DIFFERENTIAL-LINEAR CRYPTANALYSIS OF ASCON AND DRYGASCON

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ASLI BAŞAK CIVEK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CYBER SECURITY

JUNE 2021

Approval of the thesis:

DIFFERENTIAL-LINEAR CRYPTANALYSIS OF ASCON AND DRYGASCON

submitted by **ASLI BAŞAK CIVEK** in partial fulfillment of the requirements for the degree of **Master of Science in Cyber Security Department, Middle East Technical University** by,

Prof. Dr. Deniz ZEYREK BOZŞAHİN
Dean, **Graduate School of Informatics** _____

Assist. Prof. Dr. Cihangir TEZCAN
Head of Department, **Cyber Security** _____

Assist. Prof. Dr. Cihangir TEZCAN
Supervisor, **Cyber Security Department** _____

Examining Committee Members:

Assoc. Prof. Dr. Cengiz ACARTÜRK
Cognitive Science Dept., METU _____

Assist. Prof. Dr. Cihangir TEZCAN
Cyber Security Dept., METU _____

Prof. Dr. Ali Aydın SELÇUK
Computer Engineering Dept., TOBB ETÜ _____

Date: 14.06.2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Aslı Başak Civek

Signature :

ABSTRACT

DIFFERENTIAL-LINEAR CRYPTANALYSIS OF ASCON AND DRYGASCON

Civek, Aslı Başak
M.S., Department of Cyber Security
Supervisor: Assist. Prof. Dr. Cihangir TEZCAN

June 2021, 72 pages

Due to rapidly developing technology, devices have become smaller along with their performance capacity and memory. If possible, existing NIST-approved encryption standards should be used on these resource-constrained devices. When an acceptable performance cannot be achieved in this way, there is a need for more lightweight algorithms. Since taking individual measures leads to simplistic designs when designing lightweight algorithms, ciphers can become more vulnerable to cryptographic attacks. Hence some regulation is necessary. To satisfy this need, NIST has decided to start a lightweight cryptography competition to select one or more lightweight algorithms.

In this study, we examined Second Round NIST Lightweight Cryptography Standardization Competition candidates to contribute to the course of the competition. Then we focused on two different but structurally very similar cipher suites Ascon and Drygascon to compare their security. We observed 2, 3, 3.5-round truncated differential and 5-round differential-linear distinguishers that were given for Drygascon are erroneous. We present the corrected results and provide the longest practical differential-linear distinguisher of Drygascon. After that, we compared the security of Ascon and Drygascon. We observed that the practical data complexity of the two

is very close. However, since Ascon has more rounds than Drygascon, we concluded that Ascon might be more resistant against differential-linear cryptanalysis.

Keywords: lightweight cryptography, cryptanalysis, differential-linear analysis, nist

ÖZ

ASCON VE DRYGASCON ŞİFRELERİNİN DİFERANSİYEL-LİNEER KRİPTANALİZİ

Civek, Aslı Başak
Yüksek Lisans, Siber Güvenlik Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Cihangir TEZCAN

Haziran 2021 , 72 sayfa

Hızla gelişen teknoloji nedeniyle cihazlar performans kapasiteleri ve bellekleri ile birlikte küçülmüştür. Mümkünse bu kaynak kısıtlı cihazlarda var olan NIST-onaylı şifreleme standartları kullanılmalıdır. Bu şekilde kabul edilebilir bir performans sağlanamadığında daha hafif algoritmalara ihtiyaç duyulmaktadır. Hafif-sıklet algoritmaları tasarlarken alınan önlemler tasarımları basitleştirebilmekte ve şifrelerin kriptografik saldırılara karşı daha zayıf hale gelmesine neden olabilmektedir. Bu nedenle bazı düzenlemeler gereklidir. Bu ihtiyacı karşılamak adına NIST, bir veya daha fazla hafif-sıklet algoritma seçmek için bir hafif-sıklet kriptografi yarışması başlatmaya karar vermiştir.

Bu çalışmada, yarışmanın gidişatına katkı sağlamak amacıyla İkinci Tur NIST Hafif Kriptografi Standardizasyon Yarışması adaylarını inceledik. Ardından, güvenliklerini karşılaştırmak için iki farklı ancak yapısal olarak çok benzer şifre paketi Ascon ve Drygascon'a odaklandık. Drygascon için verilen 2, 3, 3.5 tur kesik diferansiyel ve 5-tur diferansiyel-lineer ayırt edicilerinin hatalı olduğunu gözlemledik. Düzeltilmiş sonuçları sunuyor ve Drygascon için en uzun pratik diferansiyel-lineer ayırt ediciyi

sağlıyoruz. Ardından Ascon ve Drygascon'un güvenliğini karşılaştırdık ve pratikteki veri karmaşıklıklarının çok yakın seviyede olduğunu gözlemledik. Ancak, Ascon'un tur sayısı Drygascon'dan daha fazla olduğundan, Ascon'un diferansiyel-lineer kriptanalize karşı daha dayanıklı olabileceği sonucuna vardık.

Anahtar Kelimeler: hafif sıklet kriptografi, kriptanaliz, diferansiyel-lineer analiz, nist

to my mother

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my thesis advisor Assist. Prof. Dr. Cihangir TEZCAN who introduced me to cryptanalysis. Throughout the thesis, he answered my endless questions patiently and helped me repeatedly without hesitation when I have a hard time to understand some subjects. His fast correspondence, understanding, and insight have been a great help for me during this study. I am lucky to have him as not just my advisor, but my mentor.

I would also like to thank the academic personnel in the Department of Cyber Security, but especially Assoc. Prof. Dr. Cengiz ACARTÜRK. He always helped me in any matter and provided his guidance without hesitation whenever I needed it. His wisdom, point of view, and enthusiasm for science have always broadened my perspective. It has been a great pleasure to be in his lectures.

I would like to thank Sebastian Riou for sharing some codes he used in his analysis of Drygascon. Being able to review these codes helped me to advance this study.

I would like to thank my family for their support. But my special thanks go to my mother. She was always there for me with her love and compassion. She always lifted me whenever I was down and supported me in any matter. I am grateful for everything she has done for me.

Last but not least, I would like to thank all my friends who taught me to be calm and make fun of life. I am grateful for their love, patience, and support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
AE	Authenticated Encryption
AEAD	Authenticated Encryption with Associated Data
CAESAR	Competition for Authenticated Encryption: Security, Applicability, and Robustness
DES	Data Encryption Standard
DESL	Lightweight Data Encryption Standard
GE	Gate Equivalent
ID	Identification Number
IoT	Internet of Things
ISO	International Organization for Standardization
LWC	Lightweight Cryptography
NIST	National Institute of Standards and Technology
RFID	Radio Frequency Identification
SHA	Secure Hashing Algortihm
TDEA	Triple Data Encryption Algorithm

CHAPTER 1

INTRODUCTION

1.1 Cryptography and Cryptanalysis

Today the point where technology has come makes life easier in many ways. But although being advantageous, it brings some security concerns. Because since the communication carries through in an insecure channel, the data could be captured or altered by third parties. Although the risks seem to be greater today, the privacy of the data is actually not a new issue. It has existed since the beginning of written communication. Therefore, thousands of years ago, the science of cryptology was born to provide secure communication in an insecure channel. It is not just used for confidentiality, but integrity, limiting unauthorized access to private information, and non-repudiation.

Today, cryptography tries to achieve confidentiality by using some collection of operations and algorithms called cryptosystem. Cryptosystem includes an encryption algorithm, namely the *cipher*. Then the private data, namely the *plaintext*, is complicated by passing it through this algorithm with some secret information called *secret key*. In this way, plaintext becomes a random-looked sequence called *ciphertext* for protecting it from adversaries. According to [Shannon, 1949], an encryption algorithm should include confusion and diffusion steps to obscure the relation between ciphertext, plaintext, and the secret key. It is important that cipher algorithms are publicly available. What matters is keeping the key secret. Because according to Shannon, the system already is accessible by the adversaries through bribes, burglary, and blackmail. Therefore it should be approached with the assumption that the enemy already knows the system [Shannon, 1949].

The security of a cryptosystem is based on keeping the key secret. The secret key can be used both for encryption and decryption if this is a symmetric cryptosystem. But asymmetric cryptosystems use different keys for these processes. Also, there are keyless algorithms like hash functions, which take arbitrary-length input bits and turn them into fixed-size output bits. Those hash functions are used in areas like generating digital signatures, message authentication codes, and many more.

Symmetric cryptosystems use the same key or closely related keys for encryption and decryption. They contain encryption primitives such as block ciphers and stream ciphers. Block ciphers basically divide the input into $b - bit$ blocks, encrypt each of them with a secret key and provide $b - bit$ ciphertext blocks. Then various modes of operations can be used to combine these output blocks. Stream ciphers operate on small input blocks, and sometimes these blocks can be small as a single bit. They transform these bits in a timely manner by using their state.

Block ciphers, a type of symmetric cryptosystem, can be categorized as Substitution Permutation Network (SPN) and Feistel Network. SPN and Feistel Network have some layers to provide confusion and diffusion for the cipher. The combination of these layers can be called a *round*, and a cipher depends on applying this round r times. SPN contains a key addition layer, a substitution layer that provides confusion, and a permutation (linear) layer that provides diffusion to the $b - bit$ block input. A subkey is determined in each round by using a key scheduling algorithm that processes the secret key. But in Feistel Network, $b - bit$ block input is divided into two parts. And while a round function with a key material is applied on one part, the output is XORed with the other part. Then the places of these parts are changed with a swap operation.

Block ciphers can be converted into hash functions, stream ciphers, and permutation-based constructions like sponge functions [Bertoni et al., 2007]. Sponge constructions [Bertoni et al., 2007] use fixed-permutations as primitives. Block ciphers can be used as iterative permutations in sponge functions if their secret keys are fixed. Because unlike block ciphers, sponge functions do not include a key-scheduling algorithm. Instead, they use a limited-length initial state that includes the secret key. By using this state, the message with the desired length can be encrypted.

Sponge constructions can be used in different ways. One of the ways to use sponge construction is Duplex construction [Bertoni et al., 2011]. Although some processes in duplex construction are applied differently, the security of both construction depends on the security of the used permutation. Therefore the security of this permutation should be thoroughly analyzed when used in these constructions.

Cryptography aims to provide secure communication in an insecure channel. But even though the communication seems obscure and random in plain sight after cryptographic operations, it is still possible to obtain information from it. This can be achieved via analyzing the cryptosystem and finding a repeating pattern on it. This is called cryptanalysis, and it aims to obtain information about the secret key, plaintext, or system by finding a statistical weakness in the cipher. Since the enemy may have knowledge about the system, the vulnerabilities in it should be analyzed and taken countermeasures at an early stage. In order to that, the algorithm must be public so it can take more analysis. And that way, the system can be more secure against adversaries. For this reason, competitions are held from time to time to select an encryption standard for different purposes. These competitions can last for a couple of years. During this time, candidate algorithms are open to the public and reviewed by cryptanalysts. As a result of this, the most optimal algorithms that can maintain a threshold between security, speed, and cost are selected and standardized. Advanced Encryption Standard (AES) [Nechvatal et al., 2001] can be given as an example of these competitions.

Rijndael [Daemen and Rijmen, 2002] is the winner of the AES competition that was held between 1997 and 2000. It was standardized in 2001 as AES cipher and it has continued to receive analyses since. It is a symmetric-key algorithm that uses SPN as a primitive. It has a block size of 128-bit and three different key sizes with minimum security of 128-bit. It can be used as 10, 12, 14 rounds with the key sizes of 128, 192, or 256 bits respectively. It is still frequently used as a strong encryption standard today, even used for protecting classified information [1]. To achieve reasonable security and performance, it should be used in all systems that require cryptographic operations if it is possible.

[1] https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/cnss15fs.pdf

1.2 Lightweight Cryptography

With the rapidly developing technology, there has been an increase in the production and use of constrained devices such as radio frequency identification devices (RFID), automotive systems, mobile tokens, medical implants, smart grid, sensor networks, distributed control systems, cyber-physical systems, and internet of things (IoT). These devices are generally interconnected and communicate wirelessly with one another. Some of these devices may not be able to use existing cryptographic standards effectively. Since the standardized algorithms were designed for environments like server/desktop most of the time, they may not meet the needs of some of these constrained devices. Or the constrained devices may not serve up acceptable performance criteria when they used them. So more lightweight designs were needed for those types of devices, nearly for 20 years.

Since the academia could not keep up with this need and did not receive such a request, they started working on the subject approximately 10 years after this requirement began. This led the industry to come up with solutions of its own by developing algorithms that were not carefully advanced or adequately analyzed. This approach caused some disasters.

An example would be the attacks performed on Mifare Classic 1k, a frequently used contactless smart card produced by NXP Semiconductors. It has wide usage areas such as public transportation, university identification number (ID) cards, electronic toll collection. To provide its security, the company used the Crypto1 cipher that was developed by its own team. Although the company has not disclosed the algorithm it uses to the public, the structure of the design [Nohl et al., 2008] and the communication layer [Garcia et al., 2008] have been discovered by academicians using reverse engineering. Later they showed that the weaknesses in its initial design led to the card being cloned. That can be done via a side-channel attack, which is an attack type that aims to implementation of a system, rather than the cipher algorithm. For this attack to work, it is enough to accomplish wireless interaction with the card for a few minutes [Meijer and Verdult, 2015]. The other method is brief eavesdropping of the card's communication [Tezcan, 2017] then applying an offline brute force attack. Brute force attack, or exhaustive search, basically means trying every possible key

that can be used on the system to obtain something intelligible. Although the newer versions have been hardened against that, the key should be randomly generated and the default one should not be used for the card to be secure.

Another example is the one for recovering the secret keys of car locks that use a lightweight Keeloq algorithm. Keeloq is a block cipher designed for hardware. It is used for keyless entry in places like cars, garages, apartment doors. This algorithm, which many famous car companies use, has been given many attacks in the literature. One of them is a type of side-channel attack called differential power analysis attack. For this attack, [Paar et al., 2009] stated that cloning a remote is possible from only ten power traces. Then they demonstrated that the secret key of the receiver and the remote transmitter can be obtained with a practical key recovery attack within a few minutes.

A similar example would be the attack that is done to Tesla Model S key fob [Wouters et al., 2019]. In this attack, researchers analyzed and performed reverse engineering to Passive Keyless Entry and Start systems used in this key fob. And they found out the system uses a 40-bit key for both the unlock and start of the car. Then they showed that the key fob could be cloned in seconds.

National Institute of Standards and Technology (NIST) is a non-regulatory organization within the United States Department of Commerce[2]. Its researchers can develop guidelines and standards with the help of the government, academia, and industry when no acceptable standard exists. It organized competitions before to choose the AES [Nechvatal et al., 2001] and Secure Hashing Algorithm-3 (SHA-3) [Dworkin, 2015]. In order to avoid greater disasters, NIST launched a project for lightweight cryptography in 2013. It aimed to analyze how existing NIST-approved algorithms work on constrained devices and if it is necessary to initiate a new standardization process. Then, it held two workshops in 2015 and 2016 in order to understand the needs of the industry, design principles of current lightweight cryptography algorithms, and the requirements of target devices. In 2017, NIST published a report [McKay et al., 2016]. One of the issues covered in this report was the examination of the lightweight algorithms that have been developed so far.

[2] https://www.nist.gov/director/pao/nist-general-information

According to report that is published by NIST [McKay et al., 2016], some lightweight designs [Biryukov and Perrin, 2018] emerged over the time. Some of these designs were developed by simplifying existing ones such as Lightweight Data Encryption Standard (DESL) [Leander et al., 2007], while others were made from scratch, such as PRESENT [Bogdanov et al., 2007]. In these lightweight designs, some measures were taken to meet the sufficient performance and limited memory requirements needed for constrained devices. These measures were the usage of smaller key sizes, simpler key schedules, minimal implementations, simpler rounds, smaller block sizes. Although these features increased performance, they also cause some security weaknesses. For example,

- While having smaller block sizes like 64 bits instead of 128 bits (e.g., PRESENT) helps save memory, it increases some attack risks like key recovery and plaintext recovery.
- Having smaller key sizes under the 112 bits (e.g., PRESENT-80, Enocoro-80 [Watanabe et al., 2008], Trivium [De Canniere, 2006]) not only increases the efficiency but also some attack risks like key recovery. The minimum recommended key size by NIST is currently 112 bits [Barker and Roginsky, 2011].
- Simpler rounds with 4-bit S-boxes over 8-bit ones were preferred in lightweight designs to save the area (e.g., PRESENT). Or instead of using complex linear layers in the hardware designs, recursive Maximum Distance Separable (MDS) matrices (e.g., PHOTON [Guo et al., 2011]), or bit permutations (e.g., PRESENT) were used. It was observed that this situation forces the iteration to be kept more to provide security.
- While using simpler key schedules decreases the power consumption, latency and helps to save the memory, it could lead to some key-related attack risks.

So, the designs made so far have not been successful in eliminating the need for lightweight cryptography. Because besides it was difficult to provide a single standard for all types of IoT devices; it was not an easy task to embed these algorithms into an IoT device. Furthermore, the design choices made did not only increase the performance but the attack possibility.

One of the other issues covered in this report [McKay et al., 2016] was the study of the applicability of the current NIST-approved algorithms like AES [Daemen and Rijmen, 1999] and Triple Data Encryption Algorithm TDEA [Barker and Mouha, 2017] that were approved by NIST on constrained devices. Furthermore, the needs of the target devices were also analyzed. According to this study, they came to the following conclusions:

- Most microcontrollers owned by IoT devices have a few simple instruction sets and limited memory to be cost-effective. In such an environment, even simple operations can cause too many instructions to be executed. On account of this, trying to run today's NIST-approved algorithms in IoT systems may lead to too many cycles to be executed. Therefore, applications could run too slow than intended, or their energy-consumption could be too high. So, powerful cryptographic systems like AES can be too demanding for IoT sensors that work with limited computational power and battery.

