PARALLEL, SPONGE-BASED AUTHENTICATED ENCRYPTION WITH SIDE-CHANNEL PROTECTION AND ADVERSARY-INVISIBLE NONCES

MOHAMUD AHMED JIMALE

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

UNIVERSITI MALAYA

KUALA LUMPUR

2023

PARALLEL, SPONGE-BASED AUTHENTICATED ENCRYPTION WITH SIDE-CHANNEL PROTECTION AND ADVERSARY-INVISIBLE NONCES

MOHAMUD AHMED JIMALE

THESIS SUBMITTED IN FULFILMENT

OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

UNIVERSITI MALAYA

KUALA LUMPUR

2023

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: MOHAMUD AHMED JIMALE

Matric No: wva180031/17198096

Name of Degree: Doctor of philosophy (Ph.D.)

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

PARALLEL, SPONGE-BASED AUTHENTICATED ENCRYPTION WITH SIDE-CHANNEL PROTECTION AND ADVERSARY-INVISIBLE NONCES.

Field of Study: Information Security

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes, and any excerpt or extract from, reference to or reproduction of any copyrighted work has been disclosed expressly and sufficiently, and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge, nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyrighted work;
(5) I hereby assign all and every right in the copyright to this work to the University of Malaya ("UM"), who henceforth shall be the owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if, in the course of making this Work, I have infringed any copyright, whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date: 12/07/2023

Subscribed and solemnly declared before,

Witness's Signature Date: 12/07/2023

Name:

Designation:

PARALLEL, SPONGE-BASED AUTHENTICATED ENCRYPTION WITH SIDE-CHANNEL PROTECTION AND ADVERSARY-INVISIBLE NONCES

ABSTRACT

Since its birth in 2000, Authenticated Encryption (AE) has been a hot research topic. AE plays a crucial role in secure communications today since it is the backbone of standard, secure communication protocols like SSH, SSL, and TLS. In this regard, many new features have been proposed to boost its security, efficiency, or performance. AE is a cryptographic scheme that simultaneously provides two essential security services, i.e., confidentiality and authenticity. The block cipher was the dominant underlying primitive in constructing AE schemes with few others, like stream ciphers and compression functions. Sponge construction is a cryptographic primitive that emerged in 2007 and was first used for AE in 2011. It relies on an iterated permutation or transformation that can be used to implement reseedable pseudorandom generators, hashing, and AE schemes. Sponge-based AE schemes provide functional characteristics such as parallelizability, incrementality, and being online. They also offer security features for protection against active or passive adversaries. Parallel sponge-based AE schemes are not protected against side-channel attacks such as simple power analysis (SPA) and differential power analysis (DPA). On the other hand, sponge-based AE schemes that protect against such attacks are serial and cannot be parallelized. Furthermore, sponge-based AE schemes handle the nonces in a way that could allow misuse. So, sponge-based AE schemes that hide the nonce from adversaries are also an open problem. This work aims to bridge these gaps by proposing a parallel sponge-based AE with side-channel protection and adversary-invisible nonces (PSASPIN), using parallel fresh rekeying and the duplex mode of the sponge construction. A leveled implementation is used to implement the key generation part using a pseudorandom function (PRF) based on the Galois field multiplication. The data processing (the rekeyed) part is implemented using the sponge-based duplex mode.

Finally, the security proof of the proposed scheme is provided using game-based theory according to the PRP/PRF switching lemma, and its performance is analyzed. The analysis demonstrates the effectiveness of the proposed scheme in terms of security and performance. The security analysis shows that the proposed scheme is secure in the ideal permutation model. The performance analysis shows that the proposed scheme is comparable to existing sponge-based AE schemes in processing larger message sizes, despite offering unique features that combine SCAs protection, nonce-obliviousness, and parallelism.

Key words: Integrity, Authenticated Encryption, Authentication, Confidentiality, CAESAR Competition, Message Authentication Code, NIST-LW Competition, Cryptographic Sponge Function.

AE BERASASKAN SPAN SELARI DENGAN PERLINDUNGAN SALURAN SISI DAN NONCE YANG HALIMUNAN KEPADA MUSUH (PSASPIN)

ABSTRAK

Sejak kelahirannya pada tahun 2000, Penyulitan Disahkan (AE) telah menjadi topik penyelidikan yang hangat. AE memainkan peranan penting dalam komunikasi selamat hari ini kerana ia merupakan tulang belakang protokol komunikasi yang standard dan selamat seperti SSH, SSL dan TLS. Dalam hal ini, banyak ciri baharu telah dicadangkan untuk meningkatkan keselamatan, kecekapan atau prestasinya. AE ialah skim kriptografi yang menyediakan dua perkhidmatan keselamatan penting secara serentak, iaitu kerahsiaan dan integriti. Sifir blok adalah asas primitif yang dominan dalam membina skema AE dengan beberapa yang lain, seperti sifir aliran (*stream ciphers*) dan fungsi pemampatan (*compression functions*). Pembinaan span (*sponge construction*) ialah primitif kriptografi yang muncul pada tahun 2007 dan mula-mula digunakan untuk AE pada tahun 2011. Ia bergantung pada pilih atur atau transformasi lelaran yang boleh digunakan untuk melaksanakan penjana *pseudorandom* boleh benih semula (*reseedable pseudorandom generators*), pencincangan (*hashing*) dan skim AE. Skim AE berasaskan span menyediakan ciri kefungsian seperti kebolehselarian (*parallelizability*), penambahan (*incrementality*) dan berada dalam talian. Mereka juga menawarkan ciri keselamatan untuk perlindungan daripada musuh aktif atau pasif. Terdapat skim AE berasaskan span selari, tetapi ia tidak dilindungi daripada serangan aliran sisi (*side channel attacks*) seperti analisis kuasa mudah (*simple power analysis*, SPA) dan analisis kuasa pembezaan (*differential power analysis*, DPA). Sebaliknya, skim AE berasaskan span yang melindungi daripada serangan sedemikian adalah bersiri dan tidak boleh disejajarkan. Tambahan pula, skim AE berasaskan span mengendalikan *nonce* dengan cara yang boleh membenarkan penyalahgunaan. Justeru, skim AE berasaskan span yang

menyembunyikan *nonce* daripada musuh juga merupakan masalah terbuka. Kerja ini bertujuan untuk merapatkan jurang ini dengan mencadangkan AE berasaskan span selari dengan perlindungan saluran sisi dan *nonce* yang halimunan kepada musuh (PSASPIN), menggunakan penguncian semula selari (*fresh rekeying*) dan mod dupleks pembinaan span. Pelaksanaan bertingkat digunakan untuk melaksanakan bahagian penjanaan utama menggunakan fungsi *pseudorandom* (PRF) berdasarkan pendaraban medan Galois. Bahagian pemprosesan data (yang dikunci semula) dilaksanakan menggunakan mod dupleks berasaskan span. Akhir sekali, bukti keselamatan skim yang dicadangkan disediakan menggunakan teori berasaskan permainan mengikut lema pensuisan PRP/PRF, dan prestasinya dianalisis. Analisis menunjukkan keberkesanan skim yang dicadangkan dari segi keselamatan dan prestasi. Analisis keselamatan menunjukkan bahawa skim yang cadangan adalah selamat dalam model pilih atur ideal. Analisis prestasi menunjukkan bahawa skim yang dicadangkan adalah setanding dengan skim-skim AE berasaskan span sedia ada dalam memproses saiz mesej yang lebih besar, walaupun menawarkan ciri unik yang menggabungkan perlindungan SCA, nonce-obliviousness, dan paralelisme.

Kata kunci: Integriti, Penyulitan Disahkan, Pengesahan, Kerahsiaan, Pertandingan CAESAR, Kod Pengesahan Mesej, Pertandingan NIST-LW, Fungsi Span Kriptografi.

ACKNOWLEDGEMENT

All praise and glory be to Allah for blessing, guiding, and strengthening me every single moment of my being. I am grateful to my supervisors, Prof. Ts. Dr. Miss Laiha Binti Mat Kiah and Dr. Muhammad Reza Bin Z'aba, for their kindness, guidance, patience, and caring attention from beginning my studies to completing my safe journey to Ph.D. I am sure I cannot pay back their genuine help at the most needed time, but I thank them and pray for them from my deepest heart. I was lucky to have such great people at my side. I thank the University of Malaya for allowing me to stand on the shoulders of those giants.

I am grateful to African Education and Development Trust (AEDT), which gave me a loan without interest to pursue my Ph.D. and pay back at my convenience. I doubt I could think of doing my Ph.D. without their generous help. Furthermore, I thank Jamhuriya University Of Science and Technology (JUST) in Mogadishu, Somalia, for giving me study leave and supporting me during the hard times of my studies. Without their help, my wish for a doctoral degree would have remained a dream.

I am incredibly thankful to my beloved wife, Lul Mohamed Hussein, and children, Fatima, Mohamed-Amin, Abdulrahman, Ramla, and Aisha, for their patience with me during my studies and for forgiving me for being away while they needed me beside them. I am equally thankful to all family members, relatives, and friends who supported me materially or morally during my studies.

I wish I could make their support, prayers, and encouragement worthwhile.

DEDICATION

To my late mother

You encouraged me with your prayers, enthusiasm, and unconditional love at the beginning of my journey.

I wish you shared with me the sense of triumph at its end.

May Allah shower his mercies on you!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviation	Meaning
AD	Associated Data
ADV	Advantage
AE	Authenticated Encryption
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
BBB	Security beyond birth bound
BC	Block Cipher
CAESAR	Competition for Authenticated Encryption: Security, Applicability, and Robustness
CBC	Cipher Block Chaining
CCA	Chosen Ciphertext Attack
CDH	Computational Diffie-Hellman
CF	Compression function
CIA	Confidentiality, Integrity, Availability
CPA	Chosen Plaintext Attack
CTR	Counter Mode
CTX	Ciphertext
DE	Dedicated schemes
DES	Data Encryption Standard
DL	Discrete Logarithm
DPA	Differential Power Analysis
EME	Encrypt Mix Encrypt
FRK	Fresh Rekeying
GCM	Galois Counter Mode

GF	Galois Field
HB	Hybrid primitives
HN	Hide Nonce
IAs	Implementation Attacks
IDE	Integrated Development Environment
IND	Indistinguishability
INT	Integrity
IoT	Internet of Things
IP	Internet Protocol
MAC	Message Authentication Code
MRAE	nonce misuse-resistant AE
NAE	Nonce-based AE
NBE	Nonce-Based Encryption
NBS	National Bureau of Standards
NIST	National Institute of Standards and Technology
NIST-LW	NIST Lightweight Competition
NMP	Non-malleability
NMR	Nonce misuse resistant
OAE	Online AE
OCB	Offset Codebook Mode
OCCA	Online Chosen Ciphertext Attack
OMD	Offset Merkle-Damgård
OPRP	Online Pseudorandom permutation
PAAs	Power Analysis Attacks
PR	Permutation
PRF	Pseudorandom function

PRP	Pseudorandom permutation
PSASPIN	Parallel Sponge-based Authenticated Encryption with Side-channel Protection and adversary-invisible Nonces
PTXT	Plaintext
RFID	Radio Frequency identification
ROM	Radon Oracle Model
RUP	Release of unverified plaintext
SC	Stream Cipher
SCAs	Side-channel Attacks
SHA	Secure Hashing Algorithm
SIV	Synthetic Initialization vector
SLR	Systematic Literature Review
SMN	Secret Message Number nonce
SP	Sponge-based
SPA	Simple Power analysis
SSH	Secure Shell
SSL	Secure socket layer
TBC	Tweakable Block Cipher
TCP	Transmission Control Protocol
TLS	Transport Layer Security
WPA	Wi-Fi Protected Access

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

This chapter provides an overview of this thesis, presenting the statement of the problem, the objectives, and the methodology used to solve the research problem. Section 1.1 contains the background of the study, Section 1.2 includes the motivation of the study, and Section 1.3 presents the problem statement. Section 1.4 specifies the objectives of the study; Section 1.5 presents the proposed methodology. Finally, the organization of this study is outlined in Section 1.6.

1.1 Background

The secrecy of confidential information has always been a concern in the known history of human beings. Narratives about tools for hiding information from opponents and others to crack it is available in medieval and modern accounts. As far as cryptography (hiding information by encoding it) and steganography (hiding the existence of data) are concerned, cryptanalysis (breaking the secret codes) almost co-existed with them to expose the opponent's secrets (G. R & Ganesh, 2018).

In modern days, cryptography is the backbone of secure communications and provides security mechanisms that offer services such as confidentiality, integrity, & availability. As depicted in Figure 1.1, cryptography can be broadly classified into symmetric key, asymmetric key, and hash functions. The first two categories in this classification differ in the number of keys needed to encrypt and decrypt the ciphertext. Hash functions are special cryptographic primitives that produce message digests to ensure message integrity (Stallings, 2014).

Symmetric key systems use the same key for encrypting and decrypting messages; thus, they need secure ways to share the secret keys. On the other hand, asymmetric key

systems use two different keys, a public key and a private key, for encrypting and decrypting messages. Symmetric key ciphers can be generally classified according to their functions into encryption schemes like block ciphers and stream ciphers, which protect the secrecy, and message authentication codes (MACs), which ensure the authenticity of messages (G. R & Ganesh, 2018; Stallings, 2014).

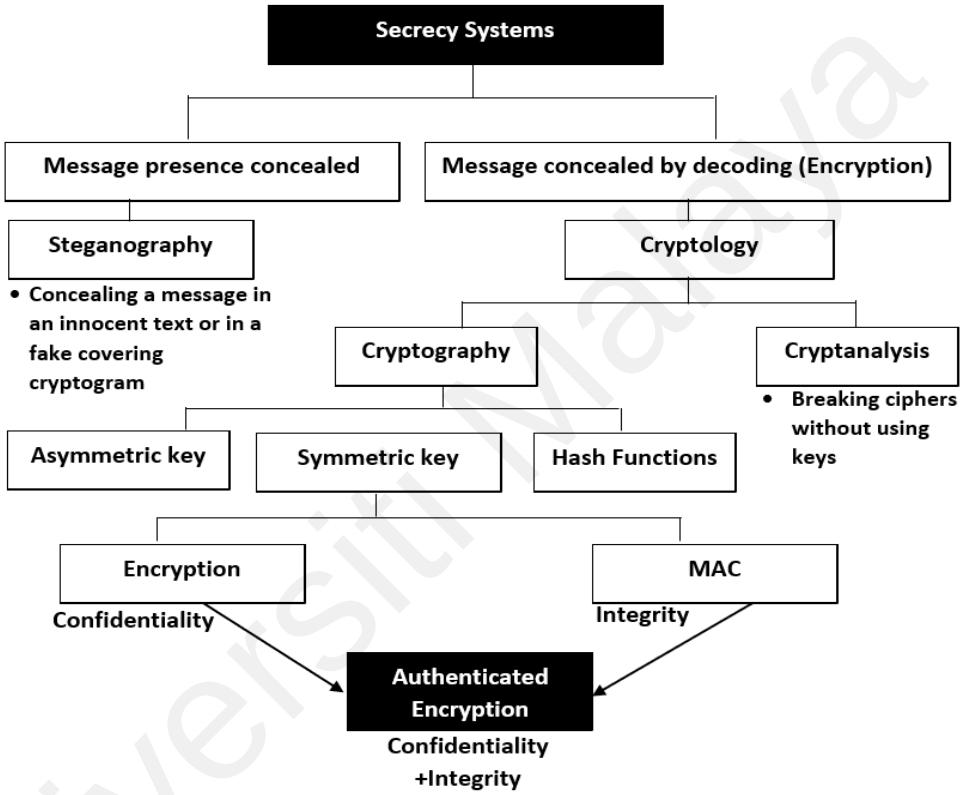

Figure 1. 1 Security systems, cryptography, steganography & cryptanalysis

The emergence of authenticated encryption (AE) was motivated by the need for a cryptographic scheme that simultaneously provides confidentiality and authenticity since encryption alone is not enough for a system to be secure. Encryption primitives such as block and stream ciphers only provide confidentiality of messages, i.e., messages are protected from being viewed by unauthorized entities. Such primitives cannot be naively used in secure communications since it is trivial for an adversary to tamper with the encrypted message (i.e., ciphertext) without detection.

In that regard (Bellare & Namprempre, 2000a) and (Katz & Yung, 2001) came up with the first AE schemes as "authenticated encryption" and "unforgeable encryption," respectively, by interweaving individual encryption and message authentication code (MAC) schemes. In addition, an extension of AE called AE with associated data (AEAD) provides authentication of additional data without encrypting them (Hawkes & Rose, 2003; Jonsson, 2003; Morris Dworkin, 2007; Riou, 2019). A typical example is a network packet header, where only the payload needs to be encrypted, but both the header and the encrypted payload must also be authenticated. Figure 1.2 illustrates a schematic structure of AE.

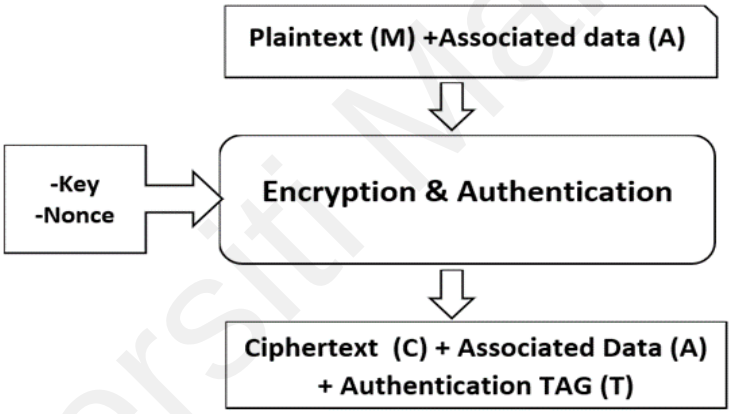

Figure 1. 2 A schematic structure of Authenticated Encryption

AE schemes are widely used in IPsec and Transport Layer Security (TLS) protocols. Moreover, the latest version of TLS, i.e., 1.3, has removed support for non-AE schemes such as AES in cipher block chaining (CBC) mode as of August 2018 (Rescorla, 2018). They are also used to provide end-to-end encryption in popular messaging applications such as WhatsApp, Telegram, and Signal.

The combination of ciphers and MACs can be achieved in several ways depending on the order in which encryption and authentication algorithms are applied and are usually termed "generic compositions": (1) Encrypt-Then-MAC, in which the message is

encrypted, then the authenticated tag is generated on it (2) Encrypt-and-MAC, in which the encryption and authentication algorithms are indecently applied and combined (3) MAC-then-Encrypt in which the authenticated tag is generated on the plain text, then the tag and the message are encrypted; however, Bellare and Namprempre reported in 2000 that most of the approaches mentioned above are weak when analyzed under several notions of security (Bellare & Namprempre, 2000b, 2008).

Various other attacks soon ensued that highlighted the intricacies of the generic composition approach (Bellare, Kohno, & Namprempre, 2004; A. Boldyreva & Kumar, 2011; Degabriele & Paterson, 2010; Paterson & Watson, 2012; Vaudenay, 2002). Although Encrypt-Then-MAC has been shown to be provably secure (Bellare & Namprempre, 2008; Krawczyk, 2001), it can still be attacked, exploiting practical details in its implementation (Bellare & Namprempre, 2008).

Due to the delicate nature of independently combining encryption and MAC to achieve a secure construction, a single primitive that intrinsically provides confidentiality and authenticity was highly sought after. Hence, dedicated AE schemes were developed to provide an efficient solution to this onerous problem. Though the idea may have been mulled much earlier in 1987 by (Andreeva et al., 2013); Jansen and Boekee (1988), the earliest practical designs came at the turn of the 21st century by Katz and Yung (2001). Other researchers follow this swiftly (Gligor & Donescu, 2002; Jutla, 2001; Rogaway, Bellare, Black, & Krovetz, 2001). The new breed of dedicated AE schemes utilizes a single key in contrast to the traditional approaches that necessitate using two separate keys: one for encryption and the other for authentication, to differentiate their purpose (Martin, 2012).

Six AE schemes were standardized in 2009 as ISO/IEC 19772: OCB 2.0 (Rogaway, 2004a) to foster compatibility: Key Wrap (Morris Dworkin, 2007), CCM

(Morris Dworkin, 2007; Whiting, Housley, & Ferguson, 2003), EAX (Bellare, Rogaway, & Wagner, 2003), EtM, and GCM (Dworkin, 2007). OCB 2.0 was later removed from the 2020 edition of the ISO/IEC 19772 standard due to the security flaw discovered by Inoue et al. (Inoue, Iwata, Minematsu, & Poettering, 2020).

Since the seminal articles of Bellare and Namprempre (2000a); (Bellare & Namprempre, 2008), AE has undergone continuous enhancements in implementation logic, performance, and functional characteristics. The belief that AE schemes could still be refined paved the way for the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) project, jointly initiated in 2013 by the US National Institute of Standards and Technology (NIST) and Dan Bernstein (D. Bernstein, 2013). The final CAESAR portfolio was announced in 2018, which consists of six schemes (Andreeva et al., 2016; Dobraunig, Eichlseder, Mendel, & Schläffer, 2016b; Jean, Nikolić, & Peyrin, 2016; Krovetz & Rogaway, 2016; Wu, 2016; Wu & Preneel, 2016). In the same year, triggered by the rise of the Internet-of-Things (IoT), which mainly consists of resource-constrained devices, the NIST solicited a call to standardize lightweight AE schemes (hereafter referred to as NIST-LW schemes). By lightweight, we mean that the schemes should be suitable for implementation in devices with scarce resources such as memory and power. On March 29, 2021, NIST announced ten finalists out of the 32 candidates from Round 2 as candidates in the concluding round. At the time of writing, NIST has yet to announce the final portfolio for standardization.

AE protects data confidentiality and authenticity in two forms, each branch into two notions of security. Confidentiality (or privacy) protects the secrecy of information in case of the Chosen Plaintext Attack (IND-CPA) against eavesdropping adversaries and Chosen Ciphertext Attack (IND-CCA) against active adversaries (Bellare & Namprempre, 2000b; Katz & Yung, 2001; Rogaway, 2002a). Integrity/authenticity guarantees that the messages are from legitimate sources and have not been tampered

with during transit or at rest. It protects the integrity of plaintext under the INT-PTXT model and the integrity of the ciphertext under the INT-CTXT model.

An AE scheme may rely on a user-supplied nonce (number used once) that is not supposed to be reused to encrypt different plaintexts under the same key to prevent predictability (Rogaway, 2002a; Rogaway et al., 2001). Examples of nonce include a counter that is incremented with every new encryption. As nonce-based AE (NAE) schemes do not handle nonce generation, implementors must ensure that the nonces are correctly used (Gueron & Lindell, 2015a). Application developers are responsible for determining how nonces are generated. Such practice is prone to misuse because reusing nonces (intentionally or otherwise) can have dire consequences. Various protocols and applications have been violated due to the mishandling of nonces. Examples include Wired Equivalent Privacy (WEP) (Borisov, Goldberg, & Wagner, 2001), WinZip (Kohno, 2004), Microsoft Office (Wu, 2005), and Wi-Fi-protected access (WPA) 2 (Vanhoef & Piessens, 2017).

To address the nonce misuse concern, in 2006, Rogaway and Shrimpton (Rogaway & Shrimpton, 2006) proposed the notion of nonce misuse-resistant AE (MRAE), which ensures an acceptable level of security even though nonces are repeated (Rogaway & Shrimpton, 2006). Furthermore, a recent study (Bellare, Ng, & Tackmann, 2019a) proposed a way to modify the existing NAE syntax so that the decryption does not take a nonce anymore. The nonce-oblivious syntax is meant to alleviate the trouble of nonce handling from developers and implementers and prevent potential misuse resulting from inappropriate nonce values.

The security of cryptographic protocols is measured to what extent they withstand attacks with specific assumed resources. But some adversaries do not take advantage of weaknesses in cryptographic algorithms but exploit sideline information from the

implementation environments instead. Those implementation-related attacks are known as side-channel attacks (SCAs). SCAs are particularly harmful when chips with sensitive information are in hackers' hands or are deployed where they are accessible to the general public, like IoT devices, sensor network nodes, and many types of smart cards. (Dobraunig, Eichlseder, et al., 2019c), (Mennink, 2020).

Several techniques are in place to prevent SCAs, including hiding (Mangard, Oswald, & Popp, 2007) and masking (Duc, Faust, & Standaert, 2015; Ishai, Sahai, & Wagner, 2003; Mennink, 2020; Prouff & Rivain, 2013). Fresh rekeying is another cheaper way to protect against SCAs (Abdalla & Bellare, 2000; Medwed, Standaert, Großschädl, & Regazzoni, 2010; Mennink, 2020). In fresh rekeying, we do not use only the target cipher but also a subkey generation function that uses the master key as input.

AE schemes are built on some underlying constructions or building blocks. Some of the most used building blocks are block ciphers that accept as input a plaintext block of fixed length and a secret key. A key scheduling algorithm takes the secret key and derives a series of round subkeys. The input plaintext is processed iteratively where, in each round, one of the subkeys is applied. The final round outputs the corresponding ciphertext block equal in length to the input plaintext block. Famous block ciphers include the AES (Daemen & Rijmen, 2002), SKINNY (Beierle et al., 2016), and GIFT (Banik et al., 2017). An extension to the traditional block cipher called tweakable block cipher accepts an additional public input value known as a tweak. Stream ciphers encrypt bits individually by adding a bit from a key stream to a plaintext bit by taking a secret key of a fixed length to generate a key stream of variable length. Stream ciphers can be used as a core primitive in authenticated encryption to achieve confidentiality and authenticity/integrity if the cipher is secure (Tahir, Javed, & Cheema, 2008).

Permutation-based structures use dedicated and keyless permutations as underlying primitives. Schemes in this category do not use permutations in a sponge-like mode but apply other techniques like XOR, Encrypt XOR, Encrypt Mix Encrypt (EME), and derivations of the Even-Mansour construction (Alizadeh, Aref, & Bagheri, 2014b). The sponge-based structure is the most used form of keyless permutation. Specific schemes use keyless permutations in a sponge-like mode of operation, like the Keccak-f permutation used in the SHA3 hash function, whereas others rely on dedicated permutations (Assche, 2011). AE can also be based on other building blocks, such as the hash function/compression function (CF) (Cogliani et al., 2014). Furthermore, some schemes use dedicated structures as their underlying structure, like those in (Peña & Torres, 2016) (Xin, Zhi, & Deng-Guo, 2004).

In addition to security-related properties, other essential features boost AE schemes' performance and efficiency, including the following: Parallelizability, a speed parameter, reflects the ability of a scheme to process the i^{th} block independently of the j^{th} block (Iwata, Minematsu, Guo, & Morioka, 2017). A parallelizable cryptographic algorithm allows messages to be handled in parts that can be processed simultaneously on multiple processes, threads, or processors. The cipher speed is one of the most crucial features of cryptographic algorithms, and even small advantages of speed may affect the users' choice of ciphers. However, robust cryptographic algorithms are often slow and time-consuming; particularly efficient implementation is necessary for an online application. Therefore, it is vital to parallelize AE schemes for faster data processing. Unfortunately, parallelizability is not very common in sponge-based AE schemes because only 4 out of 39 sponge-based AE constructions in our SLR (M. Jimale et al., 2022) were parallel.

Online is the ability of an AE scheme to compute the i^{th} ciphertext block after having seen the first i plaintext blocks and does not need to know any plaintext beyond this block (Bellare, Boldyreva, Knudsen, & Namprempre, 2001). Inverse-free: An AE scheme is inverse-free if the underlying primitives do not require their inverses to perform encryption or decryption (Dobraunig, Eichlseder, et al., 2016b; Jean et al., 2016).

Another important property is incrementality: The ability to update parts affected only by the last action, given a previous ciphertext–tag pair (C, T) (Sasaki & Yasuda, 2016). The incrementality of a scheme is an efficiency feature that dictates that the effort of updating a document should be proportional to the size of the change in the source data. Incremental cryptography is advantageous when cryptographic operations process the same data multiple times. For instance, when someone is disseminating the same document to various parties, they could apply the cryptographic functions to the whole document at once but sign only the information about a specific receiver instead of signing the entire data as many times as the number of receivers. Unfortunately, Incrementality is a rare feature in sponge-based AE schemes. For instance, in our literature of 217 AE schemes, only 2 of the 39 sponge-based schemes claimed to support incrementality (M. Jimale et al., 2022).

Single-pass indicates that an AE scheme processes the plaintext only once to achieve confidentiality and authenticity. Many AE schemes require two separate operations for achieving privacy and integrity, especially those following the generic composition paradigms, requiring two passes over the ciphertext for encryption and authentication. However, fulfilling the two purposes with a single operation renders schemes more compact and efficient. In this sense, most sponge-based AE schemes are

single-pass; for instance, 24 out of 39 sponge-based AE schemes in our SLR are single-pass. (Boorghany, Bayat-Sarmadi, & Jalili, 2016; Reyhanitabar, Vaudenay, & Vizár, 2016b).

The lightweight property determines whether the scheme is suitable for use by resource-constrained devices (Agrawal, Chang, & Sanadhya, 2015; Chakraborti, Chattopadhyay, Hassan, & Nandi, 2015). In the NIST-LW competition dedicated to lightweight AE, many schemes are a low resource, such as those proposed in (Andreeva et al., 2013; Engels, Saarinen, Schweitzer, & Smith, 2011).

AE schemes can be classified according to the following six characteristics: The category they belong to, underlying building blocks, security definitions, functional features, modes and design, and cryptographic primitives they use. We propose a classification framework for AE schemes in section 2.3.2 of this thesis.

1.2 The motivation for this study

AE plays a crucial role in secure communications today since it is the backbone of popular, secure communication protocols like SSH, SSL, and TLS. However, despite the continuous endeavors to enhance its security and efficiency, issues remain to resolve. For instance, AE schemes based on sponge construction came with unique design features; Still, since the sponge construction is not parallelizable at the algorithmic level, most AE schemes based on sponge construction are serial (Assche, 2011; Dobraunig, Eichlseder, Mendel, & Schläffer, 2019). Several works proposed parallelizable AE based on sponge construction but did not protect against SPA and DPA attacks (Aumasson, Jovanovic, & Neves, 2016; Gligoroski, Mihajloska, Samardjiska, Jacobsen, El-Hadedy,

et al., 2014). On the other hand, sponge-based AE schemes that protect against such attacks lack the parallelizability feature (Dobraunig, Eichlseder, et al., 2019a). Furthermore, sponge-based AE schemes offered so far handle nonces in a way that allows adversaries to view or play with them, so hiding them in the sponge-based AE schemes is also an open problem (Bellare et al., 2019a).

This work is a complementary part of the continuous endeavors to enhance the AE schemes interns of security and performance by proposing a sponge-based AE scheme with the following features: (a) Parallelizability, (b) protection against SPA and DPA, (c) nonce-obliviousness and (d) NMR.

1.3 Statement of the problem

The pervasive use of AE in secure communications resulted in continuous revision and more scrutiny of its characteristics. Thus, recent studies focused on balancing security and performance without sacrificing other preferable design features. For instance, there have been challenges in combining nonce-obliviousness, protection against SCAs, and parallelism features in sponge-based AE schemes.

Despite the benefits that nonces brought to strengthen security, they have also become a tool to spoil nonce-based encryption schemes' protection; their security claims hold as long as nonces are unique. The valid nonces format and the way to transmit them also stimulated a hot debate among the research community. Although there have been many innovative trials to solve nonce-related anomalies in AE schemes, such as repetition misuse and defective nonce formats, there is still room for improvement in nonce handling regarding security and efficiency. In this regard, (Bellare, Ng, & Tackmann, 2019b) proposed nonce-oblivious AE hiding the nonces in the ciphertext to be inaccessible to the adversaries. They demonstrated ways to transform the conventional nonce-based schemes

into nonce oblivious and concretized them for block cipher-based algorithms using several nonce-hiding transforms. However, nonce oblivious AE schemes based on Sponge construction remain an open problem.

In addition to analyzing an AE scheme under the security models mentioned above, attacks may benefit sideline information from its implementation environment to break systems. Such attacks, classified as Side-Channel Attacks (SCAs), are particularly harmful when devices with sensitive information are accessible to the general public (Dobraunig, Eichlseder, et al., 2019c), (Mennink, 2020; Picek et al., 2017). Several techniques are in place to prevent SCAs, including hiding (Mangard et al., 2007), masking (Duc et al., 2015; Ishai et al., 2003; Mennink, 2020; Prouff & Rivain, 2013), and code morphing techniques (Antognazza, Barenghi, & Pelosi, 2021). However, rekeying is a less resource-intensive way to protect against side-channel attacks (SCAs) (Abdalla & Bellare, 2000; Medwed et al., 2010; Mennink, 2020). Sponge-based AE schemes that protect against SPA and DPA lack the merit of parallelizability, an important performance feature. Therefore, it is crucial to instill parallelizability in them. In addition to being a performance booting feature, parallelizability also contributes to the protections against SCAs, according to Mangard et al. (2007).

Besides the security-related characteristics, other essential attributes boost the performance and efficiency of AE schemes, like the ability to process data in parallel, online, inverse-free, incrementality, single-pass, and lightweight. Unfortunately, existing parallel, online, and incremental sponge-based AE schemes do not protect against certain types of SCAs and cannot hide nonce from adversaries. Therefore, developing sponge-based AE schemes with these desirable features and protecting against SPA and DPA is imperative.

1.4 Statement of Objectives

1. To investigate the status of AE schemes in the symmetric key setting and explore their security characteristics and other functional features in-depth.
2. To propose and implement a sponge-based AE scheme with the following features: nonce-oblivious, single-pass, nonce-misuse resistant (NMR), parallelizable, incremental, and protection against SPA and DPA. The scheme will be described using algorithms and implemented in C programming Language.
3. To analyze the security of the proposed scheme based on its implementation levels using game-based theory according to the PRP/PRF switching lemma and its performance level using a well-established benchmarking framework for AE schemes employed by similar schemes.

1.5 Proposed Methodology

We investigate the status of AE schemes in symmetric key setting and explore their security and functional features highlighting the tremendous work that has been done so far and identifying the existing gaps. In addition, the study classifies the different lines of work that have appeared since its inception in 2000.

We identify the research problem in sponge-based authenticated encryption schemes in terms of supporting desirable efficiency, performance, and protection against SCAs. Then, we investigate several solutions proposed and the shortcomings of each—for instance, sponge-based AE schemes that are Single-pass and parallelizable and those that protect against SCAs (Bellizia et al., 2019a; Dobraunig, Eichlseder, et al., 2019a), and the need for nonce-hiding schemes (Bellare et al., 2019a). Finally, we identified the gaps, defined problems, and formulated research objectives.

A parallel sponge-based AE scheme with SPA and DPA protection and adversary inviable nonces is proposed to solve the identified problem. The new solution would combine the merit of being parallel and single-pass and the protection against certain types of SCAs using fresh rekeying realized as a leveled implementation.

The proposed solution is implemented using algorithms, then coding in the C programming language using Microsoft Visual Studio Code version 1.63.2 (user setup) and GCC version of (Rev5, Built by MSYS2 project) 11.2.0.

