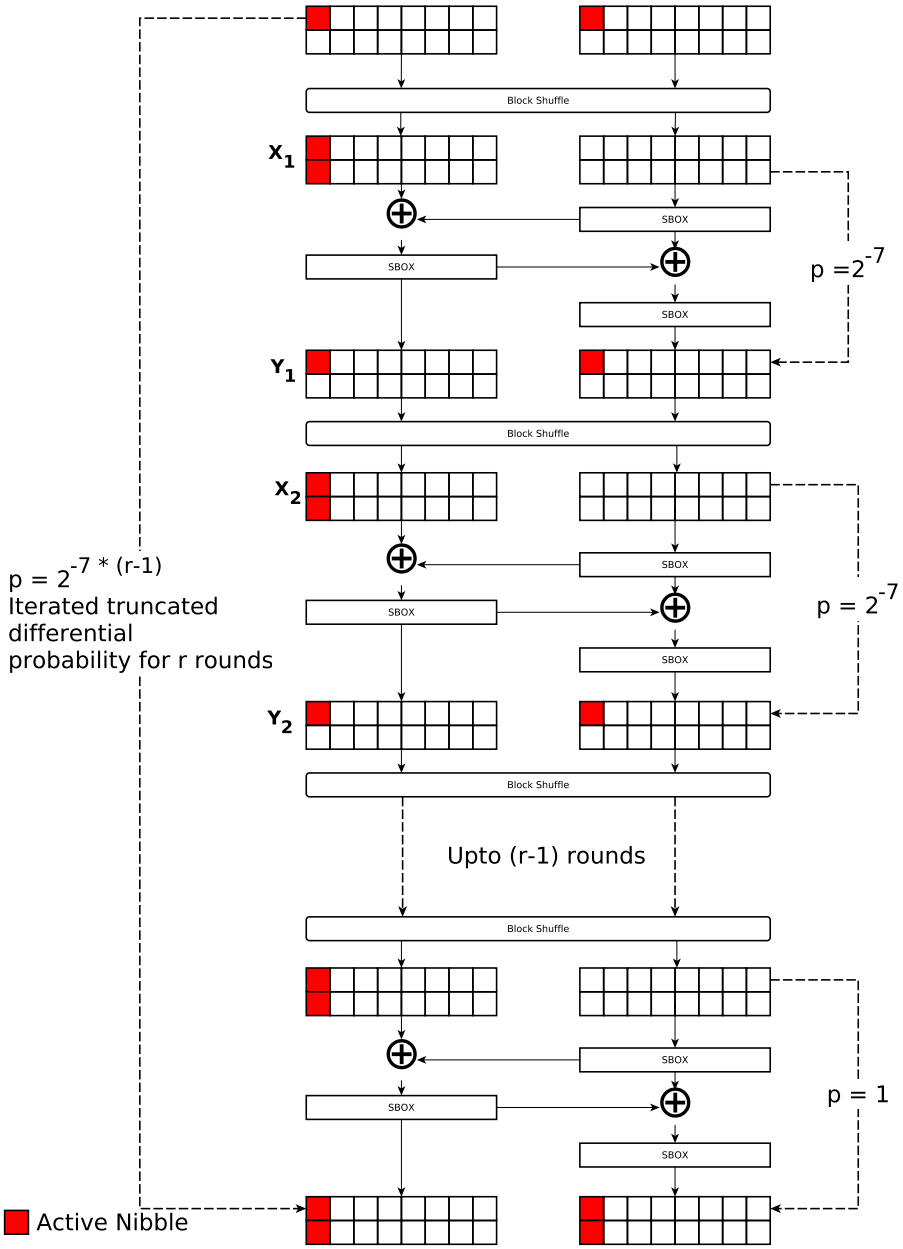


Figure 3-1: Iterated Truncated Differential with One-round probability of 2^{-7}. Note that, the key-addition is not shown, since it has no effect on the trail

Flex-x if $B[m]$ or $B[\frac{x}{2\times 8}+m]$ is active in X_i, then either $B[2m]$ or $B[2m+1]$ is active in $X_{(i+1)}$. Now, the mechanism of transforming these distinguishers to key recovery attacks is detailed.

Table 3.3: Iterated Differential Trails

Block Size	f	r_{max}	**Trail Probability**
64	1	7	2^{-42}
	2	6	2^{-28}
128	1	16	2^{-105}
	2	15	2^{-91}
	3	12	2^{-63}
256	1	21	2^{-140}
	2	21	2^{-123}
	3	21	2^{-126}
	4	21	2^{-119}

Table 3.4: Comparison of Differential Probabilities

BlockSize	**#R**	**Active S-boxes**	$\mathcal{P}^{\dagger}_{\mathbf{D}}$	$\mathcal{Q}^{*}_{\mathbf{D}}$
64	15	28	2^{-168}	2^{-98}
128	18	34	2^{-204}	2^{-119}
256	21	40	2^{-240}	2^{-119}

† Probability of the classical differential trail claimed by the designers

∗ Probability of the iterated truncated differential trail

3.1.2 Key Recovery Using Iterated Truncated Differential

At the end of each round, the difference in a pair of symmetric bytes after S-box transits to the same nibble with probability 2^{-7}. This has been used as a filtering technique to eliminate wrong key bytes. Let the first subkey, K_α for Flex-128 is being recovered. Using iterated truncated differential for r rounds a right pair can be identified with probability $2^{-7\times(r-f)}$, where f is number free rounds. Suppose, in the right pair the initial difference is in $B[i]$ and $B[i+8]$. So, we guess key byte $K_\alpha[i]$ and $K_\alpha[i+8]$. There are 2^{16} possible guesses and these are used to verify whether at the end of first-round byte to nibble transition occur. Out of 2^{16}, 2^9 key-byte candidates remain. For further filtering, two more right pairs are used. The second right pair reduces the candidate numbers to 2^2. After filtering using three different right pairs, it is expected only one candidate should remain for the key byte pair

$\left(2^{16} \times (2^{-7})^3 = 2^{-5} < 1\right)$. For the remaining symmetric key bytes, the procedure is repeated 7 more times. In the end, it is expected that only one key candidate should pass the test. The other subkeys can be recovered similarly. After recovering the first subkey, the values of the plaintexts are exactly known till the second subkey whitening. The same key recovery attacks are applicable for Flex-64 and Flex-256. In the next subsections, details about the complexities of all attacks and experimental verification of practical ones are provided.

3.1.3 Complexity Evaluation

First, we discuss about the complexity analysis of the distinguishing attacks and then analysis regarding the key recovery attacks are provided.

Distinguisher.

To distinguish iterated truncated differential for r rounds, $2^{7\times(r-f)}$ number of plaintext pairs are required, where f is the number of free rounds at the end. In devising the distinguishers, difference can be kept in 2 bytes only in X_1, which yields $\binom{2^{16}}{2} \approx 2^{31}$ pairs of plaintexts. For distinguishers requiring more than 2^{31} pairs, a different set of states is needed. So, the data complexity is $\frac{2^{7\times(r-f)}}{2^{31}} \times 2^{16} = \frac{2^{7\times(r-f)}}{2^{15}}$ encryption queries. Time complexity involves the memory accesses required to compute the specified collisions, which is the number of plaintext pairs needed, i. e., $2^{7\times(r-f)}$. Memory complexity is 2^{16} Flex-x states, which is the memory required for storing different states.

Consider a particular case for 21-round Flex-256. According to Inequality 3.1, the value of f can be set to 4. For this case

1. Data Complexity is $\frac{2^{7\times17}}{2^{15}} = 2^{104}$ encryption queries..

2. Time Complexity is 2^{119} memory accesses.

3. Memory Complexity is 2^{16} Flex-256 states = 2^{17} Flex-128 states.

Key Recovery.

Complexities of key recovery attack of Flex-x depends on distinguisher. To recover each pair of key-byte, three different right pairs are required. This procedure also needs to be repeated $\frac{x}{16}$ times for recovering the full key. Therefore, data complexity, time complexity and memory complexity of distinguisher needs to be multiplied by a factor of $3 \times \frac{x}{16}$. Moreover, candidate key-byte recovery for each pair of byte can be computed in parallel. To recover the other subkey, a plaintext, ciphertext pair $\left(p_1, c_1\right)$ is chosen and PF_k round functions till the second subkey whitening is computed offline and XOR-ed with c_1. So, the complexities of r-round Flex-x with f free rounds are-

1. Data Complexity is $3 \times \frac{x}{16} \times \frac{2^{7\times(r-f)}}{2^{15}}$ encryption queries.
2. Time Complexity is $3 \times \frac{x}{16} \times 2^{7\times(r-f)}$ memory accesses.
3. Memory Complexity is $3 \times \frac{x}{16} \times 2^{16}$ Flex-x states.

The complexities of particular cases for 7-round Flex-64 with f=1, 16-round Flex-128 with f=1 and 21-round Flex-256 with f=4 have been listed in Table 3.2.

3.1.4 Experimental Verification

The key recovery attack using iterated differentials has been experimentally verified for 8 rounds Flex-128 with f=3. The attack initiates after a key is chosen randomly. The number of key candidates after using the first right pairs for each pair of symmetric bytes (from (K_α[0],K_α[8]) to (K_α[7],K_α[15])) are 316, 520, 632, 448, 568, 484, 368 and 356 respectively. It conforms to the theoretical analysis, which states that the number of candidates should be around 2^9. After using the second right pairs, the number of candidates is reduced to 2, 12, 4, 4, 6, 5, 2 and 5 respectively which is close to the theoretical value of 2^2. The third right pair reduces the number for all pairs of bytes to 1. The key recovery attack correctly recovers the subkeys.

In the following section, we show how to mount forgery attacks on FlexAEAD variants using the idea of iterated truncated differentials.

3.2 Forgery Attacks on FlexAEAD

Eichlseder *et al*. have shown forgery attacks on FlexAEAD by applying several strategies [103]. All those strategies are also applicable using the differentials described in this chapter. The main difference between these two approaches is the differential characteristics for the sequence generation. First, the differential characteristic of the sequence generation step is shown.

3.2.1 Differential Characteristics in Sequence Generation

A sequence of bits is used by FlexAEAD for authenticated encryption. These sequences are generated by using PF_k, with initial state being the nonce. For details on sequence generation refer to [96]. The difference between two consecutive sequence numbers is that their last call to PF_k differ by a `INC32` call. `INC32` is a 32-bit word operation which acts as an XOR operation with probability 2^{-1}.

Consider, m 32-bit words in a r-round Flex-x state. Due to `INC32` with probability 2^{-m}, m nibbles at $\frac{m}{2}$ symmetric positions become active between two subsequent sequence generation steps. Due to `BlockShuffle`, those m active nibbles is converted to $\frac{m}{2}$ active bytes which occupies $\frac{m}{4}$ symmetric positions. In the next round, those active bytes transits to $\frac{m}{8}$ symmetric positions ($\frac{m}{4}$ active bytes) at the cost of 2^{-2m}. In the next round, $\frac{m}{16}$ symmetric positions get occupied at the cost of 2^{-m}. After repeating the process, $\left(\log_2(m) - 2\right)$ times, only one symmetric position remains occupied by the active byte. For the rest $\left(r - \log_2(m) + 2\right)$ rounds, with probability 2^{-8} for each round the position of two active nibbles in the output get fixed (Note that, in the iterated truncated differential, the position of active is not fixed and that is why the probability of 2^{-7} is paid). With 2^{-8} probability the value of the active nibbles can be fixed to a specific value.

By following this approach, the difference of two consecutive sequence numbers can be fixed to a specific value with probability 2^{-50} for FlexAEAD-64, 2^{-60} for FlexAEAD-128 and 2^{-80} for FlexAEAD-256 (Corresponding complexities of forgery attacks are computed by taking the inverse of these probabilities). Differential char-

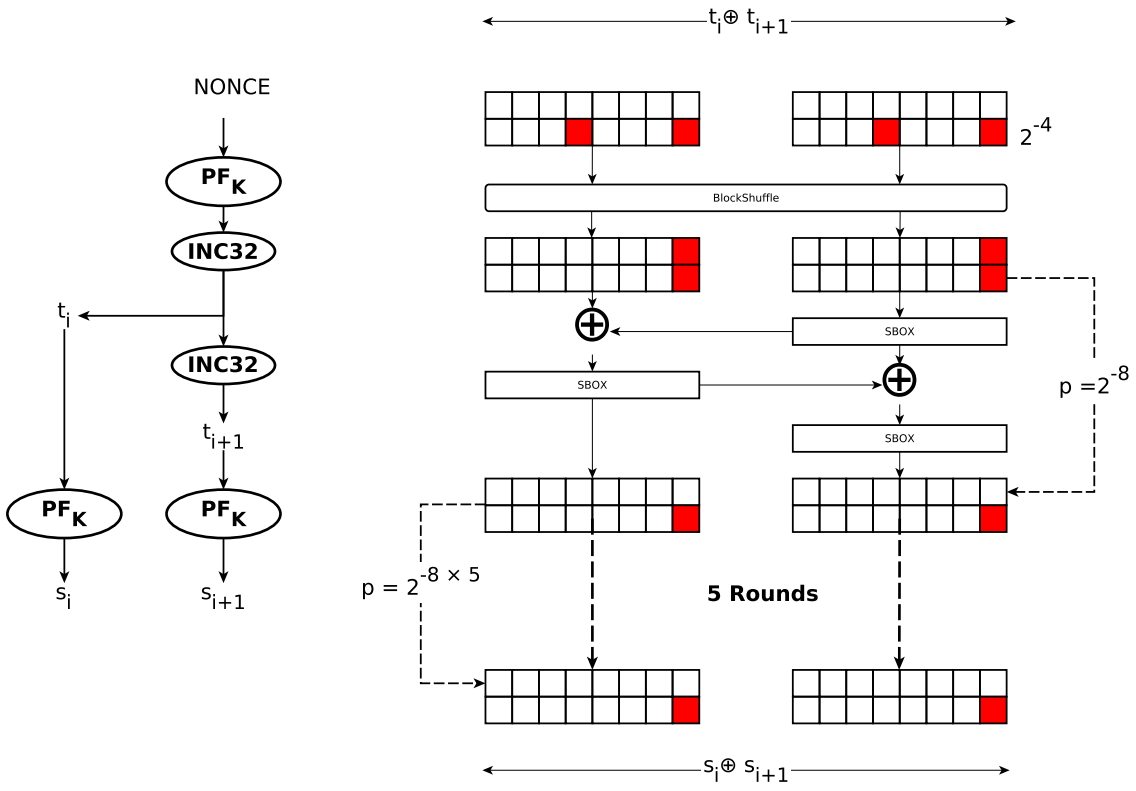


Figure 3-2: Differential Characteristics of Sequence Generation for FlexAEAD-128. Note that, plaintext difference or associated data difference can cancel out difference in $S_i \oplus S_{i+1}$ with probability 2^{-8}.

Table 3.5: Comparison of Forgery Attacks on FlexAEAD

Scheme	Complexity	Technique	Reference
FlexAEAD-64	2^{50}	*Changing Associated Data/ Truncating Ciphertext/ Reordering Ciphertext*	Current Work
	2^{46}		[103]
FlexAEAD-128	2^{60}		Current Work
	2^{54}		[103]
FlexAEAD-256	2^{80}		Current Work
	2^{70}		[103]

acteristics of sequence generation for FlexAEAD-128 is shown in Fig 3-2. Once the output difference value is fixed, the techniques (*Changing Associated Data, Truncating Ciphertext, Reordering Ciphertext*) in [103] can be applied to forge ciphertext-tag pair. Comparison between several approaches regarding forgery attack is enlisted in Table 3.5.

3.3 Chapter Summary

In this work, we analyzed all variants of PF_k of FlexAEAD. We reported a one round differential characteristic of PF_k, which due to its iterative nature was exploited to penetrate a large number of rounds. All the attacks were easily exploited to recover the subkeys. The iterated truncated differential attack strategy was applied to the nonce-based sequence number generator which was exploited to devise similar kinds of forgery attacks on FlexAEAD as given by Eichlseder *et al.* [103]. The success probabilities of all distinguishing attacks were shown to be high. All attacks reported in this work with practical complexities were experimentally verified. All these attacks have exploited a vulnerability in the design which is based on dividing the nibbles into two parts while using `AES` s-box.

4

YOYO ATTACKS ON INTERNAL KEYED PERMUTATION OF FlexAEAD

Contents

In this chapter, we explore the application of the yoyo property which has been introduced by Rønjom *et al.* [176] on generic 2-round Substitution Permutation Networks and further extended on AES-based permutations and block ciphers [180, 27]. We have been able to devise deterministic yoyo distinguishers for 4, 6 and 8 rounds of Flex-64, Flex-128 and Flex-256 respectively which are further extended by one more round to mount key recovery attacks. All key recovery attacks (reported in this work) with their respective complexities are summarized in Table 4.1. The attacks with practical complexities are experimentally verified.

4.1 Yoyo Attacks on $\mathbf{PF}_k$

The yoyo distinguishing attack has been briefly described in Section 2.5.2. First, the result of yoyo game on 2-generic SP rounds has been applied for devising r-round Flex-x deterministic distinguisher. Then cipher specific properties has been exploited

Table 4.1: List of key recovery attacks on PF_k using the yoyo game reported in this work. Encs, Decs, MAs refers to encryption queries, decryption queries and Memory Accesses respectively. For uniformity, memory accesses and memory complexity has been provided in terms of Flex-128 state. 1 MA for Flex-128 corresponds to 2 MA in Flex-64 and 0.5 MA in Flex-256. Memory complexity is also normalized by the same ratio. #R denotes the number of rounds.

Block Size	**#R**	**Data Complexity**		**Time Complexity**	**Memory Complexity**	**Attack Type**	**Section No. of Current Work**
		Encs	**Decs**	**MAs**			
64	5	2^{10}	$2^{16.5}$	$2^{15.5}$	2^{10}	Yoyo Attack	4.1.3
128	7	$2^{10.5}$	$2^{16.5}$	$2^{16.5}$	$2^{11.5}$	Yoyo Attack	4.1.3
256	9	2^{11}	$2^{16.5}$	$2^{17.5}$	2^{13}	Yoyo Attack	4.1.3

to penetrate one more extra round and recover the key. Here, r is 4, 6 and 8 for Flex-64, Flex-128 and Flex-256 respectively. First, details about Super-Sbox of Flex-x is given.

4.1.1 Super-Sbox of PF_k

Refer to Fig. 4-1 for the Super-Sbox construction in Flex-128 block cipher. Consider the bytes $\{B[0], B[2], \cdots B[7]\}$ at X_1. Due to round function, only the symmetric bytes affect each other. So, in Y_1 every symmetric bytes depends on every symmetric bytes at X_1. Due to BS^2, $B[2i]$, $B[2i+8]$ $(0 \leq i \leq 3)$ from Y_1 constitutes the $B[4i]$, $B[4i+1]$ $(0 \leq i \leq 3)$ at X_2. Due to application of BS^3, $\{B[2i], B[2i+1], B[2i+8], B[2i{+}9]\}$, $(0 \leq i \leq 1)$ at Y_2 affects $\{B[8i], B[8i{+}1], B[8i{+}2], B[8i{+}3]\}$, $(0 \leq i \leq 1)$ at X_3. This constitutes a Super-Sbox which spans over 2.5 rounds (omitting the initial `BlockShuffle`). There are two 64-bit Super-Sbox in the Flex-128 state. In similar way, Flex-64 and Flex-256 has 32-bit and 128-bit Super-Sbox which span over 1.5 and 3.5 rounds respectively. In the next subsection, how these Super-Sboxes are used to design deterministic yoyo distinguishers is discussed.

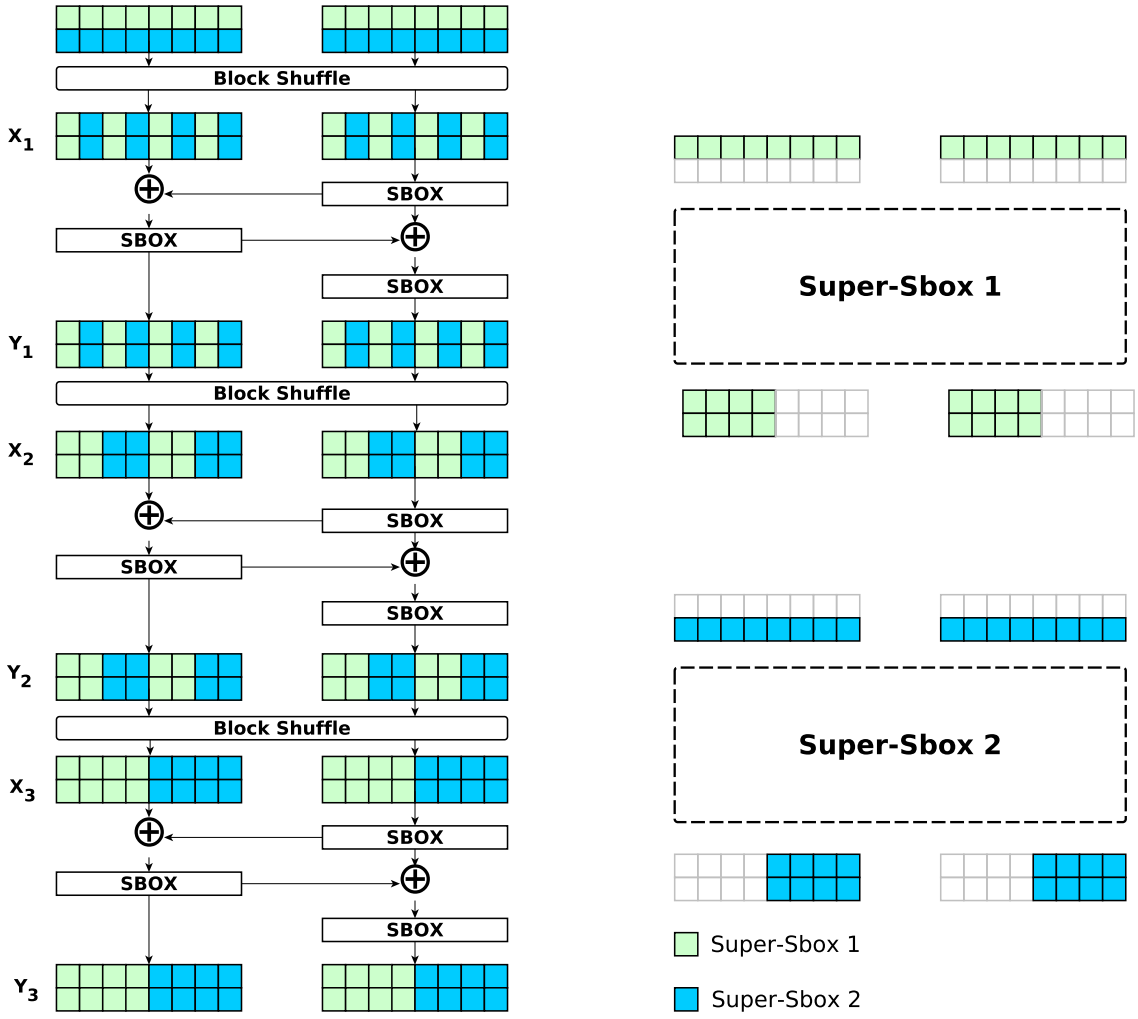


Figure 4-1: Super-Sbox of Flex-128 Block Cipher

4.1.2 Deterministic Distinguisher for r-round Flex-x

In devising this distinguisher, Theorem 1 has been used directly. For this purpose, the $S \circ L \circ S$ layers need to be identified in this construction. The S here corresponds to Super-Sbox described in Section 4.1.1 whereas the L corresponds to the `BlockShuffle` layer. A pair of plaintexts is chosen such that only one of the Super-Sbox is active at X_1. yoyo game is played using these two plaintexts to obtain a new pair of texts. The same Super-Sbox should be active in the new pair of texts and the other should be inactive. For a uniform random discrete distribution, this occurs with probability $\frac{1}{2^{\frac{x}{2}}}$. Next, attack procedures and their corresponding complexities are provided. In the attack procedure, steps pertaining to Flex-128 has been

described. Same attack strategy follows for Flex-64 and Flex-256.

Attack Procedure.

1. Choose two 128-bit plaintexts p_1, p_2 such that, $wt(\nu(p_1 \oplus p_2)) = 1$. Inverse `BlockShuffle` is applied to p_1, p_2 and then they are queried to encryption oracle to obtain c_1, c_2.

2. As there is two Super-Sboxes, so only one swapping is possible. One of the Super-Sbox is swapped between c_1 and c_2 to form c_1', c_2', which are queried to decryption oracle and p_1', p_2' is obtained.

3. Check whether $wt(\nu(BS(p_1') \oplus BS(p_2'))) = 1$ or not. If it is 1, then distinguish it as Flex-128; otherwise it is a random permutation.

Complexity Evaluation.

The attack needs 2 encryption queries and 2 decryption queries; its time complexity is 2 `BlockShuffle`, 2 inverse `BlockShuffle` operation and 2 Flex-128 state XOR, and the memory complexity is negligible.

4.1.3 Key Recovery for $(r+1)$-round Flex-x

For attacking $(r+1)$-round Flex-x, yoyo distinguishing attack on r-round is composed with the one round trail of iterated truncated differential. The attack for Flex-128 is shown in Fig. 4-2. With probability 2^{-7} only one Super-Sbox is active at X_2. By virtue of yoyo game, only one Super-Sbox should be active in W_2. Due to inverse `BlockShuffle`, the differences should be confined to either upper nibbles or lower nibbles in Z_1; the other half should be free. With probability 2^{-8}, two symmetric bytes become free at Z_1. There are 8 (4 and 16 for Flex-64 and Flex-256 respectively) choices for symmetric byte positions which increases the probability to 2^{-5} $\left(2^{-6}\right.$ and $\left.2^{-4}\right.$ for Flex-64 and Flex-256$\left.\right)$. Therefore, at the cost of 2^{-12}, two symmetric bytes

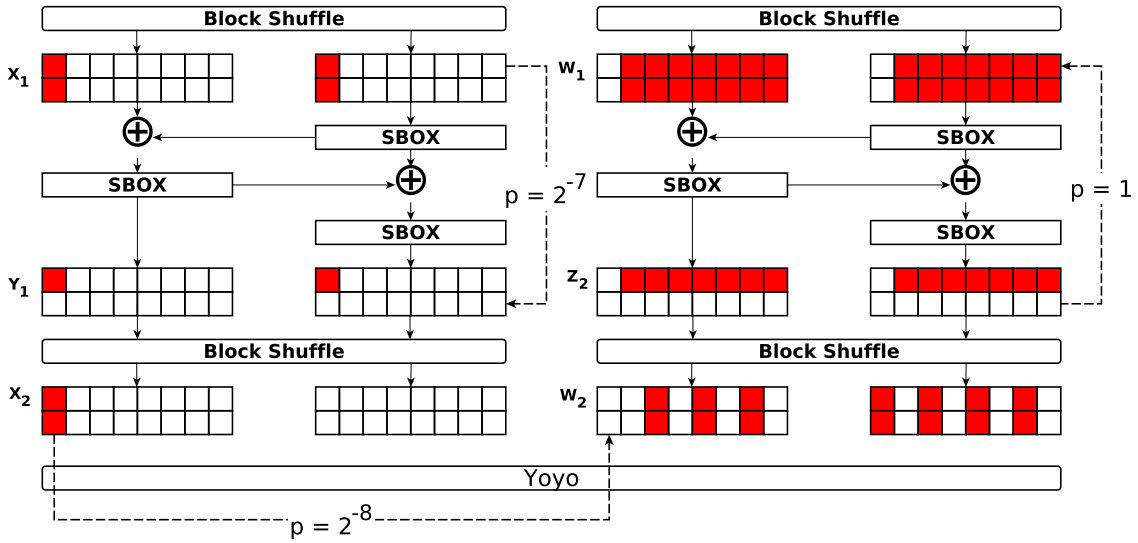


Figure 4-2: 7-round Yoyo Distinguisher for Flex-128

become free for the 7-round Flex-128. Probability of the same event for 5-round Flex-64 and 9-round Flex-256 is 2^{-13} and 2^{-11} respectively. Now, the attack steps of Flex-128, it's corresponding complexities and experimental verifications are discussed.

Attack Procedure.

1. Choose 2^6 plaintexts such that they differ only in $B[0]$ and $B[8]$. Apply inverse `BlockShuffle` on them and query them to encryption oracle to obtain corresponding ciphertexts. Consider all ciphertext pairs, swap bytes between them according to the Super-Sbox output and query them to the decryption oracle to obtain new pairs of plaintexts. Check whether the pair has a pair of free symmetric bytes. At least one such pair is expected.

2. Repeat step 1 two more times to obtain two more right pairs. Let (c_1, c_2), (c_3, c_4) and (c_5, c_6) be such pairs and their corresponding plaintexts are (p_1, p_2), (p_3, p_4) and (p_5, p_6). After byte swapping, (c_1, c_2), (c_3, c_4) and (c_5, c_6) becomes (c'_1, c'_2), (c'_3, c'_4) and (c'_5, c'_6). `BlockShuffle` is applied on the decrypted value of these modified ciphertexts to obtain (p'_1, p'_2), (p'_3, p'_4) and (p'_5, p'_6).

3. Guess key bytes 0 and 8 for K_α, run one round encryption for p'_1, p'_2 and observe

whether same nibble in $B[0]$ and $B[8]$ remains free or not for the pair. Using nibble transition, out of 2^{16} candidates, 2^7 are filtered out. Then the remaining two right pairs subsequently reduces the number of candidates for $K_\alpha[0]$ and $K_\alpha[8]$ to 2^2 and 1 respectively.

4. For the remaining 7 symmetric pairs of bytes, step 3 is repeated 7 more times. At, the end 1 key candidates are expected for K_α. For each K_α, K_β is computed by using a plaintext-ciphertext pair. If there is more than one K_α, K_β pair, they are exhaustively tried for finding the right key candidate.

Complexity Evaluation.

Let probability of the event that "two symmetric bytes become free" is 2^{-p}. So, for retrieving a right pair, $2^{\frac{p}{2}}$ encryption queries and 2^p decryption queries are required. For guessing each pair of key byte, 3 such right pairs are needed and to recover the key, this process need to be repeated $\frac{x}{16}$ times. Therefore, data complexity of the attack is $\frac{3\times x}{16} \times 2^{\frac{p}{2}}$ encryption queries and $\frac{3\times x}{16} \times 2^p$ decryption queries.
Time complexity is $\frac{3\times x}{16} \times 2^p$ memory accesses for retrieving the stored ciphertexts.
Memory complexity is $\frac{3\times x}{16} \times 2^{\frac{p}{2}+1}$ Flex-x states for storing the plaintexts and ciphertexts.
The complexities of 7-round Flex-128 key recovery attack are-

1. Data Complexity is $24 \times 2^6 \approx 2^{10.5}$ encryption queries and $24 \times 2^{12} \approx 2^{16.5}$ decryption queries.

2. Time Complexity is $2^{16.5}$ memory accesses.

3. Memory Complexity is $2^{11.5}$ Flex-128 states.

Experimental Verification.

The yoyo attack for 7-round Flex-128 has been experimentally verified. Initially the oracle chooses a master key randomly and computes the subkeys. Adversarial algorithm queries according to attack steps in Section 4.1.3 and retrieves right pairs. The

Table 4.2: Success probabilities of various distinguishers. #R denotes the number of rounds.

Distinguisher Type	**Block Size**	f	**#R**	$p \times (1+q)$	p	**Success Probability**
Iterated	64	1	7	2^{-42}	2^{-48}	0.8
	128	1	16	2^{-105}	2^{-112}	0.82
	256	4	21	2^{-119}	2^{-192}	0.84
Yoyo	64	n/a	5	2^{-13}	2^{-14}	0.61
	128	n/a	7	2^{-12}	2^{-13}	0.61
	256	n/a	9	2^{-11}	2^{-12}	0.61

number of key candidates corresponding to each symmetric bytes $\big($from $(K_\alpha[0],K_\alpha[8])$ to $(K_\alpha[7],K_\alpha[15])$ $\big)$ after filtering with first right pairs are 502, 618, 546, 496, 510, 486, 552 and 538 respectively which conforms to the theoretical value of 2^9. The second right pairs further reduces it to 6, 7 6, 7, 7, 3, 3 and 5 respectively which is close to the theoretical value of 2^2. The third pairs reduces all these values to 1. This reduction in the number of key candidates using the right pairs conforms to the theoretical analysis. At last, the algorithm successfully recovers the subkeys.

In the next section, we discuss the success probability of distinguishing attacks reported in this work.

4.2 Success Probability of Distinguishing Attacks

The effectiveness of an attack depends on its success probability. First, the success probability of all reported distinguishers is computed. Then, the success probability of practical ones is experimentally verified. To deduce the theoretical estimation of success probabilities, the following theorem from [174] has been applied.

Table 4.2 lists the success probabilities of different distinguishers presented in this chapter.

Table 4.3: Experimental Verification of Success Probability

Distinguisher	#rounds	f	$\#n$	Blackbox	Detected as $\mathcal{C}$	Detected as $\mathcal{R}$	Experimental Success Probability	Estimated Success Probability
Flex-64	5	2	100	Flex-64	65	35	0.8	0.83
				$\mathcal{R}$	5	95		
Flex-64	6	2	100	Flex-64	79	21	0.76	0.77
				$\mathcal{R}$	27	73		

Table 4.4: Comparison of Success Rate for Flex-64

f	#rounds	$p \times (1+q)$	p	Success Probability
1	6	2^{-35}	2^{-48}	0.83
2	6	2^{-28}	2^{-32}	0.77

Experimental Verification.

The values of success probabilities for 5-round and 6-round Flex-64 derived using experiments and theoretical estimations are listed in Table 4.3.

Trade-off between Success Rate and Free Rounds.

The iterated truncated differentials can have a different number of free rounds at the end. More number of free rounds reduces the trail complexity at the expense of success rate. For analysis, consider the case pertaining to 6-round Flex-64 with the number of free rounds 1 and 2. The success rate for both cases is listed in Table 4.4.

For 21-round Flex-128, the number of free rounds can take any value between 1 and 4. For each of the cases, the theoretical estimation of success probability is almost equal. The estimated success probabilities have been shown in Table 4.5. The difference between the distribution of random bitstream and 21-round Flex-128 for each case is so huge, that it has a negligible effect on the success probability.

4.3 Chapter Summary

Here, it is shown that the generalized yoyo distinguishing attack on SPN ciphers is applicable for PF_k. While deploying yoyo attack, a Super-Sbox construction of 1.5,

Table 4.5: Comparison of Success Rate for Flex-256

f	$p \times (1+q)$	p	**Success Probability**
1	2^{-140}	2^{-240}	0.84
2	2^{-133}	2^{-224}	0.84
3	2^{-126}	2^{-208}	0.84
4	2^{-119}	2^{-192}	0.84

2.5 and 3.5 rounds in 64-bit, 128-bit and 256-bit PF_k respectively were reported. All these attacks were easily exploited to recover the subkeys. All the attacks reported in this work with practical complexities were experimentally verified.

5 YOYO ATTACKS ON AES-BASED DESIGNS

Contents

Non-random behavior of a cryptographic construction has been historically seen as a sign of an inherent weakness waiting to be exploited. In this regard, devising distinguishers forms one of the fundamental aims of a cryptanalyst since they help exhibit non-randomness. A distinguisher generally constitutes a statistical or structural property of a crypto-primitive that is not expected to occur for an *equivalent* random function. The scope of distinguishers is further amplified by the probability of their possible conversion/extension to more stronger forms of attacks like key-recovery for block ciphers or collisions for hash functions. The SHA3 competition [170] witnessed the Zero-Sum distinguisher (introduced by Aumasson and Meier [22]) which was one of the most studied attacks against the internal public permutation

Keccak$-f$ of SHA3 winner Keccak. On the other hand, a multitude of distinguishing attacks have been reported on AES [87] both in the secret-key as well as known-key setting (introduced by Knudsen and Rijmen [142]). The known-key paradigm is of particular interest since it enables studying a cipher as a public permutation. Moreover, as argued by Knudsen and Rijmen, non-existence of a known-key distinguisher implies non-existence of a secret-key one, making it imperative to study the former. This work aims to explore distinguishing attacks on public permutations based on AES with the motivation that results reported here might lead to stronger attacks on constructions where these permutations are deployed as an internal transformation.

The current work intends to look at the yoyo technique as a general cryptanalysis strategy specially in the light of public permutations. In particular, we look at AESQ, the AES-based internal permutation of CAESAR [1] round 2 candidate PAEQ [9]. PAEQ, along with AESQ permutation was introduced by Biryukov and Khovratovich in ISC 2014 [56]. There are many variants of PAEQ but across all variants, the same permutation AESQ of width 512 bits is used. The designers themselves have done a lot of cryptanalysis on PAEQ and have shown a Constrained Input Constrained Output (CICO) attack with complexity 2^{32} on 8-round AESQ. They have also proposed a 12-round distinguisher with complexity 2^{256}. Bagheri et al. have reduced the complexity of the 12-round disgtinguisher to 2^{128}[25]. They have extended their work for 16-round AESQ permutation and shown a distinguisher with complexity 2^{192} using 2^{128} memory. A key recovery attack has also been devised on PAEQ targeting the diffusion of the AESQ permutation by Saha et al. [179]. They have proposed a 8-round key recovery with complexity of 2^{48}.

This work reports a family of distinguishers on AESQ which primarily capitalize on the yoyo game. This is the first time that yoyo based distinguishers have been devised for a public permutation. The basic yoyo idea is augmented with other cryptanalytic principles to penetrate a higher number of rounds. In doing so, the first practical 9-round distinguisher that works from the first round is achieved. The inside-out technique is leveraged up on to reach up to 10 rounds with practical complexities and extended to 12 rounds with 2^{126} queries. Further, we introduce the idea of

bi-directional yoyo game where two yoyo games are played in opposite directions and connected using the properties of the linear layer of AESQ. This leads to the development of a 16-round distinguisher with a complexity of 2^{126}. We summarize our results in comparison to the previous works in Table 5.1. As can be seen, the current work outperforms all previous results by a huge margin while requiring negligible memory. Finally, to emphasize the scope of the devised techniques we apply them to AES which under the known-key setting also behaves like a public permutation. Applying bi-directional yoyo on AES helps devise one of the best 8-round distinguishers with complexity of 2^{30} and negligible memory requirement, as shown in Table 5.2.

The rest of the work is organized as follows. In Section 5.1.1, a deterministic distinguisher for 8-round AESQ is presented. This deterministic distinguisher is extended and a 9-round probabilistic distinguisher is illustrated in Section 5.1.2. In Section 5.2, a brief overview of improbable differential and inside-out technique and their application to AESQ is given. In Section 5.3, impossible yoyo distinguishers for 12 and 16-round AESQ are demonstrated. In Section 5.4, impossible differential yoyo and impossible differential bi-directional yoyo techniques is applied to round-reduced AES in known-key setting. Experimental setup and results are briefly mentioned in Section 5.5. Arguments in favour of the validity of the work are discussed in Section 5.6. The chapter is summarized in Section 5.8.

5.1 Distinguishers using Direct Yoyo on AESQ

In order to adapt the yoyo trick on AESQ, we need to first identify the $S \circ L \circ S$ construction embedded in the permutation. To do that one has to recall the notion of MegaSBox whereby 3.5 rounds[1] of AESQ starting from an even round can be depicted as independent computations of four 128-bit words (Refer to Fig. 2-8). These four MegaSBoxes constitute the first S layer of the generic SPN. The subsequent MegaMixColumns corresponds to L layer while the next iteration of four MegaSBoxes represent the last S layer thereby completing the $S \circ L \circ S$ sequence. Fig. 5-1

[1]Without the MegaMixColumns of the last round

Table 5.1: Distinguishers on Round-Reduced AESQ

Rnd	Complexity			Technique	Reference
	Time	Memory	Success Prob.		
8	2^{32}			CICO	Designers [9]
8†	1	Negligible		**Yoyo**	**Section 5.1.1**
9	$2^{26.08}$	Negligible	0.71		**Section 5.1.2**
9†	5	Negligible	0.82	**Improbable Differential Yoyo**	**Section 5.2.2**
10†	2^{28}	Negligible	0.77		**Section 5.2.2**
12†	2^{126}	Negligible	0.84	**Impossible Differential Yoyo**	**Section 5.3.1**
	2^{256}	2^{256}	0.61	Rebound Attack	Designers [9]
	2^{128}	Negligible	0.83		Bagheri et al. [25]
	$2^{102.4}$	$2^{102.4}$	0.83	Time-memory Trade-off	
	$2^{128-x/4}$	2^{x}			
16†	2^{192}	2^{128}	0.83	Rebound Attack	
	2^{188}	2^{128}	0.83	Multi Ltd.-Birthday Distinguisher	
	2^{192+x}	2^{128-x}		Time-memory Trade-off	
	2^{126}	Negligible	0.84	**Impossible Differential Bi-directional Yoyo**	**Section 5.3.2**

† Starting from round 2

shows this construction starting from Round-2. So two generic SP-rounds map to 8 rounds of AESQ without the last MMC. So, the yoyo distinguisher pertaining to two generic SP-rounds as discussed above directly applies to $\texttt{AESQ}_{2\to 9}$. In the next subsection we work out the details of this distinguisher which is the first *deterministic* 8-round distinguisher for the AESQ permutation.

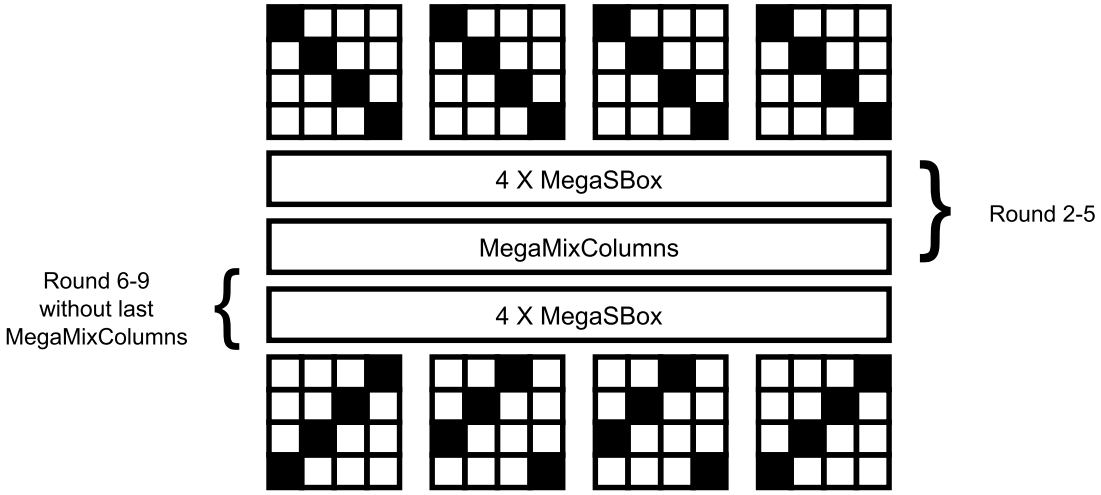


Figure 5-1: $\texttt{AESQ}_{2\to 9}$ as an $S \circ L \circ S$ construction.