- Since most IoT systems are battery-powered and some, such as medical implants, cannot replace their batteries, it is crucial that their batteries last long enough. Not to mention devices that need stinted power from the environment to operate, like RFID tags. So, cryptographic algorithms that use small amounts of gate equivalents (GEs) and that have few timing and power requirements will be a better suit for such devices. Of course, it is sometimes possible to design these devices to use existing strong cryptographic algorithms. However, since this will require adding more memory and gates, this will increase the total cost quite much for devices that are produced in high volumes such as smart cards.

- Latency is an important topic for IoT systems that need to perform data exchange in a short time, like real-time automotive applications. So, this situation can be taken into account when designing the algorithm.

- Even though most of the time there is no need to produce too much data in a short time in IoT systems, still moderate throughput is expected most of the time. And it should also be taken into account that some IoT devices may need to communicate with devices in the high-end spectrum.

- Most importantly, since attackers may get physical access to the device, it may lead to side-channel analysis. Even though ciphers are secure, measuring the properties such as power consumption may cause attackers to capture the key. Therefore, the cryptographic algorithms for constrained devices should be designed as side-channel resistant.

As a result, NIST recommends implementing AES to the constrained devices if it is possible [McKay et al., 2016]. And well-analyzed lightweight algorithms are needed for constrained devices that cannot use existing standards and require simpler designs. Therefore, with the report [McKay et al., 2016] that was published in 2017, it was announced that a competition will be held to satisfy this need. The timeline of this process is presented in Table 1.1.

Table 1.1: Timeline of NIST's lightweight cryptography competition process. Table was adapted from https://csrc.nist.gov/Projects/lightweight-cryptography

Date	Event
July 20/21, 2015	First Workshop
August 11, 2016	Publication of NISTIR 8114 draft
October 17/18, 2016	Second Workshop
October 31, 2016	End of public comment period to Draft NISTIR 8114
March 28, 2017	NISTIR 8114 Report is published.
April 26, 2017	Profiles for standardization is published.
June 16, 2017	Public comments received.
May 14, 2018	Evaluation Criteria and requirements. of the process is published.
May 14, 2018	Federal Register Notice is published.
June 28, 2018	Public comment period to the requirements ended.
August 27, 2018	Federal Register Notice is published.
August 27, 2018	Evaluation Criteria and Submission Requirements for the process is published.
January 4, 2019	Early submission deadline for initial comments
February 25, 2019	Submission deadline
March 29, 2019	Amendment Deadline
April 18, 2019	Round 1 Candidates were announced.
August 30, 2019	Round 2 Candidates were announced.
October 7, 2019	NISTIR 8268 is published.
November 4/6, 2019	Third Workshop
October 19/21, 2020	Fourth Workshop
March 29, 2021	Finalist were announced.

NIST has not decided yet whether the winning algorithm should be a sole or a portfolio. The industry prefers the winner to be a single algorithm because then it will be sufficient for them to embed a single algorithm instead of many algorithms into their devices. In this way, they will be able to reduce the cost by saving memory. However, the academicians prefer it to be a portfolio because in that way, the initial algorithm could quickly be replaced with the other one in case of a significant weakness appears on it. And since IoT devices are different than each other and one standard may not be met with all of them, the best approach might be different lightweight ciphers for different purposes.

NIST published a call for submission document [NIST, 2018] for the candidates after decided to start a competition. According to this document, they stated that the submission packets should include a cover sheet, specification of the algorithm, intellectual property statements, source codes, and their test vectors compatible with the specification. They wanted [NIST, 2018] candidate algorithms to meet Authenticated Encryption with Associated Data (AEAD), an optional hash function, implementation, and design requirements. According to requirements, AEAD algorithms had to provide a minimum of 128-bit key security while the nonce is unique. Hash functions had to resist known attacks, able to process all byte-string inputs, and provide a minimum of 256-bit outputs. And both the AEAD algorithms and the hash functions had to have better performance than current standards in the constrained devices. And participants had to take countermeasures against side-channel attacks for their algorithms.

The first 57 candidate algorithms are submitted early in 2019. NIST briefly evaluated them under the criteria of if they fulfilled the requirements of the [NIST, 2018]. As a result of this, 56 of them were accepted to the first round in April 2019 [Turan et al., 2019]. NIST had decided to evaluate the candidate algorithms under the criteria of maturity of the submission packets, side-channel resistance, cost, performance, third-part security analysis [NIST, 2018]. In the cost part, the algorithms were going to be compared in terms of GE, memory, and energy that they required. In the performance part, they were going to be compared in terms of latency, power consumption, and throughput.

For the performance part, some benchmarks were done for both hardware and software. Software benchmarks are listed below:

- NIST published its benchmarking framework to utilize the performance of all candidate algorithms on microcontrollers [NIST, 2021].
- Most of the second-round candidates were evaluated for efficiency for 8-bit and 32-bit embedded structures, especially AVR/ARM [Weatherley, 2021].
- The benchmarks of speed and memory usage on MCUs for the candidates were presented in [Sebastian Renner and Mottok, 2020].
- Optimization strategies of some of the algorithms were analyzed on a RISC-V architecture. And some optimized implementations were offered [Fabio Campos and Viguier, 2020].
- Some benchmarks for Intel, AMD, ARM Cortex-A, Qualcomm processors were presented in [ECRYPT, 2019].

These benchmarks mostly include both the reports and programming codes for further analysis. Hardware benchmarks are listed below:

- Benchmarks of Field Programmable Gate Array (FPGA) for round 2 candidates are given in [Kamyar Mohajerani and Gaj, 2020].
- Application Specific Integrated Circuit (ASIC) implementations benchmark of round 2 candidates are given in [Mustafa Khairallah and Chattopadhyay, 2020].

On the security part, NIST asked the cryptography community to analyze these ciphers. In this way, it was going to be possible to detect cryptographically weak algorithms at an early stage and confirm the security of those that do not have known weaknesses. Based on this outcome, NIST was going to be able to eliminate the candidate algorithms in each round to select the winner in the end. This is also the reason why we carried out this study; to contribute to the course of the competition by analyzing the security of some of the candidate algorithms and providing our results.

NIST asked the cryptography community to analyze the security, performance, and implementation of these algorithms for four months instead of twelve months. Because it wanted this competition to be completed in a shorter time than other competitions like AES and SHA-3. These previous competitions were taken a long time to complete; four years for AES and five years for SHA-3. Therefore it decided to focus on promising algorithms in the first phase. At the end of the first round, 24 algorithms were eliminated. They can be seen in Table 1.2.

Table 1.2: The eliminated algorithms in the first round of NIST's Competition

State	Cipher	Source
Round 1	Bleep64	[Driscoll, 2018]
Round 1	CiliPadi	[Z'aba et al., 2019]
Round 1	CLAE	[Liu et al., 2019]
Round 1	CLX	[Wu and Huang, 2019a]
Round 1	FlexAEAD	[do Nascimento and Xexéo, 2019]
Round 1	Fountain	[Zhang, 2019a]
Round 1	GAGE and InGAGE	[Otte, 2019]
Round 1	HERN and HERON	[Ye et al., 2019]
Round 1	LAEM	[Han Sui, 2019]
Round 1	Lilliput-AE	[Adomnicai et al., 2019]
Round 1	Limdolen	[Mehner, 2019]
Round 1	Qameleon	[Avanzi et al., 2019]
Round 1	Quartet	[Zhang, 2019b]
Round 1	REMUS	[Iwata et al., 2020]
Round 1	Shamash and Shamashash	[Penazzi and Montes, 2019]
Round 1	SIMPLE	[Gueron and Lindell, 2019]
Round 1	SIV-Rijndael256	[Guo and Iwata, 2019]
Round 1	SIV-TEM-PHOTON	[Zhenzhen Bao, 2019]
Round 1	SNEIK	[Hashing, 2019]
Round 1	Sycon	[Sarkar et al., 2019]
Round 1	Thank Goodness It's Friday (TGIF)	[Iwata et al., 2019]
Round 1	Triad	[Isobe et al., 2019]
Round 1	TRIFLE	[Nilanjan et al., 2019]
Round 1	Yarará and Coral	[Miguel Montes, 2019]

They were eliminated for various reasons [Turan et al., 2019]. NIST favored algorithms with well-understood design principles, and they did not consider algorithms as secure if current analysis methods could not be applied. Some algorithms were eliminated for not getting enough analysis by the cryptography community. Some were eliminated because they did not have the structure and security they claimed.

The first round of the competition was completed in 2019 [Turan et al., 2019] and 32 algorithms made it to the second round. 18 of them had permutation-based constructions, 13 of them had block cipher mode of operation-based constructions, and one of them was a stream cipher. These algorithms can be seen in Table 1.3. We present our cryptanalysis results of these ciphers in Chapter 2.

Table 1.3: The eliminated algorithms in the second round of NIST's competition

State	Cipher	Construction	Source
Round 2	ACE	Permutation-based	[Aagaard et al., 2019a]
Round 2	COMET	Block cipher	[Gueron et al., 2019]
Round 2	DryGascon	Permutation-based	[Riou, 2019]
Round 2	ESTATE	Block cipher	[Chakraborti et al., 2020]
Round 2	ForkAE	Block cipher	[Andreeva et al., 2019]
Round 2	Gimli	Permutation-based	[Bernstein et al., 2017]
Round 2	HYENA	Block cipher	[Chakraborti et al., 2019]
Round 2	KNOT	Permutation-based	[Zhang et al., 2019]
Round 2	LOTUS-LOCUS	Block cipher	[Avik Chakraborti, 2019]
Round 2	mixFeed	Block cipher	[Chakraborty and Nandi, 2019a]
Round 2	ORANGE	Permutation-based	[Chakraborty and Nandi, 2019b]
Round 2	Oribatida	Permutation-based	[Bhattacharjee et al., 2021]
Round 2	Pyjamask	Block cipher	[Goudarzi et al., 2020]
Round 2	SAEAES	Permutation-based	[Naito et al., 2019]
Round 2	Saturnin	Block cipher	[Jirotka, 2016]
Round 2	SKINNY	Block cipher	[Beierle et al., 2020]
Round 2	SPIX	Permutation-based	[AlTawy et al., 2019b]
Round 2	SpoC	Permutation-based	[AlTawy et al., 2019a]
Round 2	Spook	Permutation-based	[Bellizia et al., 2020]
Round 2	Subterranean 2.0	Permutation-based	[Daemen et al., 2020b]
Round 2	SUNDAE-GIFT	Block cipher	[Banik et al., 2019a]
Round 2	WAGE	Permutation-based	[Aagaard et al., 2019b]

The important thing for the second-round algorithms to make it to the final was their performance analysis. Therefore, NIST asked competitors to optimize their algorithms for different platforms. Then after these measurements and the third-party analysis, 10 of them made it to the finals in March 2021. They were Ascon, ISAP, Photon-Beetle, Xoodyak, TinyJambu, Elephant, GIFT-COFB, Grain128-AEAD, Romulus, Sparkle. Six of them are permutation-based, three of them are block cipher-based ciphers and one of them is a stream cipher. They can be seen in Table 1.4.

Table 1.4: The finalist algorithms of NIST Lightweight Cryptography Competition

State	Cipher	Construction	Source
Finalist	ASCON	Permutation-based	[Dobraunig et al., 2016]
Finalist	Elephant	Block cipher	[Dobraunig and Mennink, 2019]
Finalist	GIFT-COFB	Block cipher	[Banik et al., 2019b]
Finalist	Grain-128AEAD	Stream cipher	[Hell et al., 2019]
Finalist	ISAP	Permutation-based	[Dobraunig et al., 2020]
Finalist	PHOTON-Beetle	Permutation-based	[Bao et al., 2019]
Finalist	Romulus	Block cipher	[Iwata et al., 2020]
Finalist	SPARKLE	Permutation-based	[Beierle et al., 2019]
Finalist	TinyJambu	Permutation-based	[Wu and Huang, 2019b]
Finalist	Xoodyak	Permutation-based	[Daemen et al., 2020a]

In this work, we performed some cryptanalysis for former round 2 candidates. But we mainly focused on Ascon and Drygascon. While Drygascon eliminated in the second round, Ascon made it to the finals. The competition is expected to last 2 more years and the winner algorithm is to be standardized around 2023.

1.3 Ascon

Ascon [Dobraunig et al., 2016] is a cipher suite that provides authenticated encryption with associated data and hashing functionality. It was a primary choice in the lightweight applications category of the CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) competition that was held between 2014 and 2019. It was selected as one of the finalists in the NIST's Lightweight Cryptography competition.

Ascon consists of ciphers Ascon-128 and Ascon-128a that are permutation-based structures. Both versions provide 128-bit security and use the same 320-bit permutation that is defined on 64-bit words but with different round numbers. Its mode of operation is based on the monkeyDuplex construction [Bertoni et al., 2012], which is a type of Duplex construction [Bertoni et al., 2011] with doubled initialization and finalization functions. So its security depends on the uniqueness of a nonce. Detailed information about the structures of Ascon will be given in Chapter 3.

Ascon is being analyzed since 2014 when it was first submitted to the CAESAR competition [Bernstein, 2013]. There are currently more than 80 analyses and about 35 publications against its permutation, mode of operation, and implementation. Out of numerous published analyses, the best attack is a cube-like recovery attack which was performed to 7 of 12 rounds of Ascon's permutation [Li et al., 2017]. The other key recovery attacks that perform better than exhaustive search were a zero-sum attack for 11 rounds, a zero-correlation attack for 5 rounds, a differential attack for 4 rounds all provided by [Dobraunig et al., 2016]. And there are also a integral attack for 7 rounds [Todo, 2015], a truncated differential attack for 5 rounds [Tezcan, 2016], a linear attack for 4 rounds [Dobraunig et al., 2015b], differential-linear attacks for 5 rounds provided by both [Dobraunig et al., 2015a] and [Tezcan, 2020].

In this work, we studied the 4-round truncated differential distinguisher [Tezcan, 2016] and 4-round differential-linear cryptanalysis [Tezcan, 2020] of Ascon. Then we performed a similar analysis for Drygascon. The reason we did that was to compare their security because they are structurally similar cipher suites. Since Ascon had numerous analysis and still seems to be secure, we wanted to see if the changes in its permutation that made it Drygascon is better.

1.4 Drygascon

Drygascon [Riou, 2019] is a cipher suite that provides AEAD and hashing functionality. It was selected as a second-round candidate in NIST's Lightweight Cryptography competition. It did not make it to the final round.

Drygascon has two instances: Drygascon-128, and Drygascon-256. But the primary submission was Drygascon-128. It is a permutation-based construction and it uses GASCON permutation as a primitive, which is a generalized variant of Ascon. The GASCON's aim was to increase the Ascon's 128-bit security. But it differs from Ascon with some of its features. Unlike Ascon, the round numbers of Drygascon-128 are 11 instead of 12. It uses Ascon's $5x5$ S-boxes but represents it in little-endian. Besides, its linear layer differs from Ascon. In addition to the 2-different rotations, the rotation function is also different in the linear layer. And its constants are not

related to the total number of rounds. It uses a new construction DrySponge as a mode of operation, which was based on Duplex Sponge construction [Bertoni et al., 2011]. But the combining of the input with the state and the extraction of output from the state is different in DrySponge than the Duplex Sponge construction. Detailed information about the structures of Drygascon will be given in Chapter 4.

There is not much security analysis that was given to the GASCON permutation in the literature. But the designer stated that since GASCON is similar to the permutation of Ascon, the same cryptanalysis methods can be applied with some modifications [Riou, 2019]. They provided their own analysis by presenting linear approximations and a 3-round probability one truncated differential distinguisher [Riou, 2019]. They claimed that the best achievable probability one truncated differential for Drygascon-128 is 3-round. But Tezcan refutes that claim by providing 3.5-round probability one truncated differential distinguisher [Tezcan, 2020]. Tezcan also presented a 3-round subspace trail, and a 5-round differential-linear distinguisher [Tezcan, 2020]. The known analyses with the total encryption time can be seen in Table 1.5.

Table 1.5: Best known analyses of the $GASCON_{C5R11}$ permutation

Method	**Rounds**	**Time**	**Source**
Truncated Differential	3.5/11	1	[Tezcan, 2020]
Truncated Differential	3/11	1	[Riou, 2019]
Subspace Trail	3/11	1	[Tezcan, 2020]
Differential-Linear	5/11	$2^{61.28}$	[Tezcan, 2020]
Linear	3/11	2^{75}	[Riou, 2019]

In this work, we compare the security of Ascon and Drygascon. For Ascon, there is a 4-round differential-linear distinguisher that was turned into 4 and 5-round differential-linear attacks [Tezcan, 2020]. In a similar manner, Drygascon had a 5-round theoretical differential-linear distinguisher without practical results [Tezcan, 2020]. Therefore we aimed to find the practical results of this 5-round distinguisher to compare their security.

1.5 Undisturbed Bits

In this work, we analyzed the S-boxes of some candidates on NIST's competition to see if they contain any undisturbed bits. S-boxes, or substitution boxes, are non-linear components of symmetric-keyed algorithms that provide confusion for the cipher.

Undisturbed bits can be used for building longer and sometimes more preferable differentials in truncated, improbable, and impossible cryptanalysis. They were first used in cryptanalysis in [Tezcan, 2014] and these bits could be seen as probability one truncated differential for an S-box. And according to [Tezcan et al., 2014] every bijective $3x3$ S-box have undisturbed bits. Undisturbed bits are defined as follows:

Definition 1 *"For a fixed input difference, an output bit is called undisturbed if its difference remains invariant." [Tezcan, 2014]*

Undisturbed bits are actually the linear structures of S-boxes and that fact was shown in [Makarim and Tezcan, 2014] later on. Linear structures could be defined like this:

Definition 2 *"For a nonzero vector $\alpha \in F_2^n$, if an n x m S-box S has a nonzero vector $b \in F_2^m$ such that $b.S(x) \oplus b.S(x \oplus \alpha)$ has the same value $c \in F_2$ for all $x \in F_2^n$, then we say that S has a linear structure." [Evertse, 1987]*

This feature has been used in several studies in the literature. Tezcan used this to extend the 7-round improbable differential attack of Present to 13-round [Tezcan, 2014]. Then they provided the first 7-round improbable differential cryptanalysis for SERPENT by using undisturbed bits [Tezcan et al., 2014]. They also found the undisturbed bits of Ascon-128 and used that to build 3.5-round probability one truncated differentials [Tezcan, 2016]. And then they used that to attack 4 and 5 rounds of Ascon-128, in terms of truncated, improbable [Tezcan, 2016] and differential-linear cryptanalysis [Tezcan, 2020]. Last but not least, they found the undisturbed bits of Drygascon-128 and used that to provide 3.5-round probability one truncated differentials and 5-round differential-linear distinguishers for it [Tezcan, 2020]. And in this study, we use it to build probability one truncated differentials for some algorithms that were submitted to the NIST's competition.

