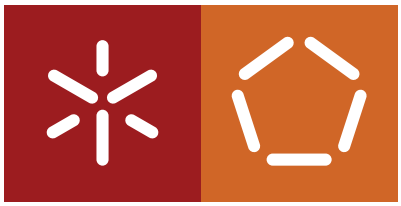
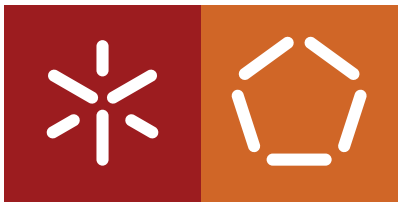


**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

João Pedro Dias Fernandes

# Security Analysis of NIST-LWC Contest Finalists



**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

João Pedro Dias Fernandes

# Security Analysis of NIST-LWC Contest Finalists

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
**José Manuel Valença**

# ACKNOWLEDGMENTS

These past years of societal trauma have taken their toll on everyone, and, despite that, I still had family and friends to guide and support me throughout my academic journey. To them I want to express my deepest gratitude, for without them this would not be possible.

To my supervisor, José Manuel Valença, I want to extend my most sincere gratitude for his guidance, help, mentorship, and unending patience, all along this process, for without him I would be lost.

Lastly, I want to thank a friend that shall remain nameless for their help with the images I used in this document.

53++!305))6*;4826)4+.)4+);806*;48!8‘60))85;]8*:+*8!83(88)
5*!;46(;88*96*?;8)*+(;485);5*!2:*+(;4956*2(5*-4)8‘8*;40692
85);)6!8)4++;1(+9;48081;8:8+1;48!85;4)485!528806*81(+9;48;
(88;4(+?34;48)4+;161;:188;+?;

## STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

## ABSTRACT

Traditional cryptographic standards are designed with a desktop and server environment in mind, so, with the relatively recent proliferation of small, resource constrained devices in the Internet of Things, sensor networks, embedded systems, and more, there has been a call for lightweight cryptographic standards with security, performance and resource requirements tailored for the highly-constrained environments these devices find themselves in.

In 2015 the National Institute of Standards and Technology began a Standardization Process in order to select one or more Lightweight Cryptographic algorithms. Out of the original 57 submissions ten finalists remain, with ASCON and Romulus being among the most scrutinized out of them.

In this dissertation I will introduce some concepts required for easy understanding of the body of work, do an up-to-date revision on the current situation on the standardization process from a security and performance standpoint, a description of ASCON and Romulus, and new best known analysis, and a comparison of the two, with their advantages, drawbacks, and unique traits.



KEYWORDS Lightweight Cryptography, NIST, ASCON, Romulus, Cryptanalysis.

## RESUMO

Os padrões criptográficos tradicionais foram elaborados com um ambiente de computador e servidor em mente. Com a proliferação de dispositivos de pequenas dimensões tanto na *Internet of Things*, redes de sensores e sistemas embutidos, apareceu uma necessidade para se definir padrões para algoritmos de criptografia leve, com prioridades de segurança, *performance* e gasto de recursos equilibrados para os ambientes altamente limitados em que estes dispositivos operam.

Em 2015 o *National Institute of Standards and Technology* lançou um processo de estandardização com o objectivo de escolher um ou mais algoritmos de criptografia leve. Das cinquenta e sete candidaturas originais sobram apenas dez finalistas, sendo *ASCON* e Romulus dois desses finalistas mais examinados.

Nesta dissertação irei introduzir alguns conceitos necessários para uma fácil compreensão do corpo deste trabalho, assim como uma revisão atualizada da situação atual do processo de estandardização de um ponto de vista tanto de segurança como de *performance*, uma descrição do ASCON e do Romulus assim como as suas melhores análises recentes e uma comparação entre os dois, frisando as suas vantagens, desvantagens e aspectos únicos.



PALAVRAS-CHAVE Criptografia Leve, NIST, ASCON, Romulus, Criptoanálise

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACRONYMS

AD - Associated Data

AE - Authenticated Encryption

AEAD - Authenticated Encryption with Associated Data

AES - Advanced Encryption Standard

ASIC - Application-Specific Integrated Circuits

CCA - Non-adaptive Chosen-ciphertext Attack

CCA2 - Adaptive Chosen-ciphertext Attack

CCAm - Nonce-misuse resilient Chosen-ciphertext Attack

CIM - Nonce-misuse resistant Ciphertext Integrity

CNF - Conjunctive Normal Form

CP - Constraint Programming

CPA - Chosen-plaintext Attack

DDT - Difference Distribution Table

DL - Differential-Linear

EtM - Encrypt-then-MAC

E&M - Encrypt-and-MAC

FPGA - Field-Programmable Gate Array

GCM - Galois Counter Mode

HDL - Higher-Order Differential-Linear

IOT - Internet of Things

IV - Initialization Vector

LAT - Linear Approximation Table

LUT - Look Up Table

LWC - Lightweight Cryptography

MAC - Message Authentication Code

MD - Merkle-Damgård

MILP - Mixed Integer Linear Programming

MQ - Multivariate Quadratic

MRAE - Nonce Misuse-resistance Authenticated Encryption

MtE - MAC-then-Encrypt

NIST - National Institute of Standards and Technology

NISTIR - NIST Internal Report

NIST-LWC - National Institute of Standards and Technology Lightweight Cryptography Standardization Process

PRNG - Pseudo-random Number Generator

RFID - Radio Frequency Identification

RUP - Release Unverified Plaintext

SHA - Secure Hash Algorithm

SMT - Satisfiability Modulo Theories

SPN - Substitution Permutation Network

TBC - Tweakable Block Cipher

XOF - Extendable Output Function

XOR - Exclusive-or

Part I

# INTRODUCTORY MATERIAL

# 1

# INTRODUCTION

## 1.1 CONTEXT

Traditional cryptographic standards are designed with a desktop and server environment in mind, therefore the security, performance and resource requirements trade-offs when setting those standards are tailored for that environment. The proliferation of resource constrained devices used in distributed control systems, sensor networks, embedded systems and more made clear that new cryptographic standards with increased performance and decreased resource requirements were needed.

In August 2018, the National Institute of Standards and Technology started the Lightweight Cryptography Standardization Process NIST to solicit, evaluate and standardize one or more authenticated encryption with associated data schemes and hashing schemes suitable for use in constrained environments. In March 2021, NIST announced the ten finalists of the standardization process and shortly after published Turan et al. (2021) describing their evaluation criteria, selection of the finalists, discussion for each candidate, and software and hardware benchmarking initiatives. This process is nearing completion with standardization scheduled for 2023.

## 1.2 OBJECTIVES

My first objective is to provide to anyone with an entry-level knowledge of cryptography an easy to understand state-of-the-art that covers what they would need to know in order to follow along, ranging from the basics to more advanced subjects like specific cryptanalysis techniques.

My second objective is to cover two finalists of the NIST-LWC, ASCON Dobraunig et al. (2021b) and Romulus Guo et al. (2021). This coverage consists of explaining how they work and what they use, either as mode of operation and underlying primitives, explaining the security claims laid out by the designers, and going over published and unpublished analysis made since the closing of the second round of the NIST-LWC.

My third and main objective is to give a personal opinion on ASCON, Romulus and their comparison, based on what I go over beforehand and the weighted criteria from NIST.

## 1.3 STRUCTURE OF THE DISSERTATION

The current chapter, Chapter 1, exposes the overarching theme of the dissertation and its objectives.

An extensive state-of-the-art covering symmetric cryptography, security and cryptanalysis, and the NIST-LWC is exposed in Chapter 2.

A brief explanation on the specification, security claims and best known cryptanalysis of ASCON and Romulus can be found in Chapters 3 and 4 respectively. My conclusions on ASCON and Romulus and their comparison can be found in Chapter 5.

My overall conclusions and prospects for further work are laid out in Chapter 6.

# 2

# STATE OF THE ART

## 2.1 ELEMENTARY CONCEPTS

### 2.1.1 *Symmetric Cryptography*

"Cryptology is about communication in the presence of adversaries." Rivest

"Cryptography is the science of keeping secrets secret." Delfs et al.

These two quotes that a student might find when beginning their journey into the world of cryptography point to the key priority when building a cryptographic scheme, *confidentiality*. In order to ensure confidentiality of information many ciphers were devised throughout history and, up until the information age with the invention of *public-key cryptography* Diffie and Hellman (2022), all of them were symmetric ciphers. In *symmetric cryptography* the key used to encrypt and decrypt the messages sent between two parties engaging in communication is either the same or very closely related.

A classic example of symmetric cryptography is the Caesar Cipher where every letter in the message is replaced by a letter that is a fixed number of positions from the original, if it were three positions, the letter A would become D, B would become E and so on. Now lets say A and B (or Aleister and Beatrice) want to communicate in secret, the only thing they would need to agree upon would be the number of positions which they would of course keep secret.

In this example we have the original message, also known as *plaintext* ($P$), the message obtained with the cipher, also known as *ciphertext* ($C$) and the number of positions, a *key* ($K$). We also have a *encryption function* ($E$) which is the cipher as described and a *decryption function* ($D$) which is the reverse of the aforementioned cipher.

### 2.1.2 *Hashes*

A *cryptographic hash function* is an algorithm that maps a binary string with a variable length (known as *message* or $M$) to a fixed length output binary string (know as *digest* or $D$) with the caveat that this mapping must be *collision-resistant* Katz and Lindell (2020).

#### *Security*

The pigeon-hole principle dictates that collisions must exist when mapping an input to a smaller output so a collision-resistant hash function is not one that lacks collisions but one where a collision is hard to find in a probabilistic polynomial time Katz and Lindell.

*Collision resistance* is the strongest notion of security when it comes to hash functions but there are two weaker ones that might be sufficient requirements when designing a cryptographic hash function:

- *Second preimage resistance*;
- *Preimage resistance*.

A hash function is second preimage resistant if for a given message $x$ it is hard to find $x'$ such that $H(x) = H(x')$ in a probabilistic polynomial time, or simply speaking, for a given message it is hard to find a different message such that the digests of them are the same Katz and Lindell.

A hash function is preimage resistant if for a given digest $y$ it is hard to find $x'$ such that $H(x') = y$ in a probabilistic polynomial time, or simply speaking, for a given digest it is hard to find a message that produced it Katz and Lindell.

Any hash function that is collision resistant is also second preimage resistant and any hash function that is second preimage resistant is preimage resistant as well Katz and Lindell.

*Building a Hash Function*

The most known method of building a cryptographic hash function is the *Merkle-Damgård construction* Merkle (1989) Damgård (1989).

In the MD construction we divide our message in blocks using an appropriate padding scheme and feed said blocks sequentially to a collision-resistant one-way *compression function*. The *initialization vector* and the first message block are fed into the first use of the compression function and its output and the next message block are fed into the next use of the compression function and so on. After digesting all the message blocks a *finalization function* is applied that among other benefits it allows the hash function to have an output length different from the compression function.

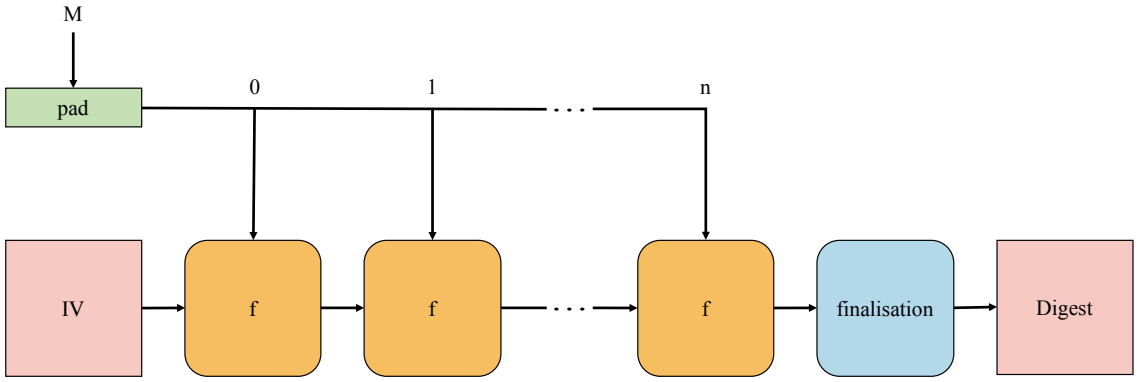



Figure 1: Merkle–Damgård hash construction

This approach allows us to reduce the problem of finding a hash function to finding a proper compression function Menezes et al. (2018), a design principle that is carried on to other constructions, like the sponge.

In order to save code space in resource-constrained devices we can use a block cipher to build a collision resistant compression function using methods like the Davies-Meyer or the Miyaguchi-Preneel Menezes et al..

### 2.1.3 *Authenticated Encryption with Associated Data*

Going back to the previous example with Aleister and Beatrice, lets insert a Malicious (or M) third party that wants to intercept and change the message. The cipher provides confidentiality, yes, but Malicious could change anything in the ciphertext, they could even change it to a completely valid ciphertext if they had information on the key, and there would be no way for Aleister or Beatrice to know if the message was tampered with.

*Authenticated encryption* (or AE) is a form of symmetric encryption that provides a way for the receiver to verify the integrity and authenticity of the message, and in the case of *authenticated encryption with associated data* (or AEAD), the *associated data* (or AD) as well. This authentication is done via *message authentication codes* (or MAC) which are obtained with *keyed hashing functions* ($H$) that take the message and the key and generate a code that can only be verified by someone holding the same key.

There are three forms of AE, *Encrypt-then-MAC*, *MAC-then-Encrypt* and *Encrypt-and-MAC*.

#### *Encrypt-then-MAC*

In Encrypt-then-MAC (or EtM) the plaintext is encrypted, then a tag (the MAC) is generated using the ciphertext and the AD, and the final message consisting of the AD, ciphertext and the tag is ready to be sent. The receiver can verify the integrity of the message by comparing the tag they received to the tag produced by the ciphertext they received and their own key.

EtM is considered the safest form of AEAD due to the fact that it does not require the receiver to decrypt the ciphertext in order to verify it's authenticity Bellare and Namprempre (2000). Furthermore, a different key must be used for encryption and authentication Menezes et al., example 9.88.

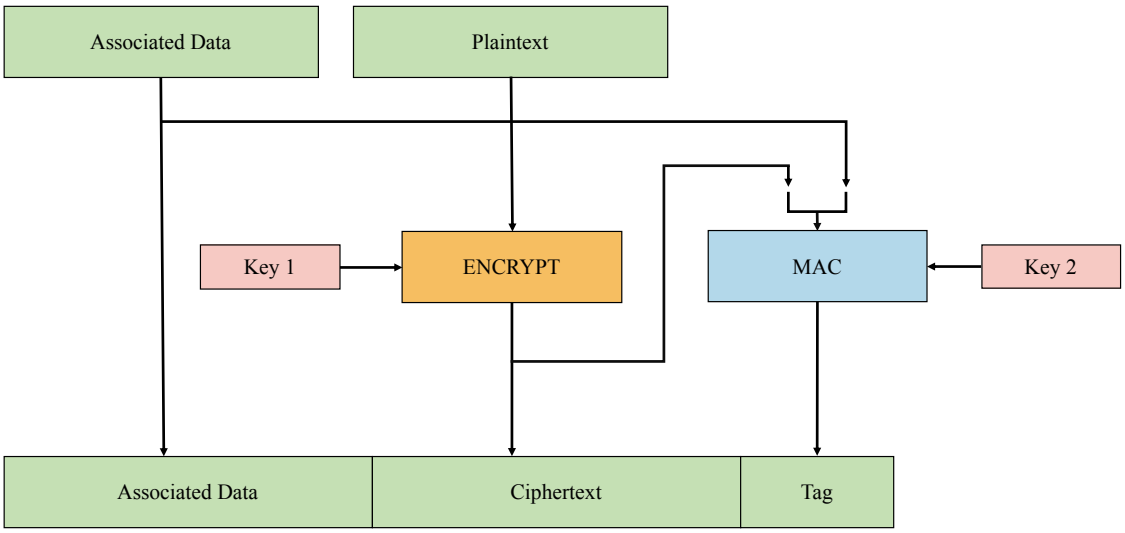



Figure 2: Authenticated encryption with associated data - Encrypt-then-MAC

*MAC-then-Encrypt*

In MAC-then-Encrypt, or MtE, the tag is generated by feeding the plaintext and the AD to the keyed hashing function and then both the plaintext and the tag are encrypted together resulting in a single ciphertext. The receiver can verify the integrity of the message they received by first decrypting the ciphertext and then comparing the tag they received to the tag produced by the plaintext, the AD and their own key.

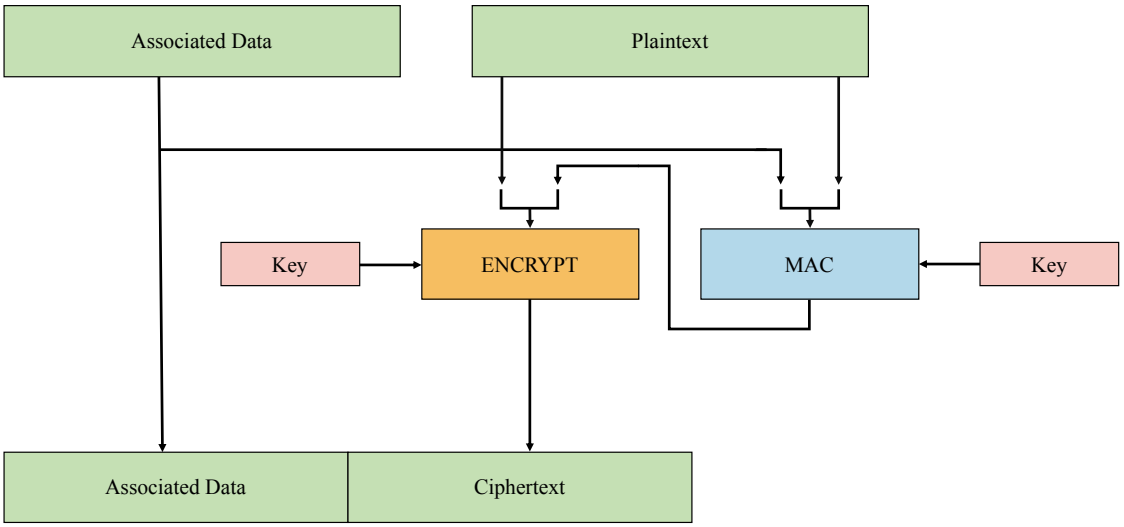



Figure 3: Authenticated encryption with associated data - MAC-then-Encrypt

*Encrypt-and-MAC*

In Encrypt-and-MAC, or E&M, the tag is generated by feeding the plaintext and the AD to the keyed hashing function and encryption function simultaneously with the final message being the AD, ciphertext and the tag. The receiver can verify the integrity of the message they received by first decrypting the ciphertext and then comparing the tag they received to the tag produced by the plaintext, the AD and their own key.