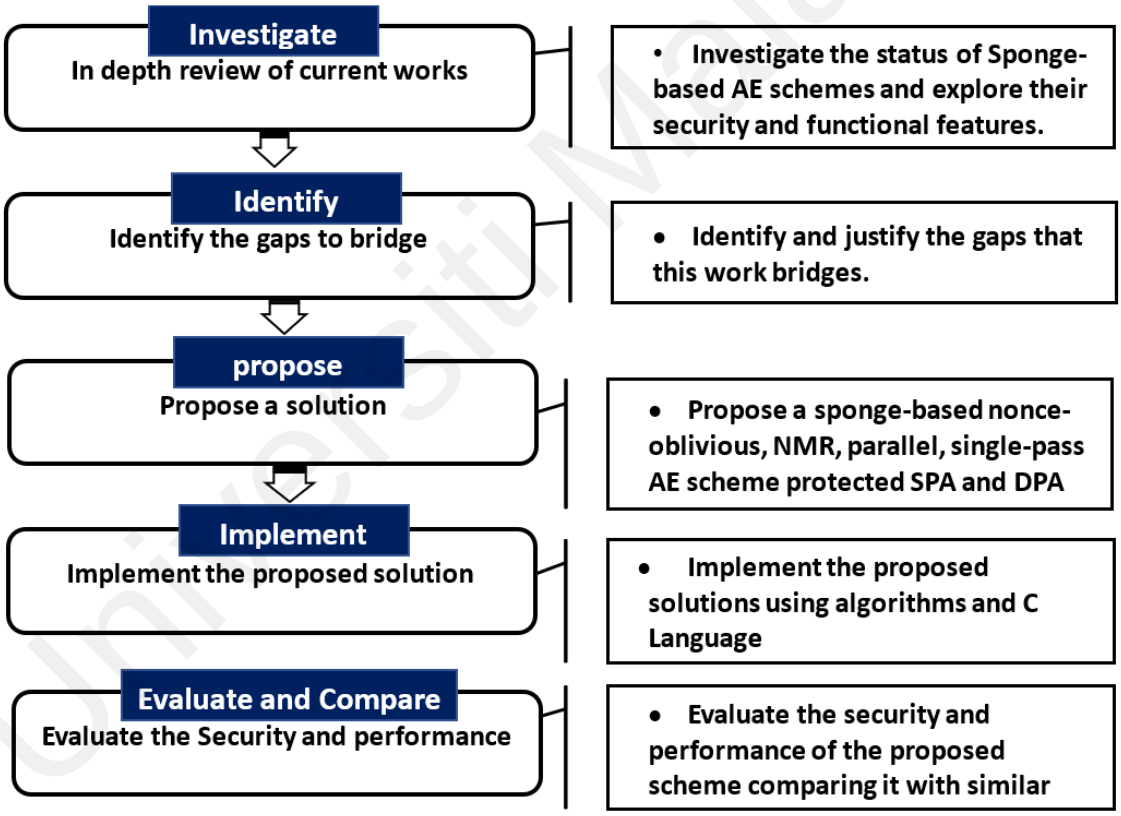

Figure 1. 3: The proposed methodology flow of activities.

The scheme's security is evaluated using game-playing theory according to PRP/PRF switching lemma. Finally, the performance of the scheme is evaluated by comparing it to similar sponge-based AE ones, including finalists of the CAESAR (Dobraunig, Eichlseder, Mendel, & Schläffer, 2016a) and NIST_LW competitions

(Dobraunig, Eichlseder, et al., 2019a; Dobraunig, Eichlseder, Mendel, et al., 2019). The performance metrics are displayed in charts and tables for clarity and easier comparisons. Figure 1.3 shows the proposed methodology in a data flow diagram.

1.6 The Layout of this Thesis

This thesis contains seven chapters, as depicted in Figure 1.4. This chapter (Chapter 1) introduces cryptography, focusing on AE schemes, specifically those based on sponge construction. The chapter also highlights the gaps, identifies research problems, and specifies the study objectives.

Chapter 2 provides a review of the state-of-the-art relevant literature on AE schemes. The chapter highlights the issues and challenges in AE schemes based on sponge construction. The chapter classifies existing works according to the lines of work they belong to and the security and performance feature they provide. Finally, it discusses possible enhancements and identifies outstanding issues for further study.

Chapter 3 describes the research problem, indicating the gaps in sponge-based AE schemes. It demonstrates how the existing parallel schemes lack the necessary protection against SPA and DPA; on the other hand, it highlights that the sponge-based schemes that protect against such attacks cannot process data in parallel. It also shows the need for nonce-hiding schemes in sponge-based AE schemes.

Chapter 4 presents the solution proposed to achieve the objective of the research. It describes the Parallel Sponge-based Authenticated Encryption with Side-channel Protection and adversary-invisible Nonces (PSASPIN). It explains the processes and the algorithms designed to achieve the study's objectives.

Chapter 5 explains the method of the security proof and how the performance is evaluated after implementing the scheme in the C programming language. Furthermore, it describes the test lab setup, software installations, and commands that are used for the test.

Chapter 6 discusses the findings, justifies the need for PSASPIN, and underlines its importance. Finally, the chapter goes over the other related works reevaluating the status quo and demonstrates where the contribution of this work fits in the existing works.

Chapter 7 concludes the thesis by reporting on the achievement of the objectives and outlining the possible limitations and future works.

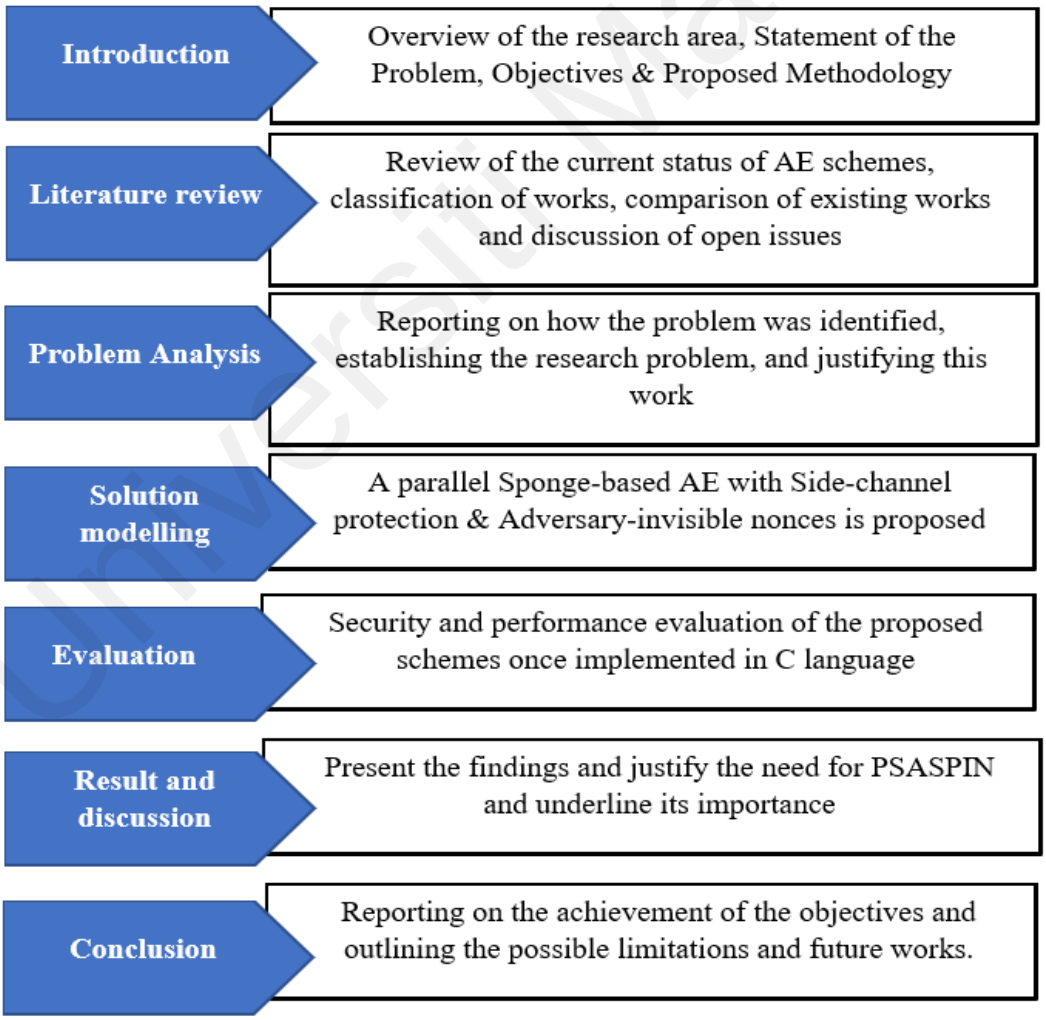

Figure 1. 4: The proposed methodology flow of activities.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter provides a review of the state-of-the-art relevant literature on AE schemes. First, the chapter highlights the issues and challenges in AE schemes based on sponge construction. Next, the chapter classifies existing AE works according to the lines of work they belong to and the security and performance feature they provide, discusses possible enhancements, and points out outstanding issues for further study. Finally, it explores existing works' security and performance characteristics to indicate gaps that need to be bridged.

2.2 Background

This section describes the concept of Authenticated Encryption, its birth, and its development, classifying it according to its lines of work and security and performance features they possess.

2.3 Authenticated Encryption

Confidentiality and integrity (and authenticity) have always been essential ingredients in security services offered by cryptography, and, together with "Availability," they constitute the CIA triad (confidentiality, integrity, availability) (Bellare & Namprempre, 2000b). Confidentiality assures that messages are accessed only by legitimate parties. In contrast, integrity ensures the messages are sourced from legitimate parties and that they were not tampered with while in transit or saved on disk (Jutla, 2001).

AE in asymmetric (public) key setting predated that of symmetric key for several years and was motivated by the work of Nyberg and Rueppel (1995) on a digital signature scheme, followed by other researchers (Horster, Michels, & Petersen, 1994; Hsu & Wu, 1998; Jian-Zhu & HuoYan, 2000; Kefei, 1998; Nyberg & Rueppel, 1995; Zheng, 1997). In addition to privacy and authenticity, AE in the public key setting provides nonrepudiation services that prevent the denial of communication by one of the parties.

For the public key setting, the security goals of cryptosystems were first defined by Bellare, Desai, Pointcheval, and Rogaway (1998); Dolev, Dwork, & Naor, 1991; Goldwasser & Micali, 1984; Rackoff & Simon, 1992). Goldwasser & Micali (1984) proposed an encryption scheme based on complexity theory. He declared that it was computationally infeasible for a polynomially-bounded attacker to deduct any useful information about the cleartext from the ciphertext.

The security goals were applied to the symmetric key settings for AE by Bellare & Namprempre (2000). They defined two notions of security related to AE concerning privacy and integrity and two concepts of integrity for symmetric key encryption schemes: integrity of plaintexts and integrity of ciphertexts, linking them to the standard notions of privacy in the same setting. They also analyzed the security of "generic composition methods" of AE schemes focusing on three of them Encrypt-and-MAC, MAC-then-Encrypt, and Encrypt-then-MAC.

The privacy goals for symmetric key schemes include indistinguishability (IND) and non-malleability (NM) under two security notions, chosen plaintext or adaptive chosen plaintext attack (CPA) and chosen ciphertext attack (CCA) attack, abbreviated as IND-CPA, IND-CCA, NM-CPA, and NM-CCA. According to Bellare et al. (1998), an

encryption's indistinguishability formalizes an adversary's inability to gain any information about the plaintext, having seen the corresponding ciphertext. Non-malleability (NM) formalizes that an adversary, given a ciphertext, should not be able to produce a different ciphertext such that the corresponding plaintexts are meaningfully related.

On the other front, three various attacks are mentioned in the literature: CPA, non-adaptive chosen-ciphertext attack (CCA1), and adaptive chosen-ciphertext attack (CCA2). Under CPA, the adversary gets a ciphertext of the target plaintext. In the public key setting, getting the public key is enough for the adversary to wage this type of attack; under the CCA1, the adversary has access to a decryption oracle and the public key. Bellare et al.(1998) discussed the possibility of 'mix-and-match' the privacy goals {IND, NM} and attacks {CPA, CCA1, CCA2} to give way to six notions of security, namely IND-CPA, IND-CCA1, IND-CCA2, NM-CPA, NM-CCA1, and NM-CCA2 (Bellare et al., 1998; Dolev, Dwork, & Naor, 1991).

Bellare and Namprempre (2000) considered two notions of integrity, the integrity of plaintexts (INT-PTXT) and ciphertexts. INT-PTXT requires that it be computationally infeasible to produce a ciphertext decrypting to a message the sender had not encrypted. INT-CTXT requires that it should be computationally infeasible to produce a ciphertext not previously generated by the sender, regardless of whether the corresponding plaintext is new. In both cases, the adversary can wage a chosen-message attack (Bellare, Desai, Jokipii, & Rogaway, 1997; Bellare & Namprempre, 2000a).

The two notions of privacy and integrity mentioned above do not automatically imply each other; for instance, a message with a strong MAC in the clear achieves

integrity but does not protect privacy. For comparison, Bellare et al.(2000) considered each notion coupled with the IND-CPA, the weakest form of privacy, focusing on INT-PTXT and IND-CPA against INT-CTXT and IND-CPA. They stated that the integrity of ciphertext, even when coupled with IND-CPA (the weakest form of privacy), remains the most robust notion of security, not only because it implies IND-CCA (the most potent form of privacy) but even more robust than it.

It is worth noting that non-malleability, under any form of the two attack forms (CPA or CCA), does not provide any form of integrity. Malleability only prevents the production of a ciphertext whose plaintext is significantly related to a specific challenge ciphertext. In contrast, integrity dictates that it is hard to produce new ciphertexts for new plaintexts regardless of whether they are related to plaintexts corresponding to existing ciphertexts (Bellare et al., 1997; Rackoff & Simon, 1992).

Traditionally, encryption-only schemes provided confidentiality (Encryption) and integrity/authenticity (MAC) as separate, independent services but were later proved to fail to protect even confidentiality only without ensuring integrity at the same time. That fact paved the way for the birth and prevalence of the notion of AE (Bellare & Namprempre, 2000b; Katz & Yung, 2001). Encryption provides confidentiality of messages under a secret key. The sender encrypts a confidential plaintext message and transmits the ciphertext; the receiver decrypts it, returning the plaintext. For authentication, the sender calculates a message authentication code (MAC) tag and attaches it to the message. The receiver employs the exact mechanism as the sender to ensure that the two tags (i.e., the received tag and the one computed) match. If they are identical, the receiver is assured that the message is authentic and accepts it; otherwise, a forgery is assumed, and the message is discarded.

Authenticated encryption simultaneously protects confidentiality and authenticity under a single secret key in the case of dedicated schemes or two separate keys in the case of generic composition modes. In AE schemes, the decryption may return either the plaintext or a special symbol ⊥ (bottom) instead of the plaintext, indicating a possible forgery attempt. The authenticity of a plaintext may also depend on some unencrypted pieces of data (Associated Data or the header), which are vital for routing the packets, such as TCP/IP information; Rogaway called it Authenticated Encryption with Associated Data (AEAD) in 2002 (Rogaway, 2002a).

An AE scheme has the following operations:

- **Encrypt and Authenticate**. Given the variable-length message, variable-length associated data (optional), and a fixed-length secret key, output the ciphertext and the corresponding fixed-length authentication tag. Both the message and associated data will be equally concatenated into fixed-length blocks prior to encryption. If the last message and/or associated data block is shorter than a complete block, then the block will be padded to have the length of a complete block.
- **Decrypt and Verify.** Given the ciphertext, secret key, and authentication tag, output the decrypted message if the tag is authentic; else, output an error message.

For authenticated encryption with associated data (AEAD), the Encrypt and Authenticate operation receives an additional input, authenticated but not encrypted data. The message can be of arbitrary length, but the secret key and tag have fixed sizes.

Although the intuitive method of designing an AE scheme is the 'generic composition,' combing a secure encryption scheme and a secure message authentication code (MAC) with two different keys, it was later proved that the wrong implementation could result in insecure schemes. An example of faulty implementations is PCBC mode

in Kerberos, as Rogaway et al. stated (Rogaway et al., 2001). There are three ways to generically combine a MAC and an encryption scheme (Bellare & Namprempre, 2000b):

- **Encrypt-and-MAC-plaintext:** This means encrypting the plaintext at first, then appending a tag (MAC) of the plaintext to the ciphertext. Given, K_e, K_m, M, we have: $\bar{\mathcal{E}}_{K_e,K_m}(M) = \mathcal{E}_{K_e}(M) \parallel \mathcal{T}_{K_m}$, the result is $C \parallel \mathcal{T}$. "Decryption/verification" is done by decrypting the ciphertext to get the plaintext, then recalculating the tag for verification.
- **MAC-then-encrypt:** This means to generate a MAC on the plaintext encrypted together with plaintext: $\bar{\mathcal{E}}_{K_e,K_m}(M) = \mathcal{E}_{K_e}(M \parallel \mathcal{T}_{K_m}(M))$, "Decrypt/verify" is then done by decrypting the ciphertext to get the plaintext and the tag and verifying the tag.
- **Encrypt-then-MAC:** $\bar{\mathcal{E}}_{K_e,K_m}(M) = C \parallel \mathcal{T}_{K_m}(C)\ where\ C = \mathcal{E}_{K_e}(M)$. Namely, encrypt the plaintext to get a ciphertext *C* and append a MAC of *C*. "Decrypt+verify" is performed by verifying the tag and then decrypting *C*.

According to Bellare and Namprempre (2000b), only the third one is guaranteed to be secure if the encryption scheme and the MAC are secure. This approach is natural and easy to analyze. However, it is also a bit slow since it requires two independent keys for encryption and authentication and is not robust to implementation errors.

The alternative to AE by generic composition is dedicated AE schemes. Soon after 2000, other AE schemes were proposed based on different structures, such as block cipher (Jutla, 2001), stream cipher (Zoltak, 2004), compression functions (Cogliani et al., 2014), cryptographic sponges (Assche, 2011), or keyed permutations. Simultaneously, other dedicated schemes are not based on any underlying primitives but are considered primitives on their own (Bilgin, Bogdanov, Knežević, Mendel, & Wang, 2013). Some of those schemes are two-pass schemes that make two passes through the data, one for confidentiality and the other for integrity. They mimic generic composition but use a single key instead of two independent keys; examples are (CCM and GCM). Other AE

schemes are single-pass schemes that run one time through the data, simultaneously achieving confidentiality and authenticity. Examples of single-pass schemes are: XCBC and OCB.(Bellare et al., 2003; Gligor & Donescu, 2002; Rogaway et al., 2001; Whiting et al., 2003)

2.3.1 Modeling Authenticated Encryption

AEAD can be seen as a function that takes four arguments: a secret key (*K*), a nonce (*N*), associated data (*A*), also called a Header (*H*), and plaintext (*P*)—as input, and produces a ciphertext (*C)* and an authentication tag (*T*) as an output—$E: K \times N \times H \times P \rightarrow C|T$—along with decryption $D: K \times N \times H \times C \rightarrow \{P, \perp\}$. Separated AE with associated data also features a verification algorithm $V: K \times N \times H \times C \times T \rightarrow \{\top, \perp\}$. The encryption algorithm is $E_K(N, H, P) = (C, T)$, and the decryption algorithm is $D_K(N, H, C) = P$ if *(C, T)* is valid; otherwise, it outputs $\perp$; the verification algorithm is $V_K(N, H, C, T) = \perp$ if a forgery is detected and decryption fails (Gligor & Donescu, 2002; Rogaway et al., 2001; Whiting et al., 2003). Figure 2.1 illustrates a schematic structure of an AE scheme.

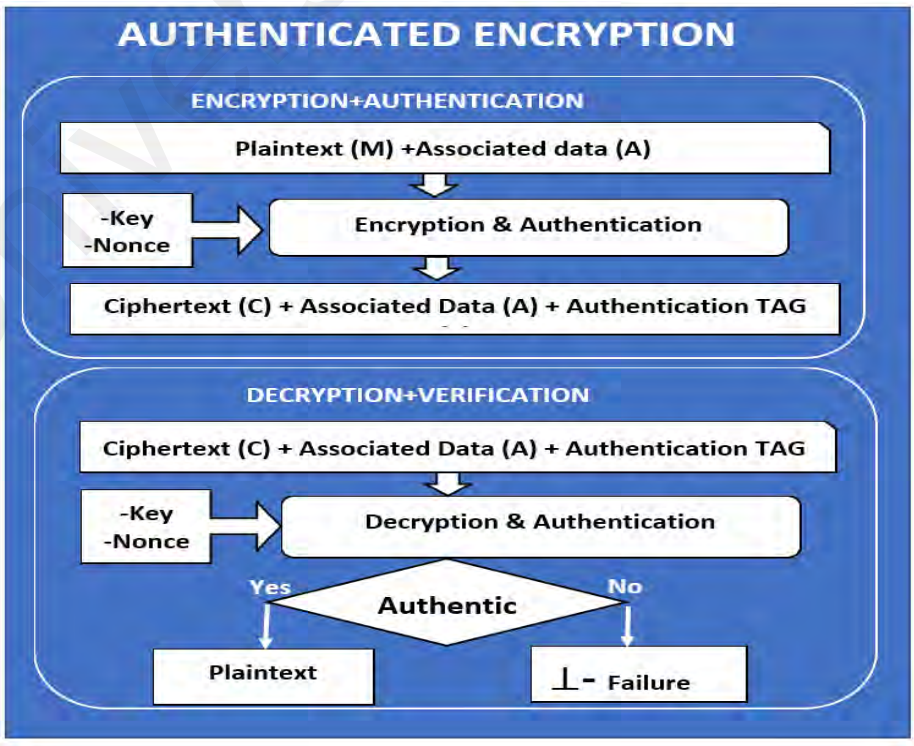

Figure 2. 1: A schematic structure of the AE scheme (M. A. Jimale et al., 2022)

2.3.2 A general Classification Framework for AE schemes

This section provides an overview of the classification of AE schemes according to the framework shown in Figure 2.2. The schemes are classified according to their categories, building blocks, security definitions, functional characteristics, modes or design approaches, and the cryptographic primitives they use. We discuss the classification themes in the following subsections.

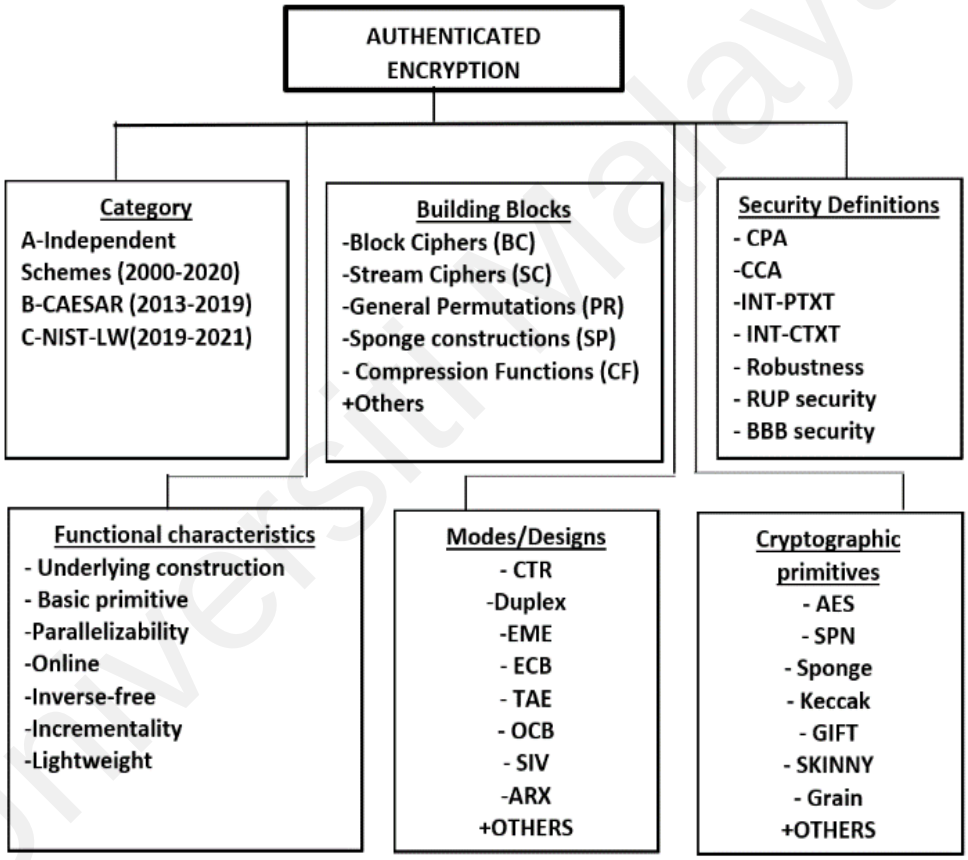

Figure 2. 2: A general classification framework of AE schemes

2.3.3 AE Categories

AE schemes can be categorized into three categories according to the line of work that they belong to, namely: (a) Independent schemes, (b) CAESAR competition schemes (c) NIST-LW competition schemes.

2.3.3.1 Authenticated encryption beyond CAESAR and NIST competitions (independent schemes)

AE schemes before the CAESAR competition, known as nonce-based authenticated encryption with associated data, were first defined by Bellare and Namprempre (2000b) and refined by Rogaway (2002a). They were designed to achieve semantic security by considering only deterministic schemes. The notation required the uniqueness of the nonce and stated that security was void if this condition was not fulfilled. Another important aspect of this notion is the associated data (AD)—pieces that should not be encrypted so that routing devices can forward packets correctly but need to be authenticated.

According to Datta and Nandi (2014), although it might be theoretically easy to implement nonce uniqueness, it isn't easy in practice. In many situations, implementation errors lead to misuse of nonces and the complete loss of confidentiality. For this reason, Rogaway and Shrimpton (2006) proposed better security (robustness) for cases in which nonces are reused, which marked the emergence of the notion of misuse-resistant (MR) authenticated encryption schemes (Rogaway & Shrimpton, 2006).

Schemes in this category laid the foundation for most AE schemes' security and functional characteristics. Security and performance improvement started, and new schemes with various features appeared for diverse applications. The proliferation of IoT devices and other resource-constrained devices further necessitated the creation of lightweight AE schemes that were crafted for that purpose. Despite the hard work done by researchers, there still seemed to be a need for improving AE schemes regarding security, performance, and robustness.

The continual refinement of AE schemes and the introduction of several enhancements to the original definitions and notions have led to the realization that further improvements were possible to the desirable features of AE schemes. This idea paved the way for the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR), which NIST and Dan Bernstein jointly initiated in 2013 (D. J. Bernstein, 2013) (competitions, 2019).

2.3.3.2 AE schemes in the CAESAR competition

Until the 1970s, cryptography was not in the public domain and was kept secret by governments for their private use; however, the need for encryption algorithms for commercial use was pressing and could not be ignored without economic consequences. As a result, in 1972, the US government called for the proposal of a standardized cipher for use in various applications; the winner was the Data Encryption Standard (DES) proposed by IBM. However, after concerns about DES security, mainly because of its 56-bit key length, the US National Institute of Standards and Technology (NIST) announced an open competition for a new Advanced Encryption Standard (AES). NIST chose Rijndael as AES (D. Bernstein, 2013; Paar & Pelzl, 2010). Likewise, in 2007 NIST announced an open competition for a new hash standard, SHA-3. NIST chose Keccak as SHA-3 based on the Sponge construction(Abed, Forler, & Lucks, 2016; D. Bernstein, 2013).

The open competitions mentioned above significantly boosted the cryptographic research community's understanding of cryptography and increased the confidence of public and commercial users and researchers. Moreover, the competitions' success in producing well-accepted cryptographic algorithms paved the way for even more similar

contests for solicitation of contributions, including the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR), following the practice of focused competitions in secret-key cryptography (D. Bernstein, 2013).

Despite the availability of various AE schemes that emphasize different aspects of confidentiality and integrity, many outstanding problems led to a loss or weakening of security, whereas other issues affected efficiency. The need to enhance AE schemes led to the idea of the CAESAR competition in 2013 (D. J. Bernstein, 2013). The question was: "Can we develop schemes that are as secure as AES-GCM and more efficient or ones that are as efficient but more secure, such that they can be widely adopted?" The organizing committee had received 56 submissions by 2014. After three rounds, the competition was concluded in 2019 with six winners for three use cases: lightweight applications for constrained devices (Dobraunig, Eichlseder, et al., 2016b; Wu, 2016), high-performance applications (Krovetz & Rogaway, 2016; Wu & Preneel, 2016), and defense in depth (Andreeva et al., 2016; Jean et al., 2016). The winners were Ascon (Dobraunig, Eichlseder, et al., 2016b), ACORN (Wu, 2016), OCB (v1.1) (Krovetz & Rogaway, 2016), AEGIS (Wu & Preneel, 2016), COLM, and Deoxys (Jean et al., 2016).

2.3.3.3 AE schemes in the NIST-LW competition

Because it is not practical to implement standard cryptographic algorithms on resource-constrained devices, researchers in the AE field were obliged to design lightweight versions of the schemes. Furthermore, devices like IoT sensors, RFID tags, and even more minor embedded chips need light AE schemes to protect the sensitive data they store or transactions they perform since they cannot accommodate the heavyweight algorithms (Li, 2017). Therefore, lightweight AE should consume less power and have less physical footprint than conventional cryptographic algorithms. However, one

potential drawback of lightweight cryptography is security weakness, which trade-offs implementation cost, speed, security, performance, and energy consumption. Therefore, the purpose of lightweight cryptography is to use less memory, less power, and fewer computing resources to secure resource-limited devices. The ISO/IEC 29192-1 standard specifies the properties of lightweight cryptography (ISO, 2019; Li, 2017).

With the experience of the CAESAR competition, researchers focused on AE applications in resource-constrained devices that could not benefit from the most prevalent schemes due to their resource intensiveness. This feeling led to setting up another competition in lightweight cryptography, the NIST-LW competition.

In August 2019, NIST published the requirements and evaluation criteria for submitting lightweight algorithms for evaluation and standardization. By February 2019, 57 submissions had been received; after eliminating one proposal, the organizers officially considered 56 submissions as candidates in Round 1 (NIST, 2020). After eliminating 24 candidates, 32 were announced in April 2019 as Round 2 candidates (NIST, 2020).

In March 2021, NIST announced ten finalists from the 32 candidates from Round 2 in the final portfolio for standardization: Ascon (Dobraunig, Eichlseder, Mendel, et al., 2019), Elephant (Beyne, Chen, Dobraunig, & Mennink, 2019), GIFT-COFB (Banik et al., 2019), Grain128-AEAD (Hell, Johansson, Meier, Sönnerup, & Yoshida, 2019b), ISAP (Dobraunig, Eichlseder, et al., 2019a), Photon-Beetle (Bao et al., 2019), Romulus (Iwata, Khairallah, Minematsu, & Peyrin, 2019), Sparkle (Beierle et al., 2019), TinyJambu (Wu & Huang, 2019), and Xoodyak (Daemen, Hoffert, Peeters, Assche, & Keer, 2019). Figure 2.3 depicts the categories and trends of AE from 2000 to 2021 in the SLR of (M. A. Jimale et al., 2022).

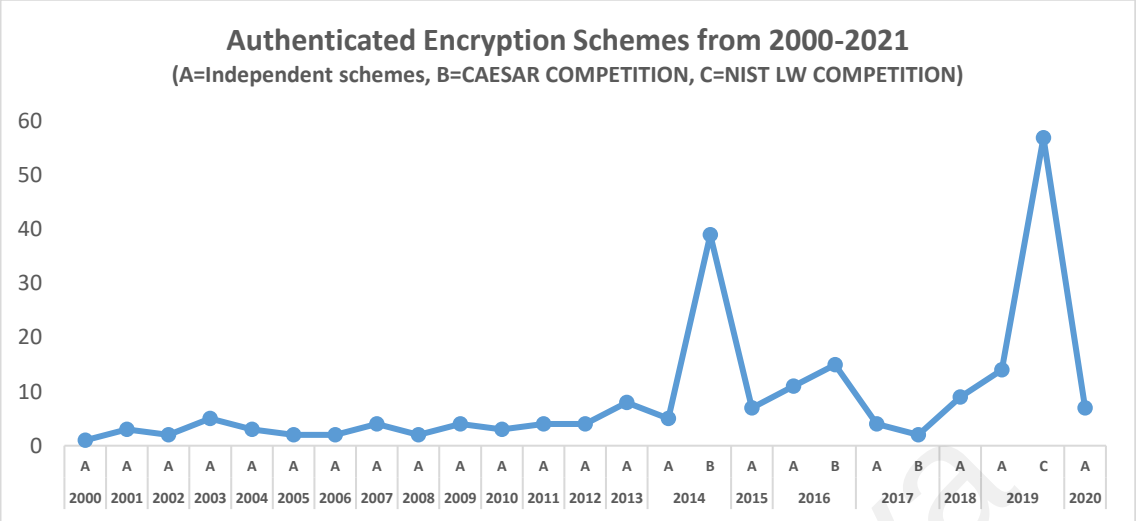

Figure 2. 3: Trends of AE schemes from 2000 to 2020 (M. A. Jimale et al., 2022)

2.3.4 AE Building Blocks

This section provides an overview of cryptographic structures used to build AE schemes.

2.3.4.1 Block cipher-based structure.

A block cipher accepts as input a plaintext block of fixed length and a secret key. A key scheduling algorithm takes the secret key and derives a series of round subkeys. The input plaintext is processed iteratively by a round function where one subkey is applied in each round. The final round outputs the corresponding ciphertext block equal in length to the input plaintext block. Typical block lengths are 64 and 128 bits, while the secret key ranges from 128 to 256 bits.

In response to NIST's call for proposals to find a single secure cryptographic algorithm for data protection, the most promising candidate was a block cipher that appeared in 1974 from a team of IBM researchers (Daemen & Rijmen, 2002; Paar & Pelzl, 2010). It was based on the first block cipher, Lucifer, which had appeared in the 1960s. Lucifer was, in turn, based on work done by Horst Feistel. Despite many concerns,

in 1977, the National Bureau of Standards (NBS) finally released all specifications of the modified IBM cipher as DES to the public (Paar & Pelzl, 2010). NBS chose it after a public invitation for submissions and some internal changes.

DES underwent major security scrutiny, and no significant weaknesses were found in it until 1990. Nonetheless, it was evident that a replacement was necessary as its key size was too small and thus was vulnerable to exhaustive key search attacks with increasing computing power at that time. For that reason, it was replaced by the AES in 1999 (Daemen & Rijmen, 2002). Unlike DES, AES is an iterative, based on the 'substitution–permutation network,' not a Feistel cipher. It comprises a series of related operations, some of which contain substituting inputs by specific outputs (substitutions), and others involve shuffling bits around (permutations), performing all its computations on bytes rather than bits (Paar & Pelzl, 2010).

AES resulted from an open competition and won among four other candidates: MARS, RC6, SERPENT, and Twofish. With its variable key sizes of 128,192, and 256 bits, AES remained secure against brute force attacks for several decades, and no practical analytical attacks against it have been known to date.

An AE scheme includes a dedicated block cipher specific to the scheme or uses readily available designs (often with some modifications). The most popular block ciphers used to construct AE schemes include the AES (Daemen & Rijmen, 2002), SKINNY (Beierle et al., 2016), and GIFT (Banik et al., 2017).

An extension to the traditional block cipher called tweakable block cipher additionally accepts a public input value called a tweak. The tweak allows for an easy way to invoke a different permutation of the block cipher without changing the key, somewhat akin to a counter baked inside the block cipher itself. The idea of a tweakable

block cipher dates back to the Hasty Pudding Cipher, a candidate in the AES competition (An, 2001). Kavun and Mihajloska later formalized the tweakable block cipher. The tweakable block cipher SKINNY (Beierle et al., 2016) is included in a new standard called the ISO/IEC18033-7 (ISO, 2022).

2.3.4.2 Stream cipher-based structure.

Stream ciphers encrypt bits individually by adding a bit from a key stream to a plaintext bit by taking a secret key of a fixed length to generate a keystream of variable length. Stream ciphers are designed to be fast and small and are often used for constrained resource environments that need lightweight algorithms. Stream ciphers can be used as a core primitive in authenticated encryption to achieve confidentiality and authenticity/integrity if the cipher is secure (Tahir et al., 2008). For instance, Ferguson et al. (2003) proposed the first AE based on a Helix stream cipher combined with a MAC to provide the AE functionality. It was designed to be efficient in hardware and software with low overhead making it suitable for small messages.