1.6 Differential Cryptanalysis

The purpose of designing a cipher is to protect data by making it appear random to the adversary. If it is possible to find a relationship between plaintext, ciphertext, or key this purpose cannot be fulfilled. Differential cryptanalysis [Biham and Shamir, 1991] aims to observe how a fixed input difference affects the output difference. If the difference in the output can be observed with high probability, that can be used to discover a property of a cipher with the aim of extracting the secret key. It was introduced as a theoretical attack on Data Encryption Standard (DES) cipher, which is a block cipher with a Feistel Network structure. Even though it is mostly applied on block ciphers, it can also be used to analyze stream ciphers and even the hash functions. In this study (Chapter 2), we perform differential cryptanalysis for some algorithms that were submitted to the NIST's competition.

Differential cryptanalysis is a chosen-plaintext attack, which means the attackers can request the encryption of N plaintexts that they determined and capture the ciphertexts that correspond to them. In chosen plaintext attacks, some conditions can be set for the requested plaintexts. These conditions usually include that the plaintext pairs have some differences. These differences are a fixed value of the XOR of the two plaintexts. Then all plaintext pairs with these differences are encrypted separately with the same key, and an inference is made using the corresponding ciphertext differences. In this way, the secret key is tried to be determined fully or partially. The plaintexts can be random as long as they satisfied the given difference condition. This attack type is feasible in real life due to scenarios such as attackers being able to capture the device.

Differential cryptanalysis is a statistical attack, like almost all other cryptanalysis methods. In statistical attacks, an insecure communication channel is listened to capture some data like ciphertexts or in some cases, corresponding plaintexts. However, since the internal values or secret key cannot be known, a statistical weakness is sought in the cipher. In other words, the aim is to distinguish the whole or reduced version of the cipher from a random permutation. The special condition that provides that is called the *distinguisher*. Extending the distinguisher is possible to turn it into a cryptanalytic attack to capture the whole or part of the secret key.

In this work, we focused on SPN-based ciphers to analyze. That means we analyzed the permutation of the ciphers who have a non-linear substitution layer that provides confusion and a linear layer that provides diffusion. Since the permutations we analyzed use S-boxes for confusion, we will explain finding a differential distinguisher in that means only.

Differential Cryptanalysis aims to observe how a fixed input difference affects the output difference. When a difference $p1 \oplus p2 = \alpha$ is introduced on the input, the key addition layer does not have any effect due to the $(p1 \oplus \alpha) \oplus (p2 \oplus \alpha) = (p1 \oplus p2) = \alpha$. And the permutation layer only changes the location of the bits, therefore this does not have any effect either. However in substitution layer, since the values $S(p1 \oplus \alpha)$ and the $S(p2 \oplus \alpha)$ are unknown, the exact value of $S(p1 \oplus \alpha) \oplus S(p2 \oplus \alpha)$ cannot be known also. But analyzing the S-box will give the probabilistic values of these occurrences. Difference Distribution Table (DDT) [Biham and Shamir, 1991] can be used for accomplish that.

In DDT, for every possible (x, y) input pairs with difference $x \oplus y = i$ and corresponding output pairs with difference $S(x) \oplus S(y) = j$ are formed the table as the ij-th entry. ij-th entry represents the occurrence of how many times the input difference i is observed with the output difference j. For an nxn S-box, dividing this value ij to the size of the DDT, which is 2^n, gives the probability of this occurrence. Therefore DDT helps to decide which values are convenient for building a distinguisher statistically. In this table, the highest value besides the first entry is called *differential uniformity* [Nyberg, 1993] and an attacker could find a good trail if this value would be higher. The best achievable differential uniformity is 2 theoretically for an S-box since the input pairs (x, y) and (y, x) provide the same difference. This property of the functions is called *almost perfect nonlinear (APN)*. While for an odd n there are known such S-boxes, for an even n, Dillon presented an example where $n = 6$ [Browning et al., 2010]. For $n = 4$ it could be seen that there is no such S-box by checking every possible $4x4$ S-boxes. Finding a 2-uniform S-box where $n = 8$ is still an open problem.

For finding a distinguisher with good probability in differential cryptanalysis, DDT can be used. The high occurrence values in DDT can be selected as a starting point.

Selecting input difference α and corresponding output difference β from DDT gives the probabilistic value for the substitution layer. Then the linear layer is applied to the output difference β with probability one. Since the difference β is diffused by the linear layer, some bits will become active. For the next round, it is decided which bits should remain active. This selection gives the new input value for the next round. This process is repeated for several rounds. And then a theoretical distinguisher can be determined by multiplying the probabilities in each round since these probabilities are assumed to be independent.

For verifying a distinguisher in practice, the selected input difference α is used for the plaintext pairs. Then the plaintext $p1$ is randomly generated. And its pair $p2$ is produced by flipping some of the bits of $p1$ in such a way that the difference is going to be $p1 \oplus p2 = \alpha$. Then $p1$ and $p2$ are separately encrypted with the same key throughout r rounds. Then it is checked if the corresponding ciphertext pairs is $c1 \oplus c2 = \beta$. This operation is repeated for N data. Since this is actually a binomial distribution, N must be chosen such that the expected value $E = N \cdot p > 1$ will be large enough, where p is the total probability. Then it is counted how many times the selected output difference β is observed. That observation gives the practical probability of this occurrence. This experiment can be repeated with a convenient number of different keys to get the average value of these probabilities.

1.6.1 Truncated Differential Cryptanalysis

Truncated differential cryptanalysis [Knudsen, 1994] is a special case of differential cryptanalysis and it is introduced by Knudsen in 1994. In this case, there is no need to fully specify the differences, only fixing some bits in the input and output differential is enough. With the usage of the undisturbed bits, truncated differentials can be propagated as probability one for some rounds throughout the cipher. In this work, we present probability one truncated differential distinguishers for some algorithms that were submitted to the NIST's competition (Chapter 2). Then we analyze Ascon and Drygascon by the combination of their probability one truncated differential distinguishers with a linear approximation.

1.7 Linear Cryptanalysis

In this study, we used linear cryptanalysis to analyze differential-linear distinguishers of Ascon-128 and Drygascon-128. Linear cryptanalysis was presented as a theoretical attack on DES cipher. Then it was applied as a practical attack on DES. It was introduced by Matsui [Matsui, 1993]. It has been used to analyze various block ciphers since then.

Linear cryptanalysis is a known-plaintext attack. Also, [Matsui, 1993] stated that when given the right conditions it can be applied as a ciphertext-only attack on DES. One of these conditions explained as plaintexts to be in form of English words in ASCII representation. But since this is a special case, we will focus on the known-plaintext version of this analysis when explaining it.

In a known-plaintext attack, the adversary can gather N plaintexts and corresponding ciphertexts. Unlike chosen-plaintext attacks, attackers are not allowed to make any choice about the conditions of the plaintexts. This attack type is feasible in real life due to scenarios such as attackers are being able to capture the device, guess some blocks of the plaintexts, or trick the key holder to encrypt a document that they know.

Linear cryptanalysis tries to find a connection between plaintext bits, subkey bits, ciphertext bits and then obtain a linear expression of the cipher. In this technique, the goal is to approximate the mechanism of the set of cipher with a linear expression using modulo-2 bitwise linear operation. This linear expression is in that form:

$$P[k_1, k_2, \ldots] \oplus C[l_1, l_2, \ldots] = K[j_1, j_2, \ldots] \tag{1}$$

In this equation (1), the values k, l, j represent the fixed bit locations, namely masks. And P, C, K represent the plaintext bits, ciphertext bits, and key bits respectively. The equation (1) can have a high or low probability of appearance, but either way, this means that the cipher is catastrophically weak. For a randomly selected data set, the equation (1) should hold around $1/2$. If it holds with high or low probability than that, it points out the cipher is not producing random data. The deviation from $1/2$ is used for linear cryptanalysis and this property is called *linear probability bias*.

For randomly chosen plaintexts and corresponding ciphertexts, if the expression (1) holds with probability p, then the bias is $|p - 1/2|$. The greater bias leads to the better applicable linear cryptanalysis, and the lesser bias means the more plaintext is required to distinguish the cipher from a random permutation.

The $p = 1$ or $p = 0$ reveals the cipher mechanism and means that cipher has a significant weakness. While $p = 0$ indicates a linear relationship and can be specified as $p > 1/2$, $p = 1$ indicates an affine relationship and can be specified as $p < 1/2$. The affine function is the complement of a linear function in mod-2 addition systems and the term linear covers both of them.

The practice in linear cryptanalysis is to find out the expression (1). To be able to determine the subkey bits, the expression (1) can be reformulated as expression (2).

$$P[k_1, k_2, \ldots] \oplus C[l_1, l_2, \ldots] \oplus \sum K = 0 \tag{2}$$

Since the subkey bits, $\sum K$ are unknown but fixed, the results of the value $\sum K$ are going to be 0 or 1. This will be determined when attacking the key. If the result of $\sum K$ is "0", the bias of (2) will have the same sign. If not, bias will have the opposite sign.

Building highly linear expressions is done by using the S-box, the cipher's non-linear component, and constructing a linear approximation table (LAT). In LAT, to construct linear approximations between input and the output bits, the nonlinearity characteristics of the S-box are enumerated. The procedure for that as follows: for an S-box that have size of kxl, the linear relations $m.x = m_1x_1 \oplus m_2x_2 \oplus \ldots \oplus m_kx_k$ and $n.y = n_1y_1 \oplus n_2y_2 \oplus \ldots \oplus n_ly_l$ should be considered for all inputs and the outputs, where $0 \leq m \leq 2^{k-1}$ and $0 \leq n \leq 2^{l-1}$. In here, m_1 and n_1 are most significant bits. $m_1, m_2, \ldots, m_k$ represent the masked bits of the input. And $n_1, n_2, \ldots, n_l$ represent the masked bits of the output. The kl-th element of the table gives the relation between them, in terms of occurrence possibility. Then the elements of the table should be divided into 2^k to have the probability p that holds the expression. After that, p should be computed for all rounds with the $\epsilon = |p - 1/2|$ bias. By concatenation of the highest bias ϵ for all rounds, the expected bias probability can be computed. With

the cancellation of midpoints, results of high biased linear expressions can be derived which involve only plaintext bits and the corresponding ciphertext bits. This could be done by using Piling-up Lemma [Matsui, 1993].

(Piling-up Lemma [Matsui, 1993]) Let X_i (1$\leq$ i $\leq$ n) be independent random variables whose values are 0 with probability p_i or 1 with probability 1-p_i. Then the probability that $X_1 \oplus X_2 \oplus \ldots \oplus X_n = 0$ is

$$1/2 + 2^{n-1} \prod_{i=1}^{n} (p_i - 1/2)$$

To be able to perform this operation automatically, a software called *lineartrails* [Dobraunig et al., 2015a] can be used with implementing the permutation algorithm of the cipher into the tool. This heuristic tool was presented in 2015 to find good linear characteristics for primitives of the SPN-structured ciphers. It was used to analyze some of the algorithms in the CAESAR competition. It works with the guess-and-determine approach. To build characteristics suitable for different attack types, it holds 3-different types of linear characteristics to search for.

- **Type-I:** In this type, there are no restrictions for finding characteristics. The active bits are allowed to be on any bits of the permutation. Therefore it may not be used for attacking sponge-constructions like Ascon. However, it can still give an idea about the resistance of the cipher against linear cryptanalysis.
- **Type-II:** In this type, the active bits must be in the outer part of the state at the end of the characteristic, and other bits should not contain any masks. It can be used for key recovery attacks on sponge constructions.
- **Type-III:** This type is similar to the Type-II characteristics. But it targets the encryption phase.

In this study, we used linear cryptanalysis to analyze and build differential-linear distinguishers for Ascon-128 and Drygascon-128. We used linear characteristics that were provided by [Dobraunig et al., 2016] and [Riou, 2019]. We also used lineartrails tool [Dobraunig et al., 2015a] to find linear characteristics of Drygascon.

1.8 Differential-Linear Cryptanalysis

In this study, we used differential-linear cryptanalysis to analyze differential-linear distinguishers of Ascon-128 and Drygascon-128. Differential-linear cryptanalysis was introduced by [Langford and Hellman, 1994] in 1994 and it is a combination of the two main techniques differential cryptanalysis [Biham and Shamir, 1991] and linear cryptanalysis [Matsui, 1993]. Although it uses shorter characteristics, it might provide better distinguishers for some ciphers. The idea is dividing the cipher E into two parts: E_0 and E_1; namely $E = E_0 \circ E_1$. In E_0 part, a truncated differential $\lambda_I \rightarrow \lambda_o$ with probability $p = 1$ is found. And in E_1 part, a linear approximation $\nabla_I \rightarrow \nabla_o$ with probability $1/2 + q$ is found, where q is represented as the bias. Then the combination of E_0 and E_1 is used to find an efficient distinguisher for the cipher E. In this combination note that the masked input bits of the linear approximation should match the zero-difference in the output bits of truncated differentials.

For distinguishing cipher E from a random permutation, a suitable number of plaintext pairs with input difference λ_I is used. After the permutation of each pair, it is observed if the corresponding ciphertexts have the same parity of the mask ∇_o. Namely $parity((c1 \odot \nabla_o)) \oplus parity((c2 \odot \nabla_o)) = parity((c1 \oplus c2) \odot \nabla_o) == 0$, where $c1, c2$ are a ciphertext pair. After this condition is checked with a suitable number of data, if the probability is approximately $1/2$, it can be said that the cipher behaves randomly. If not, the distinguisher could be turned into a key recovery attack.

[Biham et al., 2002] showed that this technique is still possible when the masked bits of the first round of linear approximation matches with the 1 difference at the end of the truncated differential. They also showed that it is possible to construct the attack when p is less than 1 and called it enhanced differential linear cryptanalysis. In this case, the bias of this distinguisher can be calculated as approximately $2pq^2$ according to Matsui's Piling-up lemma [Matsui, 1993]. The data complexity of the attack is $\theta(p^{-2}q^{-4})$ chosen plaintexts approximately, where θ is the big O notation. But when the probability one truncated differential is used for E_0, the data complexity can be calculated as approximately $\theta(q^{-4})$ chosen-plaintext and the bias is can be computed as approximately $2q^2$. Note that these numbers do not include the success probability and the used number of subkeys.

1.9 Our Contribution

In this work, we examined the NIST Lightweight Cryptography Competition second-round candidates. We checked 16 of them that we specially selected to see if they contain any undisturbed bits. Using their undisturbed bits, we found out how many rounds of probability one truncated differentials that they have. We present our results in Chapter 2 for future analysis.

Then we focused on two very similar cipher suites: Ascon and Drygascon. Ascon made it to the finals, Drygascon was eliminated in the second round. Our goal was to compare the security of the two against the current attacks. In this way, we would see if the changes in the Ascon's permutation made it better as Drygascon. For this reason, we examined how differential-linear cryptanalysis against Ascon-128 could impact the Drygascon-128. So we studied the existing differential-linear distinguishers of Ascon-128 and Drygascon-128 [Tezcan, 2020]. For Ascon-128, there was a 4-round differential-linear distinguisher that was turned into 4 and 5-round key recovery attacks [Tezcan, 2020]. And for Drygascon-128 there was a 5-round theoretical differential-linear distinguisher [Tezcan, 2020]. While experimentally verifying the distinguisher of Drygascon-128, we realized that the 2-round probability one truncated differential distinguisher that was used in 5-round differential-linear distinguisher was incorrect. When we examined the other probability one truncated differential distinguishers, we realized that these were incorrect also. One of them was provided by the designer, and covered 3-round of Drygascon [Riou, 2019]. The other one was the improved version of this 3-round and covered 3.5-round of Drygascon [Tezcan, 2020]. We think the reason for these erroneous analyses was that the bits move in the opposite direction than described in linear layer [Riou, 2019]. We corrected these results and verified the existence of a 3.5-round truncated differential, but its middle rounds were diffused differently than the previous one [Tezcan, 2020]. We present our results in Chapter 4.

After corrected the existed differential distinguishers, we found a practical 5-round differential-linear distinguisher by using a 3-round linear approximation that was provided by [Riou, 2019]. The theoretical bias of this linear approximation was 2^{-15}. But we achieved $2^{-7.96}$ bias by using 2^{29} data for the whole 5-round distinguisher. Then

we found a different 3-round linear approximation by using a heuristic tool called lineartrails [Dobraunig et al., 2015a]. Although the bias was the same as the previous one, the practical results were surprisingly better. We achieved $2^{-5.34}$ bias by using 2^{17} data. This is currently the longest differential-linear distinguisher that we know of. We present our results in Chapter 4.

The results so far were contained 4-round and 5-round differential-linear distinguishers of Ascon and Drygascon respectively. But that did not mean Ascon is better than Drygascon since different types of linear approximations were used in these analyses. The 4-round distinguisher of Ascon was contained a 2-round Type-II linear approximation and the 5-round distinguisher of Drygscon was contained a 3-round Type-I linear approximation. The reason was that it is necessary to use Type-II approximation to turn this kind of distinguisher into an attack, since the active bits at the end of the distinguisher in the outer part. But this was not an issue for analyzing Drygascon because the additional functions that were used in it changed its state. Therefore Type-I approximations were used before to check only its resistance against differential-linear cryptanalysis and we continued with this mindset. But since that was not going to be a fair comparison, we found a 4-round practical differential-linear distinguisher that contains a 2-round Type-II linear approximation for Drygascon. We observed while the best results for Ascon was $2^{-1.68}$ in that matter, it was $2^{-1.69}$ in Drygascon. These are very close results. But since the round numbers were 12 for Ascon and 11 for Drygascon, we may say that Ascon might be more resistant against differential-linear cryptanalysis.