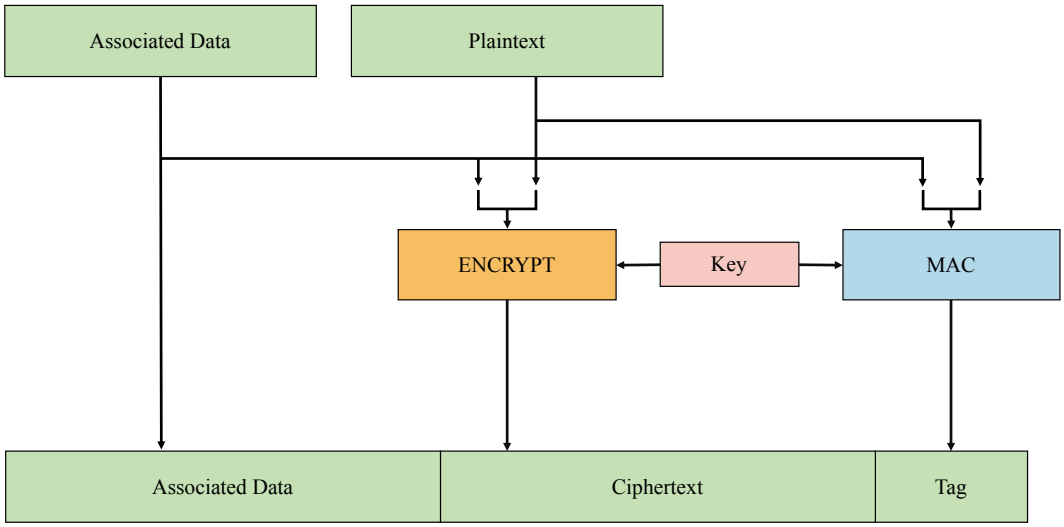



Figure 4: Authenticated encryption with associated data - Encrypt-and-MAC

### 2.1.4 *Modes of Operation and Primitives*

When building an AEAD or hashing mode of operation for a cipher suite one must decide on what the mode will be based on and what the underlying primitive in use (if any) will be. From tweakable block cipher-based modes of operation with an underlying tweakable block cipher primitive to duplex-based constructions with a substitution-permutation network primitive, there are many different choices in mode of operation and respective underlying primitive.

#### *Block Ciphers*

A *block cipher* is a symmetric key cipher that operates on fixed-length groups of bits called blocks. A block cipher consists on two paired algorithms, $E$ for encryption and $D$ for decryption. Both algorithms take two inputs: a block $P$ of size $n$ bits and a key $K$ of size $k$ bits and yield a $n$-bit sized block. The decryption function $D$ is the inverse function of $E$.

A block cipher's encryption function can be defined as Menezes et al.:

$E_K(P) = E(K,P) : \{0,1\}^k \times \{0,1\}^n \longrightarrow \{0,1\}^n,$

which takes as input a key $K$, of bit length $k$ and a bit string $P$, of length $n$, and returns a string $C$ of $n$ bits. A block cipher's decryption function can be defined as:

$E_K^{-1}(C) = D_K(C) = D(K,C) : \{0,1\}^k \times \{0,1\}^n \longrightarrow \{0,1\}^n,$

which takes as input a key $K$ and a ciphertext $C$ and returns the plaintext $P$ such that:

$\forall K : D_K(E_K(P)) = P.$

Modes of operation like AEAD based on block ciphers use a small primitive, that could be another block cipher, in conjunction with a nonce, a number that is unique to a message, in order to assure authenticity.

Take, for example, two separate messages with two blocks each being encrypted with the same key. If any block is identical to a block in the other message then the resulting blocks in the ciphertexts will be identical. A nonce per message and a simple mechanism like a counter per block are necessary for ensuring the identical blocks between different plaintexts or the same plaintext, respectively, do not correspond to identical blocks in the ciphertexts.

#### *Tweakable Block Ciphers*

A *tweakable block cipher* (TBC) functions like a block cipher with the addition of a *tweak* which has a similar function to a *nonce* or an initialization vector (or IV) Liskov et al. (2002).

We can define the encryption function of a TBC as:

$E_{K,T}(P) = E(K,T,P) : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \longrightarrow \{0,1\}^n,$

which takes as input a key $K$ of length $k$, a tweak $T$ of length $t$ and a plaintext of variable length ($n$) and outputs a ciphertext $C$ of length $n$.

We can define the decryption function as:

$E_{K,T}^{-1}(C) = D_{K,T}(C) = D(K,T,C) : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \longrightarrow \{0,1\}^n,$

which takes as input the key $K$, the tweak $T$ and the ciphertext $C$ returning the plaintext $P$ such that:
$\forall K : D_{K,T}(E_{K,T}(P)) = P$.

The tweak can be public and must be cheap to change, as opposed to a key, and its main goal is to provide variability to an adversary that is trying to attack the cipher Liskov et al..

The logic applied AEAD-based block ciphers is the same here.

*Stream Ciphers*

A *stream cipher* uses a *pseudo-random number generator* (or PRNG) to generate a *stream* of pseudo-random bits using the key as a seed and XORs it to the plaintext resulting in a ciphertext. Stream ciphers, as opposed to block ciphers, operate bit by bit or digit by digit (such as a byte) which allows them to accept variable length messages and to be very efficient Katz and Lindell.

We can define the encryption function as:
$E_{G,K}(P) = E_G(K,P) = G(K,1^{|p|}) \oplus P = C : \{0,1\}^k \times \{0,1\}^p \longrightarrow \{0,1\}^n$,

which takes as input a key $K$ of length $k$, a plaintext $P$ of length $p$ and uses a PRNG $G$ that takes the key as a seed and the length of the plaintext as input, the output of the generator is then XORed to the plaintext.

We can define the decryption function as:
$E^{-1}_{G,K}(C) = D_G(K,P) = G(k,1^{|c|}) \oplus C = P : \{0,1\}^k \times \{0,1\}^n \longrightarrow \{0,1\}^n$.

The security of this scheme is reliant on the security of the PRNG so in some literature stream cipher is used in reference to the PRNG algorithm and not the whole scheme Katz and Lindell.

*Sponge and Duplex Constructions*

The *sponge* construction Bertoni et al. (2007) is a mode of operation that uses a fixed-length permutation function ($f$) to map a variable-length input to an arbitrary-length output, much like an hash or extendable output function.

These fixed-length permutations are applied to a *state* of length, or *width*, $b$. This state is equal to a data block of length, or *bitrate* (or *rate*), $r$ and an additional block of length , or *capacity*, $c$.

This process starts with a preliminary phase where the state is initialized, for example, with an IV. Following that comes the *absorbing* phase where $r$-bit long input blocks, obtained from a reversibly padded message, are XORed to the first $r$ bits of the state, interleaved with applications of the function $f$ to the entire state. After processing all input blocks the process switches to the *squeezing* phase. In the squeezing phase the first $r$ bits of the state are returned as output blocks, interleaved with applications of the function $f$.

The $c$-bit long block in the internal state never receives a direct input, nor is it ever used as an output. The purpose of this block is to provide security to the scheme, with its length being directly tied to it Bertoni et al. (2007).

The *duplex* is a construction for symmetric encryption closely related to the sponge with equivalent security Bertoni et al. (2011a).

The main difference between the sponge and the duplex is the fact that the duplex has a single phase, the *duplexing* phase. In the initialization phase of the duplex the state is initialized, for example with a key and a

nonce. Following the initialization comes the duplexing phase where the properly padded input blocks are XORed to the first $r$-bits of the state, $f$ is applied to the state and then the first $r$-bits of the block are output as a ciphertext block. This process is repeated until all the input blocks are processed.

The sponge and duplex are proven modes of operation (Bertoni et al. (2008) Bertoni et al. (2011b) to name a few) with several advantages Keccak Team:

- From a design standing point, the proof of security of a specific cipher based on a sponge/duplex is reduced to the proof of security of the underlying permutation;
- For keyed sponge and duplex schemes there is no need for key scheduling, saving space in memory and code;
- A cipher suite that implements AE and hashing capabilities can use the same permutation for both the duplex-based AE mode and the sponge-based hashing or XOF mode.

### 2.1.5 *Confusion and Diffusion*

In 1949 Claude Shannon laid the foundation for modern cryptography where he, among many other things, introduced the *confusion-diffusion* paradigm. Confusion and diffusion are needed in order for a cipher to behave as close as possible to a random permutation, which as he proved, is perfectly secret.

Confusion can be seen as obscuring the impact of the key in the ciphertext, or in other words, a change in a bit in the key should affect all bits in the ciphertext. Diffusion can be seen as obscuring the relation between the plaintext and the ciphertext so any change in the plaintext should reflect on the entirety of the ciphertext Shannon, Katz and Lindell.

Different ciphers achieve confusion and diffusion by different means. The primitive used to achieve confusion and diffusion in block ciphers is the *substitution-permutation network* (or SPN). The textbook SPN is composed of three steps or structures: key mixing, where a subkey is added (XOR) to the plaintext, *substitution layer*, where the $n$-bit long plaintext is split into $n/s$ blocks that are fed to $s$ *S-boxes*, and the *linear layer*, where a permutation is applied to the plaintext. These three steps are repeated for a given number of times, or *rounds*.

The S-box is responsible for providing confusion to the cipher by means of a nonlinear mapping of its input, typically using a lookup table. The robustness provided by the S-box, in terms of confusion and non-linearity, is paramount as most cryptanalytical techniques targeting SPNs take advantage of the properties of the S-box in use.

The linear layer, or permutation layer, is responsible for providing diffusion of the key to the cipher by means of a reversible permutation of the bits of its input. There are many different ways of doing this permutation, one being a lookup table.

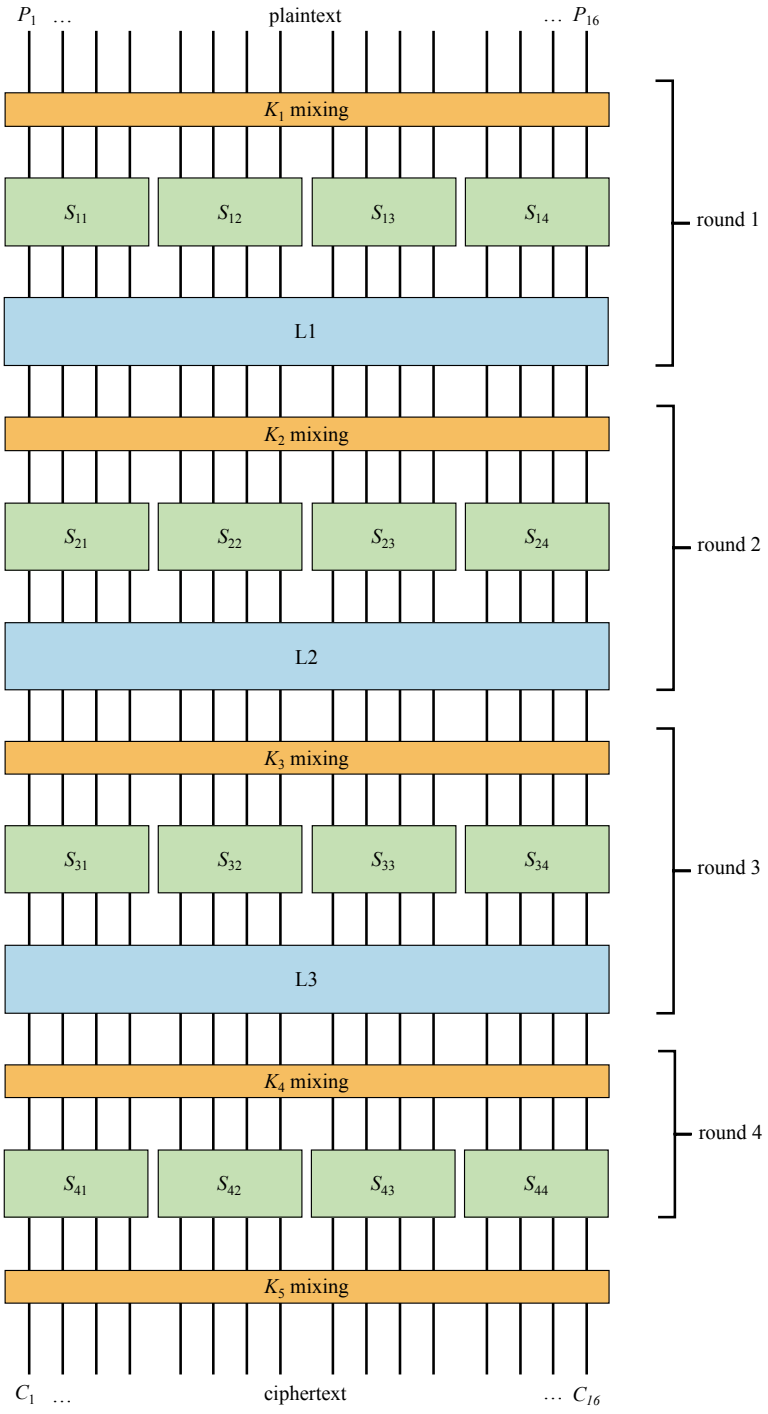



Figure 5: Substitution-Permutation Network used in Heys

## 2.2 SECURITY IN CRYPTOGRAPHY

### 2.2.1 *Encryption Scheme Security*

In order to evaluate and compare the security of different encryption schemes a strict notation was developed, consisting of an encryption goal that would be targeted by an adversary and the resources at their disposal.

*Encryption Goals*

These are the three main encryption goals to be considering when proving the security of an encryption scheme:

- *indistinguishably*, Goldwasser and Micali (1982);
- *non-malleability*, Dolev et al. (1991);
- *plaintext-awareness*, Bellare and Rogaway (1994).

Indistinguishably (or IND) formalizes the inability for an adversary to obtain any information from the ciphertext about the original plaintext, invoking a strong sense of privacy Bellare et al. (1998) Katz and Yung (2000).

Non-malleability (or NM) formalizes the inability for an adversary to, given a ciphertext $y$, to produce another ciphertext $y'$ such that the corresponding plaintexts $x$ and $x'$ are meaningfully related, invoking a sense of tamper-proof ciphertexts Bellare et al. (1998) Katz and Yung (2000).

In symmetric cryptography, contrary to what happens in public-key cryptography Bellare et al. (1998), NM does not necessarily imply IND Katz and Yung (2006).

Plaintext-awareness (or PA) is a public-key encryption exclusive encryption goal that formalizes that inability for an adversary to produce a valid ciphertext without having information on the plaintext Bellare and Rogaway Bellare et al. (1998).

*Resources*

When attacking a cipher an adversary has access to different levels of resources:

- non-adaptive chosen-plaintext attack (or CPA), Goldwasser and Micali (1982);
- adaptive chosen-plaintext attack;
- non-adaptive chosen-ciphertext attack (or CCA), Naor and Yung (1990);
- adaptive chosen-ciphertext attack (or CCA2), Rackoff and Simon (1991).

In order of increasing strength we have non-adaptive chosen-plaintext attacks where the adversary has unlimited access to a *encryption oracle* before the challenge ciphertext is revealed and adaptive chosen-plaintext attacks where the adversary retains access to the encryption oracle after the ciphertext is revealed Katz and Yung (2000).

In symmetric cryptography non-adaptive and adaptive chosen-plaintext attacks are equivalents Katz and Yung (2000).

Next we have CCA where the adversary has access to a *decryption oracle* only before the ciphertext is revealed to them Katz and Yung (2006).

Finally we have CCA where the adversary retains access to the decryption oracle after the ciphertext is revealed, with the condition that they cannot query the decryption oracle with the challenge ciphertext Katz and Yung (2000).

### 2.2.2 *Algebraic Cryptanalysis*

*Algebraic cryptanalysis*, simply put, is a method of cryptanalysis that focuses on transforming ciphers in systems of equations and then solving them for a key or plaintext Bard et al. (2007).

Relying on the extensive research of the MQ problem, which is NP-hard, we begin by transforming a problem, like a key recovery attack on a cipher, in a system of multivariate quadratic (MQ) equations Courtois and Bard (2007). Despite the "hardness" of this problem, equation systems derived from cryptographic schemes are efficiently solvable Courtois and Bard.

Another NP-hard problem is finding a satisfying assignment for a logical expression in several variables (the SAT problem) Bard et al., this problem has extensive research done on it that, inadvertently, has been proven to be very useful for algebraic cryptanalysis. By converting the low-degree sparse multivariate quadratic equation systems into a conjunctive normal form satisfiability (CNF-SAT) problem we can use a wide variety of SAT-solvers to solve these systems Bard et al..

Other algorithms for solving MQ systems are the XL-Algorithm (eXtended Linearization) Courtois et al. (2000) and Gröbner bases Buchberger (2006).