On the other hand, Furuya & Sakurai (2003) proposed a single-pass AE scheme based on a stream cipher based on ε-almost universal hash functions and proved its security according to the Real-or-Random paradigm (Furuya & Sakurai, 2003). Other researchers proposed NMR AE schemes based on stream cipher examples (Krovetz, 2014) and (Chakraborti & Nandi, 2014). Recent work by Campbell (2020) proposed a deterministic AE scheme based on SALSA and CHACHA stream ciphers for high security, performance, and easier deployment. The NIST_LW competition finalists contain one AE scheme, Grain-128, a lightweight stream cipher that is based on a Linear Feedback Shift Register (LFSR) design (Hell, Johansson, Meier, Sönnerup, & Yoshida, 2019a).

2.3.4.3 Permutation-based structure.

Permutation-based AE schemes use dedicated and keyless permutations as the underlying primitive. Schemes in this category do not use permutations in a sponge-like mode but apply other techniques like XOR, Encrypt XOR, Encrypt Mix Encrypt (EME) (Hoang, Krovetz, & Rogaway, 2014), and derivations of the Even-Mansour construction (Hoang et al., 2014; Wu & Huang, 2019). For example, TinyJAMBU is a NIST-LW competition finalist that is NMR and is based on keyed permutation, and Sparkle is another NIST-LW competition candidate (Alizadeh et al., 2014b; Wu & Huang, 2019). On the other hand, Schwaemm is another NIST-LW competition finalist based on the Sparkle permutation and Addition, Rotation, XOR (ARX) design (Dinu et al., 2016) to provide fast software encryption for all platforms while using as few CPU cycles as possible (Beierle et al., 2019). WAGE is another lightweight AE mode based on iterated permutation of 259 bits inspired by the Welch-Gong cipher operating on a unified duplex sponge mode to construct an AEAD scheme (Aagaard, AlTawy, Gong, Mandal, Rohit, et al., 2019).

2.3.4.4 Sponge-based structure.

Sponge functions, in cryptography, provide a way to generalize hash functions to more general arbitrary length functions. It is a simple iterated construction producing a variable-length input and variable-length output function based on a fixed-length permutation or transformation. The sponge function can also be used as a stream cipher providing a variety of functionalities with hash functions. (Assche, 2011; Bertoni, Daemen, Peeters, & ASSCHE, 2011; Jovanovic, Luykx, & Mennink, 2014a).

According to Bertoni et al. (2011), the roots of Sponge functions go back to the design stage of another construction, RADIOGATUN (Bertoni, Daemen, Peeters, &

Assche, 2006). It is a hash function with variable-length input and a variable-length output, where the designers had the problem expressing pertinent security claims. Because in the case of fixed-input, fixed-output functions, the security claims are equivalent to that of the Random Oracle with its output truncated to *n* bits, implying the traditional security boundary that withstands $2^{n/2}$ for collision and 2^{n} for second preimage attacks. For constructions like RADIOGATUN with variable-length input and variable output, the conventional security claims of the hash functions are not applicable. The goal was to establish the security of a hash function that behaves like the Random Oracle, except that it might have some inner collisions (Assche, 2011; Biryukov & Khovratovich, 2014; Chakraborty, Jha & Nandi, 2019).

Sponge functions allow modeling a finite memory that any cryptographic construction can access. The security of a random sponge function can simulate that of a random oracle, except for the limitations imposed by the finite memory status providing an alternative to the random oracle model for declaring security claims (Bertoni et al., 2011).

Together with its sister construction, the duplex construction, the sponge construction is employed to construct many symmetric cryptography functionalities, including hashing, pseudo-random bit generators, key derivation, encryption, message authentication code (MAC) computation, stream ciphers, and AE (Assche, 2011). This flexibility enables the users to achieve a lot of functionalities from single fixed-size permutations, making the implementation more straightforward and flexible. Furthermore, the designers of cryptographic structures may also find the development of a strong permutation beneficial without worrying about the complexities, like key

scheduling, associated with other primitives like block ciphers (Assche, 2011; Bertoni et al., 2011; Bogdanov et al., 2011).

The most used form of keyless permutation is sponge construction. Specific schemes use keyless permutations in a sponge-like mode of operation, like the Keccak-f permutation used in the SHA3 hash function, whereas others rely on dedicated permutations (Bogdanov et al., 2011; "SHA Zoo," 2013).

i) The Sponge Construction

The sponge construction builds a function $SPONGE[f, pad, r]$ with the domain $\mathbb{Z}_2^*$ and co-domain $\mathbb{Z}_2^\infty$ using a fixed-length permutation (or transformation), a sponge-compliant padding rule, 'Pad,' and a bit rate *r.* It operates on a state of *b* bits at bit-rate *r* and capacity *c*, where $b = r+c$ (Bertoni et al., 2011). The sponge first absorbs its inputs, block by block, before processing and squeezing them out afterward. After initializing all bits of the initial state to zero, in the absorbing phase, the padded *r*-bit input message blocks are XORed with the first *r*-bit of the state interweaved by the application of the *f* function, which can be either a permutation or a transformation. The operation switches to the squeezing phase when all input message blocks are absorbed. In the squeezing mode, the first *r* bits of the state are returned as the output block interweaves with the *f* function application. The last *c* bits of the state (the capacity) are used as a security parameter and are not affected by the input or the output processing (Assche, 2011; Bertoni et al., 2011; Bertoni, Daemen, Peeters, & Assche, 2012).

ii) The duplex construction

The duplex construction is similar to the sponge construction that it builds a function $DUPLEX[f, pad, r]$, uses a fixed-length permutation (or transformation),

padding tule and a bit-rate *r*. But the difference is that the sponge function is stateless between calls, and the duplex construction produces an object that can accept calls that take input strings and results in output strings that depend on all messages inputs received so far (Assche, 2011). Figure 2.4 depicts the sponge construction, and Figure 2.5 illustrates The Duplex Mode of the Sponge Construction.

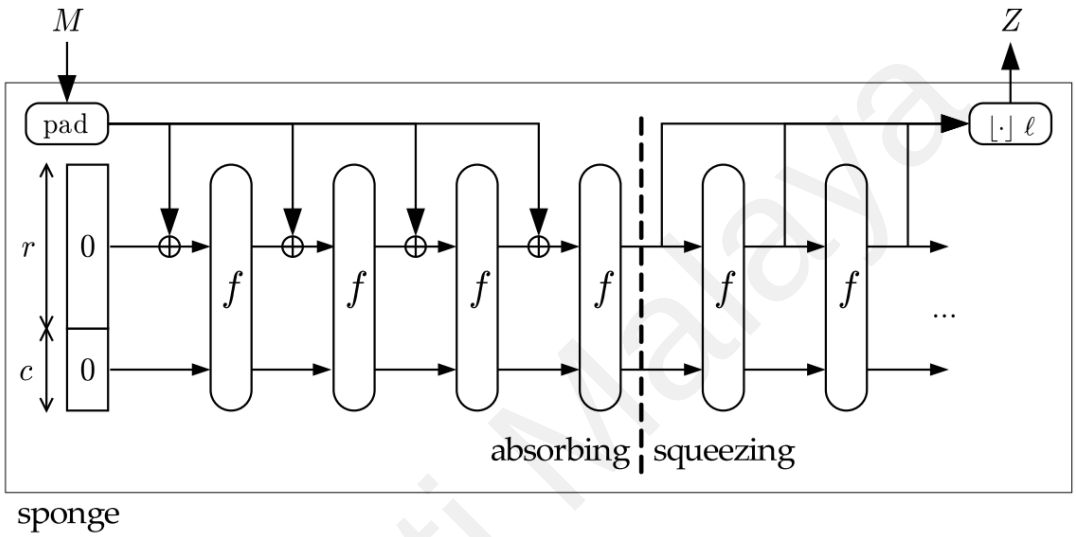

Figure 2. 4: The Sponge Construction, Source: Bertoni et al. (Assche, 2011)

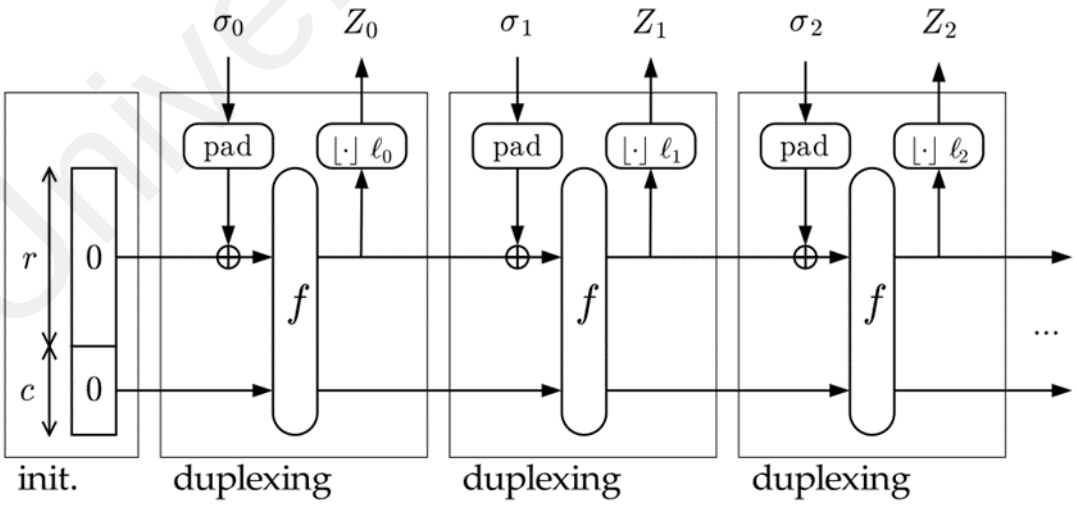

Figure 2. 5: The Duplex Mode of the Sponge Construction

2.3.4.5 Hash function/compression function (CF).

Some AE schemes use compression functions. A hash function maps strings of arbitrary length to a fixed-length output or hash value. Any change to even one bit of the input should change the entire output and allow an adversary to find a collision, preimage, and second preimage. For instance, Cogliani et al.(2014) proposed the Offset Merkel-Damgård (OMD) mode using a compression function to construct an online and inverse-free AEAD where the privacy and authenticity operations are tightly coupled to reduce the number of compression function calls (Cogliani et al., 2014).

2.3.4.6 Others.

Some AE schemes are dedicated to using a structure based on primitives that do not fall into the above categories, like the Type-3 Feistel schemes. For example, the Artemia, a Round 1 CAESAR competition scheme, is a dedicated, online AEAD scheme that uses a permutation-based on JHAE mode, which is provably secure in the ideal permutation model (Alizadeh, Aref, & Bagheri, 2014a). Other schemes are based on hybrid primitives (HB) with structures that have the characteristics of more than one cipher, like the stream cipher and the block cipher. For instance, the Hummingbird-2 AE scheme provides encryption with a 128-bit secret key and an initialization vector and optionally produces a tag for each processed message. Hummingbird-2 and its predecessor Hummingbird-1 were designed for implementation in low-end microcontrollers like wireless sensors and RFID tags (Engels et al., 2011; Harris, 2014). Finally, the Enchilada is another hybrid first-round CAESAR competition candidate that uses ChaCha and Rijndael with ten rounds and a 128-bit block size; its authors claim that its encryption is identical to that of AES-128 with modifications of key schedule and adding key whitening (Harris, 2014).

Dedicated schemes (DE) that are not based on any symmetric key primitive have also been developed, although there are few. For instance, Peña & Torres proposed an AE scheme based on finite automata using the 'encrypt-then-MAC' generic composition method based on a symmetric encryption scheme and a message authentication code built from a finite automaton (Peña & Torres, 2016). In addition, Xin et al.(2004) developed a dedicated AE scheme using quantum states for parties that use classical bit string as their secret keys. The authors claim that the scheme is computationally and statistically secure (Xin et al., 2004).

2.3.5 Security-related Definitions

Authenticated encryption is intended to protect confidentiality and authenticity and is assumed secure only if it satisfies the relevant notions of security. This section provides a general description of the security relations, definitions, and assumptions about AE schemes. First, we discuss provable security and indistinguishability in our adversarial models. Then we consider general security notions relating to confidentiality and authenticity following Rogaway and Shrimpton (2006), Bellare and Namprempre (2008), and Bellare et al. (2001). Finally, our discussion considers a security model where a computationally bounded adversary $\boldsymbol{A}$ interacts with a given set of oracles ($\boldsymbol{O}$), acting like a black box to the adversary. For an AE scheme to be secure, $\boldsymbol{A}$'s advantage in all cases should be negligible.

2.3.5.1 Provable Security

Provable security, also known as reductionist security, is methodology designers use to ensure that a scheme is secure relative to particular security definitions against a given adversarial model under specific assumptions. Cryptographers provide security proofs in a theoretical model that abstracts their underlying primitive such as PRF or PRP

(Bellare & Rogaway, 2005), primarily in the Standard or Random Oracle Model. In the Standard Model, the adversary is limited by the amount of time and computer power it has. The Random Oracle Model assumes that pseudorandom functions are replaced by random oracles that return random values upon invocation (Black, 2004). See Figure 2.6 for details.

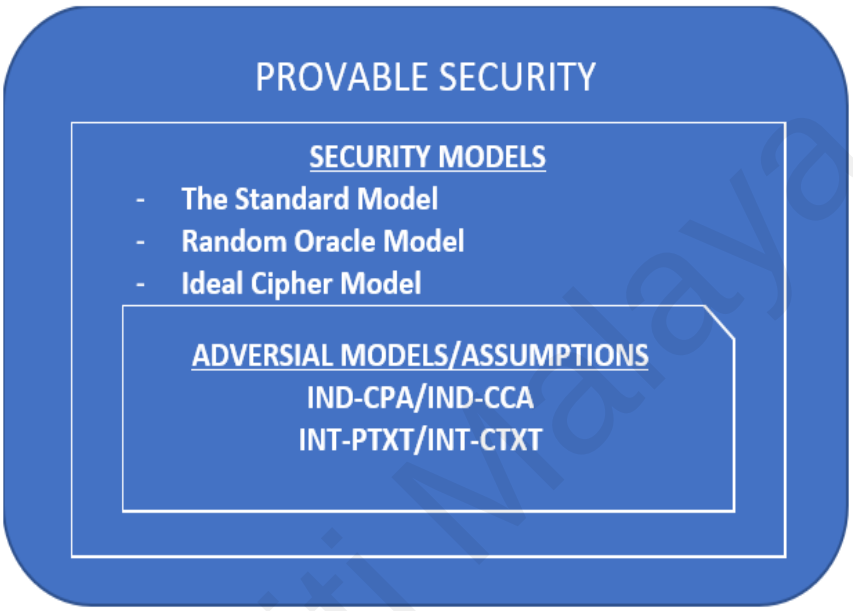

Figure 2. 6: Security definitions model for AE schemes

(A) Indistinguishability of ciphers (IND)

Computational indistinguishability is an essential concept in cryptography that requires that an adversary with defined capabilities and resources cannot distinguish between two ciphertexts, one encrypted with the cipher in question and another from an equal-length random string. To formalize it, we resort to the concept of distributions $X = \{x_k\}_{k\subseteq\mathbb{N}}$, one for each security parameter value (Kamara, 2017).

Definition 1. Two sets, S_1 and S_2, are indistinguishable if for all adversaries A that outputs a bit: $|Pr[A(X_1) = 1] - Pr[A(X_2) = 1]| \leq negl(k)$ (2.1)

The definition guarantees that no efficient adversary can tell apart with non-negligible probability given a sample from X_1 or X_2 because the output is either 1 or 0 with equal probabilities.

(B) Pseudorandom Functions

A pseudorandom function (PRF) is a deterministic function sampled uniformly at random from a finite function space that takes a key K, an input x. It gives an output y that is indistinguishable from a truly random function. An adversary that can provide input and get an output to and from either a PRF or a truly random function cannot distinguish them with non-negligible probability. For some integers $k, l, L \geq 1$ of a function $F: K \times D \rightarrow R, (K = K, D = \{0,1\}^l, R = \{0,1\}^L)$. The function F_n can be in one of two worlds. In the real world, the adversary is interacting with a random instance of F, and in the random world, it is interacting with a random function F_n with the domain R. To succeed in the experiment, the job of the adversary is to distinguish between these two worlds with probability more than ½, with the advantage being a number between 0 and 1.

Definition 2. (Pseudorandom Functions): a Function $F: \{0,1\}^k \times \{0,1\}^l \rightarrow \{0,1\}^l, for\ l = l(k) = poly(k)$, is a pseudorandom function if for all PPT adversaries A,

$$|Pr[A^{F_K(.)}(1^k) = 1] - Pr[A^{f(.)}(1^k) = 1]| \leq negl(k) \quad (2.2)$$

Where $K \xleftarrow{\$} \{0,1\}^k$, and f is chosen at random from the set of functions from $\{0,1\}^l\ to\ \{0,1\}^l$

(C) Pseudorandom Permutations

A pseudorandom permutation (PRP) is a bijective PRF that adversary *A* cannot distinguish from a random permutation. A PRP is efficient if both the permutation and its inverse can be computed efficiently in a polynomial time. We refer to a strong notion of security as Strong PRP when we mean an indistinguishable permutation, even when the adversary has access to both the permutation *(P)* and its inverse *(P^{-1})*. (Bellare & Rogaway, 2005; Kamara, 2017).

Definition 3. (A Strong Pseudorandom Permutation). A function $P: \{0,1\}^k \times \{0,1\}^l \rightarrow \{0,1\}^l \; for \; l = l(k) = ply(k)$ is a strong pseudorandom permutation if for all PPT adversary *A*,

$$|Pr\,[A^{P_K(.),P_K^{-1}(.)}(1^k) = 1] - Pr[A^{f(.),f(.)^{-1}}(1^k) = 1]| \leq \; negl(k) \qquad (2.3)$$

Where $K \xleftarrow{\$} \{0,1\}^k$ And f is chosen uniformly randomly from the set of permutations (Aagaard, AlTawy, Gong, Mandal, & Rohit, 2019).

Following the approach of Abel et al. (Abed et al., 2016), we describe the advantage of adversary *A* against CPA and CCA.

Definition 4. (PRP-advantage under CPA). Let $F: K \times D \rightarrow D$ be a family of functions and *A* an adversary which interacts with an oracle and returns a bit; the PRP-advantage of A is given by:

$$ADV_F^{PRP-CPA}(A) = |Pr[Real_F^A \Rightarrow 1] - P\,r[Perm_D^A \Rightarrow 1]| \qquad (2.4)$$

Definition 5. (PRP-advantage Under CCA). Let $F: K \times D \rightarrow D$ be a family of functions, and *A* be q, q, μ bounded adversary, where *t* is time complexity, *q* is the number of queries, and μ is the total length of all adversarial queries. It is worth noting that PRP-CCA secure

scheme is also PRP-CPA secure, but the reverse is not true. The PRP-CCA advantage of A is given by:

$$ADV_F^{PRP-CCA}(A) = |Pr[Real_F^A \Rightarrow 1] - P\,r[Perm_D^A \Rightarrow 1]| \quad (2.5)$$

Definition 6. (IND-CPA and IND-CCA). Let $\Pi\ (K, E, D)$ be an authenticated encryption scheme, and A as a *t,q,l* bound adversary that can interact with the real world (Real) and the random world (Random) with complexity time *t*, making *q* queries of total length *l*. In the IND-CPA case, A can access an encryption oracle; in the IND-CCA case, it can also have a decryption oracle. The adversary's goal is to distinguish between the two worlds. In both cases, A's advantage, with reasonable resources, should be negligible.

$$ADV_\Pi^{IND-CPA}(A) = Pr\left[K \overset{\$}{\leftarrow} K\colon (A)^{E(.,.)} \Rightarrow 1\right] - P\,r\left[(A)^{\$(.,.)} \Rightarrow 1\right] \quad (2.6)$$

$$ADV_\Pi^{IND-CCA}(A) = Pr\left[K \overset{\$}{\leftarrow} K\colon (A)^{E(.,.)} \Rightarrow 1\right] - P\,r\left[(A)^{\$(.,.)} \Rightarrow 1\right] \quad (2.7)$$

Definition 7. (INT-PTXT and INT-CTXT). Let $\Pi\ (K, E, D)$ be an AE scheme, and $A_{int\text{-}ptxt}$ and $A_{int\text{-}ptxt}$ be *t,q,l* bound adversaries that have access to Encryption oracle $E_k(.,.)$ and Decryption oracle $D_K(.,.)$. Adversary $A_{\text{int-ptxt}}$ wins if it submits to the decryption oracle a ciphertext that does not match a plaintext previously queried to the encryption oracle. $A_{\text{int-ctxt}}$ wins if it submits a valid ciphertext not previously produced by the encryption oracle to the encryption oracle. The scheme Π is considered secure if the advantage of $A_{\text{int-ptxt}}$ and $A_{int\text{-}ctxt}$ is negligible.

$$ADV_\Pi^{int-ptxt}(A) \leq Pr\ [K \overset{\$}{\leftarrow} K\colon (A)^{E(.,.),D(.,.)} \Rightarrow Forges \quad (2.8)$$

$$ADV_\Pi^{int-ctxt}(A) \leq Pr\ [K \overset{\$}{\leftarrow} K\colon (A)^{E(.,.),D(.,.)} \Rightarrow Forges \quad (2.9)$$

Shrimpton (Shrimpton, 2004) introduced a variation of the standards Chosen Ciphertext security, combining IND-CPA and IND-CTXT into a single notion known as IND-CCA3.

Definition 8. (IND-CCA3). Let $\Pi = (K, E, D)$ be an AE scheme, and A *a t,q,l* bound adversary that has access to encryption in the real world, but in decryption oracle in the random world, we replace the decryption oracle in the random world with an oracle that always returns INVALID ($\perp$). We assume that A never asks queries to which it already knows the answer.

$$ADV_{\Pi}^{ind-CCA3}(A) = Pr\left[K \xleftarrow{\$} K: A^{E_K(.),D_K} \Rightarrow 1\right] - Pr\left[A^{E_K(\$|.|),\perp(.)} \Rightarrow 1\right] \quad (2.10)$$

Shrimpton in (Shrimpton, 2004) demonstrated that the IND-CCA3 advantage of an adversary A on an AE is upper-bounded by the total of the maximal of A's advantage over Π INT-CTXT advantage and the maximal of A's advantage over Π IND-CPA advantage. So, the IND-CCA3 advantage over all adversaries A that run in time t and make q queries of length l is given by:

$$ADV_{\Pi}^{CCA3}(q, t, l) \leq ADV_{\Pi}^{IND-CPA}(q, t, l) + ADV_{\Pi}^{INT-CTXT}(q, t, l) \quad (2.11)$$

(D) Security of online AE schemes

Bellare et al. (Bellare et al., 2001) introduced the study of online ciphers, which can take input of large-size plaintext and varying lengths and output the j^{th} block of the ciphertext after having processed only the first j blocks of the plaintext, and they provided security definitions for them. So we define CCA3 security for the online AE schemes (OCCA3) following the approach of Abed et al. (Abed et al., 2016).

Definition 9. (OCCA3 security). Let $\Pi = (K, E, D)$ be an online AE scheme and let $P \xleftarrow{\$} Operm_n$ be a random online permutation, then define an adversary A such that:

$$ADV_{\Pi}^{OCCA3}(A) = Pr\left|\left[K \xleftarrow{\$} K: A^{E_K(.),D_K} \Rightarrow 1\right] - \Pr\left[A^{O^P(.,.)\perp(.,.,.)} \Rightarrow 1\right]\right| \quad (2.12)$$

And $ADV_{\Pi}^{OCCA3}(q,t,l) = max_A\{ADV_{\Pi}^{OCCA3}(A)\}$, the maximum advantage over all OCCA3 *q,t,l* bounded adversaries, that as *q* number of queries of *l* blocks long with time complexity of *t*. Based on the definitions above and those in (Shrimpton, 2004), (Bellare et al., 2001), and (Abed et al., 2016), we can claim that:

$$ADV_{\Pi}^{OCCA3}(A) \leq ADV_{\Pi}^{OPRP-CPA}(q,t,l) + ADV_{\Pi}^{INT-CTXT}(q,t,l) \quad (2.13)$$

Definition 10. (OPRP-CCA security). Let K be a *k* bit key, *P* a random permutation, $\psi: \{0,1\}^k \times (\{0,1\}^n)^* \rightarrow$ be an online cipher. OPRP-CCA advantage of an adversary *A* can be defined as follows:

$$ADV_{\psi}^{OPRP-CCA}(A) = \left|Pr\left[A^{\psi_K(.),\psi_K^{-P}(.)} \Rightarrow 1\right] - Pr\left[A^{P(.),A^{-P}(.)} \Rightarrow 1\right]\right| \quad (2.14)$$

Then we can define $ADV_{\psi}^{OPRP-CCA}(q,t,l)$ as the maximum advantage over all OPRP-CCA adversaries making *q* number of queries of length *l* with a time of complexity of *t*.

2.3.5.2 Nonce-based authenticated encryption.

An AE scheme may rely on a user-supplied nonce (number used once), an input to the AE scheme that is not supposed to be reused to encrypt different plaintexts under the same key (Rogaway, 2002a; Rogaway et al., 2001). Why do we require that nonces be unique? Imagine that Bob receives an encrypted document sent by Alice. If Alice wishes to send the same data again to Bob, if the same nonce is reused and an adversary is tapping the communication, they can infer that the two documents are the same. Such knowledge benefits an adversary and can be exploited in an attack.

Nonces do not need to be random; they just need to be different for each subsequent use. Examples include a counter that is increased with every new encryption.

As nAE schemes do not handle nonce generation, implementors must ensure that the nonces are correctly generated and used (Gueron & Lindell, 2015a).

(A) Nonce misuse-resistant AE (MRAE)

The dominant belief about symmetric encryption schemes was that they should be probabilistic or stateful before Rogaway proposed a deterministic scheme using an additional input, an Initialization factor (IV). He stated that the user, not the encryption algorithm, was responsible for flipping coins to maintain the state in the encryption and decryption processes (Rogaway, 2004b). The author was interested in the case where the IV was a nonce, a value used at most once per session. Nonces' main advantage is preventing predictability and replay attacks, among others.

Despite the significant contribution of nonces in boosting privacy and authenticity, there are times when nonces usage could give a false sense of security since their mishandling could derail the security while designers and implementers are unaware. Application developers are responsible for determining how nonces are generated. However, such practice is prone to misuse because reusing nonces (intentionally or otherwise) can have dire consequences (Gueron & Lindell, 2015b; Rogaway & Shrimpton, 2006).

There are several examples in which nonce misuse weakened or destroyed the security of schemes. Furthermore, various protocols and applications have been violated due to the mishandling of nonces. Examples include Wired Equivalent Privacy (WEP) (Borisov et al., 2001), WinZip (Kohno, 2004), Microsoft Office (Wu, 2005), and Wi-Fi-protected access (WPA) 2 (Vanhoef & Piessens, 2017). Therefore, it is desirable to have AE schemes that provide a reasonable level of protection in case of such misuse.

To address this concern, in 2006, Rogaway and Shrimpton (Rogaway & Shrimpton, 2006) proposed the notion of a nonce misuse-resistant AE (MRAE). An MRAE scheme ensures an acceptable level of security even though nonces are repeated. In this regard, they proposed a deterministic AE, which is resistance to IV misuse when used differently from what is intended by the scheme. If a scheme NMRAE, when nonces are repeated, the integrity is protected, and privacy is violated only to the extent that one could tell if the plaintext is equal to a prior and exposed if the message and its header had been used with that particular IV. (Andreeva et al., 2013; Rogaway & Shrimpton, 2006).

Most of the NMRAE schemes in the literature are based on the block cipher. Andreeva et al. (2013) proposed the first sponge-based AE, which is an NMRAE (Andreeva et al., 2013), followed by others, including Gligoroski, Mihajloska, Samardjiska, Jacobsen, Jensen, et al. (2014), Agrawal et al. (2015), Andreeva, Bilgin, et al. (2014) and Cassiers, Guo, Pereira, Peters, and Standaert (2019). Only 5 of the 39 sponge-based schemes in our systematic review (M. Jimale et al., 2022) qualified as NMRAE.

(B) Nonce-oblivious Authenticated Encryption

Cryptographic algorithms are defined with syntaxes, security definitions, and assumptions. Security requirements evolved from requiring privacy to privacy and integrity, leading to AE's birth. The syntax must be either randomized or stateful. Rogaway et al. (2001) first proposed that, for algorithms to be stateful, it is necessary to use an additional input value, a nonce, which is supposed to be unique. The nonce idea evolved into the current form proposed by Rogaway (2002a); (Rogaway, 2004b; Rogaway et al., 2001).

Despite the benefits that nonces brought to strengthen security, they have also become a tool to spoil nonce-based encryption schemes' protection; their security claims hold as long as nonces are unique. The valid nonces format and the way to transmit them also stimulated a hot debate among the research community. Some claim nonces can be any text or value, like counters, and can be sent in the clear to the receiver along with ciphertext (McGrew, 2008; Rogaway, 2002a, 2004b; Rogaway, Bellare, & Black, 2003). For instance, RFC 5116 does not see any problems sending the nonce in the clear as long as the whole nonce is available to the decryptor (McGrew, 2008). Conversely, Bellare et al. (2019b) stated that sending nonces in clear or using values like Device-ID could entail serious security breaches.

Bellare et al. (2019) indicated the existence of a gap between the theory and practice in handling the nonces, significantly how it is created, saved, or transformed to the receiver/decrypter. They raised serious concerns about how that could result in confidentiality concerns in practical implementations. For example (Rogaway, 2002b) indicates that however the nonce is transmitted to the receiver is outside of the model, or the sender could send it out-of-the-band (Rogaway, 2004b). However (Bellare et al., 2019a) stressed that the implementation could not ignore this and must find a way (in the model) to send the nonce to the receiver. According to the authors, the source of the prevailing belief was a quote from (Rogaway et al., 2001), claiming that the nonce could be transmitted in the clear. However, Bellare et al. (2019) stated that sending the nonces in the clear could destroy message confidentiality. Furthermore, they underlined that the negative impact of visible nonce could go further because innocuous nonce values like counter and device identifiers could reveal information about the systems since nonce are meta-data (Bellare et al., 2019a).

To bridge that gap, they proposed an approach to prevent the nonce burden and its potential mishandling by hiding it like the message, thus eliminating the nonces from the decryption algorithm of the schemes so that the receiver does not have to worry about handling nonces anymore. Finally, Bellare et al. (2019b) demonstrated simple ways to turn traditional nonce-based authenticated encryption schemes into nonce- oblivious for the block cipher-based AE schemes. They suggested a nonce hiding (HN) syntax for AE schemes. They concretized their proposal in the context of Block cipher-based schemes defining several options for processing and transmitting the nonces and defining the level of security they targeted (HN1 to HN5 transforms), as shown in Table 2.1.

Table 2.1: Hide Nonce schemes as proposed by (Bellare et al., 2019a)

NBE2 scheme	**AE2-security provided**	
	Basic	Advanced
HN1[SE1, F]	Yes	Yes
HN2[SE1, ℓ, E, Spl]	Yes	Yes if $\ell \geq 128$
HN3[SE1, F]	Yes	No
HN4[SE1, ℓ, F]		Yes
HN5[TE, ℓ, ℓ_z]		Yes

2.3.5.3 Release of unverified plaintext (RUP)

AE aims to protect confidentiality and authenticity/authenticity together. The decryption happens in two steps: to verify the integrity and to recover the plaintext. In the classical model of AE, the plaintext should be available only after the verification is complete. However, this requires storing the data before validation occurs, which is not feasible in resource-constrained environments like IoT devices and smart cards. Despite

the validity of the classical model of AE, there are circumstances where the release of some plaintext before verification is indispensable. Examples are when storage capacity is scarce and when real-time processing is required. In addition to that (Andreeva, Bogdanov, et al., 2014) stated that using dedicated schemes secure against the release of unverified plaintext can boost efficiency.

RUP security is essential for AE schemes for its intensive use in lightweight cryptography, but only a few schemes in the literature provide it. For instance, only 20 of 217 articles in our systematic review (Jimale et al., 2022) supported RUP security. In practical terms, a two-pass AE scheme can be used to avoid releasing unverified plaintext into a device with insecure memory, one pass for verification of MAC and another for plaintext recovery. However, a single-pass AE scheme is enough if it enjoys RUP security (Andreeva, Bogdanov, et al., 2014; Tsang & Smith, 2008).

According to Andreeva, Bogdanov, et al. (2014), a typical AEAD scheme should allow releasing the resulting plaintext before the verification process when performing decryption; otherwise, the application must allocate memory to store unverified plaintexts, which may not be tolerable in resource-constrained environments. In addition, an AEAD scheme is secure under the RUP if the released information does not help an adversary forge valid ciphertexts or decrypt valid messages (Agrawal et al., 2015; Andreeva, Bogdanov, et al., 2014; Chang et al., 2020).

Snooping the released plaintext to deduce meaningful information is not the only option for attackers. They could also study the properties of the plaintext through other means like the padding oracle, where the existence of an error or the absence of an acknowledgment could be helpful to the adversary. One example was demonstrated by Vaudenay (2002) mounting a padding oracle on the current version of OpenSSL that day,

exploiting timing differences in the TSL decryption process (Andreeva, Bogdanov, et al., 2014).

2.3.5.4 Security Beyond the Birthday Bound (BBB)

The birthday bound for encryption or MAC algorithm describes that with an n block length, an adversary will succeed with a forgery after $2^{n/2}$ queries. If one can prove that a cryptographic construction guarantees security beyond this birthday limit, for instance, if an adversary needs $2^{n/3}$ encryption or MAC queries to break it, one can claim protection beyond the birthday bound (BBB) security (Minematsu, 2009). Figure 2.8 presents the developments of security-related properties of AE schemes from 2000 to 2021.

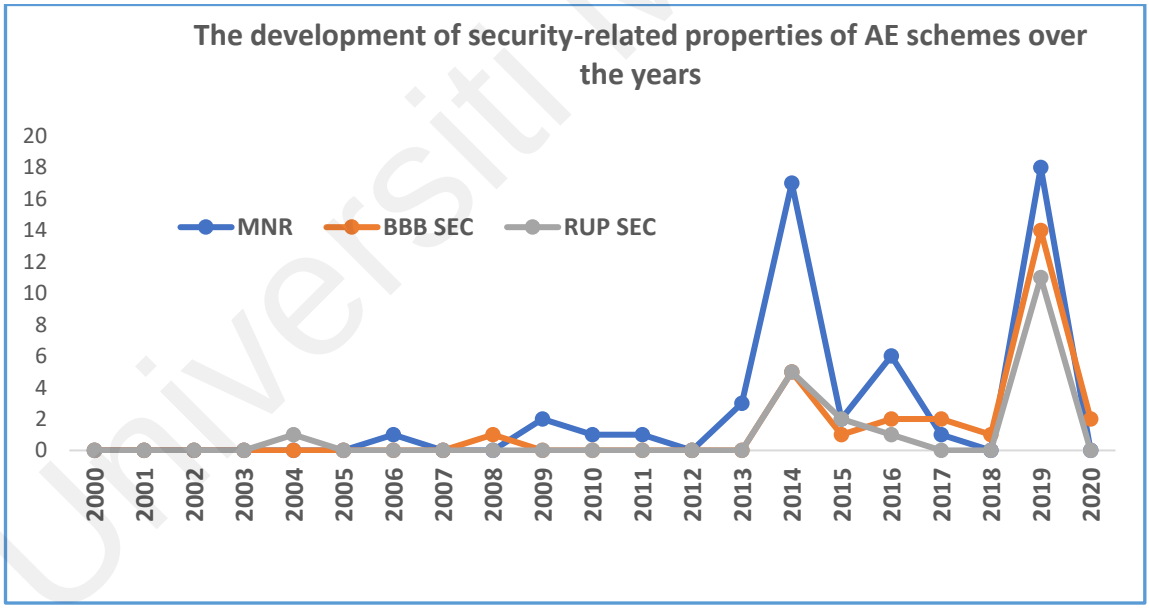

Figure 2. 7: The growth of security-related properties of AE schemes over the years (M. Jimale et al., 2022)

Most AE schemes provide security up to the birthday bound, which is $O(\frac{\sigma^2}{2^n})$, where σ is the number of the ciphertext blocks, and n is the block length. However, birthday-bound security is not always adequate in cases where protection beyond the

birthday bound is required, primarily for schemes whose $n^{n/2}$ value is small, like those with 64-bit security (Iwata, 2008; Mennink, 2020). Several AE schemes in the literature offer BBB security (Cassiers et al., 2019; Iwata, 2008; Naito & Sugawara, 2019), as stated in our SLR (Jimale et al.,2022).