CHAPTER 2

UNDISTURBED BITS

In this work, we analyzed the candidate algorithms in the Lightweight Cryptography Competition organized by NIST to be able to find good distinguishers. We used undisturbed bits to be able to provide longer truncated differential distinguishers for the algorithms. There were 56 candidate algorithms in the beginning. When we started this study, 24 of them had already been eliminated. So there were only 32 ciphers that would make sense to work on for us. These were the second round algorithms and we decided to find their undisturbed bits to be able to provide probability one truncated differentials as a beginning.

To be able to find undisturbed bits, the ciphers had to have structures that contain a substitution layer and a linear layer. This situation reduced our workspace. Then, we observed the S-boxes of the ciphers were of various sizes like 3-bit, 4-bit, 5 bit, 7 bit, and 8 bit. We decided to focus on the ciphers that have 3-bit, 4-bit, and 5 bit S-boxes. Because in this way, it was going to be easier to analyze their S-boxes. That choice left us with 16 ciphers instead of 32. Those were Ascon, Drygascon, ISAP, Saturnin, Spook, PHOTON-Beetle, ORANGE, Pyjamask, ForkAE, KNOT, Elephant, HyENA, ESTATE, GIFT-COFB, LOTUS-AEAD and LOCUS-AEAD, SUNDAE-GIFT. Their S-boxes can be seen in Table 2.1 and Table 2.2.

Table 2.1: NIST Second Round Lightweight Cryptography Competition Candidates that have 5-bit S-Boxes which are Ascon, Drygascon and Isap

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	04	0b	1f	14	1a	15	09	02	1b	05	08	12	1d	03	06	1c
x	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$S(x)$	1e	13	07	0e	00	0d	11	18	10	0c	01	19	16	0a	0f	17

Table 2.2: NIST Second Round Lightweight Cryptography Competition Candidates that have 3-bit and 4-bit S-Boxes

Cipher	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
Elephant	E	D	B	0	2	1	4	F	7	A	8	5	9	C	3	6
Estate Gift-Cofb Hyena Lotus-Locus Sundae-Gift	1	A	4	C	6	F	3	9	2	D	B	7	5	0	8	E
ForkAE	C	6	9	0	1	A	2	B	3	8	5	D	4	E	7	F
KNOT	4	A	0	7	B	E	1	D	9	F	6	8	5	2	C	3
ORANGE PHOTON-Beetle	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2
Pyjamask 3-bit	1	3	6	5	2	4	7	0								
Pyjamask 4-bit	2	D	3	9	7	B	A	6	E	0	F	4	8	5	1	C
Saturnin	0	6	E	1	F	4	7	D	9	8	C	5	2	A	3	B
Spook	0	8	1	F	2	A	7	9	4	D	5	6	E	3	B	C

We analyzed the whole 16 algorithms to get their undisturbed bits. We observed some of the ciphers had the same S-boxes. Namely Ascon, Drygascon, and Isap were using the same 5-bit S-boxes of Ascon. Estate, Gift-Cofb, Hyena, Lotus-Locus, Sundae-Gift were using the same 4-bit S-boxes as GIFT. Orange and Photon-Beetle were using the same 4-bit S-boxes as Photon. The analysis results are given in Table 2.3.

Table 2.3: Undisturbed bits of 16 Second Round Candidates that we have picked to analyze. The ciphers that have the same S-boxes were represented as their S-box names. Namely Ascon, Photon, and Gift.

Ciphers	Input Difference	Output Difference	Ciphers	Input Difference	Output Difference
Ascon	00001	?1???	Elephant	0001	???1
Ascon	00010	1???1	Elephant	1000	???1
Ascon	00011	???0?	Elephant	1001	???0
Ascon	00100	??110	ForkAE	0001	10??
Ascon	00101	1????	ForkAE	0010	0???
Ascon	00110	????1	ForkAE	0011	1???
Ascon	00111	0??1?	ForkAE	1000	?1??
Ascon	01000	??11?	ForkAE	1001	?1??
Ascon	01011	???1?	Knot	0001	?1??
Ascon	01100	??00?	Knot	1000	?1??
Ascon	01110	?0???	Knot	1001	?0??
Ascon	01111	?1?0?	Photon	0001	???1
Ascon	10000	?10??	Photon	1000	???1
Ascon	10001	10??1	Photon	1001	???0
Ascon	10011	0???0	Pyjamask 3-bit	001	?1?
Ascon	10100	0?1??	Pyjamask 3-bit	010	1??
Ascon	10101	????1	Pyjamask 3-bit	100	??1
Ascon	10110	1????	Pyjamask's 4-bit	0001	1???
Ascon	10111	????0	Pyjamask's 4-bit	0010	??0?
Ascon	11000	??1??	Pyjamask's 4-bit	1000	1???
Ascon	11100	??0??	Pyjamask's 4-bit	1001	0?1?
Ascon	11110	?1???	Pyjamask's 4-bit	1011	??1?
Ascon	11111	?0???	Spook	0010	???1
Gift	0100	???1	Spook	0100	??10
Gift	0110	??1?	Spook	0110	???1
Gift	1000	??11	Spook	1000	??0?
Gift	1100	???0	Spook	1100	??1?
Gift	1110	??0?			

After finding the undisturbed bits of the algorithms, we studied the related cipher's structures and calculated their probability one truncated differentials. Our aim was to see if their distinguishers will give us long enough rounds to work with. The results of how many rounds we managed to go with probability one truncated differentials using undisturbed bits can be seen in Table 2.4.

Table 2.4: Probability one truncated differential distinguishers of some algorithms that were submitted into NIST's lightweight cryptography competition. Round numbers represent the minimum one that is used in their permutations.

Name	Sbox	Undisturbed Bits	Rounds	Analysis
Saturnin	4-bit	0	10	0 round
Drygascon	5-bit	13	11	3.5 round
Ascon	5-bit	13	12	3.5 round
Isap	5-bit	13	12	3.5 round
Spook	4-bit	5	12	- round
Photon-Beetle	4-bit	3	12	1.5 round
Orange	4-bit	3	12	1.5 round
Pyjamask	3-bit	3	14	1.5 round
Pyjamask	4-bit	5	14	1.5 round
Forkae (Forkskinny-64-192)	4-bit	5	17	-
Knot	4-bit	3	28	7 round
Lotus-Locus	4-bit	5	28	3 round
Hyena	4-bit	5	40	3 round
Gift-Cofb	4-bit	5	40	3 round
Estate	4-bit	5	40	3 round
Sundae-Gift	4-bit	5	40	3 round
Elephant	4-bit	3	80	-

Out of these 16 ciphers, we decided to focus on two very similar cipher suites: Ascon and Drygascon. Because other ciphers had shorter truncated differentials compared to their total number of rounds. So it could have been harder to work with these truncated differential distinguishers for further analysis.

CHAPTER 3

ASCON

Ascon [Dobraunig et al., 2016] is a cipher suite that provides AEAD and hashing functionality. It was a primary choice in the lightweight applications category of the CAESAR competition and selected as one of the finalists in the NIST Lightweight Cryptography competition. It consists of ciphers Ascon-128 and Ascon-128a that have SPN structures. Both versions provide 128-bit security and use the same 320-bit permutation that is defined on 64-bit words but different round numbers. Ascon's mode of operation is based on the monkeyDuplex construction [Bertoni et al., 2012] except keyed initialization and finalization functions are stronger.

3.1 Notation

The operators used in the description of the Ascon and Drygascon can be seen in Table 3.1. When the state needs to be interpreted as bitstring, it starts with a most significant bit in Ascon and the least significant bit in Drygascon.

Table 3.1: Notation that is used in description of Ascon and Drygascon

a$\oplus$b	XOR of a and b bitstrings
a&b	Bitwise and operetion of a and b bitstrings
a|b	Bitwise or operation of a and b bitstrings
a$\ggg$b	Right rotation of a in b times
a$\ll$b	Left shift of a for b times
a$\gg$b	Right shift of a for b times
a||b	Concatenation of a and b bitstrings

3.2 Permutation-Based Constructions

Traditional encryption algorithms often use block ciphers. Block ciphers involve dividing plaintext into fixed-length blocks and encrypting each block with a secret key. These encrypted blocks are then combined using various mode of operations. These ciphers are usually SPN or Feistel. However, it is seen that sponge constructions have recently been used as an alternative.

Sponge constructions [Bertoni et al., 2007] use fixed-permutations as primitives instead of block ciphers. Therefore the security of this structure depends on the security of this permutation. Unlike block ciphers, they do not include a key-scheduling algorithm. Instead, they use a limited-length initial state that includes the secret key. By using this state, the message with the desired length can be encrypted. They can be used to implement various cryptographic components such as authenticated encryption algorithms, hash functions, stream ciphers. Block ciphers can be converted into iterative permutations if their secret keys are fixed. Hence they can be used in sponge constructions.

There are two parts of the state produced in sponge constructions: outer part; rate, and inner part; capacity. While rate is responsible for the efficiency of the structure, capacity is concerned with its security. The size of the two varies according to each other. In other words, determining their size is a choice between security and efficiency. Then the structure goes through stages called absorb and squeeze. It was illustrated in Figure 3.1.

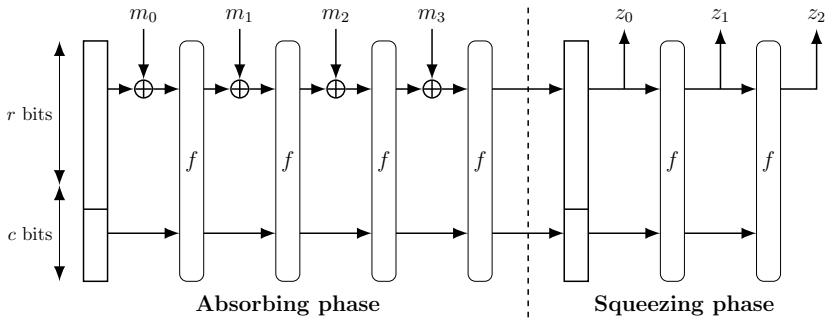

Figure 3.1: Illustration of Sponge Construction. The figure was taken from [Jean, 2016]

Duplex construction [Bertoni et al., 2011] is one of the ways to use sponge construction. Therefore, the security properties valid for sponge construction are also valid for duplex construction. Absorb and squeeze processes in duplex construction are applied slightly differently than sponge construction.

MonkeyDuplex [Bertoni et al., 2012] construction can be called the keyed version of Duplex construction. Here, obtaining the inner part of the state is as important as obtaining its key. Therefore, its security is based on the uniqueness of the nonce used.

3.3 Properties of Ascon's S-box

S-boxes are non-linear components of symmetric-keyed algorithms and provide confusion for the cipher. It is basically a look-up table that takes some n-bit as input and turns it into m-bit output. It can also be implemented as bit-sliced. Ascon has a $5x5$ S-box, and that can be seen in Table 3.2.

Table 3.2: 5-bit S-box of Ascon

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	04	0b	1f	14	1a	15	09	02	1b	05	08	12	1d	03	06	1c
x	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$S(x)$	1e	13	07	0e	00	0d	11	18	10	0c	01	19	16	0a	0f	17

Analyzing the S-box will give how many times the input difference is observed with the output difference. And that information can be used to perform differential cryptanalysis. DDT [Biham and Shamir, 1991] gives these occurrences in a table for a more systematic analysis. In this table, the highest value besides the first entry is called differential uniformity [Nyberg, 1993] and an attacker could find a good trail if this value would be higher. Therefore, when designing a cipher this is taken into account for selecting S-boxes. It is theoretically possible to obtain the 2-uniform S-boxes if the S-box is odd [Nyberg, 1993]. Ascon has an odd S-box, but the designers did not prefer a 2-uniform S-box. Instead, they used a 8-uniform S-box and that can be seen in Table 3.3.

Table 3.3: DDT of Ascon's Sbox

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
0	32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	.	.	.	.	.	.	.	.	.	4	.	4	.	4	.	4	.	.	.	.	.	.	.	.	4	.	4	.	4	.	4	.
2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4	.	4
3	.	4	.	.	.	4	.	.	.	4	.	.	.	4	.	.	4	.	.	.	4	.	.	.	4	.	.	.	4	.	.	.	.
4	.	.	.	.	.	.	8	.	.	.	.	.	.	.	8	.	.	.	.	.	.	.	8	.	.	.	.	.	.	.	8	.
5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	4	.	4	4	.	4	.	4	.	4	.	.	4	.	4
6	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2
7	.	.	4	4	.	.	4	4	.	.	4	4	.	.	4	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8	.	.	.	.	.	.	4	4	.	.	.	.	.	.	4	4	.	.	.	.	.	.	4	4	.	.	.	.	.	.	4	4
9	.	2	.	2	2	.	2	.	2	.	2	.	.	2	.	2	2	.	2	.	.	2	.	2	.	2	.	2	2	.	2	.
a	.	2	2	.	2	.	.	2	.	2	2	.	2	.	.	2	.	2	2	.	2	.	.	2	.	2	2	.	2	.	.	2
b	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2	.	.	2	2
c	.	8	.	.	.	.	.	.	8	.	.	.	.	.	.	.	8	.	.	.	.	.	.	.	.	8	.	.	.	.	.	.
d	.	2	.	2	.	2	.	2	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	.	2	.	2	.	2	.	2
e	.	4	4	.	4	.	.	4	.	.	.	.	.	.	.	.	.	4	4	.	4	.	.	4	.	.	.	.	.	.	.	.
f	.	.	.	.	.	.	.	.	4	4	.	.	4	4	.	.	.	.	.	.	.	.	.	.	4	4	.	.	4	4	.	.
10	.	.	.	.	.	.	.	.	.	8	.	8	.	.	.	.	.	.	.	.	.	.	.	.	8	.	8	.	.	.	.	.
11	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	8	.	8	.	8	.	8	.	.	.	.	.	.	.	.
12	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	2	.	2	.	2	.	2	.	2	.	2	.	2	.	2	.
13	.	.	8	.	8	.	.	.	.	.	8	.	8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
14	.	.	.	.	4	4	4	4	.	.	.	.	4	4	4	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
15	.	.	.	.	.	4	.	4	.	4	.	4	.	.	.	.	.	4	.	4	.	.	.	.	.	.	.	.	.	4	.	4
16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
17	.	.	4	.	4	.	.	.	.	4	.	4	.	.	.	.	.	4	.	4	.	.	.	.	.	4	.	4	.	.	.
18	.	.	.	.	2	2	2	2	.	.	.	2	2	2	2	.	.	.	.	2	2	2	2	.	.	.	.	2	2	2	2	.
19	.	.	.	4	.	.	4	.	4	.	.	.	.	4	.	.	4	.	.	.	.	4	.	.	.	.	.	4	.	.	4	.
1a	.	2	2	.	.	2	2	.	2	.	.	2	2	.	.	2	.	2	2	.	.	2	2	.	2	.	.	2	2	.	.	2
1b	.	.	2	2	2	2	.	.	.	.	2	2	2	2	.	.	.	.	2	2	2	2	.	.	.	.	2	2	2	2	.	.
1c	.	4	.	4	.	.	.	.	4	.	4	.	.	.	.	.	4	.	4	.	.	.	.	.	.	4	.	4	.	.	.	.
1d	.	.	.	4	.	4	.	.	4	.	.	.	.	.	4	.	4	.	.	.	.	.	4	.	.	.	.	4	.	4	.	.
1e	.	.	.	.	.	.	.	.	.	2	2	2	2	2	2	2	.	.	.	.	.	.	.	.	2	2	2	2	2	2	2	2
1f	.	.	4	4	4	4	.	.	.	.	.	.	.	.	.	.	.	.	4	4	4	4	.	.	.	.	.	.	.	.	.	.

Besides, according to the analysis of [Dobraunig et al., 2016], the algebraic degree of this S-box is 2, which makes it vulnerable to algebraic attacks theoretically. And they required a differential branch number is to be 3, which doubles the number of active S-boxes. But the designers had their reasons to choose it that way. Because they wanted the S-box to be effectively implemented as bit-sliced. This approach was not going to only increase the speed but also would help prevent some side-channel attacks. And since they wanted the bit-sliced implementation of this S-box to be cheaper, they designed this S-box with reasonable security rather than being perfect.

In a similar manner, it is possible to perform linear cryptanalysis by analyzing S-boxes. When performing linear cryptanalysis, it is tried to find a relation between

plaintext bits, ciphertext bits, and key bits by constructing a linear approximation. That can be done by using Linear Approximation Table (LAT). In Table (LAT) 3.4 it could be seen that the maximum linear probability of the S-box is 2^{-2} and the linear branch number is 3 [Dobraunig et al., 2016].