These algebraic cryptanalysis methods, unlike most methods of cryptanalysis, only need a plaintext-ciphertext pair execute a key recovery attack Courtois and Bard. This property makes algebraic cryptanalysis a technique that could be potentially very dangerous in real life scenarios Courtois and Bard.

### 2.2.3 *Differential Cryptanalysis*

#### *Introduction*

*Differential cryptanalysis* is a statistical method of cryptanalysis that exploits the effects of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs Biham and Shamir (1991).

The following is a simple and experimentally oriented explanation of differential cryptanalysis of SPNs highly inspired in Heys (2002), for a detailed explanation with a real-world example I recommend Biham and Shamir (2012).

#### *Differentials*

Lets consider a cipher with a bit-array input, or plaintext, with length $n$, $P = [P_1P_2...P_n]$ and output, or ciphertext, $C = [C_1C_2..C_n]$. Let two plaintexts to be $P$ and $P'$ with the corresponding ciphertexts $C$ and $C'$. The difference between plaintexts is given by $\Delta_I = P \oplus P'$ where $'' \oplus ''$ is the bit-wise XOR of the plaintexts. Similarly, the difference between ciphertexts is $\Delta_O = C \oplus C'$. Note that for different ciphers we can have differences other than XOR Biham and Shamir (1991).

In an ideally randomizing cipher the probability of a ciphertext difference $\Delta_O$ happening given a specific plaintext difference $\Delta_I$ is the same for every possible $\Delta_O$. In reality this is impossible so we use differential cryptanalysis to exploit scenarios where a particular $\Delta_O$ happens given a particular $\Delta_I$ with a high probability $p$. A specific pair ($\Delta_I$, $\Delta_O$) is referred to as a *differential*.

#### *Studying the S-boxes*

Lets consider the bit-array input to a S-box, given as $X$, and the output as $Y$. The difference between inputs for an S-box is given as $\Delta_X$, and the difference as $\Delta_Y$.

The first step in differential cryptanalysis is studying the S-boxes in use. We begin by computing the *difference distribution table(s)* of the S-boxes being used. The DDT of a $n \times n$ S-box has $2^n$ rows that represent every possible $\Delta_X$ and $2^n$ columns that represent every possible $\Delta_Y$.

To construct a DDT we start with a $2^n$ by $2^n$ table with its entries set to $0$ and go over every possible input pair by feeding each input to two S-boxes concurrently. We then check the difference between the outputs and increment the table entry correspondent to $\Delta_X$ and $\Delta_Y$. Alternatively we can say that for a particular $\Delta_X$, the probability of a specific $\Delta_Y$ occurring is the respective value in the DDT divided by $2^n$.

| input | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| output | E | 4 | D | 1 | 2 | F | B | 8 | 3 | A | 6 | C | 5 | 9 | 0 | 7 |

Table 1: S-box used in Heys (in hexadecimal)

| | | Output Difference | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Input Difference** | **0** | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **1** | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 2 | 4 | 0 | 4 | 2 | 0 | 0 |
| | **2** | 0 | 0 | 0 | 2 | 0 | 6 | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 0 |
| | **3** | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 2 | 0 | 0 | 4 |
| | **4** | 0 | 0 | 0 | 2 | 0 | 0 | 6 | 0 | 0 | 2 | 0 | 4 | 2 | 0 | 0 | 0 |
| | **5** | 0 | 4 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 0 | 2 |
| | **6** | 0 | 0 | 0 | 4 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 2 |
| | **7** | 0 | 0 | 2 | 2 | 2 | 0 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 4 |
| | **8** | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 4 | 0 | 4 | 2 | 2 |
| | **9** | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 4 | 2 | 0 | 2 | 2 | 2 | 0 | 0 | 0 |
| | **A** | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 2 | 0 | 0 | 4 | 0 |
| | **B** | 0 | 0 | 8 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 |
| | **C** | 0 | 2 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 6 | 0 | 0 |
| | **D** | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 2 | 0 | 2 | 0 | 2 | 0 |
| | **E** | 0 | 0 | 2 | 4 | 2 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| | **F** | 0 | 2 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 0 | 2 | 0 |

Table 2: Difference Distribution Table from Heys

In this S-box we can see that for $\Delta_X = B$ we have $\Delta_Y = 2$ in 8/16 times. It is by exploiting these highly probable difference pairs that we mount a differential attack.

*Chaining Differential Characteristics*

Lets consider the bit-array input to a specific SPN round $i$, $U_i$, and the output of the substitution layer of the same round, $V_i$. Like before, the difference between the round inputs is $\Delta_{U_i}$ and the difference between the outputs is $\Delta_{V_i}$.

The next step is to build a multi-round characteristic with a high probability. A differential characteristic is a sequence of input and output differences to the rounds so that the output difference from one round corresponds to the input difference for the next round. For this example we will study the SPN and the characteristic from Heys.

In order to build the multi-round characteristic we want we must concatenate multiple one-round characteristics. Lets say, for example, that we have $\Delta_U = [0000 \quad 1011 \quad 0000 \quad 0000]$, we know that $\Delta_V = [0000 \quad 0010 \quad 0000 \quad 0000]$ happens $8/16$ of the times, so we can say that this one-round characteristic has $p = 8/16$.

In rounds with multiple active S-boxes (S-boxes with non-zero inputs) $p$ is the product of all the difference pairs of those S-boxes, for example, with $\Delta_U = [1011 \quad 1011 \quad 0000 \quad 0000]$ we have $\Delta_V = [0010 \quad 0010 \quad 0000 \quad 0000]$ with $p = 8/16 \times 8/16 = 1/4$.

The probability of a multi-round characteristic occurring is the product of the probability all one-round characteristics occurring. Lets consider the following difference pairs:

- $S_{12} : \Delta_{U_1} = B \rightarrow \Delta_{V_1} = 2$ with probability $8/16$
- $S_{23} : \Delta_{U_2} = 4 \rightarrow \Delta_{V_2} = 6$ with probability $6/16$
- $S_{32} : \Delta_{U_3} = 2 \rightarrow \Delta_{V_3} = 5$ with probability $6/16$
- $S_{32} : \Delta_{U_3} = 2 \rightarrow \Delta_{V_3} = 5$ with probability $6/16$

The resulting three-round characteristic has probability $8/16 * 6/16 * (6/16)^2 = 27/1024$.

The linear layer and subkey mixing wildly differ in impacting the chaining of characteristics (and the study of S-boxes). The subkey mixing has no effect because the subkey is canceled out in $\Delta_U$ since: $(U \oplus K) \oplus (U' \oplus K) = U \oplus U' = \Delta_U$. In order to concatenate different-round characteristics we must ensure that, after the application of the linear layer, $\Delta V_i$ coincides with $\Delta U_{i+1}$. This is possible because the linear layer does not change the differences, it just changes their location.

Determining what the best characteristic for any differential is the main obstacle in differential cryptanalysis and is a process that gets more difficult with the number of rounds the SPN has. For an example on how to derive the best characteristics of a SPN I point to Matsui (1994) and Makarim and Rohit (2022).

*Distinguisher and Key Recovery*

After constructing a $n$-round characteristic for a SPN with $n$ rounds we can use it to distinguish the cipher from a random permutation by encrypting a number, $N$, of plaintext pairs and verifying if the difference in outputs matches $\Delta_O$ in $N \times p$ of the times.

In order to transform this distinguisher into a key-recovery attack we use a $n-1$ round characteristic and attempt to recover bits from the last subkey.

To mount a key-recovery attack we begin with an empty counter, an array with $2^{x*n}$ elements, the number of all possible subkeys, with $x$ being the number of active S-boxes in the last round, and $n$ their size in bits. Then for a number, $N$, we generate a plaintext $P$ and a plaintext $P' = P' \oplus \Delta_I$.

We encrypt the pair and check $\Delta_O$, if the bits in the positions corresponding to the active S-boxes ($[\Delta_{O_5}\Delta_{O_6}\Delta_{O_7}\Delta_{O_8}]$ and $[\Delta_{O_{13}}\Delta_{O_{14}}\Delta_{O_{15}}\Delta_{O_{16}}]$ in this case) are zero we move on to the next pair because the characteristic did not occur (these are called *wrong pairs*).

For *right pairs*, we go over every possible subkey ($K_5$) and undo the last subkey mixing and S-box application. Note that we do not need to do this for the whole message but only for the blocks of bits corresponding to active S-boxes in the last round ($[K_{5_5}K_{5_6}K_{5_7}K_{5_8}]$ and $[K_{5_{13}}K_{5_{14}}K_{5_{15}}K_{5_{16}}]$ in this case).

If the input difference to the S-boxes in the last round matches the input difference from the characteristic we built ($\Delta_X = \Delta_{U_4}$) then we increase the counter of the subkey being tested.

The subkey with the biggest counter in the end is most likely to be the correct subkey.

*Bias and Complexity*

The goal of cryptanalysis is to establish and compare the complexity or time needed to break a scheme (or a round-reduced version of it) between different techniques.

For a specific differential characteristic we say that the probability of it occurring is $p$, $\Delta_I \xrightarrow{p} \Delta_O$. In order to measure complexity, which in this case is the number of plaintext pairs needed, we do not come up with an exact number, but with a small number of plaintext pairs big enough to verify the characteristic we chose. In other words, with $N$ being the number of plaintext pairs and $c$ a small number that is close to a multiple of $1/p$ we have $N \approx c/p$.

The output of a S-box for a given difference pair is not always the same and as such we can have a differential $(\Delta_I, \Delta_O)$ with multiple differential characteristics. Thanks to this differential cryptanalysis is not about proving that differential characteristics are below an acceptable threshold but that every differential is below an acceptable threshold.

*Further Reading*

There have been multiple contributions to the field of differential cryptanalysis, such as, *truncated differentials attacks* Knudsen (1994), attacks using *higher-order differentials* Knudsen (1994) based on Lai (1994), *impossible differential attacks* Knudsen (1998), *boomerang* Wagner (1999) and *amplified boomerang attacks* Kelsey et al. (2000) and the *rectangle attack* Biham et al. (2001).

### 2.2.4 *Linear Cryptanalysis*

#### *Introduction*

*Linear cryptanalysis* is a statistical method of cryptanalysis that exploits highly probable occurrences of linear expressions involving the plaintext, ciphertext and subkey bits Heys.

The following is a very simplified and experimentally oriented explanation of linear cryptanalysis of SPNs highly inspired in Heys, for a detailed explanation with a real-world example I recommend Matsui (1993).

#### *Linear Expressions*

The goal of linear cryptanalysis is to find a *effective* linear approximation for a given cipher, borrowing the notation from the previous example, such that:

$P_1 \oplus P_2 \oplus ... \oplus P_a \oplus C_1 \oplus C_2 \oplus ... \oplus C_b = K_1 \oplus K_2 \oplus ... \oplus K_c$,

holds with probability $q$ different from $1/2$ for any random $P$ and $C$. The effectiveness of this linear expression is tied to the magnitude of $q - 1/2$.

Following (Matsui, 1993, *Algorithm 2*) we study linear expressions of the form:

$P_1 \oplus P_2 \oplus ... \oplus P_a \oplus C_1 \oplus C_2 \oplus ... \oplus C_b = 0$.

#### *Studying the S-boxes*

The first step in linear cryptanalysis is to study the linear properties of the S-boxes in use by computing their *linear approximation tables*.

Following (Matsui, 1993, *Definition 1*), for a $n \times n$ S-box $S_n$, $1 \leq \alpha \leq 2^n - 1$ and $1 \leq \beta \leq 2^n - 1$, $NS_n(\alpha, \beta)$ is the number of times out of $2^n$ input patterns of $S_n$ such that an XORed value of the input bits masked by $\alpha$ coincides with and XORed value of the output bits masked by $\beta$, formally:

$NS_n(\alpha, \beta) := \#\{x | 0 \leq x < 2^n, (\oplus_{s=0}^{n-1}(x[s] \bullet \alpha[s])) = (\oplus_{t=0}^{n-1}(S_n(x)[t] \bullet \beta[t]))\}$,

where $\bullet$ denotes a bitwise AND operation. When $NS_n(\alpha, \beta)$ is different from $2^{n-1}$ we can say that there is a correlation between the input and the output bits of $S_n$.

The LAT of a $n \times n$ S-box has $2^n$ rows that represent $\alpha$ and $2^n$ columns that represent $\beta$, with the value in each entry being $NS_n(\alpha, \beta) - 2^{n-1}$.

The LAT of the S-boxes used in Heys is as follows.

| | | Output Sum | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Input Sum** | **0** | +8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **1** | 0 | 0 | −2 | −2 | 0 | 0 | −2 | +6 | +2 | +2 | 0 | 0 | +2 | +2 | 0 | 0 |
| | **2** | 0 | 0 | −2 | −2 | 0 | 0 | −2 | −2 | 0 | 0 | +2 | +2 | 0 | 0 | −6 | +2 |
| | **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | +2 | −6 | −2 | −2 | +2 | +2 | −2 | −2 |
| | **4** | 0 | +2 | 0 | −2 | −2 | −4 | −2 | 0 | 0 | −2 | 0 | +2 | +2 | −4 | +2 | 0 |
| | **5** | 0 | −2 | −2 | 0 | −2 | 0 | +4 | +2 | −2 | 0 | −4 | +2 | 0 | −2 | −2 | 0 |
| | **6** | 0 | +2 | −2 | +4 | +2 | 0 | 0 | +2 | 0 | −2 | +2 | +4 | −2 | 0 | 0 | −2 |
| | **7** | 0 | −2 | 0 | +2 | +2 | −4 | +2 | 0 | −2 | 0 | +2 | 0 | +4 | +2 | 0 | +2 |
| | **8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | −2 | +2 | +2 | −2 | +2 | −2 | −2 | −6 |
| | **9** | 0 | 0 | −2 | −2 | 0 | 0 | −2 | −2 | −4 | 0 | −2 | +2 | 0 | +4 | +2 | −2 |
| | **A** | 0 | +4 | −2 | +2 | −4 | 0 | +2 | −2 | +2 | +2 | 0 | 0 | +2 | +2 | 0 | 0 |
| | **B** | 0 | +4 | 0 | −4 | +4 | 0 | +4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **C** | 0 | −2 | +4 | −2 | −2 | 0 | +2 | 0 | +2 | 0 | +2 | +4 | 0 | +2 | 0 | −2 |
| | **D** | 0 | +2 | +2 | 0 | −2 | +4 | 0 | +2 | −4 | −2 | +2 | 0 | +2 | 0 | 0 | +2 |
| | **E** | 0 | +2 | +2 | 0 | −2 | −4 | 0 | +2 | −2 | 0 | 0 | −2 | −4 | +2 | −2 | 0 |
| | **F** | 0 | −2 | −4 | −2 | −2 | 0 | +2 | 0 | 0 | −2 | +4 | −2 | −2 | 0 | +2 | 0 |

Table 3: Linear Approximation Table from Heys

To compute this table we begin with a $2^n \times 2^n$ table with all its entries set to 0. For every possible $\alpha$ we go over every possible $\beta$ and set a counter to 0. For all possible inputs we calculate $a = input \bullet \alpha$ and $b = S_n(input) \bullet \beta$. If the amount of bits to 1 in $a$ minus the amount of bits to 1 in $b$ is even then increment the counter. After going over all the inputs for each $\alpha$ and $\beta$ update the respective table entry with the counter minus $2^{n-1}$.

The bias of the linear equation, for example, $X_1 \oplus X_2 = Y_1 \oplus Y_4$ is $-6/16 = -3/8$ and the probability that the linear equation holds true is $q = 1/2 - 3/18 = 1/8$.

*Chaining Linear Characteristics*

The next step is to build a multi-round approximation with an *effective* probability. The way this works is very similar to differential characteristics so some of the details and notation will be omitted. Regardless of this there are some changes from the differential approach.

The first change is that, obviously we are not working with a plaintext pair but with a single plaintext so there is still the influence of the key but in a different way than expected. The second difference is that our goal is not to increase the probability but to deviate it from $1/2$ as much as possible.

We will follow the same example from Heys with a note, contrary to the differential section where the bit significance of the S-boxes input and output is in tandem with the whole cipher (least significant bit to the left), the input and output to the S-boxes in this section have a reversed bit significance, hence the difference in example in the previous example when explaining the bias and probabilities of the linear equations.

The example makes use of the same active S-boxes from before, $S_{12}, S_{22}, S_{32}, S_{34}$. For the input to the first-round S-boxes we have $[U_{1_5}U_{1_7}U_{1_8}] = [(P_5 \oplus K_{1_5})(P_7 \oplus K_{1_7})(P_8 \oplus K_{1_8})]$. Taking this input and looking at its correspondent row $B$ we have column $4$ with a value of $+4$. If the author was not aiming to use the same active s-boxes he would aim to use the linear expressions with the highest absolute value in their entries.

Following the differential approach we arrive at the following linear approximation:

- $S_{12} : X_1 \oplus X_3 \oplus X_4 = Y_2$ (row $B$, column $4$) with probability 12/16 and bias $+1/4$
- $S_{22} : X_2 = Y_2 \oplus Y_4$ (row $4$, column $5$) with probability 4/16 and bias $-1/4$
- $S_{32} : X_2 = Y_2 \oplus Y_4$ (row $4$, column $5$) with probability 4/16 and bias $-1/4$
- $S_{34} : X_2 = Y_2 \oplus Y_4$ (row $4$, column $5$) with probability 4/16 and bias $-1/4$

With this approximation we get the following linear expression:

$P_5 \oplus P_7 \oplus P_8 \oplus U_{4_6} \oplus U_{4_8} \oplus U_{4_{14}} \oplus U_{4_{16}} \oplus \sum_K = 0,$

where, by looking at the inputs to the S-boxes, we have $\sum_K = K_{1_5} \oplus K_{1_7} \oplus K_{1_8} \oplus K_{2_6} \oplus K_{3_6} \oplus K_{3_{14}} \oplus K_{4_6} \oplus K_{4_8} \oplus K_{4_{14}} \oplus K_{4_{16}}$ which is fixed at $1$ or $0$ depending on the key of the cipher. By using the *Piling-Up Lemma* Matsui (1993) we can know that the probability of this expression holding is $1/2 + 2^3(3/4 - 1/2)(1/4 - 1/2)^3 = 15/32$ (which means it has a bias of $-1/32$).