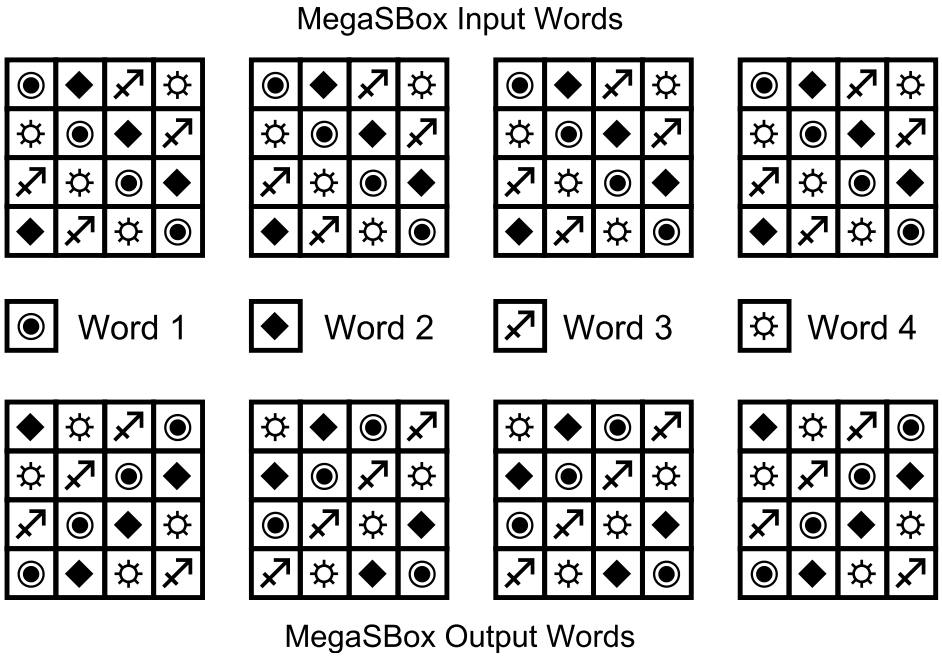


Figure 5-2: Word configuration for each MegaSBox

5.1.1 Distinguisher for 8 Rounds

Let us first look at the yoyo game for AESQ which we will call as a subroutine for the distinguishing algorithm. The yoyo game shown in Algorithm 4 is tailored w.r.t $\texttt{AESQ}_{i\to j}$ but will be analogous for any corresponding random permutation. The procedure is self-explanatory except for two things:

- The function MSwap is used to swap words between two states of AESQ. Apart from the states it accepts an argument DIRECTION which decides whether input or output words (Ref. Fig. 5-2) will be swapped. So, if DIRECTION = FORWARD, then output words will be swapped while for DIRECTION = BACKWARD, it will be done in accordance with input word pattern. This distinction takes into account the direction in which the game is being played. As will be seen later, we will need to play the game in the backward direction to penetrate a higher number of rounds. Moreover, MSwap can, at random, swap any one, two or three words of the states. As shown by the authors of [176], all such word-swap configurations are equivalent and preserve the properties of the yoyo game.

- The argument Mode is used to play either half or full of the yoyo game and respectively receives values MID or FULL. Later, in this work we will show how output of half of the game can be used to generate input states that help to distinguish up to 16 rounds of AESQ.

Algorithm 4 Yoyo Game for AESQ

procedure YOYO$(p_1, p_2, \texttt{AESQ}_{i\to j}, \texttt{Mode}, \texttt{DIRECTION})$
  $\{c_1 \leftarrow \texttt{AESQ}_{i\to j}(p_1); \quad c_2 \leftarrow \texttt{AESQ}_{i\to j}(p_2)$
  $(c_1', c_2') \leftarrow \texttt{MSWAP}(c_1, c_2, \texttt{DIRECTION})$ ▷ DIRECTION $\in$ {FORWARD, BACKWARD}
  if Mode = MID **then**
    return (c_1', c_2')
  else if Mode = FULL **then**
    $\{p_1' \leftarrow \texttt{AESQ}_{i\to j}^{-1}(c_1'); \quad p_2' \leftarrow \texttt{AESQ}_{i\to j}^{-1}(c_2')$
  end if
  return (p_1', p_2')
end procedure

With the yoyo game in place, the 8-round distinguisher that uses it, is straightforward as shown in Algorithm 5. The distinguisher accepts a permutation PERMUTE. It chooses inputs p_1 and p_2 such that $\alpha = p_1 \oplus p_2$ has a particular ZDP (of weight at least one and at most three), say $\nu(\alpha) = (1, 0, 1, 0)$. Then it plays the yoyo game generating two new inputs p_1' and p_2' with $\Delta = p_1' \oplus p_2'$. If PERMUTE = $\texttt{AESQ}_{2\to 9}$, it is ensured that ZDP of α is same as that of Δ.

Algorithm 5 Distinguisher for $\texttt{AESQ}_{2\to 9}$

Output: 1 for AESQ, -1 otherwise ▷ 8-Round AESQ without last MMC

1: **procedure** DistYoyo(PERMUTE)
2: $p_1, p_2 \leftarrow \left\{(m_1, m_2) : 3 \geq wt(\nu(m_1 \oplus m_2)) > 0\right\}$ ▷ At least one word active
3: $p_1', p_2' \leftarrow$ Yoyo$(p_1, p_2,$ PERMUTE, FULL, FORWARD$)$
4: **if** $\nu(p_1' \oplus p_2') = \nu(p_1 \oplus p_2)$ **then** ▷ Holds for $\texttt{AESQ}_{2\to 9}$ deterministically
5: **return** 1
6: **else**
7: **return** -1
8: **end if**
9: **end procedure**

The pictorial description is captured by Fig. 5-3. It is intentionally shown that the Nested ZDP might differ which will definitely happen probabilistically. This is because the yoyo principle guarantees that the ZDP will be preserved but has no claim on the activity pattern inside the active words. In the next subsection we will show how assuming a particular Nested ZDP enables us to extend the distinguisher to include the first round making it the first 9-round AESQ result that starts from round one.

5.1.2 Extension to 9-round AESQ

The inclusion of the first round relies on the notion of Nested ZDP that we introduced earlier. The basic idea is to:

- First leverage on the determinism of the 8-round yoyo while imposing some restriction on both the input and output Nested ZDP, thereby making it probabilistic. If the input and output differences of the yoyo are α and η respectively,

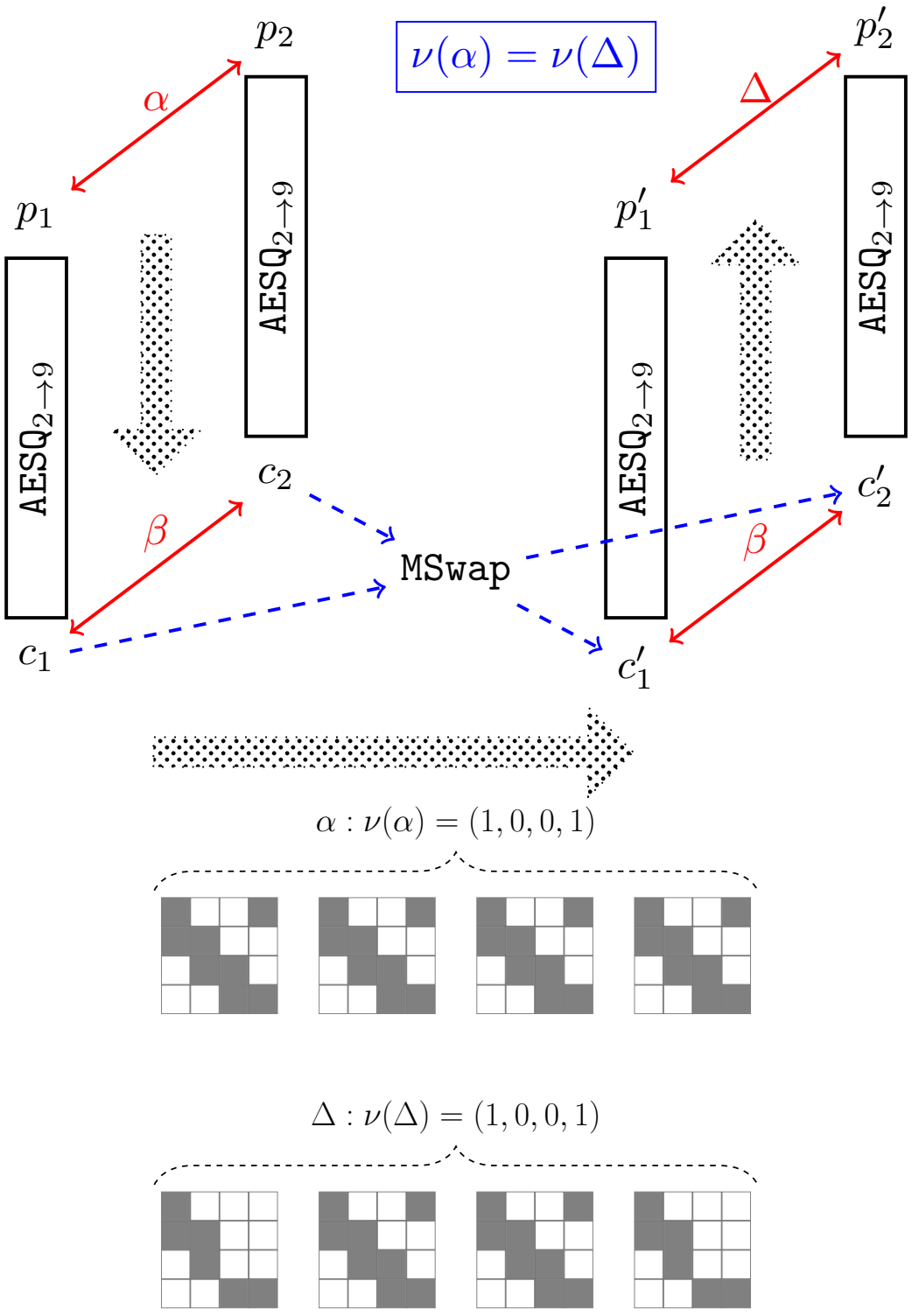


Figure 5-3: Deterministic 8-round yoyo distinguisher. ZDP of input difference α is preserved by output difference Δ of the yoyo game. A possible configuration of (α, Δ) is shown. Note that though $\nu(\alpha) = \nu(\Delta), wt(\nu^2(\alpha)) \neq wt(\nu^2(\Delta))$. We will use this kind of Nested ZDP to extend the yoyo beyond 8 rounds.

then the restrictions are of the form:

1. Input: Exactly one byte active in one word.

$$wt(\nu(\alpha)) = 3 \text{ and } \exists i : \ wt(\nu^2(\alpha_i)) = 15.$$

2. Output: Exactly one byte *inactive* in one word.

$$wt(\nu(\eta)) \overset{\texttt{Yoyo}}{=} 3 \text{ and } \exists i : \ wt(\nu^2(\eta_i)) = 1.$$

Now, the probability that α leads to η is $\left[16 \times 2^{-8} \times (\frac{255}{256})^{15}\right] \approx 2^{-4.08}$.

- The second step is to find a one-round differential to connect with the input difference α. This is standard[2] and we can, with a probability 2^{-22}, find an input difference that conforms to the input restriction for the yoyo game. This leads to the inclusion of round one in the forward direction.

- The last step is to include the round in the backward direction. One can easily note that if the output restriction is satisfied then, the extra round in the return path will automatically lead to a state that has four inactive bytes. More precisely, if the last difference is denoted by Δ then we have $\exists i : \ wt(\nu^2(\Delta_i)) = 4$. Also, the inactive bytes will belong to same diagonal due to the last inverse ShiftRows operation of Round 1.

Fig. 5-4 gives an overview of the entire extension strategy and also depicts a particular configuration of states that conform to the above statements while Algorithm 6 illustrates the distinguishing procedure.

Now, the final probability $\Pr\left[\exists i : \ wt(\nu^2(\Delta_i)) = 4\right]$ for $\texttt{AESQ}_{1\to 9}$ and a corresponding random permutation $\mathcal{R}$ is given by $p_0 = 2^{-26.08}$ and $p = 16 \times 2^{-32} \times (\frac{255}{256})^{60} \approx 2^{-28.34}$ respectively, giving us a success rate of 71% at the data complexity of $2^{26.08}$.

In the next section, we introduce the notion of improbable differential yoyo whereby we try to compose the yoyo game with improbable differentials capitalizing on the

[2]Recall, the classical $4 \to 1$ transition though `AES` MixColumns.

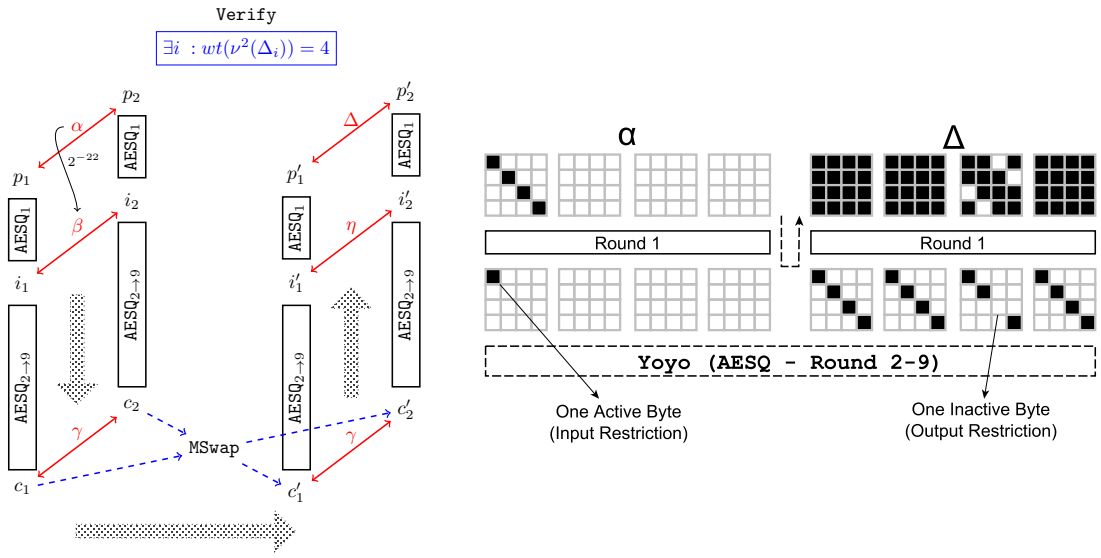


Figure 5-4: Probabilistic 9-round yoyo distinguisher. A configuration of (α, Δ) is shown.

inside-out strategy.

Algorithm 6 Distinguisher for $\texttt{AESQ}_{1\to9}$

Output: 1 for AESQ, -1 otherwise ▷ 9-Round AESQ without last MMC

1: **procedure** ImpDistYoyo(PERMUTE)
2: count $\leftarrow 0$
3: **while** count $\leq 2^{26.08}$ **do**
4: $i_1, i_2 \leftarrow \left\{(m_1, m_2) : \alpha = m_1 \oplus m_2; wt(\nu(\alpha)) = 3 \wedge \exists i : \ wt(\nu^2(\alpha_i)) = 12\right\}$
▷ All four *active* bytes should belong to the same diagonal
5: $p'_1, p'_2 \leftarrow \texttt{Yoyo}\left(i_1, i_2, \texttt{PERMUTE}, \texttt{FULL}, \texttt{FORWARD}\right)$
6: **if** $\exists i : \ wt(\nu^2(\alpha_i)) = 4$ **then return** 1
▷ All four *inactive* bytes should belong to the same diagonal
7: **end if**
8: count $\leftarrow$ count $+ 1$
9: **end while**
10: **return** -1
11: **end procedure**

5.2 Improbable Differential Yoyo

In Indocrypt 2010, Tezcan introduced the notion of improbable differential cryptanalysis [194]. The idea is to find a differential which is less probable for a given permu-

tation (say, $\mathcal{P}$) in comparison to a random permutation (say, $\mathcal{R}$). So if $Pr_{\mathcal{P}}(\Delta_{in} \rightarrow \Delta_{out}) = p_0$ while $Pr_{\mathcal{R}}(\Delta_{in} \rightarrow \Delta_{out}) = p$, then for the *improbability* criteria we must have $p_0 < p$, where $Pr_{\mathcal{P}}$ and $Pr_{\mathcal{R}}$, represent the probabilities of the input difference Δ_{in} and output difference Δ_{out} to occur for $\mathcal{P}$ and $\mathcal{R}$ respectively. Tezcan argued that improbability led to the well-known *impossibility* criteria where $p_0 = 0$. He further proposed an idea known as the *expansion technique* [194] to devise improbable differential by connecting (multiple) differentials with an impossible differential. The expansion technique is briefly stated below:

$Pr_{\mathcal{P}}(\Delta_{mid} \rightarrow \Delta_{out}) = 0$ `Impossible Differential`

$Pr_{\mathcal{P}}(\Delta_{in} \rightarrow \Delta_{mid}) = p'$ `Connecting Differential`

`Let` $Pr_{\mathcal{R}}(\Delta_{in} \rightarrow \Delta_{out}) = p$

`Let` $Pr_{\mathcal{P}}(\Delta_{in} \rightarrow \Delta_{out}) = p_0$ `if` $\Delta_{in} \not\rightarrow \Delta_{mid}$

Then probability of the *improbable* differential for a given permutation $\mathcal{P}$ is given by:

$$Pr_{\mathcal{P}}(\Delta_{in} \rightarrow \Delta_{out}) = p_0 = (1 - p') \times p + p' \times 0 = (1 - p')p \tag{5.1}$$

Our idea is to:

- First, use our notion of *Nested ZDP* with properties of the MixColumns to devise an impossible differential.

- Then, we find a differential that connects the starting ZDP of the yoyo game with input difference of the impossible differential.

Overall, by virtue of the expansion technique, we are able to devise an improbable differential leading to higher number of rounds than that covered by the yoyo game. In order to do this, we will use the *inside-out* technique described next.

5.2.1 The Inside-Out Technique

The inside-out technique [163, 22] has been used extensively in distinguishing public permutations like the Keccak-f permutation [22], whereby the idea is to start from an intermediate state to generate a set of inverted initial states that preserve a certain property (e.g. the zero-sum property in case of Keccak). Here we try to adapt the technique to incorporate the yoyo game. The idea is as follows:

- Play the yoyo game from an intermediate round to generate pairs of input states.
- In the return path of the yoyo game, we extend the number of rounds using an improbable/impossible differential.

In the next subsection we will show different distinguishers based on the following claim which leads to impossible differences at various rounds of `AESQ`.

Claim 5.2.1 (Impossible Difference)**.** *Let the input difference before the r^{th} round (where r denotes an odd round) MegaMixColumns be δ and let exactly one word of δ (say δ_i) be active i.e. $wt(\nu(\delta)) = 3$, then the following will hold*

1. *If $wt(\nu^2(\delta_i)) = 0$, then all 64 SBoxes of the state will be active before $(r+1)^{th}$ round Mega-MixColumns.*
2. *If $wt(\nu^2(\delta_i)) = 0$, then all 16 SuperSBoxes of the state will be active before $(r+2)^{th}$ round Mega-MixColumns.*
3. *Unconditionally, all 4 MegaSBoxes of the state will be active before $(r+4)^{th}$ round Mega-MixColumns.*

Proof. The proof proceeds as below:

1. $wt(\nu(\delta)) = 3$ and $wt(\nu^2(\delta_i) = 0)$ implies that every byte in word δ_i is active. Since only one word is active, we have a byte active in every column. The word-configuration will be in accordance with MegaSBox output words (See Fig. 5-2). Now, the claim follows directly from the property of `AES` MixColumns. *It is well known that the total number of active bytes before and after a MixColumns*

operation cannot be less than 5. Since, the input to every MixColumns operation in the r^{th} round has 1 byte active, then the output has all 4 bytes of the column active with a probability 1. So the entire state is active after r^{th} round MMC. Thus in the $(r+1)^{th}$ round, all SBoxes of the state have non-zero input difference and hence cannot have a zero output difference and hence will be all active deterministically up to the next MMC operation. If the output difference is denoted by η, then

$$\Pr\Big[wt(\nu^2(\eta)) = 1\Big] = 0 \qquad \leftarrow \text{Impossible Difference}$$

2. The argument remains the same with the only difference that after the r^{th} round MMC all 16 Super-Sboxes are activated that span 1.5 rounds ending just before $(r+1)^{th}$ round MMC. Since, all Super-Sboxes are active, it is impossible to have a zero-difference at the output of any of them.

3. For MegaSBox, the restriction $wt(\nu^2(\delta_i)) = 0$ is no longer required. So, we only need $wt(\nu(\delta)) = 3$. Irrespective of the status of the bytes inside the word, after r^{th} round MMC, all 4 MegaSBoxes will be activated and will span 3.5 rounds. Thus, at the end of $(r+4)^{th}$ round MMC, we cannot have the case that any of the words signifying the output of the MegaSBoxes is inactive. If the output difference is denoted by γ, then

$$\Pr\Big[wt(\nu(\gamma)) = 1\Big] = 0 \qquad \leftarrow \text{Impossible Difference}$$

□

The implications of Claim 5.2.1 are captured by Fig. 5-5 which shows the particular case of $r = 9$ relevant for this work. So, we have three impossible differentials covering 10, 10 to 11 and 10 to 13 rounds respectively. We next show how we convert the first two into improbable differentials to get a 9-round and 10-round distinguisher with practical complexities. Later, using the third one, we will devise an impossible

differential distinguisher for 12 rounds.

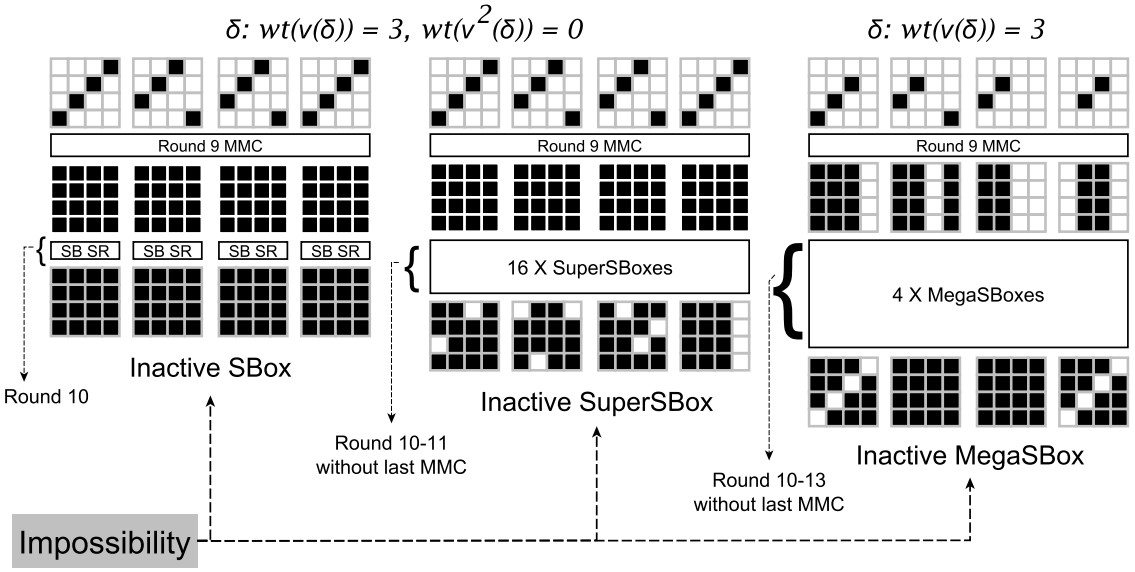


Figure 5-5: Different State Configurations Conforming to Claim 5.2.1

5.2.2 Improbable Differential Yoyo Distinguisher for 9-round and 10-round AESQ

As per the requirement of the expansion technique explained earlier, we need an impossible differential and a *connecting* differential that conforms to its input. Now, Claim 5.2.1 (1,2) already gives us the impossible differential. We are interested particularly for the case when $r = 9$. So, for $r = 9$, we have 1 and 2 round impossible differential (without last MMC as shown in Fig. 5-5). We now use the yoyo game to generate the connecting differential. The strategy is demonstrated in Fig. 5-6. The probabilities can be derived as below:

$$\alpha : wt(\nu(\alpha)) = 3$$
$$\Pr[wt(\nu(\delta)) = 3] = 1 \quad [\because \nu(\alpha) \overset{\text{Yoyo}}{=} \nu(\delta)]$$
$$\Pr[wt(\nu(\delta)) = 3 \wedge wt(\nu^2(\delta_i)) = 0] = \left(\frac{255}{256}\right)^{16} [\delta_i \leftarrow \text{Only active word}]$$

So, the technique is to use the yoyo game to generate an "arbitrary" number of

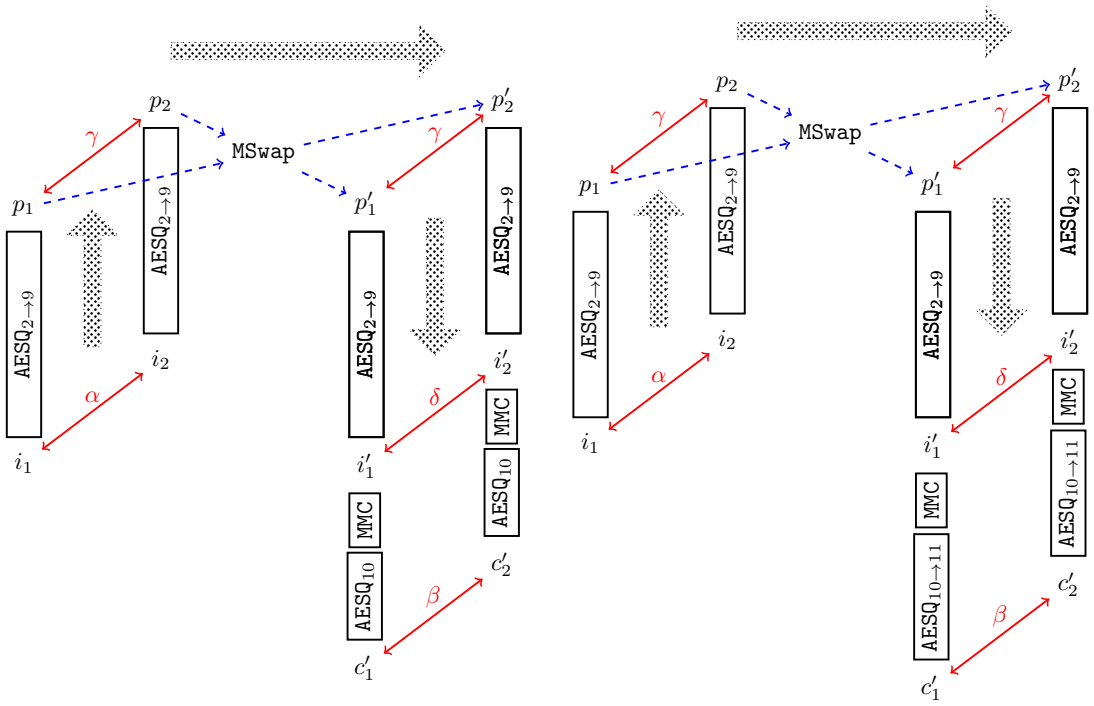


Figure 5-6: Improbable Differential Yoyo distinguisher for 9/10-round AESQ

inputs pairs (p'_1, p'_2) such that the output difference of these pairs over $\mathtt{AESQ}_{2\to 10}$ or $\mathtt{AESQ}_{2\to 11}$ can never have an inactive SBox or Super-Sbox respectively. To ascertain the data complexity one needs to find the probabilities of these events occurring for a random permutation. In the next subsection we find the data-complexity.

We directly use Eq. (5.1) to derive the probability of the combined differential. For the 9-round attack, the probability of observing at least one inactive SBox for a random permutation is $1-\left(\frac{255}{256}\right)^{64} \approx 0.22$. Similarly, for 10 rounds the probability of observing an inactive Super-Sbox is $p = 16 \times \frac{1}{2^{32}} \times \left(\frac{2^{32}-1}{2^{32}}\right)^{15} \approx 2^{-28}$. As per notations of Eq. (5.1), we have:

- For 9 rounds

$$p \approx 0.22$$
$$p' = \left(\frac{255}{256}\right)^{16} \approx 0.94$$
$$p_0 = (1-p')p \approx 0.0132$$

- For 10 rounds

$$p \approx 2^{-28};$$
$$p' = \left(\frac{255}{256}\right)^{16} \approx 0.94$$
$$p_0 = (1-p')p \approx 0.06 \times 2^{-28}$$

Algorithm 7 Distinguisher for $\texttt{AESQ}_{2\to10}/\texttt{AESQ}_{2\to11}$

Output: 1 for AESQ, -1 otherwise ▷ 9/10-Round AESQ without last MMC

procedure ImprDistYoyo(PERMUTE, RNDS)
  if RNDS $= 9$ **then**
    CMPLXTY $\leftarrow 5$
    COND $\leftarrow wt(\nu^2(c'_1 \oplus c'_2)) > 0$
  else
    CMPLXTY $\leftarrow 2^{28}$
    COND $\leftarrow wt(\nu^2(c'_1 \oplus c'_2)) = 4$ ▷ Inactive bytes $\in$ same Super-Sbox
  end if
  count $\leftarrow 0$
  while count $\leq \lceil$CMPLXTY$\rceil$ **do**
    $i_1, i_2 \leftarrow \left\{(m_1, m_2) : wt(\nu(m_1 \oplus m_2)) = 3\right\}$ ▷ One active word in $m_1 \oplus m_2$
    $p'_1, p'_2 \leftarrow \texttt{Yoyo}\left(i_1, i_2, \texttt{AESQ}^{-1}_{2\to9}, \texttt{MID}\right)$ ▷ 9-Round AESQ without last MMC
    $\begin{cases} c'_1 \leftarrow \texttt{PERMUTE}(p'_1); \\ c'_2 \leftarrow \texttt{PERMUTE}(p'_2) \end{cases}$
    if COND $=$ TRUE **then return** -1
    end if
    count $\leftarrow$ count $+ 1$
  end while
  return 1
end procedure

Thus the numbers of input pairs needed to distinguish $\texttt{AESQ}_{2\to10}$ and $\texttt{AESQ}_{2\to11}$ are around $\frac{1}{0.22} \approx 5$ and 2^{28} respectively with a success probability of 82% and 77% respectively, thereby leading to the first practical distinguishers for these rounds of AESQ. For merely 5 samples in case of 9 rounds, the Normal approximation used in [174, Theorem 2] does not hold and so we perform direct calculation of false positive and false negative errors in computing the theoretical estimate of the success probability. Algorithm 7 captures both 9/10 round distinguishers at the same time. In the next section, we introduce the notion of impossible differential yoyo in the inside-out setting. Based on that we develop two distinguishers on 12 and 16 rounds of AESQ.

5.3 Impossible Differential Yoyo

Impossible differential has been shown to be a special case of improbable differential [195, 198]. Now it is easy to note that if the connecting differential used in the expansion technique occurs with a probability 1, then the combined differential becomes impossible. The basic idea is to use the determinism of the yoyo game along with the inside-out technique to arrive at the input of an impossible differential. We do this in two ways: The first way leverages upon the third part of Claim 5.2.1. The second way tries to combine two yoyo games in two directions.

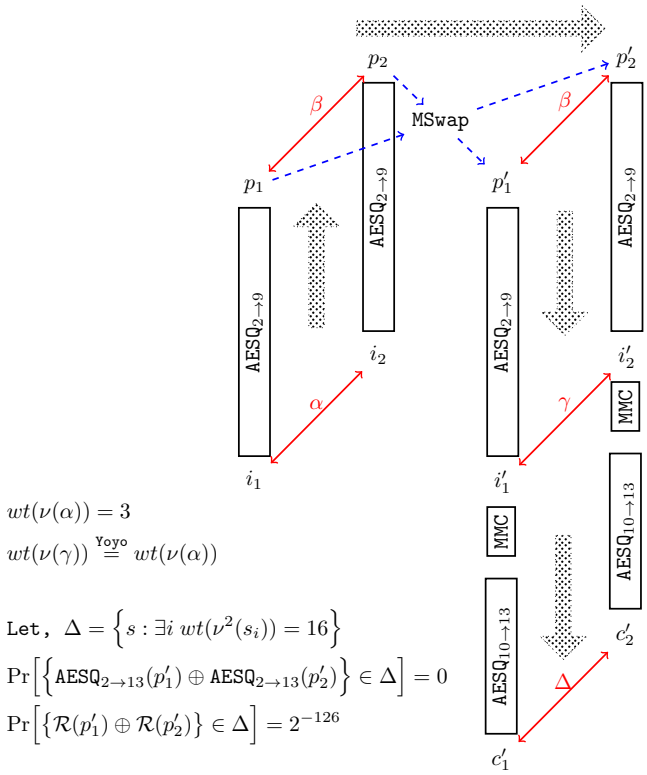
5.3.1 Impossible Differential Yoyo Distinguisher for 12-round AESQ

This attack is similar to the ones described in the previous section with only difference that we no longer have restriction on the connecting differential due to Claim 5.2.1 (3). So what we have is an impossible differential spanning 3.5 rounds due to the MegaSBox and the connecting differential that hold with probability 1 due to the yoyo game. Fig. 5-7a illustrates the strategy while the procedural details are covered by Algorithm 8.

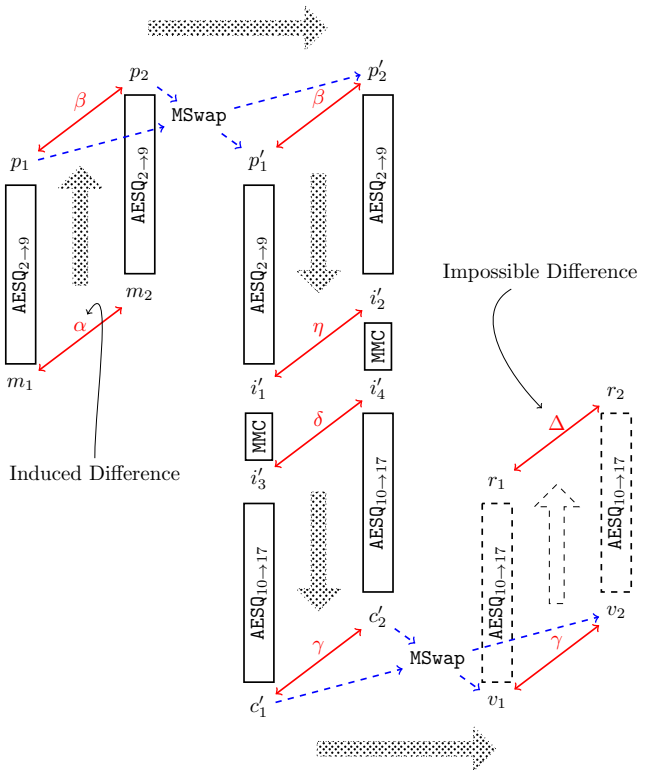
The probability that any one of the words corresponding to the MegaSBoxes is active for a random permutation is $p = \frac{4}{2^{128}}$. Further, we have $p' = 1$ and therefore $p_0 = 0$, resulting in a data complexity of 2^{126} with a success probability of 84%. Next we show an interesting way to combine two yoyo games to come up with a 16-round distinguisher starting from round 2.

5.3.2 Impossible Differential Bi-directional Yoyo Distinguisher for 16-round AESQ

As the name suggests, *bi-directional* yoyo combines two yoyo games. These games are played in opposite directions and employ the inside-out strategy. The ZDP requirements of the two games are different as stated below:



(a) Impossible Differential Yoyo



(b) Impossible Differential Bi-directional Yoyo

Figure 5-7: Impossible Differential Yoyo Distinguishers on $\mathtt{AESQ}_{2\to 13}$ and $\mathtt{AESQ}_{2\to 17}$

Algorithm 8 Distinguisher for $\texttt{AESQ}_{2\to13}$

Output: 1 for AESQ, -1 otherwise ▷ 12-Round AESQ without last MMC

procedure ImpDistYoyo(PERMUTE)
  count $\leftarrow 0$
  while count $\leq 2^{126}$ **do**
    $i_1, i_2 \leftarrow \left\{(m_1, m_2) : wt(\nu(m_1 \oplus m_2)) = 3\right\}$ ▷ One active word in $m_1 \oplus m_2$
    $p'_1, p'_2 \leftarrow \texttt{Yoyo}\left(i_1, i_2, \texttt{AESQ}^{-1}_{2\to9}, \texttt{MID}\right)$ ▷ 9-Round AESQ without last MMC
    $\begin{cases} c'_1 \leftarrow \texttt{PERMUTE}(p'_1) \\ c'_2 \leftarrow \texttt{PERMUTE}(p'_2) \end{cases}$
    if $wt(\nu(c'_1 \oplus c'_2)) = 1$ **then return** -1 ▷ Impossible difference for $\texttt{AESQ}_{2\to13}$
    end if
    count $\leftarrow$ count $+ 1$
  end while
  return 1
end procedure

- Game 1 is played with $\texttt{AESQ}^{-1}_{2\to9}$ without last MMC. Only one word should be active in the input differential. So the weight of ZDP needs to be 3.

  $i'_1, i'_2 \leftarrow \texttt{Yoyo}\left(m_1, m_2, \texttt{AESQ}^{-1}_{2\to9}, \texttt{FULL}, \texttt{BACKWARD}\right) : wt(\nu(m_1 \oplus m_2)) = 3$

- Game 2 is played with $\texttt{AESQ}_{10\to17}$ without last MMC. All words should be active in the input differential. So the weight of ZDP needs to be 0.

  $r_1, r_2 \leftarrow \texttt{Yoyo}\left(i'_3, i'_4, \texttt{AESQ}_{10\to17}, \texttt{FULL}, \texttt{FORWARD}\right) : wt(\nu(i'_3 \oplus i'_4)) = 0$

- In order to connect Game 1 and Game 2, we will use an MMC operation. One can visualize this as the MMC of Round 9 which is excluded while playing Game 1. So the claim is as follows:

  Claim 5.3.1. *If* $i'_3 = \text{MMC}(i'_1)$ *and* $i'_4 = \text{MMC}(i'_2)$*, then*

  1. $wt(\nu(i'_3 \oplus i'_4)) = 0$ *and*
  2. $\Pr\left[wt(\nu(r_1 \oplus r_2)) = 1\right] = 0$

  Proof. The first claim follows from the fact that due to Game 1, $wt(\nu(i'_1 \oplus i'_2)) = 3$. So, we have exactly one word active in $(i'_1 \oplus i'_2)$. This also implies that due to the word configuration (Recall Fig. 5-2) we can have exactly one byte active in

each column of $(i'_1 \oplus i'_2)$. Due to the property of MixColumns, every single active byte in $(i'_1 \oplus i'_2)$ will lead to a fully (all four bytes) active column in $(i'_3 \oplus i'_4)$. Since, the minimum number of active bytes in $(i'_1 \oplus i'_2)$ is one, so after after MMC on i'_1 and i'_2, we will have at least one column active in $(i'_3 \oplus i'_4)$. Now, as each byte in the active column belongs to a different word, so an active column implies four active words i.e. $wt(\nu(i'_3 \oplus i'_4)) = 0$.

The second claim can be easily inferred from Game 2. Since, the input difference of Game 2 has four active words, so we cannot have an *inactive* word in the output difference $r_1 \oplus r_2$. □

The entire bi-directional game is captured by Fig. 5-7b. Once Game 1 and Game 2 are connected, we can appreciate the fact that the combination of the second half of Game 1, the connecting MMC layer and the first half of Game 2 actually behaves like $\texttt{AESQ}_{2\to 17}$ without the last MMC. This leads us in the direction of the distinguishing strategy described in Algorithm 9. So one can arbitrarily generate pairs of inputs for 16 round `AESQ` starting from round 2 excluding last MMC. The corresponding outputs under `MSwap` when subjected to $\texttt{AESQ}^{-1}_{10\to 17}$ without Round 10 MMC can never lead to output difference having one *inactive* word. For a random permutation this happens with a probability of 2^{-126}. So, the data complexity and the success probability remain the same as the 12-round distinguisher.

In the next section, we investigate the Known-Key security of `AES` in the light of the impossible differential yoyo strategies developed above.

5.4 Applications to `AES` in the Known-Key Setting

Rønjom et al. have already shown application of yoyo on `AES` in the secret key paradigm and argued that the maximum penetration was up to 6 rounds. In contrast, here we are more interested in public permutations which is motivated by our need to engage strategies like inside-out and start-in-the-middle which are implicitly inhibited in the secret-key setting. So an obvious direction would be to look at the known-key

Algorithm 9 Distinguisher for $\texttt{AESQ}_{2\to 17}$

Output: 1 for AESQ, -1 otherwise ▷ 16-Round AESQ without last MMC

procedure ImpDistBiYoyo(PERMUTE)
  $\texttt{count} \leftarrow 0$
  while $\texttt{count} \leq 2^{126}$ **do**
    $i_1, i_2 \leftarrow \left\{(m_1, m_2) : wt(\nu(m_1 \oplus m_2)) = 3\right\}$ ▷ One active word in $m_1 \oplus m_2$
    $p'_1, p'_2 \leftarrow \texttt{Yoyo}\left(i_1, i_2, \texttt{AESQ}^{-1}_{2\to 9}, \texttt{MID}, \texttt{BACKWARD}\right)$ ▷ 9-Round AESQ without last MMC
    $\begin{cases} c'_1 \leftarrow \texttt{PERMUTE}(p'_1) \\ c'_2 \leftarrow \texttt{PERMUTE}(p'_2) \end{cases}$
    $(v_1, v_2) \leftarrow \texttt{MSwap}(c'_1, c'_2, \texttt{FORWARD})$ ▷ Excludes possibility of trivial extension
    $\begin{cases} r_1 \leftarrow \texttt{AESQ}^{-1}_{10\to 17}(v_1) \\ r_2 \leftarrow \texttt{AESQ}^{-1}_{10\to 17}(v_2) \end{cases}$ ▷ 9-Round AESQ without last MMC
    if $wt(\nu(r_1 \oplus r_2)) = 1$ **then** ▷ Impossible difference for $\texttt{AESQ}_{2\to 17}$
      return -1
    end if
    $\texttt{count} \leftarrow \texttt{count} + 1$
  end while
  return 1
end procedure

notion under which AES behaves as a public permutation and which opens up the avenue to expose AES to our extension strategies. As suggested, *known-key* refers to the scenario where the attacker has access to the key. Introduced by Knudsen and Rijmen [142] in Asiacrypt 2007, the idea was mainly motivated by the fact that non-existence of known-key distinguishers would imply non-existence of secret-key ones. Additionally, since block-ciphers are often used as primitives in hash functions where key-input could be totally or partially controllable, such kind of known-key analysis is imperative. The known-key security of block ciphers has received a lot of attention with Andreeva et al. attempting to formalize it first [19] and later being systematically treated by Mennink and Preneel [166] in the context of hash functions. Below, we explore how some of the techniques introduced so far adapt to AES in this setting. In the process, we are able to device the one of the *most efficient 8-round known-key distinguisher* in terms of overall cost. It is assumed that the reader is familiar with AES and the notations used here are analogous to AESQ. The basic approach, as also taken in [176] and earlier in this work for AESQ, is to capitalize on the well-known AES Super-Sbox.

Impossible Differential Yoyo for 6-round AES The first idea is to apply the basic impossible differential yoyo technique described in Section 5.3.1. So we use the inside-out philosophy to devise a connecting differential as per the last part of Claim 5.2.1 which is easily adapted to be applicable on AES. So we initiate the yoyo game such that weight of ZDP is three. By virtue of the game, we get back the same ZDP at the end of 3.5 rounds. Now due to MixColumns (MC) of fourth round, all Super-Sboxes get activated. Thus, propagating forward for two rounds, due to the Super-Sbox property, we cannot have the case, that the output difference has at least one *inactive* Super-Sbox. The same for a random permutation would occur with a probability of 2^{-30}.

Impossible Differential Bi-directional Yoyo for 8-round AES The bi-directional yoyo trick introduced in the last section extends easily to the known-key model of AES.