Table 3.4: LAT of Ascon's Sbox

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
0	16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	.	.	.	.	.	.	8	.	.	4	4	.	.	-4	4	.	.	.	4	4	.	.	4	-4	4	.	-4	.	-4	.	-4	.
2	.	.	.	.	.	.	-8	8	.	.	4	4	.	.	4	4	.	.	4	4	.	.	-4	-4	.	.	.	.	.	.	.	.	.
3	.	8	.	.	.	.	.	.	.	4	.	4	.	4	.	-4	-8	.	.	.	.	.	.	.	4	.	4	.	4	.	-4	.
4	.	.	.	4	.	-4	.	.	.	.	4	.	.	4	-4	-4	.	.	4	.	-4	.	.	.	.	-8	.	-4	-4	.	4	-4
5	.	.	.	4	.	4	.	.	.	-4	.	.	.	.	.	-4	.	.	.	-4	4	.	-4	-4	4	.	-4	4	.	-8	.	-4
6	.	.	.	4	.	-4	.	.	.	.	.	-4	.	4	.	.	.	.	.	-4	-4	.	-4	-4	.	8	.	-4	-4	.	-4	4
7	.	.	.	-4	.	-4	.	.	.	4	4	4	.	.	-4	.	.	.	-4	.	-4	.	.	.	-4	.	-4	4	.	-8	.	4
8	.	.	.	.	.	.	.	.	.	.	4	4	.	.	-4	-4	.	.	.	.	.	.	.	.	.	8	-4	4	.	8	4	-4
9	.	.	.	.	.	.	.	-8	.	-4	.	4	.	4	.	4	.	.	4	4	.	.	-4	4	4	.	.	4	-4	.	.	4
a	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	4	4	.	.	4	4	.	8	4	-4	.	-8	4	-4
b	.	8	.	.	.	.	.	.	.	-4	4	.	.	-4	-4	.	8	.	.	.	.	.	.	.	4	.	.	-4	4	.	.	4
c	.	.	-8	4	-8	-4	.	.	.	.	.	4	.	-4	.	.	.	.	-4	.	4	.	.	.	.	.	4	.	-4	.	.	.
d	.	.	.	-4	-8	4	.	.	.	4	-4	-4	.	.	-4	.	.	.	.	4	-4	.	-4	-4	4	.	.	.	.	.	4	.
e	.	.	.	-4	8	-4	.	.	.	.	-4	.	.	-4	-4	-4	.	.	.	4	4	.	-4	-4	.	.	4	.	-4	.	.	.
f	.	.	8	-4	-8	-4	.	.	.	-4	.	.	.	.	.	-4	.	.	4	.	4	.	.	.	-4	.	.	.	.	.	-4	.
10	.	.	.	.	.	.	-8	.	.	4	.	-4	-4	.	-4	.	.	.	.	.	4	-4	4	4	4	.	-4	.	-4	.	-4	.
11	.	.	.	.	.	.	.	.	-8	.	-4	4	-4	-4	.	.	.	8	4	-4	-4	-4	.	.	.	.	.	.	.	.	.	.
12	.	-8	.	.	.	.	.	.	.	-4	4	.	-4	.	.	-4	.	.	-4	4	-4	-4	.	.	4	.	4	.	4	.	-4	.
13	.	.	.	.	.	.	-8	-8	.	.	.	.	4	-4	4	-4	.	.	.	.	-4	4	4	-4	.	.	.	.	.	.	.	.
14	.	.	.	4	.	4	.	.	.	4	4	-4	-4	-4	.	-4	.	.	4	.	.	4	-4	4	-4	.	4	4	.	.	.	4
15	.	.	.	4	.	-4	.	.	.	.	.	-4	4	.	-4	4	.	8	.	4	.	4	.	.	.	.	.	4	4	.	-4	-4
16	.	.	.	-4	.	-4	.	.	.	4	.	.	-4	4	4	.	8	.	.	-4	.	4	.	.	4	.	4	4	.	.	.	-4
17	.	.	.	4	.	-4	.	.	8	.	-4	.	-4	.	.	.	.	.	4	.	.	-4	4	-4	.	.	.	4	4	.	4	4
18	.	.	.	.	.	.	.	-8	.	4	4	.	-4	.	.	4	.	.	.	.	4	-4	-4	-4	-4	.	.	-4	4	.	.	-4
19	.	.	.	.	.	.	.	.	.	.	.	.	4	-4	-4	4	.	-8	4	-4	-4	-4	.	.	.	.	4	4	.	.	-4	-4
1a	.	8	.	.	.	.	.	.	.	-4	.	-4	-4	.	4	.	.	.	-4	4	-4	-4	.	.	-4	.	.	4	-4	.	.	-4
1b	.	.	.	.	.	.	.	.	8	.	-4	4	-4	-4	.	.	.	.	.	.	-4	4	-4	4	.	.	-4	-4	.	.	-4	-4
1c	.	.	8	4	.	-4	.	.	.	4	.	.	4	-4	4	.	.	.	-4	.	.	-4	-4	4	4	.	.	.	.	.	4	.
1d	.	.	.	-4	.	4	.	.	8	.	4	.	4	.	.	.	.	8	.	-4	.	-4	.	.	.	.	4	.	-4	.	.	.
1e	.	.	.	4	.	4	.	.	.	4	-4	4	4	4	.	-4	8	.	.	4	.	-4	.	.	-4	.	.	.	.	.	-4	.
1f	.	.	8	4	.	4	.	.	.	.	.	4	-4	.	-4	4	.	.	-4	.	.	4	4	-4	.	.	4	.	-4	.	.	.

3.4 General Structures of Ascon

Ascon has two instances, Ascon-128 and Ascon-128a. They use the same length key, nonce, tag, round number which are respectively $128, 128, 128$. But their data block size and round numbers are different. While data block size is 64 in Ascon-128, it is 128 in Ascon-128a. The round number a is 12 in both Ascon-128 and Ascon128-a. And the round number b is 6 and 8 for Ascon-128 and Ascon128-a respectively.

The encryption of Ascon has 4 steps: Initialization, processing of associated data, processing the plaintext, and finalization. The encryption process takes inputs as a secret key K, a nonce N with 128-bits, associated data A of arbitrary length if it exists, and plaintext P that has arbitrary length. It produces ciphertext C that has the same length as the plaintext P. It also produces the authentication tag T with 128-bits. The mode of operation of Ascon was illustrated in Figure 3.2.

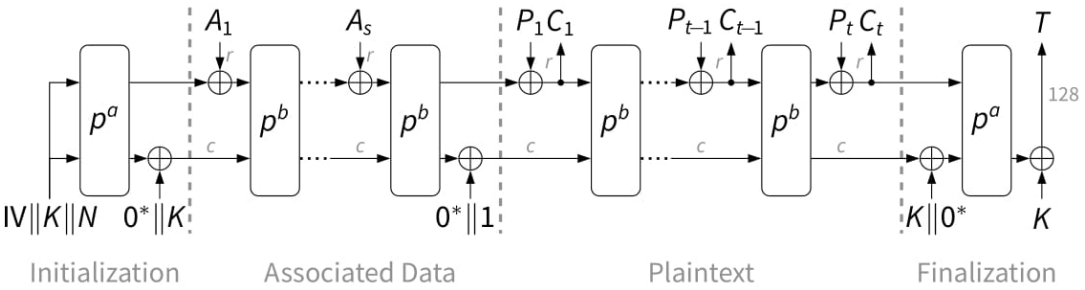

Figure 3.2: The duplex sponge mode for Ascon v1.2 authenticated encryption, figure is from its official website https://Ascon.iaik.tugraz.at/images/aead_encrypt.pdf

The decryption process takes input as 128-bit key K, 128-bit nonce N, tag T, ciphertext C, and associated data A if it exists. When decrypting the cipher if the tag verification is correct, it produces the plaintext P. If not, it gives an error.

In the initialization process, the 64-bit IV, 128-bit secret key, 128-bit nonce are form the 320-bit state S. This state contains five 64-bit words as x_0, x_1, x_2, x_3, x_4. In here, x_0 word is IV, x_1, x_2 words are secret key, and x_3, x_4 words are nonce.

$$S = S_r||S_c = x_0||x_1||x_2||x_3||x_4$$

S could be interpreted as big-endian format. Rate r and capacity $320 - r$ expressed in the outer part S_r and inner part S_c respectively. The value of IV is $80400c0600000000$ for Ascon-128 and $80800c0800000000$ for Ascon-128a.

In the initialization part, the state updates with permutation p as α times which represented as p^α. Then the secret key K is XORed with the state. The permutation p that updates the state has some layers. In here, first, a 5-bit constant is added to the x_2. These constants are changed according to the round numbers α and β. They can be seen in Table 3.5.

Table 3.5: Constants of Ascon. While p^{12} and p^6 are used for Ascon-128, p^{12} and p^8 are used for Ascon128-a.

p^{12}	p^8	p^6	**Constant**	p^{12}	p^8	p^6	**Constant**
0			00000000000000000f0	6	2	0	0000000000000000096
1			00000000000000000e1	7	3	1	0000000000000000087
2			00000000000000000d2	8	4	2	0000000000000000078
3			00000000000000000c3	9	5	3	0000000000000000069
4	0		00000000000000000b4	10	6	4	000000000000000005a
5	1		00000000000000000a5	11	7	5	000000000000000004b

Then the substitution layer that uses the $5x5$ S-box updates the state S 64 times in parallel. That can be seen in Figure 3.3.

Figure 3.3: Substitution layer of Ascon. Figure was taken from the cipher's official websitehttps://ascon.iaik.tugraz.at/images/state_vertical_small.png

After the substitution layer, the linear diffusion layer comes. In this layer, the function $\Sigma_i(x_i)$ is applied to each word xi. This $\Sigma_i(x_i)$ uses simple boolean operations. Therefore it is fast.

$$x_i \leftarrow \Sigma_i, 0 \leq i \leq 4$$

$$x_0 \leftarrow \Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)$$

$$x_1 \leftarrow \Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)$$

$$x_2 \leftarrow \Sigma_2(x_2) = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6)$$

$$x_3 \leftarrow \Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)$$

$$x_4 \leftarrow \Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)$$

In the associated data processing part, a single 1 and the smallest number of 0s are added to the associated data A. Then A is divided into blocks that have a size of $r - bit$. No padding is required if A is empty. After that, each block of A XORed with the first r bits S_r, then b-round permutation is applied to S. Finally, S is XORed with the 1-bit domain separation constant.

In the plaintext processing part, padding operation is applied to the plaintext P such that the addition of single 1 and the smallest number of 0s, if it is necessary. Then P is split into blocks that have a size of r-bit. In the encryption, the first r bits of S_r and one padded plaintext block is XORed with each other and produce the ciphertext block. State S is updated with permutation p^b for each block to the last one. The last output block is truncated to the size of the last plaintext block, which is unpadded. So the total length of the ciphertext C would be the same as the plaintext.

In the finalization part, the internal state and the secret key K are XORed with each other, then the permutation p^a updates the state. The last 128-bits of the key and the last 128-bits of the state XORed with each other to produce the Tag T. So after the encryption, there would be a tag and the ciphertext. And after the decryption, if the received tag value is the same as the calculated one, plaintext could be obtained.

3.5 Differential-Linear Distinguishers of Ascon

Ascon uses MonkeyDuplex construction as a mode of operation. In MonkeyDuplex construction, obtaining the inner part of the state is as important as obtaining its key [Bertoni et al., 2012]. Hence its security is based on the uniqueness of the nonce. Nonce means "number used once". In a communication, the arbitrary number nonce can be used just once as its name suggests. While implementing a cipher in a system, sometimes this property is overlooked. Therefore the cryptanalytic attacks are given according to whether this feature is taken into account or not. Namely, a nonce-respecting scenario or a nonce-misuse scenario.

The attacks on Ascon can be categorized into two sections: key recovery attacks and forgery attacks. Key recovery attacks target the initialization phase or processing plaintext phase whether it is a nonce-respecting scenario or not, respectively.

In key recovery attacks, differences are given to the nonce, namely the words x_3 and x_4. And since the plaintext is XORed with x_0 to generate the ciphertext, the differences in the output are only examined for x_0.

Forgery attacks aim to forge tags and therefore target the finalization phase. In this scenario, differences are given to the word x_0. And other words should not have any difference since the analyst has no control over them. And because the tag is generated from words x_3 and x_4 with the key, the output differences can be observed in them.

In this work, we mainly focused on the differential-linear analysis of Ascon-128. This method was applied in [Dobraunig et al., 2015b] for key recovery attacks to the 4 and 5 rounds of Ascon. They used a 2-round differential characteristic that has a probability of 2^{-5} with the 2-round linear approximation that has a probability of 2^{-8} and obtained a 4-round characteristic with probability $2pq^2 = 2^{-20}$. But the practical results were about 2^{-2} according to that study. The huge gap between these results was explained as the existence of multiple characteristics on the differential and linear analysis part. Then [Bar-On et al., 2019] showed that this theoretical bias is actually 2^{-5} by introducing DLCT. As can be seen, there were still considerable differences with this theoretical bias 2^{-5} and practical bias 2^{-2}, and it was explained as the slow diffusion with the existence of multiple characteristics.

Recently, [Tezcan, 2020] improved the results given by [Dobraunig et al., 2015b] by using 2-round probability one truncated differential with the 2-round linear approximation and provided 4-round differential-linear characteristics. Then they used it to attack 4 and 5 rounds of Ascon. Actually, in the previous work of Tezcan [Tezcan, 2016], they observed the S-box of Ascon has 35 undisturbed bits and they used it to build a 3.5-round probability one truncated differential distinguisher for Ascon. But they did not use this 3.5 round distinguisher when building the differential-linear distinguisher, because it has contained differences in words x_0, x_3, and x_4. But to perform a key recovery attack, the input differences could have been only in the nonce,

namely x_3, and x_4. So they used a 2-round probability one truncated differential distinguisher that only had an input difference on x_3 and x_4 instead. And they combined it with a 2-round Type-II linear approximation with bias 2^{-8} provided by [Dobraunig et al., 2015a]. They have used the Type-II characteristic because the last round of the approximation should have masks only in word x_0, and the rest should have been free from any masks. The characteristics they have used to build a 4-round differential-linear distinguisher can be seen in Table 3.6.

Table 3.6: 2-round truncated probability one truncated differential with the combination of 2-round linear approximation with bias 2^{-8} in hexadecimal notation. They were used for building 4-round differential-linear distinguisher by [Tezcan, 2020].

Round	**4- Round Differential-Linear Path for Ascon-128**
I	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000001000000000000000000000000000000000000000000000000000000
	0000000001000000000000000000000000000000000000000000000000000000
S1	000000000?0000000000000000000000000000000000000000000000000000000
	000000000?0000000000000000000000000000000000000000000000000000000
	000000000?0000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	000000000?0000000000000000000000000000000000000000000000000000000
P1	000000000?0000000000000000?00000000?0000000000000000000000000000
	000000?00?0000000000000000000000000000000000000?0000000000000000
	000000000??0000?0000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	000000000?000000?0000000000000000000000000000000?00000000000000
S2	000000?00??0000??000000000000?00000000?0000000000?0?00000000000000
	000000?00??0000??000000000000?00000000?0000000000?0?00000000000000
	000000?00??0000??000000000000000000000000000000?0?00000000000000
	000000?00??0000??000000000000?00000000?0000000000?0?00000000000000
	000000?00?000000?000000000000?00000000?0000000000?0?00000000000000
P2	0?0?0??00??0?0???00000000?00??0000??0??0000??00??0?00000?0000000
	000?00??0??0??0??000000?0?00?00000?00?0000000?0????000??00000000
	000000??0????00???000??000000000000000000000000000????00?0?0000000
	0?0?00?00??0000??00??00?0????000??000??000000?0??0?000?000?0?000
	00000??00?000??0?000000?0?0??000000?0?000000?00??0?0000?0?0000000
Round	**State**
0	....2.4.. .....2.4.1 .....2.....8. .....2.....8.
1	....1 1
2	9224b6d24b6eda49 ....

According to this, the new theoretical bias was 2^{-15} instead of 2^{-20} and the practical results were slightly better. The practical results of these biases are $2^{-2.68}, 2^{-3.68}, 2^{-3.30}$, and $2^{-2.30}$ in Dobraunig's work while key bits are (0, 0), (0, 1), (1, 0), and (1, 1) in the activated S-box, respectively. They were required 2^{12} samples to capture these key bits. In Tezcan's work, the improved results were $2^{-2.41}, 2^{-1.68}, 2^{-2.41}$ and $2^{-1.68}$ and the data complexity was reduced to 2^8 for capturing the first key bit. Besides, while capturing the whole 128-bit key required with time complexity of $64 \cdot 2^{12} = 2^{18}$ in Dobraunig's approach, it required $64 \cdot 2 \cdot 2^8 = 2^{15}$ in Tezcan's.

To perform this attack, Tezcan used 2^{24} random nonces and performed this experiment with 1000 random keys for the 4-round permutation [Tezcan, 2020]. They repeated this experiment for 4 possible key pairs. Although they had better results in capturing the first key bit, which was the usage of 2^8 data, they could not distinguish the second key bit because they observed the same biases regardless of the second key bit. So they captured the second key bit using another 2^8 samples and then they have got the whole 128-bit key by rotating the initial difference.

To extend this attack to 5-rounds, Dobraunig performed some precomputations and completed the attack with about 2^{36} time complexity [Dobraunig et al., 2015a]. Tezcan got 2^{32} time complexity with a different technique by using a 3-round differential and 2-round linear approximation [Tezcan, 2020]. Then in the same study, they extended this experiment to 6-rounds by using 2^{42} random nonces and repeating the experiment with 128 random keys by rotating the input difference to every possible position. This operation was performed with $2 \cdot 2^{42} \cdot 128 \cdot 64 \cdot 4 = 2^{58}$ complexity.

In this work, we studied the approach of [Tezcan, 2020] and used it to find practical 5-round differential-linear distinguishers for Drygascon. [Tezcan, 2020] used a Type-II linear approximation for finding this distinguisher. In this way, it was going to be possible to attack the cipher. Since the additional functions of Drygascon change the state of it, using Type-II approximation would not have a significant meaning. Therefore we mainly focused on Type-I approximation. But for the sake of comparison, we also provided a 4-round differential-linear distinguisher that has Type-II linear approximation. We present them in the next chapter.

CHAPTER 4

DRYGASCON

Drygascon [Riou, 2019] is a cipher suite that provides AEAD and hashing functionality. It was selected as a second-round candidate in NIST's Lightweight Cryptography competition. It did not make it to the final round. It uses GASCON permutation as a primitive, which is a generalized variant of Ascon. The aim of the GASCON is to increase the Ascon's 128-bit security. Drygascon has two instances: Drygascon-128 and Drygascon-256, but the primary submission was Drygascon-128. Drygascon128 has a 128-bit block size and 3 different key sizes: These are small setup with 128-bit, fast setup with 256-bit, and full setup with 448-bit. Therefore, it provides a security level with a minimum of 128-bit. And Drygascon-256 has a 128-bit block size and a 256-bit key size. Therefore, it provides a security level of 256-bit. The security in both versions is based on the assumption that nonce will not be reused. Also in Drygascon-128, one should not perform encryptions more than 2^{64} bytes with the same key. This number is 2^{128} for Drygascon-256. Drygascon uses a new construction DrySponge as a mode of operation, which is based on Duplex Sponge construction [Bertoni et al., 2011]. But the combination of the input with the state and the extraction of output from the state is different in DrySponge than the Duplex Sponge construction.

In Drysponge, the security claims are related to the size of the capacity, as in the other sponge constructions. The state is created with a key setup function. This function produces capacity c and optional x parameter.