2.3.5.5 AE Cryptographic Primitives

The workhorses of AE schemes and their encryption modes are the encryption algorithms that ensure the confidentiality of messages. They are units that scramble the plaintext so that no one can easily decipher it without knowing the key. In addition to security, encryption primitives contribute to other desirable features that enhance efficiency and performance and make for compact and elegant design.

Cryptographic primitives are algorithms used to build cryptographic protocols for secure communications. They include encryption and MAC algorithms that are used to protect confidentiality and authenticity, respectively. Our SLR, Jimale et al. (2022) listed 112 cryptographic primitives used to build AE schemes, as shown in Table 2.1. The most frequently used cryptographic primitives were AES, SPN, Sponge, TBC, Keccak, GIFT, SKINNY, and Grain. (Abed et al., 2016; Agrawal, Zhou, & Chang, 2019; F. Zhang et al., 2018).

Table 2.2: The 15 most used cryptographic primitives in AE schemes(M. Jimale et al., 2022)

SN	Primitive	Occurrences	Description
1.	AES	64	Advanced Encryption Algorithm
2.	SPN	12	Substitution Permutation network
3.	Sponge	9	Sponge Permutation
4.	TBC	5	Tweakable Block Cipher
5.	Keccak	4	Keccak Function
6.	Permutation	3	Keyed Permutation
7.	PRIMATE-s	3	PRIMATE-s Permutation
8.	GIFT	3	GIFT Block Cipher
9.	Prøst	3	Prost Permutation
10.	SPRING	2	SPRING Tweakable Block Cipher
11.	SKINNY	2	SKINNY Block Cipher
12.	Grain	2	Grain
13.	Deoxys-BC	2	Deoxys Block Cipher
14.	sLiSCP-light	2	sLiSCP-Light Permutation
15.	ICE	2	ICE Cipher
16.	Others	97	96 schemes used a unique primitive once, while two schemes used the same primitives

2.3.5.6 AE Design/Modes

Encryption algorithms are primarily designed to encrypt single data blocks; however, we process large amounts of data in real life. Modes of operation are meant to give the encryption primitives more capabilities and valuable functional features. Thus,

encryption algorithms work with different modes of operation and design philosophies that make them behave differently in terms of security or performance. For instance, AES behaves differently with modified capabilities when used with Counter Mode (CTR) and Cipher Block Chaining (CBC) mode. Several modes and design-specific constructions for symmetric key-authenticated AE schemes use different underlying cryptographic primitives. Examples of designs/modes for AE schemes include CTR, Duplex, EME, ECB, TAE, OCB, SIV, and ARX (Abed et al., 2016; Agrawal et al., 2019). Our SLR (Jimale et al.,2022) listed 128 modes of operations used in 217 AE schemes in their Systematic Literature review, shown in Table 2.2.

Table 2. 3: The 15 most used modes/designs in AE schemes(M. Jimale et al., 2022)

Mode/Design	**No. Schemes**	**Description**
Duplex	14	Duplex design
CTR	10	Counter Mode
LFSR	10	Linear Feedback Shift Register
EME	9	Encrypt Mix Encrypt
OCB	7	Off-set Codebook Mode
TAE	6	Tweakable AE
SIV	6	Synthetic Initialization Vector
ECB	5	Electronic Code Book Mode
ETM	5	Encrypt Then MAC
Sponge	4	Sponge Construction
ARX	4	Add Rotate XOR

Mode/Design	No. Schemes	Description
OTR	4	Off-set Two Round
MonkeyDuplex	4	Monkey Duplex
SPN	3	Substitution Permutation Network
XEX	3	XOR Encrypt XOR
Others	123	113 modes, Most of Them Used Only Once

2.3.5.7 Protection Against Side Channel Attacks (SCAs)

Hackers employ different techniques to attack cryptographic algorithms. These attacks are generally classified as cryptanalytic and implementation attacks (IAs). Classical cryptanalytic attacks use linear/differential cryptanalysis or brute force attacks. In classical cryptanalysis, the cryptographic algorithms are mathematically analyzed to find flaws and recover some parts of plaintext out of the number of ciphertexts available. In a brute force attack, the hacker tries out all possible secret key values. This attack has a lower possibility of success with most modern crypto algorithms with the standard key sizes increasing. For instance, breaking a key length of 256 or 128-bit key is computationally infeasible (IRTF, 2019; Kalai & Reyzin, 2019; Mangard et al., 2007).

IAs do not need to rely on the mathematical weakness in cryptographic algorithms to gain helpful information. Instead, they use the side-line information emitted from devices that run cryptographic protocols while they process the data, which exposes patterns from which secret keys could be deducted (Mangard et al., 2007; Popp, 2009).

IAs take advantage of the implementation environments of cryptographic devices and gained popularity with the rise of electronic integrated circuits, like encryption processors, smart cards, USB tokens, and other chips that perform operations to protect secret information using cryptographic protocols. Therefore, a mathematically robust crypto algorithm could be vulnerable to attacks in the side channel if proper countermeasures are not in place(Kim & Shin, 2022; Kwon, Kim, & Hong, 2021; Mangard et al., 2007).

IAs can be categorized as either passive or active attacks. Passive attacks do not involve any actions that engage the target device. Instead, it observes the behavior of the working machine to extract helpful information generally emitted as a result of its operations, like the power consumption or the time it takes to perform certain operations. On the other hand, active attacks usually involve physically manipulating the target device to force it to function abnormally and exploit it to wage attacks or gain useful information (IRTF, 2019; Kalai & Reyzin, 2019; Mangard et al., 2007; Popp, 2009).

IAs could also be classified as invasive, semi-invasive, and non-invasive attacks. Invasive attacks include all possible accesses and manipulations, including probing, cutting wires, or rerouting circuitry channels and modifying logical states or signals in the target device. In semi-invasive attacks, target devices are physically manipulated by removing the non-functional parts like covers, for example. Active circuits are not physically contacted or altered. In non-evasive attacks, only information directly available from accessible interfaces is exploited; the target device is not physically engaged (Popp, 2009).

The main types of IAs include Side-Channel analysis, Fault Analysis, probing attacks and hybrid attacks that combine some of the above, and reverse engineering, in which cryptographic algorithms' hardware or software implementations are analyzed to know their inner workings and break them.

Side-channel attacks (SCAs) exploit leakages of signals emitted from implementation devices, such as execution time, power consumption patterns, and other electromagnetic emanations. The basic principle of these attacks is to determine the secret key information by influencing the signal patterns (Dobraunig, Eichlseder, et al., 2019b; Mennink, 2020).

Fault Attacks are active attacks that stress cryptographic electronic devices like USB tokens or smartcards by an external means (e.g., voltage, light) to force them to malfunction. The attacks benefit from that abnormal condition to wage an attack or gain helpful information that might reveal the secret key, accept false signatures or allow PIN recovery (Mangard et al., 2007; Popp, 2009). A successful fault attack involves two steps: Fault injection entails injecting a fault at the appropriate time during the process and depends on the target hardware. The second step, fault exploitation, involves exploiting the erroneous result or unexpected behavior and depends on the software design and implementation (Benot, 2011). Figure 2.9 depicts a general classification of cryptographic attacks.

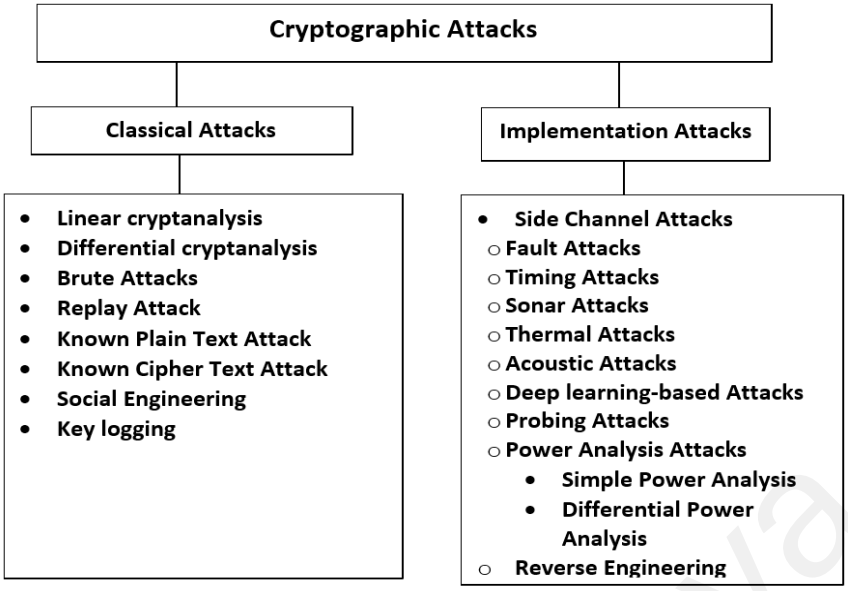

Figure 2. 8: Classification of cryptographic attacks

Another powerful implementation attack is the Probing attack, an invasive method for circumventing security measures by observing the physical silicon implementation of a chip in the regular operation of the cryptographic circuit. One needs expensive equipment like a chip probing station to initiate such an attack, after which the attacker may need to reverse-engineer the device. Moreover, probing attacks get more difficult these days because of the continuously shrinking sizes of semiconductor technologies and the increasing complexity of chip circuits. Nonetheless, the effort may still be worth it because successful probes allow, in the best case, to read out bits of secret keys directly. A designer must identify possible targets and take appropriate measures to protect against such attacks (Benot, 2011; Mangard et al., 2007; Popp, 2009).

An acoustic SCA is a category of SCAs that exploit sounds generated by computers or other gadgets like smartphones. For example, sonar systems use radio and sound waves to track objects, including humans and have been employed in human-computer interaction in recent years. Most acoustic side-channel attacks are passive, and

the attacker snoops singles generated by the victim. However, Cheng et al. (Cheng, Bagci, Roedig, & Yan, 2020) presented an active acoustic attack signal generated by the attacker.

Thermal (temperature-based attacks) have been known for years, but the degree of their significance has not been exposed until recently. Most publications, for instance, mentioned its existence or provided some basic information about it without further details (Anagnostopoulos et al., 2021; Barenghi, Breveglieri, Koren, & Naccache, 2012; J. Brouchier, Kean, Marsh, & Naccache, 2009; Hutter & Schmidt, 2014; Karaklajić, Schmidt, & Verbauwhede, 2013). For example, Brouchier et al. (Julien Brouchier, Dabbous, Kean, Marsh, & Naccache, 2009; J. Brouchier et al., 2009) showed that a cooling fan could carry information about the handled data through the dispelled temperature of a CPU.

Furthermore, the authors indicated that information could be leaked between IP cores of FPGAs in the same system through temperature-side channels. Other works (Anagnostopoulos et al., 2018; Samyde, Skorobogatov, Anderson, & Quisquater, 2002) discussed active thermal attacks that temper the environmental temperature of the target device either by heating or cooling. Most of those works demonstrate how efficient low-temperature attacks are; by cooling SDRAM devices by $-50℃$. They could freeze the data and recover the contents of the memory even after seconds of powering down.

On the other hand, Quisquater and Samyde (2002) investigated higher temperature attacks by observing memory error after hours of extensive heating. Finally, a recent study by Kwon et al. (2022) proposed enhancing DL-based non-profiled SCAs. The authors suggested three methods to solve problems that prevented observing the intermediate result. In addition, they offered several enhancements that allowed monitoring the metrics

in the intermediate process, training a single network with no need to re-train the same model repeatedly, solving memory problems in parallel architecture, and helping achieve better performance (Kwon, Hong & Kim, 2022).

SCAs take advantage of the relationship between the cryptographic algorithms and the patterns of radiations from the implemented environments. The main philosophy of these attacks is deducing the secret key from their relationships with the side channel signal behavior (Mangard et al., 2007; Popp, 2009). SCAs are especially dangerous when cryptographic devices are placed where they can be physically accessible by adversaries.

The security strength of cryptographic structures is typically measured based on the assumptions that adversaries behave according to security models defined with conditions and limitations specified by the protocol (Kwon et al., 2021; Mennink, 2020). Furthermore, adversaries traditionally took advantage of weaknesses in cryptographic algorithms to breach security. SCAs doubt this model's trustworthiness (Mennink, 2020). These attacks acquire side-line information about cryptographic functions through passive attacks such as SPA(Chari, Jutla, Rao, & Rohatgi, 1999; P. Kocher, Jaffe, & Jun, 1999; Mangard et al., 2007; Popp, 2009), DPA (P. C. Kocher, 1996; Mangard et al., 2007), timing patterns (P. C. Kocher, 1996), power consumption (Antognazza et al., 2021) or electromagnetic emissions (Kuhn & Anderson, 1998).

In addition to the traditional types mentioned above, SCAs have been evolving. For example, recent works show more recent varieties based on techniques like deep learning (Kwon et al., 2021), Artificial Neural networks (Mukhtar, Fournaris, Khan, Dimopoulos, & Kong, 2020), and thermal sensors (Kim & Shin, 2022).

(A) Power Analysis Attacks (PAAs).

Power analysis is a type of SCAs where data related to power consumption is utilized as the side channel to gain secret information about the target systems. First, power traces are recorded using an oscilloscope device during the operation of the cryptographic device. Then collected power traces are statistically analyzed using diverse techniques to extract the secret keys. The two main categories of power analysis attacks are SPA and DPA (IRTF, 2019; Mangard et al., 2007; Medwed et al., 2010).

(B) Simple Power Analysis

SPA aims to infer useful information about the used key by looking for patterns in the power traces related to the executed operations. SPA directly exploits the dependency between power consumption and secret bits of information; if the patterns in the power trace can be directly linked to some values or processes depending on secret data like key bits, then this attack is very effective. For instance, implementing RSA decryption and signature using modular exponentiation (that uses the square-and-multiply algorithm) is vulnerable to SPA if the two operations can be recognized in the power trace. On the other hand, the power consumption pattern of AES operations that are implemented using instructions of the microcontroller that consist of arithmetic operations can cause recognizable power consumption patterns if prevention measures are not in place (Dobraunig & Mennink, 2019; Mangard et al., 2007; Medwed et al., 2010; Mennink, 2020).

The general aim of SPA is to recover the key when a small number of traces are available for a few plaintexts. In single-shot SPA attacks, only one power trance is recorded. While in multiple-shot SPA attacks, we can measure the same plaintext numerous times or different plaintexts to reduce the signal noise and calculate the mean

of the recorded power traces. In either case, the principle of SPA remains the same; the attacker needs to view and monitor the power traces of the target device and infer helpful information from them. The more closely the power consumption pattern of the device under attack is related to the key, the more the success probability of the SPA attack.

(C) Differential Power Analysis

Kocher et al. (1999) stated that DPA uses a more advanced statistical technique than SPA by modeling the theoretical power consumption for each key. It is the most popular type of power analysis attack because DPA does not require detailed knowledge about the target device. Knowledge about the target device's cryptographic algorithm is usually enough to launch DPA. Furthermore, DPAs are highly successful even if the available power traces are noisy. DPA requires a large number of power traces in contrast to SPA; for that reason, the attacker must be in control of the targeted attack for some time for a DPA attack to succeed. The attacker could perform more operations (transactions on smart cards, for example) to generate more power traces, if necessary, which the attacker could then employ to recover the key (P. Kocher et al., 1999; Mangard et al., 2007).

One significant difference between SPA and DPA is how the recorded power traces are analyzed. According to (Mangard et al., 2007); Popp (2009), in SPA, the device's power usage is analyzed along the time access, and the attacker seeks to find related patterns or match templates in a single trace. In contrast, a DPA attack examines how the power consumption at fixed moments depends on the data processing operation. In contrast to SPA, DPA has a five-step general attack strategy, summarized in the following:

The first step of the DPA attack process is to choose an intermediate result of the algorithm target executes. The intermediate result should be the function of the data values and the cryptographic key, where the data values could be either plaintext or ciphertext. The second step is to measure the power consumption of the target device during its operation while it encrypts or decrypts. Finally, the attackers try to know the corresponding data values used in calculating intermediate values. In every run, the attacker records a signal power trace for each data block, and a matrix of trails and data values is obtained (Mangard et al., 2007; Popp, 2009).

The alignment of the power traces is vital for the DPA attacker; for that reason, the trigger signal for the oscilloscope should be generated and recorded precisely in the same order. The third step of a DPA attack is calculating a hypothetical intermediate value for each key value by obtaining a vector of the total number of possible values for the key, called a key hypothesis. The attacker can then calculate the hypothetical intermediate values using the vectors of data values and hypothetical keys for all encryption/decryption operations. In the fourth step of the process, the attacker uses simulation techniques to map the obtained hypothetical intermediate values to a matrix of power consumption values (Mangard et al., 2007; Popp, 2009).

The attacker simulates the power consumption of the target device for each hypothetical intermediate value to obtain hypothetical power consumption values for use in the next step. The simulation quality is directly proportional to the attacker's knowledge of the target device. The better the simulation matches the real power consumption characteristics, the more effective the DPA attack is. In the last step, the attacker compares the hypothetical power consumption of each key hypothesis with

recorded power traces at each position using algorithms to discover the device's key (Abdalla, Belaïd, & Fouque, 2013; Mangard et al., 2007; Popp, 2009).

(D) Countermeasures Against Side-channel Attacks

Countermeasures against SCA aim to remove the relationship between leaked side-channel information and computed intermediate values. Several mechanisms have been proposed in the literature for protection against SCAs, including hiding (Mangard et al., 2007) and masking(Duc et al., 2015; Ishai et al., 2003; Mennink, 2020; Prouff & Rivain, 2013). However, these countermeasures imply severe heavy performance penalties in resource-constrained environments such as IoT devices and smart cards. Fresh rekeying (Abdalla & Bellare, 2000; Medwed et al., 2010) is a less resource-intensive way to obtain SCA protection than other ways. Furthermore, fresh rekeying protects SCA by preventing the attackers from getting the intermediate key materials by confining the use of every session key to once or a few times (Abdalla & Bellare, 2000; IRTF, 2019; Krämer & Struck, 2020; Medwed et al., 2010).

i- Masking

Masking adds random items as masks to modify the intermediate values so that the attacker cannot guess them. For example, a masked intermediate value *(v_m)* is produced by an operation combining data (m) and a secret (random) value (v) that the attacker does not know in the following manner: $m_v = v \oplus m$. The operation could be either modular addition (+) or multiplication (×). One advantage of masking over hiding is that one does not need to change the power consumption properties of the device. Still, it can be implemented at the algorithmic level instead. Independence of intermediate values and power consumption patterns can be achieved with masking even if the device's

power consumption pattern is data-dependent (Duc et al., 2015; Mangard et al., 2007; Medwed et al., 2010; Prouff & Rivain, 2013).

ii- Hiding

The motive of the hiding is to make the power consumption of the target device independent of the operations performed while processing the intermediate values. The device should consume either a random amount of power in each clock cycle or an equal amount of power in each cycle to achieve that goal (Dobraunig, Eichlseder et al., 2019b; IRTF, 2019; Mangard et al., 2007). One way to randomize power consumption is to perform operations of the algorithms at different moments. Or by directly changing the power consumption characteristics of the performed procedures. For example, hiding schemes randomly change the execution times of the operation to be attacked or the vertical height of the side-channel signals of the process to be attacked (Lee & Han, 2020).

iii- Fresh Re-keying

The key-life time, the maximum amount of data that can be encrypted under a single key, is limited as dictated by several cryptanalysis methods that can recover the keys when a certain amount of data is processed. The birthday attack that put the limitation to $2^{(n/2)}$ is an example of such a method. So, it is vital to change the key when approaching key-lifetime boundaries. In real life, however, the amount of data permitted by this limitation is not enough. One obvious solution to this barrier could be the regeneration of a new key after a threshold amount of data is processed; however, this would entail performance penalties since it would require additional services and steps, some of which are resource-intensive (Abdalla & Bellare, 2000; IRTF, 2019; Medwed et al., 2010).

Fresh rekeying is a method to protect a cryptographic scheme against SCAs. It is an easily protected but cryptographically not heavy function that generates a subkey from the master key. The subkey is then used to encrypt or decrypt a single or few messages (Mennink, 2020). There are two types of rekeying functions: parallel rekeying, in which subkeys (session keys) are generated independently from the master key and serial generators. The generated subkeys depend on the previous state that continuously updates (Abdalla & Bellare, 2000). According to IRTF (2019), parallel rekeying is necessary when parallel access to data is used.

iv- Combined Countermeasures

Embedded devices such as Internet of Things (IoT) equipment and smart cards against SCA countermeasures are sometimes combined when only one is insufficient to protect cryptographic algorithms. For instance, Applying the first-order or second-order masking scheme and hiding schemes to AES make it more secure but slower (Lee & Han, 2020; Mangard et al., 2007). This work uses combined masking, hiding, and fresh rekeying to protect our sponge-based AE scheme against SCAs. See Chapter 4 for details.

2.3.6 AE Functional Features

In addition to security-related properties, other essential features according to which AE schemes can be classified and grouped include the following:

2.3.6.1 Parallelizability

An AE scheme's encryption is parallelizable if a block's encryption does not depend on the encrypted computation of any other block. The same definition can be provided for decryption. It reflects the ability of a scheme to process the i^{th} block independently of the j^{th} block (Iwata et al., 2017).

The continuous development of programs that require more computing power increased the demand and need to increase the speed of software applications. Furthermore, sensitive digital documents and commercial transactions have exposed the critical need for encryption algorithms. However, robust cryptographic algorithms are often slow and time-consuming; particularly efficient implementation is necessary for an online application; in this sense, a promising approach is the parallel implementation of algorithms. Parallel processing enhances the speed of algorithms and makes them more efficient, using multicore processors executing algorithms in multiple cores in parallel. A related but weaker type of this feature is pipelineable AE (Datta & Nandi, 2013). We call an operation pipelineable if the encryption and decryption operations can be split into multiple parts $g = g1 \circ g2$ such that the first *g1* can process $(x+1)^{-th}$ input block before *g2* has finished x^{th} block (Abed, Forler, & Lucks, 2014).

2.3.6.2 Online

An encryption scheme can be categorized as online or offline (Bellare et al., 2001). An online encryption scheme permits the computation of the i^{th} ciphertext block after seeing the first *i* plaintext blocks. In other words, to encrypt the i^{th} ciphertext block, we do not need to know any plaintext beyond this block. In the case of AE, if the message is viewed as the concatenation of several message blocks, it allows each block to be individually authenticated by producing a tag (intermediate tag) for each block. (Aumasson et al., 2016; Bertoni, Daemen, Peeters, Assche, & Keer, 2016; Alexandra Boldyreva & Taesombut, 2004; Fouque, Joux, Martinet, & Valette, 2004). On the contrary, an offline scheme outputs only the tag until all message blocks have been processed. An advantage of an online scheme is that the recipient can perform ciphertext block decryption and authentication on the fly at the receiving end.

Despite being a nice feature in AE schemes, the online feature could deprive the schemes of an important security feature. For instance, being online contradicts being NMR. For instance, Fleischmann et al. (2013) claim that their scheme (McOE) is online and NMR simultaneously and deals with nonce-respecting and general adversaries (Fleischmann, Forler, Lucks, & Wenzel, 2013). Later, Hoang et al. (2015) refuted this claim and emphasized that a scheme cannot be online and NMR simultaneously, redefining online AE and providing a different formulation called OAE2. Yet, they declared that nonce-reuse is devastating for their proposed notion, stating that no online AE can tolerate nonce-reuse (Hoang, Reyhanitabar, Rogaway, & Vizár, 2015).

2.3.6.3 Inverse free

An AE scheme is inverse-free if the underlying primitives do not require their inverses to perform encryption or decryption. This feature makes the scheme economical for implementation, as the same code and circuit can be used for different purposes. An AE scheme incurs additional implementation costs if inverses are needed. (Dobraunig, Eichlseder, et al., 2016b; Jean et al., 2016) .

For the reasons stated, inverse-free encryption modes are important designs for designing AE schemes. For instance, a construction that needs only the encryption circuit in its encryption and decryption has numerous advantages over those that require two separate operations for the encryption and decryption in having a lower footprint in a combined implementation. Therefore, several inverse-free AE schemes have been proposed based on diverse cryptographic structures like block ciphers (Peyrin, 2019), stream ciphers (Hell et al., 2019a), and sponge construction (Assche, 2011). Figure 2.10 shows how Building blocks contributed to the richness of functional features of AE schemes.

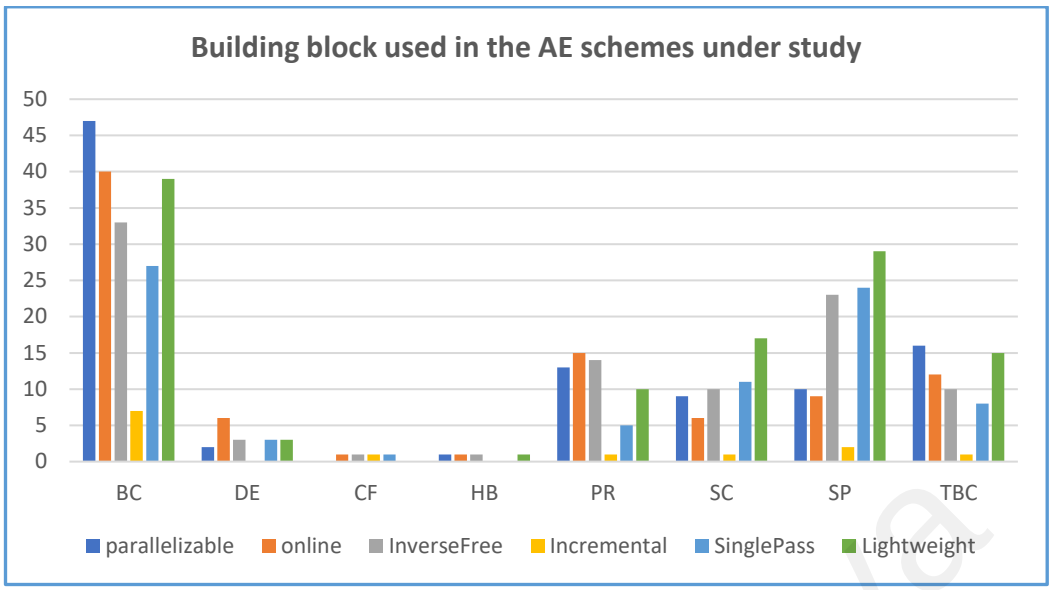

Figure 2. 9: Building blocks contributed to the richness of functional features of AE schemes (Jimale et al.(2022))

2.3.6.4 Incrementality

It is the ability to update parts affected only by the last action, given a previous ciphertext–tag pair *(C, T)* (Sasaki & Yasuda, 2016). An AE scheme provides incrementality if, given a previously computed ciphertext and a tag for a given plaintext M, encrypting another plaintext M' that differs only slightly from M is significantly faster than encrypting M' from scratch. Imagine a document that is frequently and continually updated, where the changes between edits may not be substantial, such as a set of appointment letters that are very similar in content but differ in the recipient's name.

Bellare et al.(1995) stated that incremental cryptography aims to develop cryptographic algorithms applied to a document; it is feasible to quickly update the result for the modified document without re-computing it from scratch. In cases where the frequent application of encryption and signatures is needed, incrementality can result in

dramatic efficiency improvements; authentication tags for virus protection are one example. (Khan, Kiah, Khan, Madani, & Khan, 2013).

Bellare, Goldreich, and Goldwasser (1995) investigated the concept of incremental cryptography applied to encryption. However, they first applied the idea of incremental cryptography to hash functions and digital signatures (Bellare, Goldreich, & Goldwasser, 1994).

2.3.6.5 Single-pass

One crucial indicator of the efficiency of AE schemes is the number of times the scheme processes the text for confidentiality and integrity. Two common ways are used: processing the plaintext once with one call to the underlying primitive to provide privacy and integrity and processing the data more than once to provide confidentiality and integrity with separate calls to the underlying primitive. Being single-pass renders a scheme more efficient (Boorghany et al., 2016; Reyhanitabar et al., 2016b).

This property makes the AE scheme more efficient, which means encryption and authentication involve only a single call to the underlying primitive per block. For instance, Bertoni et al. (2011) proposed a single-pass AE scheme based on the duplex mode of the sponge construction that requires only one call to the underlying permutation (Assche, 2011).

2.3.6.6 Lightweight

This feature determines whether the scheme suits resource-constrained devices (Agrawal et al., 2015; Chakraborti et al., 2015). Since the resource-limited devices cannot accommodate the standard cryptographic algorithms designed for powerful machines and

the increasing use of these devices to process or store sensitive data, developing lightweight versions of AE schemes that suit them was mandatory. So several lightweight versions of AE schemes were proposed in the literature based on stream ciphers in 2003 (Furuya & Sakurai, 2003), followed by other researchers (Lucks, 2005). Furthermore, as a sign of the growing importance of resource-constrained environment protection in 2013, NIST announced a separate competition for lightweight AE schemes, the NIST-LW competition (Andreeva et al., 2013; Engels et al., 2011).

2.3.7 Open Issues & Challenges

AE schemes have undergone tremendous improvements since their birth, simplifying many complexities, but gaps still need to be bridged.

The clear challenge in developing authenticated encryption schemes is striking a balance between properties with sometimes conflicting effects. From our findings, it is clear that researchers attempt to achieve efficient performance without compromising security. We found that the only parallelizable scheme, online, single pass, inverse free, and incremental, proposed in (L. Zhang, Wu, Sui, & Wang, 2014), fails to satisfy all three properties of NMR, RUP security, and BBB security. Similarly, the only scheme that was NMR secure, RUP secure, and provided BBB security, the one (Chakraborti & Nandi, 2014), was not online and did not provide incrementality, two essential features influencing the performance of AE schemes.

One future research direction should be to develop schemes that provide the maximum possible security with some performance gains by considering the prevalence of constrained devices in the future. With the rise of cloud and edge computing, another research direction is the application of homomorphic encryption and searchable encryption, which allow users to access data saved in the cloud without enabling the hosting service

provider to read or understand it. In our literature review, the authors found only one study related to homomorphic encryption (Cheon et al., 2018).

With the potential threats exhibited by quantum computing, many researchers have claimed that it would render current cryptographic algorithms ineffective. Quantum AE is thus expected to become a popular subject of research in the near future. It is also widely believed that quantum attacks do not threaten symmetric cryptography, but recent work (Kaplan, Leurent, Leverrier, & Naya-Plasencia, 2016; Santoli & Schaffner, 2016) has shown that many AE modes can be compromised in the superposition model. The authors found only two sources [179, 286] related to quantum AE in this review.

From the extensive literature search in this section, the authors also found a pressing need for parallel, sponge-based AE schemes that protect against SPAs and DPAs that hide the nonces from adversaries and provide the best possible security for nonce reuse. This work aims to fill this gap.

As this study primarily focused on AE in the symmetric key setting, conducting a comprehensive systematic literature review of AE schemes in public key settings is also an open problem.

2.4 Chapter Summary

This chapter provided a review of the state-of-the-art relevant literature on AE schemes. First, the chapter highlighted the issues and challenges in AE schemes based on sponge construction. Next, the chapter classified existing works according to the lines of work they belong to, and the security and performance feature they provide. Finally, it discussed possible enhancements and identified outstanding issues for further study.

CHAPTER 3: PROBLEM ANALYSIS

3.1 Introduction

This chapter analyzed the research problem, indicating the gaps in sponge-based AE schemes. It demonstrates how the existing parallel schemes lack the necessary protection against SPA and DPA; on the other hand, it underlines that the sponge-based constructs that protect against such attacks cannot process data in parallel. It also shows the need for nonce-hiding in sponge-based AE schemes and the merits of nonce misuse resistance for more reliable AE schemes.

3.2 Sponge-based AE schemes

Researchers used different constructs to implement AE schemes. For instance, we identified eight building blocks in the literature: Block cipher (BC) and tweakable block cipher (TBC), stream ciphers (SC), sponge constructions (SP), hash functions (HA), hybrid structures (HB), keyless permutations (PR), and dedicated independent building block (DE). Then, we grouped the works according to categories based on their lines of work as separate schemes, schemes proposed in the CAESAR competition, and those that were part of the NIST lightweight AE competition, as shown in Table 3.1.

Table 3. 1: AE schemes categorized according to building block.

Building blocks	Independent (Category A)	Caesar Competition (Category B)	NIST-LW Competition (Category C)	Total
BC	50	24	18	92
SP	14	6	19	39
SC	15	6	6	27
TBC	14	5	7	26
PR	4	9	6	19
DE	6	4	1	11

Building blocks	Independent (Category A)	Caesar Competition (Category B)	NIST-LW Competition (Category C)	Total
HB	1	1	0	2
CF	0	1	0	1
Total	104	56	57	217

Bertoni et al. (Bertoni, Daemen, Peeters, & Van Assche, 2008) first proposed Sponge construction in 2007, but using it in AE schemes started in 2011(Assche, 2011). As a result, many AE schemes in the CAESAR and NIST competitions used sponge-based designs. Furthermore, the number of AE works, and the variety of building blocks increased from 2014 onwards, indicating the impact of the CEASAR and NIST competitions in soliciting contributions from the cryptography community. The upsurge of publications in Figure 3.1 depicts the increase from 2014 to 2020. The figure also shows that the block ciphers were the only constructions used in AE, followed by stream ciphers and some dedicated structures, until sponges and permutations appeared in 2011.

The used building blocks significantly impact the features that AE schemes provide. For instance, while the block ciphers and tweakable block ciphers are favorable for their strong security, permutations seem to have the upper hand when lightweight features are the motive. For instance, 60% of the winners of the NIST-LW competition used permutations as underlying primitives. As shown in Table 3.1, most independent schemes used block ciphers, followed by sponges and stream ciphers.