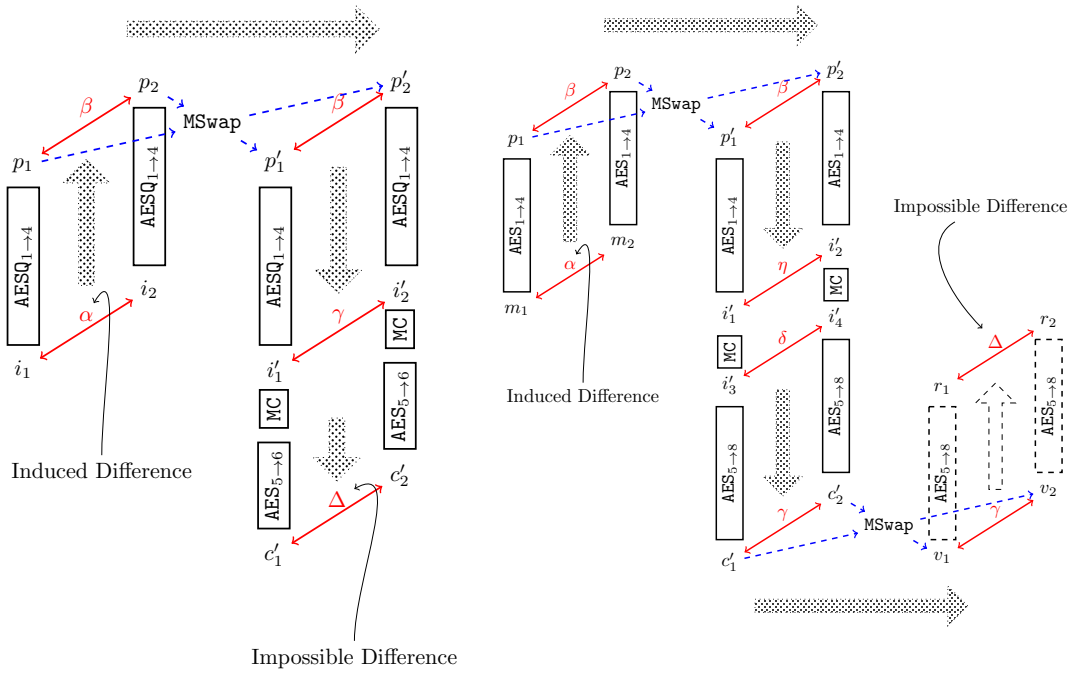


Figure 5-8: Impossible Differential Yoyo based Known-Key distinguisher for 6/8-round AES

Since a single $S \circ L \circ S$ instance covers 4 rounds barring the last MixColumns (MC), two back-to-back yoyo games with MC in between extends the attack to 8 rounds. As argued earlier, since the same impossible differential is used here, we are able to devise an 8-round known-key distinguisher with a complexity of 2^{30} with negligible memory. Fig. 5-8 depicts both the 6 and 8 round distinguishers.

Since, the introduction of known-key model, AES, in particular, has been analyzed extensively. Below we look at the state-of-the-art in devising 8-round known-key distinguishers on AES. Our inclination to 8 rounds stems from the urge to make a direct comparison with the maximum rounds we are able to penetrate here. This is captured in Table 5.2, where all the complexities (including ours) correspond to a success probability of 84%. Grassi and Rechberger [117] provide a near exhaustive analysis of known-key distinguishers while improving most of the available ones and also reporting new ones. Their main contribution was to show that the idea proposed by Gilbert [113] is not limited to 10 rounds and can be further extended to 12 rounds. Going back to results on 8 rounds, we can see from Table 5.2 that the result reported here is only superseded by the extended multiple differential trail attack by Grassi

and Rechberger while incurring some extra memory complexity.

Table 5.2: 8-round Known-Key Distinguishers on AES

Time Complexity	Memory Complexity	Property	Reference
2^{64}	2^{64}	Uniform Distribution	[113]
2^{48}	2^{32}	Differential Trail	[114]
2^{44}	2^{32}	Multiple Differential Trail	[132]
2^{30}	**negligible**	**Impossible Differential Bi-directional Yoyo**	Section 5.4
2^{23}	2^{16}	Extended 7-Round Multiple Differential Trail	[117]

5.5 Practical Verification

All distinguishers having practical complexities, i.e., $\texttt{AESQ}_{2\to9}$, $\texttt{AESQ}_{1\to9}$, $\texttt{AESQ}_{2\to10}$, $\texttt{AESQ}_{2\to11}$, $\texttt{AES}_{1-6}$ and $\texttt{AES}_{1-8}$ were implemented on a Java based hyper-threaded environment and verified to be conforming to the expected results. The details are furnished in Appendix 5.7. The experimental details of the success probabilities computed for the distinguishers are provided in Appendix 5.7.1. We now provide a discussion on all the distinguishing strategies introduced in this work.

5.6 Discussion

Distinguishing public permutations has always been seen as tricky due to the *unkeyed* nature of these crypto primitives. Two important things that are needed to be ensured to make a distinguisher in this setting *meaningful* are *non-triviality* and *randomness*.

A distinguisher should not be trivial in the sense that it should not be trivially extendible meaning that it is not supposed to work for any arbitrary number of rounds. Let us now discuss all distinguishers presented here in the light of this intended property. It is easily noticeable that the limitation of the yoyo principle to hold only for $(S \circ L \circ S)$ is the first line of defence against non-triviality. Thus

rounds covered based on only a single yoyo game cannot be extended beyond any 8 rounds of AESQ excluding last MMC while starting from an even round. The same is true for 4 rounds of AES without the last MC. As regards the strategies that were composed with yoyo, they mostly rely on differentials that work over certain specified rounds and hence are not arbitrarily extendible. The only exception comes with the bi-directional yoyo distinguisher where the last verification might seem non-standard. However, the non-triviality is ensured by the last MSwap operation (for example, Step 7 of Algorithm 9 for AESQ). Without that operation the distinguisher would be trivial because one could append any number of rounds as a part of first half of the second yoyo game and invert the *same* number of rounds in the verification step.

The requirement of randomness is fundamental to devising distinguishers in general and for public permutations in particular. This is primarily because due to the unkeyed nature one could easily enumerate the permutation and employ the inverse to have a *trivial* verification. The distinguishing strategy should allow in principle sufficient randomness in choosing the inputs. In this respect, all distinguishers developed in the current work allow for that. Most of the distinguishers use first half of the yoyo game as a subroutine and can generate almost arbitrary number of inputs which conform to certain input differences. These inputs lead to certain required differences in the middle either deterministically by virtue of the yoyo technique or probabilistically augmenting yoyo with probable, improbable or impossible differentials.

This work explores many ways to extend the yoyo game. The authors in [176], have shown attacks on 3/5 rounds AES, where they extend the basic yoyo game. However, with the exception of the $\texttt{AESQ}_{2\to 9}$ and $\texttt{AESQ}_{1\to 9}$ distinguishers, the strategies reported here differ from the ones shown in [176]. This is mostly because of the inside-out philosophy used here which becomes inapplicable in the secret-key setting. The main contribution of this work comes in the form of the idea of using the inside-out technique to partially deploying the yoyo game as an input generator. The notion of *Nested ZDP* introduced here seems to work nicely as a combiner of yoyo and classical differential cryptanalysis. Along with MixColumns, the techniques used here exploit the properties of SuperSBoxes and MegaSBoxes. The bi-directional yoyo

Table 5.3: Distinguishers reported in this work

	#R	Start → End	Complexity	Strategy	Remarks
AESQ	8	2 → 9	1	Yoyo	Basic Yoyo
	9	1 → 9	$2^{26.08}$	Yoyo + Nested ZDP	First 9 round Distinguisher starting from Round 1
	9	2 → 10	5	Improbable Differential Yoyo	Uses the inside-out technique
	10	2 → 11	2^{28}		
	12	2 → 13	2^{126}	Impossible Differential Yoyo	
	16	2 → 17	2^{126}	Bi-directional Impossible Differential Yoyo	Uses inside-out with bi-directional Yoyo
AES	6	1 → 6	2^{30}	Impossible Differential Yoyo	Uses the inside-out technique
	8	1 → 8	2^{30}	Bi-directional Impossible Differential Yoyo	Uses inside-out with bi-directional Yoyo

game is the most effective strategy leading to doubling of the number of rounds penetrated. One might look critically at the last verification which uses $\texttt{AESQ}^{-1}_{10\to17}$. However, usage of such kind of verification is available in literature of distinguishers on Feistel schemes [153]. Moreover, as argued earlier, the strategy ensures non-triviality. Except the 12 and 16 round distinguishers of AESQ, all other distinguishers of AESQ and AES rely on practical data complexities and negligible memory. The closest comparable results for AESQ are due to Bagheri et al. [25] who report rebound and time-memory trade-off attacks. Though the maximum number of rounds is same, the current work exponentially outperforms the former both in terms of data and memory requirements.

In case of 8-round AES, in the known-key setting, with the exception of [117], our result beats all other works, while being the only one that requires negligible memory. Table 5.3 summarizes the attacks presented here. It should be noted however that

comparing attacks in the known-key model only by their complexity is not completely fair, as one has to take into consideration also the rate of *simplicity* of the found non-random property, which may affect the chances to extend the distinguisher to more rounds or to more powerful attacks. In this respect, our attack is not directly comparable to several of the previous results, as the non-random property we find is somewhat complex.

5.7 Experimental Verification

Most of the distinguishers presented in this chapter have practical complexities. These have been performed experimentally and their complexities have been verified. All the experiments have been performed on a system with Intel core i7-6700 CPU@3.40 $\times$ 8 and memory 16GB. For programming, we have used Java `openjdk` version `1.8.0_181`. For implementing `AES` functionalities, we have used publicly available code [93].

The distinguisher for $\texttt{AESQ}_{2\to 9}$ is deterministic in nature and this attack is performed in negligible time using Algorithm 5. The complexity of the distinguisher for $\texttt{AESQ}_{1\to 9}$ is $2^{26.05}$. For the attack, pairs of plaintexts having only one word difference have been chosen at random. Plaintexts pairs whose differences have not mitigated into a byte after the first round of `AESQ` are filtered out. Among the remaining pairs, all possible swapping are done between the corresponding ciphertexts. Algorithm 6 describes the distinguisher. This attack has been performed using a single thread and it took 17435867($\approx 2^{24.05}$) iterations in 557 seconds for successfully finding a pair of plaintexts and a swap vector which conforms to our claim. The following pair of texts conform to our claim when swapped after forward permutation using the given vector.

$$\begin{bmatrix} \texttt{fa b1 5a 2f} & \texttt{00 68 a1 e5} & \texttt{b3 55 81 01} & \texttt{c3 3d 4c 8a} \\ \texttt{b7 64 0e f1} & \texttt{92 d5 10 b0} & \texttt{89 cb 6f 51} & \texttt{b1 63 6c 00} \\ \texttt{f8 9f 22 15} & \texttt{97 1a 44 7b} & \texttt{2d c3 64 c6} & \texttt{67 64 b2 43} \\ \texttt{28 87 32 25} & \texttt{18 df 4f 98} & \texttt{c6 b0 c7 a4} & \texttt{28 2a 59 9d} \end{bmatrix}$$

$$\begin{bmatrix} 2e\ b1\ 5a\ 2f & 00\ 68\ a1\ e5 & b3\ 55\ 81\ 01 & c3\ 3d\ 4c\ 8a \\ b7\ 70\ 0e\ f1 & 92\ d5\ 10\ b0 & 89\ cb\ 6f\ 51 & b1\ 63\ 6c\ 00 \\ f8\ 9f\ f2\ 15 & 97\ 1a\ 44\ 7b & 2d\ c3\ 64\ c6 & 67\ 64\ b2\ 43 \\ 28\ 87\ 32\ 4c & 18\ df\ 4f\ 98 & c6\ b0\ c7\ a4 & 28\ 2a\ 59\ 9d \end{bmatrix}$$

$$vector = \begin{bmatrix} 01\ 01\ 01\ 00\ 01\ 01\ 00\ 01\ 01\ 01\ 00\ 01\ 01\ 01\ 01\ 00 \end{bmatrix}$$

- Initial difference of two input states:

$$\begin{bmatrix} d4\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 \\ 00\ 14\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 \\ 00\ 00\ d0\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 \\ 00\ 00\ 00\ 69 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 \end{bmatrix}$$

- Difference of two states after one round

$$\begin{bmatrix} 02\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 \\ 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 \\ 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 \\ 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 & 00\ 00\ 00\ 00 \end{bmatrix}$$

- Two states before swapping (before last MixColumns and ShiftRows)

$$\begin{bmatrix} 4e\ 12\ 6c\ 05 & 2c\ ab\ 29\ 96 & c7\ 76\ 26\ 7c & 68\ a6\ b2\ e1 \\ b2\ 38\ 9b\ 7a & f3\ 68\ e0\ 02 & 10\ f1\ 40\ f6 & dc\ 42\ 0c\ 59 \\ 67\ 97\ 9b\ 0a & 6c\ 61\ 25\ 0e & 0f\ f4\ 92\ cd & 59\ 33\ 18\ d3 \\ 53\ d2\ 4c\ 06 & 2e\ da\ e9\ 29 & 88\ fa\ 5f\ 2d & b1\ ae\ b7\ be \end{bmatrix}$$

$$\begin{bmatrix} 24\ 2c\ e7\ 0d & 40\ 4d\ a4\ 20 & 92\ 30\ a8\ fa & b6\ 0e\ 8a\ 2d \\ 8b\ 89\ a0\ 1a & 14\ c5\ e7\ e9 & c6\ 84\ 2e\ 86 & 49\ 32\ 13\ a5 \\ c1\ 31\ 6b\ 2f & a1\ 87\ 1c\ cf & ea\ ca\ ce\ cd & e0\ 46\ c3\ 1f \\ 03\ c4\ c4\ e5 & 6f\ e5\ f3\ ad & cf\ 8d\ be\ 5a & 8c\ ad\ 7a\ bd \end{bmatrix}$$

- Difference of two permuted states before swapping

$$\begin{bmatrix} 6a & 3e & 8b & 08 & 6c & e6 & 8d & b6 & 55 & 46 & 8e & 86 & de & a8 & 38 & cc \\ 39 & b1 & 3b & 60 & e7 & ad & 07 & eb & d6 & 75 & 6e & 70 & 95 & 70 & 1f & fc \\ a6 & a6 & f0 & 25 & cd & e6 & 39 & c1 & e5 & 3e & 5c & 00 & b9 & 75 & db & cc \\ 50 & 16 & 88 & e3 & 41 & 3f & 1a & 84 & 47 & 77 & e1 & 77 & 3d & 03 & cd & 03 \end{bmatrix}$$

- Two states after swapping (before last MixColumns and ShiftRows)

$$\begin{bmatrix} 4e & 12 & 6c & 0d & 2c & ab & a4 & 96 & c7 & 76 & a8 & 7c & 68 & a6 & b2 & 2d \\ b2 & 38 & 9b & 1a & f3 & 68 & e7 & 02 & 10 & f1 & 2e & f6 & dc & 42 & 0c & a5 \\ 67 & 97 & 9b & 2f & 6c & 61 & 1c & 0e & 0f & f4 & ce & cd & 59 & 33 & 18 & 1f \\ 53 & d2 & 4c & e5 & 2e & da & f3 & 29 & 88 & fa & be & 2d & b1 & ae & b7 & bd \end{bmatrix}$$

$$\begin{bmatrix} 24 & 2c & e7 & 05 & 40 & 4d & 29 & 20 & 92 & 30 & 26 & fa & b6 & 0e & 8a & e1 \\ 8b & 89 & a0 & 7a & 14 & c5 & e0 & e9 & c6 & 84 & 40 & 86 & 49 & 32 & 13 & 59 \\ c1 & 31 & 6b & 0a & a1 & 87 & 25 & cf & ea & ca & 92 & cd & e0 & 46 & c3 & d3 \\ 03 & c4 & c4 & 06 & 6f & e5 & e9 & ad & cf & 8d & 5f & 5a & 8c & ad & 7a & be \end{bmatrix}$$

- Difference of two permuted states after swapping

$$\begin{bmatrix} 6a & 3e & 8b & 08 & 6c & e6 & 8d & b6 & 55 & 46 & 8e & 86 & de & a8 & 38 & cc \\ 39 & b1 & 3b & 60 & e7 & ad & 07 & eb & d6 & 75 & 6e & 70 & 95 & 70 & 1f & fc \\ a6 & a6 & f0 & 25 & cd & e6 & 39 & c1 & e5 & 3e & 5c & 00 & b9 & 75 & db & cc \\ 50 & 16 & 88 & e3 & 41 & 3f & 1a & 84 & 47 & 77 & e1 & 77 & 3d & 03 & cd & 03 \end{bmatrix}$$

- Difference of states after 1st round (during inverse permutation)

$$\begin{bmatrix} d7 & 00 & 00 & 00 & d9 & 00 & 00 & 00 & 46 & 00 & 00 & 00 & 75 & 00 & 00 & 00 \\ 00 & c3 & 00 & 00 & 00 & 06 & 00 & 00 & 00 & 15 & 00 & 00 & 00 & c8 & 00 & 00 \\ 00 & 00 & ec & 00 & 00 & 00 & e6 & 00 & 00 & 00 & 65 & 00 & 00 & 00 & 16 & 00 \\ 00 & 00 & 00 & ec & 00 & 00 & 00 & \mathbf{00} & 00 & 00 & 00 & 8f & 00 & 00 & 00 & 02 \end{bmatrix}$$

- Difference of two states after inverse permutation

$$\begin{bmatrix} b5 & d0 & e8 & 68 & 6c & b4 & 07 & \mathbf{00} & 90 & 19 & 58 & 4e & 4c & b1 & ff & b1 \\ 06 & 1a & 63 & cb & \mathbf{00} & 48 & 08 & cf & 9a & 77 & 37 & 1b & bf & 0d & 35 & c5 \\ e4 & f5 & 61 & de & d1 & \mathbf{00} & de & fb & a7 & 65 & 6c & fa & 34 & 2d & 4a & ed \\ 1c & c0 & 83 & d6 & bf & 9f & \mathbf{00} & d4 & 25 & a3 & c1 & 67 & 18 & 2e & f0 & 0f \end{bmatrix}$$

For performing distinguishing attack on $\texttt{AESQ}_{2\to 11}$, 2^{28} pairs of plaintexts have been used. To reduce the running time, 16 threads have been used. It is an improbable differential distinguisher. For $\texttt{AESQ}_{2\to 11}$, the distinguisher took 5047.616 seconds to run for all 2^{28} iterations and have not found the improbable differential property; while for the random permutation it took 225885057 ($\approx 2^{27.75}$) iterations to find the improbable differential property; and thus it conforms to our claim. The distinguisher for $\texttt{AESQ}_{2\to 10}$ is also improbable differential in nature with a complexity of 5. In negligible time, its property has been verified. Algorithm 7 describes both of these distinguishers. We did not implement distinguishers for $\texttt{AESQ}_{2\to 13}$ and $\texttt{AESQ}_{2\to 17}$ as their complexities are impractical.

The distinguishing attacks on $\texttt{AES}_{1\to 8}$ introduced in this chapter have practical complexity. This distinguishing attack is similar to Algorithm 9 with reduced complexity. The distinguishing algorithm ran for 2^{30} iterations in 2977.575 seconds in the above machine and have not found the impossible differential with an inactive SuperSBox. For random permutation, we have found this differential in 187840320 ($\approx 2^{27.48}$) iterations.

5.7.1 Success Probability

For experimental verification of success probabilities, a blackbox is considered which can act as a cipher $\mathcal{C}$ (we consider $\mathcal{C}$ =AESQ/AES) or as a random permutation $\mathcal{R}$. To calculate the success probability, consider the following confusion matrix in Table 5.4.

Table 5.4: Confusion Matrix of $\mathcal{C}$ and $\mathcal{R}$

Actual \ Observed	$\mathcal{C}$	$\mathcal{R}$
$\mathcal{C}$	$o_c - n_{FP}$	n_{FN}
$\mathcal{R}$	n_{FP}	$o_r - n_{FN}$

The experiment is performed considering $\mathcal{C}$ in the blackbox a_c times and $\mathcal{R}$ in the

Table 5.5: Experimental Verification of Success Probability

Distinguisher	$\#n$	Blackbox	Detected as AESQ/AES	Detected as $\mathcal{R}$	Experimental Success Probability	Estimated Success Probability
$\texttt{AESQ}_{1\to9}$	1000	$\texttt{AESQ}_{1\to9}$	826	174	0.75	0.70
		$\mathcal{R}$	324	676		
$\texttt{AESQ}_{2\to10}$	1000	$\texttt{AESQ}_{2\to10}$	645	355	0.70	0.82
		$\mathcal{R}$	258	742		
$\texttt{AESQ}_{2\to11}$	100	$\texttt{AESQ}_{2\to11}$	100	0	0.86	0.77
		$\mathcal{R}$	29	71		
$\texttt{AES}_{1\to6}$	100	$\texttt{AES}_{1\to6}$	100	0	0.81	0.83
		$\mathcal{R}$	39	61		
$\texttt{AES}_{1\to8}$	100	$\texttt{AES}_{1\to8}$	100	0	0.82	0.83
		$\mathcal{R}$	37	63		

blackbox a_r times. Based on the output, $\mathcal{C}$ is decided o_c times and $\mathcal{R}$ is decided o_r times. The numbers of false positives and false negatives are denoted by n_{FP} and n_{FN} respectively. Then the success probability is given by

$$\begin{aligned} Pr[Success] &= \frac{(o_c - n_{FP}) + (o_r - n_{FN})}{o_c + o_r} \\ &= \frac{(o_c - n_{FP}) + (o_r - n_{FN})}{a_c + a_r}. \end{aligned}$$

Table 5.5 shows the experimental results of various distinguishers and their corresponding success probabilities. The success probabilities calculated by experimentation is close enough to their respective theoretically estimated values.

5.8 Chapter Summary

In this work we explored the impact of the yoyo cryptanalytic strategy on public permutation AESQ as well as AES in the known-key model. We deployed the basic yoyo technique to get a deterministic 8-round distinguisher for AESQ and extended it

using our notion of Nested ZDP to include the first round using around 2^{26} queries. In addition to this we used the inside-out strategy to augment yoyo using classical, improbable and impossible differentials to reach 9, 10, 12 rounds starting from round 2 with data complexities of about $2^2, 2^{28}$ and 2^{126} respectively. The final strategy devised here allows us to combine two yoyo games giving a 16-round distinguisher using 2^{126} queries. The impossible difference based yoyo strategies when applied to `AES` lead to known-key distinguishers for 6 and 8 rounds with a complexity of 2^{30}. One may note that all improbable distinguishers reported can be converted to impossible ones while paying some extra cost in terms of data complexity. The success probabilities of the attacks have been computed to be high enough and all distinguishers with practical complexities were verified using computer simulations.

6

BOOMEYONG ATTACKS ON AES-BASED DESIGNS

Contents

Cryptanalysis is one of the most important ways of determining the strength of a cryptosystem. Ever since the introduction of differential cryptanalysis by Biham and Shamir [47], a multitude of cryptanalytic techniques that build upon the basic idea of differential cryptanalysis has been proposed. Among these, a certain class of attacks particularly aims to divide a cipher into multiple sub-ciphers and study the sub-ciphers individually often analyzing the interactions between them. These methods find high probability trails (primarily due to the lesser number of rounds) for the sub-ciphers and compose them efficiently to mount an attack on the complete cipher. Some of the prominent candidates of this class are the boomerang attack [201], amplified boomerang attack (rectangle attack) [136], impossible differential attack [39], rebound attack [163]. These techniques have been widely applied

to several ciphers: like the rectangle attack on Serpent [42, 43], Kasumi [44]; impossible differential attacks on AES [157, 216, 46], CLEFIA, Camellia, LBlock, Simon, ARIA [209, 210, 74, 73], Rijndael-160 and Rijndael-224 [167], rebound attack on Whirlpool and Grøstl [163, 164], KECCAK [98] and boomerang attack on AES in single-key setting [53] and in the related-key setting [55, 57, 115, 190, 107]. A recent addition to the class include the retracing boomerang attack [99] and the extended truncated differential attack [29] on AES. The retracing boomerang attack has been proposed in Eurocrypt 2020 by Dunkelman *et al*. and at the outset it tries to additionally spatially divide the sub-ciphers. The extended truncated differential attack mounts distinguishing and key-recovery attack on 5-round and 6-round AES by prepending a round that starts from the diagonal subspaces as proposed in [118].

In particular, the boomerang attack is the center of interest concerning this work as the techniques developed here extensively rely on it. Boomerang attack, introduced by Wagner, makes use of two differentials to construct a distinguisher spanning over a large number of rounds when it is not possible to devise a single differential. As stated earlier, it conceptually divides a cipher into two sub-ciphers where each differential corresponds to each sub-cipher. Though initially thought to be independent, it has been shown that the differentials can rely on each other based on their interaction at the boundary of the sub-ciphers. The dependency can either lead to an incompatibility as shown by Murphy [168] or can be exploited to improve the number of rounds as shown later by the idea of *s-box switch*, *ladder switch* [57, 55] and further generalized by the sandwich attack [100, 101]. This also leads to the introduction of new tools like the boomerang connectivity table (BCT) [83], Feistel BCT [71] and the boomerang distribution table (BDT) [203]. Another interesting cryptanalytic technique that is structurally similar to boomerang (though it does not divide the cipher into sub-ciphers) is the yoyo game which was introduced by Biham *et al*. to analyze Skipjack [38]. In Asiacrypt 2017, it has been used to devise a deterministic distinguisher for generic 2-round Substitution-Permutation Network (SPN) [176] which leads to key recovery attacks on 5-round AES. The concept of yoyo game is further extended and applied to AES in known-key setting [180] and on ForkAES [27]

Table 6.1: Comparisons of key recovery attacks on AES and Pholkos. Note that, time complexity is measured in terms of one AES and Pholkos encryption respectively (where no unit is mentioned). Memory complexity is measured in terms of memory required to store a single state of the primitive. All the attacks tabulated for AES are key recovery attacks. For 10-round Pholkos, key recovery attack using boomeyong is compared with the distinguishing attack given by the designers. CP and ACC are Chosen Plaintext and Adaptively Chosen Ciphertext respectively. Mix. Diff., Ret. Boom., and ETD refers to Mixture Differential, Retracing Boomerang and Extended Truncated Differential respectively.

Primitive	**Attack Type**	**Complexity**			**Ref.**
		Data	**Time**	**Mem.**	
5-round AES-128	Improved Square	2^{33} CP	2^{35}	Negl.	[106]
	Boomerang	2^{39} ACC	2^{39}	2^{33}	[53]
	Mix. Diff.	2^{24} CP	2^{24}	$2^{21.5}$	[30]
	Mix. Diff.	2^{32} CP	2^{34}	2^{32}	[116]
	Yoyo Attack	$2^{13.3}$ ACC	2^{33}	Negl.	[176]
	Partial Sum	2^{8} CP	2^{40}	Negl.	[200]
	Ret. Boom.	2^{9} ACC	2^{23}	2^{9}	[99]
	Ret. Boom.	2^{15} ACC	$2^{16.5}$	2^{9}	[99]
	Boomeyong	**2^{49} ACC**	**2^{48} XOR**	**2^{23}**	**Section 6.2.1**
6-round AES-128	Yoyo Attack	$2^{122.8}$ ACC	$2^{121.8}$ XOR	Negl.	[176]
	Exchange Attack	$2^{88.2}$ CP	$2^{88.2}$	Negl.	[33]
	Boomerang	2^{71} ACC	2^{71}	2^{33}	[53]
	Ret. Boom.	2^{26} ACC	2^{80}	2^{35}	[99]
	Partial Sum	$2^{34.5}$ CP	2^{44}	2^{32}	[106]
	ETD	$2^{71.3}$ CP	$2^{78.7}$	-	[29]
	Boomeyong	**$2^{79.72}$ ACC**	**2^{78}**	**2^{28}**	**Section 6.2.2**
10-round Pholkos	Boomerang	2^{260} ACC	2^{260}	2^{32}	[70]
	Boomeyong	**$2^{189.8}$** ACC	**$2^{188.8}$** XOR	**2^{122}**	**Section 6.3.2**

in secret-key setting. Table 6.1 lists the complexities of the attacks presented in this chapter on 5-round AES-128 (variant of AES with key length of 128 bits), 6-round AES-128 and 10-round Pholkos respectively along with the other attacks. It is evident

from the table that for 10-round Pholkos, the boomeyong attack performs fairly well with respect to the boomerang attack in terms of data and time complexities. In addition, using the boomerang strategy distinguishing attack is mounted on Pholkos; whereas using the boomeyong technique secret key is recovered.

The work investigates the yoyo technique further to essentially extend the number of rounds that it can penetrate. The new approach can be visualized like an embedding of the yoyo game inside a boomerang trail, where the upper trail of the boomerang essentially conforms to the yoyo while the lower trail is a standard but specially crafted differential trail. It applies the concept of the s-box switch and the ladder switch in the boundary of the upper and lower trail. The primary motivation is to construct the lower trail in such a way that the difference added to the ciphertexts leads to a yoyo *word-swap* in the boundary of upper and lower trails. This in turn satisfies the essential criteria of the yoyo and leads to return of the yoyo with probability 1 which can be verified at the top, like the classical yoyo trick. We prove how the s-box switch and ladder switch help achieve the required word-swap (see Fig. 6-2). The proof idea stems from the fact that the words that swap can be mapped to an equivalent s-box switch while the words that remain unchanged make a ladder switch. The price we pay is the construction of a truncated differential trail superimposed on the yoyo which behaves like the upper trail of the boomerang. This is the motivation for using the term *embedding* while visualizing this setting. So in classical boomerang terms if the truncated upper trail has a complexity p and the lower trail has a complexity q, owing to the word-swap happening at the boundary, the complexity of the complete boomerang distinguisher is pq^2. We save a factor of p while going up due to the yoyo property.

As a natural application, first of all, $5(= 4+1)$-round `AES` is considered, where the yoyo covers the first 4 rounds constituting the upper trail and the lower trail covers 1 round. By embedding yoyo within boomerang, first a distinguisher is reported at the expense of 2^{47} oracle queries contributing to the data complexity and 2^{46} XOR operations contributing to the time complexity. The distinguisher is used further to correctly recover the secret key of `AES`-128 (variant of `AES` with key length of 128 bits)

Table 6.2: Key recovery attacks reported in this work. ACC is adaptive chosen ciphertexts.

Attack	Complexity			Ref.
	Data	Time	Memory	
5-round AES	2^{49} ACC	2^{48} XOR	2^{23}	Section 6.2.1
6-round AES	$2^{79.72}$ ACC	2^{78}	2^{28}	Section 6.2.2
10-round Pholkos †	$2^{189.8}$ ACC	$2^{188.8}$ XOR	2^{122}	Section 6.3.2

† 512-bit key

with the time complexity of 2^{48} XOR operations. The next result is the application to 6-rounds which is achieved by sandwiching the yoyo in-between a classical 1-round differential on top and the lower boomerang trail developed in the 5-round attack. The result is a key recovery attack on 6-round AES-128 with the time complexity of 2^{78} AES encryptions and the data complexity of $2^{79.72}$ adaptive chosen ciphertexts. Note that, the distinguishing attack on the AES is independent of the key size whereas the key recovery attacks described in the chapter are applicable on AES-128. However, the key recovery attacks can be further extended to recover the key of 6/7-round of a variant of AES with 256-bit key. In the rest of the chapter, unless otherwise mentioned, AES-128 is referred to as AES.

Finally, to show the versatility of the strategy a 10-round key recovery attack is mounted on a very recently proposed AES based tweakable block cipher Pholkos. We support all our claims with theoretical arguments. The combination of the two strategies seems to be an interesting proposition and may lead to improved results for other SPN ciphers as well thereby providing better insights. One can appreciate the fact that the proposed technique bears structural similarity with some of the well-known results. For instance, the 6-round attack can easily be seen in the framework of the sandwich attack where 4 rounds of AES form the middle layer. In that sense, this work reports the first result where the middle layer consists of 4 rounds of AES. On the other hand, the attack can also be shown to have a close relation to the retracing attack, a discussion on which is furnished later (See Section 6.5). The contributions

of the current work are summarized in Table 6.2.

The chapter is organized as follows. The notion of embedding yoyo within boomerang is introduced and thoroughly illustrated in Section 6.1. In Section 6.2, the developed cryptanalytic technique is applied on 5-round and 6-round `AES` to mount key recovery attacks. As an additional application of the developed techniques, a 10-round attack on `Pholkos` [70] is shown in Section 6.3. Section 6.5 illustrates the close relation between retracing boomerang attack and the attacks presented in this chapter. In Section 6.4, the key recovery attacks on 5-round and 6-round `AES`-128 are extended to recover the key of a variant of `AES` with key size of 256 bits. Finally, the chapter is summarized in Section 6.6.

6.1 Boomeyong: Embedding Yoyo within Boomerang

The central notion of this work is to devise a cryptanalytic technique by combining two powerful techniques: yoyo and boomerang. The same conceptual division as used in the boomerang attack is considered for embedding yoyo within boomerang leading to a new strategy which we call *boomeyong*. Proposition 1 states that there is a deterministic distinguisher for $S \circ L \circ S$ construction irrespective of the internal structure of S and L layer (Here, S corresponds to the substitution layer and L corresponds to linear layer). The trick is to use this $S \circ L \circ S$ layer as the upper trail in devising the boomerang trail. The problem of embedding yoyo game within boomerang is that the previous is based on classical differential whereas for the latter one truncated forms are considered. Refer to Fig. 6-1 for the attack. Let $E : \mathbb{F}_{2^k}^n \mapsto \mathbb{F}_{2^k}^n$ be a cipher which is divided into two parts: E_0 (upper) and E_1 (lower). E_0 is comprised of initial $S \circ L \circ S$ layers and the remaining parts of the cipher is considered as E_1. Now, P^1, P^2, P^3 and P^4 be four plaintexts which are encrypted by E to obtain C^1, C^2, C^3 and C^4 respectively. Aim is to simulate yoyo game in the upper trail E_0. Let $Q^i = E_0(P^i)$ for $1 \leq i \leq 4$. Therefore, if P^1, P^2 is considered as initial pair, then by virtue of yoyo game $\nu(P^1 \oplus P^2) = \nu(P^3 \oplus P^4)$. This also implies that Q^3, Q^4 can be obtained by swapping words between Q^1, Q^2. Therefore, $Q^1 \oplus Q^2 = Q^3 \oplus Q^4 = \beta$

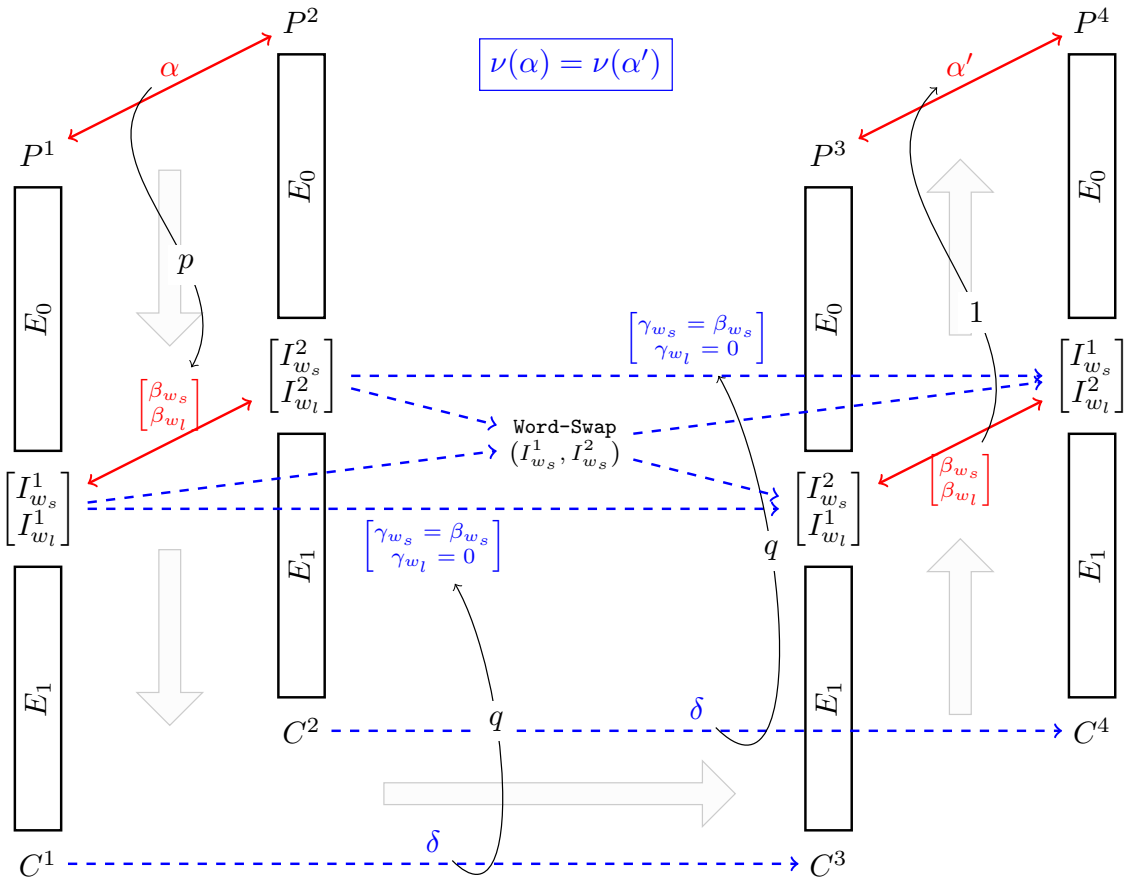


Figure 6-1: Embedding yoyo within boomerang. Note that, for the yoyo game E_0 corresponds to $S \circ L \circ S$ layer, whereas for the boomerang there is no such constraints. Here, the trail superimposed on yoyo is $\alpha \to \beta$ with a probability p. The words of β that are intended to be swapped are denoted by β_{w_s}. These words will be switched using corresponding words in the lower trail $\delta \to \gamma$ which holds with probability q using the idea of s-box switch. The remaining words γ i.e. γ_{w_l} in the lower trail are zero thereby leading to a ladder switch of the corresponding words in β i.e. β_{w_l}. Note that $\Pr[\beta \to \alpha'] = 1$ due to the yoyo trick.

(say). Consider, $P^1 \oplus P^2 = \alpha$, $P^3 \oplus P^4 = \alpha'$. The difference of boomerang with the attack developed in this work is that for the former one $\alpha = \alpha'$, whereas for the latter one $\alpha = \alpha'$ does not hold always; instead $\nu(\alpha) = \nu(\alpha')$ must hold.

Constructing the lower trail is quite similar to the construction of the lower trail in the boomerang attack. Let $Q^1 \oplus Q^3 = \gamma$, which gives $Q^2 \oplus Q^4 = Q^1 \oplus Q^3 = \gamma$. Now, for the lower half a trail $\delta \xrightarrow{E_1^{-1}} \gamma$ needs to be constructed. For realizing the 'Swapping of Words' in the middle (the boundary of E_0 and E_1), a special kind of

relationship must exist between β and γ.

Theorem 5. *Let $Q^1, Q^2, \gamma \in \mathbb{F}_{2^k}^n$ and $Q^1 \oplus Q^2 = \beta$. Consider, $Q^i = Q_1^i||Q_2^i||\cdots||Q_n^i$, $\beta = \beta_1||\cdots||\beta_n$ and $\gamma = \gamma_1||\cdots||\gamma_n$. If $J \subset \{1, \cdots, n\}$, $J \neq \emptyset$ and for all $j \in J$, $\gamma_j = \beta_j$; otherwise, $\gamma_j = 0$, then, $Q^1 \oplus \gamma$, $Q^2 \oplus \gamma$ can be formed by swapping words between Q^1, Q^2.*

Proof. Construct $v \in \mathbb{F}_2^n$ as

$$v_j = \begin{cases} 0, & \text{if } j \in J; \\ 1, & \text{Otherwise.} \end{cases}$$

for $(1 \leq j \leq n)$. Now, following the Definition 4 construct $\rho^v(Q^1, Q^2)$. For $(1 \leq j \leq n)$

$$\rho^v(Q^1, Q^2)_j = \begin{cases} Q_j^1, & \text{if } v_j = 1; \\ Q_j^2, & \text{if } v_j = 0. \end{cases}$$

$$\implies \rho^v(Q^1, Q^2)_j = \begin{cases} Q_j^1, & \text{if } \gamma_j = 0; \\ Q_j^2, & \text{if } \gamma_j = \beta_j. \end{cases}$$

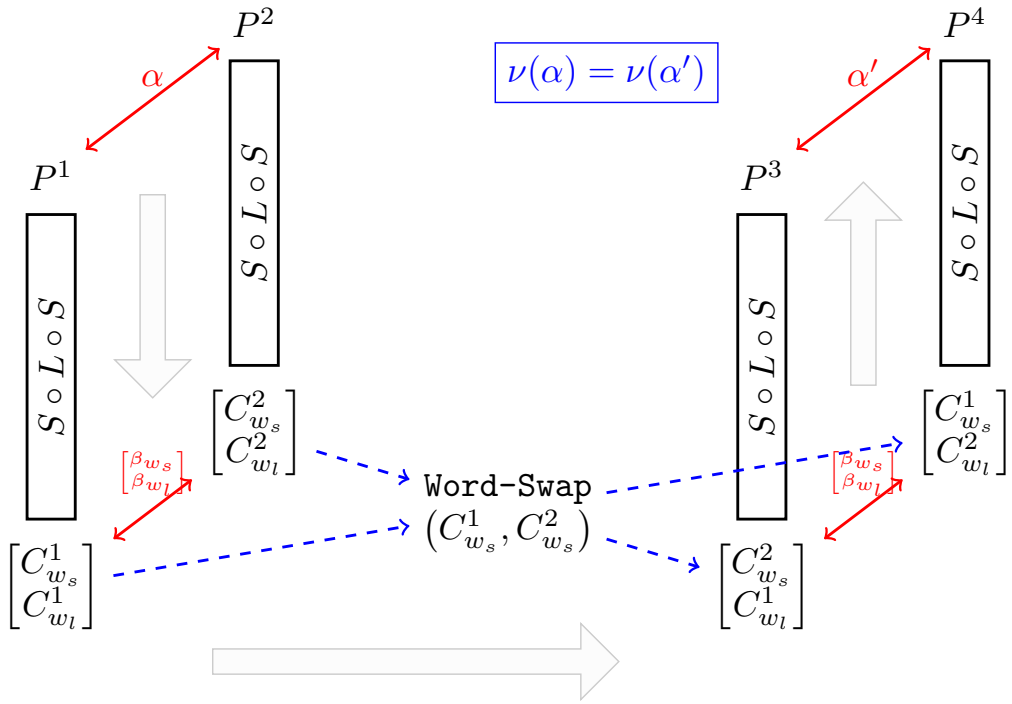
$$\implies \rho^v(Q^1, Q^2)_j = \begin{cases} Q_j^1, & \text{if } \gamma_j = 0; \\ Q_j^1 \oplus \beta_j, & \text{if } \gamma_j = \beta_j. \end{cases}$$

$$\implies \rho^v(Q^1, Q^2)_j = Q_j^1 \oplus \gamma_j$$

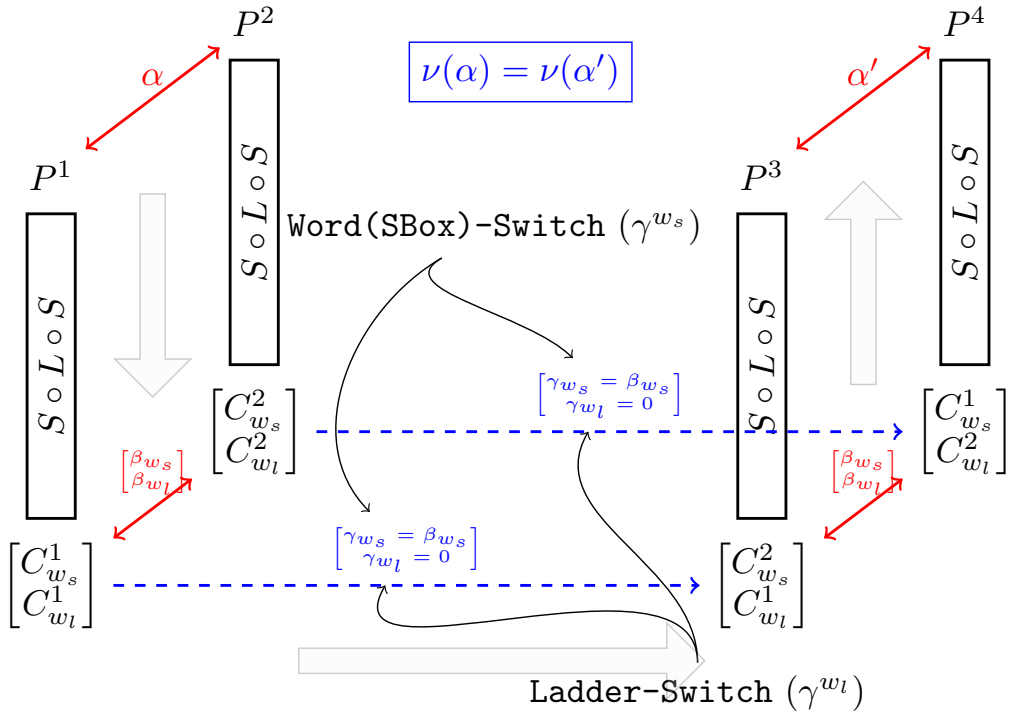
Therefore, $\rho^v(Q^1, Q^2) = Q^1 \oplus \gamma$. In similar way, it can be proved that $\rho^v(Q^2, Q^1) = Q^2 \oplus \gamma$. □

Theorem 5 states that the words in γ either should be zero or equal to the value of the same word in β. This ensures that in the middle swapping of words has taken place between the initial pair and thus for E_0 yoyo game is run. Fig. 6-2 shows the swapping mechanism in the middle.