The absorbing stage is carried out with a function called F. The squeezing stage is carried out with a function called G. Domain separation is made with the input in function F, which is a different property from regular sponge constructions. F takes

input state (c, x), 128-bit data input i, and domain separator DS as input. DS could be partially modified by the attacker. The input i is assumed to be controllable by the attacker. (c, x) is always kept secret. Then it outputs the modified state (c, x), 128-bit input i and r. F function uses the Mix function to process i and DS, and the G function to generate r. F fiunction contains the following stages:

- Mix: This function handles the input bits. It is claimed that a known difference to (c, x) cannot be injected by manipulating i or DS. This function is a design parameter for Drysponge, as it can be implemented in various ways.
- Core: It is included in the G function. It contains an iterative permutation that processes hidden data, namely GASCON. So the security of the algorithm is based on the security of this permutation.
- Accumulate: It is included in the G function. It simply operates XOR to all input bits to generate the output bits r.

Its AEAD uses a precomputed optional static data S, a nonce whose size is a design parameter, and associated data A, plaintext P, ciphertext C and tag T. Encryption operation takes input as secret key K, optional static data S, a nonce N, associated data A, and plaintext P. And it outputs the ciphertext C and a tag T. Decryption operation takes input as secret key K, an optional static data S, a nonce N, associated data A, ciphertext C and tag T. And it outputs the plaintext P.

4.1 Gascon Permutation

Gascon stands for 'Generalized Ascon'. It is a concrete sampling of the Drygascon's CoreRound function and it was designed to support a wider size than the Ascon's permutation. It has different key sizes, and rotation sizes are also varied in the linear layer. Drygascon-128 uses $GASCON_{C5R11}$, which is a 320-bit state formed by 64-bit words. It uses Ascon's $5x5$ S-box except represents it in little-endian. And unlike Ascon, its round number is 11 instead of 12. However, Drygascon-256 uses $GASCON_{C9R12}$, which has 12 rounds and it uses 576 bits from nine 64-bit words.

Hence its S-box is a $9x9$ one. Unlike Ascon, constant addition does not depend on the total number of rounds. The constants are added using Equation (3).

$$x_{2,4} = x_{2,4} \oplus ((0xfull - r) \ll 4)|r \tag{3}$$

In Drygascon-128, 5-bit constant is added to the x_2 , where $r = 1 + round$. And in Drygascon-256 5-bit constant is added to the x_4, where $r = round$. In here, $round$ stands for the current round.

Linear layer of Drygascon-128 is similar to the linear layer of Ascon-128. But in Drygascon two rotations are different, namely Σ_1 and, Σ_4. They were changed into 39 to 38 in Σ_1 and 41 to 40 in Σ_4. The rotation for both Drygascon-128 and Drygascon-256 are as follows:

$$\Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)$$

$$\Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 38)$$

$$\Sigma_2(x_2) = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6)$$

$$\Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)$$

$$\Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 40)$$

$$\Sigma_5(x_5) = x_0 \oplus (x_5 \ggg 31) \oplus (x_5 \ggg 26)$$

$$\Sigma_6(x_6) = x_1 \oplus (x_6 \ggg 53) \oplus (x_6 \ggg 58)$$

$$\Sigma_7(x_7) = x_2 \oplus (x_7 \ggg 9) \oplus (x_7 \ggg 46)$$

$$\Sigma_8(x_8) = x_3 \oplus (x_8 \ggg 43) \oplus (x_8 \ggg 50)$$

And since each word is in a bit interleaved representation, the rotation is different from Ascon's. According to [Riou, 2019], every 64-bit word rotates once with an odd shift to make sure that a difference in half of an input word will be propagated to the other half of the matching output word. Namely *intra word diffusion*. See Algorithm 1 that provided by [Riou, 2019] for detailed information.

Algorithm 1 BiRotR: 64 bit rotation in bit interleaved format [Riou, 2019]

Input: in: 64 bit value in bit interleaved format, $shift$

Output: out: in rotated right by $shift$

$i0 \leftarrow in \ \& \ 0xFFFFFFFF$

$i1 \leftarrow in \gg 32$

$shift2 = [x/2]$

if $shift\&1$ **then**

  $t \leftarrow i1 \ggg shift2$

  $i1 \leftarrow i0 \ggg ((shift2 + 1) mod 32)$

  $i0 \leftarrow t$

else

  $i0 \leftarrow i0 \ggg shift2$

  $i1 \leftarrow i1 \ggg shift2$

end if

return $(i1 \ll 32)|i0$

4.2 Differential-Linear Distinguishers of Drygascon

Since Ascon and Drygascon have similar designs, [Riou, 2019] stated that the cryptanalysis of Ascon can be applied on Drygascon with some modifications. Therefore [Tezcan, 2020] applied the differential-linear analysis that they performed on Ascon to Drygascon. When performing the analysis they only focused on $GASCON_{C5R11}$ permutation, the constrained version of Drygascon. Because [Riou, 2019] stated the Mix128 function does not really have any effect on the Drygascon's security and analysis should be performed on only $GASCON_{C5R11}$ permutation.

[Riou, 2019] presented some analysis results that they performed. These were linear approximations for various round numbers along with a truncated differential distinguisher of Drygascon. With that, they presented the undisturbed bits of Drygascon and stated there was no truncated differential distinguisher with probability one longer than 3-round, unlike Ascon. And as can be seen in Table 4.1, they provided a 3-round probability one truncated differential distinguisher.

Table 4.1: 3 round probability one truncated differential that was reported wrongly for $GASCON_{C5R11}$ by [Riou, 2019]

Round	**3- Round Truncated Differantial Distinguisher of** $GASCON_{C5R11}$
	0000000100000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
I	0000000000000000000000000000000000000000000000000000000000000000
	0000000100000000000000000000000000000000000000000000000000000000
	0000000100000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000?00000000000000000000000000000000000000000000000000000000
S1	0000000?00000000000000000000000000000000000000000000000000000000
	0000000?00000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000?000000000000?0000000000000000000?00000000000000000000000
P1	0000?00?000000000000000000000000000000?0000000000000000000000000
	00?0000?000000000000000000000000000000000000000000000000000000?0
	0000000000000000000000000000000000000000000000000000000000000000
	00?0?00?000000000000?00000000000000000?0?000000000000000000000?0
	00?0?00?000000000000?00000000000000000?0?000000000000000000000?0
S2	00?0?00?000000000000?00000000000000000?0?000000000000000000000?0
	00?0?00?000000000000?00000000000000000?0?000000000000000000000?0
	00?0000?000000000000?0000000000000000000?000000000000000000000?0
	00?0?0??000000000000???00?000?0?000000?0?0?00000?0000000?0?00??0
	???0?00??0?0000?0?00?000000000000000?0??0?00?0000000?0?00000000?0
P2	0??0?0???00000000?00?000000000??0?0?0??0?0000000000?0000000?00?0
	?0?0?00?0000000?0000?0?000000???0?0?00?0?00?0000000000000?0?00?0
	?0?00?0?000000?0000??000000?0000000?0000?0?00000?000?000000000?0
	???0?????0?000??0?0????00?0?0???0?0?0??0?0??0000?00???00????0??0
	???0?????0?000??0?0????00?0?0???0?0?0??0?0??0000?00???00????0??0
S3	???0?????0?000??0?0??0?0000?0???0?0?0??0?0??0000?00???000?0?00?0
	???0?????0?000??0?0????00?0?0???0?0?0??0?0??0000?00???00????0??0
	???0?????0?000??0?0????00?0?0???0?0?0??0?0??0000?00???00????0??0
	?????????0????????????????????????????????????0???0????????????0
	?????????0??????0???????0???????????0???????00???0??????????????
P3	?????????0???0????0????0????????????????????0??0???????0????????
	???????????????????????0??????????0?0??0?0???????0??????????????
	????????????????????????????0???????????????0?0?????????????????

But [Tezcan, 2020] refuted Riou's claim by providing a 3.5-round probability one differential distinguisher. Because as can be seen in Table 4.2, the non-zero values in the third round actually were revealing some characteristics for the next layer.

Table 4.2: 3.5 round probability one truncated differential where the intermediate differences are reported wrongly [Tezcan, 2020].

Round	3.5- Round Truncated Differential Distinguisher of $GASCON_{C5R11}$
I	10000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
S1	?0000000000000000000000000000000000000000000000000000000000000000000000000000000
	10000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	?0000000000000000000000000000000000000000000000000000000000000000000000000000000
	?0000000000000000000000000000000000000000000000000000000000000000000000000000000
P1	?000000000000000000?000000000000000000000000000000000000000?000000000
	10000000000001000000000000000000010000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	?000000000000000000000000?000000000000000000000000000?000000000
	?0000000000?000000000000000000000000000000000000000000000000?000
S2	?000000000000??0000?00000000?00000?000000000000000000000??0000?000
	?000000000000??0000?00000000?00000?000000000000000000000??0000?000
	?000000000000?10000000000000?000001000000000000000000000?0000?000
	?000000000000?10000?00000000?000001000000000000000000000??0000?000
	?000000000000??0000?00000000?00000?000000000000000000000??0000?000
P2	?000?0000000???000??0000?00?00??0???0000??0000?00?0?00??0000?000
	?00?0000?000??0000?00000????00??0?0??0000?000??0000?00??0000?000
	?10000000?10?1000000000??00???000100000000?10000000?00?0??0?01?
	?000000?1000????00?0?0?0010?0000010?10000?0000000??000??0000?000
	?000000?0000??0000???000??0?00?00???0000??000??0000000??0000?000
S3	??0??00???0????00???0?????????0????000??0????00????0??0??0?0??
	??0??00???0????00???0?????????0????000??0????00????0??0??0?0??
	?10?000???10????00???0?????????0????000??0?1??00????0??0??0?01?
	?10??00???10????00???0?????????0????000??0?1??00????0??0??0?01?
	?00??00??000????00???0?0????00??0????000??000??00???00??0000?000
P3	??0?????????????????0???????
	??0???
	???
	??????????????????0?????????????????????????1??????????????????1???????
	?????????0???????0???0?0????00???????000????0???????????0???????
S4	??a??????????????????b???????
	??a??????????????????b???????
	??a??????????????????b???????
	??a??????????????????b???????
	??a??????????????????b???????

Since [Riou, 2019] have already provided some linear approximations by using linearTrails tool [Dobraunig et al., 2015a], Tezcan used the 3-round one to build a 5

round differential-linear distinguisher [Tezcan, 2020]. Unlike the distinguisher that they [Tezcan, 2020] have found for Ascon, they used a Type-I linear approximation instead of Type-II. And they gave the initial difference in x_1 and x_2, instead of x_3 and x_4. That can be seen in Table 4.3.

Table 4.3: 2-round truncated probability one truncated differential with the combination of 3-round Type-I linear approximation with a bias of 2^{-15}. Since the differential part is wrong, this 5-round distinguisher [Tezcan, 2020] is also wrong.

Round	**5- Round Differential-Linear Distinguisher of** $GASCON_{C5R11}$
I	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000010000000000000000000000000000000000000000000000000000
	00000000000000000000000000010000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
S1	00000000000000000000000000?00000000000000000000000000000000000000000000000000000
	00000000000000000000000000?00000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000?00000000000000000000000000000000000000000000000000000
P1	000000000000?000000000000000?0000000000000000000000?0000000000000000
	0000000?00000000000000000000?0000000000000000000000000000000000?0000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	000000?000000000000000000000?0000000000000000000000000000000?0000000000
S2	000000??0000?000000000000000?0000000000000000000000?00000?0000?0000
	000000??0000?000000000000000?0000000000000000000000?00000?0000?0000
	000000??00000000000000000000?0000000000000000000000000000?0000?0000
	000000??0000?000000000000000?0000000000000000000000?00000?0000?0000
	000000??0000?000000000000000?0000000000000000000000?00000?0000?0000
P2	000000??0000??0000?00000???000?000?00000?0000?00?00000?0000???00
	000000??0000?00000???000???00?00000?000??0000?00?00000?0000?0?00
	000??0??000000000000000??00??000000000??000000000000?00?0??0?0000
	0??000???000?0?0000?0?0000?00000000?0000000?0000??0000?0000?0??0
	000000??0000??0000??0000?0?00000000??000??0000000?00000?0000??000

Round	**State**				
0	1.....8.21.1.21	.....	.....	.....18.	1.....8.21.11a.
1	.....	.....	.....	.....8..1....1	.....8..1....1
2	.....	.....	.....	.....	1
3	.....	e37c4f1b6e8d53e6	e.8629e8e4b766af	.....	.....

Because building the state of Drygascon is different than Ascon, the nonce is not in the words x_3 and x_4. So there was no point in giving the initial difference to x_3 and x_4 and observing the output difference only in x_0. So they just aimed to find some characteristics [Tezcan, 2020]. As a result of this, they calculated the total bias as

$2pq^2 = 2 \cdot 1 \cdot (2^{-15})^2 = 2^{-29}$. And they stated that they can distinguish it from a random permutation by using $2^{61.28}$ samples according to Algorithm 1 of [Blondeau et al., 2011].

Since Tezcan has already presented the theoretical 5-round differential-distinguishers for $GASCON_{C5R11}$ [Tezcan, 2020], we decided to verify its result experimentally. But Tezcan's claim was they needed around $2^{61.28}$ samples to distinguish it from a random permutation [Tezcan, 2020], and we did not have such computational power. Because 5-round encryption of Drygascon with 2^{35} data took 2 hours in our Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz 2.59 GHz system. So it would have taken us over 233016 years to try that with more than 2^{60} data. But based on our experience from Ascon, we knew there was going to be a gap between the theoretical and practical results. So we concluded that this experiment was applicable because while the theoretical bias was around 2^{-15} in the 4-round differential-linear distinguisher of Ascon, the practical was around 2^{-2}. Since the Ascon and Drygascon had similar designs, the practical bias that comes from the linear approximation should have been at least around 2^{-7} instead of 2^{-15}. And according to Matsui's Piling-up lemma [Matsui, 1993], data complexity should have been around $p^{-2}q^{-4} = 1^{-2} \cdot (2^{-7})^{-4} = 2^{28}$.

First, we implemented the $GASCON_{C5R11}$ ourselves with using C programming language. Then we wrote a 5-round differential-linear distinguishing algorithm of $GASCON_{C5R11}$, again by using C programming language. We used Tezcan's initial difference with Riou's linear approximation when testing this distinguisher. We used 2^{30} random data as a beginning. But the results were around $1/2$, and that meant the cipher was random and there was no bias q to examine. We concluded since Drygascon's permutation has some differences from Ascon's, maybe we needed more data to test it. So we increased our random data size from 2^{30} to 2^{35}, then 2^{38}. Results were still random. So we decided to examine where the bias would converge to decide how much data we would need. For that to happen, we checked the bias in every 2^{29} data to see if it really converges to some point, or it is monotone increasing. But it did not seem like it would converge to some point. The explanation for this, the truncated differential with the given initial difference were not actually matching with the linear approximation. Since Drygascon is rotation invariant, we rotated the difference 64 times and checked its compatibility with the linear approximation. When this did not

give the result we expected, we tried it with smaller rounds and met with high biases. Finally, we came to the conclusion of either the truncated differential path was wrong or the linear approximation. Therefore we decided to check them first.

We began by checking the correctness of the truncated differential distinguisher. We wrote our own C code for checking the diffusion of the initial difference provided by [Tezcan, 2020]. Surprisingly, we observed this diffusion differs from probability one truncated differential provided by [Tezcan, 2020]. That was why the experiment did not work, truncated differential distinguisher was wrong, therefore it was not compatible with the linear approximation. Our corrected results for this 2-round truncated differential distinguisher can be seen in Table 4.4

Table 4.4: Corrected results for 2-round probability one truncated differential of $GASCON_{C5R11}$. The wrong one was reported by [Tezcan, 2020] and was used to build a 5-round differential-linear distinguisher.

Round	**2- Round Truncated Differantial of** $GASCON_{C5R11}$
I	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000010000000000000000000000000000000000000
	0000000000000000000000000010000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
S1	000000000000000000000000?000000000000000000000000000000000000000
	000000000000000000000000?000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	000000000000000000000000?000000000000000000000000000000000000000
P1	00000000?000000000000000?00000000?000000000000000000000000000000
	000000000000?000000000000?000000000000000000000000000000?0000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	000000000000000?000000000000?000000000000000000000000000000000?00
S2	00000000?0000??000000000000?00000000?000000000000000000000?0000?00
	00000000?0000??000000000000?00000000?000000000000000000000?0000?00
	000000000000??000000000000?000000000000000000000000000000?0000?00
	00000000?0000??000000000000?00000000?000000000000000000000?0000?00
	00000000?0000??000000000000?00000000?000000000000000000000?0000?00
P2	00?0000??0000??0000000?000???000000?00?0000?00000?0000???0000?00
	???00000?0000??00000000?00???000000?00?0000??000?00000?0?0000?00
	0000000000000??0??0000000??00??0?000000000000??000000000?0??0?00
	0?0000?0?000???000??000000?0000?00??0000?0000000?0000??0?0000?00
	0??0000??0000??000000000000?0?000000?0000000??000??00000??0000?00

Before moving further, we decided to check all other truncated differential distinguishers that were provided so far. We checked Riou's 3-round truncated differential distinguisher, we observed the same situation applies to that also. We believe the reason was the bits move in the opposite direction in provided code than described in the linear layer. And that may have led to further erroneous analysis. The corrected results to this 3-round truncated differential distinguisher can be seen in Table 4.5.

Table 4.5: Corrected results of 3-round probability one truncated differential of $GASCON_{C5R11}$ that was reported wrongly by [Riou, 2019].