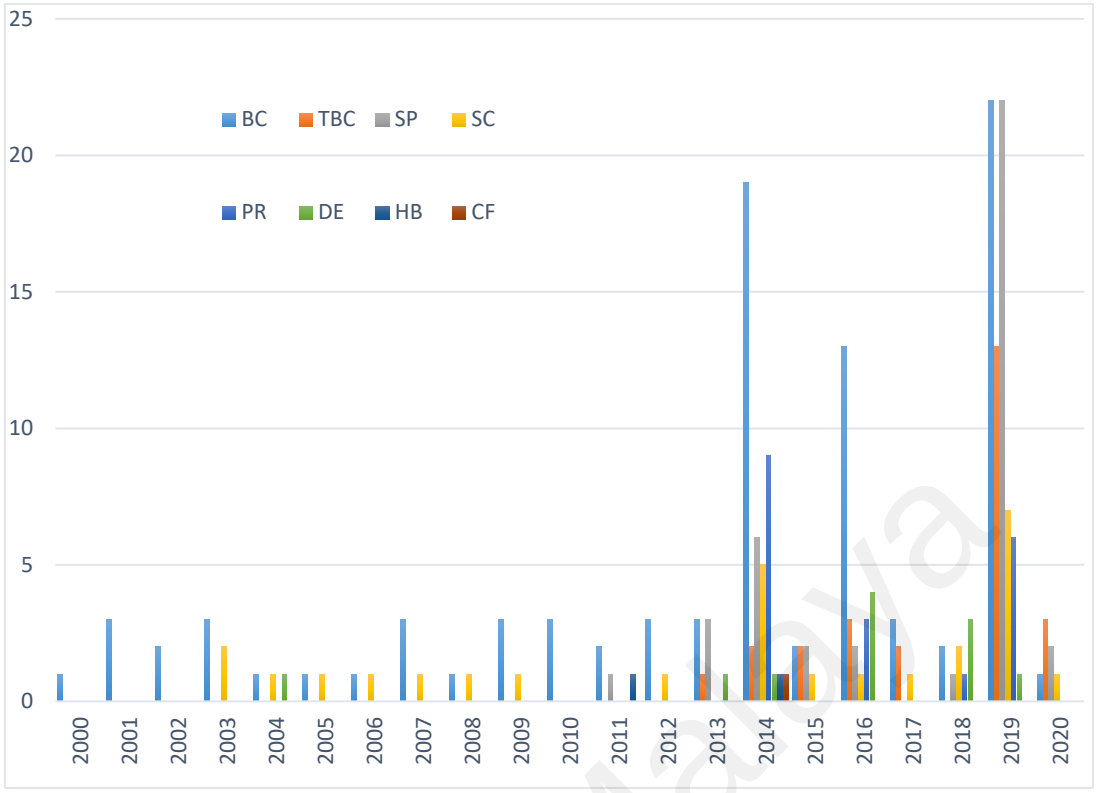

Figure 3. 1: Development of building blocks of AE schemes from 2000 to 2020.

Although it appeared years earlier, the sponge construction gained more popularity from 2013 onwards when NIST announced Keccak as the winner of its new cryptographic hash algorithm competition, a fundamental tool for cryptographic applications that ensure the authenticity of digital documents. Keccak was created by Bertoni et al. and became NIST's SHA-3 hash algorithm after speculations that SHA-2 could be under threat (FIPS, 2015).

Together with its mode of operation, the duplex construction, the sponge construction is employed to construct many symmetric cryptography functionalities, providing security and other desirable features. Furthermore, cryptographic sponges enable the designers to develop strong permutations without the concern of key scheduling hassles associated with block ciphers (Assche, 2011; Bertoni et al., 2011; Bogdanov et al., 2011).

3.2.1 Parallelism in Sponge-based AE

Despite their flexible design features, most sponge-based AE schemes lack some crucial characteristics indispensable for security and performance. For instance, parallel processing boosts the speed of algorithms and makes them more efficient by simultaneously processing more than one data item. This feature enables the AE scheme to perform operations independently of one another so that it can process the encryption or decryption of the i^{th} block separately from the j^{th} *block*; in that way, the operations can be performed concurrently by different processors. Figure 3.2 shows that Sponge-based AE schemes (SP) parallelizability was low compared to our study's block cipher-based schemes (BC).

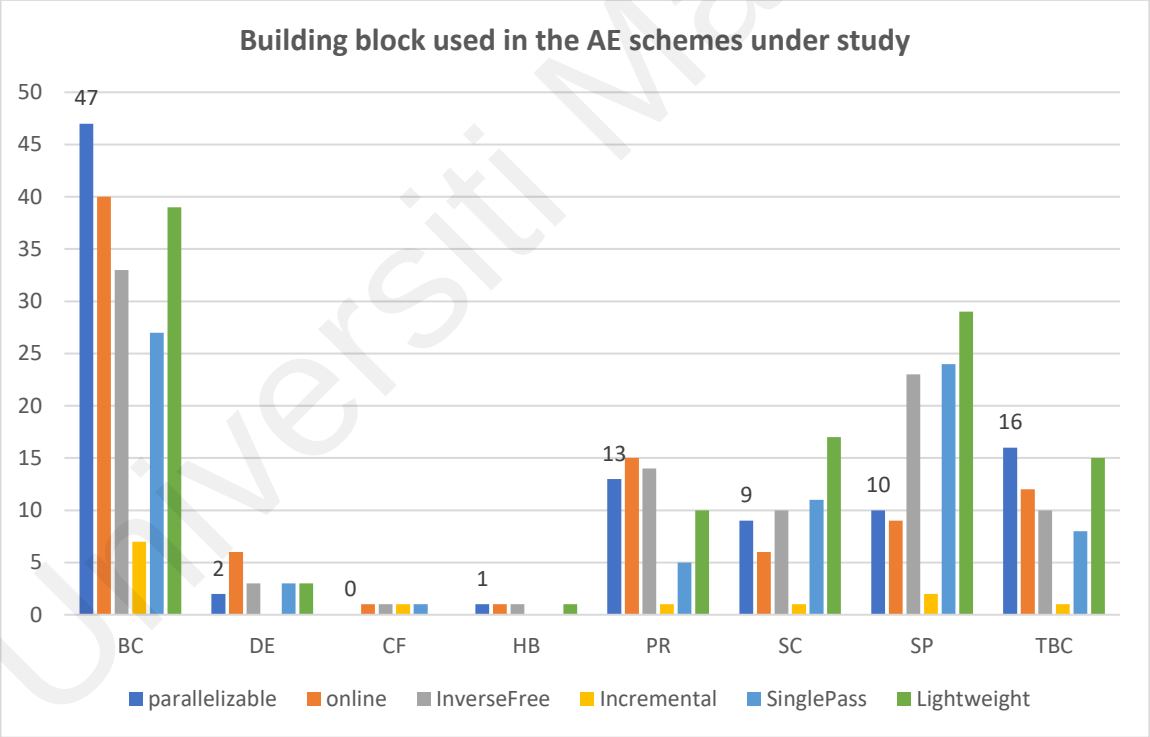

Figure 3. 2: Comparing the parallelizability of sponge-based and block cipher-based AE schemes (Jimale et al., 2022)

Most sponge-based AE constructions were serial before (Morawiecki & Pieprzyk, 2013) proposed the first parallel schemes based on the duplex mode of the sponge, followed by other schemes offered for the CAESAR competition. For instance,

Gligoroski et al. proposed a round-2 AE scheme based on the sponge construction with intermediate tags, based on the ARX design, Called the $\pi - cipher$, with a 'Triplex component' that performs the operations parallelly (Gligoroski, Mihajloska, Samardjiska, Jacobsen, El-Hadedy, et al., 2014). On the other hand, Aumasson et al. proposed a finalist sponge-based AE based on the duplex construction and the LRX design, called NORX (Aumasson et al., 2016). Figure 3. 3 depicts the $\pi - cipher$

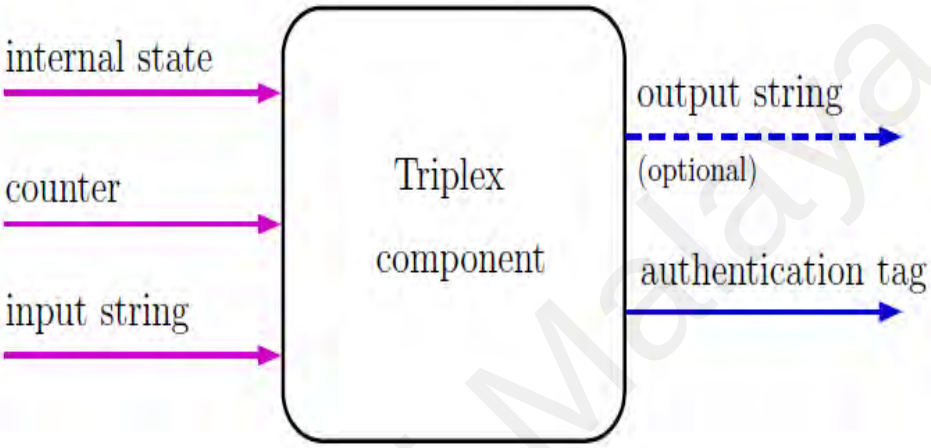

Figure 3. 3: The triplex component of the $\pi - cipher$ (Gligoroski et al.,2014)

However, the shortcomings relating to the AE schemes based on the sponge construction are not only the parallelizability, as with their counterparts based on the other underlying structures; there is a pressing need for protection against SCAs, discussed in the following subsection.

3.2.2 Sponge-based AE and Protection Against SCAs

The security of cryptographic protocols determines to what extent they withstand attacks with specific assumed resources. But some adversaries do not take advantage of weaknesses in cryptographic algorithms but take advantage of sideline information from the implementation environments. For instance, while a scheme might be secure in the

provable security paradigm, it could be vulnerable to such implementation environment leakages as power consumption patterns leading to SCAs (Bellare, 1998).

For instance, a power analysis attack is a type of SCAs where data related to power consumption is utilized as the side channel to gain secret information about the target systems. SPA directly exploits the relationship between power consumption patterns and encrypted data to infer useful information about the used key and needs detailed knowledge about the target device. DPA, the most popular type of power analysis attack, uses a more advanced statistical technique than SPA by modeling the theoretical power consumption for each key. Furthermore, it does not require detailed knowledge about the target device. An acoustic SCA is a category of SCAs that use sounds generated by computers or other gadgets like smartphones. Fault attacks are active attacks that stress cryptographic electronic devices like USB tokens or smartcards to force them to malfunction. Finally, thermal power analyses are SCAs that exploit the power consumption pattern of the device under attack.

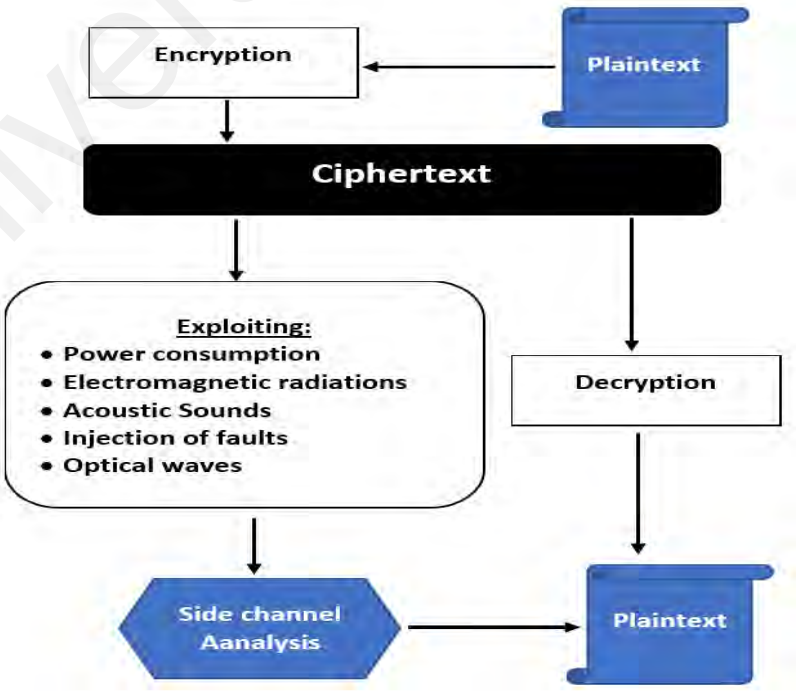

Figure 3. 4: A general scheme of SCAs

Those implementation-related attacks are particularly harmful when chips with sensitive information are in hackers' hands or are deployed where they are accessible to the general public, including many types of smartcards like IoT devices and sensor network nodes. Those cryptographic devices contain the elements that either implement cryptographic algorithms or work with data related to these operations. Other examples are chips like microcontrollers that run implementations of cryptographic processes and protocols, as well as encryption coprocessors and Read-only memories (ROMs) that store up cryptographic keys (Dobraunig, Eichlseder et al., 2019c), (Mennink, 2020).

Designers of sponge-based AE schemes did a lot of work to augment the exciting features of sponge construction to strengthen its protection against SCAs. For instance, Dobraunig et al.(Dobraunig, Eichlseder, et al., 2019c) proposed AEAD that focused on protection against SPAs and DPAs that combined variants of the sponge-based structures with lightweight permutations. That fortification is crucial whenever cryptographic devices are accessible to potential hackers, and secure software and firmware updates are essential and challenging. On the other hand (Bellizia et al., 2019b) proposed a sponge-based AEAD scheme called Spook designed to protect SCAs at a low energy cost by mixing a leakage-resistant mode of operation with bit-slice ciphers for low-latency applications. Likewise, Degabriele et al.(2019) proposed a leakage-resilient AEAD and laid the foundation for building such schemes from sponges. Furthermore, they developed the SALE construction, initiated it with T-sponge, and proved it secure in the non-adaptive leakage setting (Degabriele, Janson, & Struck, 2019).

The protection of sponge-based AE schemes against SCAs was a good step forward. Still, those schemes were serial and lacked the merit of parallelizability, which is vital for achieving good performance in architectures that can simultaneously process

many data items. Figure 3.5 shows a gap in the sponge-based AE schemes regarding parallelizability and protection against SCAs.

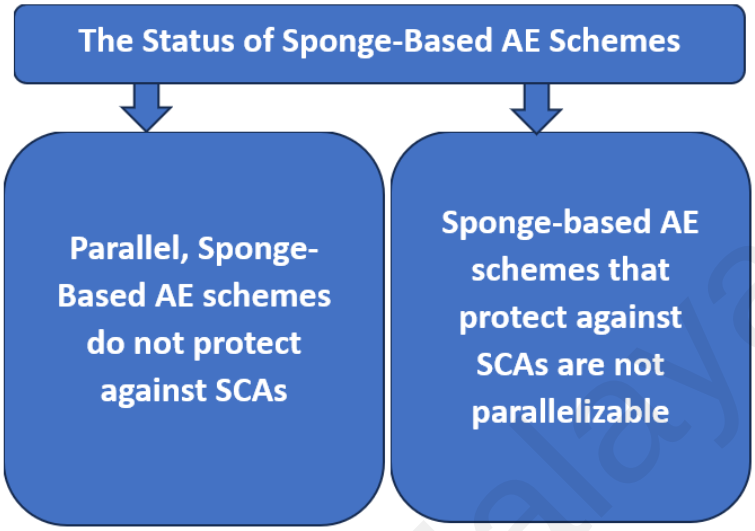

Figure 3. 5: A possible gap in sponge-based AE schemes

3.2.3 Countermeasures against SCAs

SCAs rely on the dependency between data emissions through a side channel and the secret key information. As a result, cryptographers proposed many techniques to protect against SCAs in the last decades to confuse attackers, render exploitation impossible, or require too much effort that attackers cannot afford. In technical terms, SCA countermeasures aim to break the dependency between the data values and operations occurring in a cryptographic algorithm and the side-channel signals of a cryptographic device(J. Brouchier et al., 2009; Dziembowski & Pietrzak, 2008).

Several techniques are in place to prevent SCAs. The first method is masking (also called blinding in the public key setting), which randomizes the intermediate results of algorithms before processing them in the devices (Duc et al., 2015; Ishai et al., 2003; Mennink, 2020; Prouff & Rivain, 2013). The second method is hiding (Mangard et al.,

2007), which obscures the relationship between the values processed and the side-channel signal patterns. Masking and hiding can be applied at the software or hardware level of cryptographic devices. However, it is worth noting that these two methods are primarily implementation-specific and resource-intensive (Medwed, Petit, Regazzoni, Renauld, & Standaert, 2011; Mennink, 2020; Mukhtar et al., 2020). Figure 3.6 depicts possible ways to protect against SCAs.

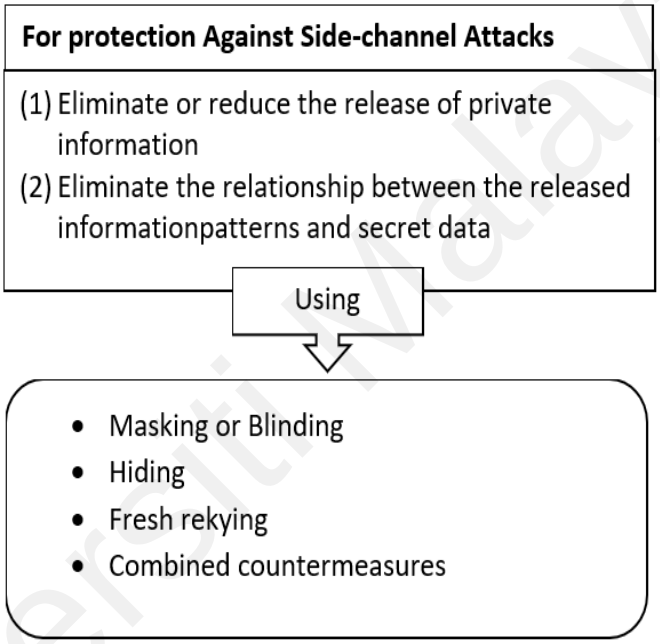

Figure 3. 6: Ways to protect against SCAs

One other cheaper and more practical way to protect against SCAs is fresh re-keying, in which we do not use the target cipher only but also a subkey generation function that uses the master key as input on top of it. The base cipher must be cryptographically strong but does not have to be heavily-side-channel protected since it uses the generated subkey only once or a few times. On the other hand, the key generating function does not need to be cryptographically strong but needs to be side-channel protected (Abdalla & Bellare, 2000; Dobraunig, Eichlseder, et al., 2019b; Medwed et al., 2011; Medwed et al., 2010).

Fresh rekeying is an efficient way to achieve side-channel protection since the generated subkeys are used to encrypt a single or few messages. Abdalla et al. (2000) first proposed it as a way to increase the lifetime of a secret key and investigated its security semantically. In fresh rekeying (also known as key derivation), the master private key K is not used directly, but subkeys $K_1, K_2, K_3, \cdots$ are derived from the master key and used in the encryption/decryption operations. There are three main types of fresh rekeying approaches: parallel rekeying, serial rekeying, and hybrid tree-based rekeying. Considering F as a function that maps a K-bit key and an additional input y to a K-bit output $F(K, y)$. The parallel rekeying produces many keys to be used by parallel algorithms such that $K_i = F(K, i) for\ i = 1,2,3, \ldots$. The serial key produces a single key at a time that it sets $K_0 = K$ and then sets $K_i = F(K, i) for\ i = 1,2,3, \ldots$. Hybrid rekeying is based on a balanced tree that constructs a key tree with a master key K at the root node (level 0) and frame keys $K^1, K^2, \ldots$at the last level. Assuming that the tree height is h and the number of keys is Lj, where j is $\{1,\ldots,h\}$, the subkeys are derived from parent keys at the higher levels according to a specific protocol and limited by the lifetime of the key and derivation functions (IRTF, 2019). Figure 3.7 compares the parallel and serial versions of fresh rekeying.

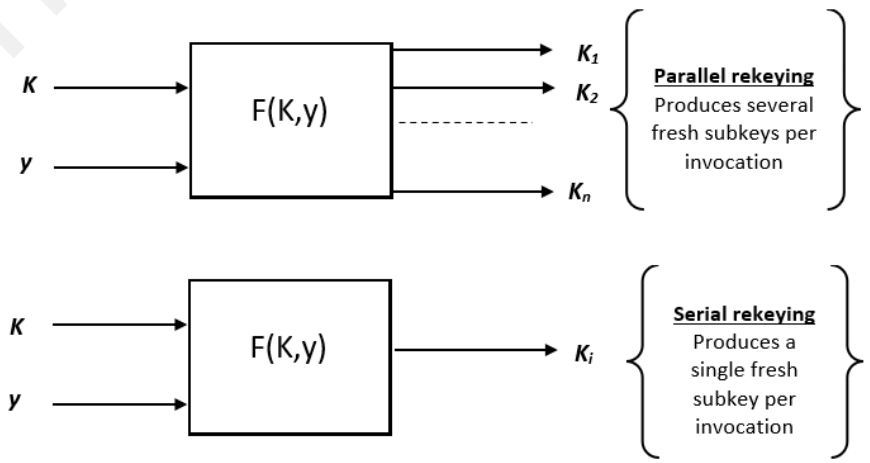

Figure 3. 7: Parallel and serial fresh rekeying methods

3.2.4 Nonce-obliviousness in sponge-based AE schemes

Nonce-based Authentication encryption schemes use non-repeating values called 'nonces' for deterministic encryption as an alternative to being probabilistic or a state-dependent function. In AE schemes, the idea first came from (Rogaway, 2002a, 2004b; Rogaway et al., 2003) but later gained widespread use in most authenticated encryption schemes published so far, paving the way for NBAE.

Despite the benefits that nonces brought to strengthen security, they have also become a tool to derail nonce-based encryption schemes' protection; their security claims hold as long as nonces are unique. Moreover, nonce repetition is not the only way to misuse them; other malicious methods exist. For instance, the hacker accessing the nonce sent in the clear with the ciphertext is potentially dangerous.

Bellare et al. (2019) analyzed the status of nonce-based encryption (NBE) and AE, which they called NBE1 and AE1, respectively. Rogaway et al.(2000) initially described NBAE as a deterministic algorithm that takes a key (K), a nonce (N), a message (M), and Associated Data (A) to return a ciphertext (C) and an authentication tag (T). The decryption algorithm takes as input a key (K), a ciphertext (C), a nonce (N), and an associated data (A) to return the plaintext if authentication succeeds or the failure symbol ($\perp$) if it detects forgery (Bellare et al., 2019a). On the other hand, the security of AE1 requires confidentiality of the message (M). Moreover, it requires integrity, and privacy for M and A, as long as the nonces are unique. In addition to nonce repetition misuse, its valid format and how to send it have also stimulated a hot debate among the research community.

Bellare et al.(2019) highlighted a gap between theory and practice in how AE schemes treat nonces. The prevalent view said nonces could be any values such as Device-ID and could be sent in the clear attached to the cyphertext. However, the authors stated that sending nonces in clear could compromise security. In addition, they noted that the claim that nonce could be anything like device-ID is not prudent and could devastate privacy since nonces are metadata and need careful generation, transmission, and handling. Finally, the authors criticized the standard that claimed 'full ciphertext,' consisting of core ciphertext and the nonce, and declared it wrong. They finally proposed an alternative way of handling the nonce in the AE schemes (Bellare et al., 2019a; Rogaway, 2002a, 2004b).

To bridge the gap between theory and practice, (Bellare et al., 2019b) proposed an approach to prevent the nonce burden and its potential mishandling by hiding it like the message. Nonce hiding eliminates the nonces from the decryption algorithm of the schemes so that the receiver does not have to worry about handling nonces anymore. Finally, the authors demonstrated simple ways to turn traditional nonce-based authenticated encryption schemes into nonce-oblivious ones and concretized them for the block cipher-based AE schemes. They also upgraded the security goals and formulated five ways to transform the existing AE schemes to conform to the new nonce-hiding syntax and called them Hide Nonce (HN) transforms (HN1 -HN5). The syntax covers basic security (unique nonces) and advanced security (NMR). Table 3.2 shows the HN transforms and their descriptions. It is an open problem to concretize the HN transforms for the sponge-based AE schemes. See Table 3.2 for the definition of HN transforms in the seminal article of Bellare et al. (2019).

Table 3. 2: The Hide Nonce (HN) transforms as Bellare et al. (2019a) proposed.

SN	Hide-Noce Transform	Descriptions
1	HN1	Requires the minimal core-ciphertext length to be non-trivial (at least 128), which is invalid for all schemes.
2	HN2	Requires ciphertext splitting (stealing); the L parameter of ciphertext stealing should be at least 128 in practice, which will call for a 224-bit block cipher.
3	NH3	Does not provide NMR
4	NH4	Provides NMR (Our choice for PSASPIN)
5	NH5	It depends on the Tweakable Block cipher, which is not conveniently compatible with the sponge-based case.

3.2.5 NMR sponge-based AE schemes

Encryption and AE schemes use nonce to boost securing by preventing predictability as long as nonces remain unique. However, nonce or IVs repetition may occur deliberately or for pathological reasons in the case of device malfunctioning or software bugs (Belenko, 2014; Gueron & Lindell, 2015b). Rogaway and Shrimpton (Rogaway & Shrimpton, 2006) proposed the SIV operation model, coming up with the idea of Nonce Misuse Resistance Authenticated Encryption (NMRAE) such that the authenticity remains protected. Privacy is protected to the extent of leaking some minimal information, revealing whether two plaintexts are equal and if the message (*M*), the header (*H*), and the particular Nonce (*IV*) are reused together, which can happen with negligible probability.

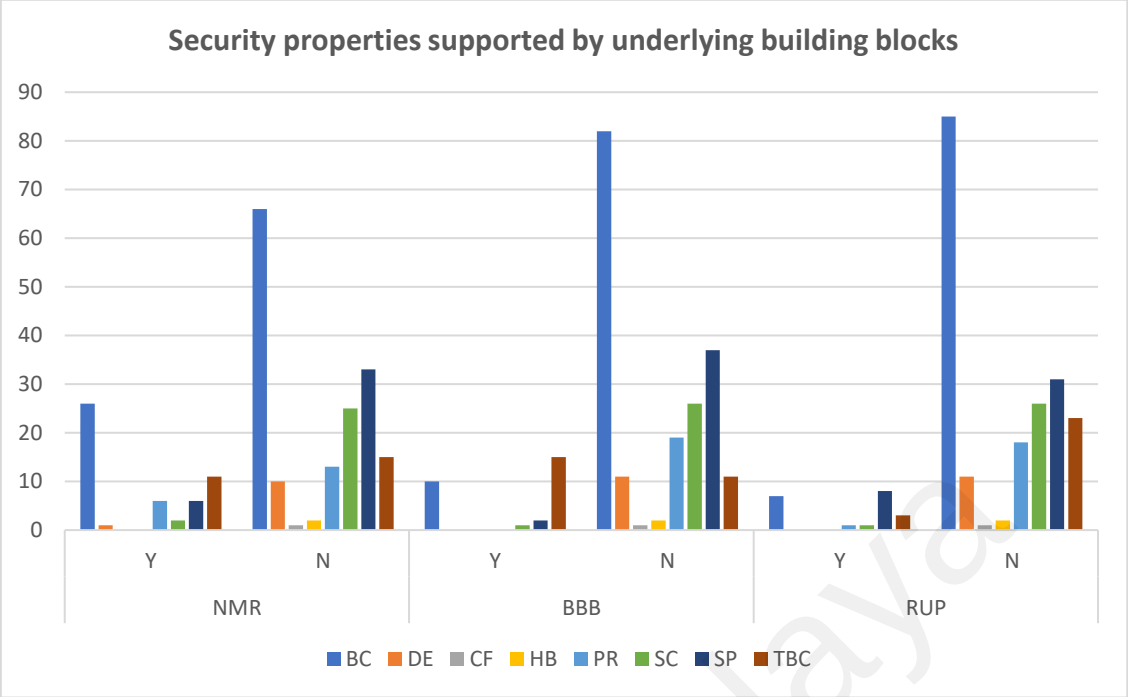

Figure 3. 8: How Underlying Building Blocks Support Security Properties

Sponge-based AE schemes have seldom provided NMR so far. For instance, only six of the 39 sponge-based schemes in our SLR (Jimale et al.,2022) claimed to offer some degree of misuse resistance in the case of nonce repetitions and were partially NMR. For that reason, boosting the security sponge-based scheme security in the case of nonce repetition is a possible future research direction. Figure 3.8 shows that the support of NMR in sponge-based AE schemes (SP) is meager compared to block ciphers (BC).

3.3 Possible Improvements

Despite the tremendous work done in sponge-based AE schemes, there are still spaces for improvement. For instance, besides the security features, functional characteristics, e.g., Parallelizability, incrementality, and single-pass, are indispensable for AE schemes because they contribute to the performance and efficiency of the schemes. Some sponge-based AE schemes are incremental, online, single pass, but they are not protected against side-channel attacks (Gligoroski, Mihajloska, Samardjiska, Jacobsen,

El-Hadedy, et al., 2014; Morawiecki & Pieprzyk, 2013). On the other side, sponge-based AE schemes that protect against side-channel attacks are not parallelizable, incremental, or single-pass (Bellizia et al., 2019a; Degabriele et al., 2019; Dobraunig, Eichlseder, et al., 2019a). Moreover, the parallel sponge-based schemes that protect against SPA and DPA can be improved further to accommodate additional features like nonce hiding.

This first possible enhancement to sponge-based AE schemes is to construct a parallel, incremental scheme shielded against side-channel attacks, specifically SPA, and DPA, using parallel fresh re-keying under the leveled implementation design.

Another possible improvement is to modify the dominant traditional syntax of sponge-based AE schemes similar to what Bellare et al. (2019) proposed for the block ciphers-based AE constructions so that the nonce is transmitted securely and invisible to the adversary. Offering a nonce-oblivious syntax will obviate the nonce burden from the receivers and prevent attackers from accessing and tampering with nonces. Adding the ability to keep a reasonable security level in the event of reuse (NMR) will also provide an additional security layer to sponge-based AE schemes.

This work is intended to fill the gaps in the sponge-based AE schemes by proposing and implementing a parallel sponge-based AE with Side-channel protection and adversary-invisible nonces abbreviated as PSASPIN. Table 3.3 compares the existing sponge-based schemes with the proposed solution.

Table 3. 3: Existing Sponge-based schemes compared with the proposed solution.

Schemes	Parallel	Incremental	Single-pass	SCA Protection	Nonce oblivious	NMR
Parallel AE with the duplex construction (Morawiecki & Pieprzyk, 2013)	Y	Y	Y	N	N	N
π–Cipher v11 (Gligoroski et al, 2014)	Y	Y	N	N	N	N
Spook (Bellizia et al., 2019)	N	N	N	Y	N	N
ISAP ((Dobraunig et al., 2019)	N	N	N	Y	N	N
SALE (Degabriele, Janson, Struck, 2019)	N	N	N	Y	N	N
PSASPIN (Proposed Scheme)	Y	Y	Y	Y	Y	Y

3.4 Chapter Summary

Although researchers have been doing tremendous work to enhance AE schemes based on sponge construction in terms of security, performance, and flexibility, there are always remaining gaps to bridge and spaces for improvement. This chapter analyzed the research problem, indicating shortcomings in sponge-based AE schemes. It demonstrated how the existing parallel schemes lack the necessary protection against SPA and DPA; on the other hand, it underlines that the sponge-based constructs that protect against such attacks cannot process data in parallel. In addition, the chapter showed the need for nonce-hiding in sponge-based AE schemes and the merits of nonce misuse resistance for more reliable AE schemes. The chapter finally highlighted the possible improvements in sponge-based AE schemes, laying the foundation for the proposed solution's features in this work.

CHAPTER 4: PARALLEL SPONGE-BASED AE WITH SIDE-CHANNEL PROTECTION AND ADVERSARY INVISIBLE NONCES (PSASPIN)

4.1 Introduction

This chapter presents the solution proposed to achieve the objective of the research. First, the chapter describes the Parallel Sponge-based Authenticated Encryption with Side-channel Protection and adversary-invisible Nonces (PSASPIN). Next, it outlines the parameters used by the scheme, introduces the security notations, and draws a schematic structure of the scheme. Finally, it explains the initialization, encryption, authentication, decryption, verification processes and the algorithms designed to implement the system.

PSASPIN is an Authenticated encryption with associated data based on the duplex mode of the sponge construction protected against SPA and DPA using Parallel fresh re-keying. In fresh re-keying, we do not use the target cipher alone but also a subkey generation function that uses the master secret key as input and generates subkeys to encrypt a single or few messages. That way, we ensure that hackers will have difficulty deducting the secret key since it changes frequently.

In addition to the protection against SCAs, the proposed scheme is parallelizable, enabling it to perform operations independently of one another. For example, a parallel algorithm can process the encryption or decryption of the i^{th} block separately from the j^{th} block; in that way, different processors can perform the operations concurrently. Parallel processing boosts the speed of algorithms and makes them more efficient by simultaneously processing more than one data item.

PSASPIN applies a modified syntax of Bellare et al. (2019) to sponge-based AE schemes such that it hides nonces from the adversary by taking a nonce as part of the encryption input but omitting it from the decryption. See Figure 4.1 for a schematic representation of the schemes. Nonces are encrypted with ciphertext, extracted at the destination, and used in the decryption to obtain plaintext. In that way, the nonce is invisible to the adversary and alleviates the burden of nonce management from the implementors and developers. In addition, the scheme is NMR keeping a reasonable level of security if nonces (which generally should not be reused) are repeated deliberately by attackers or in the case of software or hardware malfunctioning.

The new solution would combine the merit of being parallel and single-pass and the protection against certain types of SCAs using fresh re-keying realized as a leveled implementation and being nonce-oblivious (preventing the adversary from viewing/ accessing the nonces)

4.2 Parameters

The following are the main parameters used in the encryption and decryption algorithms of PSASPIN: key size, block size, nonce size, and the tag size are 128 bits, and the number of rounds is eight rounds.

4.3 Notations

Here we introduce the notations used in this work. We denote the key, the authentication tag, the nonce, and the initialization vector by *K, T, N*, and *SMN*, respectively. By *M, C,* and *A***,** we denote the plaintext, the ciphertext, and the associated data, respectively. By $\perp$ (bottom), we mean an Error or failure of verification. By **S,** we denote the state of the sponge construction, which is 320 bits. S_c stands for the internal

state (the capacity part), while S_r stands for the outer state (the rate part). By P, we denote the Sponge permutation. By 0^k, we mean an all 0-bit string of length k. By $|X|$, we indicate the length of string X. By $X||Y$, we denote the string *'X'* concatenated to the string *'Y.'* By $X \oplus Y$ we denote the XOR of *'X'* and *'Y'* strings. By $\lceil X \rceil^k$ We denote a bitstring X truncated to the most significant (last) k bits. By $\lfloor X \rfloor_k$, we denote a bitstring X truncated to the least significant bit (first) k bits.

4.4 PSASPIN Authenticated Encryption Scheme

PSASPIN AE takes four parameters: A 128-bit secret session K^* derived from the master key K, an arbitrary length plaintext message M, an arbitrary-length Associated data A, and a public message number (nonce) 128-bit N. The scheme also takes a 128-bit Secret Message Number nonce *(SMN)* in the encryption. The decryption takes A 128-bit secret session K^* derived from the master key K, cyphertext C, and an arbitrary-length Associated data A. The scheme uses a modified syntax of NAE so that the decryption does not take a nonce as an input parameter. The scheme is based on the duplex mode of sponge construction but uses fresh rekeying to get a fresh key for every invocation of encryption/decryption and authentication functions.

The scheme runs its processes in parallel, performing its operations independently of one another so that it can process the encryption or decryption of the i^{th} block separately from the j^{th} block so that the functions can be performed concurrently by different processors. First, the scheme divides the plaintext message M and associated data A into smaller blocks. At the initialization, a counter is initialized to keep track of the number of parallel threads. The counter value also controls when the Nonce is hidden in the first ciphertext (in the encryption) and when extracted from the ciphertext (in the decryption).

PSASPIN protects against certain types of SCAs, specifically SPAs, and DPAs. SCAs exploit the dependency emissions from side channels and the secret key information. Cryptographers use many techniques to protect against SCAs to render exploitation impossible. This work generates parallel fresh subkeys for every initialization and finalization. In addition, every encryption/decryption process uses a separate new key to prevent the adversaries from collecting enough key materials under SCAs. PSASPIN use this technique to break the dependency between the processed data values and operations occurring in a cryptographic algorithm and the patterns of signals of a cryptographic device that implement them.