For the upper trail E_0, α is not fixed; instead $\nu(\alpha)$ is fixed. Let $Pr[\{\alpha|\nu(\alpha) = t\} \xrightarrow{E_0} \beta] = p$ and $Pr[\delta \xrightarrow{E_1^{-1}} \gamma] = q$. Therefore, at the cost of pq^2 probability Q^3, Q^4



(a) Yoyo Word-Swap



(b) Equivalent S-box and Ladder switch

Figure 6-2: Visualizing Yoyo Word-Swap as a combination of S-box switch and Ladder switch operations

are formed by swapping words between Q^1, Q^2 and thus with the same probability it is expected that $\nu(P^3 \oplus P^4) = \nu(P^1 \oplus P^2)$. Let $wt(\nu(P^1 \oplus P^2)) = t$. If E is a random permutation, then this event would occur with probability 2^{-tk}. While embedding yoyo within the boomerang distinguishers, such upper and lower trails should be considered for which $pq^2 > 2^{-tk}$.

Attack Idea. Based on the analysis, the following are the steps of devising a distinguisher by embedding yoyo within boomerang. Suppose, access to oracle $\mathcal{O}$ is given and the distinguisher tries to distinguish that whether $\mathcal{O}$ is E or a random permutation.

1. Choose two plaintext P^1, P^2 such that $wt\Big(\nu(P^1 \oplus P^2)\Big) = t$. Encrypt them using $\mathcal{O}$ to obtain C^1, C^2 respectively.
2. Prepare $C^3 = C^1 \oplus \delta$, $C^4 = C^2 \oplus \delta$ and decrypt them by $\mathcal{O}$ to obtain P^3, P^4.
3. Check whether $\nu(P^1 \oplus P^2) = \nu(P^3 \oplus P^4)$ or not.
4. If $\nu(P^1 \oplus P^2) = \nu(P^3 \oplus P^4)$, then distinguish $\mathcal{O}$ as E; otherwise, repeat step 1 to step 3 $\frac{1}{pq^2}$ times. Even after repeating $\frac{1}{pq^2}$ if distinguisher fails to find a quartet (P^1, P^2, P^3, P^4) such that $\nu(P^1 \oplus P^2) = \nu(P^3 \oplus P^4)$, then distinguish $\mathcal{O}$ as a random permutation.

Now, the techniques developed here are extensively applied to 5-round and 6-round `AES` and 10-round `Pholkos`.

6.2 Boomeyong Attacks on `AES`

In the previous section, it is shown how to embed yoyo within a boomerang. The first application of this technique is mounting attacks on 5-round and 6-round `AES`. The main disadvantage of appending a boomerang trail under the yoyo is that it is no longer possible to swap words between ciphertexts deterministically. In this regard, first of all, a probabilistic yoyo game needs to be devised. The next three definitions

define the diagonals, inverse diagonals and columns of an AES state. The notations in [33] are used in these definitions whereas the notions behind these definitions are described in [118]. The notation $\subset_\phi$ is used to denote non-null proper subset.

Definition 5. *[118, 33] For a set $I \subset_\phi \{0,1,2,3\}$, let $\mathcal{C}_I = \{(i,j) : i \in S, j \in I\}$. For an AES state X, a set of columns I is represented by $\mathcal{C}_I(X)$.*

Definition 6. *[118, 33] For a set $I \subset_\phi \{0,1,2,3\}$, let $\mathcal{D}_I = \{(i, j+i \mod 4) : i \in S, j \in I\}$. For an AES state X, a set of diagonals I is represented by $\mathcal{D}_I(X)$.*

Definition 7. *[118, 33] For a set $I \subset_\phi \{0,1,2,3\}$, let $\mathcal{ID}_I = \{(i, j-i \mod 4) : i \in S, j \in I\}$. For an AES state X, a set of inverse diagonals I is represented by $\mathcal{ID}_I(X)$.*

The following two lemmas provide the basis of devising a probabilistic yoyo game for AES. However, probabilistic yoyo was already considered in [33] and Lemma 1, Lemma 2 are the special cases of Theorem 5, Theorem 6 respectively in [33]. While [33] avoids the adaptive setting, there are adaptive considerations of [33] in the distinguisher setting [32, 31]. The main motivation of devising such a game is to penetrate more rounds at the expense of probability. For 5-round AES, the aim is to add such a difference in the ciphertext so that in the fourth round before mixcolumns swapping of inverse diagonals is realized.

Lemma 1. *[33] Let $I, J \subset_\phi \{0,1,2,3\}$ and $p^1, p^2 \in \mathbb{F}_{2^8}^{4\times 4}$. Then the probability that a set of inverse diagonals J are swapped between p^1, p^2, given that a set of columns I are swapped is given by $P_{ID}(|I|,|J|) = 2^{-8\times\left(4(|I|+|J|)-2|I||J|\right)}$.*

Proof. Note that, $|\mathcal{C}_I \cap \mathcal{ID}_J| = |I||J|$. So, $|\mathcal{C}_I \cup \mathcal{ID}_J| = 4(|I|+|J|) - |I||J|$. Among all these $\mathcal{C}_I \cup \mathcal{ID}_J$ bytes, if the bytes only in $\mathcal{C}_I \cap \mathcal{ID}_J$ are active between p^1, p^2 then column swap is equivalent to inverse diagonal swap. Therefore, bytes in $(\mathcal{C}_I \cup \mathcal{ID}_J) \setminus (\mathcal{C}_I \cap \mathcal{ID}_J)$ needs to be inactive. $|(\mathcal{C}_I \cup \mathcal{ID}_J) \setminus (\mathcal{C}_I \cap \mathcal{ID}_J)| = 4(|I|+|J|) - 2|I||J|$. Hence, the required probability $P_{ID}(|I|,|J|) = 2^{-8\times\left(4(|I|+|J|)-2|I||J|\right)}$ is achieved. □

Lemma 2. *[33] Let $p^1, p^2 \in \mathbb{F}_{2^8}^{4\times 4}$ and $c^1 = f(p^1), c^2 = f(p^2)$, where f is $MC \circ AK \circ SB \circ SR \circ AK$. Then the probability of occurence of certain p^1, p^2, such that swapping*

of a set of inverse diagonals $I \subset_\phi \{0,1,2,3\}$ *between* c^1, c^2 *is equivalent to swapping of inverse diagonals between* p^1, p^2 *is given by* $P_{swap}(|I|) = \sum_{j=1}^{3} \binom{4}{j} P_{ID}(|I|, j)$.

Proof. It is easy to visualize that due to SR, swapping of $\mathcal{ID}_I$ between c^1, c^2 is equivalent to swapping of $\mathcal{C}_I$ between p^1, p^2. Lemma 1 states that swapping of $\mathcal{C}_I$ is equivalent to swapping of $\mathcal{ID}_J$ (where $J \subset_\phi \{0,1,2,3\}$) when bytes in $(\mathcal{C}_I \cup \mathcal{ID}_J) \setminus (\mathcal{C}_I \cap \mathcal{ID}_J)$ of $(p^1 \oplus p^2)$ are inactive. Such p^1, p^2 occur with probability $P_{ID}(|I|, |J|)$. By taking sum over all possible choices of J, $P_{swap}(|I|) = \sum_{j=1}^{3} \binom{4}{j} P_{ID}(|I|, j)$. □

For $|I| = 1$, $P_{swap} \approx 2^{-46}$, which is its maximum value. For a better visualization of Lemma 1 consider the case when $I = \{3\}$ and $J = \{2,3\}$. In Fig. 6-3 only the bytes in $\mathcal{C}_{\{3\}} \cap \mathcal{ID}_{\{2,3\}}$ are active. So, swapping the last column between p^1, p^2 can also be considered as swapping of last two inverse diagonals.

Now, we give example of Lemma 2. Let p^1, p^2, c^1 and c^2 be four `AES` states as shown in Fig. 6-4a where $c^1 = f(p^1)$ and $c^2 = f(p^2)$ (Here, f is the same function as described in Lemma 2). $\mathcal{ID}_{\{3\}}$ is swapped between c^1 and c^2 to obtain c'^1 and c'^2 as shown in Fig. 6-4b. Let $p'^1 = f^{-1}(c'^1)$ and $p'^2 = f^{-1}(c'^2)$. Now, swapping of $\mathcal{ID}_{\{3\}}$ between c^1 and c^2 can be equivalently considered as swapping of $\mathcal{C}_{\{3\}}$ between p'^1 and

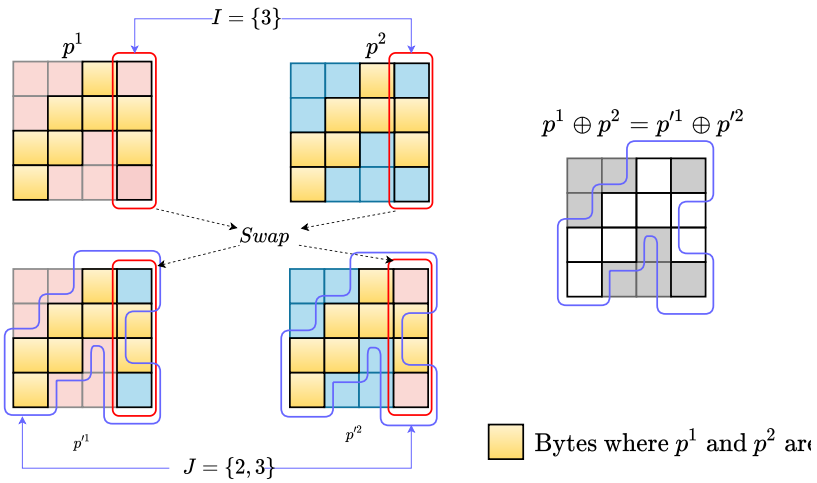
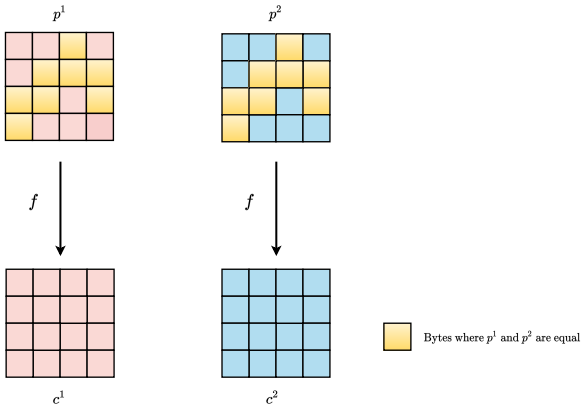
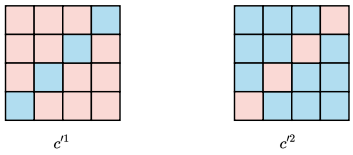


Figure 6-3: Visualization of Lemma 1 when $I = \{3\}$ and $J = \{2,3\}$. As $I = \{3\}$ the last column between p^1 and p^2 is swapped, which is equivalent to swapping of the third and fourth inverse diagonals between p^1 and p^2 because of the positions of inactive bytes in $p^1 \oplus p^2$. Note that, in the last column of p'^1 and p'^2 there are two swapped bytes.

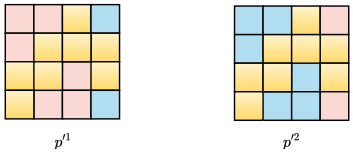
p'^2 due to the function f. Due to the bytes that are equal in p^1 and p^2, this can also be considered as swapping of $\mathcal{ID}_{\{2,3\}}$ between p^1 and p^2 (Fig. 6-4c).



(a) Depiction of p^1, p^2, c^1 and c^2.



(b) c'^1 and c'^2 which is obtained by swapping $\mathcal{ID}_{\{3\}}$ between c^1 and c^2.



(c) p'^1 and p'^2 which can be equivalently obtained by swapping $\mathcal{ID}_{\{2,3\}}$ between p^1 and p^2.

Figure 6-4: An example elaborating a case described in Lemma 2.

Note that, in this example a case is shown when $J = \{2, 3\}$; but, in Lemma 2 all the cases are considered where $J \subset_\phi \{0, 1, 2, 3\}$.

Next, we apply these results to devise a yoyo game embedded within a boomerang for 5-round `AES`.

6.2.1 Distinguishing and Key Recovery Attacks on 5-round `AES`

The attack strategy discussed above is applied to devise a 5-round `AES` distinguisher, which is subsequently converted into a key recovery attack. First of all, 5-round `AES` is divided into two parts- before MC of the 4-th round is termed as E_0 and the remaining part of the cipher is termed as E_1. Note that, E_0 is comprised of $S \circ L \circ S$ layer where S and L corresponds to `AES` Super-Sbox and MC respectively.

Fig. 6-5 depicts the E_0 and E_1 partition in AES. Now, Theorem 1 and Lemma 2 are

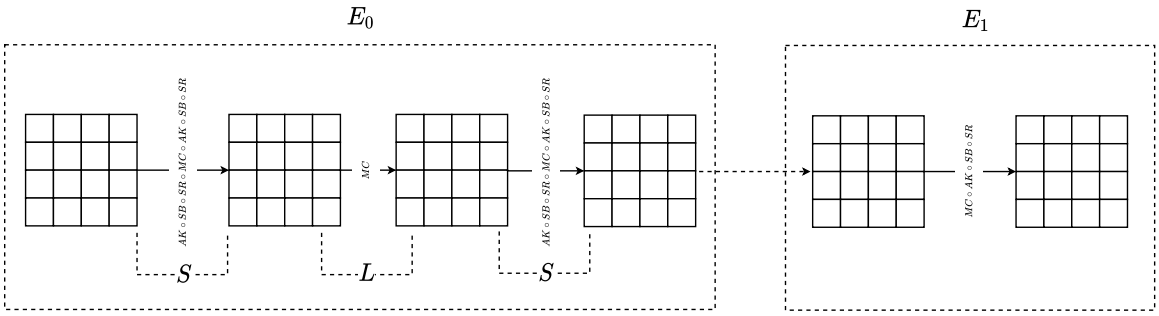


Figure 6-5: Partitioning 5-round AES in E_0 and E_1

combined to design a 5-round AES distinguisher by devising a probabilistic yoyo game by embedding yoyo within boomerang.

Definition 8. *Let $\alpha \in \mathbb{F}_{2^8}^{4\times4}$ be a state and $v \in \mathbb{F}_2^4$ be a vector. Then a state $\tau^v(\alpha) \in \mathbb{F}_{2^8}^{4\times4}$ is constructed from α such that for $0 \leq i \leq 3$*

$$\mathcal{ID}_{\{i\}}(\tau^v(\alpha)) = \begin{cases} \mathcal{ID}_{\{i\}}(\alpha), & \text{if } v_i = 0; \\ 0, & \text{Otherwise.} \end{cases}$$

.

Lemma 3. *Let p^1, $p^2 \in \mathbb{F}_{2^8}^{4\times4}$ and $c^1 = R_5(p^1)$, $c^2 = R_5(p^2)$ where R_5 is 5-round AES or alternatively $R_5 = E_1 \circ E_0$. For any vector $v \in \mathbb{F}_2^4$ such that $1 \leq wt(v) \leq 3$, let $c'^1 = c^1 \oplus \tau^v(c^1 \oplus c^2)$, $c'^2 = c^2 \oplus \tau^v(c^2 \oplus c^1)$ and $p'^1 = R_5^{-1}(c'^1)$, $p'^2 = R_5^{-1}(c'^2)$. Then $\nu(p^1 \oplus p^2) = \nu(p'^1 \oplus p'^2)$ occurs with probability $P_{swap}\Big(4 - wt(v)\Big)$.*

Proof. Let $s^1 = E_0(p^1)$ and $s^2 = E_0(p^2)$. Due to Lemma 2, the probability of occurence of certain s^1, s^2 such that swapping of $\mathcal{ID}_I$ between c^1, c^2 is equivalent to swapping of $\mathcal{ID}_J$ (where $I, J \subset_\phi \{0,1,2,3\}$) between s^1, s^2 is $P_{swap}(|I|)$. Let $s'^1 = E_1^{-1}(c'^1)$ and $s'^2 = E_1^{-1}(c'^2)$. Due to the existence of Super-Sbox in E_1, the intermediate pair s'^1, s'^2 can be considered as constructed from s^1, s^2 as follows. $s'^1 = s^1 \oplus \gamma$ and $s'^2 = s^2 \oplus \gamma$, where some inverse diagonals in γ are zero and some of them are exactly equal to the same inverse diagonal in $s^1 \oplus s^2$. Thus by Theorem 5, s'^1, s'^2 is constructed from s^1, s^2 using word swap. Then by Proposition 1, this new

pair should preserve the zero difference property. So, the zero difference property over $E_1 \circ E_0$ ($E_1 \circ E_0$ is R_5) can be preserved at the expense of $P_{swap}(|I|)$ probability. From Definition 8 it can be concluded that $|I| = 4 - wt(v)$. □

Note that, in Lemma 3, the value of $P_{swap}\Big(4 - wt(v)\Big)$ is maximum ($\approx 2^{-46}$) when $wt(v) = 3$. Next, the upper trail and the lower trail are constructed for 5-round AES distinguisher by leveraging on Lemma 3. For lower trail, $v = 1110$ ($I = \{3\}$) is considered and for better understanding of the upper trail, $J = \{3\}$ is shown in Fig. 6-6.

Constructing the Upper Trail. Refer to Fig. 6-6 for the upper trail. For α, pair of plaintexts p^1, p^2 are chosen such that $wt\Big(\nu(p^1 \oplus p^2)\Big) = 1$. In β, at the cost of 2^{-48}, 6 bytes in $(\mathcal{C}_{\{3\}} \cup \mathcal{ID}_{\{3\}}) \setminus (\mathcal{C}_{\{3\}} \cap \mathcal{ID}_{\{3\}})$ remain inactive. By considering the cases when $J = \{0\}, \{1\}$ or $\{2\}$, the probability is increased to 2^{-46}. We ignore the cases when $|J| > 1$, as it has a negligible effect on the cumulative probability.

Constructing the Lower Trail. For 5-round AES, the construction of the lower trail partially depends on the upper trail. At least one word of γ should be equal to a word in the same position of β. In Fig. 6-6, β_3 is equal to γ_3; $\gamma_0 = \gamma_1 = \gamma_2 = 0$. To generate such γ, dependency on the upper trail is required while constructing δ. Let p^1, p^2 are encrypted to obtain c^1, c^2. For $0 \leq i \leq 3$, δ is constructed as follows-

$$\delta_i = \begin{cases} c_3^1 \oplus c_3^2, & \text{if } i = 3; \\ 0, & \text{Otherwise.} \end{cases}$$

Note that, by Definition 8, $\delta = \tau^v(c^1 \oplus c^2)$. In the upper trail, β occurs with probability 2^{-46}. In the lower trail, one may think that $\delta \xrightarrow{E_1^{-1}} \gamma$ occurs probabilistically. But assuming that β has occured, $\delta \xrightarrow{E_1^{-1}} \gamma$ occurs deterministically. This determines that the overall complexity of the attack is 2^{-46}.

Attack Overview.

1. Prepare a structure of 2^{23} plaintexts p^i, $i \in \{1, 2, \cdots 2^{23}\}$ such that all bytes are constant except the bytes in principal diagonal (0^{th} diagonal), which are

different for each p^i.

2. For $0 \leq i \leq 2^{23}$, query encryption oracle with each p^i to obtain c^i.

3. For $0 \leq i \leq 2^{23} - 1$ and for $i + 1 \leq j \leq 2^{23}$,

   (a) Construct δ as-

$$\delta_m = \begin{cases} c_3^i \oplus c_3^j, & \text{if } m = 3; \\ 0, & \text{Otherwise.} \end{cases}$$

   for $0 \leq m \leq 3$.

   (b) Prepare $c'^i = c^i \oplus \delta$ and $c'^j = c^j \oplus \delta$. Query decryption oracle with c'^i, c'^j to obtain p'^i, p'^j.

   (c) Check whether $\nu(p^i \oplus p^j) = \nu(p'^i \oplus p'^j)$. If yes, distinguish oracle as 5-round `AES` and refer to (p^i, p^j, p'^i, p'^j) as a quartet.

4. If no quartet is found, distinguish oracle as random permutation.

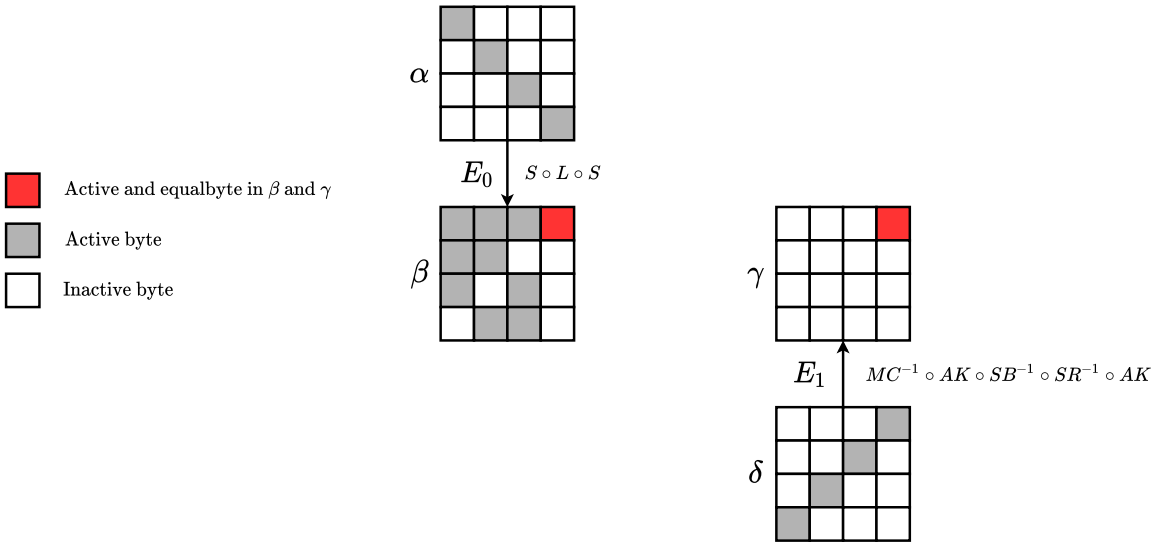


Figure 6-6: Upper and Lower Trail of 5-round `AES`. In this trail, the red-colored byte should be equal in β and γ in order to realize the inverse diagonal swap in the boundary of E_0 and E_1.

Analysis. The data complexity of the attack is (2^{23} encryption queries + 2^{47} decryption queries) $\approx 2^{47}$ decryption queries. The time complexity of the attack is 2^{46} XOR operations. The memory complexity is 2^{23} `AES` state which is used to store the encrypted plaintexts.

Experimental Verification. Due to high data complexity, it is quite difficult to run the complete attack. Instead, an experiment is run to verify the existence of such claimed trails. One such trail is listed in Appendix A. In addition, an experiment for the distinguishing attack is run on 64-bit AES whose details are provided in Section 6.2.3.

Key Recovery Attack.

The key recovery attack is an extension of the distinguishing attack. Refer to Fig. 6-7 for the attack. Let's assume distinguisher has successfully found a quartet (p^1, p^2, p'^1, p'^2) and its corresponding ciphertexts (c^1, c^2, c'^1, c'^2). Consider the active bytes of $(c^1 \oplus c'^1)$ in Z. SR^{-1} aligns the bytes in the last column. Guess the last column of K, invert the bytes using SB^{-1}. Consider the differential in Y, apply MC^{-1} to it and check whether it transits to a single byte or not in X. The guesses for which only a single active byte is obtained in X are right guesses. The active byte in X can have 255 different values; thus for the active diagonal $255 \approx 2^8$ right key candidates are obtained. The process is repeated for the remaining three diagonals which gives a total 2^{32} right key candidates. An exhaustive search is done over these 2^{32} candidates to recover the right key.

Analysis. For guessing each column, four different right pairs are required. So, the data complexity and the memory complexity is $4 \times 2^{47} = 2^{49}$ adaptive chosen plaintexts and ciphertexts and 2^{23} `AES` states respectively. Once a right pair is found using the distinguisher, 2^8 key candidates for a column can be retrieved by doing $2^{32} \times 2$ one round `AES` encryption for a column. To retrieve key candidates for all the columns, $2^{32} \times 2 = 2^{33}$ one round `AES` encryption needs to be done. Considering five such operations as 5-round `AES`, $2^{33}/5 = 2^{30.5}$ `AES` encryptions are required. For exhaustive search, 2^{32} more encryptions are required. So, the total time complexity

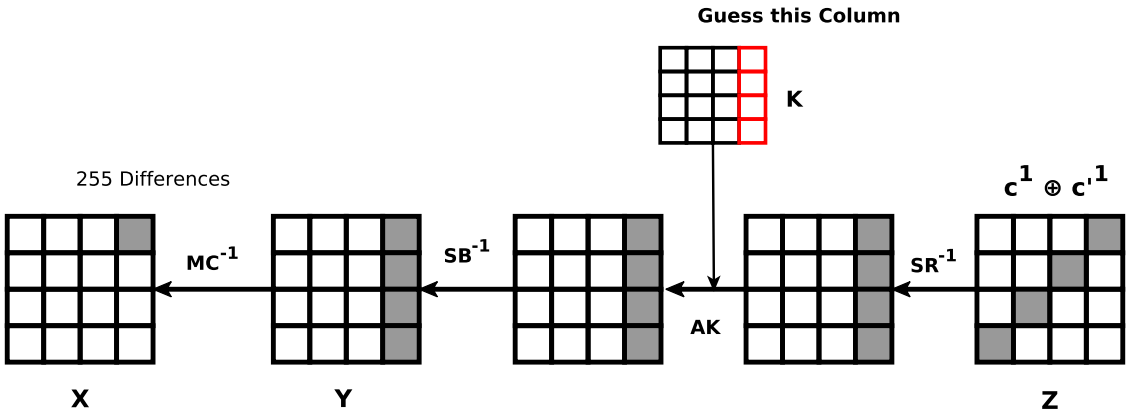


Figure 6-7: Key Recovery Attack on 5-round AES

is $2^{32} + 2^{30.5} \approx 2^{32.4}$ AES encryptions and $4 \times 2^{46} = 2^{48}$ XOR operations.

6.2.2 Key Recovery Attack on 6-round AES

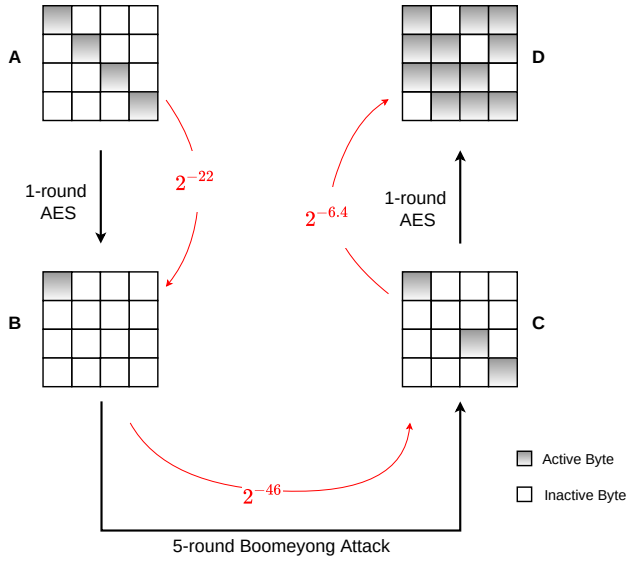
The 6-round key recovery attack on AES is the extension of the 5-round attack described in this chapter. The 6-round attack extensively uses the *4-to-1* property of the AES in the initial round. One round is prepended to the 5-round boomeyong attack. As shown in Fig. 6-8a, if a diagonal is inactive in **D** for a pair then it is included in the candidate set. The main problem is that the candidate set contains right and wrong pairs as for a random pair, any one of the diagonals is inactive with probability $4 \times 2^{-32} = 2^{-30}$. However, using the boomeyong attack such a pair can be obtained with much lesser probability. Hence, to retrieve the right key candidate using the candidate pairs the notion of the signal-to-noise ratio is applied.

Attack Idea. Refer to Fig. 6-8a for the attack. Choose pairs of plaintexts such that only 4 bytes of a diagonal of the pairs are active; the remaining bytes are inactive. Query the pairs to the encryption oracle to obtain corresponding ciphertext pairs. An inverse diagonal is swapped between the ciphertexts to obtain new pair of texts which are queried to the decryption oracle to obtain new pair of plaintexts. As already stated in Section 6.2.1, with probability 2^{-46} swapping an inverse diagonal between the ciphertexts is equivalent to swapping an inverse diagonal between the intermediate states in the previous round. This is a base condition for the yoyo

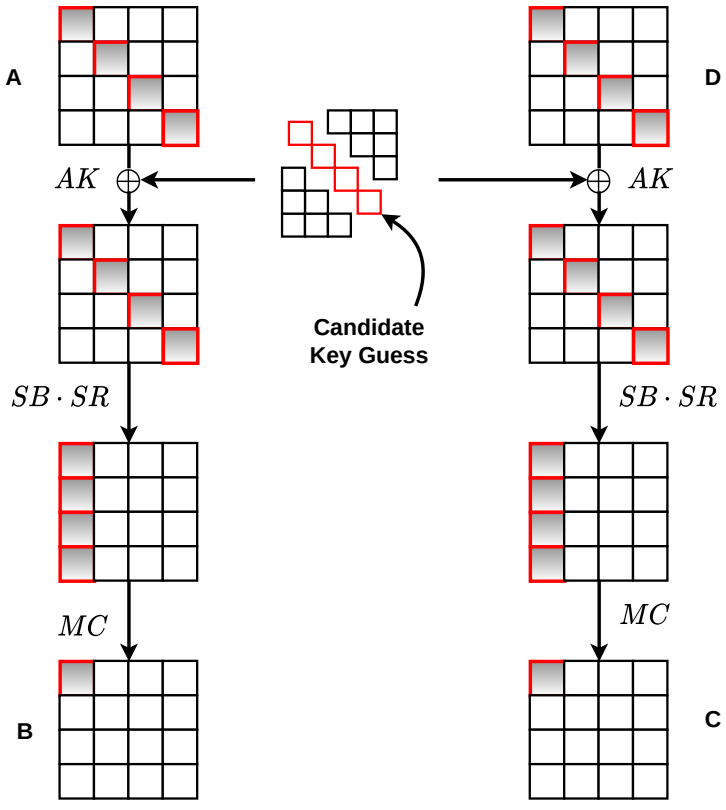
Algorithm 10 Algorithm for Key Recovery Attack on 6-round AES

Output: The secret key

procedure
  $v \leftarrow 1110$
  for $0 \leq m \leq 1$ initialize $K_m = \phi$ **do**
    for $0 \leq i < 2^{32}$ **do**
      $ctr[i] \leftarrow 0$
    end for
    for $0 \leq i < 2^{77.72}$ **do**
      Choose 2 AES state P_i^1, P_i^2 such that only the 4-bytes in $\mathcal{D}_{\{m\}}(P_i^1 \oplus P_i^2)$ are active
      $C_i^1 = Enc\Big(P_i^1\Big)$, $C_i^2 = Enc\Big(P_i^2\Big)$
      $C_i^3 = C_i^1 \oplus \tau^v\Big(C_i^1 \oplus C_i^2\Big)$ and $C_i^4 = C_i^2 \oplus \tau^v\Big(C_i^1 \oplus C_i^2\Big)$
      $P_i^3 = Dec\Big(C_i^3\Big)$ and $P_i^4 = Dec\Big(C_i^4\Big)$
      if $\mathcal{D}_{\{m\}}(P_i^3 \oplus P_i^4)$ is inactive **then**
        Discard P_i^1, P_i^2, P_i^3, P_i^4
        Go to step 8
      end if
      if $\nexists j \in \{0,1,2,3\}$ and $j \neq m$ such that $\mathcal{D}_{\{j\}}(P_i^3 \oplus P_i^4)$ is inactive **then**
        Discard P_i^1, P_i^2, P_i^3, P_i^4
        Go to step 8
      end if
      for $0 \leq j < 2^{32}$ **do**
        $X \leftarrow MC^m \circ s^4\Big(\mathcal{D}_{\{m\}}(P_i^1) \oplus j\Big) \oplus MC^m \circ s^4\Big(\mathcal{D}_{\{m\}}(P_i^2) \oplus j\Big)$
        $Y \leftarrow MC^m \circ s^4\Big(\mathcal{D}_{\{m\}}(P_i^3) \oplus j\Big) \oplus MC^m \circ s^4\Big(\mathcal{D}_{\{m\}}(P_i^4) \oplus j\Big)$
        if there is only one active byte in X and Y and its position is same in both X and Y **then**
          $ctr[j] \leftarrow ctr[j] + 1$
        end if
      end for
    end for
    Include the first 2^7 key candidates with highest counter value in K_m
  end for
  K_2 and K_3 are populated with all 2^{32} candidates
  Exhaustively search for the right subkey in $K_0 \times K_1 \times K_2 \times K_3$
  Finds the secret key from the subkey
end procedure



(a) Candidate pair for Boomeyong Attack on 6-round AES



(b) Candidate Key for Boomeyong Attack on 6-round AES

Figure 6-8: Key Recovery Attack on 6-round AES

property, under which it is expected that there is one inactive Super-Sbox between the intermediate state before one round decryption (Position **C** in Fig. 6-8a). Now in **C**, out of 3 active bytes, one byte becomes inactive with probability $3 \times 2^{-8} = 2^{-6.4}$ (The inactive diagonal in **D** should not be the same as the active diagonal in **A**, otherwise the number of candidate keys increases significantly. So, the corresponding byte in **C** should be active). The transition **A**$\rightarrow$ **B** occurs with probability $4 \times 2^{-24} = 2^{-22}$. Hence, the cumulative probability of obtaining an inactive diagonal is $2^{-22} \times 2^{-46} \times 2^{-6.4}$=$2^{-74.4}$. For a random pair, a pair of texts with such an inactive diagonal can be obtained with probability $3 \times 2^{-32} = 2^{30.4}$. Therefore, by this attack, a set of right and wrong pairs can be obtained and there is no way to distinguish the right ones from the wrong ones. If $2^{74.4}$ pairs are queried then it is expected that there are around $2^{74.4} \times 2^{-30.4} = 2^{44}$ wrong pairs and one right pair. The right key candidate is suggested by the right pair whereas the wrong pairs can suggest both right and wrong key candidates. The diagonal of the key corresponding to the active diagonal of the initial plaintext pairs are guessed (refer to Fig. 6-8b). So, the size of the guessed key space is 32 bits and thus a counter for each of the 2^{32} keys is maintained to count the key suggestions. To determine the required number of right pairs, the notion of the signal-to-noise ratio is applied.

Determining the required number of right pairs. With reference to Section 2.7.2, the values of p and k are $2^{-74.4}$ and 32 respectively. Now, the number of keys (right and wrong) suggested by each wrong pair needs to be determined. Consider P^1, P^2 be a pair of texts which are encrypted, diagonals are swapped between their corresponding ciphertexts and decrypted to obtain P^3, P^4. Let the first diagonal of the key be guessed. So, the first diagonal of P^1, P^2 is partially encrypted for one round using the guessed key and checked whether *4-to-1* transition occurred or not. Similar experiment is done with P^3, P^4. If *4-to-1* occurs for both the cases, then the value of the counter corresponding to the key is incremented. After *4-to-1* the position of the active byte should be same for both cases. Hence, for a fixed wrong pair and a fixed guessed key, the counter value is incremented with probability $4 \times 2^{-24} \times 2^{-24} = 2^{-46}$. So, the average number of keys suggested by a wrong pair is $2^{-46} \times 2^{32} = 2^{-14}$

Table 6.3: Required number of plaintext-ciphertext pairs versus the success probability for key recovery attack on 6-round AES. The value of r is considered 2^7 for all the cases.

Pairs Required	Success Probability
$2^{76.95}$	0.65
$2^{77.14}$	0.7
$2^{77.31}$	0.75
$2^{77.51}$	0.8
$2^{77.72}$	0.85
$2^{77.96}$	0.9

($\eta = 2^{-14}$). Note that, if the first diagonal of P^3, P^4 is inactive, then the pair needs to be discarded as this pair suggests $4 \times 2^{-24} \times 2^{32} = 2^{10}$ keys and to recover the correct key the data complexity may need to be increased. Hence, with probability $3 \times 2^{-32} = 2^{-30.4}$ a wrong pair survives. Therefore, $S/N = \frac{2^{32} \times 2^{-74.4}}{(1-2^{-74}) \times 2^{-30.4} \times 2^{-14}} \approx 2^2$. Plugging in the values of r as 2^7 in Proposition 5, the number of plaintexts-ciphertexts pairs required to recover a diagonal of the correct key for various success probabilities are listed in Table 6.3. From Table 6.3, the number of plaintexts-ciphertexts pairs required for key recovery with success probability 0.85 is $2^{77.72}$. As p is $2^{-74.4}$, $2^{77.72} \times 2^{-74.4} = 9.98$ right pairs are required to recover four bytes of the right key. The process is repeated one more time for another diagonal. The remaining part of the key is recovered using exhaustive search. In order to minimize the cost of the exhaustive search, the value of r is considered as 2^7. Hence, the cumulative success probability is $0.85 \times 0.85 \approx 0.72$. Details regarding key recovery attack on 6-round AES are given in Algorithm 10. Note that, in Step 21 and Step 22 in Algorithm 10, s^4 is four parallel application of subBytes on four bytes and MC^m is application of MC on m-th column.

Analysis. With reference to the Step 7, $2^{77.72}$ pairs are required to be queried to both the encryption and the decryption oracle for the first and second diagonal. Hence, the data complexity is $2 \times 2 \times 2 \times 2^{77.72} = 2^{80.72}$ encryption/decryption queries. Time

complexity involves $2^{79.72}$ XOR operations, computations of $MC \circ SB \circ AK$ operations for a single column in Step 21 and Step 22 and exhaustive search for finding the right key. After filtering, the remaining number of pairs is $2^{77.72} \times 2^{-30.4} = 2^{47.32}$. So, the total number of $MC \circ SB \circ AK$ operations is $2^{47.32} \times 4 \times 2 = 2^{50.32}$. As four such operations approximately constitute one round `AES` encryption, it is assumed that 24 such operations are equivalent to one AES (6-round) encryption. So, the total number of such operations are $2^{50.32}/24 \approx 2^{45.73}$ `AES` encryptions. In Step 31, $|K_0|=|K_1|=2^7$ and $|K_2|=|K_3| = 2^{32}$. Therefore, $2^7 \times 2^7 \times 2^{32} \times 2^{32} = 2^{78}$ offline computations of `AES` encryptions are required to recover the right key. The cost of $2^{79.72}$ XOR operations is lesser in comparison to the 2^{78} `AES` encryptions (even if 6 XOR operations are considered as one encryption of 6-round `AES`, then the $2^{79.72}$ XOR operations are equivalent to $2^{77.14}$ `AES` encryptions). Memory requirement for this attack is the memory used for storing the counter. As a byte is sufficient for storing the value for each index of the counter, 2^{32} bytes are required which is equivalent to $2^{32}/16 = 2^{28}$ `AES` states that constitutes the memory complexity.

Reducing the Encryption Queries. Refer to β in the upper trail in Fig. 6-6. Only four combinations corresponding to the position of the active byte in the last column are considered. But similar events can occur for the other columns also. Thus, instead of swapping only the last inverse diagonal, if all the inverse diagonals are swapped then it is possible to reduce the number of encryption queries by $\frac{3}{4}$; as for each pair of initial plaintexts, four different pairs of ciphertexts after swapping can be constructed. Thus the number of encryption queries can be reduced. The number of decryption queries can not be decreased as all the swapped pairs need to be queried to the decryption oracle. The number of encryption queries can be further reduced by using the structure technique. Hence the modified data complexity is approximately $2^{79.72}$.

One may be tempted to think that instead of repeating the algorithm for two diagonals independently, reusing the set of plaintext-ciphertext pairs that suggest the top key candidates to recover the second diagonal of the key may lead to a significant reduction in the number of wrong pairs while keeping the number of right pairs

the same. However, our investigation suggests that in the above modified strategy the number of right pairs corresponding to the second diagonal also reduces. It happens because the pairs whose second diagonal is inactive need to be discarded while recovering the second diagonal of the key. This claim about the ineffectiveness of the above mentioned strategy has also been supported by our experimental results.

Moreover, it can be noted that the key recovery attacks on 5/6-round AES-128 can be extended to mount key recovery attacks on 6/7-round AES-256 respectively. The details of those attacks are provided in Section 6.4.

6.2.3 Experimental Verification on 64-bit AES

To show the validity of the attacks presented in this chapter, experimental verification of the attacks are carried out on a small-scale variant of AES proposed by Cid *et al*. [82]. The variant that is considered has a block length of 64 bits and thus referred here as 64-bit AES. The bytes in the original AES are replaced with nibbles (4 bits). The round operations - SubBytes, ShiftRows, MixColumns and AddRoundKey are redefined to comply with the 64-bit version. As the design of 64-bit AES is quite similar to the original version, the analysis on AES presented in this chapter applies to it. Thus it provides a framework for verifying the validity of the attacks.

Distinguishing Attack on 5-round 64-bit AES

Recall the attack in Section 6.2.1. In this case, the modified probability of the occurrence of β is $4 \times 2^{-24} = 2^{-22}$. Hence, by checking 2^{22} pairs of plaintexts the validity of the attack can be established. Hence, a structure with 2^{11} plaintexts are constructed such that only the bytes in principal diagonal differ; the remaining bytes are the same for all plaintexts. Using these states, the experiment for the 5-round attack is carried out on 64-bit AES. As expected, a pair of states with the same zero difference pattern as the initial pair of states is obtained. The code for the 5-round attack on 64-bit AES is available online[1].

[1] https://github.com/de-ci-phe-red-LABS/Boomeyong-codes-ToSC_2021_3

Key Recovery Attack on 6-round 64-bit AES

To validate the theoretical claims, experiments have been conducted on the 6-round 64-bit AES [82] where key recovery attacks could successfully recover a diagonal. Here, we detail the experimental results of the attack. One can recall from Section 6.2.2 that swapping an inverse diagonal between ciphertexts is equivalent to swapping an inverse diagonal between the intermediate states in the previous round with probability $6 \times 2^{-24} = 2^{-22}$. For the rest of the discussion refer to Fig. 6-8a. For 64-bit AES it can be seen that the transition from $\mathbf{A} \rightarrow \mathbf{B}$ occurs with probability $4 \times 2^{-12} = 2^{-10}$. In $\mathbf{C}$, out of three active bytes one becomes inactive with probability $3 \times 2^{-4} = 2^{-2.4}$. Hence, the total probability of the characteristic is $2^{-10} \times 2^{-22} \times 2^{-2.4} = 2^{-34.4}$. For any random pair, any one of the three diagonals become inactive with probability $3 \times 2^{-16} = 2^{-14.4}$ (this is the filtering probability). Average number of keys suggested by each wrong pair is $2^{16} \times 4 \times 2^{-12} \times 2^{-12} = 2^{-6}$. Hence, $S/N = \frac{2^{16} \times 2^{-34.4}}{(1-2^{-34.4}) \times 2^{-14.4} \times 2^{-6}} \approx 2^2$. With reference to Proposition 5, if the values of r and P_s are set to 2^7 and 0.75 respectively, then the number of plaintext-ciphertext pairs required to recover the correct key is $2^{37.4}$ (8 right pairs are required). After the filtering, expected number of pairs (both right and wrong) is $2^{37.4} \times 2^{-14.4} = 2^{23}$. As described in Section 2.7.2, the counter values corresponding to each key follows the normal distribution. Hence, the counter with the highest value may not be the right key (if the counter values would have followed uniform distribution, then the candidate key having the highest counter value could have been considered as the right key).