Round	3- Round Truncated Differantial of $GASCON_{C5R11}$
	0000000100000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
I	0000000000000000000000000000000000000000000000000000000000000000
	0000000100000000000000000000000000000000000000000000000000000000
	0000000100000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000?00000000000000000000000000000000000000000000000000000000
S1	0000000?00000000000000000000000000000000000000000000000000000000
	0000000?00000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000?000000000000000000?0000000000?00000000000000000000000000
P1	0000000?00?0000000000000000000000000000?000000000000000000000000
	0000000?0000?0000000000000000000000000000000000?0000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000?00?0?0000000000000?0000000000?0?0000000?0000000000000000
	0000000?00?0?0000000000000?0000000000?0?0000000?0000000000000000
S2	0000000?00?0?0000000000000?0000000000?0?0000000?0000000000000000
	0000000?00?0?0000000000000?0000000000?0?0000000?0000000000000000
	0000000?0000?0000000000000?0000000000?000000000?0000000000000000
	0000000??0?0?00?0?000?00???00000000?0?0?0000000??00?0?0000000?00
	0000?0??00?0???00000000000?00?0?00?00?0??0?0000?00000000?0?00000
P2	000000???0?0??0??000000000?00?0000000?0??0?0?00?00?0000000?00000
	0000000?00?0?0????000000?0?0000?00?00?0?00?0?00?00?0?00000000000
	?000000?0?00?0?0000?000000??0000000?0?0000?0000?000000000?000?00
	?000?0?????0??????0?0?00????0?0?00??0?0??0?0?00??0????00???00?00
	?000?0?????0??????0?0?00????0?0?00??0?0??0?0?00??0????00???00?00
S3	?000?0?????0??????0?0000?0??0?0?00??0?0??0?0?00?00?0?000??00?00
	?000?0?????0??????0?0?00????0?0?00??0?0??0?0?00??0????00???00?00
	?000?0?????0??????0?0?00????0?0?00??0?0??0?0?00??0????00???00?00
	?????0??0????????????0???0
	?????0????????????????0???????0?00???????0??????????????????0???
P3	?0???0?????????????????0????0????0????????????????????0???????0?
	???????????????????????0????????????0?0??0?0????????????????0???
	??????????????????0??????????????0?????????????????????????????0

We also checked the 3.5-round truncated differential provided by [Tezcan, 2020]. As can be expected, the results were inaccurate here, also. However, they were right about the existence of a 3.5-round truncated differential distinguisher despite Riou's claim [Riou, 2019]. Moreover, surprisingly the differentials in the last round were actually accurate. It was just distributing differently in the middle rounds. We corrected this distinguisher and that can be seen in Table 4.6.

Table 4.6: Corrected results of 3.5-round probability one truncated differential of $GASCON_{C5R11}$ that was reported wrongly by [Tezcan, 2020]. Despite the middle rounds, S4 is same with Tezcan's.

Round	**3.5- Round Truncated Differantial of** $GASCON_{C5R11}$
I	10000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
S1	?0000000000000000000000000000000000000000000000000000000000000000000000000000000
	10000000000000000000000000000000000000000000000000000000000000000000000000000000
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	?0000000000000000000000000000000000000000000000000000000000000000000000000000000
	?0000000000000000000000000000000000000000000000000000000000000000000000000000000
P1	?000000000000000?000000000000000000000000000?000000000000000000000000
	10000000000000000000100000000000000000000000000000000000000000000000000010
	00000000000000000000000000000000000000000000000000000000000000000000000000000000
	?0000?000000000000000000000000000000000?000000000000000000000000
	?000000000000000000?000000000000000?000000000000000000000000000
S2	?0000?00000000?0000??000000000000000?0000??000000000000000000000?0
	?0000?00000000?0000??000000000000000?0000??000000000000000000000?0
	?0000?000000000000001?000000000000000?0000?000000000000000000000010
	?0000?00000000?00001?000000000000000?0000??000000000000000000000010
	?0000?00000000?0000??000000000000000?0000??000000000000000000000?0
P2	???00?00?0000??000???0000000?000000?0000??00?0?00?0000??0000???0
	???00????00000?0000??000?0000?00000?0000??00?0000??000?0000??0?0
	?00???00??0000000001?01?00000001?10?0??0?00?00000001?00000000010
	?0000?0100?0?0?00????0001?000000000?0000??000??0000000?00001?010
	?0?00?0??000???0000??0000?000000000?0000??0000000??000??0000???0
S3	???????????0???00????0????00??0???0?0??0??0????00????0??000????0
	???????????0???00????0????00??0???0?0??0??0????00????0??000????0
	???????????0???00????01???000?01?10?0??0??0????00??1?0??000????0
	???????????0???00????01???00??01?10?0??0??0????00??1?0??000????0
	???00????0?0???00????000??00??00000?0000??00???00??000??000????0
P3	??0?????????????????0???????
	????????????????????????????????0??????????????????????????????????
	??
	????????????????0?????????????????????1???????????????1???????
	???00????0?0???0???????0?????????????????0??????????0????000?????
S4	??a???????????????b???????
	??a???????????????b???????
	??a???????????????b???????
	??a???????????????b???????
	??a???????????????b???????

After we have corrected these results, we still needed a truncated differential path that would be compatible with the presented 3-round linear approximation [Riou, 2019]. Since it was difficult to find out which difference would work best with this linear approximation, we decided to try for all 64 positions by rotating Tezcan's difference 64 times. Then we applied this experiment for each of them. We could have had the best results experimentally due to the fact that Drygascon is rotation invariant. As a result, we discovered the initial difference $0x0000008000000000$ works best with the linear approximation provided by [Riou, 2019]. We present that in Table 4.7.

Table 4.7: 2-round probability one truncated differential with the combination of 3-round linear approximation that we used for building a 5-round differential-linear distinguisher. The input difference was changed into $0x0000008000000000$.

Round	**5- Round Differential-Linear Distinguisher of** $GASCON_{C5R11}$
I	0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000010000000000000000000000000000000000000000 0000000000000000000000010000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
S1	0000000000000000000000000?0000000000000000000000000000000000000000 0000000000000000000000000?0000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000?0000000000000000000000000000000000000000
P1	000000?000000000000000000?00000000?0000000000000000000000000000 0000000000?000000000000?0000000000000000000000000000?0000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000?000000000000?0000000000000000000000000000000000?0000
S2	000000?0000??000000000000?00000000?0000000000000000000000?0000?0000 000000?0000??000000000000?00000000?0000000000000000000000?0000?0000 00000000000??000000000000?0000000000000000000000000000?0000?0000 000000?0000??000000000000?00000000?0000000000000000000000?0000?0000 000000?0000??000000000000?00000000?0000000000000000000000?0000?0000
P2	?0000??0000??0000000?000???000000?00?0000?00000?0000???0000?0000 ?00000?0000??00000000?00???000??0?00?0000??000?00000?0?0000?0000 00000000000??0??0000000??00??000000000000000??000000000?0??0?00?0 0000?0?000???000??000000?0000?0???0000?0000000?0000??0?0000?0000 ?0000??0000??000000000000?0?0000?0?00000000??000??00000??0000?0000

Round	**State**				
0	1.....8.21.1.21	.....	.....	.....18.	1.....8.21.11a.
1	.....	.....	.....	.....8..1....1	.....8..1....1
2	.....	.....	.....	.....	1
3	.....	e37c4f1b6e8d53e6	e.8629e8e4b766af	.....	.....

At the end of the experiment, we got $2^{-7.96}$ bias by using 2^{29} data. That was far better than the theoretical bias 2^{-29} and $2^{61.28}$ data complexity. Then we wondered if we can find a better linear approximation, so we used lineartrails too [Dobraunig et al., 2015a] to search for it. We could not found a 3-round linear approximation with better than 2^{-15} bias. But when we were experimenting with this new approximation, surprisingly we got better practical results. The experiments showed that practical bias is around $2^{-5.35}$ for this new 5-round differential-linear distinguisher and around 2^{17} samples were enough to distinguish it from a random permutation, which is better than the previous results. The new 5-round distinguisher is given in Table 4.8.

Table 4.8: 2-round truncated probability one truncated differential with the combination of 3-round Type-I linear approximation with bias 2^{-15} that we found to build 5-round differential-linear distinguisher.

Round	**2- Round Truncated Differantial Distinguisher of** $GASCON_{C5R11}$
I	0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000100000000000000000000000 0000000000000000000000000000000000000001000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000
S1	000000000000000000000000000000000000000?000000000000000000000000 000000000000000000000000000000000000000?000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000?000000000000000000000000
P1	00000000000000000000000000000000?00000000?0000000000000000?00000 0000000000?0000000000000000000000000000?000000000000?0000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 00000?000000000000000000000000000000000?000000000000?000000000000
S2	00000?0000?000000000000000000000000?00000000?000000000000??0000?00000 00000?0000?000000000000000000000000?00000000?000000000000??0000?00000 00000?0000?000000000000000000000000000?000000000000??00000000000 00000?0000?000000000000000000000000?00000000?000000000000??0000?00000 00000?0000?000000000000000000000000?00000000?000000000000??0000?00000
P2	00000?0000???0000?00000?0000?00??00000???000?0000000??0000??0000 00000?0000?0?00000?000??0000?00????000???00?00000000??0000?00000 00?00?0??0?000000000??000000000000000??00??0000000??0??00000000000 ?0000?0000?0??0000?0000000?0000?0?0?0000?000000??000???000?0?000 00000?0000??00000??000??0000000???0000?0?000000000000??0000??0000

Round	**State**				
0	.....18.	1.....8.21.1.2.	1.....8.21.1.21	.....18.	.....
1	.....	.....	.....	.....8..1....1	.....8..1....1
2	.....	.....	.....	.....	1
3	.....	.....	e.8629e8e4b766af	c587ed1921757a4e	.....

We tried to extend this distinguisher for a practical 6-round experiment by increasing the truncated differential for one round. Because of our computational power, the experiment would have taken weeks to finish if we used more than 2^{40} data. So we have proceeded with using 2^{38} data to be optimal. But the results were showing randomness. Then we experimented for six rounds by rotating the initial difference 64 times with about 2^{32} data. But the data we used was not enough. Then we thought about finding a 4-round linear approximation and combining it with a 2-round truncated differential. But the best 4-round linear approximation we found had 2^{-60} bias. That can be seen in Table 4.9.

Table 4.9: 4-round Type-I linear approximation with bias 2^{-60}, hexadecimal notation.

Round	**State**				
0	c9.62.....431...	3261.1.186.84451	b261.1..a6.84641	8...8.4.....	.9128.22...31...
1	.....22.4.....	4..42.2.2.4.....	42.42.2.2.4.....	.....3.....	.....1.....
2	.....	.....	.....	.2..2...2.....	.2..2...2.....
3	.2..2...2.....	.....	.....	.....	.2.....
4	.....	.....	5fc96ecda38218a7	9c42eaf467161fb4	.....

So even if we could locate an initial difference that would match this approximation, theoretical data complexity was going to be $q^{-4} = 2^{60 \cdot 4} = 2^{240}$. And theoretical bias was going to be $2pq^2 = 2 \cdot 1 \cdot 2^{-60x2} = 2^{-119}$. Considering our previous observations, we expected that the practical total bias would not be less than 2^{-30}. And this would cost us to perform the experiment with at least 2^{60} data. Since we did not have such computational power, we decided to leave it there.

4.2.1 Comparison Between Ascon and Drygascon

So far we examined the differential-linear distinguishers of Ascon and Drygascon. But we still needed to compare the security of Ascon and Drygascon. Riou had already indicated that the theoretical linear approximation biases of Ascon and Drygascon were the same; both for Type-I and Type-II [Riou, 2019]. So we wanted to see if the practical biases were the same, also. But the given practical differential-linear analyses were not enough for a comparison. Because the 4-round differential-linear distinguisher of Ascon contained a 2-round truncated differential distinguisher with a 2-round Type-II linear approximation to turn it into an attack. But the 5-round

differential-linear distinguisher of Drygascon contained a 2-round truncated differential distinguisher with a 3-round Type-I linear approximation. So, for a fair comparison, we found the Type-II linear approximation of Drygascon (Table 4.10) and experimentally verified it.

Table 4.10: 2-round Type-II linear approximation with bias 2^{-8} in hexadecimal notation. We used that to build a 4-round differential-linear distinguisher.

Round	**State**				
0	.....	.....24.....	.....242.....	1.....2...	1.....2...
1	.....	.....	.....	.....2.....	.....2.....
2	6529b26f9284d935	.....	.....	.....	.....

Since 2^{17} data was enough for distinguishing 5 rounds of Drygascon, we just used 2^{17} data for 4 rounds of it to begin with. To be able to find a truncated distinguisher compatible with this linear approximation, we rotated the initial difference 64 times and performed the experiment for each of them. We observed the initial difference $0x0000000000800000$ gives us the best total bias of $2^{-1.67}$. And in [Tezcan, 2020], the best bias for 4-round Ascon was $2^{-1.68}$. The comparison of Ascon and Drygascon can be seen in Table 4.11.

Table 4.11: Comparison of Ascon128 and Drygascon-128

Algorithm	**Round**	**Type**	**Theoretical Bias**	**Data**	**Practical Bias**	**Data**
Ascon	4/12	Type-II	2^{-15}	2^{32}	$2^{-1.68}$	2^{8}
Drygascon	4/12	Type-II	2^{-15}	2^{32}	$2^{-1.67}$	2^{4}
Drygascon	5/12	Type-I	2^{-29}	$2^{61.28}$	$2^{-5.34}$	2^{17}

Even though these are very close results, we may say that Ascon might be more resistant against differential-linear cryptanalysis because Ascon has one more round than Drygascon and there are more data needed for distinguishing 4-rounds of it from a random permutation.

CHAPTER 5

CONCLUSIONS

Today's rapid development of technology is causing a gap between industry and academia. The fact that the industry has to respond quickly to emerging needs brings along new security vulnerabilities. The absence of regulation for encryption in IoT security, which is one of these areas, forces the industry to produce its own solutions. Failure to perform security analysis adequately while designing their cryptographic algorithms along with trying to keep their algorithms secret leads to their systems being broken. On the contrary, these ciphers need to be thoroughly studied by the cryptography community for ultimate security and performance. Because subsequent events not only negatively affect security but also cause reputational loss and high costs for them.

In 2013, NIST initiated a lightweight cryptography project to fill this gap. And in 2017, they announced that they started a competition for selecting one or more lightweight cryptography standard [McKay et al., 2016].

We carried out this work in order to contribute to the selection of a new and secure lightweight encryption standard for the constrained devices by analyzing some of the candidate algorithms. We showed the undisturbed bits of 16 of them and provided their probability one truncated differential distinguisher rounds that they have for future analysis. Then we focused on two very similar cipher suites Ascon and Drygascon. On our way to compare the security of the candidates Ascon and Drygascon, we realized that the 3-round truncated differential given to Drygascon by its designer was wrong [Riou, 2019]. Moreover, this misrepresentation and complexity of the specification of the algorithm had led to further erroneous analyses, which were a 5-round differential-linear distinguisher and a 3.5-round probability one truncated differential

distinguisher [Tezcan, 2020]. This situation shows how important it is to practically test the results that were obtained in theory. Experiments in practice will enable to check the accuracy of theoretical results. It will also show how effective the found distinguisher actually is because our research confirmed that there are significant distinctions between the theoretical data complexities of Ascon and Drygascon and their practical ones. The reasons were explained as the existence of multiple characteristics and slow diffusion in these ciphers [Bar-On et al., 2019].

In this study, we corrected 2, 3, and 3.5 rounds truncated differentials and 5-round differential-linear distinguisher given for Drygascon and presented them. We have also found a new 3-round linear approximation that will allow the 5-round differential-linear distinguisher to perform better in practice. This is the longest differential-linear distinguisher for Drygascon that we know of. That shows us the different distinguishers with the same theoretical biases can give different practical results.

We obtained these distinguishers not theoretically but practically. In theory, the masked input bits of the linear approximation should match the zero-difference of the truncated differential output bits. The reason is that since we cannot be sure how the unknown values at the end of the truncated differential will behave, we make such an assumption when calculating the theoretical bias. But that was not like that in our case. Because the unknown values at the end of the truncated differentials can give good results with linear masked bits at the beginning of the linear approximation in practice. This situation demonstrates the importance of testing theoretical results in practice.

In this work, we aimed to compare the security of Ascon and Drygascon in terms of differential-linear cryptanalysis. Since there were not such analyses that tested their security in practice under the same conditions, we reduced them into the same states. We showed their practical distinguishers are very close. But we may say that Ascon might be more resistant against differential-linear cryptanalysis. The reason is that Ascon has one more round than Drygascon.

REFERENCES

[Aagaard et al., 2019a] Aagaard, M., AlTawy, R., Gong, G., Mandal, K., and Rohit, R. (2019a). Ace: An authenticated encryption and hash algorithm. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Aagaard et al., 2019b] Aagaard, M., AlTawy, R., Gong, G., Mandal, K., Rohit, R., and Zidaric, N. (2019b). Wage: An authenticated cipher. *Submission to NIST Lightweight Cryptography Standardization Project (announced as round 2 candidate on August 30, 2019)*.

[Adomnicai et al., 2019] Adomnicai, A., Berger, T. P., Clavier, C., Francq, J., Huynh, P., Lallemand, V., Le Gouguec, K., Minier, M., Reynaud, L., and Thomas, G. (2019). Lilliput-ae: A new lightweight tweakable block cipher for authenticated encryption with associated data. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[AlTawy et al., 2019a] AlTawy, R., Gong, G., He, M., Jha, A., Mandal, K., Nandi, M., and Rohit, R. (2019a). Spoc. *Submission to NIST LwC Standardization Process (Round 2)*.

[AlTawy et al., 2019b] AlTawy, R., Gong, G., He, M., Mandal, K., and Rohit, R. (2019b). Spix: An authenticated cipher submission to the nist lwc competition. *Submitted to NIST Lightweight Standardization Process*.

[Andreeva et al., 2019] Andreeva, E., Lallemand, V., Purnal, A., Reyhanitabar, R., Roy, A., and Vizár, D. (2019). Forkae. *Submission to NIST Lightweight Cryptography Project*.

[Avanzi et al., 2019] Avanzi, R., Banik, S., Bogdanov, A., Dunkelman, O., Huang, S., and Regazzoni, F. (2019). Qameleon v. 1.0. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Avik Chakraborti, 2019] Avik Chakraborti, Nilanjan Datta, A. J. C. M. L.

M. N.-Y. S. (2019). Lotus-aead and locus-aead. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/lotus-aead-and-locus-aead-spec.pdf. Accessed: 2021-05-10.