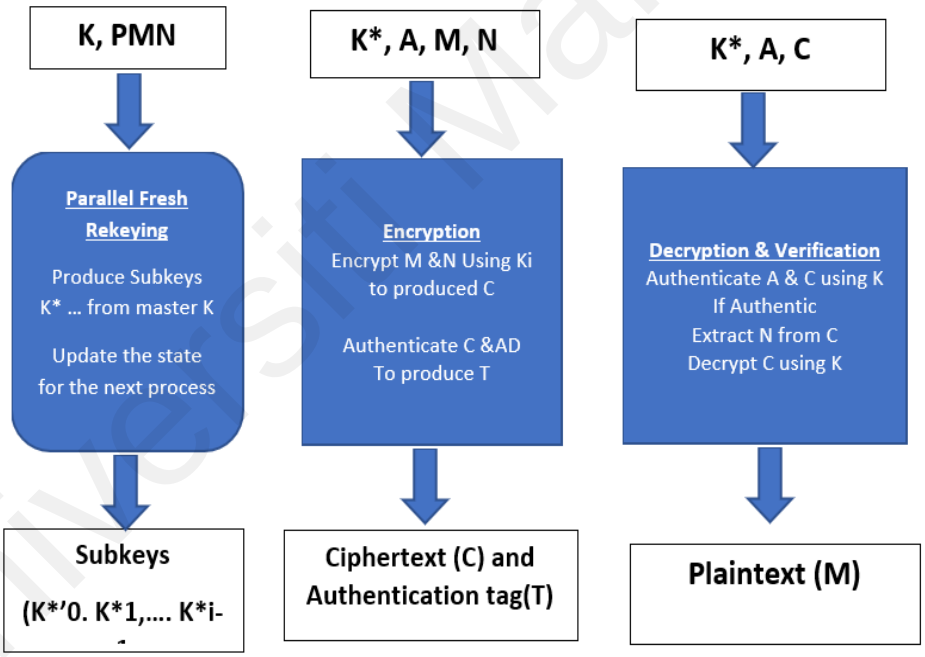

Figure 4. 1: A high-level view of PSASPIN structure.

The scheme conforms to the nonce-hiding syntax proposed and concretized for block cipher-based AE schemes by Bellare et al. (2019), specifically on their HN4 transform, which is NMR, in addition to its Nonce-hiding feature. For instance, our scheme, PSASPIN, uniquely processes nonces. First, it constructs a synthetic nonce made of the XOR of the first message block M_0, the first Associated Data block A_0, and the

original nonce N to produce the Nonce N_1, which is used in the encryption after the application of the sponge permutation function. In addition, the nonce integrates with the first ciphertext block in the encryption so that the first ciphertext C_0 contains the encrypted first plaintext block and the nonce. Unlike the traditional syntax of AE schemes, the nonce-hiding syntax does not take a nonce as an input in the decryption; the nonce is extracted from the first block of the cyphertext and used to produce the plaintext. Figure 4.1 depicts a general view of the scheme. The encryption and decryption processes of PSASPIN AE are shown in Figure 4.2 and Figure 4.3, respectively.

4.4.1 PSASPIN Processes

Several steps are necessary for PSASPIN to do its job, from initialization and encryption/decryption to finalization and tag generation. In the initialization and finalization, Parallel Fresh Rekeying (FRK) is called to feed the process with a new session key.

4.4.2 Initialization

In the initialization process, the FRK is called. It takes a master K and a Random *IV* and produces a fresh session key K_s to protect the scheme against SPA/DPA. See algorithm 1.4 for details of FRK. After generating the session key, the shared state S is updated. Next, the first Random *IV* is generated at the source and securely transmitted and will be incremented to keep the sides synchronized to the sponge permutation P. Finally, a counter (*Ctr)* is generated to keep track of the number of parallel threads incrementing by 1 with every lane.

4.4.3 Processing The Associated Data

PSASPIN first breaks the Associated data message into r bit block. Padding is done by appending '1' and a minimum number of '0's to A so that its length is a multiple

of the block size *r*. If the AD block is empty, no padding is necessary. The AD is processed one block of *r* bits at a time. $A_0||A_1||.....A_i||$, $|A|=r$. Every block of *A* is XORed with the outer part of the state (S_r), then it is concatenated with the inner part of the state (S_c). The state is updated by the permutation *P* in the following manner: $S \leftarrow P((S_r \oplus A_i) \parallel S_c)$. After processing the last A_i, a 1-bit domain separator is XORed with the state *S*: $S \leftarrow S \oplus (0^{319} \parallel 1)$ to indicate the end of Associated Data and plaintext parts and prevent attacks that change the roles of Associated Data and plaintext blocks as in Ascon (Dobraunig, Eichlseder, Mendel, et al., 2019).

4.4.4 Processing And Hiding the Nonce (SMN)

The concatenation of the first plaintext block (M_0), the first Associated Data Block (A_0), and the Public Message Number *(N)* is OXR-ed with the outer part of the state S_r, $S_r \leftarrow S_r \oplus (N \parallel A_0 \parallel M_0)$, then the new nonce *N1* is assigned to the outer state S_r: $N_1 \leftarrow S_r$, after that, the state is updated in the following manner: $Sr \leftarrow Sr \oplus (N_1)$; $S \leftarrow P((Sr \oplus N1)||Sc)$. The new nonce *N1* is a ciphertext of the original nonce and will be XORed with the first ciphertext block in the next stage.

4.4.5 Processing The Plaintext (Encryption)

After breaking the plaintext message into blocks of size *r*-bits, the first plaintext block (M_0) is XORed with S_r to produce an intermediate ciphertext block (C_{M0}): $S_r \leftarrow S_r \oplus M_0$; $C_{M0} \leftarrow S_r$. *The* C_{M0} is combined with the nonce ciphertext N_1 to produce the first ciphertext block (C_0), and the state is updated: $C_0 \leftarrow N_1 \parallel C_{M0}$; $S \leftarrow P(C_0||S_c)$. For the rest of the plaintext blocks, every block is XOR-ed with the outer part of the state S_r, the subsequent ciphertext blocks are produced, and the state is updated. $S_r \leftarrow S_r \oplus M_i$; $C_i \leftarrow S_r$; $S \leftarrow P(S_r||S_c)$. The last block is processed differently. The last plaintext block (M_z) is XOR-ed with the outer state S_r and is truncated to the length of the original unpadded plaintext

length so that the ciphertext and the original plaintext are of the same length: $S_r \leftarrow S_r \oplus M_z$; $C_z \leftarrow \lfloor S_r \rfloor_{|M| mod\ r}$

4.4.6 Decryption And Extraction of Nonce

The encryption and decryption processes of PSASPIN are identical except for the absence of the *SMN* nonce processing in the decryption. In the decryption, the *SNM* is extracted from the ciphertext. First, the ciphertext is split into blocks. The first ciphertext block (C_0) is split into the core ciphertext part (C_{m0}) and the nonce part (N_1) part, then the outer state is XOR-ed with N_1 and transformed with the permutation P. The first core ciphertext is retrieved by XORing S_r with (C_{M0}); $N_1 \parallel C_{M0} \leftarrow C_0$; $S \leftarrow P((S_r \oplus N_1)||S_c)$; $M_0 \leftarrow S_r \oplus C_{M0}$. For the rest of the ciphertext block except the last one, the ciphertext block (C_i) is XORed with the inner state S_r to produce the corresponding plaintext (M_n), and the state is updated; $M_i \leftarrow S_r \oplus C_i$; $S \leftarrow C_{Mi}||Sc$; $S \leftarrow P(S)$. The last ciphertext is produced by XORing C_n with S_r truncated to the original length of the ciphertext; $M_n \leftarrow \lfloor S_r \rfloor_{|C_n|} \oplus C_n$ The padded inner state is then updated to proceed to the finalization state.

4.4.7 Finalization

The additional key material is used to defend against side-channel and forgery attacks in the finalization phase. The state is updated by the concatenation of the secret session K_s key, the initialization vectors *(IV)*, and the string of 0s so that the length of the state S is 320-bit, the state width of PSASPIN, the final tag T calculated from intermediate tags t_j, is obtained by the truncation of the XOR of the full state and secret session key K_s to 128 bit, $T \leftarrow \lceil S \oplus K \rceil_{128}$, then the concatenation of the ciphertext blocks and the tag is returned: $C_1||...||C_i||C_z||T$. See Figures 4.3 and 4.4 for the schematic view of PSASPIN and the algorithm later in this section for its processes.

4.4.8 The Rekeying Function.

There are several options for implementing the rekeying scheme, as shown in Figure 3. This study follows the leveled implementation approach proposed in (Abdalla & Bellare, 2000) and adopted by (Medwed et al., 2011; Medwed et al., 2010), where the overall scheme is divided into the rekeying and the rekeyed data processing parts. There are several implementation options for the data processing part: block ciphers like AES, permutations like the duplex mode of sponge construction, or tweakable block ciphers (Degabriele et al., 2019). For the rekeying function, a PRF used as a pseudorandom generator (***G***) is used, which in turn offers different options: (1) leakage-resilient constructions like duplex sponges (Degabriele et al., 2019; Dobraunig, Eichlseder, et al., 2019b), (2) protected block ciphers like AEAS and SERPENT (Pereira, Standaert, & Vivek, 2015), (3) Tweakable block ciphers (4) Algebraic construction-based Galois field (GF) multiplication, or (5) Traditional block ciphers with countermeasures like masking and hiding (Barwell, Martin, Oswald, & Stam, 2017; Bellizia et al., 2019a; Krämer & Struck, 2020).

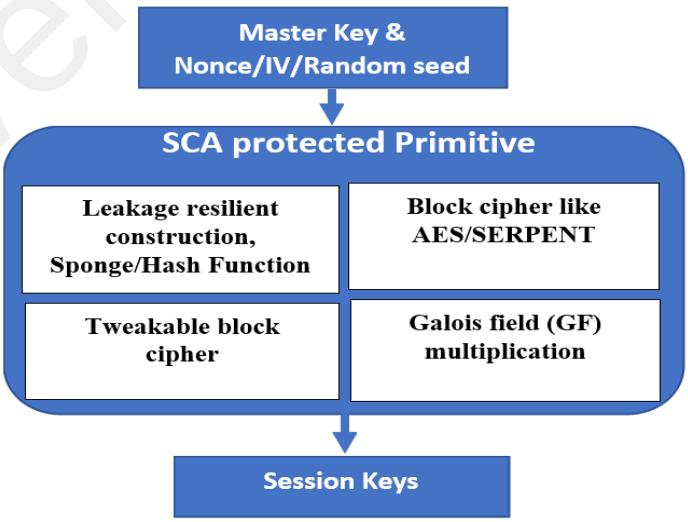

Figure 4. 2: Options for Implementation of Rekeying Function

This work uses the leveled implementation approach, with two possibilities for implementing the rekeying function (***G***). The first option is to use a function based on a

GF multiplication field as in (Abdalla et al., 2013; Medwed et al., 2011), but with slight modifications in the present case because their implementations protect only against lower-order differential power attacks (DPAs). This work combines countermeasures for protection against higher-order attacks. For instance, in Algorithm 4.1, a combination of masking and shuffling is used to protect against SPA and DPA, whereas the sponge-based core primitive uses the session key only once to protect it against SPA. The other option is using a leakage-resilient block cipher, which is a heavier construction than the first option but is preferred in hardware implementations. See Algorithm 4.1 for details of the *G* function based on the GF multiplication field using masking and shuffling combined to protect against higher-order DPA attacks.

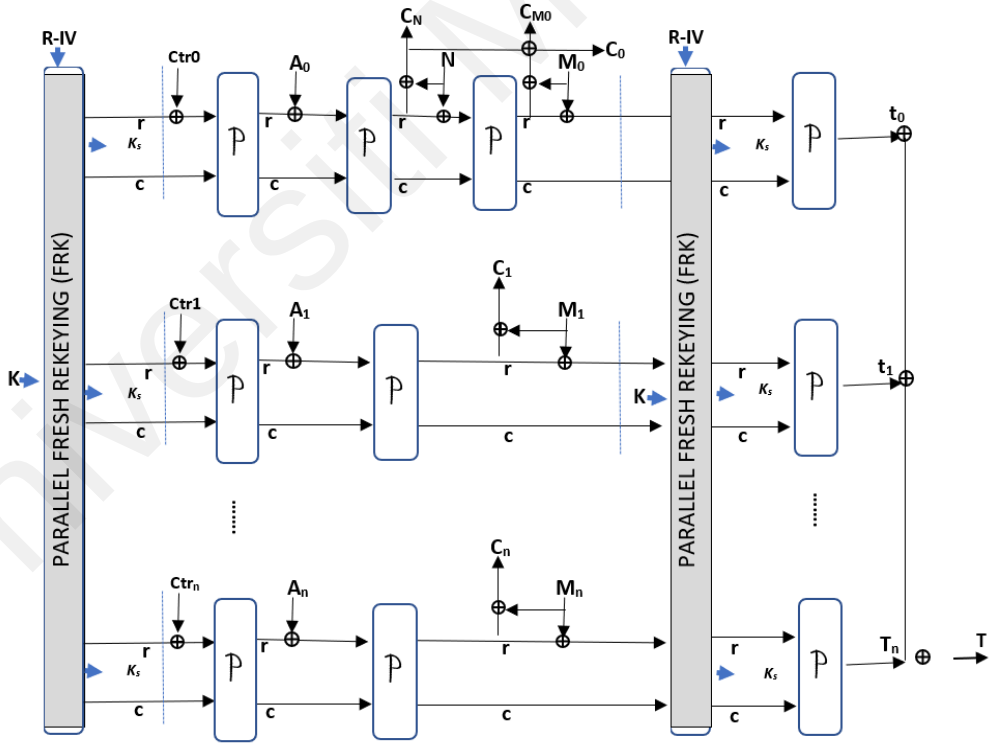

Figure 4. 3: PSASPIN encryption

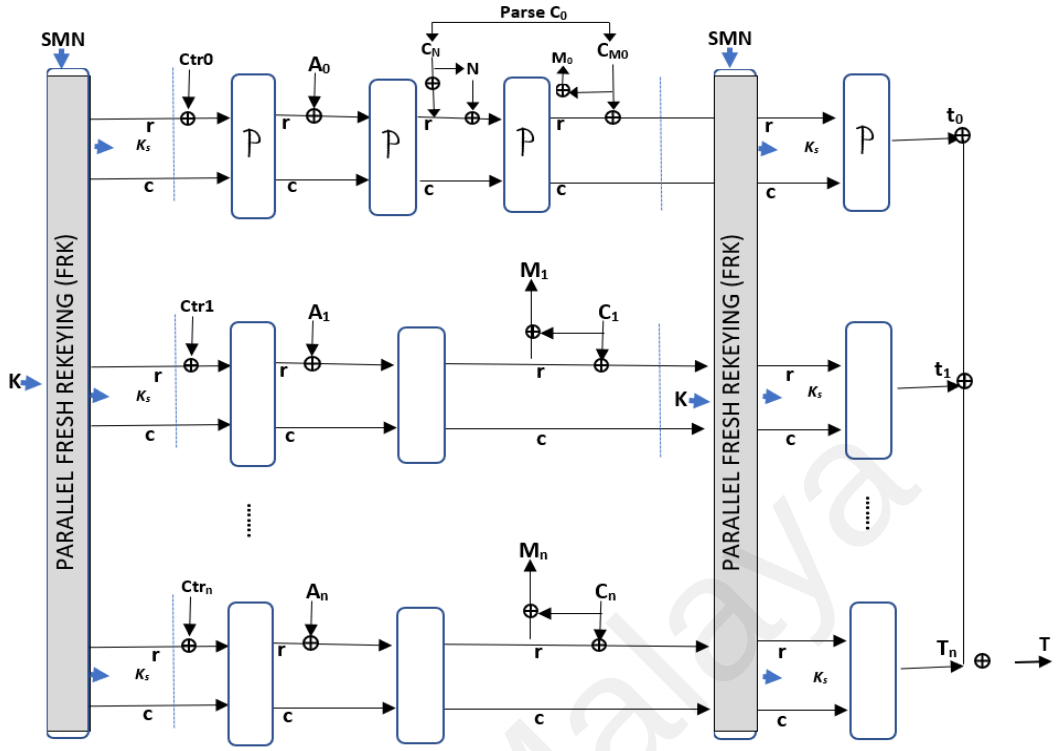

Figure 4. 4: PSASPIN decryption

Algorithm 4.1: PSASPIN processes

Authenticated Encryption
$\mathcal{E}$ (A,M,K,N)

Input: Key K ϵ {0,1}
^{k}, K≤128,
Nonce N ϵ {0,1}^{128},
Associated Data Aϵ{0,1}*,
Plaintext Mϵ{0,1}*
Output: Ciphertext Cϵ{0,1}$^{|M|}$,
Tag Tϵ{0,1}^{128}

Initialization
//Initialize a counter
Ctr=Ctr $= \lceil K \rceil_{64}$
Split A into A||1|| r-bit blocks of A_1.....A_n*
Split M into M||1|| r-bit blocks of M_1.....M_{n-1}*

//Generate a fresh subkey
K_s←FRK(K,SMN)
//Update the shared State (S)
S← P(K_s||IV||$0^{|S|-256}$)

Processing the Associated Data
for i = 1, …,x do
S←P((S$_r$⊕A$_i$)||S$_c$)
S←S⊕(0^{399}1)

Processing Nonce

//in the first thread, process on block nonce N of 128 bits.
If Ctr=0
$S_r \leftarrow S_r \oplus (N \parallel A_0 \parallel M_0)$
$N_1 \leftarrow S_r$
$S_r \leftarrow S_r \oplus (N_1)$
$S \leftarrow \mathcal{P}((S_r \oplus N_1)||S_c)$
Processing Plaintext
If Ctr=0
$S_r \leftarrow S_r \oplus M_0$
$C_{M0} \leftarrow S_r$
$C_0 \leftarrow N_1 \parallel C_{M0}$
$S \leftarrow P(C_0||S_c)$
for i = 1,…,z-1 do
$S_r \leftarrow S_r \oplus M_i$
$C_i \leftarrow S_r$
$S \leftarrow P(S_r||S_c)$
$S_r \leftarrow S_r \oplus M_z$
$C_z \leftarrow \lfloor S_r \rfloor_{|M| mod\ r}$

Finalization
$K_s \leftarrow FRK(K,SMN)$
$S \leftarrow P(S \oplus (0^r||K_s||0^{c-r-k})$
$t_i \leftarrow \lceil S \oplus K_s \rceil_{128}$
//in the last thread
$If\ Ctr = n$
$T \leftarrow t_0 \oplus t_1 \ldots . \oplus t_{z-1}$
Return $C_1||\ldots||C_i||C_z||T$

Authenticated Decryption
$\mathcal{D}$ *(A,M,K,N)*

Input: Key $K \in \{0,1\}^k$, $K \leq 128$,
Nonce $N \in \{0,1\}^{128}$,
Associated Data $A \in \{0,1\}^*$,
Plaintext $M \in \{0,1\}^*$
Output: Ciphertext $C \in \{0,1\}^{|M|}$,
Tag $T \in \{0,1\}^{128}$

Initialization
//Initialize a counter
$Ctr = Ctr = \lceil K \rceil_{64}$
Split A into A||1|| r-bit blocks of* $A_1 \ldots\ldots A_n$
Split M into M||1|| r-bit blocks of* $M_1 \ldots\ldots M_{n-1}$
//Generate a fresh subkey
$K_s \leftarrow FRK(K, SMN)$
//Update the shared State (S)
$S \leftarrow P(K_s||IV||0^{|S|-256})$

Processing the Associated Data
Split A into A||1|| r-bit blocks of* $A_1 \ldots\ldots A_x$
for i = 1, …,x do
$S \leftarrow \mathcal{P}((S_r \oplus A_i)||S_c)$
$S \leftarrow S \oplus (0^{399}1)$
// No nonce input

Processing Ciphertext

Split C into C||1|| r-bit blocks of $C_1 \ldots C_{z-1}$*

//parse C_0 into C_N and C_{M0}

$N_1 \parallel C_{M0} \leftarrow C_0$

S←𝒫((S_r⊕N_1)||S_c)

$M_0 \leftarrow S_r \oplus C_{M0}$ *;S←P(S_r||S_c)*

for I = 1,…,z-1 do

M_i←S_r⊕C_i

S←C_Mi||Sc

S←𝒫(S)

$M_z \leftarrow \lfloor S_r \rfloor_{|C_z|} \oplus C_z$

$S_r \leftarrow S_r \oplus (M_z \| 1 \| 0^*)$

Finalization

K_s←FRK(K,IV)

S ←P(S⊕ (0^r||K_s||0^{c-r-k})

$t_i \leftarrow \lceil S \oplus K_s \rceil_{128}$

//in the last thread

$If\ Ctr = n$

$T'' \leftarrow t_0 \oplus t_1 \ldots . \oplus t_{z-1}$

If T″=T

Return $M_1||\ldots||M_i||M_z$

//Fresh rekeying function

FRK(K, R)

{

*K_s=K*R*

} return K_s

Fresh Rekeying Algorithm (FRK)

$Require:\ a, b \in GF(2^8)[y]/y^d + 1$

$Ensures: c = b * b \in GF(2^2)[y]/y^d + 1$

$x \leftarrow rand(), j \leftarrow x, k \leftarrow x, with\ i = 1 - m, 0 \leftarrow st$

$while\ k \# x - 1\ mod\ d\ \boldsymbol{do}$

$k_b \leftarrow k$

$for\ i = 1\ to\ m\ \boldsymbol{do}$

$kb_i \leftarrow kb_i \oplus b_j$

$j \leftarrow j + 1\ mod\ d$

$End\ for$

$k_s \leftarrow N.bk_i$

$for\ i = 1\ to\ m\ \boldsymbol{do}$

$k_s \leftarrow N \cdot (b_j \oplus bk_i)$

$End\ for$

$End\ while$

$Return\ (k_{si}, st + 1)$

4.5 Design Rationale

The core part of the sponge construction is the permutation function, and the security of the primitive depends on it. The PSASPIN design goal was to obtain a strong permutation

with the appropriate state values. Reinforced with a rekeying function based on GF multiplication, our scheme's design philosophy protects against SCAs. The parallel design of the construction boosts the processing speed, with intermediate tags enabling earlier detection of errors. The nonce-hiding syntax of the schemes gives it an additional security layer since the nonce is not accessible to the eavesdropping adversary. If the same nonce is reused, PSASPIN is NMR and provides the best possible protection that integrity is protected and privacy is violated, revealing the minimum possible information confined to whether particular prefixes are the same. The scheme is intended to have a low memory footprint in hardware and software while still being fast, robust, and secure. It is based on Ascon permutation, which is well analyzed with known security margins. PSASPIN is single-pass, meaning it needs one forward evaluation of the permutation to perform encryption and authentication. In Summary, the scheme is designed to be fast while protecting against SPAs and DPAs, being nonce-oblivious and NMR.

4.6 PSASPIN Summary of Features:

- **Sponge-based:**
    - The scheme follows the sponge design principles, which has several advantages compared to other available construction methods like block-cipher- or hash-based modes and other dedicated constructions. For example, the sponge provides a fast and flexible permutation that eliminates the complexities of key scheduling, applies to many cryptographic primitives for hashing and AE, and is suitable for resource-constrained environments.
- **Parallelizable:**
    - The scheme performs its operations in parallel and thus is efficient, using multicore processors executing algorithms in multiple cores in parallel.

For instance, encryption or decryption of one data does not depend on the encrypted computation of any other block. In technical terms, the scheme can process the i^{th} block independently of the j^{th} block.

- **Intermediate Tags:**
  - The scheme generates intermediate tags (IT_s) after processing every message and associated data blocks at the end of every branch of the parallel threads. IT_s are accumulated by XORing them until a final tag *(T)* is generated at the end of the last block. IT_s are a vital feature to authenticate and encrypt a sequence of messages so that authenticity is assured on each (M, A) pair and the series of pairs received so far. Intermediate tags can also be helpful in practice to catch fraudulent transactions. In addition, support of intermediate tags renders a scheme well-suited for low-latency environments where messages usually contain multiple packages with small integrated checksums. Finally, IT_s also form an additional source of randomness to PSASPIN since t_j is new due to the freshness of A_i and M_i blocks, the XORing on new counter values, and the use of a new session key *(K_s)*.
- **Protection against SPAs and DPAs**
  - Using leveled implementation, PSASPIN employs combined masking, hiding, and fresh rekeying to protect its parts against SPA and DPA. For instance, the key generation function based on GF multiplication is protected using masking and shuffling to protect against DPA. In contrast, the base scheme is based on the duplex mode of the sponge construction and is protected using parallel fresh rekeying to protect against SPA. Fresh rekeying increases the key-life time (the maximum amount of data that can be encrypted under a single key). It is beneficial for preventing several

cryptanalysis methods that can recover the keys when a certain amount of data is processed.

- **Nonce-oblivious:**
    - The scheme follows the nonce-oblivious syntax of AE to prevent potential mishandling of the nonce by hiding it as part of the encrypted message. That way, PSASPIN eliminates the nonces from the decryption algorithm, so the receiver does not have to worry about handling them anymore. Since the nonce is needed in the decryption, it is extracted from the ciphertext by parsing the first ciphertext block into the ciphertext and nonce. Then, the nonce is used in the verification and the decryption at the end of the process. See algorithm 4.1 for details.
- **NMR**
    - The scheme constructs a synthetic nonce by XORing the first plaintext block *(M_0)*, the first Associated data block *(A_0)*, and the Secret Message Number *(SMN)* and applying the sponge permutation. The scheme is NMR, similar to that of Rogaway and Shrimpton (2006). It keeps its security if a repetition of nonces happens either as a result of an attack or hardware or software malfunctioning. The integrity is protected, while privacy is violated only to the extent that one can predict if the plaintext is the same as a prior and exposed only if the message and its header had been used with a particular nonce *(SMN)*. The repetition is not harmful unless a specific M_0, A_0, and *SMN* repeat together, which could happen with negligible probability.

4.7 Chapter Summary

This chapter presented and implemented the solution proposed to resolve the problems identified in this study. First, the chapter described the Parallel Sponge-based Authenticated Encryption with Side-channel Protection and adversary-invisible Nonces (PSASPIN). Next, it outlined the parameters used, introduced the security notations, and illustrated a schematic structure of the scheme. Finally, it described the initialization, encryption, authentication, decryption, and verification processes and showed the algorithms designed to implement the scheme. See the PSASPIN implementation given in the C programming language in Appendix B, also available at the following link on GitHub: https://github.com/JIMALE2/PsaspinAE

CHAPTER 5: SECURITY AND PERFORMANCE ANALYSIS

5.1 Introduction

PSASPIN protects security in several ways. First, it protects against SPA and DPA using parallel fresh rekeying. It does not use the sponge duplex construction alone but also uses a key generation function that takes a master key as input and produces several subkeys. It uses each subkey only once to make deducing the secret key challenging in case of an SCA attack.

Second, it is nonce-oblivious, using a modified syntax of NAE so that the decryption does not take a nonce as an input parameter using a nonce-hiding syntax that prevents the attackers from accessing and tampering with the nonce. In addition, it alleviates the burden of nonce handling from the receiver. Finally, the nonce is integrated with the ciphertext and extracted in the decryption to recover the plaintext.

Fourth, it is nonce-misuse resistant (NMR); that is, it provides the best security possible when the nonce is repeated. A synthetic nonce is constructed at the beginning of the process composed of the plaintext message *M*, the nonce *N*, and the associated that so that the probability of repetition is negligible.

Fifth, PSASPIN provides a security proof based on game-based theory and PRP/PRF switching lemma (Andreeva et al., 2013; Jovanovic, Luykx, & Mennink, 2014b; Rogaway & Shrimpton, 2006).

This chapter explains the method of security proof and how the performance is evaluated after implementing the scheme in the C programming language. Furthermore, It describes the test lab setup, software installations, and commands used for the test.

5.2 Security Analysis

The security of PSASPIN is measured in terms of the two levels of its implementation. At one level, the security of the rekeying function (which should be protected against DPA and SPA) and generating session keys to protect the core scheme is evaluated. At the other level, the security of the sponge function-based duplex constructions is to be protected against SPA only. In addition, the whole scheme should preserve the privacy and integrity of the data. The ability of an adversary to break the rekeyed AE can be bounded in terms of the key generation function and the base AE scheme. Several countermeasures for protection against SCAs are implemented at the hardware or software level. Examples of countermeasures are masking, hiding, using logic styles, and using session keys for a single or a small number of operations. According to Mangard et al. (2007), the best way to benefit from the countermeasures is to combine them; all the effort for protection against SCAs should not be spent on a single countermeasure. For instance, combining masking and shuffling is ideal for protection against first-order and higher-order SCAs.

A. The Security of Fresh Rekeying Function (G)

The fresh rekeying function uses an initial master key to generate session keys in the encryption of AE schemes (Abdalla & Bellare, 2000; Mennink, 2020). This method can increase the amount of data that can be encrypted with the same key (known as key lifetime). There are two types of rekeying functions: parallel rekeying, in which subkeys (session keys) are generated independently from the master key and serial generators. The

generated subkeys depend on the previous state that continuously updates (Abdalla & Bellare, 2000). According to IRTF (2019), parallel rekeying is necessary when parallel access to data is used.

Following (Abdalla & Bellare, 2000), we define the pseudorandomness of a stateful generator: Let *G =(K, N)* be a stateful generator with a block length *k*, let *n* be an integer, and Let *A* be an adversary. Consider the following experiment:

$Experiment\ EXP_{G,n,A}^{prg-real}$

$for\ i = 1, \ldots., n\ \boldsymbol{do}$

$(Out_i, St_i) \leftarrow N(St_{i-1}); \leftarrow s \parallel Out_i$

$g \leftarrow A(s)$

$return\ g$

$Experiment\ EXP_{G,n,A}^{prg-rand}$

$s \leftarrow \{0,1\}^{n.k}$

$g \leftarrow A(s)$

$return\ g$

Considering the rekeying function as a stateful pseudorandom generator, we provide the security analysis of our rekeying function regarding the security notions according to that assumption. We follow the approach of (Abdalla & Bellare, 2000), but their scheme protects a block cipher, while ours protects a sponge-based, parallel AE scheme. The desired attribute of the generator is pseudorandomness which describes the inability of adversary *A* to distinguish the generator's output from an equal-length random string. We define the advantage *(ADV)* of adversary *A* and the advantage function of the generator *G* in real and random experiments in the following manner.

$ADV_{G,n,A}^{prg} = \Pr\left[EXP_{G,n,A}^{prg-Real} = 1\right] - \Pr\left[EXP_{G,n,A}^{prg-rand} = 1\right]$

$ADV_{G,n,A}^{prg}(t) = max_A\{ADV_{G,n,A}^{prg}\}$.

The maximum is over *A* with time complexity *t*, and the time complexity is the execution time of the two experiments added to the size of the code of adversary *A*. The advantage function measures the adversary's likelihood of compromising the key generation function *G* with the mentioned resources. The security of the key generation function depends on the underlying PRF $F: \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$. Let $\{0,1\}^n \; to \; \{0,1\}^n$ be a family of functions mapping from *n* bit string to *n* bit string under a uniform distribution, if *D* is a distribution that has oracle access, then

$$ADV_{F,D}^{prf} = \Pr\left[D^{(F,K)} = 1 : K \overset{R}{\leftarrow} \{0,1\}^n\right] - \Pr\left[D^{f(.)} = 1 : f \overset{\$}{\leftarrow} R^n\right]$$

Is the advantage of the distinguisher *D*, the advantage of *F* is: $ADV_F^{prf}(t,q) = Max_D\{ADV_{F,D}^{prf}\}$,

The maximum is overall *A*, the time complexity is *t***,** and making *q* oracle queries.

The following theorem shows how the pseudorandomness of the parallel fresh rekeying function depends on the underlying PRF:

Let $F: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a PRF G[F] be a parallel fresh key generator, then: $ADV_{G[F\},n}^{prg}(t) \leq ADV_F^{prf}(t,n)$.

Proof: Let *A* be an adversary trying to compromise the pseudorandomness of *G[F]*, and let *t* be its running time of $EXP_{G[F],n,A}^{prg-real}$ and $EXP_{G[F],n,A}^{prg-rand}$. We upper bound the advantage of *A* $ADV_{G[F],n,A}^{prg}$. We construct a distinguisher *D* for *F* and relate its advantage to *A*'s advantage. The distinguisher *D* has access to an oracle *O*. It computes

$s=O(1)||.....||O(n)$ and outputs the same guess A on its input s. we could say that when the Oracle O is drawn at random from F, the probability that the distinguisher D return 1 equals the probability that $EXP_{G[F],n,A}^{prg-real}$ returns 1. On the other hand, the probability that $EXP_{G[F],n,A}^{prg-rand}$ return 1 equals that of D returning 1 when O is drawn randomly from the family of random functions R^n. As D runs in time at most t and makes at most n queries to its oracle, we obtain that: $ADV_{G[F],n,A}^{prg} \leq ADV_F^{prf}(t,n)$. If A is an arbitrary adversary and the maximum time of the two experiments is t, that concludes the proof of the theorem.

The pseudorandomness of the parallel fresh key generator (G) depends on the PRF (F) security under n queries. When F is a PRF, then we get $ADV_{G[F],n}^{prg}(t) \approx \frac{n+t}{2^k}$

B. The Security of the Rekeyed AE scheme

PSASPIN is an AE scheme based on the duplex mode of the sponge construction. It takes as input plaintext M, associated data A, a PMN nonce N, and a master secret key K used to generate session keys that are used to encrypt and decrypt messages in parallel. For The security of the sponge construction, we consider two notions of security as in the literature (Bellare & Namprempre, 2000b; Katz & Yung, 2001; Rogaway, 2002a) called privacy and integrity, which are central to the security of AE schemes. For the security proof of rekeyed part of PSASPIN, we follow the approach used by Jovanovic et al. (2014a), Andreeva et al. (2013), and Mihajloska, Mennink, and Gligoroski (2016)

C. Confidentiality (or privacy).

Confidentiality protects the secrecy of information in the case of the IND-CPA against eavesdropping adversaries and the IND-CCA against active adversaries. In the former model, the adversary is given an encryption oracle, and in the latter, the adversary is given

a decryption oracle as well; therefore, the adversary's advantage should be negligible in all cases. (Bellare & Namprempre, 2000b; Katz & Yung, 2001; Rogaway, 2002a)

Let *P* be a set of idealized permutations of a scheme *II*. Then, we define the advantage of an adversary *A*, which has access to both forward and inverse permutations in compromising the privacy of *II*:

$$ADV_{II}^{priv}(A) = |Pr_{p,K}\left(A^{p^{\pm},E_K} = 1\right) - Pr_{p,\$}(A^{PP^{\pm},\$} = 1).$$

The fact that *A* has access to both forward and inverse permutations is denoted by $P^{\pm}$. In the case of PSASPIN, *A* does not have to be nonce respecting, meaning it can use the same nonces in calling E_k and *$*. $ADV_{II}^{priv}(q_p, q_e, \lambda_e)$ denotes the maximum advantages of all adversaries that query E_k or *$*.

D. Integrity/authenticity.

Integrity ensures that the messages are from legitimate sources and have not been tampered with during transit or at rest. Furthermore, it protects the integrity of plaintext under the INT-PTXT model and the integrity of the ciphertext under the INT-CTXT model. The former ensures that the adversary cannot produce ciphertext decryption of a message that the sender had never encrypted. The latter ensures that the attacker cannot create a ciphertext that the sender has not previously produced, whether the plaintext is new or not (Bellare & Namprempre, 2000b; Katz & Yung, 2001).