The experiment is initiated by randomly choosing a 64-bit key. The experiment is conducted to recover the nibbles corresponding to the first diagonal of the key. As expected, after the filtering 6749861 pairs ($\approx 2^{22.69}$) survive. After the experiment, the counter value corresponding to the first diagonal is 15; whereas the highest value for the counter is 17. The counter value corresponding to the right key is among the top 128 values (number of key candidates corresponding to the counter value 17, 16 and 15 are 4, 2 and 10 respectively).

To further validate the success probability of the proposed attack, the partial key

recovery corresponding to a diagonal has been repeated 55 times. Out of which, 43 times the diagonal corresponding to the right key rank among the top 2^7 candidates. Hence, the practical success probability of the attack is $43/55 = 0.78$ which is close to the theoretical value of 0.75.

6.3 Boomeyong Attack on Pholkos

Next, the boomeyong technique is applied on a tweakable block cipher `Pholkos` [70]. Attack strategy quite similar to the 6-round attack on `AES` is used to mount an key recovery attack on 10-round `Pholkos` with the data, time and memory complexity of $2^{189.8}$, $2^{188.8}$ and 2^{122}. Till now, there is a distinguishing attack on 10-round `Pholkos` block cipher by the designers whose data, time and memory complexity is 2^{260}, 2^{260} and 2^{32} respectively.

6.3.1 Specification of `Pholkos`

Pholkos is a recently proposed family of tweakable block cipher which is based on `AES` round functions. It follows the design strategy of `AESQ` [56] and Haraka [144]. An instance of `Pholkos` with a block size of n bits and a key size of k bits is denoted by `Pholkos`-n-k. The tweak size in `Pholkos` is 128 bits for all variants. The secret key variants of `Pholkos` are `Pholkos`-256-256, `Pholkos`-512-256 and `Pholkos`-512-512. n-bit `Pholkos` state is considered as $n/128$ parallel `AES` substates where each substate goes through 2 rounds of `AES` operations followed by a columnwise permutation of words between substates. The substates are indexed from 0 to $\frac{n}{128}-1$ with the leftmost substate indexed as 0. The `AddRoundKey (AK)` operation in `AES` is substituted by `AddRoundTweakey (ATK)` in `Pholkos`. Like `AES`, `MC` is also omitted in the last round of `Pholkos`. The total number of rounds in `Pholkos` variants with a block size of 256 and 512 are 16 and 20 respectively. The details regarding key expansion and tweakey generation is omitted here; for more details refer to [70]. The notations are reviewed here.

- $P_i[j]$: Denotes the j-th substate in the i^{th} round of `Pholkos` state P.
- $\mathcal{X}_i^P$: Denotes the state before MC in the i^{th} round for an initial state P.
- $\mathcal{X}_i^P[j]$: Denotes the j^{th} substate before MC in the i^{th} round for an initial state P.

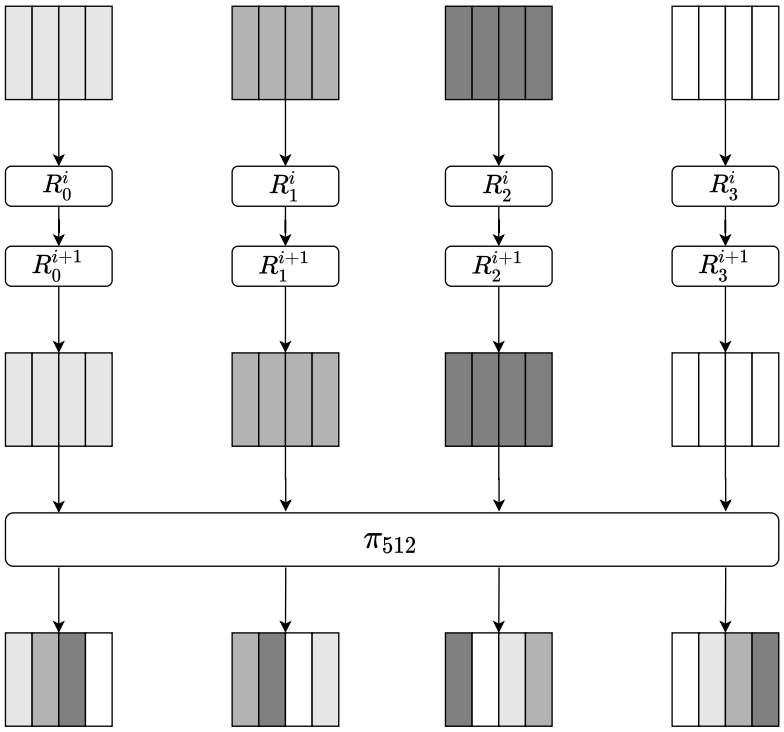


Figure 6-9: Two Rounds of `Pholkos`-512

As the attacks discussed here are independent of the key size, an instance of `Pholkos` *with block size* b *is denoted by* `Pholkos`-b. Fig. 6-9 shows round operations for `Pholkos`-512 and `Pholkos`-256. In `Pholkos`, there is a group of 128 bits which is independent of other bits in the `Pholkos` state over a certain number of rounds. This is called `MegaSbox` (cf. [86]) and details regarding this are now discussed.

`MegaSbox`. Refer to Fig. 6-10 for the `MegaSbox` construction in `Pholkos`. Four diagonals in four `AES` substates are aligned to a column in each substate due to the effect of R_j^{i-1} for $0 \leq j \leq 3$ in $i-1$ round. The subsequent π_{512} combines these columns in a single substate where they go through two rounds of `AES`. The following π_{512} breaks the substate by moving the columns to different substates and $SR \circ SB$ aligns the

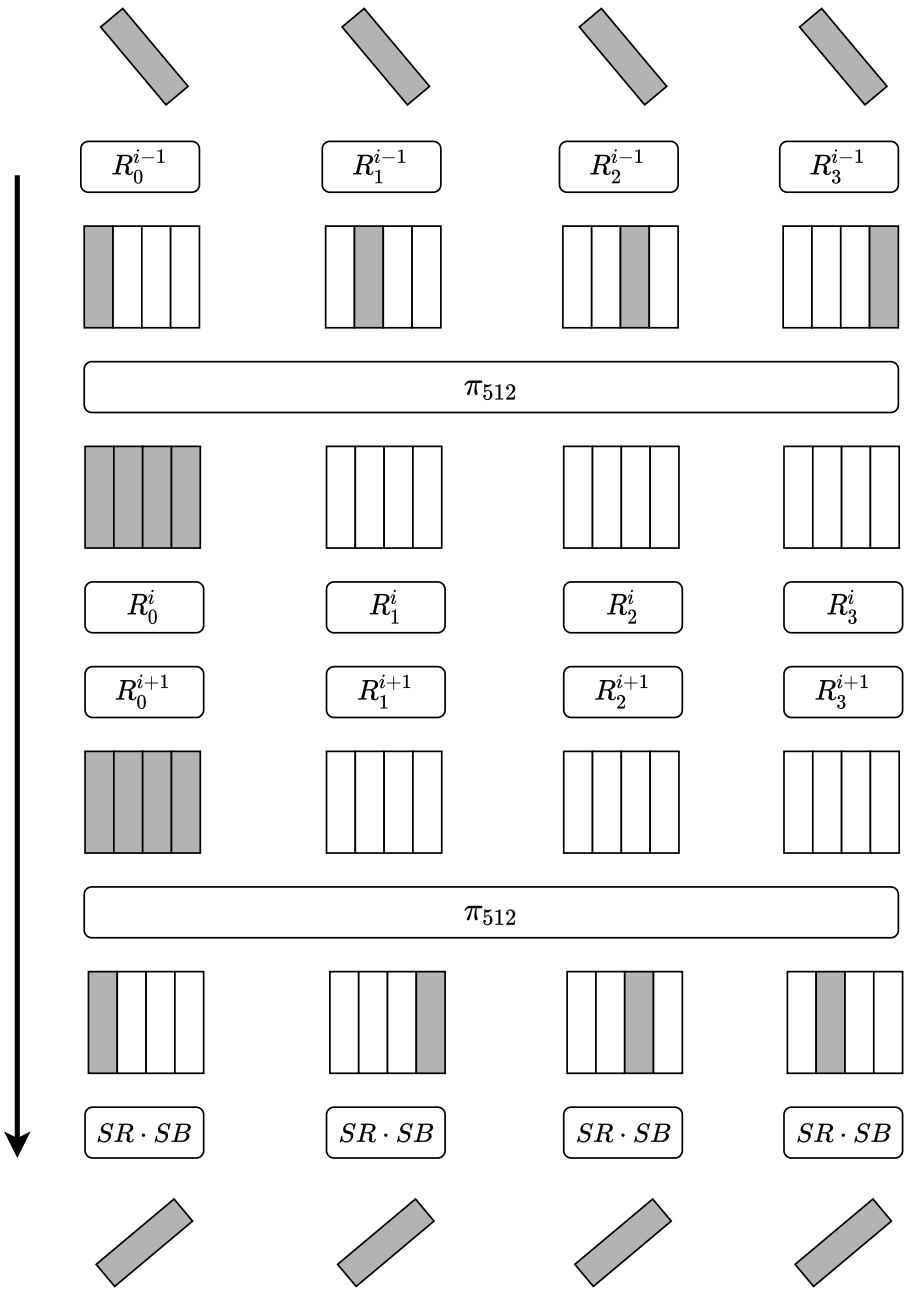


Figure 6-10: `MegaSbox` in `Pholkos`-512

bytes in inverse diagonals. The `MegaSbox` in `Pholkos`-512 spans over 3.5 rounds. 3.5 rounds `Pholkos`-512 can be considered as four parallel operations of `MegaSbox`. This `MegaSbox` is exploited while mounting the key recovery attack on `Pholkos`.

6.3.2 Key Recovery Attack on 10-round Pholkos

The key recovery attack on 10-round Pholkos is similar to the 6-round key recovery attack on AES. For the upper trail, the $S \circ L \circ S$ layer needs to be identified. Here, S and L refers to the MegaSbox and MC respectively. As MegaSbox spans over 3.5 rounds, $S \circ L \circ S$ layer starting from round 2 covers 7.5 rounds in total. The strategy remains the same- at the end of 10-round, such a δ to be added so that the inverse diagonals are swapped between the intermediate states in the previous round. Contrary to AES, here four different inverse diagonals in four substates need to be swapped and they should be a part of the same MegaSbox. Suppose, P^1, P^2 be two Pholkos states which are encrypted to obtain C^1, C^2 respectively. By Lemma 2, swapping of $\mathcal{ID}_{\{3\}}$ between $C^1[3]$ and $C^2[3]$ is equivalent to swapping of $\mathcal{ID}_J$ for $J \subset_\phi \{0,1,2,3\}$, between $\mathcal{X}_9^{P^1}[3]$ and $\mathcal{X}_9^{P^2}[3]$ with probability 2^{-46} (approx). If all other inverse diagonals corresponding to a MegaSbox in the remaining substates are inactive in the difference $\mathcal{X}_9^{P^1} \oplus \mathcal{X}_9^{P^2}$, then swapping of $\mathcal{ID}_{\{3\}}$ between $C^1[3]$ and $C^2[3]$ is equivalent to swapping of MegaSbox in $\mathcal{X}_9^{P^1} \oplus \mathcal{X}_9^{P^2}$ with probability $2^{-46} \times 2^{-32\times 3} = 2^{-142}$. Thus for the lower trail of the boomerang, δ is constructed by taking $\mathcal{ID}_{\{3\}}$ from the last substate of $C^1 \oplus C^2$ and setting all other bytes to zero. Then, with probability 2^{-142} it is known that swapping of MegaSbox has occurred in the middle.

Attack Idea. Choose a pair of plaintext P^1, P^2 such that only the four bytes in $\mathcal{D}_{\{0\}}(P^1[0] \oplus P^2[0])$ are active. P^1, P^2 are queried to the encryption oracle to obtain C^1, C^2. After one round of partial encryption only one byte becomes active with probability 2^{-22} (i. e. in $P_1^1[0] \oplus P_1^2[0]$ only one byte is active) which implies that only one MegaSbox is active. Now, a inverse diagonal is swapped between C^1, C^2 and the new states are queried to the decryption oracle to obtain P^3, P^4. Now, with probability 2^{-142} only one MegaSbox should be active in $P_1^3[0] \oplus P_1^4[0]$. t bytes out of the 16 bytes of the active MegaSbox are inactive with probability $\binom{16}{t} \times 2^{-8t}$. Hence, with probability $2^{-22} \times 2^{-142} \times \binom{16}{t} \times 2^{-8t} = 2^{-164-8t} \times \binom{16}{t}$, t diagonals are inactive in $P^3 \oplus P^4$. For a random pair of texts, t diagonals are inactive with probability $\binom{16}{t} \times 2^{-32t}$. Note that, for $7 \leq t \leq 16$, $2^{-164-8t} \times \binom{16}{t} > \binom{16}{t} \times 2^{-32t}$ and thus a

Algorithm 11 Algorithm for Key Recovery Attack on 10-round `Pholkos`

Output: The secret key

procedure
  $v \leftarrow 1110$
  Initialize a pholkos state δ by setting all bytes to 0
  for $0 \leq m \leq 2$ **do**
    Initialize $K_m = \phi$
    for $0 \leq i < 2^{128}$ **do**
      $ctr[i] \leftarrow 0$
    end for
    for $0 \leq i < 2^{186.2}$ **do**
      Choose 2 pholkos state $P^{1,i}$, $P^{2,i}$ such that only the 4-bytes in $\mathcal{D}_{\{m\}}(P^{1,i}[0] \oplus P^{2,i}[0])$ are active
      $C^{1,i} = Enc\Big(P^{1,i}\Big)$, $C^{2,i} = Enc\Big(P^{2,i}\Big)$
      $\delta[3] = \tau^v\Big(C^{1,i}[3] \oplus C^{2,i}[3]\Big)$
      $C^{3,i} = C^{1,i} \oplus \delta$
      $C^{4,i} = C^{2,i} \oplus \delta$
      $P^{3,i} = Dec\Big(C^{3,i}\Big)$ and $P^{4,i} = Dec\Big(C^{4,i}\Big)$
      $DiaCount \leftarrow 0$
      for $0 \leq j \leq 3$ **do**
        for $0 \leq k \leq 3$ **do**
          if $\mathcal{D}_{\{k\}}(P^{3,i}[j] \oplus P^{4,i}[j])$ is inactive **then**
            $DiaCount \leftarrow DiaCount + 1$
          end if
        end for
      end for
      if $DiaCount < 4$ **then**
        Discard $P^{1,i}$, $P^{2,i}$, $P^{3,i}$, $P^{4,i}$
        Go to Step 10
      end if
      for $0 \leq j < 2^{128}$ **do**
        Use the 128-bit key to partially encrypt m-th diagonal of the 4 substates
        if If there is one active byte corresponding to each of the four diagonals **then**
          $ctr[j] \leftarrow ctr[j] + 1$
        end if
      end for
    end for
    Include the top 2^7 key candidates with highest counter value in K_m.
  end for
  K_3 is populated with all 2^{128} candidates
  Exhaustively search for the right subkey in $K_0 \times K_1 \times K_2 \times K_3$
  Finds the secret key from the subkey
end procedure

right pair can be uniquely distinguished; but it requires a data complexity around $2^{206.5}$. To reduce the data complexity, instead of using a unique right pair, a set of right and wrong pairs are used and then by using the ranking test the right key candidate is guessed. Note that, as t diagonals are inactive in $P^3 \oplus P^4$, a wrong pair survives the filtering with probability $\binom{16}{t} \times 2^{-32t}$. Now, 128 bits of the key are guessed corresponding to four active diagonals in $P^3 \oplus P^4$. For a wrong pair, out of 2^{32} key guesses for each diagonal, $2^{32} \times 2^{-22} = 2^{10}$ guesses conforms to the *4-to-1* transition. So, a wrong pair suggests 2^{40} key guesses. Therefore,

$$S/N = \frac{2^{128} \times 2^{-164-8t} \times \binom{16}{t}}{\binom{16}{t} \times 2^{-32t} \times 2^{40}} = 2^{-76+24t}$$

Now, $S/N > 1$ when $t \geq 4$. As the probability of the trail is $2^{-164-8t} \times \binom{16}{t}$, so with the increasing value of t, the trail probability decreases significantly. Hence, $t = 4$ is considered. For $t = 4$, the trail probability is $2^{-185.2}$, $S/N = 2^{20}$. With reference to Proposition 5, if $r = 2^7$ is considered, then the success probability is 0.92 if $2^{186.2}$ plaintext-ciphertext pairs are used for recovering a diagonal of the key. As in Algorithm 11, this step is repeated three times, so the overall success probability of recovering the correct key is 0.78. Therefore, collecting two right pairs is enough for guessing the right key. As 128 bits of the key are guessed at a time, the size of the counter is 2^{128}. Algorithm 11 gives the details of the key recovery mechanism.

Analysis. Referring to Step 9 in Algorithm 11, $2^{186.2}$ pairs are required for encryption and decryption queries in each iteration. So, total data complexity is $3 \times 2^{186.2} \times 4 = 2^{189.8}$ encryption/decryption queries. Time complexity involves $3 \times 2^{186.2} \times 2 = 2^{188.8}$ XOR operations, computations of partial encryptions and cost of exhaustive search in Step 38. Out of $2^{186.2}$ pairs, after the filtering $2^{58.2}$ pairs remain. Each pair suggests around 2^{40} candidate keys. So, for $2^{58.2}$ pairs, $2 \times 2^{40} \times 2^{58.2} = 2^{99.2}$ partial encryptions are computed. As this process is repeated for three different sets of diagonals, total number of partial encryptions is $2^{100.8}$. By assuming four such partial encryptions as one round of Pholkos encryption, the total computation is $2^{100.8}/40 = 2^{95.5}$ Pholkos encryptions. As $|K_0| = |K_1| = |K_2| = 2^7$ and $|K_3| = 2^{128}$, Step 38

requires computations of 2^{149} Pholkos encryptions. Memory requirement for this attack is the memory used for storing the counter. As a byte is sufficient for storing the value for each index of the counter, 2^{128} bytes are required which is equivalent to $2^{128}/64 = 2^{122}$ Pholkos states and that constitutes the memory complexity.

6.4 Attacks on AES-256

The key recovery attacks on 5-round and 6-round AES-128 can be extended to mount attacks on 6-round and 7-round AES-256. This variant of AES is composed of 14 rounds and 15 subkeys are used where the first two subkeys are part of the master key. The remaining keys are derived from the master key using a key scheduling algorithm. Let K_0 and K_1 denote the first two subkeys. The attack idea is that if K_0 is correctly guessed then K_1 for 6/7-round AES-256 can be recovered by following strategies similar to the one proposed in this work for AES-128.

To mount an attack on 6/7-round AES-256, first K_0 needs to be guessed. Then intermediate states similar to the ones used for attacking AES-128 are constructed. These states are inverted one round by using the guessed value of key K_0. These inverted states form the input of AES-256, which are then queried to the encryption oracle. The states that are obtained from the decryption oracle are encrypted one round using the same guessed value of K_0 to obtain the intermediate states. It can be noted that for 6/7-round AES-256, these intermediate states along with the initially constructed ones reduce the attack to recover K_1 to a setting analogous to 5/6-round AES-128 respectively (as described in Section 6.2).

The attack depends on the guess of the key K_0. Brute-force attack is applied to recover the 128-bit key K_0 and thus 2^{128} try-outs are required. Hence, the data complexity is 2^{128} encryption queries and the time complexity is 2^{128} times of the corresponding values of the attacks on AES-128. However, the memory complexity remains the same as two independent key guesses has no effect on one another, i. e., the data obtained for one key guess have no use for another key guess. Therefore, the data, time and memory complexity of the key recovery attack on 6-round AES-256 are

$2^{49} \times 2^{128} = 2^{177}$ adaptive chosen ciphertexts, $2^{48} \times 2^{128} = 2^{176}$ XOR operations and 2^{23} AES states respectively. The corresponding complexities of the attack on 7-round AES-256 are $2^{79.72} \times 2^{128} = 2^{207.72}$ adaptive chosen ciphertexts, $2^{78} \times 2^{128} = 2^{208}$ AES encryptions and 2^{28} AES states respectively.

Next, a brief discussion about the close relationship between attacks presented in this chapter and recently proposed retracing boomerang framework [99] is given.

6.5 Relation with Retracing Boomerang Attack

Dunkelman *et al*. recently proposed the retracing boomerang attack [99] in Eurocrypt 2020. With some restrictions, those attacks can also be visualised using the boomeyong attack framework. Before discussing further, a brief description of retracing boomerang is provided.

In retracing boomerang attack, a cipher is divided into $E_{12} \circ E_{11} \circ E_0$. The E_{12} is further divided into two parts: E_{12}^L and E_{12}^R where E_{12}^L operates on b bits on the left and E_{12}^R operates on the remaining $(n-b)$ bits on the right. Let consider $Pr[\alpha \xrightarrow{E_0} \beta] = p$, $Pr[\gamma \xrightarrow{E_{11}} (\mu_L, \mu_R)] = q_1$, $Pr[\mu_L \xrightarrow{E_{12}^L} \delta_L] = q_2^L$ and $Pr[\mu_R \xrightarrow{E_{12}^R} \delta_R] = q_2^R$. In general, the probability of a boomerang distinguisher satisfying these trails is $(pq_1q_2^Rq_2^L)^2$. There are two variants of retracing boomerang attack- shifting retracing boomerang attack and mixing retracing boomerang attack. Suppose, E encrypts P^1, P^2 to C^1, C^2. In the shifting retracing boomerang attack, if $C_R^1 \oplus C_R^2 = 0$ or δ_R, only then a new pair of ciphertexts are formed by performing $C^1 \oplus \delta$ and $C^2 \oplus \delta$. This increases the probability of the boomerang distinguisher to $(pq_1q_2^L)^2q_2^R$ as in the return path the differential over E_{12}^R is deterministically satisfied. In the mixing variant, δ is constructed as $(0, C_R^1 \oplus C_R^2)$. This improves the probability of boomerang distinguisher by a factor of $(q_2^L)^{-2}(q_2^R)^{-1}$.

In particular, the mixing variant can be redefined using the boomeyong framework. Consider the last $AK \circ SR \circ SB \circ AK \circ MC$ operations of 5-round AES on first three columns as E_{12}^R and on the last column as E_{12}^L. In the boomeyong attack, δ is constructed by taking the difference of two ciphertexts in one inverse diagonal;

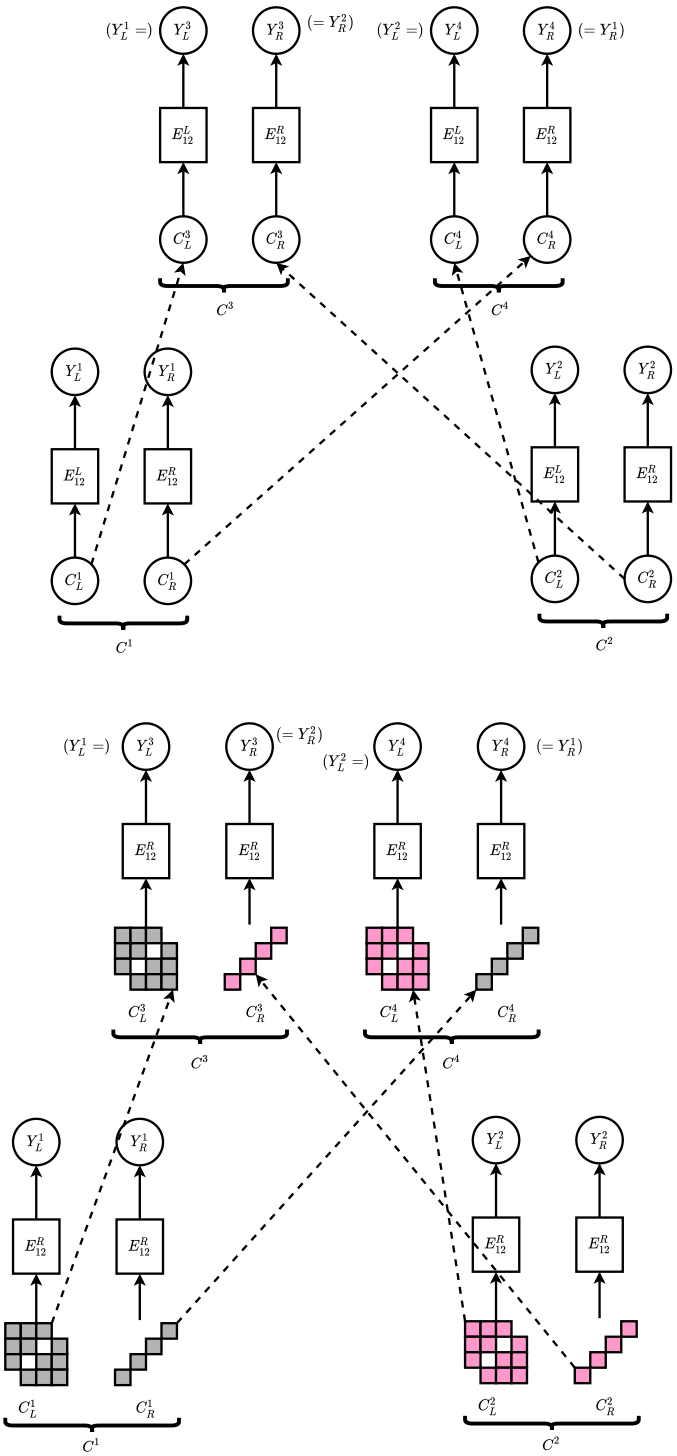


Figure 6-11: Relationship of boomeyong on `AES` with mixing retracing boomerang attack [99]. In the left, framework for mixing retracing boomerang attack is shown; whereas on the right, lower trail of the boomeyong attack on 5-round `AES` is shown. In both the attacks, a part of the state is exchanged between the ciphertexts. Here, $Y_L^i \leftarrow {E_{12}^L}^{-1}(C_L^i)$ and $Y_R^i \leftarrow {E_{12}^R}^{-1}(C_R^i)$.

the other inverse diagonals are set to zero. By following this strategy, it is possible to swap a column one round inside without incurring any probability and when the required differential in the upper trail occurs, this strategy essentially swaps inverse diagonals one round inside, which is a necessary condition for the yoyo game to occur in the upper trail. In the mixing retracing boomerang attack, δ is constructed by following a similar kind of strategy. However, the main advantage of the boomeyong attack over mixing retracing boomerang attack is that no extra cost is incurred for the return path of the upper trail as it occurs deterministically. Fig. 6-11 shows the relation between the mixing retracing boomerang attack and the boomeyong attack on AES.

6.6 Chapter Summary

In this chapter, we concentrated on devising a generic strategy for embedding the yoyo trick inside a boomerang trail. In doing so, we take a fresh look at the word-swap operation of the yoyo trick that is fundamental to the deterministic nature of the basic yoyo game. Our investigations lead to proving that the word-swap operation is a *combination* of s-box switch and ladder switch if we geometrically visualize the yoyo to be on top of the lower boomerang trail. The core idea here is to devise the lower boomerang trail in such a way that the intended s-box and ladder switches happen at the boundary thereby fulfilling the condition of the yoyo game which then leads to a deterministic transition on the way back to the top. The proposed strategy leads to new key recovery attacks on AES reduced to 5 and 6 rounds. The 5-round attack has a time complexity of 2^{48} XOR operations. The 6-round attack reaches a time complexity of 2^{78} AES encryptions. The attack is further adapted on 10 out of 20 rounds of Pholkos-512 showcasing its versatility. To the best of our knowledge, this is the first-ever third-party cryptanalysis of Pholkos. While mounting the key recovery attacks, the notion of *signal-to-noise* ratio is employed. The attacks on AES are experimentally verified by employing them on a 64-bit variant of AES (code is available online[2]). We

[2] https://github.com/de-ci-phe-red-LABS/Boomeyong-codes-ToSC_2021_3

also establish a relation of the proposed strategy with the retracing boomerang attack. It is worth mentioning that the boomeyong strategy performs better than most of the recent attacks reported on 6-round AES like extended truncated differential attack, exchange attack, yoyo attack in time/data complexity or both. Finally, the embedded yoyo-boomerang strategy helps to increase the understanding of AES and other AES-like designs and may be used as an effective cryptanalysis tool for other SPN and non-SPN ciphers as well.

7

QUANTUM ATTACKS ON SYMMETRIC DESIGNS BEYOND GROVER'S SEARCH

Contents

The polynomial-time solvability of integer factorization problem and discrete logarithm problem introduced by Shor's algorithm [187] causes a major threat to public key cryptographic primitives against quantum adversaries. In the case of symmetric key schemes, for a long time, Grover's algorithm [120] has been considered to provide the best attack by speeding up the exhaustive search of the private key by a quadratic factor. Thus, doubling the key-length resists such attacks by upgrading the quantum security of the schemes to that of the classical ones. Leveraging on the power of Simon's algorithm [189], chosen plaintext attack on 3-round Feistel [145] and the quantum attack on Even-Mansour cipher [146] by Kuwakado and Mori has opened up a new direction for cryptanalysis of symmetric key schemes in the quantum setting.

The way of realizing cryptographic protocols using quantum resources so that they can be queried using superposition queries is another aspect and in this direction, several works are carried out in which `AES` is implemented using quantum circuits

and gates to analyze the requirements of quantum resources while applying Grover's algorithm [119, 148]. In the Q_2 model, Kaplan *et al.* has shown attacks on the mode of operation for authentication and authenticated encryption by using Simon's algorithm [134]. Leander and May have shown how to combine Grover's algorithm with Simon's algorithm to mount attacks on FX construction [151]. These attacks are also based on quantum superposition queries to the quantum oracle. Bonnetain *et al.* has given attacks on several schemes without making superposition queries to the oracle [66].

Rest of the chapter is organized as follows. Initially, the method of truncating outputs of quantum oracles is described in Section 7.1. In Section 7.2, the attacks on various schemes are described. First, attacks on HCTR in both Q_1 and Q_2 model are proposed. Then attacks on $\widetilde{HCTR}$ are illustrated upon considering only Q_2 model. Finally, attacks on HCH in Q_1 and Q_2 model are discussed. Then the chapter is summarized furnishing with concluding remarks.

7.1 Output Truncation of Quantum Oracles

In the attack on 3-round Feistel cipher, Kuwakado and Morii [145] use the right half of the output from the quantum oracle to mount distinguishing attacks. In [134], it is mentioned that the outputs of 3-round Feistel oracle in the left and the right halves are entangled. So, it cannot be used as an input to the Simon's algorithm, because as described in the attack, the values of the left and the right halves need to be unentangled. In [121], it is shown how to truncate the right half of the output from the complete output when a quantum oracle is queried.

The attacks presented in this chapter are on the modes of operation of block ciphers. Essentially, a part of the ciphertexts are exploited to mount attacks. The truncation technique mentioned in [121] can be employed to take a part of the ciphertext. Let E_k encrypts $m_1||\cdots||m_s$ to $c_1||\cdots||c_s$ where m_i's, c_i's are n-bit messages and $y_1||\cdots||y_s$ are ancilla qubits. Then the corresponding quantum oracle $\mathcal{O}^k$ can be

represented as

$$\begin{aligned}\mathcal{O}^k :&|m_1\rangle \cdots |m_s\rangle |y_1\rangle \cdots |y_s\rangle \\ &\longmapsto |m_1\rangle \cdots |m_s\rangle |y_1 \oplus E^k(m_1, \cdots, m_s)\rangle \cdots |y_s \oplus E^k(m_1, \cdots, m_s)\rangle.\end{aligned}$$

Suppose, the p-th ciphertext c_p needs to be considered for further operation. Therefore, we want to simulate an

$$\mathcal{O}^k_{\{p\}} :|m_1\rangle \cdots |m_s\rangle |y_p\rangle \longmapsto |m_1\rangle \cdots |m_s\rangle |y_p \oplus E^k(m_1, \cdots, m_s)\rangle. \tag{7.1}$$

This is similar to the simulation of the oracle

$$\begin{aligned}\mathcal{O}'^k_{\{p\}} :&|m_1\rangle \cdots |m_s\rangle |y_p\rangle \overbrace{|0^n\rangle \cdots |0^n\rangle}^{(s-1) \text{ times}} \\ &\longmapsto |m_1\rangle \cdots |m_s\rangle |y_p \oplus E^k(m_1, \cdots, m_s)\rangle \overbrace{|0^n\rangle \cdots |0^n\rangle}^{(s-1) \text{ times}}.\end{aligned} \tag{7.2}$$

Let $H^{\otimes n}$ be an n-bit Hadamard gate and $|+\rangle := H^{\otimes n}(0^n)$. Considering $y_1, \cdots, y_{p-1}$, $y_{p+1}, \cdots, y_s = 0^n$ and applying Hadamard on them, the oracle representation in (7.1) can be rewritten as

$$\begin{aligned}\mathcal{O}^k :&|m_1\rangle \cdots |m_s\rangle |+\rangle \cdots |y_p\rangle \cdots |+\rangle \\ &\longmapsto |m_1\rangle \cdots |m_s\rangle |+\rangle \cdots \qquad |y_p \oplus E^k(m_1, \cdots, m_s)\rangle \cdots |+\rangle.\end{aligned}$$

Let `swap(p)` be a function that swaps $(s+1)$-th output with $(s+p)$-th. Now, the oracle $\mathcal{O}'^k_{\{p\}}$ can be defined as

$$(I_{ns+n} \otimes H^{\otimes(s-1)n}) \cdot \texttt{swap(p)} \cdot \mathcal{O}^k \cdot \texttt{swap(p)} \cdot (I_{ns+n} \otimes H^{\otimes(s-1)n}).$$

It can be verified that $\mathcal{O}'^k_{\{p\}}$ can be applied to truncate p-th ciphertext block when a quantum access to $\mathcal{O}^k$ is given. Figure 7-1 shows how $\mathcal{O}'^k_{\{p\}}$ is constructed from $\mathcal{O}^k$.

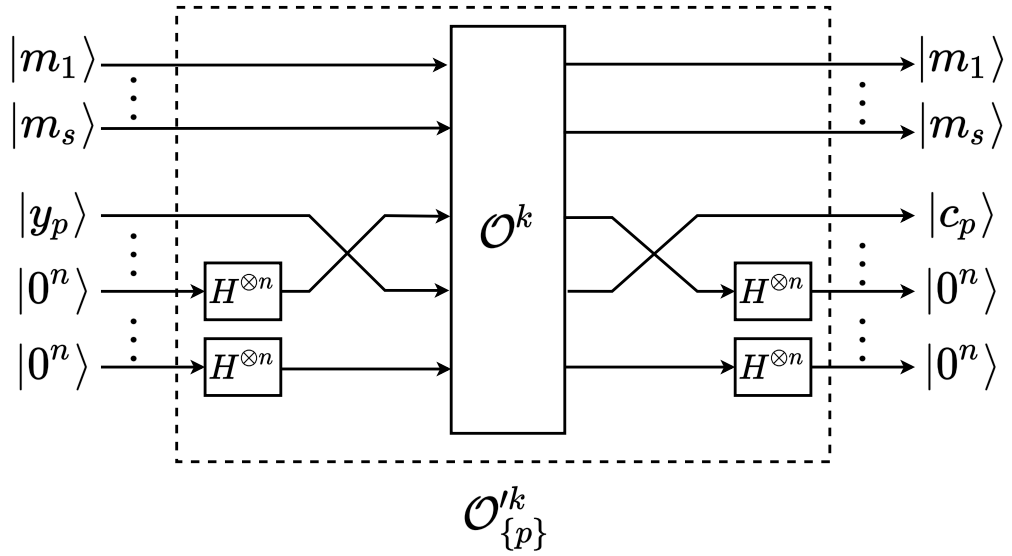


Figure 7-1: Construction of $\mathcal{O}'^k_{\{p\}}$ from $\mathcal{O}^k$

7.2 Attacks

Based on the previous theoretical explanations, it is possible to mount attacks on HCTR, Tweakable-HCTR($\widetilde{HCTR}$), and HCH constructions in Q_1 and Q_2 model. The attacks on HCTR have been discussed extensively. The remaining two attacks are quite similar to the attack on HCTR, and thus they have been briefly described. HCTR can encrypt a n-block message $(M_1||M_2||\cdots M_r)$ to produce $(C_1||C_2||\cdots C_r)$. For mounting attack, the second ciphertext block C_2 has been used. Instead of C_2, any $C_i(2 \leq i \leq r)$ can be used in order to perform the attack. Similar strategies have been followed for $\widetilde{HCTR}$ and HCH.

7.2.1 Attack on HCTR

Our first attack is on HCTR or Hash-Counter which is a tweakable enciphering scheme proposed by Wang, Feng, and Wu [206]. It is a strong tweakable pseudorandom permutation and hash-encipher-hash based construction where the middle layer uses counter mode. It is a length preserving tweakable enciphering scheme which supports input with arbitrary variable length. Fig. 7-2 shows the HCTR construction. HCTR

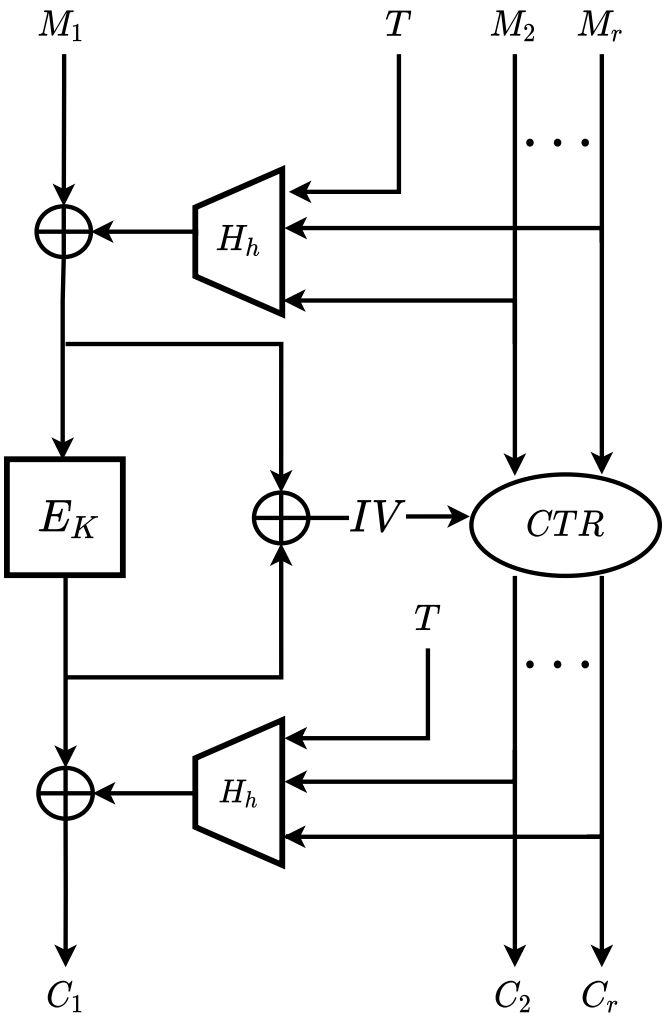


Figure 7-2: Construction of $HCTR$

uses a block cipher

$$E : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$$

and a universal hash function

$$H = \Big\{ H_h : \{0,1\}^* \to \{0,1\}^n | h \in \{0,1\}^n \Big\}.$$

Let $M_1||M_2||\cdots||M_r$ be encrypted by

$$HCTR[E,H] : \{0,1\}^{m+n} \times \{0,1\}^t \times \{0,1\}^{\geq n} \to \{0,1\}^{\geq n}$$

to obtain $C_1||C_2||\cdots||C_r$, then

$$C_1||C_2||\cdots||C_r = HCTR_K^T(M_1||M_2||\cdots||M_r),$$

where $T \in \{0,1\}^t$ is a tweak and $K \in \{0,1\}^m$ is the key of the underlying block cipher. To consider only the i-th ciphertext block, we introduce the operator Π^i. Note that, as all the blocks in ciphertexts are entangled; it is not trivial to truncate the i-th ciphertext block. In this regard, the method described in Section 7.1 can be followed for truncating a specific block of the cipher.

In the original construction, the tweak length is fixed and can be zero. In the following attacks, the tweak length is considered non-zero and each message block is n-bit. The attack is performed using two message blocks, which can be easily extended for an arbitrary number of message blocks.

Consider, HCTR is used to encrypt a message $M_1||M_2$ using a tweak t to obtain $C_1||C_2$ and K is the key of the underlying block cipher. Then,

$$\begin{aligned}
&C_1||C_2 = HCTR_K^T(M_1||M_2),\\
&IV = H_h(T||M_2) \oplus M_1 \oplus E_K\Big(H_h(T||M_2) \oplus M_1\Big),\\
&C_2 = E_K(IV \oplus 1) \oplus M_2,\\
&C_1 = E_K\Big(H_h(T||M_2)\Big) \oplus H_h(T||C_2).
\end{aligned}$$

Attack in Q_2 Model.

In Q_2 model, quantum superposition queries can be made to HCTR oracle. $x||M_2$ is queried with tweak T_0, T_1 and output C_2 is used to construct $g(x)$.

$$\begin{aligned}
g(x) &= \Pi^2\Big(HCTR_K^{T_0}(x||M_2)\Big) \oplus \Pi^2\Big(HCTR_K^{T_1}(x||M_2)\Big)\\
&= E_K\Big(H_h(T_0||M_2) \oplus x \oplus E_K\Big(H_h(T_0||M_2) \oplus x\Big) \oplus 1\Big)\\
&\quad \oplus E_K\Big(H_h(T_1||M_2) \oplus x \oplus E_K\Big(H_h(T_1||M_2) \oplus x\Big) \oplus 1\Big)
\end{aligned} \tag{7.3}$$

Clearly, $g(x)$ is a periodic function with period $H_h(T_0||M_2) \oplus H_h(T_1||M_2)$ and it can be recovered by applying Simon's algorithm on $g(x)$ by making $O(n)$ queries. Therefore,

$$g(x) = g(x \oplus H_h(T_0||M_2) \oplus H_h(T_1||M_2)).$$

So,

$$\Pi^2\Big(HCTR_K^{T_1}(x \oplus s||M_2)\Big) \oplus \Pi^2\Big(HCTR_K^{T_0}(x \oplus s||M_2)\Big) \\ \oplus \Pi^2\Big(HCTR_K^{T_1}(x||M_2)\Big) \oplus \Pi^2\Big(HCTR_K^{T_0}(x||M_2)\Big) = 0,$$

where $s = H_h(T_0||M_2) \oplus H_h(T_1||M_2)$. For a random function, Simon's algorithm outputs zero, as it is the trivial period for such cases. In case of HCTR, Simon's algorithm outputs a non-zero value which clearly distinguishes HCTR. Figure 7-3 shows how the Simon function $g(x)$ is constructed. As shown in Equation 7.3, the message block $x||M_2$ is queried to quantum $HCTR$ oracle using the tweaks T_0 and T_1 respectively; and the second ciphertext block of the corresponding outputs are XOR-ed to construct the required Simon function $g(x)$. $\Pi^2(HCTR)$ returns the second ciphertext block for the corresponding message blocks that are queried to the oracle. As discussed in Section 7.1, given a quantum oracle access to HCTR, $\Pi^2(HCTR)$ can be constructed.

Attack in Q_1 Model.