[Banik et al., 2019a] Banik, S., Bogdanov, A., Peyrin, T., Sasaki, Y., Sim, S. M., Tischhauser, E., and Todo, Y. (2019a). Sundae-gift. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Banik et al., 2019b] Banik, S., Chakraborti, A., Iwata, T., Minematsu, K., Nandi, M., Peyrin, T., Sasaki, Y., Sim, S. M., and Todo, Y. (2019b). Gift-cofb. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Bao et al., 2019] Bao, Z., Chakraborti, A., Datta, N., Guo, J., Nandi, M., Peyrin, T., and Yasuda, K. (2019). Photon-beetle authenticated encryption and hash family. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Bar-On et al., 2019] Bar-On, A., Dunkelman, O., Keller, N., and Weizman, A. (2019). Dlct: A new tool for differential-linear cryptanalysis. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 313–342. Springer.

[Barker and Mouha, 2017] Barker, E. and Mouha, N. (2017). Recommendation for the triple data encryption algorithm (tdea) block cipher. Technical report, National Institute of Standards and Technology.

[Barker and Roginsky, 2011] Barker, E. and Roginsky, A. (2011). Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. *NIST Special Publication*, 800:131A.

[Beierle et al., 2019] Beierle, C., Biryukov, A., dos Santos, L. C., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q., and Biryukov, A. (2019). Schwaemm and esch: Lightweight authenticated encryption and hashing using the sparkle permutation family. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Beierle et al., 2020] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., and Sim, S. M. (2020). Skinny-aead and skinny-hash. *IACR Transactions on Symmetric Cryptology*, pages 88–131.

[Bellizia et al., 2020] Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., et al. (2020). Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. *IACR Transactions on Symmetric Cryptology*, pages 295–349.

[Bernstein, 2013] Bernstein, D. (2013). Caesar: Competition for authenticated encryption: Security, applicability, and robustness. https://competitions.cr.yp.to/caesar.html. Accessed: 2021-05-10.

[Bernstein et al., 2017] Bernstein, D. J., Kölbl, S., Lucks, S., Massolino, P. M. C., Mendel, F., Nawaz, K., Schneider, T., Schwabe, P., Standaert, F.-X., Todo, Y., et al. (2017). Gimli: A cross-platform permutation. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 299–320. Springer.

[Bertoni et al., 2007] Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. (2007). Sponge functions. In *ECRYPT hash workshop*, volume 2007. Citeseer.

[Bertoni et al., 2011] Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. (2011). Duplexing the sponge: Single-pass authenticated encryption and other applications. In *International Workshop on Selected Areas in Cryptography*, pages 320–337. Springer.

[Bertoni et al., 2012] Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. (2012). Permutation-based encryption, authentication and authenticated encryption. *Directions in Authenticated Ciphers*, pages 159–170.

[Bhattacharjee et al., 2021] Bhattacharjee, A., López, C. M., List, E., and Nandi, M. (2021). The oribatida v1. 3 family of lightweight authenticated encryption schemes. *Journal of Mathematical Cryptology*, 15(1):305–344.

[Biham et al., 2002] Biham, E., Dunkelman, O., and Keller, N. (2002). Enhancing differential-linear cryptanalysis. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 254–266. Springer.

[Biham and Shamir, 1991] Biham, E. and Shamir, A. (1991). Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72.

[Biryukov and Perrin, 2018] Biryukov, A. and Perrin, L. (2018). Lightweight block ciphers. https://www.cryptolux.org/index.php/Lightweight_Block_Ciphers/. Accessed: 2021-05-10.

[Blondeau et al., 2011] Blondeau, C., Gérard, B., and Tillich, J.-P. (2011). Accurate estimates of the data complexity and success probability for various cryptanalyses. *Designs, codes and cryptography*, 59(1):3–34.

[Bogdanov et al., 2007] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., and Vikkelsoe, C. (2007). Present: An ultra-lightweight block cipher. In *International workshop on cryptographic hardware and embedded systems*, pages 450–466. Springer.

[Browning et al., 2010] Browning, K., Dillon, J., McQuistan, M., and Wolfe, A. (2010). An apn permutation in dimension six. *Finite Fields: theory and applications*, 518:33–42.

[Chakraborti et al., 2020] Chakraborti, A., Datta, N., Jha, A., Mancillas-López, C., Nandi, M., and Sasaki, Y. (2020). Estate: A lightweight and low energy authenticated encryption mode. *IACR Transactions on Symmetric Cryptology*, pages 350–389.

[Chakraborti et al., 2019] Chakraborti, A., Datta, N., Jha, A., and Nandi, M. (2019). Hyena. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Chakraborty and Nandi, 2019a] Chakraborty, B. and Nandi, M. (2019a). mixfeed. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Chakraborty and Nandi, 2019b] Chakraborty, B. and Nandi, M. (2019b). Security proof of orange-zest. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Daemen et al., 2020a] Daemen, J., Hoffert, S., Peeters, M., Assche, G. V., and Keer, R. V. (2020a). Xoodyak, a lightweight cryptographic scheme. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Daemen et al., 2020b] Daemen, J., Massolino, P. M. C., Mehrdad, A., and Rotella, Y. (2020b). The subterranean 2.0 cipher suite. *IACR Transactions on Symmetric Cryptology*, pages 262–294.

[Daemen and Rijmen, 1999] Daemen, J. and Rijmen, V. (1999). The rijndael block cipher: Aes proposal. In *First candidate conference (AeS1)*, pages 343–348.

[Daemen and Rijmen, 2002] Daemen, J. and Rijmen, V. (2002). *The design of Rijndael*, volume 2. Springer.

[De Canniere, 2006] De Canniere, C. (2006). Trivium: A stream cipher construction inspired by block cipher design principles. In *International Conference on Information Security*, pages 171–186. Springer.

[do Nascimento and Xexéo, 2019] do Nascimento, E. M. and Xexéo, J. A. M. (2019). Flexaead-a lightweight cipher with integrated authentication. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Dobraunig et al., 2020] Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., and Unterluggauer, T. (2020). Isap v2. 0. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Dobraunig et al., 2015a] Dobraunig, C., Eichlseder, M., and Mendel, F. (2015a). Heuristic tool for linear cryptanalysis with applications to caesar candidates. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 490–509. Springer.

[Dobraunig et al., 2015b] Dobraunig, C., Eichlseder, M., Mendel, F., and Schläffer, M. (2015b). Cryptanalysis of ascon. In *Cryptographers' Track at the RSA Conference*, pages 371–387. Springer.

[Dobraunig et al., 2016] Dobraunig, C., Eichlseder, M., Mendel, F., and Schläffer, M. (2016). Ascon v1. 2. *Submission to the CAESAR Competition*.

[Dobraunig and Mennink, 2019] Dobraunig, C. and Mennink, B. (2019). Elephant v1. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Driscoll, 2018] Driscoll, K. (2018). Lightweight crypto for lightweight unmanned

arial systems. In *2018 Integrated Communications, Navigation, Surveillance Conference (ICNS)*, pages 1–15. IEEE.

[Dworkin, 2015] Dworkin, M. J. (2015). Sha-3 standard: Permutation-based hash and extendable-output functions. *Standard NIST FIPS-202*.

[ECRYPT, 2019] ECRYPT (2019). ebacs: Ecrypt benchmarking of cryptographic systems. http://bench.cr.yp.to/. Accessed: 2021-05-02.

[Evertse, 1987] Evertse, J.-H. (1987). Linear structures in blockciphers. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 249–266. Springer.

[Fabio Campos and Viguier, 2020] Fabio Campos, Lars Jellema, M. L. L. M. D. S. and Viguier, B. (2020). Lightweight crypto on risc-v. https://github.com/AsmOptC-RiscV/Assembly-Optimized-C-RiscV. Accessed: 2021-05-02.

[Garcia et al., 2008] Garcia, F. D., de Koning Gans, G., Muijrers, R., Van Rossum, P., Verdult, R., Schreur, R. W., and Jacobs, B. (2008). Dismantling mifare classic. In *European symposium on research in computer security*, pages 97–114. Springer.

[Goudarzi et al., 2020] Goudarzi, D., Jean, J., Kölbl, S., Peyrin, T., Rivain, M., Sasaki, Y., and Sim, S. M. (2020). Pyjamask: Block cipher and authenticated encryption with highly efficient masked implementation. *IACR Transactions on Symmetric Cryptology*, pages 31–59.

[Gueron et al., 2019] Gueron, S., Jha, A., and Nandi, M. (2019). Comet: Counter mode encryption with authentication tag. *Second Round Candidate of the NIST LWC Competition*.

[Gueron and Lindell, 2019] Gueron, S. and Lindell, Y. (2019). Simple: A simple aead scheme, a submission to the nist lightweight cryptography standardization process. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Guo and Iwata, 2019] Guo, J. and Iwata, T. (2019). Siv-rijndael 256 authenticated encryption and hash family. *Submission to NIST Lightweight Cryptography Project*.

[Guo et al., 2011] Guo, J., Peyrin, T., and Poschmann, A. (2011). The photon family of lightweight hash functions. In *Annual Cryptology Conference*, pages 222–239. Springer.

[Han Sui, 2019] Han Sui, Wenling Wu, L. Z. D. Z. (2019). Laem (lightweight authentication encryption mode). https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/LAEM-spec.pdf. Accessed: 2021-05-10.

[Hashing, 2019] Hashing, C. (2019). Sneiken and sneikha. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Hell et al., 2019] Hell, M., Johansson, T., Meier, W., Sönnerup, J., and Yoshida, H. (2019). Grain-128aead-a lightweight aead stream cipher. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Isobe et al., 2019] Isobe, S. T., Meier, W., and Zhang, B. (2019). Triad v1. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Iwata et al., 2020] Iwata, T., Khairallah, M., Minematsu, K., and Peyrin, T. (2020). Duel of the titans: The romulus and remus families of lightweight aead algorithms. *IACR Transactions on Symmetric Cryptology*, pages 43–120.

[Iwata et al., 2019] Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T., Sasaki, Y., Sim, S. M., and Sun, L. (2019). Thank goodness it's friday (tgif). *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Jean, 2016] Jean, J. (2016). TikZ for Cryptographers. https://www.iacr.org/authors/tikz/. Accessed: 2021-05-10.

[Jirotka, 2016] Jirotka, Z. (2016). *Saturnin*. Charles University in Prague, Karolinum Press.

[Kamyar Mohajerani and Gaj, 2020] Kamyar Mohajerani, Richard Haeussler, R. N. F. F. A. A. J.-P. K. and Gaj, K. (2020). Athena: Fpga benchmarking. https://cryptography.gmu.edu/athena/index.php?id=LWC. Accessed: 2021-05-02.

[Knudsen, 1994] Knudsen, L. R. (1994). Truncated and higher order differentials. In *International Workshop on Fast Software Encryption*, pages 196–211. Springer.

[Langford and Hellman, 1994] Langford, S. K. and Hellman, M. E. (1994). Differential-linear cryptanalysis. In *Annual International Cryptology Conference*, pages 17–25. Springer.

[Leander et al., 2007] Leander, G., Paar, C., Poschmann, A., and Schramm, K. (2007). New lightweight des variants. In *International Workshop on Fast Software Encryption*, pages 196–210. Springer.

[Li et al., 2017] Li, Z., Dong, X., and Wang, X. (2017). Conditional cube attack on round-reduced ascon. *IACR Transactions on Symmetric Cryptology*, pages 175–202.

[Liu et al., 2019] Liu, D., Nepal, S., Pieprzyk, J., and Susilo, W. (2019). Clae: a lightweight aead scheme of resisting side channel attacks. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Makarim and Tezcan, 2014] Makarim, R. H. and Tezcan, C. (2014). Relating undisturbed bits to other properties of substitution boxes. In *International Workshop on Lightweight Cryptography for Security and Privacy*, pages 109–125. Springer.

[Matsui, 1993] Matsui, M. (1993). Linear cryptanalysis method for des cipher. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 386–397. Springer.

[McKay et al., 2016] McKay, K., Bassham, L., Sönmez Turan, M., and Mouha, N. (2016). Report on lightweight cryptography. Technical report, National Institute of Standards and Technology.

[Mehner, 2019] Mehner, C. E. (2019). Limdolen. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Meijer and Verdult, 2015] Meijer, C. and Verdult, R. (2015). Ciphertext-only cryptanalysis on hardened mifare classic cards. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 18–30.

[Miguel Montes, 2019] Miguel Montes, D. P. (2019). Yarara and coral v1. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/yarara_and_coral-spec.pdf. Accessed: 2021-05-10.

[Mustafa Khairallah and Chattopadhyay, 2020] Mustafa Khairallah, T. P. and Chattopadhyay, A. (2020). Asic benchmarking. https://github.com/mustafam001/lwc-aead-rtl. Accessed: 2021-05-02.

[Naito et al., 2019] Naito, Y., Matsui, M., Sakai, Y., Suzuki, D., Sakiyama, K., and Sugawara, T. (2019). Saeaes. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Nechvatal et al., 2001] Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., and Roback, E. (2001). Report on the development of the advanced encryption standard (aes). *Journal of Research of the National Institute of Standards and Technology*, 106(3):511.

[Nilanjan et al., 2019] Nilanjan, D., Ashrujit, G., Debdeep, M., Sikhar, P., Stjepan, P., and Rajat, S. (2019). Trifle. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[NIST, 2018] NIST (2018). Submission requirements and evaluation criteria for the lightweight cryptography standardization process. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf/. Accessed: 2021-05-10.

[NIST, 2021] NIST (2021). Benchmarking of lightweight cryptographic algorithms on microcontrollers. https://github.com/usnistgov/Lightweight-Cryptography-Benchmarking. Accessed: 2021-05-02.

[Nohl et al., 2008] Nohl, K., Evans, D., Starbug, S., and Plötz, H. (2008). Reverse-engineering a cryptographic rfid tag. In *USENIX security symposium*, volume 28.

[Nyberg, 1993] Nyberg, K. (1993). Differentially uniform mappings for cryptography. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 55–64. Springer.

[Otte, 2019] Otte, D. (2019). Gage and ingage v1. 03. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Paar et al., 2009] Paar, C., Eisenbarth, T., Kasper, M., Kasper, T., and Moradi, A. (2009). Keeloq and side-channel analysis-evolution of an attack. In *2009 Work-*

shop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pages 65–69. IEEE.

[Penazzi and Montes, 2019] Penazzi, D. and Montes, M. (2019). Shamash (and shamashash)(version 1). *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Riou, 2019] Riou, S. (2019). Drygascon. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Sarkar et al., 2019] Sarkar, S., Mandal, K., and Saha, D. (2019). Sycon v1. 0 submission to lightweight cryptographic standards. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Sebastian Renner and Mottok, 2020] Sebastian Renner, E. P. and Mottok, J. (2020). Avr/arm/risc-v microcontroller benchmarking. https://lwc.las3.de/. Accessed: 2021-05-02.

[Shannon, 1949] Shannon, C. E. (1949). Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715.

[Tezcan, 2014] Tezcan, C. (2014). Improbable differential attacks on present using undisturbed bits. *Journal of Computational and applied mathematics*, 259:503–511.

[Tezcan, 2016] Tezcan, C. (2016). Truncated, impossible, and improbable differential analysis of ascon. *IACR Cryptol. ePrint Arch.*, 2016:490.

[Tezcan, 2017] Tezcan, C. (2017). Brute force cryptanalysis of mifare classic cards on gpu. In *International Conference on Information Systems Security and Privacy*, volume 2, pages 524–528. SCITEPRESS.

[Tezcan, 2020] Tezcan, C. (2020). Analysis of ascon, drygascon, and shamash permutations. *International Journal of Information Security Science*, 9(3):172–187.

[Tezcan et al., 2014] Tezcan, C., Taşkın, H. K., and Demircioğlu, M. (2014). Improbable differential attacks on serpent using undisturbed bits. In *Proceedings of the 7th International Conference on Security of Information and Networks*, pages 145–150.

[Todo, 2015] Todo, Y. (2015). Structural evaluation by generalized integral property. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 287–314. Springer.

[Turan et al., 2019] Turan, M. S., McKay, K. A., Çalik, Ç., Chang, D., Bassham, L., et al. (2019). Status report on the first round of the nist lightweight cryptography standardization process. *National Institute of Standards and Technology, Gaithersburg, MD, NIST Interagency/Internal Rep.(NISTIR)*.

[Watanabe et al., 2008] Watanabe, D., Ideguchi, K., Kitahara, J., Muto, K., Furuichi, H., and Kaneko, T. (2008). Enocoro-80: A hardware oriented stream cipher. In *2008 Third International Conference on Availability, Reliability and Security*, pages 1294–1300. IEEE.

[Weatherley, 2021] Weatherley, R. (2021). Avr/arm microcontroller benchmarking. https://rweather.github.io/lightweight-crypto/. Accessed: 2021-05-02.

[Wouters et al., 2019] Wouters, L., Marin, E., Ashur, T., Gierlichs, B., and Preneel, B. (2019). Fast, furious and insecure: Passive keyless entry and start systems in modern supercars. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 66–85.

[Wu and Huang, 2019a] Wu, H. and Huang, T. (2019a). Clx: A family of lightweight authenticated encryption algorithms. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/CLX-spec.pdf. Accessed: 2021-05-10.

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

[Ye et al., 2019] Ye, D., Shi, D., Ma, Y., and Wang, P. (2019). Hern & heron: Lightweight aead and hash constructions based on thin sponge (v1). *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Z'aba et al., 2019] Z'aba, M. R., Jamil, N., Rohmad, M. S., Rani, H. A., and Shamsuddin, S. (2019). The cilipadi family of lightweight authenticated encryption. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Zhang, 2019a] Zhang, B. (2019a). Fountain: A lightweight authenticated cipher (v1). *A Submission to the NIST Lightweight Cryptography Standardization Process*, 1.

[Zhang, 2019b] Zhang, B. (2019b). Quartet: A lightweight authenticated cipher. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/Quartet-spec.pdf. Accessed: 2021-05-10.

[Zhang et al., 2019] Zhang, W., Ding, T., Yang, B., Bao, Z., Xiang, Z., Ji, F., and Zhao, X. (2019). Knot: Algorithm specifications and supporting document. *A Submission to the NIST Lightweight Cryptography Standardization Process*.

[Zhenzhen Bao, 2019] Zhenzhen Bao, Jian Guo, T. I. L. S. (2019). Sivtem-photon authenticated encryption and hash family. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/SIV-TEM-PHOTON-Spec.pdf. Accessed: 2021-05-10.