Since $\sum_K$ is fixed we have:

$P_5 \oplus P_7 \oplus P_8 \oplus U_{4_6} \oplus U_{4_8} \oplus U_{4_{14}} \oplus U_{4_{16}} = 0,$

that must hold with a probability of either $15/32$ or $1 - 15/32 = 17/32$ depending on whether $\sum_K = 0$ or $1$ respectively.

Just like in differential cryptanalysis finding good approximations is the main challenge in linear cryptanalysis so, for an example on how to derive the best approximations, I recommend Matsui (1994) and Dobraunig et al. (2015a).

*Distinguisher and Key Recovery*

For simplicity lets assume that $P_1 \oplus P_2 \oplus ... \oplus P_a$ is $\lambda_I$ and $C_1 \oplus C_2 \oplus ... \oplus C_b$ us $\lambda_O$.

After constructing a $n$-round characteristic for a SPN with $n$ rounds we can use it to distinguish the cipher from a random permutation by encrypting a number, $N$, of plaintexts and verifying the absolute deviation of $\lambda_I \oplus \lambda_O = 0$ from $N/2$.

In order to transform this distinguisher into a key-recovery attack we use a $n - 1$ round characteristic and attempt to recover bits from the last subkey.

We begin with an empty counter, an array with $2^{x*n}$ elements, the number of all possible subkeys, with $x$ being the number of active S-boxes in the last round, and $n$ their size in bits.

We encrypt a random number (to be discussed) of random plaintexts and go over every possible subkey ($K_5$) and undo the last subkey mixing and S-box application. Note that we do not need to do this for the whole message but only for the blocks of bits corresponding to active S-boxes in the last round ($[K_{5_5}K_{5_6}K_{5_7}K_{5_8}]$ and $[K_{5_{13}}K_{5_{14}}K_{5_{15}}K_{5_{16}}]$ in this case).

If the linear expression derived from the characteristic holds true we increase the counter of the respective subkey.

Following (Matsui, 1993, *Algorithm 2*) the subkey counter that deviates the most from half the number of plaintexts encrypted is most likely to be the correct subkey.

*Bias and Linear Hull*

For a specific linear approximation, $\lambda_I \xrightarrow{q} \lambda_O$, we consider the magnitude of its bias $q$ when deciding the complexity of the attack. Matsui (1993) shows that $N$, the number of plaintexts necessary to mount an attack, is proportional to $q^{-2}$,

$N \approx 1/q^2$,

but in practice a small multiple of $N$ is better when aiming for an attack that is the most successful.

The Piling-Up Lemma assumes that each S-box approximation in independent from the others but, like what was seen in differential cryptanalysis, it is not the case. There can be multiple linear approximations that make use of the same plaintext and last-round input bits combining in order to give a higher linear probability than expected. This concept is referred to as *linear hull* Nyberg (1994). The approach outlined in Heys works well because the independence assumption is a reasonable approximation and when one linear approximation scenario of a particular set of active S-boxes has a high bias, it tends to dominate the linear hull Heys.

*Further Reading*

There have been multiple contributions to the field of linear cryptanalysis, such as, Selçuk (2008) that formulates a better *success probability* for linear and differential cryptanalysis as well as a better study on the influence of the key in linear cryptanalysis, Nyberg (1994) where she introduces the concept of *linear hull*, Hermelin et al. (2009) that improves *Algorithm 2* from Matsui (1993) and Bogdanov and Wang (2012) where a linear counterpart to impossible differentials is introduced, *zero-correlation attacks*.

### 2.2.5 *Differential-Linear Cryptanalysis*

#### *Introduction*

*Differential-linear cryptanalysis* as presented in Langford and Hellman (1994) is a cryptanalysis technique that combines the differential and linear techniques. The way this is done is by splitting the cipher that is being analyzed in two, the first part has a strong truncated differential and the second has an effective linear approximation.

In Langford and Hellman the differential is chosen in a way that assures the parity, or the probability of the linear approximation in the input of the S-boxes in the linear part, is unchanged for all plaintext pairs. Biham et al. (2002) presents an extension of Langford and Hellman by allowing the use of a differential characteristic that does not have an assured parity.

Chabaud and Vaudenay (1994) studies the links between differential and linear cryptanalysis, showing that the probability of a differential can be expressed as a sum of correlations of linear approximations Blondeau et al. (2017), presenting new classes of functions which are optimally resistant to differential and linear cryptanalysis, and proving that linear-resistant functions are differential-resistant as well.

Blondeau et al. applies Chabaud and Vaudenay to differential-linear cryptanalysis and expresses, for the first time, the bias of a differential-linear approximation as an enclosed expression.

#### *Differential-Linear Characteristics*

Lets consider that the cipher in use, $E$, can be divided in two parts with $E = E_1 \circ E_0$. $E_0$ has a strong differential with probability $p$, $\Delta_I \xrightarrow{p} \Delta_O$, and $E_1$ has an effective linear approximation, that is highly influenced by the differential Langford and Hellman, with bias of magnitude $q$, $\lambda_I \xrightarrow{q} \lambda_O$. Both $E_0$ and $E_1$ can be viewed as round-reduced versions of $E$.

To build a differential-linear characteristic we simply concatenate the differential characteristic with the linear approximation. Obviously, this is easier said than done and there are many techniques on how to find good differential-linear characteristics, both experimentally and analytically.

#### *Distinguisher and Key Recovery*

The same technique of turning our $n$-round characteristic into a $n-1$ round one is use here in order to turn a distinguishing attack into a key-recovery attack. Following the example in (Biham et al., 2002, section 7) the generalization for a DL key recovery attack is as follows.

We begin with an empty counter, an array with $2^{x*n}$ elements, the number of all possible subkeys, with $x$ being the number of active S-boxes in the last round, and $n$ their size in bits.

Then we encrypt an appropriate number $N$ of plaintext pairs that match the input difference from the DL approximation.

For each ciphertext pair we go over all possible subkeys and do a partial decryption of both ciphertexts in order to get their inputs to the last round of S-boxes. If the parities of both ciphertexts are the same then increment the counter of the subkey in question.

The subkey with the highest bias magnitude ($|counter - N/2|$) is the correct subkey.

*Bias and Differential-Linear Hull*

The differential-linear bias has been studied extensively, with its first expression as an enclosed expression in Blondeau et al..

Lets consider $E'$, a cipher, with its inputs $P$ and $P'$, and its outputs $C$ and $C'$. For an input difference $\Delta_I$ and output parity $\lambda$ we have:
$\epsilon^{E'}_{\Delta_I,\lambda} = Pr[C \cdot \lambda = C' \cdot \lambda | P \oplus P' = \Delta_I] - 1/2$
and for input and output parities $\lambda_I$ and $\lambda_O$:
$c^{E'}_{\lambda_I,\lambda_O} = 2(Pr[C\dot{\lambda}_O = C' \cdot \lambda_O | P \cdot \lambda = P'\dot{\lambda}] - 1/2)$,
with $q = c/2$.

Assuming that $E_0$ and $E_1$ are independent we have Blondeau et al.:
$\mathcal{E}_{\Delta_I,\lambda_O} = \sum_{\lambda_I} \epsilon^{E_0}_{\Delta_I,\lambda_I} (c^{E_1}_{\lambda_I,\lambda_O})^2$.

Analogous to the linear hull and similar properties in differential cryptanalysis, there is a differential-linear hull. Just like before with multiple linear approximations with the same output parity in the same plaintext and different input differences producing the same output differences not being represented in the final bias there is one more factor needing to be taken into account. When going from $E_0$ to $E_1$ there are multiple $\lambda_I$ produced by $\Delta_O$ that lead to the same $\lambda_O$. These three factors make so that the experimental bias is higher than the theoretical one.

### 2.2.6 *Differential-Linear Connectivity Table*

#### *Introduction*

The Differential-Linear Connectivity Table and framework introduced in Bar-On et al. (2019) aims to partially take the effects of round dependency between the differential and linear parts of the cipher into account when calculating the overall differential-linear bias of the cipher.

#### *DLCT - Definition*

The DLCT of a vectorial Boolean function $S$ ($S : \{0,1\}^n \rightarrow \{0,1\}^m$) is a $2^m \times 2^m$ table whose rows correspond to input differences to $S$ and whose columns correspond to bit masks of outputs of $S$. Formally, for $\Delta \in \{0,1\}^n$ and $\lambda \in \{0,1\}^m$, the DLCT entry $(\Delta, \lambda)$ is:

$DLCT_S(\Delta, \lambda) := |\{x|\lambda \cdot S(x) = \lambda \cdot S(x \oplus \Delta)\}| - 2^{n-1}$.

The normalized DLCT entry is:

$\overline{DLCT}_S(\Delta, \lambda) := \frac{DLCT_S(\Delta,\lambda)}{2^n} = Pr[\lambda \cdot S(x) = \lambda \cdot S(x \oplus \Delta)] - \frac{1}{2}$.

$DLCT_S(\Delta, \lambda)$ is equal (up to normalization) to the bias $\mathcal{E}_{\lambda,\Delta}$.

#### *DLCT - Framework*

When using the DLCT framework $E$ is divided into $E = E_1^{'} \circ E_m \circ E_0^{'}$, with $E_0^{'}$ being equivalent to $E_0$, $E_m$ usually covering the first round of $E_1$ and $E_1^{'}$ covering the remainder of $E_1$. When using this framework the bias of the differential-linear distinguisher is:

$$\mathcal{E}_{\Delta_I,\lambda_O} = \sum_{\Delta,\lambda} Pr[\Delta_I \underset{E_0^{'}}{\longrightarrow} \Delta] \cdot \overline{DLCT}_{E_m}(\Delta, \lambda)(c_{\lambda,\lambda_O}^{E_1^{'}})^2.$$

The advantage that the DLCT framework offers over Blondeau et al. is that it takes into account the dependence between $E_0$ and $E_1$. Nevertheless, round dependence inside $E_0^{'}$ and $E_1^{'}$ still persists, making the DLCT framework a non-perfect improvement over the previous attempts at theoretically express the differential-linear bias.

## 2.3 NIST LIGHTWEIGHT CRYPTOGRAPHY STANDARDIZATION PROCESS

In 2018 NIST initiated their Lightweight Cryptography Standardization Process (NIST-LWC) in order to standardize one or more AEAD and hashing schemes for use in highly constrained environments such as sensor networks, distributed control systems, RFID tags and embedded systems.

This initiative was preceded by several years of study and research culminating in McKay et al. (2016) where the authors summarize their findings and announce their plans for the NIST-LWC.

The standardization process underwent a preliminary phase Turan et al. (2019), a second phase that narrowed down the candidates to ten finalists Turan et al. (2021) and is currently in its third and final phase predicted to conclude in late 2022 to 2023 Turan (2022).

### 2.3.1 *On Lightweight Cryptography*

Lightweight cryptography is a subfield of cryptography that is tailored for use in resource-constrained environments. These environments could be a decentralized network of small devices working with a common goal but they could also be a centralized network with small devices communicating with a larger aggregator that should not be constrained. This means that lightweight schemes must not just be designed for use in hardware in constrained devices but for use in software as well McKay et al..

#### *Lightweight Primitives*

The base for any cryptographic scheme is its underlying primitive so, in order to tackle the difficulties posed by the environments these schemes are used in, we must use the advancements made in this field in order to design schemes with a better balance between performance, resource requirements and security than the existing standards McKay et al..

Some of the design choices made when designing a lightweight *block cipher* can be McKay et al.:

- Using a *smaller block size* in order to save memory at the cost of decreasing the maximum limit of plaintext blocks that can be encrypted at the same time;
- Using *smaller key sizes* in order to save memory at the cost of decreased security;
- Designing *simpler rounds* with smaller S-boxes and simpler linear layers for more efficiency in hardware at the cost of increasing the number of rounds needed to achieve security;
- Using *simpler key schedules* in order to save memory and decrease latency and power consumption at the cost of opening the scheme to related key, weak key, known key or chosen key attacks;
- Using *minimal implementations* since some applications may not require the whole cipher suite but only the encryption function or the decryption function for example.

One relevant aspect about the block size is that in these constrained environments messages are usually short so it can be sufficient or even beneficial to not go for the smallest block size possible as it would mean more encryptions per message which leads to less performance.

Lastly, lightweight block ciphers are susceptible to the same attack techniques as regular block ciphers but, with simpler rounds, they have a reduced algebraic degree which makes them much more susceptible to algebraic cryptanalysis.

Despite the fact that *stream ciphers* are falling in disuse thanks to their reduced security when compared to block ciphers Katz and Yung (2000), they are suited for use in lightweight schemes since they offer increased performance over their block cipher counterparts.

Lightweight *hash functions* differ from conventional hash functions in some aspects such as McKay et al.:

- *Smaller internal state and output sizes*. A large output size is important for applications where collision resistance is needed but for cases where collision resistance is not needed a smaller internal state and output size can be used. In cases where collision resistance is needed a smaller internal state size might be used;
- *Smaller message size*. Conventional hash functions are expected to support very large inputs (around $2^{64}$ bits). The expected input sizes in environments these lightweight hash functions are designed for are very small (around $256$ bits) so, hash functions that are optimized for small input sizes are desired.

### 2.3.2 *Submission Requirements*

In 2018 NIST outlined the requirements for submissions for the standardization process.

#### *AEAD*

The first requirement for the submissions is to include an AEAD algorithm in their cipher suite.

These algorithms must ensure the confidentiality of the plaintext under adaptive chosen-plaintext attacks and the integrity of the ciphertext under adaptive forgery attempts.

The candidate can submit a family of one or more (up to ten) AEAD algorithms. This family must have a primary member with a key of at least $128$ bits, a nonce of at least $96$ bits and a tag of at least $64$ bits, and an input size limit of at least $2^{50} - 1$ bytes.

Any element of this family of algorithms must have a key at least $128$ bits long and require at least $2^{112}$ classical computations for any cryptanalytical attack in a single-key setting.

#### *Hashing*

The candidates can optionally submit a hashing scheme with their submission.

These schemes must feature collision and (second) preimage resistance, with attacks that try to exploit these features requiring at least $2^{112}$ classical computations. Additionally, the digest must be at least $256$ bits.

The candidate can submit a family of one or more (up to ten) hashing schemes. This family must have a primary member with a digest size of exactly $256$ bits and a maximum input size of at least $250 - 1$ bytes.

Submissions must also state what are the design elements in common between the AEAD and hashing algorithms and how they lead to a reduced implementation cost.

### 2.3.3 *Evaluation*

The candidates are evaluated in four main criteria Turan (2022):

- Security;
- Performance in hardware;
- Performance in software;
- Additional features that do not fall in with the other three categories.

#### *Security*

Security is always the key priority in any cryptographic subfield, when it comes to lightweight cryptography it is a question of balancing our security needs with our performance or resource requirement needs, with a constrained environment in mind of course.

Security includes some sub criteria such as maturity of designs Turan et al. (2019), the security claims and proofs made by the candidates and third party analysis.

#### *Performance*

Performance in hardware and software fall under the same umbrella of overall performance.

When it comes to performance in general, the candidates are compared to existing NIST standards and between each other. Side channel resistance and flexibility between different platforms and environments are also compared.

#### *Additional Features*

Features like the easiness of standardization or design diversity, for example Romulus Guo et al. which is the only TBC among the finalists, are favored.

Other security features such as post-quantum security or the impacts of state recovery that are security adjacent are considered additional features and are favored as well.

### 2.3.4 *Candidates*

After publishing 2018, NIST received 57 submissions of which 56 passed to the first phase of the standardization process Turan et al. (2019).

After the first phase, 32 candidates were selected to move up to the second phase. These candidates were chosen based mainly on their maturity and cryptographic analysis Turan et al. (2019). Following this, 10 finalists were chosen based on their security, performance, tweak plans, design diversity and how they fit in in the existing NIST portfolio of existing cryptographic standards Turan et al. (2021).

The ten finalists are as follows:

- ASCON (AEAD and hashing), a classical sponge with an underlying public permutation Dobraunig et al.;
- Elephant (AEAD only), a parallel nonce-based encrypt-then-mac construction with an underlying permutation Beyne et al.;
- GIFT-COFB (AEAD only), a combined feedback block cipher with an underlying block cipher Banik et al.;
- Grain-128AEAD (AEAD only), a stream cipher based scheme Hell et al.;
- ISAP (AEAD only), a nonce-based encrypt-then-mac construction with an underlying permutation Dobraunig et al.;
- PHOTON-Beetle (AEAD and hashing), a modified sponge with an underlying public permutation Bao et al.;
- Romulus (AEAD and hashing), a tweakable block cipher with an underlying tweakable block cipher Guo et al.;
- SPARKLE (SCHWAEMM for AEAD and ESCH for hashing), a modified sponge with an underlying public permutation Beierle et al.;
- TinyJAMBU (AEAD only), a classical sponge with an underlying secret permutation Wu and Huang;
- Xoodyak (AEAD and hashing), a classical sponge with an underlying public permutation Daemen et al.;.

## 2.4 SECURITY IN THE NIST-LWC

The main concern in the NIST-LWC is the cryptographic security of the candidates. NIST will base their decision not only on the security claims and proofs of the candidates but on their own analysis and third party analysis as well.

### 2.4.1 *Security Analysis*

For the selection of the finalists NIST followed the security criteria outlined in the previous section with an emphasis on third-party results that challenged the validity of the security claims, such as distinguishers on underlying permutations, practical forgery attacks, weak key classes and key recovery attacks Turan et al. (2021).