In the Q_1 model, a quantum superposition state is formed from classical oracle queries. While mounting such kind of attacks, the enciphering scheme needed to be reduced to Problem 3. $g(x)$ can be classically queried (online) to obtain $|\psi_g\rangle$ and then $f_i \oplus g$ can be tested offline whether periodic or not using Simon's and Grover's search algorithm. As mentioned in [66], instead of querying the whole classical codebook, the advantage of algebraic structures have been taken into account while mounting the attack.

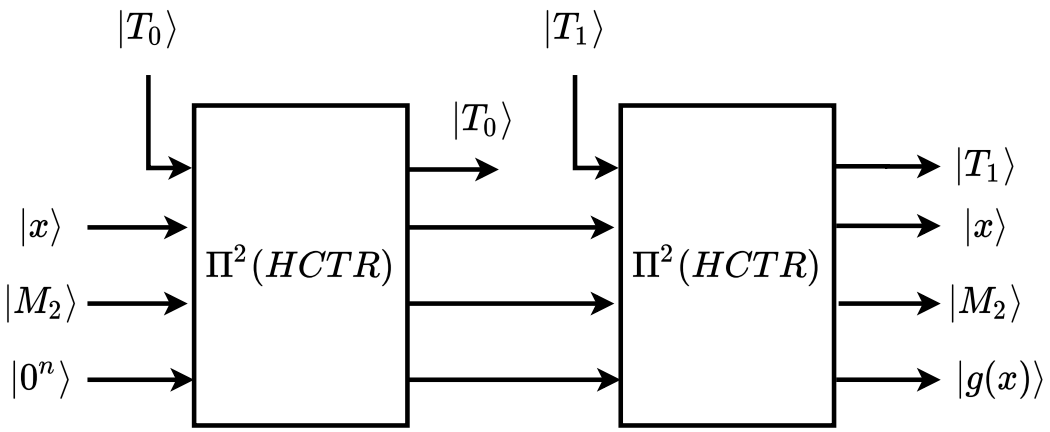


Figure 7-3: Simon function for $HCTR$. In the figure, $\Pi^2(HCTR)$ truncates the ciphertext block C_2 and it is constructed by following the approach in Section 7.1. Note that, input and output lines corresponding to C_1 are not shown.

Attack Description. Like previous attack, here also two message blocks have been considered. The last message block and last $(n-u)$ bits of first message block are kept constant. The queries to the oracle is of the form $(x||0^{n-u})||M_2$, where $x||0^{n-u}$ and M_2 are the first and second message block respectively and $0 \leq u \leq n$. For constructing a periodic function, the second ciphertext block C_2 is considered. The value of M_2 is fixed and by varying the value of x, 2^u classical queries are made to HCTR oracle to form $|\psi_g\rangle$. Define $F : \{0,1\}^{m+n-u} \times \{0,1\}^u \mapsto \{0,1\}^n$ by

$$F(i||j,x) = f_{i||j}(x) = E_i\Big(x||j \oplus E_i(x||j \oplus 1)\Big)$$

for $i \in \{0,1\}^m$, $j \in \{0,1\}^{n-u}$ and define $g : \{0,1\}^u \to \{0,1\}^n$ by

$$g(x) = \Pi^2\Big(HCTR_K^T((x||0^{n-u})||M_2)\Big).$$

Then,

$$\begin{aligned} g(x) &= \Pi^2\Big(HCTR_K^T((x||0^{n-u})||M_2)\Big) \\ &= E_K\Big(H_h(T||M_2) \oplus (x||0^{n-u}) \oplus E_K\Big(H_h(T||M_2) \oplus (x||0^{n-u})\Big) \oplus 1\Big). \end{aligned} \tag{7.4}$$

Let the first u bits of $H_h(T||M_2)$ be denoted by $l^{(1)}$ and last $n-u$ bits are denoted

by $l^{(2)}$. Then $g(x)$ can be rewritten as

$$\begin{aligned} g(x) &= E_K\Big((l^{(1)}||l^{(2)}) \oplus (x||0^{n-u}) \oplus E_K\Big((l^{(1)}||l^{(2)}) \oplus (x||0^{n-u}) \oplus 1\Big)\Big) \\ &= E_K\Big((l^{(1)} \oplus x)||l^{(2)} \oplus E_K\Big((l^{(1)} \oplus x)||l^{(2)} \oplus 1\Big)\Big). \end{aligned} \tag{7.5}$$

Consider the function $F(i||j, x) \oplus g(x)$. It has a hidden period $l^{(1)}$ for $F(K||l^{(2)}, x) \oplus g(x)$. The attack steps are listed below.

1. Alg-ExpQ1 is run for F and g to recover K and $l^{(2)}$.
2. Alg-SimQ1 is run on $f_{K||l^{(2)}} \oplus g$ to recover $l^{(1)}$.

Note that, by the above approach key of the underlying block cipher can be recovered. Although, it is unable to recover hash key h, but using $l^{(1)}$ and $l^{(2)}$, $H_h(T||M_2)$ can be constructed. The attack can be extended for an arbitrary number of message blocks.

Analysis. The analysis of the attack is similar to the analysis of the attack on Even-Mansour cipher in [66]. First, it is assumed that the size of keyspace of the underlying block cipher is in $O(n)$. In the attack, if u is kept too small, although too few queries are required to construct $|\psi_g\rangle$, the cost of Grover's search increases significantly. Under the constraints that u is not too small and E is a secure block cipher, we can assume that

$$\max_{\substack{t \in \{0,1\}^u \setminus \{0^u\}, \\ x \leftarrow \{0,1\}^u}} Pr_x[(f_{i||j} \oplus g)(x \oplus t) = (f_{i||j} \oplus g)(x)] \leq \frac{1}{2}$$

holds for $(i||j) \neq (K||l^{(2)})$. By virtue of this, Proposition 2 and Proposition 3 hold for Alg-ExpQ1 and Alg-SimQ1 respectively. Overall, the key of underlying block cipher and $l^{(1)}||l^{(2)}$ is recovered by following this attack using $D = O(2^u)$ classical queries to $HCTR_K^T$ and performing $T = O(n^3 2^{\frac{m+n-u}{2}})$ offline computations. Here, it is also assumed that one evaluation of F is in $O(1)$ which makes $T_F = O(1)$. The trade-off $DT^2 = n^3 2^{m+n}$ is applied; data and time complexity balances at $D = O(2^{\frac{m+n}{3}})$ and $T = O(n^3 2^{\frac{m+n}{3}})$. As mentioned in [66], by construction of Alg-ExpQ1 and Alg-SimQ1

our attack uses qubits in the order of polynomial and negligible classical bits. Note that, generic attacks takes $O(2^{\frac{m}{2}})$ time. So, this attack is better than generic attacks when $n^3 2^{\frac{m+n}{3}} < 2^{\frac{m}{2}} \implies m > 6\log_2(n^3 2^{\frac{n}{3}})$.

7.2.2 Attack on Tweakable-HCTR

Tweakable-HCTR or $\widetilde{HCTR}$ was proposed by Dutta and Nandi [102] which is a variant of HCTR where each block cipher call is replaced by tweakable block cipher (TBC). Another major difference between HCTR and $\widetilde{HCTR}$ is the use of tweak. In

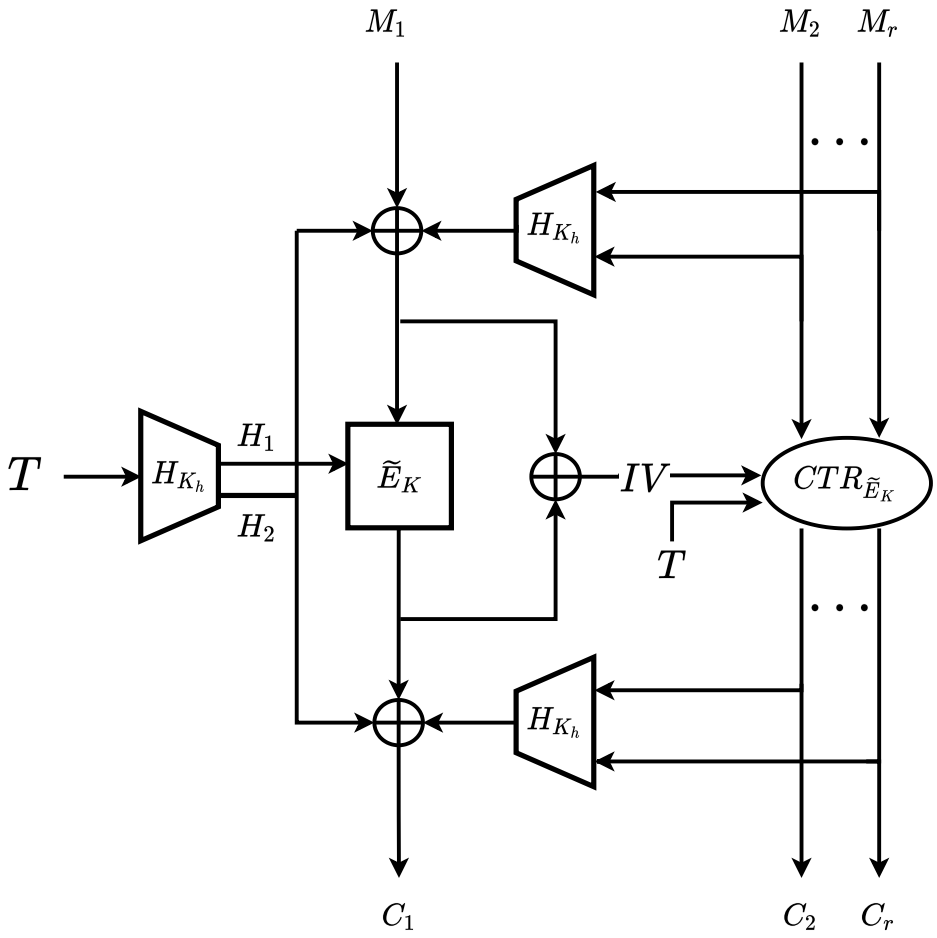


Figure 7-4: Construction of $\widetilde{HCTR}$

$\widetilde{HCTR}$ instead of using the tweak in upper and lower hash functions, it is used in an independent keyed $(n+t)$-bit hash function H'_L. The output of H'_L is divided into two parts: n-bit H_1 which is masked with the input and the output of leftmost TBC

and t-bit H_2 which acts as a tweak for the underlying TBC. Underlying TBC

$$\widetilde{E} : \{0,1\}^m \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$$

is denoted by $\widetilde{E}_K^{H_2}$ where K is m-bit key and H_2 is t-bit tweak. Let $M_1||M_2||\cdots||M_r$ is encrypted by $\widetilde{HCTR}$ to obtain $C_1||C_2||\cdots||C_r$, then

$$C_1||C_2||\cdots||C_r = \widetilde{HCTR}_K^T(M_1||M_2||\cdots||M_r),$$

where $T \in \{0,1\}^*$ is a tweak and $K \in \{0,1\}^m$ is the key of underlying block cipher. Figure 7-4 shows the construction of $\widetilde{HCTR}$.

The Q_1 and Q_2 attacks for $\widetilde{HCTR}$ are quite similar to the attacks on HCTR. For the sake of simplicity, only corresponding periodic functions are mentioned here. Consider the encryption of two n-bit message blocks. Then

$$IV = \widetilde{E}_K^{H_2}\Big(H_{K_h}(M_2) \oplus M_1 \oplus H_1\Big) \oplus H_{K_h}(M_2) \oplus M_1 \oplus H_1, \tag{7.6}$$

$$C_2 = \widetilde{E}_K^{H_2}(IV \oplus 1) \oplus M_2. \tag{7.7}$$

Attack in Q_2 Model.

Consider the function $g(x)$ constructed from second ciphertext block and $H'_L(T) = (H_1, H_2)$.

$$\begin{aligned} g(x) &= \Pi^2\Big(\widetilde{HCTR}_K^T(x||M_2)\Big) \oplus \Pi^2\Big(\widetilde{HCTR}_K^T(x||M'_2)\Big) \\ &= \widetilde{E}_K^{H_2}\Big(\widetilde{E}_K^{H_2}\Big(H_{K_h}(M_2) \oplus x \oplus H_1\Big) \oplus H_{K_h}(M_2) \oplus x \oplus H_1\Big) \\ &\oplus \widetilde{E}_K^{H_2}\Big(\widetilde{E}_K^{H_2}\Big(H_{K_h}(M'_2) \oplus x \oplus H_1\Big) \oplus H_{K_h}(M'_2) \oplus x \oplus H_1\Big) \oplus M_2 \oplus M'_2 \end{aligned} \tag{7.8}$$

7.2.3 Attack on HCH

Another variant of HCTR is HCH or Hash-Counter-Hash, proposed by Chakraborty and Sarkar which is based on hash-encrypt-hash paradigm [79]. In HCH, tweak T

is not directly used by the polynomial hash; instead it is encrypted twice to obtain R and Q which are used with the hash function H (In HCH, the hash function is denoted by $H_{R,Q}$). Figure 7-5 shows the construction of HCH. In the attacks presented, as generation of R and Q is not used, the fact that for a fixed T, R and Q remains fixed is considered. In the counter-mode, instead of IV, S is used for initialization which is obtained by encrypting the input and output of leftmost block cipher. If $M_1||M_2||\cdots||M_r$ ($|M_i| = n$) is encrypted using HCH to obtain $C_1||C_2||\cdots||C_r$ ($|C_i| = n$), then

$$C_1||C_2||\cdots||C_r = HCH_K^T(M_1||M_2||\cdots||M_r),$$

where K is the key of underlying block cipher $E : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$ denoted by $E_K(.)$ and T is the tweak. Our attack is based on the second ciphertext block, which is given as

$$C_2 = E_K(S) \oplus M_2 \tag{7.9}$$

where

$$S = E_K\Big(E_K\Big(H_{R,Q}(M_2) \oplus M_1\Big) \oplus H_{R,Q}(M_2) \oplus M_1\Big). \tag{7.10}$$

In the following attacks, only the periodic functions are mentioned as the attacks are almost same as the attacks on HCTR.

Attack in Q_2 Model.

Consider the function $g(x)$ constructed from second ciphertext block.

$$\begin{aligned} g(x) &= \Pi^2\Big(HCH_K^T(x||M_2)\Big) \oplus \Pi^2\Big(HCH_K^T(x||M_2')\Big) \\ &= E_K\Bigg(E_K\Big(E_K\Big(H_{R,Q}(M_2) \oplus x\Big) \oplus H_{R,Q}(M_2) \oplus x\Big)\Bigg) \\ &\oplus E_K\Bigg(E_K\Big(E_K\Big(H_{R,Q}(M_2') \oplus x\Big) \oplus H_{R,Q}(M_2') \oplus x\Big)\Bigg) \oplus M_2 \oplus M_2' \end{aligned} \tag{7.11}$$

Note that, $g(x \oplus H_{R,Q}(M_2) \oplus H_{R,Q}(M_2')) = g(x)$. So, period is $H_{R,Q}(M_2) \oplus H_{R,Q}(M_2')$. Applying Simon's algorithm on $g(x)$ recovers the period in $O(n)$ queries. Figure 7-6

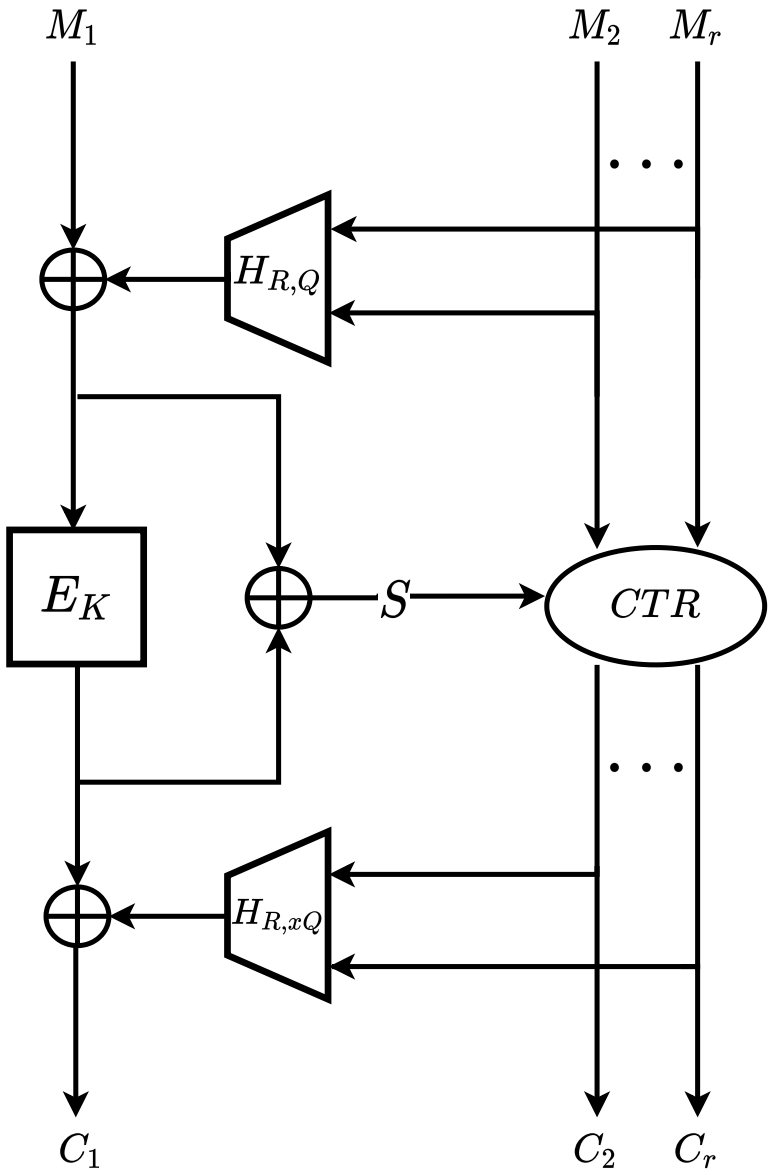


Figure 7-5: Construction of HCH

shows the process of generating $g(x)$.

Attack in Q_1 Model.

Let u be a integer and $0 \leq u \leq n$. Define $F : \{0,1\}^{m+n-u} \times \{0,1\}^u \to \{0,1\}^n$ by

$$F(i||j,x) = f_{i||j}(x) = E_i\Big(E_i\Big(E_i\big(x||j\big) \oplus x||j\Big)\Big) \tag{7.12}$$

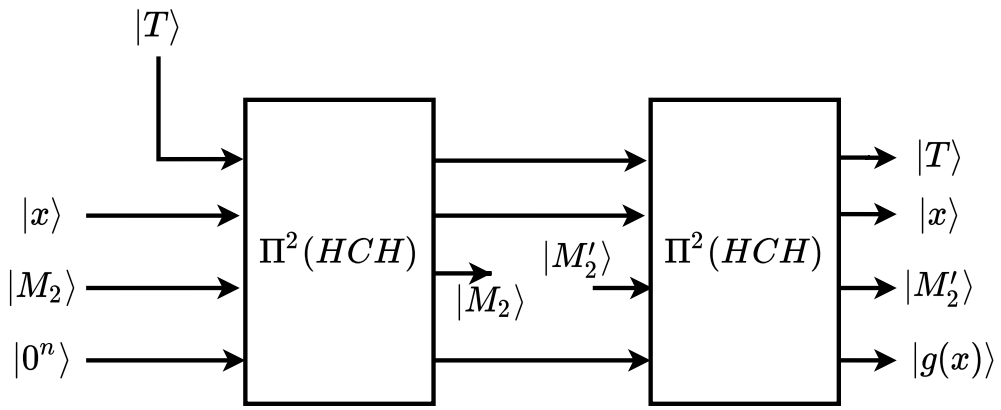


Figure 7-6: Simon function for HCH. In the figure, $\Pi^2(HCH)$ truncates the ciphertext block C_2.

and define $g : \{0,1\}^u \to \{0,1\}^n$ by

$$g(x) = \Pi^2\Big(HCH_K^T\big((x||0^{n-u})||M_2\big)\Big).$$

Then

$$g(x) = E_K\Bigg(E_K\Big(E_K\big(H_{R,Q}(M_2) \oplus (x||0^{n-u})\big) \\ \oplus H_{R,Q}(M_2) \oplus (x||0^{n-u})\Big)\Bigg) \oplus M_2.$$

Let first u bits of $H_{R,Q}(M_2)$ be $l^{(1)}$ and last $(n-u)$ bits be $l^{(2)}$. Then

$$\begin{aligned} g(x) &= E_K\Bigg(E_K\Big(E_K\big(l^{(1)}||l^{(2)} \oplus (x||0^{n-u})\big) \oplus l^{(1)}||l^{(2)} \oplus (x||0^{n-u})\Big)\Bigg) \oplus M_2 \\ &= E_K\Bigg(E_K\Big(E_K\big((l^{(1)} \oplus x)||l^{(2)}\big) \oplus (l^{(1)} \oplus x)||l^{(2)}\Big)\Bigg) \oplus M_2. \end{aligned}$$

Consider the function $f_{i||j}(x) \oplus g(x)$. The function $f_{K||l^{(2)}} \oplus g(x)$ is periodic with period $l^{(1)}$.

The analysis of this attack is similar to the analysis of the attack on HCTR and hence the details are omitted. The data and time complexity of this attack is $O(2^{\frac{m+n}{3}})$ and $O(n^3 2^{\frac{m+n}{3}})$ respectively.

7.3 Chapter Summary

In this chapter, we analyzed the HCTR, Tweakable-HCTR and HCH in the quantum adversarial model. The work presented here develops upon the previous works in [134, 66, 151]. All our attacks have made use of encryption oracle only. This arises a question of whether the availability of decryption oracle can make a significant benefit in terms of the complexity of mounting such attacks.

8 QUANTUM RESOURCE ESTIMATION

Contents

Recent progress in the area of development of viable quantum computers has threatened the security of existing cryptographic schemes. Introduction of Shor's algorithm [188, 187] has put those public key schemes at risk which are designed considering the hardness to solve the integer-factorization and discrete-logarithm problem. For symmetric-key schemes, threat due to generic attack using the Grover's algorithm [120] is considered only for a long time [211]. However, this vulnerability can be defended by doubling the key length and achieving the same security as before. Later on, the introduction of quantum attacks using Simon's algorithm [189] on even-mansour cipher [146], 3-round feistel cipher [145], encryption schemes [134], HCTR-based schemes [175] has unveiled the facade of resistance of such schemes against quantum attacks. Based on computation power, quantum adversarial models are also developed [214]. With ever-increasing chance of compromise of the security of cryptographic schemes with the progress towards the development of quantum computers and the difficulty of the transition to quantum-secure cryptographic schemes,

the National Institute of Standards and Technology (NIST) put forth the proposal for standardization of post-quantum cryptographic (PQC) schemes [171].

Why Resource Estimation?

As the computing power of future quantum machines can not be predicted accurately, the security of post-quantum schemes can not be estimated with certainty based on the current scenarios. Thus NIST proposes to measure the security strengths in terms of computational resources rather than "bits of security" [171]. These computational resources can be measured in terms of the number of elementary operations, circuit size, etc. Resource estimation gives a quantitative measure of the complexity of the circuit. It provides a comparative filter between two competing design or attack implementations. As KATAN [91] is classically not broken and till now, generic quantum attack using Grover's algorithm [120] is the only way to break the cipher, resource estimation of Grover's attack on KATAN shows the efficiency of the attack in the current scenario. Recently, a lot of works explored the resource estimation of different quantum attacks on cryptographic schemes, such as resource estimation of Grover's search on symmetric-key primitives [119, 148, 15, 18, 128, 129], resource estimation for computing discrete logarithms on binary elliptic curves [26], resource estimation of pre-image attacks using Grover's search on hash functions [17], etc. In addition, study regarding the resource estimation of fault-tolerant quantum random-access memory (qRAM) are also conducted [160].

Grover's attack on symmetric-key schemes is mostly defined in terms of "bits of security". However, NIST's proposal has prompted redefining security in terms of computational resources. Earlier, Grassl *et al.* has estimated the computational resources of mounting Grover's attack on Advanced Encryption Standard (`AES`) [119]. Later on, Langenberg *et al.* [148] and Almazrooie *et al.* [15] further reduces the cost of implementing AES using quantum gates and qubits. These analyses are based on reducing the number of qubits. In Eurocrypt 2020, Jaques *et al.* studied the cost of implementing Grover's attack on `AES` by focusing on minimizing the depth of the circuit rather than the width [130]. Apart from `AES`, resource estimation for Grover's

attack is also studied for feedback shift register-based schemes [18], Speck [128] and GIFT [129].

The chapter is organized as follows. Section 8.1 gives an overview regarding the design rationale of the quantum circuits. In Section 8.2, the resource requirement for mounting Grover's attack on Katan is estimated. Section 8.3 studies the cost of mounting Grover on Present block cipher. Finally, the chapter is summarized in Section 8.4.

8.1 Design Rationale

Here, the insights behind designing the circuits in this chapter are discussed. The designs in this work consider security strength defined in NIST's proposal for post-quantum cryptography (PQC) standardization, the efficiency in implementing Grover's algorithm.

8.1.1 NIST PQC Standardization

Constructing a combinatorial circuit by optimizing the depth, the number of gates is an intractable problem. Although Boyar *et al.* have proposed heuristics-based methods to construct low-depth circuits [75], the gate cost of the circuit increases significantly when restrictions on the depth are more tightened. NIST has put a restriction on the circuit depth to match the time-boundness; however, there is no restriction on the number of qubits to be used. Owing to this factor, the quantum circuits in this work are designed to minimize the overall depth; whereas no restrictions are put in using the ancillary qubits.

8.1.2 Implementation Issues of the Grover's Algorithm

Depth Constraints In the standardization process of Post Quantum Cryptography [171], NIST introduces a parameter `MAXDEPTH` owing to the difficulty in running long serial computations. Thus the resources of a quantum adversary are bounded

by the maximum depth of a circuit. Once this `MAXDEPTH` is reached, multiple instances of Grover's algorithm are needed to run in parallel. The permissible values of `MAXDEPTH` ranges from 2^{40} to 2^{96}. Jaques *et al.* concludes that if the limits of resources in terms of depth, width and gate count are fixed, then to mount an optimal attack the depth should be fully utilized and parallelization should be minimized as much as possible [130].

Parallelization The efficiency of Grover's algorithm reduces significantly if parallelization is considered and thus parallelization inside the oracle circuit is advantageous. The study regarding the parallelization of Grover's algorithm is studied in [36, 35, 213]. Zalka reports that using S quantum machines a factor of $\sqrt{S}$ can be gained irrespective of the case of running full Grover's algorithm on S identical quantum machines or partitioning the entire search space into S parts and assigning each part to each of the S identical machines and concludes that it is more advantageous to parallelize quantum searching algorithms by dividing the search space and assigning each space to different independent quantum machines [213]. Kim *et al.* formally defines the notion of these two settings as outer and inner parallelization[138]. In outer parallelization, Grover's algorithm is run on the entire search space on S identical machines; whereas in inner parallelization the search space is divided into S parts and Grover's algorithm is run on each part on S identical machines. In both, the cases required the number of iterations is less than that of running only a single instance of Grover's algorithm, as in outer parallelization the success of recovering the right key depends on the success of any one of the S machines which reduce the required number of iterations; whereas for inner parallelization required number of iterations is reduced for a smaller search space. However, by using inner parallelization the probability of obtaining spurious keys is reduced and thus inner parallelization is more preferable to the outer one.

8.1.3 Cost Metrics.

For cost analysis of mounting Grover's attack, two cost metrics proposed in [131] are considered. Consider a quantum circuit that has a depth and width of D and W and consists of G quantum gates. The two cost metrics are G-cost metric which considers $\Theta(G)$ RAM operations and DW-cost metric which considers $\Theta(DW)$ RAM operations. It is shown by Jaques *et al.* that G-cost and DW-cost is minimized by minimizing the number of parallel machines [130].

8.1.4 Automated Resource Estimation.

In this work, the circuits for Grover oracle of Present are designed and implemented in ProjectQ [191, 122] framework. ProjectQ provides a module for automated estimation of required resources in terms of gate counts, circuit depth and width. In general, it is considered that for fault-tolerant quantum computation, clifford + T gate set forms a good universal gate [77]. Thus the circuits are designed using clifford + T gate set only. Logical T gates are considered more expensive than the clifford gates [108] and thus along with normal depth, T-depth is also considered as a viable cost function. T-depth was first considered as a cost function by Amy *et al.* [16]. However, till now ProjectQ does not estimate the T-depth of a circuit. Previous works regarding the resource estimation for mounting Grover attack on several ciphers considered T-depth as a resource contraint [15, 148, 119, 130, 18]. While measuring T-depth, all gates apart from the T gates are ignored. In the case of measuring total depth, all gates are assigned a weight of 1. Like previous works, uncontrolled `SWAP` operations are regarded as free.

8.1.5 Realization of Classical '*AND*' Operation in Quantum Circuits

The `AND` operations in KATAN can be replaced with CCNOT gates in quantum circuits. However, CCNOT gate is not a basic gate operation and thus instead of applying it directly its decomposition into clifford+T set is considered. Resource

estimation of the proposed design is compared in terms of the cost metrics and thus several decompositions of CCNOT gates are considered to compare the corresponding cost metrics. As the codes for resource estimation are run on the ProjectQ framework, the default decomposition of CCNOT gates in that framework is considered initially. This decomposition has T-count 7, has T-depth 4 and is shown in Fig. 8-1.

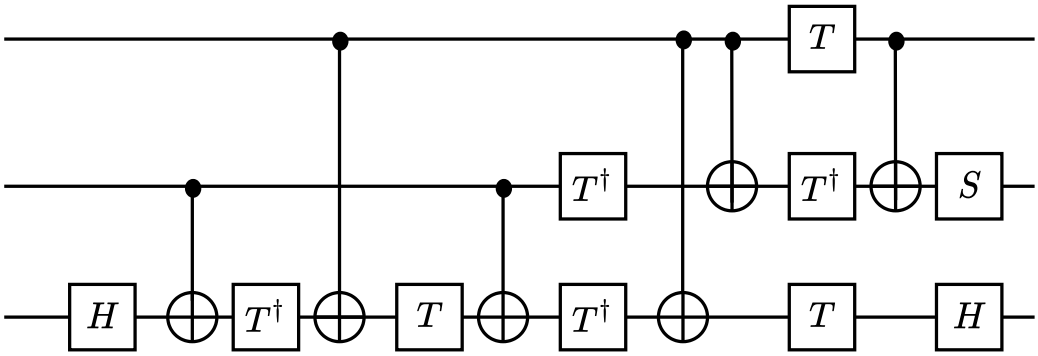


Figure 8-1: Decomposition of toffoli Gate into clifford + T set with T-depth 4.

There are two more decompositions of CCNOT/toffoli gate into clifford+T set. The decomposition proposed by Amy *et al.* has T-depth 3 and does not use any ancillary qubit [16]. Selinger proposed another decomposition which uses 4 ancillary qubits and has T-depth 1. Both decompositions are considered in our design and computational resources are estimated for the reversible quantum circuit of KATAN based on those decompositions.

Another design is considered where instead of using CCNOT gate, an `AND` gate is used which has T-depth 1. This `AND` gate is designed by Mathias Soeken [130] and is based on the work of Selinger [183] and Jones [133]. Fig. 8-2 shows the design of the `AND` gate used in this work.

8.2 Grover on Katan: Resource Estimation

KATAN, introduced in CHES 2009, is a family of lightweight block ciphers fulfilling the essential criteria to run in resource-constrained devices [91]. It is a hardware-oriented cipher specially designed for sensor networks, RFID tags and *Internet-of-*

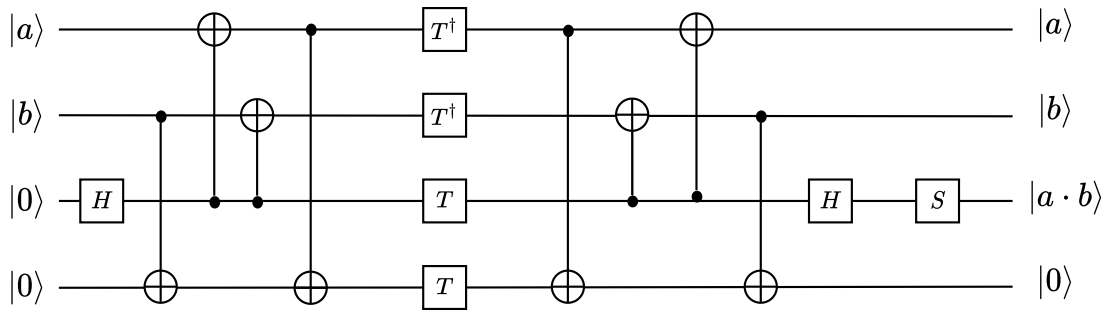


Figure 8-2: Design of `AND` gate.

things [91, 20]. Over the years, extensive cryptanalysis on KATAN is done, like, differential attacks [13], conditional differential cryptanalysis [139, 140], meet-in-the-middle attacks [218, 124, 109, 63], related-key boomerang attacks [123, 80], cube attack [12], subkey recovery attack [125], linear hull cryptanalysis [186], MILP-aided division property based analysis [193], etc. All these attacks are on the round-reduced versions of KATAN. Instead of having a relatively small key length (80-bit), till now, there is no such attack that penetrates all rounds of KATAN. This motivates us to analyze KATAN against the generic attack using Grover's algorithm. As a general rule of thumb, 80-bit security of KATAN against quantum adversary can be achieved by doubling the key length. However, extending the key length of non-linear feedback shift register-based block cipher is not straightforward and the results in this chapter might provide insights in designing an extended version of KATAN. To the best of our knowledge, till now no study is conducted regarding the concrete cost analysis about the Grover's key search on KATAN.

8.2.1 Resource Estimation of KATAN Implementation

Here, first of all, a reversible quantum circuit for KATAN using quantum gates and qubits is designed. The construction of the complete block cipher is composed of designing the round operations and designing the key scheduling operations. Then the cost analysis for the designed circuit is estimated.

Designing the Key Schedule

The key scheduling algorithm of KATAN consists of `XOR` operations for generating a key bit and thus its equivalent quantum circuit can be realized easily by using CNOT gates in place of the `XOR` operations. All variants of KATAN have 254 rounds in total and in each round two key qubits are required. Therefore, in total $254 \times 2 = 508$ key qubits are required. Among these 508 key qubits, 80 qubits are initially given and hence (508-80)=428 qubits are required to be generated. In the proposed design, four CNOT gates are required to generate a qubit and thus $428 \times 4 = 1712$ CNOT gates and 428 ancillary qubits are required for generating all the necessary key qubits. To construct the reversible circuit, all these operations are uncomputed after the full run of the KATAN block cipher.

Designing the Round Operation

Designing of the round operations mainly involves the realization of the non-linear feedback functions $f_a(\cdot)$ and $f_b(\cdot)$ using quantum gates and qubits. Both the functions are composed of `AND` operations and `XOR` operations. The `XOR` operation can be realized using the CNOT gate; whereas the `AND` operation can be realized using the CCNOT gate. After the round operations, to store the new feedback qubit, an ancillary qubit is used. In each round, $f_a(\cdot)$ and $f_b(\cdot)$ are computed once, twice and thrice for KATAN32, KATAN48 and KATAN64 respectively. So, the number of new qubits in each round for KATAN32, KATAN48 and KATAN64 is 2, 4 and 6 respectively and thus the total number of ancillary qubits are used for implementing $f_a(\cdot)$ and $f_b(\cdot)$ is 508, 1016 and 1524 respectively. After the completion of 254 rounds, the qubits corresponding to the ciphertext are fanned out to a quantum register and the round operations are uncomputed to realize the reversible circuit. The design of the reversible quantum circuit for KATAN is described in Algorithm 12. Note that, $C(a, t)$ refers to the application of CNOT gate on target qubit t with control qubit a and $Tof(a, b, c)$ refers to the application of CCNOT/toffoli gate on target qubit t with control qubits a and b. The values x_i and y_i corresponds to the values in Table 2.2.

Algorithm 12 Quantum Circuit for Round Operation of KATAN

INPUT: Message register $\mathcal{M}$, ciphertext register $\mathcal{C}$, key register $\mathcal{K}$

1. $numRounds \leftarrow 254$ and $iter \leftarrow c$ (The value of c is 1, 2 and 3 for KATAN32, KATAN48 and KATAN64 respectively.)
2. Initialize a register $\mathcal{I}$ with the round values of the irreducible polynomial IR.
3. $\mathcal{M}$ is divided into L_1 and L_2.
4. $numAQ \leftarrow numRounds \times iter$. $numAQ$ ancillary qubits are added to L_1 and L_2.
5. $(2 \times numRounds\text{-}80)$ ancillary qubits are added to $\mathcal{K}$.
6. Compute the new 428 key qubits and store them in the ancillary qubits of $\mathcal{K}$.
7. For $0 \leq j \leq numRounds$ perform the following operations-
   (a) $C(L_2[numAQ - j + y1], L_1[numAQ - 1 - j])$
   (b) $C(L_2[numAQ - j + y2], L_1[numAQ - 1 - j])$
   (c) $Tof(L_2[numAQ - j + y3], L_2[numAQ - j + y4], L_1[numAQ - 1 - j])$
   (d) $Tof(L_2[numAQ - j + y5], L_2[numAQ - j + y6], L_1[numAQ - 1 - j])$
   (e) $C(\mathcal{K}[2 * j + 1], L_1[numAQ - 1 - j])$
   (f) $C(L_1[numAQ - j + x1], L_2[numAQ - 1 - j])$
   (g) $C(L_1[numAQ - j + x2], L_2[numAQ - 1 - j])$
   (h) $Tof(L_1[numAQ - j + x3], L_1[numAQ - j + x4], L_2[numAQ - 1 - j])$
   (i) if($\mathcal{I}[j]$=1)
   $$C(L_1[numAQ - j + x5], L_2[numAQ - 1 - j])$$
   (j) $C(\mathcal{K}[s2 * j], L_2[numAQ - 1 - j])$
8. Fan out the qubits correspnding to the ciphertext to $\mathcal{C}$
9. Uncompute the operations in Step 6 and Step 7.

Table 8.1 lists the resource estimation for designing the KATAN cipher with the decomposition provided in ProjectQ.

Resource estimation using the decomposition of toffolli gate having T-depth 3 and T-depth 1 are listed in Table 8.2 and Table 8.3 respectively.

Table 8.4 lists the cost estimates of implementing the quantum reversible circuit

Table 8.1: Resource Estimation for Reversible Quantum Circuit of Katan Block Cipher using a decomposition of Toffoli gate with T-depth 4

Variant	#CNOT	#1qCliff	$\#T$	$\#M$	T-Depth	D	W
KATAN32	13870	2032	10668	32	1654	4461	1080
KATAN48	26332	4064	21336	48	2058	6253	1620
KATAN64	38794	6096	31944	64	1836	5752	2160

Table 8.2: Resource Estimation for Reversible Quantum Circuit of Katan Block Cipher using a decomposition of Toffoli gate with T-depth 3

Variant	#CNOT	#1qCliff	$\#T$	$\#M$	T-Depth	D	W
KATAN32	15394	3556	10668	32	1651	4071	1080
KATAN48	29380	7112	21336	48	1858	5825	1620
KATAN64	43366	10668	32004	64	1722	5367	2160

Table 8.3: Resource Estimation for Reversible Quantum Circuit of Katan Block Cipher using a decomposition of Toffoli gate with T-depth 1

Variant	#CNOT	#1qCliff	$\#T$	$\#M$	T-Depth	D	W
KATAN32	29110	2032	10668	32	508	3055	7176
KATAN48	56812	4064	21336	48	584	4661	13812
KATAN64	84514	6096	32004	64	536	4295	20448

using `AND` gate for the variants of KATAN block cipher.

Table 8.4: Resource Estimation for Reversible Quantum Circuit of KATAN Block Cipher using `AND` gate.

Variant	#CNOT	#1qCliff	$\#T$	$\#M$	T-Depth	D	W
KATAN32	16918	4572	6096	32	636	3315	2604
KATAN48	32428	9144	12192	48	779	4467	4668
KATAN64	47938	13716	18288	64	694	4075	6732

Cost Metrics of the Designs

Now, the cost metrics of the several designs are compared. As NIST puts no bound on the width of the circuit, so instead of the DW-cost metric only the depth of the

circuit is compared here. Table 8.5 compares the G cost and depth of various designs of the KATAN block cipher. From the table, it is evident that designs based on `AND` gate and T-depth one toffoli/CCNOT gate have comparatively lower depth. Hence, for constructing the Grover oracle these two designs are considered.

Table 8.5: Comparison of G-cost metric and Depth of the Designs.

Design/	KATAN32		KATAN48		KATAN64	
Decomposition	G	D	G	D	G	D
T-depth 4	$2^{14.7}$	$2^{12.12}$	$2^{15.66}$	$2^{12.61}$	$2^{16.23}$	$2^{12.49}$
T-depth 3	$2^{14.86}$	$2^{11.99}$	$2^{15.82}$	$2^{12.51}$	$2^{16.39}$	$2^{12.39}$
T-depth 1	$2^{15.35}$	$2^{11.58}$	$2^{16.33}$	$2^{12.19}$	$2^{16.9}$	$2^{12.09}$
`AND` gate	$2^{14.75}$	$2^{11.69}$	$2^{15.72}$	$2^{12.13}$	$2^{16.29}$	$2^{11.99}$

8.2.2 Quantum Resource Estimation of Grover on KATAN

Now, the resource requirement of Grover's oracle and Grover's search on KATAN is estimated. Initially, no parallelization is considered and it is assumed that the Grover operator is running in serial. Later, NIST's `MAXDEPTH` is considered and resource requirement based on `MAXDEPTH` is estimated. The reversible quantum circuits based on T-depth one tofolli gate and `AND` gate are used to estimate the resources for Grover's search on KATAN.

Resource Estimation of Grover's Oracle

Consider a block cipher with a block length of n-bit and a key length of k-bit. For such a block cipher, the probability of finding a unique key using r plaintext-ciphertext pairs is $e^{-2^{k-rn}}$ [130]. As discussed in § 8.1.2, the value of r should be at least $\lceil \frac{n}{k} \rceil$ to uniquely identify the correct key. These $\lceil \frac{k}{n} \rceil$ number of encryptions can be implemented in Grover oracle in parallel. The key length for all variants of KATAN is 80 bits. Based on the value of n, the value of r and the corresponding success probabilities are determined.

For KATAN32, $n = 32$ and $k = 80$; therefore r should be $\lceil \frac{80}{32} \rceil = 3$ and the probability of finding a unique key is approximately 0.99. Fig. 8-3 shows the Grover oracle for KATAN32. The value of r is set to 2 for KATAN48 and KATAN64 to recover a unique key with overwhelming probability. Grover oracle for KATAN48/KATAN64 is shown in Fig. 8-4. The estimated resources for implementing the Grover oracle for KATAN bock cipher is listed in Table 8.6.