Let us denote *P* as a set of underlying idealized permutations of AE scheme *II*. Then, we define the integrity-related goals of AE as captured by the inability of adversary *A* to come up with a new plaintext that had not been produced by a valid decryption $(D_k(C)$ algorithm by using the secret key *K* in the following way:

$ADV_{II}^{auth} = Pr_{P,K}\left(A^{P^{\pm},E_K,D_K}\ Forges\right)$. The probability is taken over random choices of *A, K,* and *P*. The adversary succeeds in forging if D_k returns a message that is different

from ⊥ on an input *(N, A, C, T)* where (*A, C*) have never been produced by E_k after taking *(N, A, M)* as input. We also assume that the adversary can either be nonce-respecting or non-nonce-respecting in the case of privacy. We symbolize authenticity $ADV_{\Pi}^{auth}(q_p, q_E, \lambda_E, q_D, \lambda_D)$. We denote the maximum advantage taken over all adversaries that query. $P^{\pm}$ at most q_p times that make at most q_E queries of total length at most λ_E blocks to E_K and at most q_D queries of the total length λ_D to $D_K/\perp$

In the proofs of privacy and integrity of PSASPIN in the following sections, we consider an adversary that makes q_P permutation queries and q_E encryption queries of the total length λ_E. For the proof of integrity, adversary *A* can also make q_D decryption queries of the total length λ_E we compute the number of permutation calls via q_E The exact computation is done for encryption queries with similar parameter definitions. Let us consider q_E , consisting of $\boldsymbol{c}$ Associated data blocks and *f* message blocks, and *T* intermediate tags, we describe the corresponding *n* state values in the following manner:

$$\left(init.S_0 \begin{bmatrix} A_{S1,0} & M_{S1,0} & T_{s1,0} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ A_{Sn+1,c} & M_{Sn+1,f} & T_{sn+1,T} \end{bmatrix} \right) \tag{5.1}$$

In this manner, if the j^{th} query is *c+f* blocks, then the number of state values ($\sigma_{e,j}$) is *c+f+4*; therefore, the number of *Π*- function evaluations via the encryption query is calculated as follows:

$$\sigma_E := \sum_{j=1}^{q_E} \sigma_{j,E} \leq q_E(c + f + 4) = \lambda_E + 4q_E \tag{5.2}$$

The same calculation is done for σ_D and $\sigma_{j,D}$.

E. Nonce-oblivious AE

In the light of the work of Bellare, Ng & Tackmann (Bellare et al., 2019a), nonce oblivious AE integrates the nonce with the core ciphertext in the encryption process so that the scheme does not take a nonce in the decryption process but is extracted from the ciphertext for recovery of the plaintext. Bellare et al. (2019a) proposed five ways to turn the traditional NBAE (NBE1) into a nonce-oblivious AE (NBE2) in what they termed as Hide-Nonce Transforms (from HN1 to HN5). In this work, we are interested in HN4, which provides NMR and nonce hiding, and we intend to concretize it for sponge-based AE schemes. In HN4 (Bellare et al., 2019a), A PRF *F* is applied to the triple *(M, N, H)*, as in SIV(Gueron & Lindell, 2015b), to produce a synthetic nonce N_1, which is sent as part of the Ciphertext C_2. As with SIV (Gueron & Lindell, 2015b), the security of HN4 assumes tidiness (Namprempre, Rogaway, & Shrimpton, 2014). For all *K, N, C1, H* if $S1.DEC(K,N,C1,H)=M\neq\perp$, then $SE1.ENC(K, N, M, H)=C$. Assuming that *F* is a PRF of $SE_{NH4} = HN4[SE1,l, F]$ is inherited from tradition SE1, and the authenticity is assumed only tidiness of SE_1.

Let $SE_{NH4} = HN4[SE1,l, F]$ be a nonce hiding AE as obtained above, and assume that SE1 satisfies tidiness. The adversary $A_2 \in A^{ae2}_{n-nmh} \bigcap A^{ae2}_{priv}$ making q_n queries to its new Oracle and q_e queries to it encryption queries to Encryption oracle, we construct an adversary $A_1 \in A^{ae1}_{r-n} B \bigcap A^{ae1}_{priv}$ and B_1 such that: $ADV^{ae2}_{SE2}(A_2) \leq ADV^{prf}_{F}(B_1) + ADV^{ae1}_{SE1}(A_1)$

Adversary preserves the resources of A_2 up to increasing the length of messages in Encryption queries by length (l). Adversary B_1 makes q_n to its new oracles and q_e queries to its Encryption oracle, and its running time is about that of A_2. Also given $A_2 \in A^{ae2}_{n-nmh}$ making q_n queries to its new oracle, q_e queries to its Encryption oracle, and q_v

queries to its V_F oracle, we construct an adversary B_2, such that: $\mathrm{ADV}_{\mathrm{SE2}}^{\mathrm{ae2}} \leq \mathrm{ADV}_{\mathrm{F}}^{\mathrm{prf}} + \frac{\mathrm{q_n q_v}}{2^{\mathrm{SE1.nl}}}$. Adversary B_2 makes q_n queries to its new oracle and q_e+q_v (per user) to its Fin Oracle, and its running time is about that of A_2

F. Nonce Misuse Resistant Authenticated Encryption (MRAE)

Nonces are supposed not to be repeated since most AE schemes guarantee security as long as nonces are unique, but that is not always feasible in practice. Nonce repetitions can happen because of mistakes, deliberate actions by malicious users, or a result of applications and devices malfunctioning like hardware resetting or virtual machine cloning problems (Borisov et al., 2001; Kohno, 2004; Vanhoef & Piessens, 2017; Wu, 2005). Rogaway and Shrimpton Proposed SIV construction (Rogaway & Shrimpton, 2006) as NMRAE that provides the best security possible if nonces are repeated. In SIV, the nonce is constructed by applying a PRF to the pair of the Message *(M)* and Header *(D)*, then the message is processed with the synthetic nonce *(N_1)*.

Let $F: K_1 \times \{1,0\}^{**} \rightarrow \{1,0\}^n$ be a PRF and let $\Pi = (K_1, \mathcal{E}, D)$ be a traditional IV-based encryption scheme with message space X and IV-length n. Let $\widetilde{\Pi} = SIV[F, \Pi]$. Let A be an adversary (for attaching $\widetilde{\Pi}$) with running time t and asking q queries with a total length of μ. Then there exists an adversary B and D such that:

$ADV_{\Pi}^{priv\$}(B) + ADV_{F}^{prf}(D) \geq ADV_{\widetilde{\Pi}}^{dae}(A) - \frac{q}{2^n}$; Furthermore, B and D run in time $\acute{t} = t + Time_{\Pi}(\mu) + c\mu$ for some absolute constant c and ask at most q queries with total length μ.

G. PSASPIN adversary model

Adversary *A* in this work is assumed to be powerful, having access to the communication channel, aiming to violate confidentiality and integrity, can encrypt different messages with the same nonce, and having access to encryption and decryption oracles. Figure 6 depicts the PSASPIN adversary modeled according to Do et al. framework (Do, Martini, & Choo, 2019).

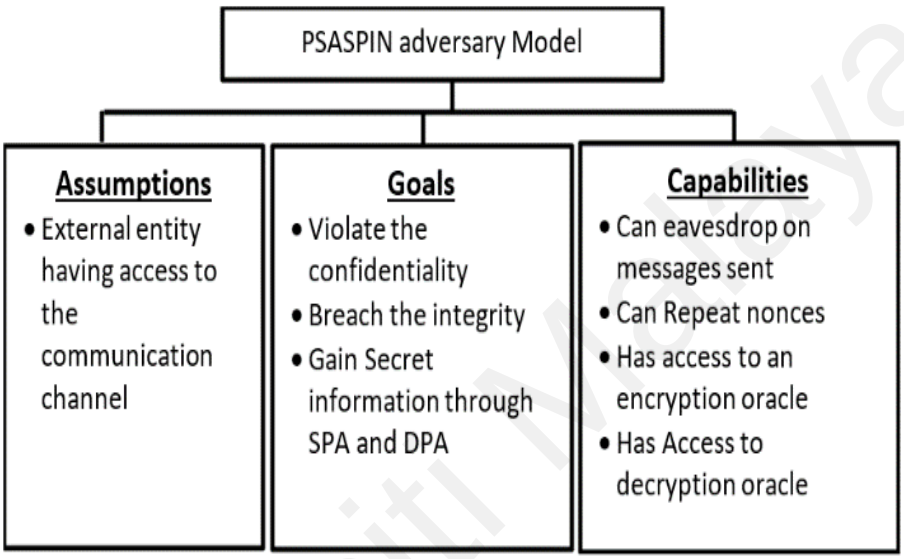

Figure 5. 1: PSASPIN adversary model

PSASPIN is secure as long as *A*, with the defined assumptions, capabilities, and goals, cannot violate its security with a non-negligible probability. It is worth noting that although *A* can use the same nonce to encrypt several messages, it cannot access the nonce (encrypted and integrated with the cyphertext) since PSASPIN hides the nonces from adversaries.

H. The Security Proof

In this subsection, the security of PSASPIN is proved in the Ideal Permutation Model, where the underlying permutation is assumed to be perfectly random, as stated by Jovanovic et al. (2014a), Andreeva et al. (2013) and Mihajloska et al. (2016), under the adversary model described in section 5. G.

1) Privacy of PSASPIN

Theorem 1: Let Π = (*E, D*) be a sponge-based AEAD based on an ideal permutation *P*, then:

$ADV_{\Pi}^{priv}(q_p, q_E, \lambda_E) \leq \frac{3(q_P+\sigma_E)^2}{2^{b+1}} + \left(\frac{8eq_P\sigma_E}{2^b}\right)^{\frac{1}{2}} + \frac{rq_P}{2^c} + \frac{q_P+\sigma_E}{2^k}$, where ϱ_E is the total number of primitive evaluations using primitive queries.

Theorem 1 implies that PSASPIN protects privacy so long as the total complexity q_p+q_E does not go beyond min $\{2^{b/2}, 2^k\}$ and the number of primitive queries does not exceed $2^{c/r}$. The proof assumes that PSASPIN is indistinguishable from a random permutation if direct and indirect evaluations of *P* do not collide. Because of using a fresh session key K_s for every encryption/decryption, the uniqueness of the nonce (*PMN and SMN*), XORing a new CounterID with the state in each branch, state values collide with probability $1/2^b$. The collisions between direct calls to *P* and indirect calls via E_K, could happen with a probability of $1/2^c$ but do not significantly affect the bound according to the multiplicity principle (Bertoni, Daemen, Peeters, & Van Assche, 2010), which limits the maximum number of states with the same rate parts.

Now Let's focus on an adversary that can interact with either ($p^{\pm}$,E_K) or ($p^{\pm}$, \$)*W*hose challenge is to distinguish between these two views. Here, the advantage can be expressed as follows:

$$ADV_{\Pi}^{priv}(A) = \Delta_A(p^{\pm}, E_K; p^{\pm}, \$). \quad (5.3)$$

To make the analysis simpler, $p^{\pm}$ is replaced by a random function $\boldsymbol{f}$ following the PRP/PRF switch lemma (Jovanovic et al., 2014a), moving from $p^{\pm}$ to $f^{\pm}$ as following: The primitive $f^{\pm}$ at first maintains an initially empty list *Q* of tuples *(x,y) of* queries and responses. The domain and range of *Q* are denoted by *dom(Q)* and *rng (Q),* respectively. For a forward query *f*(*x*), if $x \subset dom(F)$, the value which corresponds to value *y=f(x)* is retrieved. When a fresh, forward query is made, the value *x* is selected randomly from $\{0,1\}^b$. If the value *y* is already in rng(*f*), the primitive aborts, setting the flag bad to true; otherwise, the tuple *(x,y)* is added to *dom(Q*) and *rng(Q),* respectively. The description of f^{-1} is similar. Now $p^{\pm}$ and $f^{\pm}$ behave identically so long as $f^{\pm}$ does not set the 'bad' flag. Given that the adversary makes at most q_p+q_E evaluations of *f*, that abort (setting the bad flag) may happen with a likelihood of $\frac{\binom{q_p+q_E}{2}}{2^b} \leq \frac{(q_p+q_E)^2}{2^{b+1}}$. Applying the PRF/PRF switch to both sides of the inequality:

$$\triangle_A\,(p^{\pm}, E_K; p^{\pm}, \$) \leq \triangle_A\,(f^{\pm}, E_K; f^{\pm}, \$) + \frac{(q_p+\sigma_E)^2}{2^b}. \tag{5.4}$$

Specifically, let us consider an adversary *A* that has oracle access to $(f^{\pm}, F)$, where $F \in \{E_K, \$\}$. The adversary does not have to be nonce-respecting; she only makes full-block queries, and no padding rules are applied.

Queries to the function $f^{\pm}$ are represented as *(x,y)* for *i* = 1 …q_p, whereas queries to *F* are represented as elements of $(N; A_j, M_j, C_j, t_j)$ for *j* = 1…. q_E. When *F* = $E_K$the state values are as follows:

$$\left(init.S_0 \begin{bmatrix} A_{S1,0} & M_{S1,0} & t_{s1,0} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ A_{Sn+1,c} & M_{Sn+1,f} & t_{sn+1,t} \end{bmatrix} \right). \tag{5.5}$$

Two collision events, ***guess*** and ***hit***, are defined for the proof. Event ***guess*** corresponds to a primitive call to an encryption query colliding with a direct primitive query or vice-versa. Event ***hit*** is triggered when two independent states (coming from previous state values) collide in the encryption query:

let $i \in \{1 \dots q_{p\}, j, j' \in \{1 \dots q_E, k\{1,\dots,\sigma_{E,j}\}, k' \in \{1,\dots,\sigma_{E,j'}\}}$:

$$\boldsymbol{guess}(i; j, k) \equiv x_i = s_{j,k}, hit(j, k; j', k') \equiv \boldsymbol{parent}(s_j, k) \neq \boldsymbol{parent}(s_{j'}, k') \wedge s_j, k = s_{j'}, k'.$$

The remaining part of the proof is built upon the following two lemmas. **Lemma 1** shows that $(f^{\pm}, E_K)$ and $(f^{\pm}, \$)$ are indistinguishable as long as $\neg\boldsymbol{event}$ holds: $\Delta_A(f^{\pm}, E_K; f^{\pm}, \$) \leq \Pr\left(A^{f^{\pm}, E_K} \; sets \; \boldsymbol{event}\right)$.

Lemma 2 bounds these terms by $\frac{q_p q_E + \sigma_E^2/2}{2^b} + \left(\frac{8eq_p\sigma_E}{2^b}\right)^{1/2} + \frac{rq_p}{2^c} + \frac{q_p + \sigma_E}{2^k}$.

Lemma 1: Assuming that ***event*** is not triggered, $(f^{\pm}, E_K)$ and $(f^{\pm}, \$)$ are indistinguishable.

Proof: The outputs of the function $f^{\pm}$ are sampled uniformly at random in both cases of $(f^{\pm}, E_K)$ and $(f^{\pm}, \$)$.

The exception is when a collision occurs and ***guess*** occurs; however, this event is already excluded by assuming $\neg\boldsymbol{event}$.

Therefore, only queries to the oracle $F \in \{E_K, \$\}$ must be considered. Let N_j be a fresh nonce used in the j^{th} F-query with state values $(N_j; A_j, M_j)$ and with the

corresponding ciphertext and tag (C_j, t_j), and let *c* and *f* be the number of padded plaintext and associated data blocks, respectively.

By the definition of $ in the ideal world $(C_j, t_j) \xleftarrow{\$} \{0,1\}^{|M|+\tau}$, It can be proven that (C_j, t_j) is distributed identically in the real world, assuming that $\boldsymbol{guess} \vee \boldsymbol{hit}$ is not triggered.

In PSASPIN, several facts contribute to the freshness and hence the uniqueness of state values: (1) using a fresh session key in each encryption/decryption; (2) the uniqueness of the nonce values (PMN and SMN). (3) XORing the thread counter in each of the parallel lanes. So, it can be claimed that $A_{j,c}$ is fresh and that $f(A_{Sj,u})$ does not have a collision with any other *F*-query; otherwise, $\neg\boldsymbol{event}$ would have been triggered. Because $M_{Sj,0} = f(A_{Sj,c}) \oplus Ctr_0$, it can be claimed that the state $M_{Sj,0}$ is fresh and hence unique; otherwise, '**event'** would have been triggered. In the same manner, $M_{Sj,i}$ is fresh for $i > 0$. Therefore, the ciphertext blocks are computed as $C_{Sj,i} = M_{Sj,i} \oplus [f(M_{Sj,i-1})]^r$. Because the state $M_{Sj,i-1}$ has not been evaluated by *f*, it outputs a fresh random value from $(0,1)^n$, and hence $C_{j,i} \xleftarrow{\$} \{0,1\}^r$. In the tag formation process, the output of the final permutation *P* is XORed with a fresh session key and truncated to the last 128 bits $(t_i \leftarrow \lceil S \oplus K_s \rceil_{128})$. Therefore, it can be claimed that every intermediate tag is fresh and uniformly sampled from $t_j \xleftarrow{\$} \{0,1\}^\tau$ and assuming that t_j is a new input to *f*, it follows that $t_j = [f(s_j^{tag}]^\tau \xleftarrow{\$} \{0,1\}^\tau$. The Final tag *T* is produced from the unique intermediate tags $t_{js;}$ for that reason, it is also assumed to be unique.

Lemma 2: Bounding the terms

$$\Pr\left(A^{f^{\pm},E_K} \text{ sets } \boldsymbol{event}\right) =\leq \frac{q_p q_E+\frac{\sigma_E^2}{2}}{2^b} + \left(\frac{8eq_p\sigma_E}{2^b}\right)^{\frac{1}{2}} + \frac{rq_p}{2^c} + \frac{q_p+\sigma_E}{2^k} . \quad (5.6)$$

Proof: Let A be an adversary interacting with oracles $(f^{\pm}, E_K)$, and let Pr $(\boldsymbol{guess} \vee \boldsymbol{hit})$ be the probability to be bounded. For $i \in \{1, \dots, q_p\}$, the following events are defined: $key(i) \equiv [x_i]^k = K_s\ (fresh\ session\ key)$ and $key(i) = \vee_i\ key(i)$. Event $key(i)$ corresponds to the case where a primitive query collides with the session key. Let $j \in \{1, \dots, q_E\}\ and\ k \in \{1, \dots, \sigma_{E,j}\}$, and considering a threshold $\rho \geq 1$, the following is defined:

$$multi(j,k) \equiv \left[max_{\alpha\in\{0,1\}^r}|\{j' < j, 1 < k' \leq k: \alpha \in \{\left[s_{j'}, k'\right]^r, [f(s_{j'}, k')]^r\}t\}|\right] > \rho$$

The even **multi(j,k)** bounds the number of states that collide in the *r* part. The first state values $(S_{j',1})$ are not considered here because they are covered by the **key(*i*)** event. The definition $multi = multi(q_E, \sigma_{E,q_E})$ is now proposed. By basic probability:

$$\mathbf{P\,r}(guess \vee hit) \leq \mathbf{P\,r}\left(guess \vee hit \middle|_{+\boldsymbol{P}\,\boldsymbol{r}(key \vee multi)}^{\neg(key \vee multi)}\right) \quad (5.7)$$

The probabilities are bounded by considering the i^{th} forward query or inverse primitive query or the k^{th} state of the j^{th} encryption query and bounding the probability that this evaluation triggers the event $\boldsymbol{guess} \vee \boldsymbol{hit}$**,** under the assumptions that this query does not set $(\boldsymbol{key} \vee \boldsymbol{multi})$ and also that $(\boldsymbol{guess} \vee \boldsymbol{hit} \vee \boldsymbol{key} \vee \boldsymbol{multi})$ has not been triggered. For the analysis of $(\boldsymbol{key} \vee \boldsymbol{multi})$ a similar method can be used.

***Event Guess*:** this event can be triggered by the i^{th} permutation query (for $i = 1, \dots, q_p$) or in evaluating any state value of the j^{th} construction query (for $j = 1, \dots, q_E$).

Let us represent the values of the state in the j^{th} construction as in (1). Consider that any evaluation assumes that this query does not trigger $\boldsymbol{key} \vee \boldsymbol{multi}$ and also assumes that $\boldsymbol{guess} \vee \boldsymbol{hit} \vee \boldsymbol{key} \vee \boldsymbol{multi}$ has not been triggered before. First, note that $x_i = S_j^{init}$ for some i,j would imply that event $key(i)$ has been triggered and thus would annul our assumption. Therefore, $x_i = S_j^{init}$ is excluded from further analysis on $\boldsymbol{guess}$. Let $j_i \in \{1, \dots, q_E\}$ for $i = 1, \dots, q_p$ be the number of encryption queries before the i^{th} primitive query. Likewise, $e_j \in \{1, \dots, q_p\}$ for $j = 1, \dots, q_E$ shall be the number of primitive queries made before the j^{th} encryption query.

1. Consider a forward or an inverse primitive query (x_i, y_i) for $i \in \{1, \dots, q_p\}$, that has not been posed to $p^{\pm}$. If it is a forward query x_i, by $\neg multi$, there are at most ρ state values, $[x_i]^r = [s]^r$, and therefore $x_i = s$ with probability at most $\rho/2^c$. Note that the capacity part is not known to the adversary, and therefore it can guess that part with probability at most $1/2^c$. For inverse queries, the reasoning is slightly more complex. Denote the inverse query as y_i. If y_i is taken from the set of all encryption queries made before the i^{th} primitive query, the likelihood that a direct query triggers the event **guess** to the primitive evaluation is at most $\frac{q_p \rho}{2^c} + \sum_{i=1}^{q_p} \sum_{j=1}^{j_i} \frac{q_{E,j}}{2^b}$

Next, consider the likelihood that the j^{th} construction query triggers $\boldsymbol{guess}$ for $j \in \{1, \dots, q_E\}$. Consider the labeling in (1). PSASPIN branching begins at the initialization phase before the associated data part and continues until the calculation of the intermediate tag. For the associated data and the message parts, the state values are depicted as follows:

$$\begin{pmatrix} A_{S_{j,c}} \\ \vdots \\ A_{Sn+1,n} \end{pmatrix} = f(S_1^{init}) \oplus \begin{pmatrix} Ctr_0 \\ \vdots \\ Ctr_{n-1} \end{pmatrix}, \begin{pmatrix} M_{S_{j,i}} \\ \vdots \\ M_{Sn+1,n} \end{pmatrix} = f(A_{S_c}) \oplus \begin{pmatrix} Ctr_0 \\ \vdots \\ Ctr_{n-1} \end{pmatrix},$$

The *Ctr* are distinguished by XORing the associated data and the message with a branching number equal to the parallel lane number. Note that any of these nodes equals x_i of the primitive query with the probability for the associated data part $A_{j_i}/2^b$ and for the message part $M_{j_i}/2^b$, where i_j is the number of primitive queries made before the j^{th} encryption query.

In conclusion, $\Pr(guess|\neg(key \vee multi)) \leq \frac{q_p \rho}{2^c} + \sum_{i=1}^{q_p} \sum_{j=1}^{j_i} \frac{\sigma_{E,j}}{2^b} + \sum_{j=1}^{q_E} \frac{i_j \sigma_{E,j}}{2^b} = \frac{q_p \rho}{2^c} + \frac{q_p \sigma_E}{2^b}$.

This argument uses $\sum_{i=1}^{q_p} \sum_{j=1}^{j_i} \sigma_{E,j} + \sum_{j=1}^{q_E} \sum_{k=1}^{\sigma_{E,j}} i_j = q_p \sigma_E$ which follows from the counting argument.

The event **hit** is triggered when two states with different previous state values hit in the encryption queries. At initialization, it is clear that $S_j^{init} \neq S_{j'}^{init}$ because of the uniqueness of the nonce and the use of fresh K_s in every instance of PSASPIN. Here, any state value $s_{j,i}$ for $i > 1$ (out of a total of $\sigma_E - q_E$) hits, and initial state $s_{j'}$ only if $[S_{j,k}]^k = K_s(session\ key)$; this happens with probability $\sigma_E/2^k$, assuming $S_{j,k}$ is generated randomly. Finally, the other two states, $S_{j,k}\ and\ S_{j',k'}$ for $k, k' > 1$, collide with the probability $\binom{\sigma_E - q_E}{2}/2^b$. Hence, it can be concluded that $\Pr(hit|\neg(key \vee multi) \leq \frac{\binom{\sigma_E}{2}}{2^b} + q_E/2^k$.

Event key: This event is triggered when the leftmost k bit of x_i from the i^{th} primitive query, where $i \in \{1, \ldots, q_p\}$, collides with the session key K_s. The adversary A makes at most q_p attempts, and therefore $\Pr(key \leq q_p/2^k$.

Event multi: This event bounds the number of state values that collide in the rate *(r)* part of the state. Considering any new state value $s_{j,k-1}$; for a fixed value $x \in \{0,1\}^b$, it satisfies $f(S_{j,k-1}) = x$ or $S_{j,k} = f(S_{j,k-1}) \oplus v = x$ for a predetermined value of *v* with probability at most $\frac{2}{2^b}$. Now, $\alpha \in \{0,1\}^r$. More than ρ state values collide with α with probability at the most $\binom{\sigma E}{2}\left(\frac{2}{2^r}\right)^\rho \leq \left(\frac{2_e\sigma_E}{\rho 2^r}\right)^\rho$ using Stirling's approximation $\left(x! \geq \left(\frac{x}{e}\right)^x \text{ for any } x\right)$. Taking into account any possible choice of α, $\Pr(multi) \leq 2^r\left(\frac{2_e\sigma_E}{\rho 2^r}\right)^\rho$.

With the addition of the four bounds by means of (2):

$$Pr(guess \vee hit) \leq \frac{q_p\sigma_E + \sigma_E^2/2}{2^b} + \frac{q_p\rho\rho}{2^c} + \frac{q_p + \sigma_E}{2^k} + 2^r\left(\frac{2_e\sigma_E}{\rho 2^r}\right)^\rho.$$

Replacing $\rho = \max\{r, \left(\frac{2_e\sigma_E 2^c}{q_p 2^r}\right)^{1/2}\}$ gives: $\Pr(guess \vee hit) \leq \frac{q_p\sigma_E + \frac{\sigma_E^2}{2}}{2^b} + 2\left(\frac{2_e q_p\sigma_E}{2^b}\right)^{\frac{1}{2}} + \frac{rq_p}{2^c} + \frac{q_p+\sigma_E}{2^k}$, assuming that $\frac{q_p\sigma_E+\sigma_E^2/2}{2^b} < 1$; otherwise, the bound would have been invalid. This completes the proof of Theorem 1.

2) Integrity of PSASPIN

Theorem 2. Let Π = *(K, E, D)* be a sponge-based AEAD where permutation *Π* is replaced by an ideal permutation that works on a state of *b* bits, where *b* = *r*+*c*. Then:

$$ADV_\Pi^{auth}(q_p, q_E, \lambda_E, q_D, \lambda_D) \leq \frac{(q_p + \sigma_e + \sigma_D)^2}{2^b} + \left(\frac{8_e q_p\sigma_E}{2^b}\right)^{\frac{1}{2}} + \frac{rq_p}{2^c} + \frac{q_p + \sigma_E + \sigma_D}{2^k} + \frac{(q_p + \sigma_E + \sigma_D)\sigma_D}{2^c} + \frac{q_D}{2^\tau}$$

where σ_E *and* σ_D are defined in (4).

Theorem 2 indicates that PSASPIN protects integrity if it protects privacy (as in theorem 1), the number of adversarial forgery attempts σ_D is limited, and the total complexity $q_p + \sigma_E + \sigma_D$ does not exceed $\sigma_D/2^c$.

Proof. Consider an adversary *A* that has oracle access to $(p^{\pm}, E_K, D_K)$ and tries to forge an output different from $\perp$ that was not produced by the encryption oracle. The PRP/PRF switch lemma can be applied as in Theorem 1 to find that:

$$ADV_{\Pi}^{auth}(A) = \Pr\left(A^{p^{\pm},E_K,D_K}\ forges\right) \leq \Pr\left(A^{f^{\pm},E_K,D_K}\ forges\right) + \frac{\left(q_p+\sigma_E+\sigma_D\right)^2}{2^{b+1}} \quad (5.8)\ .$$

Adversary *A* is allowed to reuse nonces, and we assume that it makes only full-block queries.

In this proof, the same setting, as privacy proof, regarding the **guess** and **hit** events is adopted but has been extended to *D*-version-related events, $D_{guess}\ and\ D_{hit}$. The state values are the same as in (5) with the addition of ϑ to the subscripts, where $\vartheta\{E, D\}$. Let $i \in \{1, \dots, q_p\}, (D, j, k)$ be a decryption query index and (δ', j', k') be an encryption or decryption query index:

$$D_{guess}(i; j, k) \equiv x_i = S_{D,j,k},$$

$$D_{hit}(j, k; \vartheta', j', k') \equiv parent\left(S_{D,j,k}\right) \neq parent\left(S_{\vartheta',j',k'}\right) \wedge \left(S_{D,j,j}\right) = S_{\vartheta',j',k'},$$

Hence, it is possible to write $D_{guess} = \vee_{i;j,k}\, D_{guess}(i; j, k)$ and $D_{hit} = \vee_{j,k;\vartheta',j',k'}\, D_{hit}(j, k; \vartheta', j', k')$ and then to define:

$$event = guess \vee hit \vee D_{guess} \vee D_{hit}$$

Then according to (8):

$$\Pr\left(A^{f^{\pm},E_K,D_K}\ forges\right)$$

$$\leq \Pr\left(A^{f^{\pm},E_K,D_K}\ forges|\neg event\right) + \Pr\left(A^{f^{\pm},E_K,D_K}\ sets\ event.\ \right) \quad (5.9)$$

Lemma 3 will bound the probability that *A* sets ***event.***

This proof focuses on the probability that adversary *A* produces a forgery, assuming that an ***event*** has not happened. This case can occur when $[f(S_j^{tag})]^{\tau} = t_n$ for decryption query *j*. However, every intermediate tag in PSASPIN is new due to the freshness of the associated data and message blocks, the XORing on fresh counter values, and the use of a fresh session key K_s. Therefore, a forgery in this context could happen only with probability $1/2^{\tau}$ and summing the decryption queries q_D yields: $\Pr\left(A^{f^{\pm},E_K,D_K}\ forges|\neg event\right) \leq \frac{q_D}{2^{\tau}}$.

Lemma 3

$$\Pr\left(A^{f^{\pm},E_K,D_K}\ sets\ even\right) \leq \frac{q_p\sigma_E + \frac{\sigma_E^2}{2}}{2^b} + \left(\frac{8_e q_p \sigma_E}{2^b}\right)^{\frac{1}{2}}$$
$$+\frac{rq_p}{2^c} + \frac{q_p + \sigma_E + \sigma_D}{2^k} + \frac{(q_p + \sigma_E + \sigma_D)\sigma_D}{2^c} + \frac{(q_p + \sigma_E)\sigma_D + \frac{\sigma_D^2}{2}}{2^c}\ .$$

Proof. From Lemma 2, remember that $event = guess \vee hit \vee D_{guess} \vee D_{hit}$, and therefore:

$$\Pr\left(guess \vee hit \vee D_{guess} \vee D_{hit}\right) \leq$$
$$\Pr\left(guess \vee hit \vee D_{guess} \vee D_{hit} \middle| \neg(key \vee multi)\right)$$
$$+ \Pr(key \vee multi)\,. \quad (5.10)$$

The same techniques used to prove Lemma 2 can be used here, considering all queries and measuring the probability that '**event'** is triggered, assuming that **event** was

not set before. Note that previous queries are not influenced by the assumption that $D_{guess} \vee D_{hit}$ has not been set before.

Event D_{guess}: Note that adversary *A* is allowed to choose the rate part, and the ciphertext and the tag are known. Consequently, this event is triggered whenever there are primitive state and decryption state values that collide in the capacity part, and this can occur with probability at most $\Pr(D_{guess} | \neg(key \vee multi) \leq q_p \sigma_D / 2^c$.

Event D_{hit}: Although adversary *A* is allowed to reuse the nonces in PSASPIN, a fresh counter value Ctr_n is XORed with each branch of the parallel thread, in addition to using a fresh session key K_s and an *SMN* in every encryption or decryption. This measure gives PSASPIN an additional defensive barrier to protect confidentiality and authenticity. Note that a decryption state can collide with the initial state value with probability at most $\sigma_D / 2^k$.

Consider the j^{th} decryption query *(N; A, C; T)* that consists of $A_0, \ldots, A_{n-1}$ and $C_0, \ldots, C_{n-1}$ and write its state values as in (5). Let $(N_{\vartheta,j}; A_{\vartheta,j}, C_{\vartheta,j}; T_{\vartheta,j})$ be a previous associated data and ciphertext tuple that shares the longest common prefix with *(N; AD, C; T)*, keeping in mind that this tuple may not be unique and might come from an encryption or decryption query. Assuming that this query consists of $\vartheta c, j$ AD blocks and $z\vartheta, j$ ciphertext blocks, the state values can be written as in (5). For the rest, sub-cases can be used:

- $(N; A, C) = \left(N_{\vartheta,j}, A_{\vartheta,j}, C_{\vartheta,j}\right)$, but $T \neq T_{\vartheta,j}$. The probability is zero because all the intermediate tags in PSASPIN are new. Because of the freshness of the tag, XORing with the *Ctr* values, and the use of fresh session keys and *SMN*, the state values before the tags are new in every operation.

- $(N;A) = (N_{\vartheta,j}, A_{\vartheta,j})$, but $C \neq C_{\vartheta,j}$. If the ciphertexts $C \neq C_{\delta,j}$ are different in all their blocks; this means that the state is new and can collide with an older state value with a probability at most $1/2^c$. By summing up the encryption attempts, the $\bar{j}th$ decryption query triggers the event D_{hit} with probability $\sum_{k=1}^{\sigma_{D,\bar{j}}} \frac{\sigma_E+\sigma_{D,1}+\cdots+\sigma_{D,\bar{j}-1}+(k-1)}{2^c}$;
- If C shares the longest common prefix l with $C_{\delta,j}$ and $l<m$, then $s^C_{j,l,0} = s^C_{\vartheta,j,l,0}$ and $s^C_{j,l,1} = C_l || [s^C_{\vartheta,j,l,1}]_c \neq s^C_{\vartheta,j,l,1}$; this ensures that $s^C_{j,l,1}$ is a fresh input to f, and it can hit an older state with a probability of $1/2^c$ which is the same bound as previously determined.
- In **PSASPIN**, intermediate tags are generated at every encryption and decryption. A fresh rekeying function (FRK) feeds the finalization process with a new session key. Let ciphertexts $C \neq C_{\vartheta,j}$ be different in all their blocks; then $s^C_{j,l,1} \neq s^C_{\delta,j,l,1}$ and the intermediate tag is generated in the following manner after undergoing the final permutation: $[f(s^C_{j,l,1}) \oplus K]_{128} = t_{j,i}$. The probability that $t_{j,i}$ collides with some older state is at most $1/2^\tau$ for all possible older queries.
- $(N) = (N_{\vartheta,j})$, but $A \neq A_{\vartheta,j}$. This process is the same as in the ciphertext case, and the XORing, a fresh *Ctr* value in every process, ensures that the state value is new.
- $(N) \neq (N_{\vartheta,j})$: The nonce is fresh, and consequently, the initial state value and all subsequent state values are new. In this case, the state values do not share a prefix with any older state values.
- Summing over all queries:

- $\Pr(D_{hit} | \neg(key \vee multi)) \leq \sum_{j=1}^{q_D} \sum_{k=1}^{\sigma_{D,\bar{j}}} \frac{\sigma_E+\sigma_{D,j}+\cdots+\sigma_{D,\bar{j}-1}+(k-1)}{2^c} + \frac{\sigma_D}{2^k} \leq \frac{\sigma_E \sigma_D \binom{\sigma_D}{2}}{2^c} + \frac{\sigma_D}{2^k}$

Lemma 2 and (1) lead to:

$$\Pr(even) \leq \frac{q_p\sigma_E + \sigma_E^2/2}{2^b} + \left(\frac{8_e q_p \sigma_E}{2^b}\right) + \frac{rq_p}{2^c} + \frac{q_p + \sigma_E + \sigma_D}{2^k} + \frac{(q_p + \sigma_E)\sigma_D + \sigma_D^2/2}{2^c}$$

This completes the proof.