### 2.4.2 *Side-Channel Resistance*

Bellizia et al. (2020) introduced a framework on which to qualify the achievable security of an AE algorithm in the face of nonce-misuse and side-channel leakage. This framework presents a grade system to assign to the algorithms based on confidentiality and integrity retained in different nonce-misuse and side-channel leakage scenarios. *Grade-3*, the strongest notion of achievable security, considers nonce misuse-resilient CCA security with an unique challenge nonce (CCAm) and nonce misuse-resistant ciphertext integrity (CIM) in the L2 leakage model (encryption and decryption leakage). *Grade-2* considers the same CIML2 notion from *Grade-3* and CCAmL1, nonce misuse-resilient CCA security in the L1 leakage model (encryption leakage only).

The finalists that achieve *Grade-2* and above are:

- *Grade-2* (CCAmL1, CIML2): ASCON;
- *Grade-3* (CCAmL2, CIML2): ISAP and Romulus-T.

Following Turan et al. (2021) there has been a lot of investigation on the side-channel resistance of the finalists with the example of Verhamme et al. (2022). In this article the authors conclude that a qualitative point of view when comparing candidates is more important than a quantitative one. This point of view is shared by the other investigations in the field.

### 2.4.3 *Nonce-Misuse Security*

The notion of nonce misuse-resistance authenticated encryption, or MRAE, was introduced in Rogaway and Shrimpton (2006). In this notion the adversary may repeat the nonce as many times as they please as long as they do not repeat the same query.

The notion of nonce misuse-resilient CCA security introduced in Ashur et al. (2017), later named CCAm, dictates that confidentiality must hold as long as the nonce being reused is *fresh*.

Out of the ten finalists:

- Elephant provides authenticity under nonce-reuse;
- Romulus-T provides nonce-misuse resilience, or CCAm security;
- Romulus-M provides nonce-misuse resistance, MRAE;

### 2.4.4 *Releasing Unverified Plaintext Security*

The release of unverified plaintext (RUP) attack model introduced in Andreeva et al. (2014) is an attack where the adversary recovers an unverified plaintext regardless of the verification result. A scheme that achieves integrity of the ciphertexts in the RUP model is INT-RUP secure and a scheme that achieves privacy of the plaintext is PA secure.

Out of the ten finalists:

- Elephant achieves INT-RUP security due to its EtM mode of operation;
- Both Romulus-M and Romulus-T achieve INT-RUP and PA1 security Guo et al..

### 2.4.5 *Impacts of State Recovery*

Candidates based on a (tweakable) block cipher or a keyed permutation guarantee no security in the event of an internal state recovery due to the implication of the disclosure of the secret key in the event of a state recovery. Candidates based on sponge-type modes with keyed initialization and finalization, such as ASCON and ISAP, are resistant against internal state recovery.

### 2.4.6 *Post-Quantum Security*

In order to increase post-quantum security candidates feature variants with increased key size. Among the finalists, ASCON, SPARKLE and TinyJAMBU feature said variants, with ASCON having a variant specifically designed for post quantum-resistance, ASCON-80pq Turan et al. (2021).

## 2.5 PERFORMANCE IN THE NIST-LWC

Due to the fact that these lightweight cryptographic algorithms will be used by resource-constrained devices, performance is a top priority, following security. Thanks to this, a thorough benchmarking of the candidates is needed. The benchmarking is divided in two categories, software and hardware benchmarking.

### 2.5.1 *Software Benchmarks*

Software benchmarks are done on platforms ranging from 8-bit to 64-bit microcontrollers and measure different metrics or compare the candidates to preexisting algorithms.

#### *NIST*

NIST evaluated the performance of second round candidates on microcontrollers, comparing their AEAD variants to AES-GCM Dworkin (2007) and their hashing variants to SHA-256 Dang et al. (2015). This benchmarking measured code size and execution time, and used different input lengths, ranging from 8 to 1024 bytes.

The platforms used by this benchmark were:

| Microcontroller | Processor | Word size |
|---|---|---|
| ATmega328P | AVR | 8-bit |
| ATmega4809 | AVR | 8-bit |
| SAM D21 | ARM Cortex-M0+ | 32-bit |
| nRF52840 | ARM Cortex-M4 | 32-bit |
| PIC32MX340F512H | MIPS32 M4K | 32-bit |
| ESP8266 | Tensilica L106 | 32-bit |

Table 4: Platforms used in NIST

The results of this benchmark were summarized in Turan et al. (2021), these are those results with a focus on the ten finalists:

- For AEAD:
  - PHOTON-Beetle achieved the smallest code size on 8-bit platforms;
  - On the 8-bit platform ATmega4809, the code sizes of GIFT-COFB and TinyJAMBU are also less than that of AES-GCM;
  - ASCON, GIFT-COFB, and TinyJAMBU are smaller than AES-GCM on all 32-bit platforms;
  - Elephant, SPARKLE, and Xoodyak have smaller code sizes on some of the 32-bit platforms;
  - Overall, TinyJAMBU has the smallest code size across all platforms, and ASCON and Xoodyak perform particularly well on 32-bit platforms.
- For Hashing:
  - SPARKLE achieves smaller code size than SHA-256 on all the tested platforms;
  - ASCON outperforms SHA-256 except on SAM D2 and ESP8266;
  - Xoodyak outperforms SHA-256 except on nRF52840;
  - PHOTON-Beetle achieves the smallest code size on 8-bit platforms while performing worse on 32-bit platforms.

*Renner et al.*

The team behind Renner et al. (2019) developed an open-source benchmarking framework for AEAD algorithms and applied it to the second round candidates of the NIST-LWC, measuring execution time, size of the compiled binary, and RAM usage (for one platform).

The platforms used by this benchmark were:

| Microcontroller | Processor | Word size |
|---|---|---|
| ATmega328P | AVR | 8-bit |
| STM32F103 | ARM Cortex-M3 | 32-bit |
| STM32F746ZG | ARM Cortex-M7 | 32-bit |
| ESP32 WROOM | Tensilica Xtensa LX6 | 32-bit |
| Kendryte K210 | RISC-V (Dual Core) | 64-bit |

Table 5: Platforms used in Renner et al.

The results of this benchmark, for the primary variants, were summarized in Turan et al. (2021), these are those results with a focus on the ten finalists:

- On the ATmega328P:
  - SCHWAEMM, TinyJAMBU and GIFT-COFB had the fastest implementations ;
  - PHOTON-Beetle, TinyJAMBU, Xoodyak, and GIFT-COFB had the smallest code sizes.
- On the STM32F103:
  - Xoodyak, ASCON, SCHWAEMM, TinyJAMBU, and GIFT-COFB were the fastest;
  - TinyJAMBU, Xoodyak, and SCHWAEMM used the least ROM.
- On the STM32F746ZH:
  - TinyJAMBU, ASCON, Xoodyak, and SCHWAEMM were the fastest;
  - ASCON, TinyJAMBU, Xoodyak, and SCHWAEMM used the least ROM;
  - TinyJAMBU, ASCON and SCHWAEMM used the least RAM.
- On the ESP32 WROOM:
  - TinyJAMBU, Xoodyak, and GIFT-COFB were the fastest;
  - Grain-128AEAD, TinyJAMBU, and ASCON were the smallest.
- On the Kendryte K210:
  - ASCON, TinyJAMBU, and Xoodyak were the fastest;
  - ASCON, TinyJAMBU, Xoodyak, and SCHWAEMM used the least ROM.
- Overall, ASCON, GIFT-COFB, SCHWAEMM, TinyJAMBU, and Xoodyak stood out as top performers.

*Weatherley*

Weatherley (2021) performed time measurements on optimized implementations of second-round candidates comparing the AEAD variants to ChaChaPoly Nir and Langley (2015) and the hashing variants to BLAKE2s Aumasson et al. (2013).

The platforms used by this benchmark were:

| Microcontroller | Processor | Word size |
|---|---|---|
| ATmega2560 | AVR | 8-bit |
| AT91SAM3X8E | ARM Cortex-M3 | 32-bit |
| ESP32 | Tensilica Xtensa LX6 | 32-bit |

Table 6: Platforms used in Weatherley

The results of this benchmark were summarized in Turan et al. (2021), these are those results with a focus on the ten finalists:

- For the 8-bit platform:
  - For AEAD:
    - SPARKLE, GIFT-COFB, ASCON, and TinyJAMBU were the top AEAD performers, with their performance being more than twice that of ChaChaPoly;
    - Xoodyak, Romulus, and PHOTON-Beetle are faster than ChaChaPoly by a factor less than two;
    - Grain-128AEAD, Elephant, and ISAP are slower than ChaChaPoly.
  - For Hashing:
    - SPARKLE and Xoodyak performed hashing faster than BLAKE2s;
    - ASCON and PHOTON-Beetle were slower than BLAKE2s.
- For the 32-bit platforms:
  - For AEAD algorithms, SPARKLE, Xoodyak, ASCON, and TinyJAMBU were faster than ChaChaPoly;
  - No hashing algorithm was faster than BLAKE2s.

*Other Initiatives*

Turan et al. (2021) outlines other benchmarking initiatives taken into account for the selection of the finalists.

Campos et al. (2020) studies the effectiveness of different optimization methods on six of the second-round candidates when implemented on RISC-V platforms. Their aim was not to compare the performance of the candidates, but, to compare different implementation optimization strategies.

Nisanci et al. (2019) measures the code size and execution time of the primary AEAD variants of the second-round candidates on RISC-V platforms, in both Linux and Windows, and using different optimization flags.

Bernstein and Lange is a repository of software performance benchmarking results on conventional processors. Results from both eBAEAD (ECRYPT Benchmarking of Authenticated Ciphers) and eBASH (ECRYPT Benchmarking of All Submitted Hashes) were considered for server and hub devices. In eBAEAD, ASCON and Xoodyak consistently outperform other candidates on 64-bit platforms when considering all tested input lengths Turan et al. (2021). In eBASH, ASCON and Xoodyak implementations performed best overall, followed by SPARKLE Turan et al. (2021).

Santos and Großschädl (2020) measured the code size and execution time of assembler implementations of the permutations of ASCON, GIMLI, SCHWAEMM (SPARKLE), and XOODYAK (XOODOO).

Fotovvat et al. (2021) measured the power consumption, random access memory usage, and execution time metrics of the second-round candidates on IOT platforms (Raspberry Pi 3, Raspberry Pi Zero W, and iMX233).

*Low-latency*

Low-latency cryptography is a recent paradigm of LWC. In the fifth Lightweight Cryptography Workshop by NIST there were two presentations, one by Intel Corporation Ghosh (2022) and one by Google Yalcin and Ghandali (2022). In these presentations, the importance of low-latency in the candidates was outlined, something that has been partially ignored in most benchmarking. Out of the ten finalists, ASCON and Xoodyak stood out as the most attractive options for low-latency.

### 2.5.2 *Hardware Benchmarks*

Hardware benchmarks measure different metrics on small devices, such as field-programmable gate arrays or application-specific integrated circuits, that are customized/assembled specifically by the teams performing these benchmarks to run the different cryptographic algorithms.

Below are the three hardware benchmarking initiatives of the NIST-LWC candidates outlined in Turan et al. (2021).

*Mohajerani et al.*

The George Mason University Cryptographic Engineering Research Group Mohajerani et al. (2021) studied the implementations of 27 second-round candidates and measured their throughput, energy per bit, maximum clock frequency, and resource utilization. This benchmarking was done using their tool suite: ATHENa ATHENa, Minerva Farahmand et al. (2017), and Xeda Mohajerani and Nagpal (2020).

Below are the results outlined in Turan et al. (2021), with a focus on the finalists:

- On the Xilinx Artix-7:
  - ASCON, Xoodyak, GIFT-COFB, Elephant, TinyJAMBU, and Romulus had higher throughput than AES-GCM;
  - TinyJAMBU and Romulus had very compact implementations that were under 1000 Look Up Tables;

  - Xoodyak and ASCON performed hashing faster than SHA-2;
  - Xoodyak was the finalist with the smallest implementation supporting hashing;
  - SPARKLE had the only implementation that exceeded the resource limit.

- On the Intel Cyclone 10 LP:
  - ASCON, Xoodyak, GIFT-COFB, Elephant, TinyJAMBU, Romulus, and PHOTON-Beetle had higher throughput than AES-GCM;
  - TinyJAMBU and Romulus were the finalists with the smallest implementations;
  - Xoodyak performed hashing faster than SHA-2;
  - Xoodyak was the finalist with the smallest implementation supporting hashing.

- On the Lattice ECP5:
  - Xoodyak, ASCON, GIFT-COFB, Elephant, and TinyJAMBU had higher throughput than AES-GCM;
  - Xoodyak performed hashing faster than SHA-2;
  - Xoodyak was the finalist with the smallest implementation supporting hashing.

*Khairallah et al. (2020)*

Khairallah et al. synthesized ten of the second-round candidates on 65nm and 28nm technologies and performed ASIC benchmarking on them focusing on two use cases. The first was performance efficiency where the throughput per area was the main concern. The second was lightweight protocols with Bluetooth and Bluetooth Low-Energy representing such protocols. The implementations were synthesized with the following optimization goals: balanced, low-area, high-speed, and low-frequency.

Below are the results outlined in Turan et al. (2021), with a focus on the finalists:

- When optimizing the implementations for low area TinyJAMBU and Romulus had the smallest implementations and strong performance;
- When area was limited to 9000 GE, TinyJAMBU, Romulus and Xoodyak displayed the best results.

*Aagaard and Zidaric (2021)*

Aagaard and Zidaric synthesized 22 of the second-round candidates, as well as two implementations of AES-GCM, on 65nm, 90nm, and 130nm technologies and performed ASIC benchmarking studying their throughput to area, throughput to energy and throughput to area$\times$energy.

Below are the results outlined in Turan et al. (2021), with a focus on the finalists:

- TinyJAMBU had the smallest footprint followed by Romulus, however, they both had relatively poor performance;

- The finalists with the most energy-efficient implementations were Xoodyak and ASCON;
- TinyJAMBU was very energy efficient at lower throughputs;
- TinyJAMBU was very efficient when considering area $\times$ energy;
- Xoodyak and ASCON had low area $\times$ energy with good throughput, followed by Romulus.

# Part II

# CORE OF THE DISSERTATION

# 3

# ASCON

## 3.1 INTRODUCTION

ASCON Dobraunig et al. (2021b) is a sponge-based cipher suite with AEAD and hashing functionalities. Besides being one of the finalists of the NIST-LWC, ASCON was also the primary choice for lightweight AEAD in the CAESAR competition CAESAR Committee. Thanks to this, ASCON is one of the NIST-LWC candidates with the most security analysis and performance benchmarking to its name.

## 3.2 ASCON FAMILY

### 3.2.1 *AEAD*

The AEAD mode of operation of ASCON consists of two functions, authenticated encryption and verified decryption.

The authenticated encryption function takes as input a key $K$, a nonce $N$, the AD $A$ and a plaintext $P$ and produces a pair consisting of a ciphertext $C$ and a tag $T$. The encryption function also has four parameters, the key length $k$, the rate, or data block size, $r$ and the internal round numbers $a$ and $b$. These parameters differ between differ variants. The encryption function can be defined as:
$\mathcal{E}_{k,r,a,b}(K, N, A, P) = (C, T)$.

The verified decryption function takes as input a key $K$, a nonce $N$, the AD $A$, the ciphertext $C$ and the tag $T$ and outputs a plaintext $P$ or an error $\perp$ depending on the success of the verification. The decryption function has the same parameters as the encryption function. The decryption function can be defined as:
$\mathcal{D}_{k,r,a,b}(K, N, A, C, T) \in (P, \perp)$.

The mode of operation used for AEAD is based on duplex modes like MonkeyDuplex Bertoni et al. (2012), but with double keyed initialization and keyed finalization functions. Being based on a duplex structure removes the need for key scheduling or an inverse permutation for decryption, saving space in memory and in code.

The underlying primitive in use is the permutation ASCON$p$, for initialization and finalization $p^a$ is used, and for processing the AD and the plaintext $p^b$ is used. These two permutations $p^a$ and $p^b$ differ only in the number of internal rounds, $a$ and $b$.

The AEAD family of ASCON consists of three members, ASCON-128, ASCON-128a and ASCON-80pq. ASCON-128 is the primary variant of ASCON, with a key length of 128 bits, a nonce length of 128 bits, a tag length of 128 bits, a rate of 64 bits, and $a$ and $b$ equal to 12 and 6 respectively. ASCON-128a, compared to ASCON-128 has an increased rate of 128 bits and two more rounds $b$ to compensate. This increased rate grants ASCON-128a an advantage in performance, when compared to ASCON-128, at the cost of robustness. ASCON-80pq uses the same parameters as ASCON-128 save for the key length which is increased to 160 bits. This increase in key size gives ASCON-80pq quantum resistance against adversaries using Grover's algorithm for key search Grover (1996).

### 3.2.2 *Hashing*

The family of hashing and *eXtendable output functions* (or XOF) can be defined as a single extendable output function.

This function is parameterized by the rate $r$, the round numbers $a$ and $b$, and an output length limit $h$. The function takes as input a message $M$, a number $l$, and outputs a digest $H$ of length $l \leq h$. This extendable output function can be written as:
$\mathcal{X}_{h,r,a,b}(M, l) = H$.

The mode of operation used for hashing is based on sponges Bertoni et al. (2007), saving space in memory with a fixed 320-bit long internal state and allowing the use of the same permutation used in AEAD, saving space in code and memory Dobraunig et al. (2021b).

Like in AEAD, the permutation in use is ASCON$p$, with $p^a$ being used in the initialization step and $p^b$ in the absorbing and squeezing phases of the process.

The family of hashing and extendable output functions in the NIST-LWC submission consists of four members, ASCON-HASH and ASCON-HASHA, and ASCON-XOF and ASCON-XOFA. ASCON-HASH, the primary variant, has $h$ equal to 256 bits, $r$ equal to 64 bits, and 12 rounds for $p^a$ and $p^b$. ASCON-HASHA when compared to ASCON-HASH has 8 rounds for $p^b$. The XOF are the same as their hashing counterparts but $h$ equal to 0, granting it unlimited output length.