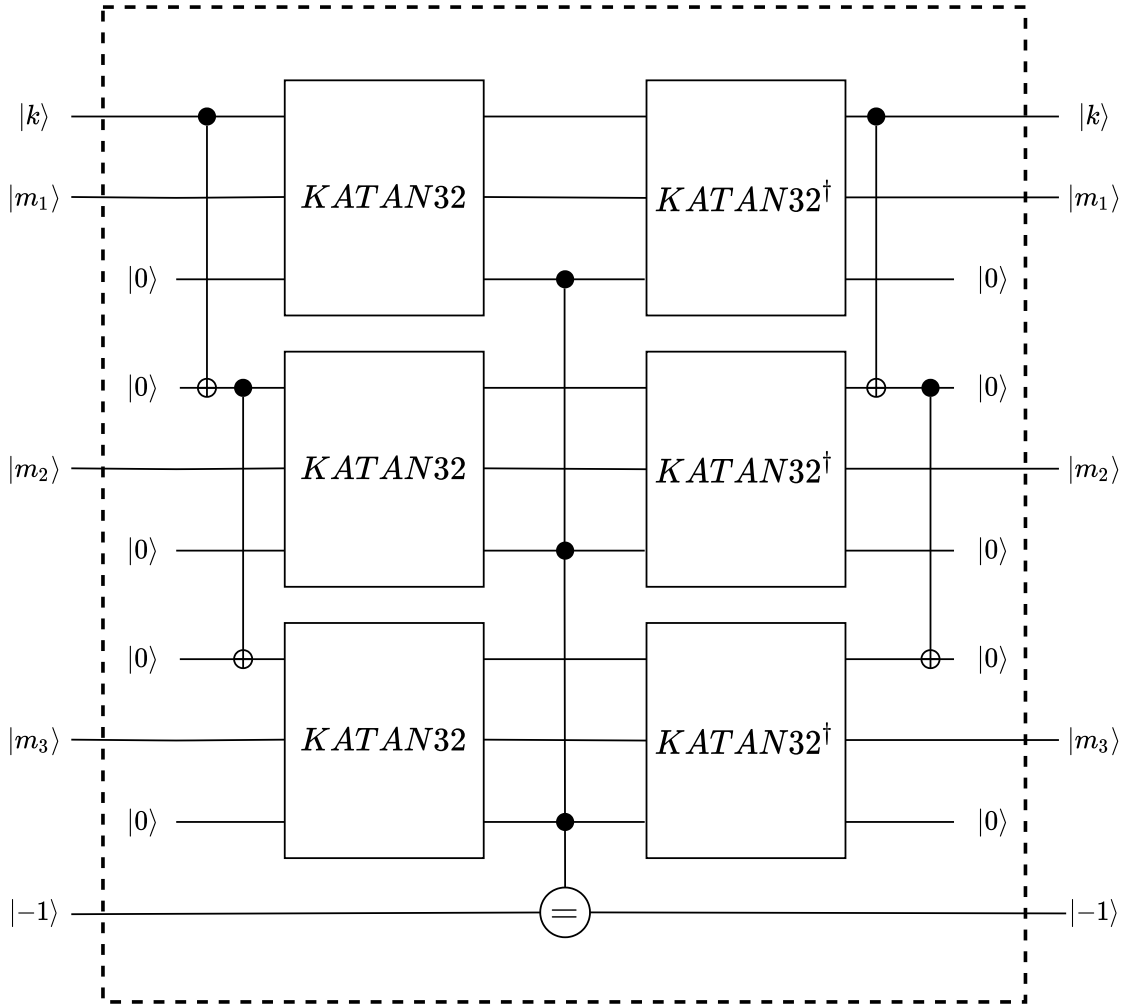


Figure 8-3: Grover Oracle of KATAN32

Cost Estimation of Grover's Search

For implementing the Grover's search, the Grover operator G needs to be applied $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ times (See § 2.6.2). For estimating the cost of Grover's search, cost related

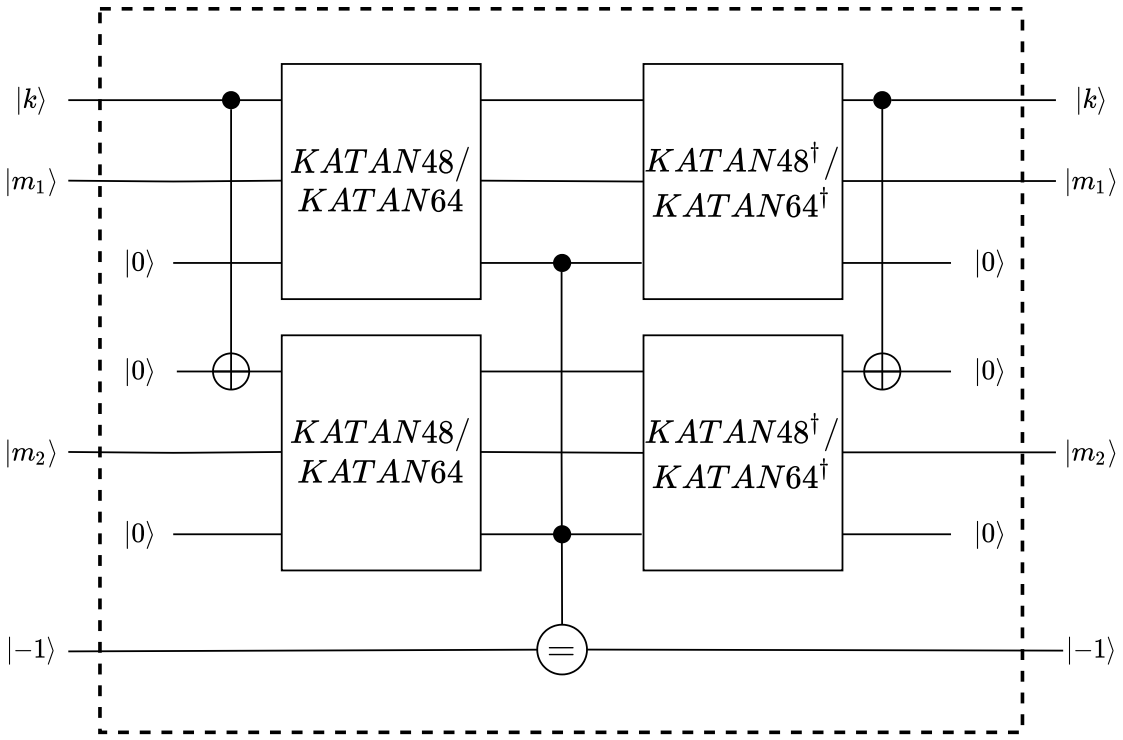


Figure 8-4: Grover Oracle of KATAN48/KATAN64

Table 8.6: Resource Estimation for Grover Oracle of Katan Block Cipher

Design	Variant	#CNOT	#1qCliff	$\#T$	T-Depth	D	W
T-depth 1 CCNOT	KATAN32	175556	12352	64630	1006	6052	42688
	KATAN48	227696	16304	85680	1160	9262	54944
	KATAN64	338600	24448	128464	1066	8548	81440
`AND` Gate	KATAN32	102404	27592	37248	1259	6560	15256
	KATAN48	130160	36624	49104	1546	8872	18368
	KATAN64	192296	54928	73600	1379	8104	26576

to U_f is considered; the cost of $U_{\Psi^\perp}$ is ignored. It is assumed that no parallelization is used and thus all the resources (except width) are increased by a factor of $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$. Table 8.7 lists the cost estimation of implementing Grover's search on KATAN without parallelization.

Table 8.7: Resource Estimation for Grover's Search on Katan Block Cipher

Design	Variant	#CNOT	#1qCliff	$\#T$	T-Depth	D	W
T-depth 1 CCNOT	KATAN32	$2^{57.07}$	$2^{53.24}$	$2^{55.63}$	$2^{49.63}$	$2^{52.21}$	$2^{55.03}$
	KATAN48	$2^{57.45}$	$2^{53.64}$	$2^{56.04}$	$2^{49.83}$	$2^{52.83}$	$2^{55.4}$
	KATAN64	$2^{58.02}$	$2^{54.23}$	$2^{56.62}$	$2^{49.71}$	$2^{52.71}$	$2^{55.96}$
AND Gate	KATAN32	$2^{56.29}$	$2^{54.4}$	$2^{54.84}$	$2^{49.95}$	$2^{52.33}$	$2^{53.54}$
	KATAN48	$2^{56.64}$	$2^{54.81}$	$2^{55.23}$	$2^{50.25}$	$2^{52.77}$	$2^{53.82}$
	KATAN64	$2^{57.2}$	$2^{55.4}$	$2^{55.82}$	$2^{50.08}$	$2^{52.64}$	$2^{54.35}$

Cost Estimation under a Depth Limit

Cost estimation in Table 8.7 is listed without considering parallelization. However, to respect a depth limit parallelization becomes inevitable. In the call of the proposal for post-quantum cryptography standardization, NIST has put a restriction on the depth limit [171]. The depth limit is referred to as `MAXDEPTH` and its value can range from 2^{40} to 2^{96}. This forces to mount the attack using parallelization of Grover's search algorithm. As discussed in Section 8.1.2, it is assumed that inner parallelization is used to respect the depth limit.

To estimate the gate cost by considering the depth limit, a formula is provided by NIST. Consider a circuit that runs non-parallel Grover's search with depth D and total gates G where $D = d\times$`MAXDEPTH` for some $d \geq 1$. Now, to mount an attack by achieving the same success probability as before while fitting the depth limit `MAXDEPTH`, d^2 parallel machines are required where each machine with gate cost of G/d run for $1/d$ fraction of the total time. Hence, the total gate cost is $(G/d) \times d^2 = GD/$`MAXDEPTH`. Based on this formula, gate costs for mounting Grover's attack under the constraint of different plausible values of `MAXDEPTH` are listed in Table 8.8.

Table 8.8: Resource Estimation for Grover's Search on Katan Block Cipher with Depth Limit

Design	Variant	GD	MAXDEPTH		
			2^{40}	2^{64}	2^{96}
T-depth 1 CCNOT	KATAN32	$2^{109.8}$	$2^{69.8}$	$2^{45.8}$	$2^{13.8}$
	KATAN48	$2^{110.81}$	$2^{70.81}$	$2^{46.81}$	$2^{14.81}$
	KATAN64	$2^{111.27}$	$2^{71.27}$	$2^{47.27}$	$2^{15.27}$
AND Gate	KATAN32	$2^{109.33}$	$2^{69.33}$	$2^{45.33}$	$2^{13.33}$
	KATAN48	$2^{110.14}$	$2^{70.14}$	$2^{46.14}$	$2^{14.14}$
	KATAN64	$2^{110.58}$	$2^{70.58}$	$2^{46.58}$	$2^{14.58}$

8.3 Grover on Present: Resource Estimation

Present [62] is an ultra-lightweight hardware-optimized block cipher specifically designed for area-constrained and power-constrained devices. Over the years, its efficient hardware performance along with strong security has prompted researchers to perform a lot of security analysis. There are several analysis on its round-reduced version; like, linear cryptanalysis [81, 172, 84, 68, 150, 10, 78, 149, 156, 155], differential cryptanalysis [196, 197, 154, 59, 204, 205], improbable differential attack [195], related-key differential attack [173, 105], algebraic cryptanalysis [147], fault attacks [158, 24, 202], differential power analysis [215], side channel cube attacks [212, 217], biclique cryptanalysis [184, 152, 11], integral attack [208], deep learning based distinguishers [127], known-key distinguishers [61], truncated differential attack [60], etc. However, except for the known key distinguisher, full Present block cipher is still impregnable to classical attacks. This motivates us to analyze the security of Present against quantum attacks. Like all other block ciphers, Present is also susceptible to Grover's attack; but, it does not provide any concrete idea regarding the security of Present in the post-quantum world. Hence, estimating the resources for mounting Grover's attack on Present helps us to analyze its security more appropriately against quantum adversaries.

8.3.1 A Quantum Circuit on Present

The implementation of Present in ProjectQ is discussed in this section. First of all, reversible quantum circuit for round operations (AddRoundKey, pLayer, sBoxLayer) and key scheduling algorithm are designed. Then this circuits are combined together to obtain a reversible quantum circuit for Present block cipher.

Decomposition of Toffoli Gate

For a fair comparison with other related works, the quantum circuits are required to be designed using Clifford+T gate set. However, in several occasions in this work, toffoli gates are used to realize the quantum circuits. In such cases, decomposition of the toffoli gates using Clifford+T gate set becomes necessary. For that purpose, three decompositions are followed in this work.

First, the default decomposition of toffoli gates into Clifford+T set employed in ProjectQ framework is considered. This decomposition uses seven T gates, three 1-qubit clifford gates and three qubits.

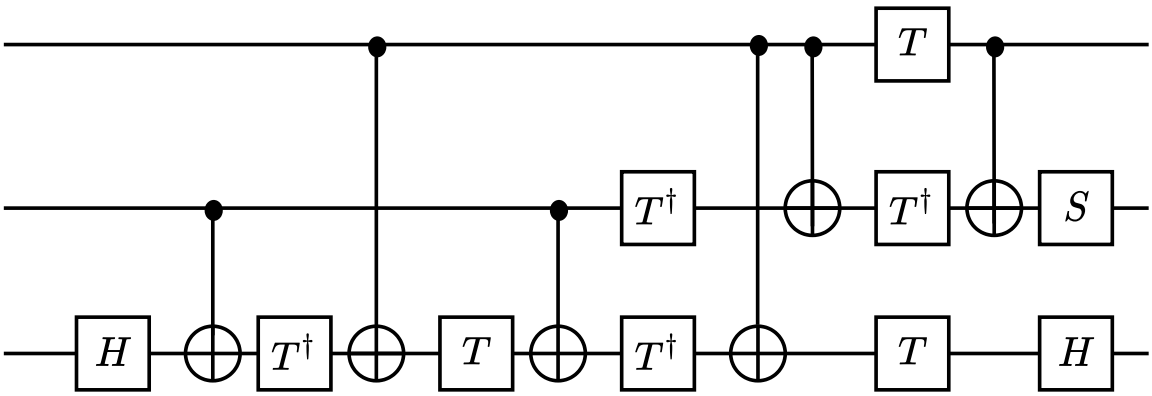


Figure 8-5: Decomposition of Toffoli gate into Clifford+T Set with T-depth of 4

Next, consider the decomposition of toffoli gate into Clifford+T gate set provided by Amy *et al.* [16] shown in Fig 8-6. This decomposition uses 6 CNOT gates, 7 T gates, 3 clifford gates and its T-depth is 3. It is conjectured that T-depth of 3 is optimal for quantum circuits without ancillas obtained through the decomposition of toffoli gate [16].

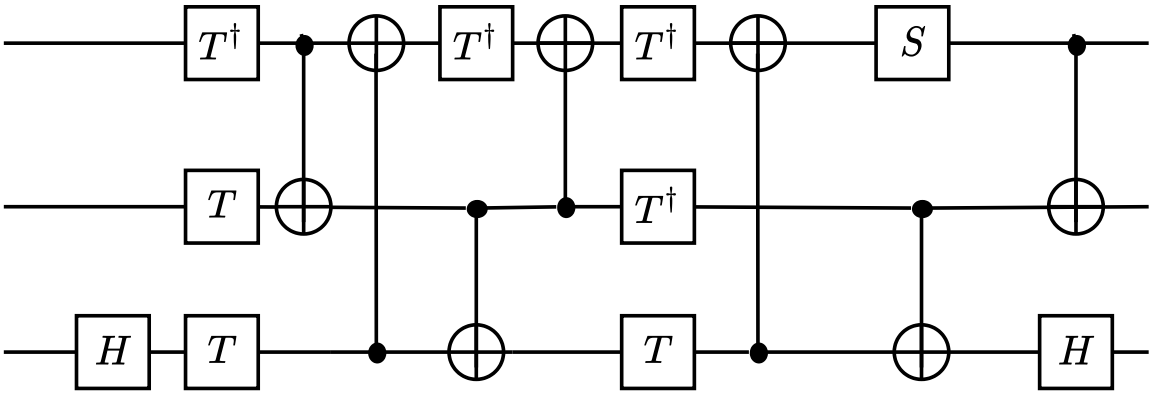


Figure 8-6: Decomposition of toffoli gate into Clifford+T Set with T-depth of 3

Selinger gives a representation of toffoli gate using clifford + T gate of T-depth 1 [183] which is shown in Fig. 8-7. Along with lower T-depth, this representation has lower depth than the representation in Fig. 8-6 but it uses some ancilla qubits; hence the width is increased.

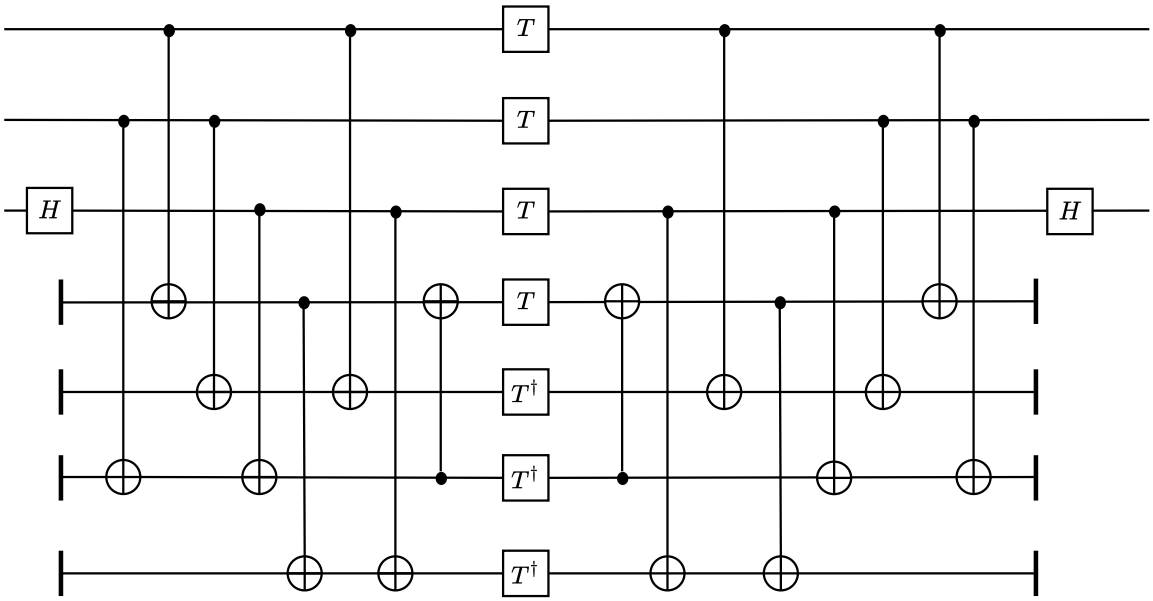


Figure 8-7: Decomposition of toffoli gate into Clifford+T Set with T-depth 1

Designing the Round Operations

Each round of Present is constituted of three operations, namely, AddRoundKey, pLayer and sBoxLayer and the complete Present block cipher is comprised of 31 such rounds. For both variant of Present the block length is 64 bits. Hence, same quantum circuit for operation is valid for both variants. Here, a reversible quantum circuit for each round operation is provided.

AddRoundKey. This operation consists of XOR-ing of a 64-bit round key to the internal state of Present. This operation can be realized in the quantum circuit by using 64 CNOT gates where the key bits acts as control bits and internal state bits are target bits.

pLayer. This operation is a linear permutation of the state bits which can be realized using SWAP gates. However, in ProjectQ, we explicitly do not use SWAP gates to implement the pLayer; instead input to each s-box is maintained based on the output of pLayer and ProjectQ internally use SWAP gates to bring inputs to each s-box to a neighborhood. Thus pLayer is implicitly realized in the implementation.

sBoxLayer. Present uses 4×4-bit s-box. The input and output of the s-box are shown in Table 2.4. `Revkit` [7] is integrated into ProjectQ to find reversible logic by using automated synthesis routines. `PermutationOracle` operation of the `Revkit` library is used to synthesis a reversible circuit for a permutation. As the s-box of Present is permutation over 2^4 elements, `PermutationOracle` automatically finds a reversible circuit over 4 qubits to realize the permutation. The circuit is generated using toffoli gates, CNOT gates and NOT gates. However, instead of using toffoli gates, its equivalent decomposition using 1-qubit clifford + T set is considered. Quantum circuit of the s-box using toffoli gates is shown in Algorithm 13. Note that, $Tof(a, b, c)$ denotes the application of tofolli gate on target qubit c using the control qubit a and b; $CNOT(a, b)$ denotes application of CNOT gate on target qubit b using the control qubit a; and $X(a)$ denotes the application of NOT gate on qubit a. Fig. 8-8 shows

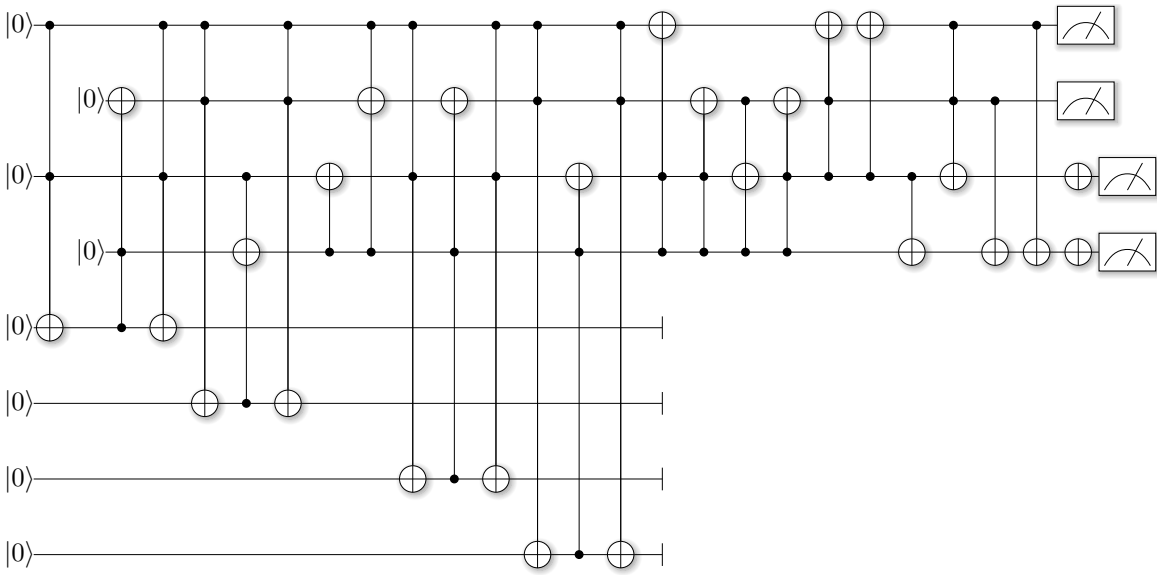


Figure 8-8: Quantum Circuit for Present S-box using toffoli Gate

the circuit of Present s-box using toffoli gates. This s-box circuit uses 19 toffoli gates, 5 CNOT gates and 2 NOT gates. However, to realize the circuits using 1-qubit clifford + T set, CNOT gates and NOT gates, several decompositions of toffoli gates (discussed in Section 8.3.1) are considered.

In Table 8.9, resource requirement for Present s-box under various decompositions are listed. It is clearly evident that by using toffoli gate of T-depth 1, T-depth as well as overall depth of the circuit decreases significantly; whereas the width of the circuit also increases significantly. As restrictions on width are not considered in NIST's call for proposal for post-quantum cryptography [171, §4.A.5], decomposition of the s-box using toffoli gate of depth 1 become interesting, in particular, for its lower depth. The quantum circuits for the s-box are unit tested in ProjectQ for its correctness.

Designing the Key Scheduling Algorithm.

There are two variants of Present block cipher on the basis of key size- 80-bit key and 128-bit key. Round operation of KSA of both the variant consists of a rotation operation, XOR-ing of a 5-bit round counter and application of s-box to some bits. For the 80-bit variant, in each round a single s-box is applied; whereas two s-boxes

Algorithm 13 Quantum Circuit for S-box of Present

INPUT: Qubits: q_0, q_1, q_2, q_3 where q_3 is the most significant qubit and q_0 is the least sigificant qubit

1. Initialize an ancillary register ANC with 4 qubits
2. $Tof(q_0, q_1, ANC[0])$
3. $Tof(ANC[0], q_3, q_1)$
4. $Tof(q_0, q_2, ANC[0])$ and then $Tof(q_0, q_1, ANC[1])$
5. $Tof(q_2, ANC[1], q_3)$ and then $Tof(q_0, q_1, ANC[1])$
6. $CNOT(q_3, q_2)$
7. $Tof(q_0, q_3, q_1)$
8. $Tof(q_0, q_2, ANC[2])$ and then $Tof(ANC[2], q_3, q_1)$
9. $Tof(q_0, q_2, ANC[2])$ and then $Tof(q_0, q_1, ANC[3])$
10. $Tof(ANC[3], q_3, q_2)$ and then $Tof(q_0, q_1, ANC[3])$
11. $Tof(q_3, q_2, q_0)$
12. $Tof(q_3, q_2, q_1)$
13. $Tof(q_3, q_1, q_2)$
14. $Tof(q_3, q_2, q_1)$
15. $Tof(q_2, q_1, q_0)$
16. $CNOT(q_2, q_0)$
17. $CNOT(q_2, q_3)$
18. $Tof(q_0, q_1, q_2)$
19. $CNOT(q_1, q_3)$
20. $CNOT(q_0, q_3)$
21. $X(q_2)$ and $X(q_3)$
22. Measure q_3, q_2, q_1, q_0 for the output of the s-box

are applied on 8 bits for the 128-bit variant. For designing the quantum circuit, XOR-ing of round counter can be realized by using the CNOT gate where counter

-

Decomposition	#CNOT	#1qCliff	#T	#M	T-Depth	Full Depth	Width
T-Depth 4	119	36	133	4	63	190	8
T-Depth 3	138	59	133	4	34	177	8
T-Depth 1	309	36	133	4	19	139	84

Table 8.9: Quantum Resources Required for Present S-box for Several Decompositions

bits are the control bits and corresponding key bits acts as target. In ProjectQ, XOR-ing of round counter is realized by using `AddConstant` operation. The rotation operation is not explicitly implemented; rather application of s-box and XOR-ing of round counter on corresponding qubits are controlled. ProjectQ internally uses `SWAP` gate to realize the rotation operation. Fig.8-9 shows the quantum circuit for the KSA of 80-bit key. In terms of resource requirement, 128-bit KSA is quite similar with the 80-bit KSA; only difference is the extra usage of a s-box in each round of 128-bit KSA. Resource requirement for realisation of KSA in quantum circuit under several kind of synthesis is listed in Table 8.10. In quantum circuit for KSA, T gates are required only for designing the s-box. As the number of s-boxes used in 128-bit KSA is twice the number of s-boxes used in 80-bit KSA, the number of T-gates is double for 128-bit KSA with respect to 80-bit KSA when similar decomposition of toffoli gate is cosidered.

Implementation of Full Present

Now, a reversible quantum circuit for full Present is designed. Quantum circuits for round operations and key scheduling algorithm are combined to obtain the quantum circuit for the complete Present block cipher. The resource estimation for the reversible circuit considering the different decompositions of the toffoli gate are listed in Table 8.11.

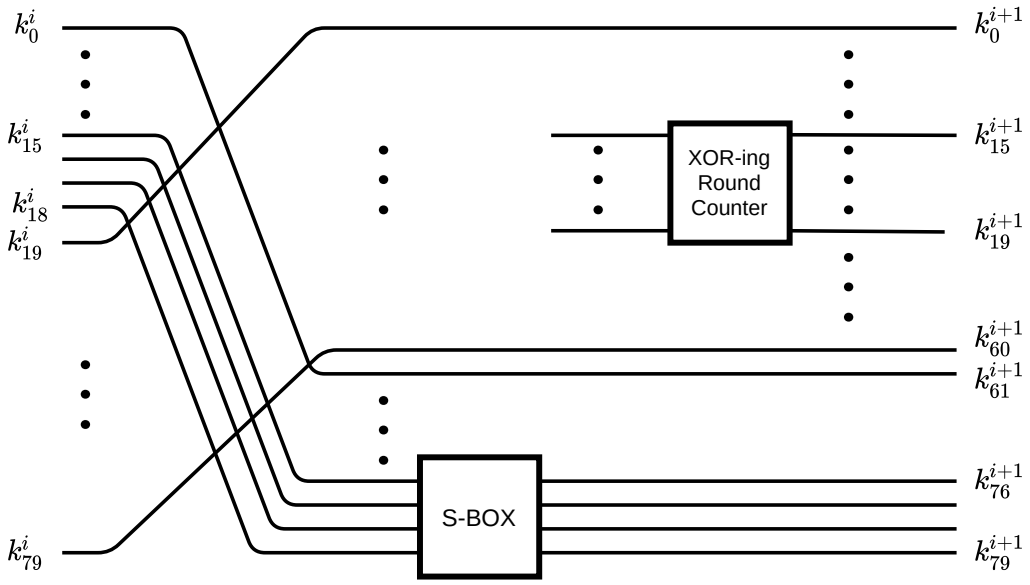


Figure 8-9: Present Key Scheduling Function of 80-bit Key

Comparison using Cost Metrics

The proposed designs can be compared using the cost metrics discussed in Section 8.1.3. Along with the G-cost and DW-cost, the full depth of all the proposed quantum circuits for Present are listed in Table 8.12. Although, the circuits designed using the toffoli gates of T-depth 1 have lowest depth, but their G-cost and DW-cost is the highest among all the designs. In comparison to toffoli gates of T-depth 4, the overall depth of toffoli gates of T-depth 3 is lower without using any ancillary qubits at the expense of using more quantum gates. And thus the circuits designed using toffoli gates of T-depth 3 have lowest DW-cost; whereas the circuits designed using the toffoli gates of T-depth 4 have lowest G-cost. Toffoli gates of T-depth 1 uses more qubits and gates to reduce the depth and T-depth; and thus the G-cost and DW-cost of the corresponding quantum circuits of Present are high.

8.3.2 Quantum Resource Estimation of Grover on Present

Now, using the proposed quantum circuit of Present, a concrete resource estimation is conducted for mounting Grover's attack on Present block cipher. First, Grover oracle is designed for Present to mount grover search. Then, based on the design

Key Size	Decomposition of Toffoli Gate	#CNOT	#1qCliff	#T	T-Depth	Full Depth	Width
80-bit	T-Depth 4	119	71	133	63	189	148
	T-Depth 3	138	90	133	57	176	164
	T-Depth 1	309	71	133	19	138	224
128-bit	T-Depth 4	238	107	266	65	189	200
	T-Depth 3	276	145	266	57	176	200
	T-Depth 1	618	107	266	19	138	352

Table 8.10: Resource Estimation for Key Scheduling Algorithm of Present

Key Size	Decomposition of Toffoli Gate	#CNOT	#1qCliff	#T	T-Depth	Full Depth	Width
80-bit	T-Depth 4	64761	19912	70091	2010	6004	2316
	T-Depth 3	74774	29925	70091	1818	5619	2316
	T-Depth 1	164891	19912	70091	606	4407	42368
128-bit	T-Depth 4	68450	21028	74214	2015	5833	2488
	T-Depth 3	79052	31630	74214	1767	5461	2488
	T-Depth 1	174470	21028	74214	589	4283	44896

Table 8.11: Resource Estimation for Reversible Quantum Circuit of Present

Decomposition of	Present-80			Present-128		
Toffoli Gate	D	G	DW	D	G	DW
T-depth 4	$2^{12.55}$	$2^{17.24}$	$2^{23.73}$	$2^{12.51}$	$2^{17.32}$	$2^{23.79}$
T-depth 3	$2^{12.46}$	$2^{17.42}$	$2^{23.63}$	$2^{12.41}$	$2^{17.5}$	$2^{23.7}$
T-depth 1	$2^{12.11}$	$2^{17.96}$	$2^{27.48}$	$2^{12.06}$	$2^{18.04}$	$2^{27.52}$

Table 8.12: Comparison of Reversible Quantum Circuit of Present using G-cost Metric and DW-cost Metric

of Grover oracle, resources required to perform a key recovery attack on Present is

estimated. Finally, due to NIST's restriction on depth-limit, cost-estimation under depth restrictions is conducted.

Resource Estimation of Grover Oracle

Here, quantum circuit for Grover oracle of Present is designed. While designing the Grover oracle, the number of plaintext-ciphertext pairs are required to recover the right key uniquely needs to be determined. Jaques *et al.* shows that for a block cipher with block length of n-bit and key length of k-bit, if r plaintext-ciphertext pairs are used, then $r \geq \lceil \frac{k}{n} \rceil$ [130]. In such case, then the probability of uniquely recovering the correct key is $e^{-2^{k-rn}}$ [130].

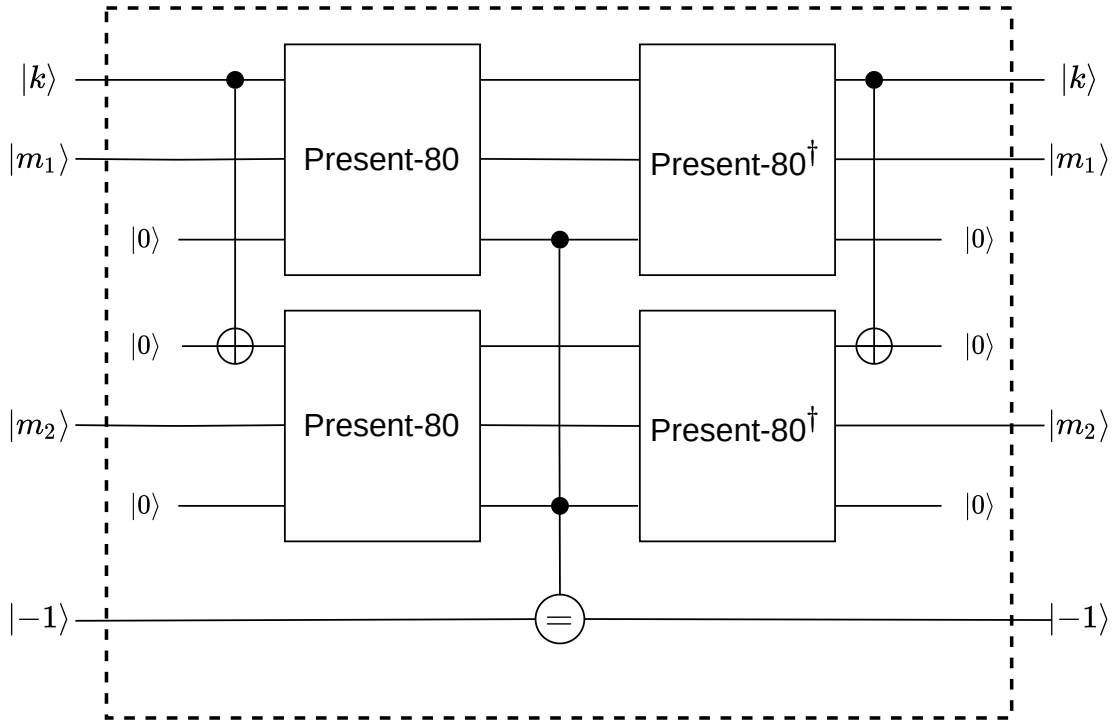


Figure 8-10: Grover Oracle for Present-80

For Present-80, $n = 64$ and $k = 80$; thus $r \geq \lceil \frac{80}{64} \rceil \implies r \geq 2$ and the probability of finding a unique key is 0.99 for $r = 2$. For Present-128, $n = 64$ and $k = 128$; so, $r \geq 2$ and the success probability is 0.36 for $r = 2$. For $r = 3$, the success probability for Present-128 is 0.99. Grover oracle for Present-80 and Present-128 is shown in Fig. 8-10 and Fig. 8-11 respectively. Table 8.13 lists the resources required to design

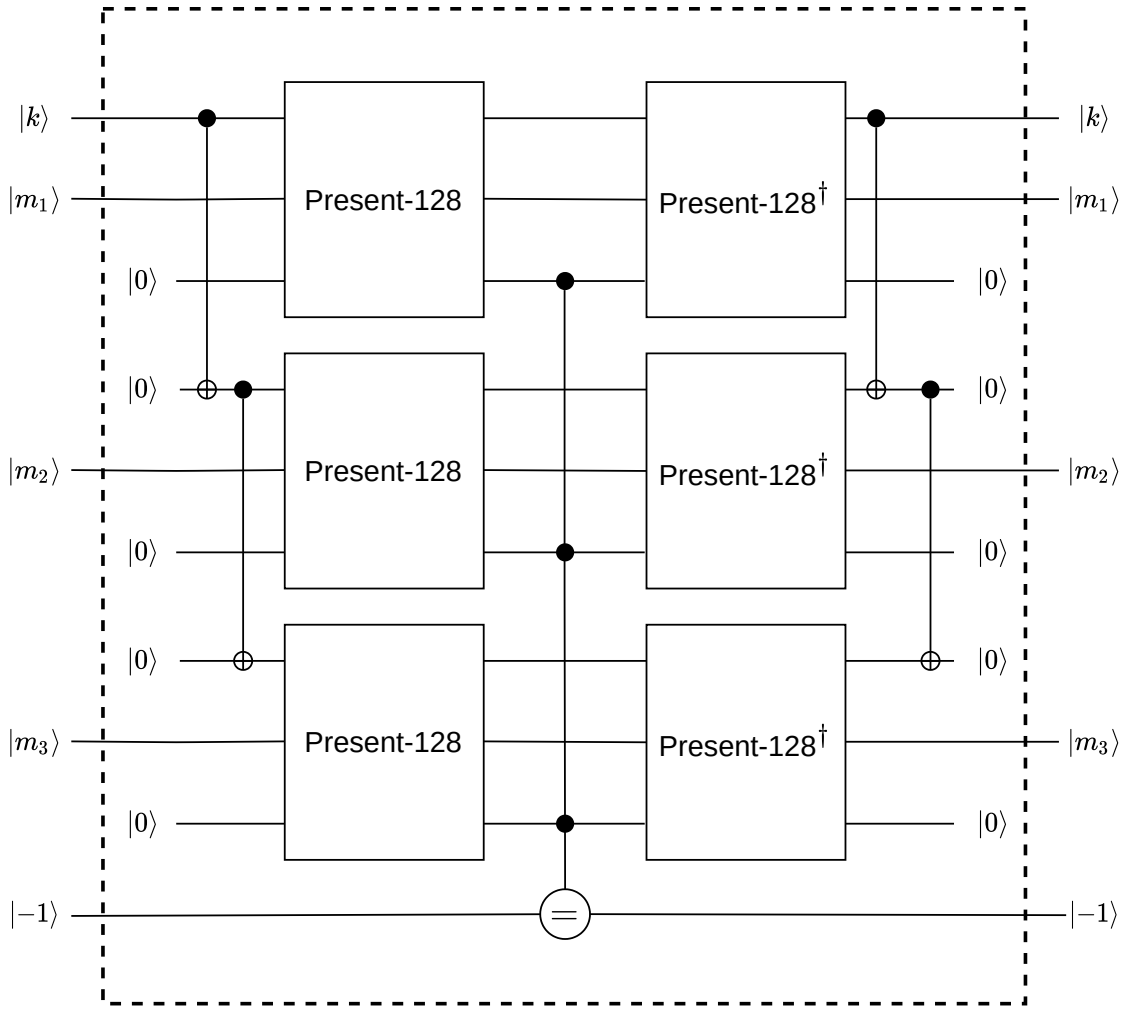


Figure 8-11: Grover Oracle for Present-128

the Grover oracle with their corresponding success probabilities.

Resource Estimation of Grover's Search

To mount key recovery attack on a block cipher using Grover's search, $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ iterations of Grover operator G is required. While estimating the resources, cost incurred by the operator U_f is considered only; cost imposed by the operator $U_{\psi^\perp}$ is ignored. In this case, no restriction on depth limit is considered and assumed that Grover operator is applied in serial. Hence, to estimate the resources of mounting Grover's search, the resources (except width) in Table 8.13 are multiplied by $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$. As it is assumed that no parallelization is involved, so width remains the same as in Grover

Key Size	r	p_s	Decomposition of Toffoli Gate	#CNOT	#1qCliff	$\#T$	T-Depth	Full Depth	Width
80-bit	2	0.99	T-depth 4	259588	79712	280812	4049	11999	8912
			T-depth 3	299640	99242	300838	4120	11248	8912
			T-depth 1	660108	79712	280812	1216	8824	169120
128-bit	2	0.36	T-depth 4	274248	84136	297248	3941	11673	9448
			T-depth 3	316656	105344	318448	4003	10932	9448
			T-depth 1	698392	84152	297304	1182	8576	179080
128-bit	3	0.99	T-depth 4	412020	126428	446544	3948	11694	14176
			T-depth 3	475632	158240	478344	4010	10953	14176
			T-depth 1	1048236	126452	446628	1189	8597	268624

Table 8.13: Resource Estimation for Grover Oracle of Present. p_s denotes the Success Probability of Recovering the Right Key Uniquely.

Key Size	r	p_s	Decomposition of Toffoli Gate	#CNOT	#1qCliff	$\#T$	T-Depth	Full Depth	Width
80-bit	2	0.99	T-depth 4	$2^{57.64}$	$2^{55.93}$	$2^{57.75}$	$2^{51.63}$	$2^{53.2}$	$2^{13.12}$
			T-depth 3	$2^{57.84}$	$2^{56.25}$	$2^{57.85}$	$2^{51.66}$	$2^{53.11}$	$2^{13.12}$
			T-depth 1	$2^{58.98}$	$2^{55.93}$	$2^{57.75}$	$2^{49.9}$	$2^{52.76}$	$2^{17.37}$
128-bit	2	0.36	T-depth 4	$2^{81.72}$	$2^{80.01}$	$2^{81.83}$	$2^{75.6}$	$2^{77.16}$	$2^{13.21}$
			T-depth 3	$2^{81.92}$	$2^{80.34}$	$2^{81.93}$	$2^{75.62}$	$2^{77.07}$	$2^{13.21}$
			T-depth 1	$2^{83.06}$	$2^{80.01}$	$2^{81.83}$	$2^{73.86}$	$2^{76.72}$	$2^{17.45}$
128-bit	3	0.99	T-depth 4	$2^{82.3}$	$2^{80.6}$	$2^{82.42}$	$2^{75.6}$	$2^{77.16}$	$2^{13.79}$
			T-depth 3	$2^{82.51}$	$2^{80.92}$	$2^{82.52}$	$2^{75.62}$	$2^{77.07}$	$2^{13.79}$
			T-depth 1	$2^{83.65}$	$2^{80.6}$	$2^{82.42}$	$2^{73.87}$	$2^{76.72}$	$2^{18.04}$

Table 8.14: Resource Estimation for Grover Search on Present

oracle. Resource estimation for mounting Grover's search is listed in Table 8.14.

Table 8.15 compares between the circuits used for mounting Grover's search on

Decomposition of Toffoli Gate	Present-80, r=2			Present-128, r=2			Present-128, r=3		
	D	G	DW	D	G	DW	D	G	DW
T-depth 4	$2^{53.2}$	$2^{58.89}$	$2^{66.32}$	$2^{77.16}$	$2^{82.97}$	$2^{90.37}$	$2^{77.16}$	$2^{83.56}$	$2^{90.95}$
T-depth 3	$2^{53.11}$	$2^{59.07}$	$2^{66.33}$	$2^{77.07}$	$2^{83.15}$	$2^{90.28}$	$2^{77.07}$	$2^{83.74}$	$2^{90.86}$
T-depth 1	$2^{52.76}$	$2^{59.61}$	$2^{70.13}$	$2^{76.72}$	$2^{83.69}$	$2^{94.17}$	$2^{76.72}$	$2^{84.28}$	$2^{94.76}$

Table 8.15: Comparison of Quantum Circuit for Grover Search on Present using G-cost Metric and DW-cost Metric

Decomposition of Toffoli Gate	Variant	r	GD	MAXDEPTH		
				2^{40}	2^{64}	2^{96}
T-depth 4	Present-80	2	$2^{112.09}$	$2^{72.09}$	$2^{48.09}$	$2^{16.09}$
	Present-128	2	$2^{160.13}$	$2^{120.13}$	$2^{96.13}$	$2^{64.13}$
	Present-128	3	$2^{160.72}$	$2^{120.72}$	$2^{96.72}$	$2^{64.72}$
T-depth 3	Present-80	2	$2^{112.18}$	$2^{72.18}$	$2^{48.18}$	$2^{16.18}$
	Present-128	2	$2^{160.22}$	$2^{120.22}$	$2^{96.22}$	$2^{64.22}$
	Present-128	2	$2^{160.81}$	$2^{120.81}$	$2^{96.81}$	$2^{64.81}$
T-depth 1	Present-80	2	$2^{112.37}$	$2^{72.37}$	$2^{48.37}$	$2^{16.37}$
	Present-128	2	$2^{160.41}$	$2^{120.41}$	$2^{96.41}$	$2^{64.41}$
	Present-128	2	2^{161}	2^{121}	2^{97}	2^{65}

Table 8.16: Gate Cost for Grover's Search on Present with Depth Limit

Present proposed in this chapter. It can be concluded from the table that using low-depth toffoli gate is a costly affair for mounting key recovery attack on Present.