The designers recommend pairing the primary variants (ASCON-128 and ASCON-HASH) as they have the same rate or the secondary variants (ASCON-128a and ASCON-HASHA) as they have the same number of rounds for $p^b$.

### 3.2.3 *ASCONp*

The ASCON$p$ permutation is a 320-bit long SPN-based round transformation. The permutation ($p$) consists in three steps $p_C$, $p_S$ and $p_L$ with:

$p = p_L \circ p_S \circ p_C$.

The input to this permutation, the state of the duplex or sponge, $S$ is split into five 64-bit words $x_i$ with $S = x_0||x_1||x_2||x_3||x_4$.

The first step, $p_C$ adds a round constant to the state by XORing a round constant with $x_2$.

The second step, $p_S$ is the substitution layer, applying a S-box to each bit-slice of the five 64-bit words in $S$.

The final step, $p_L$ is the linear layer, applying a different linear function to each of the five 64-bit words in $S$.

## 3.3 SECURITY CLAIMS

### *AEAD*

The AEAD mode of operation of ASCON is a duplex structure which means that it benefits from extensive the analysis and security proofs done in the past years such as Bertoni et al. (2011b), Jovanovic et al. (2014), Andreeva et al. (2015) and Daemen et al. (2017). These duplex structures can provide security beyond the birthday bound on their capacity $c$, as long as the online data complexity remains below the birthday bound of $2^{c/2}$ Bertoni et al. (2011b) Jovanovic et al. (2014).

ASCON uses a double keyed initialization and finalization, making the task of recovering a key or producing a forgery in the event of an internal state recovery much more difficult Dobraunig et al. (2021b).

All three variants have 128 security bits for the confidentiality and integrity of plaintexts, integrity of associated data and integrity of public message number in a nonce-respecting setting, assuming the data complexity is below the birthday bound.

### *Hashing*

Much like with authenticated encryption, the sponge-based hashing mode of operation benefits from extensive analysis and security proofs, such as Bertoni et al. (2008) and Lefevre and Mennink (2022).

The two hash variants have 128 bits of security for collision-resistance and (second) preimage attacks. The XOFs have the least bits of security between 128 and $l/2$ for collision-resistance and the least bits of security between 128 and $l$ for (second) preimage resistance.

## 3.4 CRYPTANALYSIS

Below is a description of the different ways and settings in which ASCON is analyzed and a summary of the best known analysis and a look at the different parts of ASCON being analyzed with an emphasis on analysis following Turan et al. (2021).

### 3.4.1 *Analyzing ASCON*

Analysis done on ASCON targets either the AEAD or hashing modes of operation or the permutation.

When targeting the AEAD mode of operation there are three attack goals:

- *Key recovery*: these attacks target the initialization;
- *Forgery*: these attacks target the finalization;
- *State recovery*: these attacks target the iterations. A successful state recovery attack can be extended to a key recovery attack with complexity based on the capacity $c$.

When targeting the hashing mode of operation there are three attack targets, (second) preimage and collision resistance. All attacks that target the permutation are of a distinguishing nature.

*Best Known Analysis*

This is the current best known analysis of ASCON Eichlseder et al. (2022).

| Type | Target | Rounds | Time | Notes | Method | Reference |
|---|---|---|---|---|---|---|
| Key Recovery | ASCON initialization | 7/12 | $2^{97}$ | 1 | Cube-like | Li et al. (2017a) |
| | ASCON initialization | 7/12 | $2^{104}$ | 2 | Cube-like | Li et al. (2017b) |
| | ASCON initialization | 7/12 | $2^{123}$ | – | Cube | Rohit et al. (2021) |
| | ASCON initialization | 6/12 | $2^{74}$ | 2 | Cond. HDL | Hu and Peyrin (2022) |
| | ASCON initialization | 5/12 | $2^{31}$ | – | DL | Tezcan (2020) |
| | ASCON-128a iteration | 7/8 | $2^{118}$ | 1 2 | Cond. cube | Chang et al. (2022b) |
| | ASCON-80pq iteration | 6/6 | $2^{130}$ | 1 3 | Cond. cube | Chang et al. (2022a) |
| Forgery | ASCON-128 finalization | 6/12 | $2^{33}$ | 1 | Cube tester | Li et al. (2017a) |
| | ASCON-128 finalization | 4/12 | $2^{102}$ | 2 | Differential | Dobraunig et al. (2015b) |
| | ASCON-128 finalization | 3/12 | $2^{97}$ | 2 | Differential | Gerault et al. (2021) |
| | ASCON-128 finalization | 3/12 | $2^{20}$ | | Differential | Gerault et al. (2021) |
| State Recovery | ASCON-128 iteration | 6/6 | $2^{40}$ | 1 | Cond. cube | Baudrin et al. (2022) |
| | ASCON-128 iteration | 6/6 | $2^{45}$ | 1 | Cond. cube | Chang et al. (2022a) |
| | ASCON-128 iteration | 5/6 | $2^{66}$ | 1 | Cube-like | Li et al. (2017a) |
| | ASCON-128a iteration | 7/8 | $2^{118}$ | 1 2 | Cond. cube | Chang et al. (2022b) |
| | ASCON-128a iteration | 3/8 | $2^{117}$ | – | Differential | Gerault et al. (2021) |
| | ASCON-128a iteration | 2/8 | – | – | Sat-Solver | Dwivedi et al. (2016) |

Table 7: Best known analysis on ASCON AEAD modes Eichlseder et al. (2022).
1 - Nonce misuse, 2 - exceeds data limit of $2^{64}$ blocks, 3 - time exceeds $2^{128}$
weak-key variants omitted.

| Type | Target | Output Size | Rounds | Time | Notes | Method | Reference |
|---|---|---|---|---|---|---|---|
| Preimage | ASCON-XOF | 64 | 6/12 | $2^{63.3}$ | – | Algebraic | Dobraunig et al. (2019) |
| | ASCON-XOF | 64 | 2/12 | $2^{39}$ | – | Cube-like | Dobraunig et al. (2019) |
| Collision | ASCON-XOF | all | 4/12 | – | 4 | Differential | Dobraunig et al. (2019) |
| | ASCON-XOF | 64 | 2/12 | $2^{15}$ | – | Differential | Zong et al. (2019) |
| | ASCON-HASH | 256 | 2/12 | $2^{125}$ | – | Differential | Zong et al. (2019) |
| | ASCON-HASH | 256 | 2/12 | $2^{103}$ | – | Differential | Gerault et al. (2021) |

Table 8: Best known analysis on ASCON hashing modes Eichlseder et al. (2022).
4 - Chosen IV.

| Type | Target | Rounds | Time | Notes | Method | Reference |
|---|---|---|---|---|---|---|
| Distinguisher | Permutation | 12/12 | $2^{55}$ | 5 | Zero-sum | Hu and Peyrin (2022) |
| | Permutation | 11/12 | $2^{85}$ | 5 | Zero-sum | Dobraunig et al. (2021b) |
| | Permutation | 8/12 | $2^{46}$ | – | Integral | Hu and Peyrin (2022) |
| | Permutation | 7/12 | $2^{65}$ | – | Integral | Todo (2015) |
| | Permutation | 7/12 | $2^{60}$ | – | Integral | Rohit et al. (2021) |
| | Permutation | 7/12 | $2^{34}$ | 5 | Limited-Birthday | Gerault et al. (2021) |
| | Permutation | 5/12 | $2^{109}$ | – | Truncated Differential | Tezcan (2016) |
| | Permutation | 5/12 | $2^{80}$ | – | Rectangle | Gerault et al. (2021) |
| | Permutation | 5/12 | – | – | Zero-Correlation | Dobraunig et al. (2021b) |
| | Permutation | 5/12 | – | – | Impossible Differential | Dobraunig et al. (2021b) |
| | Permutation | 4/12 | $2^{107}$ | – | Differential | Dobraunig et al. (2021b) |
| | Permutation | 4/12 | $2^{101}$ | – | Linear | Dobraunig et al. (2015b) |
| | Permutation | 3/12 | – | – | Subspace Trails | Leander et al. (2018) |

Table 9: Best known analysis on ASCON permutation Eichlseder et al. (2022).
5 - non-black-box distinguisher.

### 3.4.2 *AEAD*

The currently best known key recovery attacks on the authenticated encryption of ASCON target only 7/12 rounds, giving it a security margin of 42%. The best attack in a standard setting takes about $2^{123}$ time Rohit et al. (2021) while the best attack in a nonce-misuse setting takes $2^{97}$ time Li et al. (2017a).

With a stagnation of advancements in the cryptanalysis of standard settings, research has moved towards misuse settings such as, nonce misuse, decryption misuse or implementation attacks Eichlseder et al. (2022). Nonce-misuse scenarios are not as rare as one would wish in lightweight cryptography since counter desynchronization is a factor to keep in mind in highly-constrained devices.

Both Baudrin et al. (2022) and Chang et al. (2022b) feature state recovery attacks on ASCON-128 (and on ASCON-80pq in Chang et al. (2022b)) for 6/6 rounds of the internal permutation in a nonce-misuse scenario. While these attacks do not go against the claims made by the designers in Dobraunig et al. (2021b), since they

cannot be extended to a key recovery attack in under $2^{128}$ time, they reinforce the idea that implementation errors cannot be ruled out.

### 3.4.3 *Hashing*

The currently best known attack targeting preimage resistance of the hashing mode of operation targets 6/12 rounds of the extended output function for 64-bit long outputs with a time of $2^{63.3}$ Dobraunig et al. (2019). The currently best known attack targeting the collision resistance of ASCON-HASH targets 2/12 rounds with a time of $2^{103}$ Gerault et al. (2021).

### 3.4.4 *Permutation*

When selecting the substitution and diffusion layers for the permutation, the designers had to balance security and performance. For example, more secure options for the S-box were available at the expense of performance, so, the designers went for the approach of having a non-ideal and cheaper S-box repeated for more rounds instead of a perfect but expensive S-box repeated for less rounds Dobraunig et al. (2021b).

#### *Differential and Linear Properties*

The design choices made in regards to the permutation aim not for ideal individual differential or linear properties but for good combined properties.

Following Turan et al. (2021) there have been advancements on the provable differential and linear bounds, the number of active S-boxes in differential characteristics, known characteristics, and differential-linear and higher-order differential-linear (HDL) attacks.

Gerault et al. (2021) presents a methodology that uses Constraint Programming (or CP) Gerault et al. (2016) to automatically find differential characteristics and applied it to ASCON. They also propose a split in the definition of distinguishers in two types, black-box and non-black box distinguishers, referring to how they treat the permutation (for example, keyed permutation distinguishers have to treat the permutation as a black-box). Lastly, they used the newly found characteristics in the rectangle and limited-birthday attacks presented in Table 9.

In Makarim and Rohit (2022) the authors use a hybrid approach of Satisfiability Modulo Theories (or SMT) Ganesh and Dill (2007) and Mixed Integer Linear Programming (or MILP) Mouha et al. (2011) to find new differential and linear characteristics, and improving the upper bounds of differentially active S-boxes for 4 and 5 rounds.

In Erlacher et al. (2022) the authors use SAT solvers to prove the lower bounds for 4 and 6 rounds of both differentially and linearly active S-boxes.

In Hu and Peyrin (2022) the authors use higher-order differential and HDL distinguishers to improve existing attacks targeting 7 and 8 rounds of ASCON in a black-box setting and 12 rounds in a non-black-box setting.

| N | #S | $p$ | Reference | Method |
|---|---|---|---|---|
| 1 | 1 | $2^{-2}$ | | DDT |
| 2 | 4 | $2^{-8}$ | | DDT, $\mathcal{B}$ |
| 3 | 15 | $\leq 2^{-30}$ | Dobraunig et al. (2015b) | SMT |
| 4 | $\geq 36$ | $\leq 2^{-72}$ | Erlacher et al. (2022) | SAT |
| 5 | – | | | |
| 6 | $\geq 54$ | $\leq 2^{-108}$ | Erlacher et al. (2022) | SAT |

Table 10: Provable bounds for differential cryptanalysis of $N$-round ASCON permutation with the minimum active S-boxes (#S) and maximum probability ($p$).
"$\geq, \leq$" indicates not necessarily tight bounds without a matching characteristic. $\mathcal{B}$ = Branch number of the linear layer. Erlacher et al. (2022)

| N | #S | $p$ | Reference | Method |
|---|---|---|---|---|
| 1 | 1 | $2^{-2}$ | | DDT |
| 2 | 4 | $2^{-8}$ | | DDT, $\mathcal{B}$ |
| 3 | 15 | $2^{-40}$ | Dobraunig et al. (2015b) | nldtool |
| 4 | 44 | $2^{-107}$ | Dobraunig et al. (2015b) | nldtool |
| | 43 | – | Makarim and Rohit (2022) | SMT & MILP |
| 5 | 78 | $2^{-190}$ | Dobraunig et al. (2015b), Gerault et al. (2021) | CP |
| | 72 | – | Makarim and Rohit (2022) | SMT & MILP |

Table 11: Best known characteristics for differential cryptanalysis of $N$-round ASCON permutation with the minimum active S-boxes (#S) and/or maximum probability ($p$).
$\mathcal{B}$ = Branch number of the linear layer. nldtool is a dedicated guess-and-determine tool for differential cryptanalysis Mendel et al. (2011). Erlacher et al. (2022)

| N | #S | $c^2$ | Reference | Method |
|---|---|---|---|---|
| 1 | 1 | $2^{-2}$ | | LAT |
| 2 | 4 | $2^{-8}$ | | LAT, $\mathcal{B}$ |
| 3 | 15 | $\leq 2^{-26}$ | Dobraunig et al. (2015b) | SMT |
| 4 | $\geq 36$ | $\leq 2^{-72}$ | Erlacher et al. (2022) | SAT |
| 5 | – | | | |
| 6 | $\geq 54$ | $\leq 2^{-108}$ | Erlacher et al. (2022) | SAT |

Table 12: Provable bounds for linear cryptanalysis of $N$-round ASCON permutation with the minimum active S-boxes (#S) and maximum squared correlation ($c^2$).
"$\geq, \leq$" indicates not necessarily tight bounds without a matching characteristic. $\mathcal{B}$ = Branch number of the linear layer. Erlacher et al. (2022)

| N | #S | $p$ | Reference | Method |
|---|---|---|---|---|
| 1 | 1 | $2^{-2}$ | | LAT |
| 2 | 4 | $2^{-8}$ | | LAT, $\mathcal{B}$ |
| 3 | 13 | $2^{-28}$ | Dobraunig et al. (2015b) | lineartrails |
| 4 | 43 | $2^{-98}$ | Dobraunig et al. (2015b) | lineartrails |
| 5 | 67 | $2^{-186}$ | Dobraunig et al. (2015b) | lineartrails |
| | 78 | $2^{-184}$ | Makarim and Rohit (2022) | SMT & MILP |

Table 13: Best known characteristics for linear cryptanalysis of $N$-round ASCON permutation with the minimum active S-boxes (#S) and/or maximum probability ($p$).
$\mathcal{B}$ = Branch number of the linear layer. *lineartrails* a heuristic search tool for finding linear characteristics Dobraunig et al. (2015a). Erlacher et al. (2022)

*Algebraic Properties*

The S-box used in the permutation has an algebraic degree of 2, making the permutation more susceptible to algebraic-based attacks. This was done in order to allow efficient implementation of side-channel countermeasures such as threshold implementation and masking Dobraunig et al. (2021b).

Recent attacks that exploit the algebraic properties of the permutation and represent an improvement on preexisting attacks are, for example, Rohit et al. (2021), Baudrin et al. (2022).

Both Rohit et al. (2021) and Baudrin et al. (2022) employ cube and cube-like attacks, attacks that exploit the low algebraic degree of the S-box in use.

# 4

# ROMULUS

## 4.1 INTRODUCTION

Romulus Guo et al. (2021) is a TBC-based cipher suite with AEAD and hashing functionalities. Romulus uses Skinny Beierle et al. (2016) as its underlying primitive. Both the structures used in the different Romulus variants and the underlying primitive are well studied, with the latter being a participant of the CAESAR competition CAESAR Committee.

## 4.2 ROMULUS FAMILY

### 4.2.1 *AEAD*

The AEAD modes of operation consist of two functions, authenticated encryption and verified decryption. These functions can be defined in the same way as in ASCON, with the only difference being the parameters. These parameters are: the nonce length of 128 bits, or $nl$, the message block length of 128 bits, or $n$, the key length of 128 bits, or $k$, the counter length of 56 bits, or $d$, and the tag length of 128 bits, or $\tau$.

The Romulus family of AEAD variants has three members, Romulus-N, Romulus-M, and Romulus-T, all using the same underlying primitive, Skinny.

#### *Romulus-N*

Romulus-N, the primary AEAD variant of Romulus, shares a similar structure to iCOFB Chakraborti et al. (2017) with significant improvements in order to improve performance and security.

#### *Romulus-M*

Romulus-M is a nonce-misuse resistant AE mode of operation following the structure of Synthetic IV Rogaway and Shrimpton (2006) and Counter-in-tweak Peyrin and Seurin (2016). Romulus-M reuses components from Romulus-N, inheriting its implementation advantages and security.

*Romulus-T*

Romulus-T is a leakage-resilient mode of operation following the structure of TEDT Berti et al. (2019), sharing similar proofs and security bounds. Romulus-T is designed in order to limit the extension of side-channel attacks.

### 4.2.2 *Hashing*

*Romulus-H*

The Romulus family has one hashing mode of operation variant, Romulus-H. This variant can be defined as a hashing function in the same way as ASCON, differing in the parameters. The parameters of Romulus-H are: the message block length of 256 bits, or $2n$, and the hash value length, or digest size, of 256 bits, or $2\tau$.

Romulus-H is based on the MDPH Naito (2019) construction, using Skinny as the underlying block cipher. Romulus-H can provide XOF functionality Guo et al. (2022a).

### 4.2.3 *Skinny*

The version of Skinny used in Romulus is Skinny-128-384+, a 40-round version of Skinny-128-384. Skinny follows the TWEAKEY framework Jean et al. (2014), meaning, the TBC takes a single tweakey input as opposed to key or a key/tweak pair.

Skinny consists of a preliminary initialization phase and a round function. In the initialization phase the internal state, a 4x4 byte matrix, is initialized with the original message block. Following this initialization, a round function is applied, 40 times in the case of Skinny-128-384+ case. The round function consists of five steps:

- *SubCells* (SC), where a S-box is applied to each byte in the internal state;
- *AddConstants* (AC), where a round-dependent constant is added to the internal state;
- *AddRoundTweakey* (ART), where a round tweakey is applied to the internal state;
- *ShiftRows* (SR), where the rows of the internal state matrix are shifted depending on their index;
- *MixColumns* (MC), where each column of the internal state matrix is multiplied by a different matrix.

## 4.3 SECURITY CLAIMS

### 4.3.1 *AEAD*

The claims for the AEAD modes of operation are made with two models of adversaries, nonce-respecting (NR) and nonce-misusing (NM), for both privacy and authenticity. The results are summarized as follows:

| Member | NR-Priv | NR-Auth | NM-Priv | NM-Auth |
|---|---|---|---|---|
| Romulus-N | $n$ | $n$ | – | – |
| Romulus-M | $n$ | $n$ | $n/2 \sim n$ | $n/2 \sim n$ |
| Romulus-T | $n - log_2 n$ | $n - log_2 n$ | – | $n - log_2 n$ |

Table 14: Security claims of Romulus-N, Romulus-M and Romulus-T Guo et al. (2021).

The claims made regarding Romulus-N are beyond birthday bound for the nonce-respecting scenario. Furthermore, Romulus-N offers perfect security for privacy in nonce-misuse resilience and $n/2$ bits of security for authenticity in nonce-misuse resilience Inoue et al. (2022).

The claims made regarding Romulus-M are beyond birthday bound for the nonce-respecting and up to birthday bound for the nonce-misuse scenarios. Furthermore, Romulus-M retains authenticity under RUP Iwata et al. (2020).

Romulus-T offers 121-bit security for a nonce-respecting adversary and 121 bits of authenticity for a nonce-misusing adversary, giving Romulus-T nonce-misuse resistance. Furthermore, Romulus-T offers beyond birthday bound CIML2 and CCAmL2.

In Lee (2022) reviews Romulus-N and Romulus-M using a different approach from Guo et al. (2021) and arrives at the same conclusions.

The claims made for Romulus-N and Romulus-M hold if Skinny remains a tweakable pseudorandom permutation, that is, it is computationally hard to to distinguish it from the set of uniform random permutations indexed by the tweak, using chosen-plaintext queries in the single-key setting Guo et al. (2021). Similarly, Romulus-T holds if it is hard to distinguish Skinny from a random block cipher. This means that analysis done on any of these variants is essentially reduced to analyzing Skinny.

### 4.3.2 *Hashing*

The security of MDPH, and therefore Romulus-H, was proved in Naito (2019). The security claims are as follows:

| Member | Collision | Preimage | Second Preimage |
|---|---|---|---|
| Romulus-H | $n - log_2 n$ | $n - log_2 n$ | $n - log_2 n$ |

Table 15: Security claims of Romulus-H Guo et al. (2021).

In Guo et al. (2022b) the authors review the original proof from Naito (2019) patching a gap in the proof, retaining the same security bounds.

## 4.4 CRYPTANALYSIS

Since the analysis of the modes of operation of Romulus can be reduced to the analysis of Skinny, this section is dedicated to the best and most recent known results on Skinny-128-334.

| Rounds | Time | Data | Notes | Method | Reference |
|---|---|---|---|---|---|
| 30/56 | $2^{341}$ | $2^{122}$ | 1 | Rectangle | Qin et al. (2021) |
| 30/56 | $2^{361}$ | $2^{125}$ | 1 | Rectangle | Hadipour et al. (2020) |
| 25/56 | $2^{226}$ | $2^{124}$ | 1,2 | Rectangle | Qin et al. (2021) |
| 24/56 | $2^{209}$ | $2^{125}$ | 1,2 | Rectangle | Hadipour et al. (2020) |

Table 16: Best known key recovery attacks on Skinny-128-384.
1 - Related Key, 2 - TK2 Model

| Rounds | Probability | Notes | Method | Reference |
|---|---|---|---|---|
| 25/56 | $2^{-116.6}$ | – | Boomerang | Hadipour et al. (2020) |
| 24/56 | $2^{-86}$ | – | Boomerang | Delaune et al. (2021) |
| 22/56 | $2^{-101.5}$ | – | Boomerang | Qin et al. (2021) |
| 21/56 | $2^{-114}$ | 1 | Boomerang | Hadipour et al. (2020) |
| 20/56 | $2^{-96}$ | 1 | Boomerang | Delaune et al. (2021) |
| 19/56 | $2^{-86}$ | 1 | Boomerang | Qin et al. (2021) |

Table 17: Best known distinguishers for Skinny-128-384.
1 - TK2 Model

The best known attacks and distinguishers for Skinny-128-384 all use techniques based on differential cryptanalysis, modified boomerang Wagner (1999) and rectangle attacks Biham et al. (2001).

# 5

# ASCON VS ROMULUS

## 5.1 INTRODUCTION

Both ASCON and Romulus are among the top contenders of the NIST-LWC, with each having features that distinguish them from most or even all of the other finalists. Before going over what differentiates them I want to mention a few finalists that also distinguish themselves from the rest: Elephant is the only finalist with parallel functionalities, GIFT-COFB, TinyJAMBU, and Xoodyak are among the top performers in speed, and TinyJAMBU and Xoodyak are among the top performers in code size.

## 5.2 SECURITY CLAIMS

Where it comes to security both Romulus-N and Romulus-M hold the advantage of being provably secure in the standard model while ASCON is provably secure when respecting the data limit of $2^{64}$. Regardless of this fact, the security claims made by both teams still hold with the security margins of Romulus and ASCON being $> 50\%$ and $\sim 42\%$ respectively against the currently best known attacks.

While Romulus relies mainly on the effectiveness of its underlying primitive to keep this security margin, ASCON relies not only on the effectiveness of its underlying primitive but on the difficulty of a key recovering attack following an internal state recovery as well, thanks to its double-keyed initialization and finalization.

## 5.3 ADDITIONAL FEATURES

### 5.3.1 *Competitions and Standards*

ASCON benefits greatly from participating in CAESAR Committee with an extensive security analysis verifying the claims laid by the designers. Romulus also benefits from an extensive security analysis of SKINNY, granted not as much as with ASCON, with the primitive being part of ISO 18033-7:2022.

### 5.3.2 *Nonce-misuse*

Where it comes to nonce-misuse resistance, Romulus-M is the only finalist variant that provides strong nonce-misuse resistance. Romulus-T (and Elephant) provides nonce-misuse resilience. No other finalists offer protection in the event of nonce-misuse.

### 5.3.3 *Side-channel*

Romulus-T (and ISAP) provides the strongest protection against side-channel attacks with CIML2 and CCAmL2 security. ASCON follows closely with CIML2 and CCAmL1 security. No other finalists provide a similar degree of side-channel resistance.

### 5.3.4 *Release of Unverified Plaintext*

Romulus-M and Romulus-T (along with Elephant) are the only finalist variants that provide protection in the event of the release of unverified plaintext.

## 5.4 PERFORMANCE

### 5.4.1 *In Software*

Overall, ASCON outperformed Romulus in software benchmarks of the primary AEAD variants, with Romulus having a smaller code size in some cases. When most benchmarking was done Romulus still used the 56-round SKINNY-128-384 instead of the current 40-round SKINNY-128-384+ so there is an argument for the difference in performance being smaller than initially thought.

*Low-latency Cryptography*

Like mentioned before, Low-latency cryptography is an emerging field in LWC. In both presentations from the fifth Lightweight Cryptography Workshop, Ghosh (2022) and Yalcin and Ghandali (2022), ASCON stood out with the overall lowest latency among the finalists.

### 5.4.2 *In Hardware*

Overall, ASCON outperformed Romulus in hardware benchmarks of the primary AEAD variants in terms of throughput, while Romulus outperformed ASCON in terms of implementation size. Like before, it remains to be seen if the optimizations to both families change the outcomes of future benchmarking.

## 5.5 CONCLUSIONS

The Romulus suite ticks all the boxes of additional features, be it nonce-misuse resistance, RUP security or side-channel protection, granted that the user is willing to pay the additional cost of Romulus-M or Romulus-T.

ASCON follows the *one-size-fits-all* philosophy, with the primary variant offering almost everything that Romulus-N offers at a smaller cost. Additionally, ASCON offers nearly the same side-channel protection as Romulus-T with no added cost. On top of all that, at the cost of robustness, ASCON-128a, the secondary variant of AEAD, offers an increase in performance.

In my opinion the debate between ASCON and Romulus boils down to one point: how much do we value the additional features provided by the Romulus suite over the performance provided by ASCON.

If the additional decryption leakage protection in a side-channel scenario is seen as worth the additional cost of Romulus-T (or ISAP) then, yes, Romulus is better suited for side-channel protection.

If the nonce-misuse resistance/resilience and the RUP security of both Romulus-M and Romulus-T (and Elephant) is seen as worth their additional cost then, yes, Romulus is a more flexible candidate.

If higher performance, in some cases best in class, comparable security and second best side-channel protection is deemed sufficient, then ASCON is a more suitable candidate for selection.

# 6

# CONCLUSIONS AND FUTURE WORK

## 6.1 CONCLUSIONS

When reading up on the specifications and analysis or benchmarks of the candidates I picked apart what I thought were the most relevant subjects to explain in the state-of-the-art. Starting with the simpler concepts that I learned in my curriculum and moving up to some more advanced concepts that were new to me, I went over published literature and tried to pinpoint their introduction and evolution, summarizing what I thought to be pertinent.

What followed was a proper state-of-the-art of the current state of the NIST-LWC, going over the results from the second phase, and the security analysis and performance benchmarks that were done afterwards, with a focus on ASCON and Romulus.

After this preliminary work it was time to compare the candidates and form an opinion. The Romulus family achieves many objectives such as security, additional features and implementation size when considering the whole cipher suite while ASCON achieves the required security, some of the additional features, and a higher performance, with just one design.

## 6.2 FUTURE WORK

In the state-of-the-art lies the foundation of what would be a differential-linear analysis of one or both of the candidates. Unfortunately, with changes in scope, availability of resources, and relative inexperience, it was not possible but remains something endearing to me. To follow through with this I think that the first step is to find better characteristics, using and adapting what already exists and only then try my hand at using these techniques. Something that I also found very interesting, and have a minor background in, is performance benchmarking, software performance benchmarking in particular. Of course, for this I would need access to resources that are outside my financial grasp.

One last thing I want to mention is that, at a personal level, I found cryptography, like most areas in academia, and cryptanalysis in specific, to be an area with a high entry level of difficulty. The descriptions of techniques and concepts are presented in a very formal fashion, requiring vast knowledge outside of most undergraduates', or

people pursuing a masters degree coming from Software Engineering like me, curricula. This, along with Heys (2002), really inspired me to write the state-of-the-art I have written, in order to summarize the subjects one has to learn when entering this field. I find it alluring and intend to keep doing this work and compile all my current and future knowledge on these and adjacent subjects, even if it is only for personal satisfaction and no further gain.

# BIBLIOGRAPHY


Mark D. Aagaard and Nusa Zidaric. *ASIC Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process*. Cryptology ePrint Archive, Paper 2021/049, 2021. URL https://eprint.iacr.org/2021/049. https://eprint.iacr.org/2021/049.

Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. *How to Securely Release Unverified Plaintext in Authenticated Encryption*. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 105–125, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-45611-8.

Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. *Security of Keyed Sponge Constructions Using a Modular Proof Approach*. In Gregor Leander, editor, *Fast Software Encryption*, pages 364–384, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. ISBN 978-3-662-48116-5.

Tomer Ashur, Orr Dunkelman, and Atul Luykx. *Boosting Authenticated Encryption Robustness with Minimal Modifications*. In *Annual International Cryptology Conference*, pages 3–33. Springer, 2017.

ATHENa. *ATHENa: Automated Tools for Hardware EvaluatioN*. URL https://cryptography.gmu.edu/athena/index.php?id=LWC. Project Homepage.

Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. *BLAKE2: Simpler, Smaller, Fast as MD5*. In Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security*, pages 119–135, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-38980-1.

Subhadeep Banik, Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, and Siang Meng Sim Yosuke Todo. *GIFT-COFB v1.1*. Submission to the NIST Lightweight Cryptography Standardization Process, 2021.

Zhenzhen Bao, Avik Chakraborti, Nilanjan Datta, Jian Guo, Mridul Nandi, Thomas Peyrin, and Kan Yasuda. *PHOTON-Beetle Authenticated Encryption and Hash Family*. Submission to the NIST Lightweight Cryptography Standardization Process, 2021.

Achiya Bar-On, Orr Dunkelman, Nathan Keller, and Ariel Weizman. *DLCT: A New Tool for Differential-Linear Cryptanalysis*. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 313–342. Springer, 2019.

Gregory V Bard, Nicolas T Courtois, and Chris Jefferson. *Efficient Methods for Conversion and Solution of Sparse Systems of Low-degree Multivariate Polynomials over GF (2) via SAT-solvers*. *Cryptology ePrint Archive*, 2007.

Jules Baudrin, Anne Canteaut, and Léo Perrin. *Practical Cube-attack Against Nonce-misused ASCON*. FSE 2022 Rump Session, 2022.

Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. *The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS*. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 123–153, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. ISBN 978-3-662-53008-5.

Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Amir Morad, Léo Perrin, Aein Rezaei Shahmirzadi, Aleksei Udovenko, Vesselin Velichkov, and Qingju Wang. *SCHWAEMM and ESCH: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family*. Submission to the NIST Lightweight Cryptography Standardization Process, 2021.

Mihir Bellare and Chanathip Namprempre. *Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm*. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 531–545. Springer, 2000.

Mihir Bellare and Phillip Rogaway. *Optimal Asymmetric Encryption*. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 92–111. Springer, 1994.

Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. *Relations Among Notions of Security for Public-key Encryption Schemes*. In *Annual International Cryptology Conference*, pages 26–45. Springer, 1998.

Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. *Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography: A Practical Guide Through the Leakage-Resistance Jungle*. Cryptology ePrint Archive, Paper 2020/211, 2020. URL https://eprint.iacr.org/2020/211. https://eprint.iacr.org/2020/211.

Daniel J. Bernstein and Tanja Lange. *eBACS: ECRYPT Benchmarking of Cryptographic Systems*. https://bench.cr.yp.to [Accessed: 21 October 2022].

Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. *TEDT, a Leakage-Resist AEAD Mode for High Physical Security Applications*. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 256–320, 11 2019. doi: 10.46586/tches.v2020.i1.256-320.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. *Sponge Functions*. In *ECRYPT Hash Workshop*, volume 2007. Citeseer, 2007.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. *On the Indifferentiability of the Sponge Construction*. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 181–197, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-78967-3.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. *Duplexing the Sponge: Single-pass Authenticated Encryption and Other Applications*. In *International Workshop on Selected Areas in Cryptography*, pages 320–337. Springer, 2011a.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. *On the Security of the Keyed Sponge Construction*. 2011b.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. *Permutation-based Encryption, Authentication and Authenticated Encryption. Directions in Authenticated Ciphers*, pages 159–170, 2012.

Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. *Elephant v2*. Submission to the NIST Lightweight Cryptography Standardization Process, 2021.

Eli Biham and Adi Shamir. *Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology*, 4(1):3–72, 1991.

Eli Biham and Adi Shamir. *Differential cryptanalysis of the Data Encryption Standard*. Springer Science & Business Media, 2012.

Eli Biham, Orr Dunkelman, and Nathan Keller. *The Rectangle Attack — Rectangling the Serpent*. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 340–357. Springer, 2001.

Eli Biham, Orr Dunkelman, and Nathan Keller. *Enhancing Differential-Linear Cryptanalysis*. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 254–266. Springer, 2002.

Céline Blondeau, Gregor Leander, and Kaisa Nyberg. *Differential-Linear Cryptanalysis Revisited. Journal of Cryptology*, 30(3):859–888, 2017.

Andrey Bogdanov and Meiqin Wang. *Zero Correlation Linear Cryptanalysis with Reduced Data Complexity*. In *International Workshop on Fast Software Encryption*, pages 29–48. Springer, 2012.

Bruno Buchberger. *Bruno Buchberger's PhD Thesis 1965: An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal. Journal of Symbolic Computation*, 41(3-4): 475–511, 2006.

CAESAR Committee. *Competition for Authenticated Encryption: Security, Applicability, and Robustness*. https://competitions.cr.yp.to/caesar-submissions.html [Accessed: 1/10/2022].

Fabio Campos, Lars Jellema, Mauk Lemmen, Lars Müller, Daan Sprenkels, and Benoit Viguier. *Assembly or Optimized C for Lightweight Cryptography on RISC-V?* In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *Cryptology and Network Security*, pages 526–545, Cham, 2020. Springer International Publishing. ISBN 978-3-030-65411-5.

Florent Chabaud and Serge Vaudenay. *Links Between Differential and Linear Cryptanalysis*. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 356–365. Springer, 1994.

Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. *Blockcipher-Based Authenticated Encryption: How Small Can We Go?* In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, pages 277–298, Cham, 2017. Springer International Publishing. ISBN 978-3-319-66787-4.

Donghoon Chang, Hong Dekjo, and Jinkeon Kang. *Conditional Cube Attacks on ASCON-128 and ASCON-80pq in a Nonce-misuse Setting*. NIST LWC Workshop 2022, 2022a.