Cost Estimation under a Depth Limit

In Table 8.14, the values are computed without considering any restriction on the depth of the circuit. However, NIST has put restriction on the maximum depth of the circuit (`MAXDEPTH`) in its call for the proposal for post-quantum cryptography standardization [171]. The minimum and maximum plausible value of `MAXDEPTH` is 2^{40} and 2^{96} respectively. The restriction on depth-limit alters the total gate cost of

mounting key search. Consider, a non-parallel circuit for Grover's search have G gate cost and D depth. If Grover's search is parallelized by restricting the depth-limit to `MAXDEPTH`, then the modified gate cost is $GD/$`MAXDEPTH` [171]. Table 8.16 lists the gate cost of Grover's search under the depth restriction.

8.4 Chapter Summary

Resource estimation for Grover's key search on the family of KATAN block cipher and Present is explored with respecting the NIST's `MAXDEPTH` depth restrictions. Several designs of the reversible quantum circuit for both the block ciphers are proposed focusing on minimizing the depth. Design based on `AND` gates produce relatively low depth circuits for KATAN48 and KATAN64; whereas for KATAN32 design based on T-depth one toffoli gate produce shallow circuit. For Present, circuits for Grover oracle designed using toffoli gates of T-depth 4 has the lowest G-cost; whereas Grover oracle for Present-128 designed using toffoli gates of T-depth 3 have lowest DW-cost.

9 CONCLUSION

Contents

The main topic of the thesis is cryptanalysis of symmetric key schemes. Our main results are included in Chapter 3, 4, 5, 6, 7 and 8.

9.1 Summary

In the first contributory work, we report iterated truncated differential based attacks on the internal keyed permutation of FlexAEAD. These attacks are further modified to key recovery attacks and applied on the nonce-based sequence number generator to mount forgery attack on FlexAEAD.

Next, we have applied the yoyo attack on the internal keyed permutation of FlexAEAD to devise deterministic distinguishers. In doing so, we have identified the underlying Super-Sbox constructions of these keyed permutations. These attacks are also exploited to recover the secret key.

Then, we analyze the impact of yoyo attack on some `AES`-based public permutations. We proposed new cryptanalytic techniques by augmenting yoyo with classical, improbable and impossible differentials. To show the effectiveness of the proposed techniques, several attacks are mounted on the round-reduced version of AESQ and

AES in the known key model.

We presented another new cryptanalytic technique by embedding the yoyo within boomerang and named it as boomeyong. The proposed strategy is used to devise key recovery attacks for 5-round and 6-round AES. To show the versatility of the technique, round-reduced version of AES-based block cipher Pholkos is also shown vulnerable to the proposed attack.

Next, we mount quantum attacks on HCTR, Tweakable-HCTR and HCH. We have shown that all the three schemes are susceptible to attacks in the Q_2 model whereas HCTR and HCH are also vulnerable in the Q_1 model.

Finally, we study the resource estimation for mounting Grover's attack on KATAN and Present. We proposed reversible quantum circuits for both the block ciphers which are used to concretize the resource estimation.

9.2 Open Problems

Based on the work in the thesis, some interesting open problems can be considered as future work. We list them as below.

1. In Chapter 5, yoyo game is used to mount attacks on AESQ permutation and on AES in the known key setting. It would be interesting to find out other ways to extend the yoyo game to mount attacks on public permutations.

2. In Chapter 5 and Chapter 6, new cryptanalytic techniques improbable differential yoyo, impossible differential yoyo, impossible differential bi-directional yoyo and boomeyong are introduced. Finding additional application of these techniques would be an interesting research area.

3. In Chapter 5, the extension of yoyo attacks on substitution-permutation networks depend on the MC operation. An interesting problem needs to be investigated is which type of linear layers can resist such kind of attacks.

4. In Chapter 4, we have applied the existing result of yoyo attack on generic 2-round substitution-permutation networks. Whether the yoyo attack can be

extended for generic 3-round substitution-permutation networks is an interesting question worthy of further investigation.

5. In Chapter 8, we have provided resource estimation for mounting Grover's attack on Present and Katan. Investigation regarding the development of a tool to automate the resource estimation of such kind of ciphers is quite interesting.

A

SAMPLE TRAIL FOR 5-ROUND AES-128

Here, a trail for 5-round AES-128 as claimed in Section 6.2.1 is provided as an illustration. Note that, the trail is searched using only 2^{23} encryptions and checking whether the six specific bytes in the intermediate state after 4 rounds of encryption are inactive or not. The existence of such trails strengthens the validity of the attacks on 5-round and 6-round AES discussed in this Chapter 6. The pair of plaintexts p_1, p_2, the key and other intermediate states are stipulated in hexadecimal form.

$$p_1 = \begin{bmatrix} \texttt{E8} & \texttt{0} & \texttt{0} & \texttt{0} \\ \texttt{0} & \texttt{77} & \texttt{0} & \texttt{0} \\ \texttt{0} & \texttt{0} & \texttt{91} & \texttt{0} \\ \texttt{0} & \texttt{0} & \texttt{0} & \texttt{BF} \end{bmatrix} \qquad p_2 = \begin{bmatrix} \texttt{AC} & \texttt{0} & \texttt{0} & \texttt{0} \\ \texttt{0} & \texttt{7D} & \texttt{0} & \texttt{0} \\ \texttt{0} & \texttt{0} & \texttt{18} & \texttt{0} \\ \texttt{0} & \texttt{0} & \texttt{0} & \texttt{3F} \end{bmatrix}$$

$$key = \begin{bmatrix} \texttt{A0} & \texttt{85} & \texttt{AF} & \texttt{1B} \\ \texttt{39} & \texttt{9C} & \texttt{95} & \texttt{4B} \\ \texttt{79} & \texttt{29} & \texttt{EB} & \texttt{60} \\ \texttt{34} & \texttt{7E} & \texttt{D7} & \texttt{8A} \end{bmatrix}$$

- Initial difference of p_1 and p_2.

$$\begin{bmatrix} \texttt{44} & \texttt{0} & \texttt{0} & \texttt{0} \\ \texttt{0} & \texttt{0A} & \texttt{0} & \texttt{0} \\ \texttt{0} & \texttt{0} & \texttt{89} & \texttt{0} \\ \texttt{0} & \texttt{0} & \texttt{0} & \texttt{80} \end{bmatrix}$$

- Difference of intermediate states after 4 rounds of encryption (excluding the last mixcolumns operation).

$$\begin{bmatrix} 61 & B5 & EB & 16 \\ E7 & 7E & 0 & 0 \\ 37 & 0 & 2C & 0 \\ 0 & 79 & 17 & 0 \end{bmatrix}$$

- Difference of ciphertexts after 5 rounds of encryption.

$$\begin{bmatrix} 79 & 96 & BE & 76 \\ B1 & 96 & DE & F7 \\ E3 & 4B & 1A & 64 \\ 68 & 11 & 02 & 56 \end{bmatrix}$$

- Difference of states after swapping the last column between ciphertexts and subsequent 5 rounds of decryption.

$$\begin{bmatrix} 6A & 0 & 0 & 0 \\ 0 & 5D & 0 & 0 \\ 0 & 0 & A4 & 0 \\ 0 & 0 & 0 & 12 \end{bmatrix}$$

BIBLIOGRAPHY

[1] CAESAR Competition. https://competitions.cr.yp.to/caesar.html.

[2] eSTREAM: the ECRYPT Stream Cipher Project. https://www.ecrypt.eu.org/stream/.

[3] National Institute of Standards and Technology (NIST): AES Development (1997). https://www.cosic.esat.kuleuven.be/nessie/.

[4] National Institute of Standards and Technology (NIST): Lightweight cryp- tography standardization process (2019). https://csrc.nist.gov/projects/lightweight-cryptography.

[5] National Institute of Standards and Technology (NIST): SHA-3 Standardization Process (2007). https://csrc.nist.gov/projects/hash-functions/sha-3-project.

[6] New European Schemes for Signatures, Integrity, and Encryption (NESSIE). https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development.

[7] Revkit. https://msoeken.github.io/revkit.html.

[8] Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, 2001.

[9] Biryukov A and Khovratovich D. PAEQ v1. http://competitions.cr.yp.to/round1/paeqv1.pdf, 2014.

[10] Mohamed Ahmed Abdelraheem. Estimating the Probabilities of Low-Weight Differential and Linear Approximations on PRESENT-Like Ciphers. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *Information Security and Cryptology – ICISC 2012*, pages 368–382, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[11] Farzaneh Abed, Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. Biclique Cryptanalysis Of PRESENT, LED, And KLEIN. Cryptology ePrint Archive, Report 2012/591, 2012. https://eprint.iacr.org/2012/591.

[12] Zahra Ahmadian, Shahram Rasoolzadeh, Mahmoud Salmasizadeh, and Mohammad Reza Aref. Automated Dynamic Cube Attack on Block Ciphers: Cryptanalysis of SIMON and KATAN. *IACR Cryptology ePrint Archive*, 2015:40, 2015.

[13] Martin R. Albrecht and Gregor Leander. An All-In-One Approach to Differential Cryptanalysis for Small Block Ciphers. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography*, pages 1–15, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[14] Hoda A. Alkhzaimi and Martin M. Lauridsen. Cryptanalysis of the SIMON Family of Block Ciphers. Cryptology ePrint Archive, Report 2013/543, 2013. https://eprint.iacr.org/2013/543.

[15] Mishal Almazrooie, Azman Samsudin, Rosni Abdullah, and Kussay N. Mutter. Quantum Reversible Circuit of AES-128. *Quantum Information Processing*, 17(5):1–30, May 2018.

[16] M. Amy, D. Maslov, M. Mosca, and M. Roetteler. A Meet-in-the-Middle Algorithm for Fast Synthesis of Depth-Optimal Quantum Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, Jun 2013.

[17] Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, and John Schanck. Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3. In Roberto Avanzi and Howard Heys, editors, *Selected Areas in Cryptography – SAC 2016*, pages 317–337, Cham, 2017. Springer International Publishing.

[18] Ravi Anand, Subhamoy Maitra, Arpita Maitra, Chandra Sekhar Mukherjee, and Sourav Mukhopadhyay. Resource Estimation of Grovers-kind Quantum Cryptanalysis against FSR based Symmetric Ciphers. Cryptology ePrint Archive, Report 2020/1438, 2020. https://eprint.iacr.org/2020/1438.

[19] Elena Andreeva, Andrey Bogdanov, and Bart Mennink. Towards Understanding the Known-Key Security of Block Ciphers. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, pages 348–366, 2013.

[20] Michael Appel, Andreas Bossert, Steven Cooper, Tobias Kußmaul, Johannes Löffler, Christof Pauer, and Alexander Wiesmaier. Block ciphers for the iot– simon, speck, katan, led, tea, present, and sea compared.

[21] Jean-Philippe Aumasson. *Serious Cryptography: A Practical Introduction to Modern Encryption*. No Starch Press, USA, 2017.

[22] Jean-Philippe Aumasson and Willi Meier. Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. NIST mailing list, 2009. http://www.131002.net/data/papers/AM09.pdf.

[23] Roberto Avanzi. The QARMA Block Cipher Family. Almost MDS Matrices Over Rings With Zero Divisors, Nearly Symmetric Even-Mansour Constructions With Non-Involutory Central Rounds, and Search Heuristics for Low-Latency S-Boxes. *IACR Transactions on Symmetric Cryptology*, 2017(1):4–44, Mar. 2017.

[24] Nasour Bagheri, Reza Ebrahimpour, and Navid Ghaedi. New differential fault analysis on PRESENT. *EURASIP Journal on Advances in Signal Processing*, 2013(1):145, Sep 2013.

[25] Nasour Bagheri, Florian Mendel, and Yu Sasaki. Improved Rebound Attacks on AESQ: Core Permutation of CAESAR Candidate PAEQ. In *Proceedings, Part II, of the 21st Australasian Conference on Information Security and Privacy - Volume 9723*, pages 301–316, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[26] Gustavo Banegas, Daniel J. Bernstein, Iggy van Hoof, and Tanja Lange. Concrete Quantum Cryptanalysis of Binary Elliptic Curves. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):451–472, Dec. 2020.

[27] Subhadeep Banik, Jannis Bossert, Amit Jana, Eik List, Stefan Lucks, Willi Meier, Mostafizar Rahman, Dhiman Saha, and Yu Sasaki. Cryptanalysis of ForkAES. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 43–63, Cham, 2019. Springer International Publishing.

[28] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A Small Present - Towards Reaching the Limit of Lightweight Encryption. In *CHES*, 2017.

[29] Zhenzhen Bao, Jian Guo, and Eik List. Extended Truncated-differential Distinguishers on Round-reduced AES. *IACR Transactions on Symmetric Cryptology*, 2020(3):197–261, Sep. 2020.

[30] Achiya Bar-On, Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. Improved Key Recovery Attacks on Reduced-Round AES with Practical Data and Memory Complexities. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *LNCS*, pages 185–212. Springer, 2018.

[31] Navid Ghaedi Bardeh. A Key-Independent Distinguisher for 6-round AES in an Adaptive Setting. Cryptology ePrint Archive, Report 2019/945, 2019. https://ia.cr/2019/945.

[32] Navid Ghaedi Bardeh and Sondre Rønjom. Practical Attacks on Reduced-Round AES. In Johannes Buchmann, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology – AFRICACRYPT 2019*, pages 297–310, Cham, 2019. Springer International Publishing.

[33] Navid Ghaedi Bardeh and Sondre Rønjom. The Exchange Attack: How to Distinguish Six Rounds of AES with 2^88.2 Chosen Plaintexts. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 347–370. Springer, 2019.

[34] 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 *CRYPTO*, 2016.

[35] Daniel J. Bernstein and Bo-Yin Yang. Asymptotically Faster Quantum Algorithms to Solve Multivariate Quadratic Equations. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*, volume 10786 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2018.

[36] D.J. Bernstein and T. Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, September 2017.

[37] Eli Biham. New Types of Cryptanalytic Attacks Using Related Keys. *J. Cryptol.*, 7(4):229–246, December 1994.

[38] Eli Biham, Alex Biryukov, Orr Dunkelman, Eran Richardson, and Adi Shamir. Initial Observations on Skipjack: Cryptanalysis of Skipjack-3XOR. In Stafford Tavares and Henk Meijer, editors, *Selected Areas in Cryptography*, pages 362–375, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[39] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *LNCS*, pages 12–23. Springer, 1999.

[40] Eli Biham, Alex Biryukov, and Adi Shamir. Miss in the Middle Attacks on IDEA and Khufu. In Lars Knudsen, editor, *Fast Software Encryption*, pages 124–138, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[41] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. *J. Cryptol.*, 18(4):291–311, September 2005.

[42] Eli Biham, Orr Dunkelman, and Nathan Keller. The Rectangle Attack - Rectangling the Serpent. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '01, page 340–357, Berlin, Heidelberg, 2001. Springer-Verlag.

[43] Eli Biham, Orr Dunkelman, and Nathan Keller. New Results on Boomerang and Rectangle Attacks. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption*, pages 1–16, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[44] Eli Biham, Orr Dunkelman, and Nathan Keller. A Related-Key Rectangle Attack on the Full KASUMI. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, pages 443–461, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[45] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-Key Boomerang and Rectangle Attacks. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 507–525, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[46] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-Key Impossible Differential Attacks on 8-Round AES-192. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, pages 21–33, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[47] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[48] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptol.*, 4(1):3–72, 1991.

[49] Eli Biham and Adi Shamir. Differential Cryptanalysis of Feal and N-Hash. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 1991.

[50] Eli Biham and Adi Shamir. Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 156–171. Springer, 1991.

[51] Eli Biham and Adi Shamir. Differential Cryptanalysis of the Full 16-Round DES. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 487–496. Springer, 1992.

[52] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.

[53] Alex Biryukov. The Boomerang Attack on 5 and 6-Round Reduced AES. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard – AES*, pages 11–15, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[54] Alex Biryukov, Christophe De Cannière, and Gustaf Dellkrantz. Cryptanalysis of SAFER++. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 195–211. Springer, 2003.

[55] Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 1–18, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[56] Alex Biryukov and Dmitry Khovratovich. PAEQ: Parallelizable Permutation-Based Authenticated Encryption. In Sherman S. M. Chow, Jan Camenisch, Lucas C. K. Hui, and Siu Ming Yiu, editors, *Information Security*, pages 72–89, Cham, 2014. Springer International Publishing.

[57] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 231–249, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[58] Alex Biryukov, Gaëtan Leurent, and Léo Perrin. Cryptanalysis of Feistel Networks with Secret Round Functions. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography – SAC 2015*, pages 102–121, Cham, 2016. Springer International Publishing.

[59] C. Blondeau. B.: Links between theoretical and effective differential probabilities: Experiments on present. In *In: TOOLS'10. (2010)*.

[60] Céline Blondeau and Kaisa Nyberg. Links between Truncated Differential and Multidimensional Linear Properties of Block Ciphers and Underlying Attack Complexities. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 165–182, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[61] Céline Blondeau, Thomas Peyrin, and Lei Wang. Known-Key Distinguisher on Full PRESENT. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 455–474, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[62] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '07, page 450–466, Berlin, Heidelberg, 2007. Springer-Verlag.

[63] Andrey Bogdanov and Christian Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher ktantan. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *Selected Areas in Cryptography*, pages 229–240, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[64] Dan Boneh and Mark Zhandry. Quantum-Secure Message Authentication Codes. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 592–608, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[65] Dan Boneh and Mark Zhandry. Secure Signatures and Chosen Ciphertext Security in a Quantum Computing World. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 361–379, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[66] Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher. Quantum Attacks Without Superposition Queries: The Offline Simon's Algorithm. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 552–583. Springer, 2019.

[67] Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. Quantum Security Analysis of AES. *IACR Trans. Symmetric Cryptol.*, 2019(2):55–93, 2019.

[68] Julia Borghoff, Lars R. Knudsen, Gregor Leander, and Søren S. Thomsen. Cryptanalysis of PRESENT-Like Ciphers with Secret S-Boxes. In Antoine Joux, editor, *Fast Software Encryption*, pages 270–289, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[69] Johan Borst, Lars R. Knudsen, and Vincent Rijmen. Two Attacks on Reduced IDEA. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 1–13, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[70] Jannis Bossert, Eik List, Stefan Lucks, and Sebastian Schmitz. Pholkos – Efficient Large-state Tweakable Block Ciphers from the AES Round Function. Cryptology ePrint Archive, Report 2020/275, 2020. https://eprint.iacr.org/2020/275.

[71] Hamid Boukerrou, Paul Huynh, Virginie Lallemand, Bimal Mandal, and Marine Minier. On the Feistel Counterpart of the Boomerang Connectivity Table: Introduction and Analysis of the FBCT. *IACR Transactions on Symmetric Cryptology*, 2020(1):331–362, May 2020.

[72] Christina Boura and Anne Canteaut. On the Boomerang Uniformity of Cryptographic Sboxes. *IACR Trans. Symmetric Cryptol.*, 2018(3):290–310, 2018.

[73] Christina Boura, Marine Minier, María Naya-Plasencia, and Valentin Suder. Improved Impossible Differential Attacks against Round-Reduced LBlock. Research Report 2014/279, IACR Cryptology ePrint Archive, April 2014.

[74] Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and Improving Impossible Differential Attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 179–199, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[75] Joan Boyar, Magnus Find, and Rene Peralta. Small Low-Depth Circuits for Cryptographic Applications. 2018-03-24 2018.

[76] Michel Boyer, Gilles Brassard, Peter Hoyer, and Alain Tapp. Tight Bounds on Quantum Searching. Technical report, 1996.

[77] Harry Buhrman, Richard Cleve, Monique Laurent, Noah Linden, Alexander Schrijver, and Falk Unger. New Limits on Fault-Tolerant Quantum Computation. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 411–419, 2006.

[78] Stanislav Bulygin. More on linear hulls of PRESENT-like ciphers and a cryptanalysis of full-round EPCBC-96. Cryptology ePrint Archive, Report 2013/028, 2013. https://eprint.iacr.org/2013/028.

[79] Debrup Chakraborty and Palash Sarkar. HCH: A New Tweakable Enciphering Scheme Using the Hash-Encrypt-Hash Approach. In Rana Barua and Tanja Lange, editors, *Progress in Cryptology - INDOCRYPT 2006*, pages 287–302, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[80] Jiageng Chen, Je Sen Teh, Chunhua Su, Azman Samsudin, and Junbin Fang. Improved (related-key) Attacks on Round-Reduced KATAN-32/48/64 Based on the Extended Boomerang Framework. In Joseph K. Liu and Ron Steinfeld, editors, *Information Security and Privacy*, pages 333–346, Cham, 2016. Springer International Publishing.

[81] Joo Yeon Cho. Linear Cryptanalysis of Reduced-Round PRESENT. In Josef Pieprzyk, editor, *Topics in Cryptology - CT-RSA 2010*, pages 302–317, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[82] C. Cid, S. Murphy, and M. J. B. Robshaw. Small Scale Variants of the AES. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption*, pages 145–162, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[83] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang Connectivity Table: A New Cryptanalysis Tool. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT II*, volume 10821 of *LNCS*, pages 683–714. Springer, 2018.

[84] B. Collard and F. X. Standaert. A Statistical Saturation Attack against the Block Cipher PRESENT. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, pages 195–210, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[85] D. Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. *IBM J. Res. Dev.*, 38:243–250, 1994.

[86] Joan Daemen, Mario Lamberger, Norbert Pramstaller, Vincent Rijmen, Frederik Vercauteren, Esat Cosic, K U Leuven, Louvain , and Belgium . Computational aspects of the expected differential probability of 4-round AES and AES-like ciphers. 85:85–104, 06 2009.

[87] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag, Berlin, Heidelberg, 2002.

[88] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[89] Joan Daemen and Vincent Rijmen. Understanding Two-Round Differentials in AES. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *LNCS*, pages 78–94. Springer, 2006.

[90] Ivan Damgård, Jakob Funder, Jesper Buus Nielsen, and Louis Salvail. Superposition Attacks on Cryptographic Protocols. In Carles Padró, editor, *Information Theoretic Security*, pages 142–161, Cham, 2014. Springer International Publishing.

[91] Christophe De Cannière, Orr Dunkelman, and Miroslav Knežević. KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 272–288, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[92] Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. Catching the Fastest Boomerangs: Application to SKINNY. *IACR Transactions on Symmetric Cryptology*, 2020(4):104–129, Dec. 2020.

[93] Rishi Dewan. Advanced-Encryption-Standard-Algorithm. https://github.com/rishidewan33/Advanced-Encryption-Standard-Algorithm.

[94] Eduardo Marsola do Nascimento and José Antônio Moreira Xexéo. A Flexible Authenticated Lightweight Cipher using Even-Mansour Construction. In *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*, pages 1–6, 2017.

[95] Eduardo Marsola do Nascimento and José Antônio Moreira Xexéo. A Lightweight Cipher with Integrated Authentication. In *CONCURSO DE TESES E DISSERTAÇÕES - SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS (SBSEG), 18., 2018*, 2018.

[96] Eduardo Marsola do Nascimento and José Antônio Moreira Xexéo. Flex-AEAD -A Lightweight Cipher with Integrated Authentication. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/FlexAEAD-spec.pdf, 2019.

[97] E.M. do Nascimento. *Algoritmo de Criptografia Leve com Utilização de Autenticação*. PhD thesis, Instituto Militar de Engenharia, Rio de Janeiro, 2017.

[98] Alexandre Duc, Jian Guo, Thomas Peyrin, and Lei Wei. Unaligned Rebound Attack: Application to Keccak. In Anne Canteaut, editor, *Fast Software Encryption*, pages 402–421, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[99] Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. The Retracing Boomerang Attack. In *EUROCRYPT (1)*, volume 12105 of *Lecture Notes in Computer Science*, pages 280–309. Springer, 2020.

[100] Orr Dunkelman, Nathan Keller, and Adi Shamir. A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 393–410, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[101] Orr Dunkelman, Nathan Keller, and Adi Shamir. A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. *J. Cryptology*, 27(4):824–849, 2014.

[102] Avijit Dutta and Mridul Nandi. Tweakable HCTR: A BBB Secure Tweakable Enciphering Scheme. In Debrup Chakraborty and Tetsu Iwata, editors, *Progress in Cryptology – INDOCRYPT 2018*, pages 47–69, Cham, 2018. Springer International Publishing.

[103] Maria Eichlseder, Daniel Kales, and Markus Schofnegger. Forgery Attacks on FlexAE and FlexAEAD. Cryptology ePrint Archive, Report 2019/679, 2019. https://eprint.iacr.org/2019/679.

[104] Maria Eichlseder, Daniel Kales, and Markus Schofnegger. Official Comment: FleaxAEAD. Posting on the NIST LWC mailing list, 2019.

[105] Sareh Emami, San Ling, Ivica Nikolić, Josef Pieprzyk, and Huaxiong Wang. The resistance of present-80 against related-key differential attacks. *Cryptography and Communications*, 6(3):171–187, Sep 2014.

[106] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting. Improved Cryptanalysis of Rijndael. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, pages 213–230, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[107] Ewan Fleischmann, Michael Gorski, and Stefan Lucks. Attacking 9 and 10 Rounds of AES-256. In Colin Boyd and Juan González Nieto, editors, *Information Security and Privacy*, pages 60–72, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[108] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3), Sep 2012.

[109] Thomas Fuhr and Brice Minaud. Match Box Meet-in-the-Middle Attack Against KATAN. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption*, pages 61–81, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[110] Tommaso Gagliardoni. Quantum Security of Cryptographic Primitives. *CoRR*, abs/1705.02417, 2017.

[111] Tommaso Gagliardoni, Andreas Hülsing, and Christian Schaffner. Semantic Security and Indistinguishability in the Quantum World. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 60–89, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[112] Sebati Ghosh and Palash Sarkar. Breaking Tweakable Enciphering Schemes using Simon's Algorithm. *Des. Codes Cryptogr.*, 89(8):1907–1926, 2021.

[113] Henri Gilbert. A Simplified Representation of AES. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 200–222, 2014.

[114] Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers*, pages 365–383, 2010.

[115] Michael Gorski and Stefan Lucks. New Related-Key Boomerang Attacks on AES . In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology - INDOCRYPT 2008*, pages 266–278, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[116] Lorenzo Grassi. Mixture Differential Cryptanalysis: A New Approach to Distinguishers and Attacks on round-reduced AES. *IACR Trans. Symmetric Cryptol.*, 2018(2):133–160, 2018.

[117] Lorenzo Grassi and Christian Rechberger. New and Old Limits for AES Known-Key Distinguishers. Cryptology ePrint Archive, Report 2017/255, 2017. https://eprint.iacr.org/2017/255.

[118] Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. Subspace Trail Cryptanalysis and its Applications to AES. *IACR Transactions on Symmetric Cryptology*, 2016(2):192–225, Feb. 2017.

[119] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grover's Algorithm to AES: Quantum Resource Estimates. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography*, pages 29–43, Cham, 2016. Springer International Publishing.

[120] Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[121] Akinori Hosoyamada and Yu Sasaki. Quantum Demiric-Selçuk Meet-in-the-Middle Attacks: Applications to 6-Round Generic Feistel Constructions. *In: Catalano D., De Prisco R. (eds) Security and Cryptography for Networks. SCN 2018. Lecture Notes in Computer Science, vol 11035. Springer, Cham*, 2018, 2018.

[122] Thomas Häner, Damian S Steiger, Krysta Svore, and Matthias Troyer. A Software Methodology for Compiling Quantum Programs. *Quantum Science and Technology*, 3(2):020501, Feb 2018.

[123] Takanori Isobe, Yu Sasaki, and Jiageng Chen. Related-Key Boomerang Attacks on KATAN32/48/64. In Colin Boyd and Leonie Simpson, editors, *Information Security and Privacy*, pages 268–285, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[124] Takanori Isobe and Kyoji Shibutani. All Subkeys Recovery Attack on Block Ciphers: Extending Meet-in-the-Middle Approach. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography*, pages 202–221, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[125] Takanori Isobe and Kyoji Shibutani. Improved All-Subkeys Recovery Attacks on FOX, KATAN and SHACAL-2 Block Ciphers. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 104–126. Springer, 2014.

[126] Gembu Ito, Akinori Hosoyamada, Ryutaroh Matsumoto, Yu Sasaki, and Tetsu Iwata. Quantum Chosen-Ciphertext Attacks against Feistel Ciphers. In *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings*, pages 391–411, 2019.

[127] Aayush Jain, Varun Kohli, and Girish Mishra. Deep Learning based Differential Distinguisher for Lightweight Cipher PRESENT. Cryptology ePrint Archive, Report 2020/846, 2020. https://eprint.iacr.org/2020/846.

[128] Kyoungbae Jang, Seungju Choi, Hyeokdong Kwon, Hyunji Kim, Jaehoon Park, and Hwajeong Seo. Grover on Korean Block Ciphers. *Applied Sciences*, 10(18), 2020.

[129] Kyoungbae Jang, Hyunjun Kim, Siwoo Eum, and Hwajeong Seo. Grover on gift. Cryptology ePrint Archive, Report 2020/1405, 2020. https://eprint.iacr.org/2020/1405.

[130] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing Grover Oracles for Quantum Key Search on AES and LowMC. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 280–310, Cham, 2020. Springer International Publishing.

[131] Samuel Jaques and John M. Schanck. Quantum Cryptanalysis in the RAM Model: Claw-Finding Attacks on SIKE. 11692:32–61, 2019.

[132] Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Multiple Limited-Birthday Distinguishers and Applications. In *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, pages 533–550, 2013.

[133] Cody Jones. Low-overhead constructions for the fault-tolerant Toffoli gate. *Phys. Rev. A*, 87:022328, Feb 2013.

[134] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking Symmetric Cryptosystems Using Quantum Period Finding. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 207–237, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[135] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Quantum Differential and Linear Cryptanalysis. *IACR Trans. Symmetric Cryptol.*, 2016(1):71–94, 2016.

[136] John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In *Proceedings of the 7th International Workshop on Fast Software Encryption*, FSE '00, page 75–93, Berlin, Heidelberg, 2000. Springer-Verlag.

[137] Jongsung Kim, Guil Kim, Seokhie Hong, Sangjin Lee, and Dowon Hong. The Related-Key Rectangle Attack – Application to SHACAL-1. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy*, pages 123–136, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[138] Panjin Kim, Daewan Han, and Kyung Chul Jeong. Time–space Complexity of Quantum Search Algorithms in Symmetric Cryptanalysis: Applying to AES and SHA-2. *Quantum Information Processing*, 17(12), Oct 2018.

[139] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, pages 130–145, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[140] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional Differential Cryptanalysis of Trivium and KATAN. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, pages 200–212, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[141] Lars R Knudsen. Truncated and higher order differentials. In *International Workshop on Fast Software Encryption*, pages 196–211. Springer, 1994.

[142] Lars R. Knudsen and Vincent Rijmen. Known-Key Distinguishers for Some Block Ciphers. In *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, pages 315–324, 2007.

[143] Lars R. Knudsen and Matthew J. B. Robshaw. *The Block Cipher Companion*. Springer Publishing Company, Incorporated, 2011.

[144] Stefan Kölbl, Martin M. Lauridsen, Florian Mendel, and Christian Rechberger. Haraka v2 - Efficient Short-Input Hashing for Post-Quantum Applications. Cryptology ePrint Archive, Report 2016/098, 2016. https://eprint.iacr.org/2016/098.

[145] H. Kuwakado and M. Morii. Quantum Distinguisher between the 3-round Feistel cipher and the Random Permutation. In *2010 IEEE International Symposium on Information Theory*, pages 2682–2685, 2010.

[146] H. Kuwakado and M. Morii. Security on the quantum-type Even-Mansour cipher. In *2012 International Symposium on Information Theory and its Applications*, pages 312–316, Oct 2012.

[147] Lucia Lacko-Bartošová. Algebraic Cryptanalysis of Present Based on the Method of Syllogisms. *Tatra Mountains Mathematical Publications*, 53(1):201–212, 2013.

[148] B. Langenberg, H. Pham, and R. Steinwandt. Reducing the Cost of Implementing the Advanced Encryption Standard as a Quantum Circuit. *IEEE Transactions on Quantum Engineering*, 1:1–12, 2020.

[149] Martin M. Lauridsen and Christian Rechberger. Linear Distinguishers in the Key-less Setting: Application to PRESENT. In Gregor Leander, editor, *Fast Software Encryption*, pages 217–240, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[150] Gregor Leander. On Linear Hulls, Statistical Saturation Attacks, PRESENT and a Cryptanalysis of PUFFIN. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 303–322, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[151] Gregor Leander and Alexander May. Grover Meets Simon – Quantumly Attacking the FX-construction. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 161–178, Cham, 2017. Springer International Publishing.

[152] Changhoon Lee. Biclique Cryptanalysis of PRESENT-80 and PRESENT-128. *J. Supercomput.*, 70(1):95–103, October 2014.

[153] Li Lin, Wenling Wu, and Yafei Zheng. Improved Meet-in-the-Middle Distinguisher on Feistel Schemes. In *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, pages 122–142, 2015.

[154] Guo-Qiang Liu and Chen-Hui Jin. Differential cryptanalysis of PRESENT-like cipher. *Designs, Codes and Cryptography*, 76(3):385–408, Sep 2015.

[155] Guo-Qiang Liu and Chen-Hui Jin. Linear Cryptanalysis of PRESENT-like Ciphers with Secret Permutation. *The Computer Journal*, 59(4):549–558, 09 2015.

[156] Guoqiang Liu, Chenhui Jin, and Zhiyin Kong. Key recovery attack for present using slender-set linear cryptanalysis. *Science China Information Sciences*, 59(3):32110, Jan 2016.

[157] Jiqiang Lu, Orr Dunkelman, Nathan Keller, and Jongsung Kim. New Impossible Differential Attacks on AES. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology - INDOCRYPT 2008*, pages 279–293, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[158] Haoxiang Luo, Weijian Chen, Xinyue Ming, and Yifan Wu. General differential fault attack on present and gift cipher with nibble. *IEEE Access*, 9:37697–37706, 2021.

[159] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 386–397, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[160] Olivia Di Matteo, Vlad Gheorghiu, and Michele Mosca. Fault-Tolerant Resource Estimation of Quantum Random-Access Memories. *IEEE Transactions on Quantum Engineering*, 1:1–13, 2020.

[161] Alexandre Mege. Official Comment: FLEXAEAD. Posting on the NIST LWC mailing list, 2019.

[162] Willi Meier. On the Security of the IDEA Block Cipher. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 371–385, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[163] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In Orr Dunkelman, editor, *Fast Software Encryption*, pages 260–276, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[164] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Rebound Attacks on the Reduced Grøstl Hash Function. In Josef Pieprzyk, editor, *Topics in Cryptology - CT-RSA 2010*, pages 350–365, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[165] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., USA, 1st edition, 1996.

[166] Bart Mennink and Bart Preneel. On the Impact of Known-Key Attacks on Hash Functions. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 59–84, 2015.

[167] Marine Minier. Improving Impossible-Differential Attacks against Rijndael-160 and Rijndael-224. *Designs, Codes and Cryptography*, 82(1):117–129, Jan 2017.

[168] Sean Murphy. The Return of the Cryptographic Boomerang. *IEEE Trans. Information Theory*, 57(4):2517–2521, 2011.

[169] National Institute of Standards and Technology. FIPS 197. *National Institute of Standards and Technology, November*, pages 1–51, 2001.

[170] National Institute of Standards and Technology. SHA-3 : Cryptographic hash algorithm competition. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html.

[171] National Institute of Standards and Technology. Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process, 2016. https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria)#FN3.

[172] Kenji Ohkuma. Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis. In Michael J. Jacobson, Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, pages 249–265, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[173] Onur Özen, Kerem Varıcı, Cihangir Tezcan, and Çelebi Kocair. Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT. In Colin Boyd and Juan González Nieto, editors, *Information Security and Privacy*, pages 90–107, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[174] Goutam Paul and Souvik Ray. On data complexity of distinguishing attacks versus message recovery attacks on stream ciphers. *Des. Codes Cryptogr.*, 86(6):1211–1247, 2018.

[175] Mostafizar Rahman and Goutam Paul. Quantum Attacks on HCTR and its Variants. *IEEE Transactions on Quantum Engineering*, 2020.

[176] Sondre Rønjom, Navid Ghaedi Bardeh, and Tor Helleseth. Yoyo Tricks with AES. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 217–243, Cham, 2017. Springer International Publishing.

[177] Dhiman Saha. Slides on New Yoyo Tricks with AES-based Permutations, 2018.

[178] Dhiman Saha. Lecture Slides on Cryptography, IIT Bhilai, 2020.

[179] Dhiman Saha, Sourya Kakarla, Srinath Mandava, and Dipanwita Roy Chowdhury. Gain: Practical Key-Recovery Attacks on Round-reduced PAEQ. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 194–210, Cham, 2016. Springer International Publishing.

[180] Dhiman Saha, Mostafizar Rahman, and Goutam Paul. New Yoyo Tricks with AES-based Permutations. *IACR Transactions on Symmetric Cryptology*, 2018(4):102–127, Dec. 2018.

[181] Kazuo Sakiyama, Yu Sasaki, and Yang Li. *Security of Block Ciphers: From Algorithm Design to Hardware Implementation*. Wiley Publishing, 1st edition, 2015.

[182] Ali Aydın Selçuk. On Probability of Success in Linear and Differential Cryptanalysis. *Journal of Cryptology*, 21(1):131–147, Jan 2008.

[183] Peter Selinger. Quantum circuits of T-depth one. *Physical Review A*, 87(4), Apr 2013.

[184] Mohammad Hossein Faghihi Sereshgi, Mohammad Dakhilalian, and Mohsen Shakiba. Biclique cryptanalysis of mibs-80 and present-80. Cryptology ePrint Archive, Report 2015/393, 2015. https://eprint.iacr.org/2015/393.

[185] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.

[186] Danping Shi, Lei Hu, Siwei Sun, and Ling Song. Linear(hull) cryptanalysis of round-reduced versions of katan. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, pages 364–371. INSTICC, SciTePress, 2016.

[187] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.*, 26(5):1484–1509, October 1997.

[188] P.W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[189] Daniel R. Simon. On the Power of Quantum Computation. *SIAM J. Comput.*, 26(5):1474–1483, October 1997.

[190] H. Soleimany, A. Sharifi, and M. Aref. Improved Related-Key Boomerang Cryptanalysis of AES-256. In *2010 International Conference on Information Science and Applications*, pages 1–7, April 2010.

[191] Damian S. Steiger, Thomas Häner, and Matthias Troyer. ProjectQ: An Open Source Software Framework for Quantum Computing. *Quantum*, 2:49, Jan 2018.

[192] Bing Sun, Meicheng Liu, Jian Guo, Longjiang Qu, and Vincent Rijmen. New Insights on AES-Like SPN Ciphers. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 605–624, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[193] Ling Sun, Wei Wang, Ru Liu, and Meiqin Wang. MILP-aided bit-based division property for ARX ciphers. *Science China Information Sciences*, 61(11):1–3, 2018.

[194] Cihangir Tezcan. The Improbable Differential Attack: Cryptanalysis of Reduced Round CLEFIA. In Guang Gong and Kishan Chand Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010*, pages 197–209, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[195] Cihangir Tezcan. Improbable differential attacks on Present using undisturbed bits. *J. Computational Applied Mathematics*, 259:503–511, 2014.

[196] Cihangir Tezcan. Differential Factors Revisited: Corrected Attacks on PRESENT and SERPENT. In Tim Güneysu, Gregor Leander, and Amir Moradi, editors, *Lightweight Cryptography for Security and Privacy*, pages 21–33, Cham, 2016. Springer International Publishing.

[197] Cihangir Tezcan, Galip Oral Okan, Asuman Şenol, Erol Doğan, Furkan Yücebaş, and Nazife Baykal. Differential Attacks on Lightweight Block Ciphers PRESENT, PRIDE, and RECTANGLE Revisited. In Andrey Bogdanov, editor, *Lightweight Cryptography for Security and Privacy*, pages 18–32, Cham, 2017. Springer International Publishing.

[198] Cihangir Tezcan and Ali Aydin Selçuk. Improved Improbable Differential Attacks on ISO Standard CLEFIA: Expansion Technique Revisited. *Inf. Process. Lett.*, 116:136–143, 2016.

[199] Tyge Tiessen, Lars R. Knudsen, Stefan Kölbl, and Martin M. Lauridsen. Security of the AES with a Secret S-Box. In Gregor Leander, editor, *Fast Software Encryption*, pages 175–189, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[200] Michael Tunstall. Improved "Partial Sums"-based Square Attack on AES. In Pierangela Samarati, Wenjing Lou, and Jianying Zhou, editors, *SECRYPT 2012 - Proceedings of the International Conference on Security and Cryptography, Rome, Italy, 24-27 July, 2012, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 25–34. SciTePress, 2012.

[201] David Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.

[202] Gaoli Wang and Shaohui Wang. Differential Fault Analysis on PRESENT Key Schedule. In *2010 International Conference on Computational Intelligence and Security*, pages 362–366, 2010.

[203] Haoyang Wang and Thomas Peyrin. Boomerang Switch in Multiple Rounds. Application to AES Variants and Deoxys. *IACR Trans. Symmetric Cryptol.*, 2019(1):142–169, 2019.

[204] Meiqin Wang. Differential Cryptanalysis of Reduced-Round PRESENT. In Serge Vaudenay, editor, *Progress in Cryptology – AFRICACRYPT 2008*, pages 40–49, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[205] Meiqin Wang, Yue Sun, Elmar Tischhauser, and Bart Preneel. A Model for Structure Attacks, with Applications to PRESENT and Serpent. In Anne Canteaut, editor, *Fast Software Encryption*, pages 49–68, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[206] Peng Wang, Dengguo Feng, and Wenling Wu. HCTR: A Variable-Input-Length Enciphering Mode. In Dengguo Feng, Dongdai Lin, and Moti Yung, editors, *Information Security and Cryptology*, pages 175–188, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[207] John Watrous. Introduction to Quantum Computing, 2005.

[208] Shengbao Wu and Mingsheng Wang. Integral Attacks on Reduced-Round PRESENT. In Sihan Qing, Jianying Zhou, and Dongmei Liu, editors, *Information and Communications Security*, pages 331–345, Cham, 2013. Springer International Publishing.

[209] Wen-Ling Wu, Wen-Tao Zhang, and Deng-Guo Feng. Impossible Differential Cryptanalysis of Reduced-Round ARIA and Camellia. *J. Comput. Sci. Technol.*, 22(3):449–456, May 2007.

[210] Wenling Wu, Lei Zhang, and Wentao Zhang. Improved Impossible Differential Cryptanalysis of Reduced-Round Camellia. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, pages 442–456, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[211] Akihiro Yamamura and Hirokazu Ishizuka. Quantum Cryptanalysis of Block Ciphers (Algebraic Systems, Formal Languages and Computations.). Technical report, 2000.

[212] Lin Yang, Meiqin Wang, and Siyuan Qiao. Side Channel Cube Attack on PRESENT. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *Cryptology and Network Security*, pages 379–391, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[213] Christof Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60(4):2746–2751, Oct 1999.

[214] Mark Zhandry. How to Construct Quantum Random Functions. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 679–687, 2012.

[215] Jing Zhang, Dawu Gu, Zheng Guo, and Lei Zhang. Differential power cryptanalysis attacks against PRESENT implementation. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 6, pages V6–61–V6–65, 2010.

[216] Wentao Zhang, Wenling Wu, and Dengguo Feng. New Results on Impossible Differential Cryptanalysis of Reduced AES. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *Information Security and Cryptology - ICISC 2007*, pages 239–250, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[217] XinJie Zhao, Tao Wang, and ShiZe Guo. Improved Side Channel Cube Attacks on PRESENT. Cryptology ePrint Archive, Report 2011/165, 2011. https://eprint.iacr.org/2011/165.

[218] Bo Zhu and Guang Gong. Multidimensional Meet-in-the-middle Attack and its Applications to KATAN32/48/64. *Cryptogr. Commun.*, 6(4):313–333, 2014.