Donghoon Chang, Jinkeon Kang, and Meltem Sönmez Turan. *A New Conditional Cube Attack on Reduced-Round ASCON-128a in a Nonce-misuse Setting*. NIST LWC Workshop 2022, 2022b.

Nicolas Courtois and Gregory Bard. *Algebraic Cryptanalysis of the Data Encryption Standard*. In *IMA International Conference on Cryptography and Coding*, pages 152–169. Springer, 2007.

Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 392–407. Springer, 2000.

Joan Daemen, Bart Mennink, and Gilles Van Assche. *Full-State Keyed Duplex with Built-In Multi-user Support*. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 606–637, Cham, 2017. Springer International Publishing. ISBN 978-3-319-70697-9.

Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. *XOODYAK, a Lightweight Cryptographic Scheme*. Submission to the NIST Lightweight Cryptography Standardization Process, 2021.

Ivan Bjerre Damgård. *A Design Principle for Hash Functions*. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989.

Quynh H Dang et al. *Secure Hash Standard*. 2015.

Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. *Catching the Fastest Boomerangs - Application to SKINNY*. Cryptology ePrint Archive, Paper 2021/020, 2021. URL https://eprint.iacr.org/2021/020.

Hans Delfs, Helmut Knebl, and Helmut Knebl. *Introduction to Cryptography*, volume 2. Springer, 2002.

Whitfield Diffie and Martin E Hellman. *New Directions in Cryptography*. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pages 365–390. 2022.

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. *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, 2015a.

Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. *Cryptanalysis of ASCON*. In *Cryptographers' Track at the RSA Conference*, pages 371–387. Springer, 2015b.

Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. *Preliminary Analysis of ASCON-XOF and ASCON-HASH*. 2019.

Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. *ISAP v2.0*. Submission to the NIST Lightweight Cryptography Standardization Process, 2021a.

Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. *ASCON v1.2: Lightweight Authenticated Encryption and Hashing*. *Journal of Cryptology*, 34, 2021b. doi: 10.1007/s00145-021-09398-9.

Danny Dolev, Cynthia Dwork, and Moni Naor. *Non-Malleable Cryptography*. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 542–552, 1991.

Ashutosh Dhar Dwivedi, Miloš Klouček, Pawel Morawiecki, Ivica Nikolic, Josef Pieprzyk, and Sebastian Wójtowicz. *SAT-based Cryptanalysis of Authenticated Ciphers from the CAESAR Competition*. *Cryptology ePrint Archive*, 2016.

Morris Dworkin. *SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. 2007.

Maria Eichlseder, Christoph Dobraunig, Johannes Erlacher, Florian Mendel, and Martin Schläffer. *Update on the Security Analysis of Ascon*. NIST LWC Workshop 2022, 2022.

Johannes Erlacher, Florian Mendel, and Maria Eichlseder. *Bounds for the Security of ASCON against Differential and Linear Cryptanalysis*. *IACR Transactions on Symmetric Cryptology*, 2022(1):64–87, Mar. 2022. doi: 10.46586/tosc.v2022.i1.64-87. URL https://tosc.iacr.org/index.php/ToSC/article/view/9527.

Farnoud Farahmand, Ahmed Ferozpuri, William Diehl, and Kris Gaj. *Minerva: Automated Hardware Optimization Tool*. In *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–8, 2017. doi: 10.1109/RECONFIG.2017.8279804.

Amir Fotovvat, Gazi M. E. Rahman, Seyed Shahim Vedaei, and Khan A. Wahid. *Comparative Performance Analysis of Lightweight Cryptography Algorithms for IoT Sensor Nodes*. *IEEE Internet of Things Journal*, 8(10): 8279–8290, 2021. doi: 10.1109/JIOT.2020.3044526.

Vijay Ganesh and David L. Dill. *A Decision Procedure for Bit-Vectors and Arrays*. In Werner Damm and Holger Hermanns, editors, *Computer Aided Verification*, pages 519–531, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-73368-3.

David Gerault, Marine Minier, and Christine Solnon. *Constraint Programming Models for Chosen Key Differential Cryptanalysis*. In *International Conference on Principles and Practice of Constraint Programming*, pages 584–601. Springer, 2016.

David Gerault, Thomas Peyrin, and Quan Quan Tan. *Exploring Differential-Based Distinguishers and Forgeries for ASCON*. *Cryptology ePrint Archive*, 2021.

Santosh Ghosh. *Low-Latency Crypto: An Emerging Paradigm of Lightweight Cryptography*. 2022. URL https://csrc.nist.gov/csrc/media/Presentations/2022/low-latency-crypto-an-emerging-paradigm-of-lightwe/images-media/session-1-ghosh-low-latency-crypto.pdf.

Shafi Goldwasser and Silvio Micali. *Probabilistic Encryption & How to Play Mental Poker Keeping Secret all Partial Information*. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 365–377, 1982.

Lov K Grover. *A Fast Quantum Mechanical Algorithm for Database Search*. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 212–219, 1996.

Chun Guo, T Iwata, M Khairallah, K Minematsu, and T Peyrin. *Romulus v1.2*. Submission to the NIST Lightweight Cryptography Standardization Process, 2021.

Chun Guo, T Iwata, M Khairallah, K Minematsu, and T Peyrin. *Romulus as NIST LWC Finalist*. NIST LWC Workshop 2022, 2022a.

Chun Guo, Tetsu Iwata, and Kazuhiko Minematsu. *New Indifferentiability Security Proof of MDPH Hash Function*. *IET Information Security*, 16, 07 2022b. doi: 10.1049/ise2.12058.

Hosein Hadipour, Nasour Bagheri, and Ling Song. *Improved Rectangle Attacks on SKINNY and CRAFT*. Cryptology ePrint Archive, Paper 2020/1317, 2020. URL https://eprint.iacr.org/2020/1317.

Martin Hell, Thomas Johansson, Alexander Maximov, Willi Meier, Jonathan Sönnerup, and Hirotaka Yoshida. *Grain-128AEADv2 - A Lightweight AEAD Stream Cipher*. Submission to the NIST Lightweight Cryptography Standardization Process, 2021.

Miia Hermelin, Joo Yeon Cho, and Kaisa Nyberg. *Multidimensional Extension of Matsui's Algorithm 2*. In *International Workshop on Fast Software Encryption*, pages 209–227. Springer, 2009.

Howard M Heys. *A Tutorial on Linear and Differential Cryptanalysis*. *Cryptologia*, 26(3):189–221, 2002.

Kai Hu and Thomas Peyrin. *Revisiting Higher-Order Differential(-Linear) Attacks from an Algebraic Perspective: Applications to ASCON, Grain v1, XOODOO, and ChaCha*. NIST LWC Workshop 2022, 2022.

Akiko Inoue, Chun Guo, and Kazuhiko Minematsu. *Nonce-Misuse Resilience of Romulus-N and GIFT-COFB*. *Cryptology ePrint Archive*, 2022.

ISO 18033-7:2022. *Information Security — Encryption algorithms — Part 7: Tweakable Block Ciphers*. Standard, International Organization for Standardization, April 2022.

Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. *New Results on Romulus*. 2020.

Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. *Tweaks and Keys for Block Ciphers: The TWEAKEY Framework*. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 274–288, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-45608-8.

Philipp Jovanovic, Atul Luykx, and Bart Mennink. *Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes*. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 85–104, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-45611-8.

Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC press, 2020.

Jonathan Katz and Moti Yung. *Complete Characterization of Security Notions for Probabilistic Private-key Encryption*. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 245–254, 2000.

Jonathan Katz and Moti Yung. *Characterization of Security Notions for Probabilistic Private-key Encryption*. *Journal of Cryptology*, 19(1):67–95, 2006.

Keccak Team. *The Sponge and Duplex Constructions*. https://keccak.team/sponge_duplex.html [Accessed: 13/09/2022].

John Kelsey, Tadayoshi Kohno, and Bruce Schneier. *Amplified Boomerang Attacks Against Reduced-round MARS and Serpent*. In *International Workshop on Fast Software Encryption*, pages 75–93. Springer, 2000.

Mustafa Khairallah, Thomas Peyrin, and Anupam Chattopadhyay. *Preliminary Hardware Benchmarking of a Group of Round 2 NIST Lightweight AEAD Candidates*. Cryptology ePrint Archive, Paper 2020/1459, 2020. URL https://eprint.iacr.org/2020/1459. https://eprint.iacr.org/2020/1459.

Lars Knudsen. *Truncated and Higher Order Differentials*. In *International Workshop on Fast Software Encryption*, pages 196–211. Springer, 1994.

Lars Knudsen. *DEAL - A 128-bit Block Cipher*. *complexity*, 258(2):216, 1998.

Xuejia Lai. *Higher Order Derivatives and Differential Cryptanalysis*. In *Communications and Cryptography*, pages 227–233. Springer, 1994.

Susan K Langford and Martin E Hellman. *Differential-Linear Cryptanalysis*. In *Annual International Cryptology Conference*, pages 17–25. Springer, 1994.

Gregor Leander, Cihangir Tezcan, and Friedrich Wiemer. *Searching for Subspace Trails and Truncated Differentials*. *IACR Transactions on Symmetric Cryptology*, 2018(1):74–100, Mar. 2018. doi: 10.13154/tosc.v2018.i1.74-100. URL https://tosc.iacr.org/index.php/ToSC/article/view/845.

Jooyoung Lee. *Security Evaluation of Romulus*. 2022.

Charlotte Lefevre and Bart Mennink. *Tight Preimage Resistance of the Sponge Construction*. Cryptology ePrint Archive, Paper 2022/734, 2022. URL https://eprint.iacr.org/2022/734. https://eprint.iacr.org/2022/734.

Yanbin Li, Guoyan Zhang, Wei Wang, and Meiqin Wang. *Cryptanalysis of Round-reduced ASCON*. *Science China Information Sciences*, 60(3):1–2, 2017a.

Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. *Conditional Cube Attack on Round-Reduced ASCON*. *IACR Transactions on Symmetric Cryptology*, 2017(1):175–202, Mar. 2017b. doi: 10.13154/tosc.v2017.i1.175-202. URL https://tosc.iacr.org/index.php/ToSC/article/view/590.

Moses Liskov, Ronald L Rivest, and David Wagner. *Tweakable Block Ciphers*. In *Annual International Cryptology Conference*, pages 31–46. Springer, 2002.

Rusydi H. Makarim and Raghvendra Rohit. *Towards Tight Differential Bounds of ASCON: A Hybrid Usage of SMT and MILP*. *IACR Transactions on Symmetric Cryptology*, 2022(3):303–340, Sep. 2022. doi: 10.46586/tosc.v2022.i3.303-340. URL https://tosc.iacr.org/index.php/ToSC/article/view/9859.

Mitsuru Matsui. *Linear Cryptanalysis Method for DES Cipher*. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 386–397. Springer, 1993.

Mitsuru Matsui. *On Correlation Between the Order of S-boxes and the Strength of DES*. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 366–375. Springer, 1994.

Kerry McKay, Lawrence Bassham, Meltem Sönmez Turan, and Nicky Mouha. *Report on Lightweight Cryptography*. Technical report, National Institute of Standards and Technology, 2016.

Florian Mendel, Tomislav Nad, and Martin Schläffer. *Finding SHA-2 Characteristics: Searching Through a Minefield of Contradictions*. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 288–307, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25385-0.

Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2018.

Ralph C Merkle. *One Way Hash Functions and DES*. In *Conference on the Theory and Application of Cryptology*, pages 428–446. Springer, 1989.

Kamyar Mohajerani and Rishub Nagpal. *Xeda: Cross-EDA Abstraction and Automation*, 2020. URL https://github.com/XedaHQ/xeda. GitHub Repoistory.

Kamyar Mohajerani, Richard Haeussler, Rishub Nagpal, Farnoud Farahmand, Abubakr Abdulgadir, Jens-Peter Kaps, and Kris Gaj. *Hardware Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process*. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 164–169, 2021. doi: 10.23919/DATE51398.2021.9473930.

Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. *Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming*. In *International Conference on Information Security and Cryptology*, pages 57–76. Springer, 2011.

Yusuke Naito. *Optimally Indifferentiable Double-Block-Length Hashing Without Post-processing and with Support for Longer Key Than Single Block*. In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology – LATINCRYPT 2019*, pages 65–85, Cham, 2019. Springer International Publishing. ISBN 978-3-030-30530-7.

Moni Naor and Moti Yung. *Public-key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks*. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.

Yoav Nir and Adam Langley. *ChaCha20 and Poly1305 for IETF Protocols*. *RFC*, 8439:1–46, 2015.

Gorkem Nisanci, Remzi Atay, Meltem Kurt Pehlivanoglu, Elif Bilge Kavun, and Tolga Yalcin. *Will the Future Lightweight Standard be RISC-V Friendly?*, 2019. URL https://csrc.nist.gov/CSRC/media/Presentations/will-the-future-lightweight-standard-be-risc-v-fri/images-media/session4-yalcin-will-future-lw-standard-be-risc-v-friendly.pdf. NIST Lightweight Cryptography Workshop 2019.

NIST. *Lightweight Cryptography*. https://csrc.nist.gov/Projects/lightweight-cryptography [Accessed: 13/08/2022].

NIST. *Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process*, 2018.

NIST. *Benchmarking of Lightweight Cryptographic Algorithms on Microcontrollers*, 2020. URL https://github.com/usnistgov/Lightweight-Cryptography-Benchmarking. GitHub Repository.

Kaisa Nyberg. *Linear Approximation of Block Ciphers*. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 439–444. Springer, 1994.

Thomas Peyrin and Yannick Seurin. *Counter-In-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers*. In *Annual International Cryptology Conference*, pages 33–63. Springer, 2016.

Lingyue Qin, Xiaoyang Dong, Xiaoyun Wang, Keting Jia, and Yunwen Liu. *Automated Search Oriented to Key Recovery on Ciphers with Linear Key Schedule: Applications to Boomerangs in SKINNY and ForkSKINNY*. Cryptology ePrint Archive, Paper 2021/656, 2021. URL https://eprint.iacr.org/2021/656.

Charles Rackoff and Daniel R Simon. *Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack*. In *Annual International Cryptology Conference*, pages 433–444. Springer, 1991.

S Renner, E Pozzobon, and J Mottok. *NIST LWC Software Performance Benchmarks on Microcontrollers*, 2019. URL https://lwc.las3.de/. Project Webpage.

Ronald L. Rivest. *Cryptology*. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. Elsevier, 1990.

Phillip Rogaway and Thomas Shrimpton. *A Provable-Security Treatment of the Key-Wrap Problem*. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 373–390. Springer, 2006.

Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. *Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round ASCON*. *IACR Transactions on Symmetric Cryptology*, 2021(1):130–155, Mar. 2021. doi: 10.46586/tosc.v2021.i1.130-155. URL https://tosc.iacr.org/index.php/ToSC/article/view/8835.

Luan Cardoso dos Santos and Johann Großschädl. *An Evaluation of the Multi-Platform Efficiency of Lightweight Cryptographic Permutations*, 2020. URL https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2020/documents/papers/performance-evaluation-cryptographic-permutations-lwc2020.pdf. NIST Lightweight Cryptography Workshop 2020.

Ali Aydın Selçuk. *On Probability of Success in Linear and Differential Cryptanalysis*. *Journal of Cryptology*, 21(1): 131–147, 2008.

Claude E Shannon. *Communication Theory of Secrecy Systems*. *The Bell System Technical Journal*, 28(4): 656–715, 1949.

Cihangir Tezcan. *Truncated, Impossible, and Improbable Differential Analysis of ASCON*. *Cryptology ePrint Archive*, 2016.

Cihangir Tezcan. *Analysis of ASCON, DryGASCON, and Shamash Permutations*. Cryptology ePrint Archive, Paper 2020/1458, 2020. URL https://eprint.iacr.org/2020/1458. https://eprint.iacr.org/2020/1458.

Yosuke Todo. *Structural Evaluation by Generalized Integral Property*. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 287–314. Springer, 2015.

Meltem Sönmez Turan, Kerry A McKay, Çagdas Çalik, Donghoon Chang, Lawrence Bassham, et al. *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)*, 2019.

Meltem Sönmez Turan, Kerry McKay, Donghoon Chang, Cagdas Calik, Lawrence Bassham, Jinkeon Kang, John Kelsey, et al. *Status Report on the Second Round of the NIST Lightweight Cryptography Standardization Process. National Institute of Standards and Technology Internal Report*, 8369(10.6028), 2021.

Sönmez Meltem Turan. *Update on the NIST Lightweight Cryptography Standardization Process*. https://csrc.nist.gov/Presentations/2022/update-on-the-nist-lwc-standardization-process [Accessed: 1/10/2022], 2022.

Corentin Verhamme, Gaetan Cassiers, and Francois-Xavier Standaert. *Analysing the Leakage-Resistance of the NIST's Lightweight Crypto Standardization Process Finalists*. 2022.

David Wagner. *The Boomerang Attack*. In *International Workshop on Fast Software Encryption*, pages 156–170. Springer, 1999.

R Weatherley. *Lightweight Cryptography Primitives*, 2021. URL https://rweather.github.io/lightweight-crypto/index.html. Project Webpage.

Hongjun Wu and Tao Huang. *TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms (Version 2)*. Submission to the NIST Lightweight Cryptography Standardization Process, 2021.

Tolga Yalcin and Samaneh Ghandali. *Need for Low-latency Ciphers: A Comparative Study of NIST LWC Finalists*. 2022. URL https://csrc.nist.gov/csrc/media/Presentations/2022/need-for-low-latency-ciphers-a-comparative-study-o/images-media/session-1-yalcin-need-for-low-latency-ciphers.pdf.

Rui Zong, Xiaoyang Dong, and Xiaoyun Wang. *Collision Attacks on Round-Reduced Gimli-Hash/ASCON-XOF/ASCON-HASH. IACR Cryptol. ePrint Arch.*, 2019:1115, 2019.