5.3 **Performance Analysis**

The main target of any encryption or authentication algorithm design is security. However, performance (in software or hardware) is also a significant concern in practice. Therefore, assuming that cryptographic algorithms are secure, performance is essential for developers or implementers to judge algorithms (Ankele & Ankele, 2016; Dobraunig, Eichlseder, Mendel, et al., 2019).

The need to measure a cryptographic algorithm's performance usually stems from the requirement to compare several algorithms or know how well a specific algorithm performs for applicability to specific use cases. Therefore, it is essential to include such algorithms in real-world protocols.(Ankele & Ankele, 2016).

Several benchmarking frameworks for AE schemes exist in the literature, such as SUPERCORP (Bernstein, 2016) and BRUTUS (Saarinen, 2014). In addition, Ankele & Ankele (Ankele & Ankele, 2016) proposed their framework for evaluating the software performance of 2nd round candidates of the CAESAR competition. In this work, we follow the benchmarking schemes followed by ASCON developers (Dobraunig, Eichlseder, Mendel, et al., 2019) and Krovetz & Rogaway (Krovetz & Rogaway, 2011).

PSASPIN was implemented in C language following Dobraunig et al. (Dobraunig, Eichlseder, Mendel et al., 2019) to evaluate its performance and compare it with similar schemes. Despite that PSASPIN provides more features such as parallelizability protection against SPA and DPA, adversary invisible nonces, and Nonce Misuse Resistance (NMR), its performance is comparable to other sponge-based AE schemes.

5.3.1 Test Environment setup

We implemented our C language code for performance measurement using an HP SpetreX360Convertible laptop, with processor intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz processor and RAM 16 GB. The Operating system was Windows 10 Pro Operating system version 21H2 (OS Built 19044.1466), Software Integrated Development Environment (IDE) of Microsoft Visual Studio Code version 1.63.2 (user setup), and GCC version of (Rev5, Built by MSYS2 project) 11.2.0. for compile-time optimization, we enabled the GCC compiler flags: -O3 -march=native -Wall with the framework used by Dobraunig, Eichlseder, Mendel, et al. (2019).

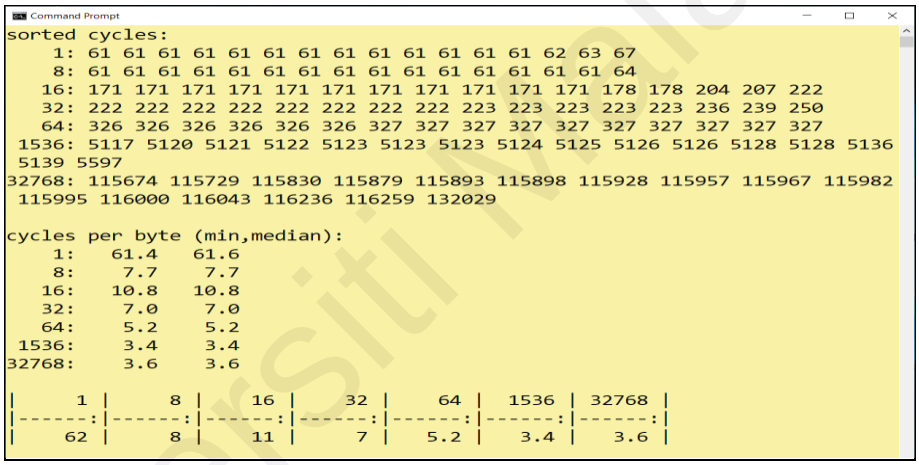

Figure 5. 2: Measuring the performance of PSASPIN

5.3.2 The result

PSASPIN uses the inverse-free ASCON permutation (Dobraunig, Eichlseder, Mendel, et al., 2019), which depends on the duplex mode of sponge construction, needing only the forward direction permutation in the encryption and its decryption. We implemented measurements on a continuous function with varying message sizes, as shown in Table 5.1 and figure 5.2. We took the mean for message sizes of all test runs. For better comparability, we represent the performance results in cycle per byte (Cpb), which is a function of the throughput of the ciphers, instead of releasing the timings and latency as done by Ankele and Ankele (2016). Figure 5.3 depicts the result and shows

that PSASPIN performance ranges from 61.4 cycles per byte for small messages and 3.6 cycles per byte for longer messages. See Table 5.1 and figure 5.4 for a comparison of PSASPIN with other AE schemes.

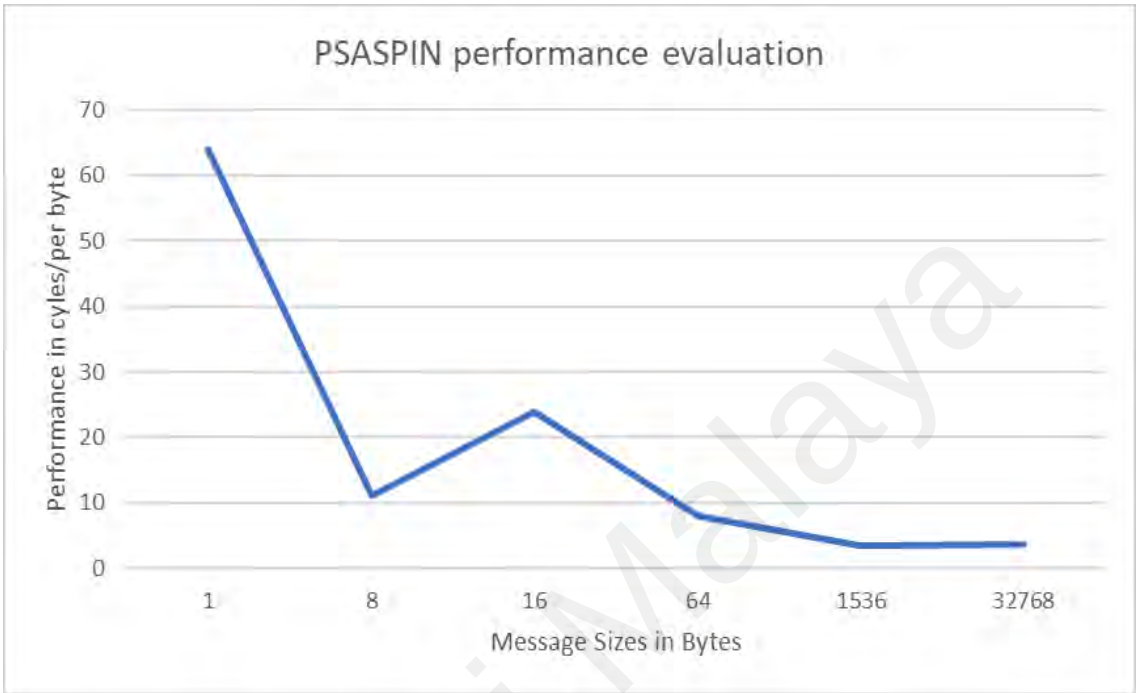

Figure 5. 3: PSASPIN performance with different message sizes

Figure 5.3 shows that PSASPIN is suited for longer messages than shorter ones, the penalty coming from calculations at the initial stages of the process. Table 5.1 and figure 5.4 compare the performance of PSASPIN to similar schemes in the literature. For instance, ASCON is a serial sponge-based AE scheme, while ISAP is a serial scheme that uses a sponge-based rekeying function for protection against SPA and DPA. At the same time, PI-cipher and NORX are sponge-based parallel AE schemes that do not protect against SPA/DPA. Finally, Table 5.1 and Figure 5.4 show that PSASPIN is not as good as the other schemes in processing shorter messages. However, it is comparable to them in longer messages while providing more features like parallelizability with side-channel protection, adversary invisible nonces, and NMR.

Table 5. 1: Comparing PSASPIN with other sponge-based AE schemes

MESSAGE SIZE IN BYTES / SCHEME	1	8	16	32	64	1536	32768
NORX (Aumasson et al., 2016)	1013	131	65	30	16.2	2.8	2.3
PI-CIPHER (Gligoroski, Mihajloska, Samardjiska, Jacobsen, El-Hadedy, et al., 2014)	2055	250	123	63	36.9	8.5	7.3
ISAP (Dobraunig, Eichlseder, et al., 2019a)	2614	337	173	93	52.9	14.4	12.9
ASCON (Dobraunig, Eichlseder, Mendel, et al., 2019)	174	20	15	9	6.2	3.3	3.1
PSASPIN	61.4	11	24	8	5.4	3.5	3.6

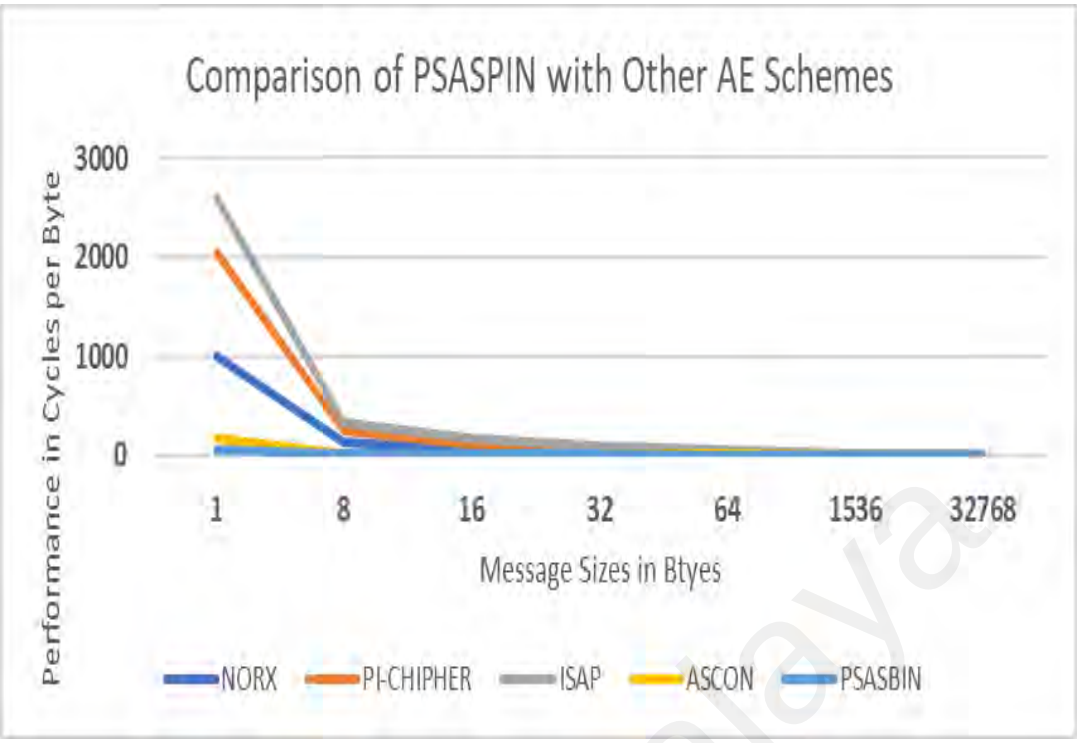

Figure 5. 4: Comparison of PSASPIN and other sponge-bases AE schemes

5.4 Chapter Summary

This chapter analyzed the security and performance of the proposed sponge-based AE scheme. The security of PSASPIN is measured in terms of the rekeying function and the sponge function-based duplex constructions. In addition, the scheme was proven to preserve the privacy and integrity of the data using game-playing theory according to the PRP/PRF switching lemma. Although the main target of any cryptographic algorithm design is security performance (in software or hardware) is also a significant concern. Several benchmarking frameworks for AE schemes exist; however, in this work, we follow the benchmarking schemes followed by ASCON developers (Dobraunig, Eichlseder, Mendel, et al., 2019) and Krovetz & Rogaway (Krovetz & Rogaway, 2011). The scheme was implemented in C language, and performance metrics were provided. See Appendix B for the implemented code in the C programming language. The code is also available at the following GitHub link: https://github.com/JIMALE2/PsaspinAE.

CHAPTER 6: DISCUSSION AND RESULTS

6.1 Introduction

PSASPIN is a complementary work to the contributions of previous researchers, from its inception to the current moment enhancing AE schemes in terms of their underlying constructions, security features, and other desirable features and boosting their performance and efficiency.

For instance, most AE schemes proposed so far were based on a block cipher and tweakable block ciphers, while sponge-based AE schemes appeared in 2011. Building blocks influence the characteristics of the AE schemes. For instance, block ciphers have a good security reputation, while stream cipher and permutation-based AEs are known for their suitability in lightweight implementations.

Many AE works focused on adding new functionalities or boosting existing algorithms for better performance using different primitives or operation modes of the same cryptographic primitives. For instance, the counter mode of the block cipher is naturally parallelizable, while the CBC mode is serial. The sponge construction and its operation, the duplex construction, are serial at the algorithm level, but several parallel sponge-based works have been proposed, although they were vulnerable to SCAs.

Other works focused on defense against SCAs, which are implementation attacks that do not rely on the weaknesses of the crypto algorithm but take advantage of sideline emissions related to power consumption patterns, deducted from processing times, or sonar or acoustic properties of the implementation environment. Several sponge-based schemes appeared in the literature for protection against SCAs, but those schemes are serial and cannot process data blocks in parallel.

This chapter reviews other related works reevaluating the status quo of sponge-based AE schemes. Finally, it discusses the findings, justifies the need for PSASPIN, underlines its importance, and compares it with previous related results to demonstrate where this work's contribution fits the existing research gaps.

6.2 Discussion

Sponge-based cryptography appeared in 2011 with Bertoni et al. (2011) and took momentum after the sponge-based hash function NIST selected Keccak as the basis for SHA3 in October 2012 ("SHA Zoo," 2013). Block ciphers were the dominant construction for AE until the first sponge-based AE was proposed by Assche (2011), followed by other researchers, including Andreeva et al. (2013); (Gligoroski, Mihajloska, Samardjiska, Jacobsen, El-Hadedy, et al., 2014; Morawiecki & Pieprzyk, 2013; Saarinen, 2013). As with other underlying constructions of AE sponge-based backed by its modes of operations like Duplex and its variants monkeyDuplex, SpongeWrap, and DonkeySponge, (Assche, 2011; Bertoni et al., 2012), came up with new design philosophies in the area of AE doing away with complexities of key scheduling and pertaining key-related attacks in block ciphers.

Protection against side-channel attacks (which benefit side-channel information instead of exploiting the weaknesses of cryptographic algorithms) is always essential. Still, it gets even more critical when cryptographic devices are deployed where they are accessible to adversaries (Dziembowski & Pietrzak, 2008; Mangard et al., 2007; Popp, 2009). Moreover, protecting against SCAs is not trivial when devices with resource constraints like IoT and low-memory smart cards are used as cryptographic devices (Dobraunig, Eichlseder, et al., 2019b; Guo, Pereira, Peters, & Standaert, 2019).

Sponge-based AE encryption schemes provide functional features that boost performance like parallelizability, and incrementality (Aumasson et al., 2016; Gligoroski, Mihajloska, Samardjiska, Jacobsen, El-Hadedy, et al., 2014), single-pass, and online (Dobraunig, Eichlseder, Mendel, et al., 2019). Morawiecki et al. (Morawiecki & Pieprzyk, 2013) proposed the first parallelizable AE scheme based on the duplex mode of the sponge construction, followed by Andreeva et al. (2013); (Aumasson et al., 2016; Gligoroski, Mihajloska, Samardjiska, Jacobsen, El-Hadedy, et al., 2014). However, the main problem with these works was that they were not protected against certain types of SCAs, especially against Simple Power Analysis (SPA) and DPA (Mangard et al., 2007; Popp, 2009). Another shortcoming of the schemes mentioned above is the potential mishandling of nonces that could compromise security if not dealt with properly with a nonce-oblivious syntax (Bellare et al., 2019a). In this work, we propose PSASPIN, a sponge-based AE that is protected against both SPA and DPA and handles the nonce in an invisible way to the adversaries.

Regarding the protection against SCAs, the authors of ISAP (Dobraunig, Eichlseder, et al., 2019b), SALE (Dobraunig, Eichlseder, et al., 2019b), and SPOOK (Bellizia et al., 2019a) proposed sponge-based AE schemes that are fortified against certain types of SCAs using different countermeasures.

For instance, AE schemes in (Degabriele et al., 2019; Dobraunig, Eichlseder, et al., 2019b) employed sponge-based structures to protect against SPA and DPA, followed by Bellizia et al. (2019a), which used a construction based on a tweakable block cipher for the same purpose. In addition, these works used leveled implantation, first proposed by Abdalla and Bellare (2000); Mangard et al. (2007). However, one shortcoming of these schemes is that they are serial and thus lack the merit of parallelizability which is vital for

performance. PSASPIN adds parallelizability to protection against SPA and DPA to bridge this gap.

Although there are several countermeasures to protect against SCA attacks, such as hiding (Mangard et al., 2007) and masking (Duc et al., 2015; Ishai et al., 2003; Mennink, 2020; Prouff & Rivain, 2013), fresh rekeying is a cheaper way to achieve the same goal. First proposed by Abdalla and Bellare (2000), the master secret key is not used directly in the scheme in fresh rekeying. Still, it is used as input to a pseudorandom generator that produces subkeys (session keys) to protect confidentiality and authenticity. Rekeying increases key lifetime, meaning the number of times the same key can be used to process data before being replaced. Abdalla and Bellare (2000); (Pereira et al., 2015) proposed a leveled implementation consisting of a rekeying part that does not have to be cryptographically strong but must be protected against both SPA and DPA, and the core scheme be cryptographically strong but needs to be protected against only SPA (Battistello, Coron, Prouff, & Zeitoun, 2016; Walter, 2001).

Medwed et al. (2011); (Medwed et al., 2010) proposed a fresh re-keying scheme for challenge-response protocols using a leveled implementation; They used a PRF based on modular multiplication, based GF(2^8) for the key generation of the AES block cipher for the rekeyed data processing part. But their scheme is vulnerable to attacks indicated by Black, Rogaway, Shrimpton, and Stam (2010), due to the weakness in key processing intermediate states of the block cipher, as indicated by (Dobraunig, Koeune, Mangard, Mendel, & Standaert, 2016). We used masking and shuffling to protect the rekeying part but applied them separately to protect their scheme against first-order DPA attacks. The proposed scheme is not vulnerable to CPA mentioned in (Dobraunig, Koeune, et al., 2016) because it uses two different constructions. In addition, PSASPIN combines masking and shuffling to protect against higher-order DPA attacks.

The authors of ISAP (Dobraunig, Eichlseder, et al., 2019b) and SALE (Degabriele et al., 2019) used a leveled implementation (although in different ways), where the rekeying part and the core rekeyed part used sponge-based constructs. According to (IRTF, 2019), although using the same primitive for rekeying and data processing can reduce the code size, the possibility of mounting a CPA attack that might compromise the following keys rises. On the other side, the authors of Spook (Bellizia et al., 2019a) used a leveled implementation where the key generation part is based on a tweakable block cipher. The data processing part is based on the sponge function (T-Sponge). Still, a tweakable block cipher is heavier than a lighter algebraic construct based on Galois field multiplication GF(2^8), which is also easier to protect against SCAs. Finally, it is worth noting that these schemes are serial and do not allow parallelism, an essential feature for cryptographic schemes. PSASPIN uses a rekeying based on GF multiplication, and the main scheme uses the duplex mode of the sponge construction. Using two different primitives for the two levels gives a safety margin against the attacks mentioned in (IRTF, 2019).

Regarding the nonce-obliviousness feature, Bellare et al. (Bellare et al., 2019a) discussed the possible weakness that could creep into AE schemes by using the wrong nonce format that could compromise privacy and proposed several options for modified syntax NAE schemes. They called those options Hide Nonce (HN) transforms. In the proposed syntax, nonces are integrated and sent as part of the ciphertext so that the schemes do not take nonces in their decryption process. Bellare et al. (2019a) Concretized their proposal for the Block cipher-based AE schemes, but it is an open problem for the sponge-based scenarios. Therefore, our scheme concretizes the nonce-hiding syntax for the Sponge-based AE schemes.

This work is motivated by ISAP (Dobraunig, Eichlseder, et al., 2019b) but differs in five ways: First, our scheme (PSASPIN) is a parallel, single-pass scheme. Second, we use different primitives for key generation and data processing (rekeyed part) for protection against the vulnerability mentioned in (IRTF, 2019). Third, our scheme provides security proof using game-playing theory based on PRP/PFR switching lemma (Andreeva et al., 2013; Jovanovic et al., 2014b; Rogaway & Shrimpton, 2006) for the sponge-based data processing part based on the work in(Abdalla & Bellare, 2000). Fourth, our scheme is nonce-oblivious; it obviates the burden of nonce communication from the application designers. Fifth, our new scheme is NMR, which tolerates nonce repetition and protects security when the nonce is reused. Finally, we used implantation of the rekeying part, which used a PRF based on polynomial multiplication based on the GF(2^8) field as in the work (Medwed et al., 2011; Medwed et al., 2010). Dobraunig et al. (Dobraunig, Koeune, et al., 2016) stated that the schemes that use Galois field multiplication (Medwed et al., 2011; Medwed et al., 2010) are vulnerable to related-key attacks that take advantage of the partial state values related to the key scheduling of block ciphers. Still, that attack is not relevant to sponge-based schemes that do not have key scheduling as in block ciphers.

6.3 Chapter Summary

This chapter went over the related works, reevaluating the status quo and demonstrating where this work's contribution fits in the existing research. Then, this chapter compared the studies done so far in sponge-based AE schemes focusing on parallelizability, protection against SCAs, nonce-hiding, and NMR. Finally, the chapter discussed the findings and justified the need for PSASPIN, underlining its importance for further advancing the research area.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Introduction

This chapter concludes the thesis by reporting on the achievement of the objectives and outlining the possible limitations and future works.

This work proposes a sponge-based nonce-oblivious, NMR, parallel, and single-pass AE scheme protected against side-channel attacks. We used a leveled implementation approach where the key generation part is a light algebraic structure based on the Galois field multiplication, and the data processing part is based on the duplex mode of the sponge construction. The security proof was provided using game-playing theory, and the performance analysis was provided after implementing the proposed scheme in the C programming language.

7.2 Research Review and Attainment of Objectives

The main aim of this research was to enhance sponge-based AE schemes in terms of security, flexibility, and performance. Three main objectives were defined at the beginning of this work; in the following, we describe how we achieved the objectives:

The first Objective of this study was to investigate the status of AE schemes in the symmetric key setting and explore their security characteristics and other functional features in-depth.

We investigated the status of AE schemes in symmetric key settings exploring their security and functional features highlighting the tremendous work that has been done so far and identifying the existing gaps. In addition, the study classified AE schemes into independent schemes, CAESAR competition schemes, and NIST lightweight competition

schemes. These schemes were then categorized according to their design approaches, security-related properties, and functional features. Our analysis reveals that a significant outstanding challenge in AE is to balance security, efficiency, and the provision of desirable features.

During the exploration, we identified several problems related to the AE schemes based on sponge construction. For instance, we noticed that most sponge-based AE schemes are serial and cannot process data blocks in parallel. Therefore, the few parallelizable schemes were vulnerable to certain SCAs, specifically SPA and DPAs. Furthermore, the sponge-based AE schemes conform to the traditional way of handling the nonces, where encryption and decryption take nonces. So, the nonce-hiding transform described by Bellare et al. (2019) would help sponge-based AE designers render nonces invisible to adversaries and boost security. Likewise, we found that only a few sponge-based AE schemes claimed to have achieved NMR, so it would be beneficial if a fully NMR sponge-based AE would be an additional security boost.

The second objective of this study was: To propose and implement a sponge-based AE scheme with the following features: nonce-oblivious, single-pass, nonce-misuse resistant (NMR), parallelizable, incremental, and protection against SPA and DPA using parallel fresh re-keying

A Parallel Sponge-based AE with Side-channel protection and Adversary invisible nonces (PSASPIN) was proposed and implemented to achieve the second objective. PSASPIN is an AEAD based on the duplex mode of the sponge construction protected against SPA and DPA using Parallel fresh re-keying. It hides nonces from the adversary by treating the nonce as part of the input to the encryption but omitting it from the decryption. Instead, it encrypts the nonces with ciphertext and extracts it at the

destination, using it in the decryption to obtain plaintext. In that way, the nonce is invisible to the adversary and alleviates the burden of nonce management from the implementors and developers.

PSASPIN AE takes four parameters: A 128-bit secret session K^* derived from the master key K, an arbitrary length plaintext message M, an arbitrary-length Associated data A, and a public message number (nonce) 128-bit N. The scheme also takes a 128-bit SMN in the encryption. Finally, the decryption takes a 128-bit secret session K^* derived from the master key K, ciphertext C, and an arbitrary-length associated data A.

Several steps are necessary for PSASPIN to do its job, from initialization and encryption/decryption to finalization and tag generation. In the initialization and finalization, Parallel Fresh Rekeying (FRK) is called to feed the process with a new session key. In addition to the traditional sponge-based AE processes, PSASPIN has an additional function for hiding the nonces called *SMN* in the encryption. The encryption and decryption processes of PSASPIN are identical except for the absence of the *SMN* nonce processing in the decryption. In decryption, the *SNM* is extracted from the ciphertext before it is used for plain text verification and recovery.

The third objective of this study was to analyze the security and performance level provided by the proposed scheme.

The implementation of PSASPIN measures security in terms of the two levels. At one level, the protection of the rekeying function (against DPA and SPA) and generating session keys to defend the core scheme is evaluated. At the other level, the protection of the sponge function-based duplex constructions against SPA only. In addition, the whole construction should preserve the privacy and integrity of the data. The ability of an

adversary to break the rekeyed AE can be bounded in terms of the key generation function and the base AE scheme. Examples of countermeasures for protection against SCAs are masking, hiding, using logic styles, and using session keys for a single or a small number of operations. However, the best way to benefit from the countermeasures is to combine them; all the efforts for protection against SCAs should not focus on a single countermeasure (Mangard et al., 2007). For instance, combining masking and shuffling is ideal for protection against first-order and higher-order SCAs. For that reason, PSASPIN uses a combination of masking and shuffling to protect its rekeying function. Finally, PSASPIN provides a security proof based on game-based theory and PRP/PRF switching lemma (Andreeva et al., 2013; Jovanovic et al., 2014b; Rogaway & Shrimpton, 2006).

Although the main motive of an encryption or authentication algorithm design is security, performance (in software or hardware) is also a significant concern in practice. Therefore, assuming that cryptographic algorithms are secure, performance is essential for developers or implementers to judge algorithms.(Ankele & Ankele, 2016; Dobraunig, Eichlseder, Mendel, et al., 2019). Several benchmarking frameworks for AE schemes exist in the literature; however, in this work, we follow the benchmarking schemes followed by ASCON developers (Dobraunig, Eichlseder, Mendel, et al., 2019) and Krovetz & Rogaway (Krovetz & Rogaway, 2011).

To evaluate its performance and compare it with similar schemes, PSASPIN was implemented in C language following the steps of Dobraunig et al. (Dobraunig, Eichlseder, Mendel, et al., 2019). Despite that PSASPIN provides more features such as parallelizability protection against SPA and DPA, adversary invisible nonces, and NMR, its performance is comparable to similar schemes, especially for large messages (see figure 5.4 in section 5.3.4 for comparison).

7.3 The Contribution of this Work

The main contribution of this work is to propose and implement a sponge-based AE scheme with the following features: nonce-oblivious, single-pass, nonce-misuse resistant (NMR), parallelizable, incremental, and protection against SPA and DPA using parallel fresh rekeying. This work is a complementary part of the continuous endeavors to enhance the AE schemes in terms of security, performance, and efficiency and is inspired by ISAP (Dobraunig, Eichlseder, et al., 2019b) and nonce-hiding schemes (Bellare et al., 2019a). However, our scheme differs from those works in five main ways: First, Parallel Sponge-based AE with Side channel protection and Adversary invisible nonces (PSASPIN) is parallel processing more than one data block simultaneously. Second, it uses the leveled implementation differently. For instance, ISAP uses sponge-based functions for rekeying and data processing, whereas PSASPIN uses a key generation PRF based on Galois Field multiplication. Although using the same construction for rekeying and data processing reduces the code size but might be susceptible to a chosen-plaintext attack (IRTF, 2019). Third, PSASPIN is nonce-oblivious, using a modified syntax of NAE so that the decryption does not take a nonce as an input parameter. Fourth, our scheme is nonce-misuse resistant (NMR), providing the best security possible when the nonce is repeated. Fifth, PSASPIN provides a security proof based on game-based theory and PRP/PRF switching lemma (Andreeva et al., 2013; Jovanovic et al., 2014b; Rogaway & Shrimpton, 2006).

Real-life applications of the proposed scheme could be anywhere AE is necessary, including Secure communications, timestamping services, internet banking, mobile money, and e-government applications where sensitive data is involved. Furthermore, PSASPIN could be particularly helpful in situations where more robust security and better performance (befitting parallelizability) are mandatory, examples being intelligence and classified government communications.

7.4 Published Articles

The following articles have been published in IEEE Access Journal. See Appendix A for the first pages of published Articles.

(1) Jimale, M., Zaba, M.R., Mat Kiah, M. L., Idris, M., Jamil, N., Mohamad, M., & Rohmad, M. (2022). Authenticated Encryption Schemes: A Systematic Review. *IEEE Access*, 1-1. doi:10.1109/ACCESS.2022.3147201

(2) Jimale, M. A., Z'aba, M.R., Kiah, M. L. B. M., Idris, M. Y. I., Rohmad, M. S. (2022). Parallel Sponge-Based Authenticated Encryption With Side-Channel Protection and Adversary-Invisible Nonces. *IEEE Access, 10*, 50819-50838. doi:10.1109/ACCESS.2022.3171853

(3) Jimale, M.A., N.A. Abdullah, M.L.B.M. Kiah, M.Y.I. Idris, M.R. Z'Aba, N. Jamil, and M.S. Rohmad, Sponge-Based Parallel Authenticated Encryption With Variable Tag Length and Side-Channel Protection. IEEE Access, 2023. 11: p. 59661-59674.

7.5 Research Limitations and Future Work

One limitation of PSASPIN is the tag-length inflexibility that it requires the use of the same tag length under the same key, and the tag size is a constant parameter per key. The issue has been raised and dealt with in the block cipher-based AE schemes by Reyhanitabar, Vaudenay, and Vizár (2016a); however, it is an open issue for sponge-based AE.

Improving the flexibility of AE schemes is also another good topic in research. The AEAD components can be flexibly arranged in the overall process. For instance, processing the plaintext before or after the associated data in environments with such flexibility is essential.

Another interesting future work could be developing schemes that provide the maximum possible security with some performance gains focusing on the prevalence of constrained devices in the future. For instance, with the rise of cloud and edge computing, another AE research direction is the application of homomorphic encryption and searchable encryption, allowing users to access data saved in the cloud without enabling the hosting service provider to read or understand it. For example, in our systematic review (Jimale et al., 2022), we found only one study related to homomorphic encryption (Cheon et al., 2018).

With the potential exhibited by quantum computing, many researchers have claimed that current cryptographic algorithms would be rendered ineffective under it. Authenticated encryption, resilient against attacks using quantum computers, is thus expected to become a popular research subject soon. Although it is believed that quantum attacks do not threaten symmetric cryptography, recent works (Kaplan et al., 2016; Santoli & Schaffner, 2016) show that many AE modes can be compromised in the superposition model.

As this study focused on AE in the symmetric key setting, conducting a comprehensive systematic literature review of AE schemes in the public key setting is also an open problem.

7.6 Chapter Summary

This chapter concluded the thesis, reported on achieving the objectives, and indicated possible future works. Using a leveled implementation approach, this work proposed a sponge-based nonce-oblivious, NMR, parallel, online, and single-pass AE scheme protected against SCAs. The key generation function used is a light algebraic structure based on the Galois field multiplication and the data processing part on the

duplex mode of the sponge construction. We proved security using game-playing theory and analyzed the performance after implementing the proposed scheme in the C programming language. Finally, the chapter indicated future some possible future enhancements to AE schemes. See Appendix B to view the implementation code in C language.

References

Aagaard, M., AlTawy, R., Gong, G., Mandal, K., & Rohit, R. (2019). ACE: An Authenticated Encryption and Hash Algorithm. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/ace-spec-round2.pdf

Aagaard, M., AlTawy, R., Gong, G., Mandal, K., Rohit, R., & Zidaric, N. (2019). WAGE: An Authenticated Cipher Submission to the NIST LWC Competition. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/wage-spec-round2.pdf

Abdalla, M., Belaïd, S., & Fouque, P.-A. (2013). *Leakage-Resilient Symmetric Encryption via Re-keying.* Paper presented at the Cryptographic Hardware and Embedded Systems - CHES, Berlin, Heidelberg.

Abdalla, M., & Bellare, M. (2000). *Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques.* Paper presented at the ASIACRYPT Berlin, Heidelberg.

Abed, F., Forler, C., & Lucks, S. (2014). General Classification of the Authenticated Encryption Schemes for the CAESAR Competition. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2014/792.pdf

Abed, F., Forler, C., & Lucks, S. (2016). General classification of the authenticated encryption schemes for the CAESAR competition. *Computer Science Review, 22*, 13-26. doi:https://doi.org/10.1016/j.cosrev.2016.07.002

Agrawal, M., Chang, D., & Sanadhya, S. (2015). A New Authenticated Encryption Technique for Handling Long Ciphertexts in Memory Constrained Devices. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2015/331.pdf

Agrawal, M., Zhou, J., & Chang, D. (2019). A Survey on Lightweight Authenticated Encryption and Challenges for Securing Industrial IoT. In C. Alcaraz (Ed.), *Security and Privacy Trends in the Industrial Internet of Things* (pp. 71-94). Cham: Springer International Publishing.

Alizadeh, J., Aref, M. R., & Bagheri, N. (2014a). Artemia v1. *Submission to CAESAR R1.* Retrieved from http://competitions.cr.yp.to/round1/artemiav1.pdf

Alizadeh, J., Aref, M. R., & Bagheri, N. (2014b). JHAE: A Novel Permutation-Based Authenticated Encryption Mode Based on the Hash Mode JH. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2014/193.pdf

An, J. H. (2001). Authenticated Encryption in the Public-Key Setting: Security Notions and Analyses. *Cryptology ePrint Archive*.

Anagnostopoulos, N. A., Arul, T., Rosenstihl, M., Schaller, A., Gabmeyer, S., & Katzenbeisser, S. (2018, 29-31 Aug. 2018). *Low-Temperature Data Remanence*

Attacks Against Intrinsic SRAM PUFs. Paper presented at the 2018 21st Euromicro Conference on Digital System Design (DSD).

Anagnostopoulos, N. A., Fan, Y., Heinrich, M., Matyunin, N., Püllen, D., Muth, P., . . . Katzenbeisser, S. (2021, 12-16 April 2021). *Low-Temperature Attacks Against Digital Electronics: A Challenge for the Security of Superconducting Modules in High-Speed Magnetic Levitation (MagLev) Trains.* Paper presented at the 2021 IEEE 14th Workshop on Low Temperature Electronics (WOLTE).

Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mendel, F., Mennink, B., . . . Yasuda, K. (2014). PRIMATEs v1. *Submission to CAESAR R1.* Retrieved from http://competitions.cr.yp.to/round1/primatesv1.pdf

Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., & Yasuda, K. (2013). APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2013/791.pdf

Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., & Yasuda, K. (2014). How to Securely Release Unverified Plaintext in Authenticated Encryption. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2014/144.pdf

Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K., . . . Nandi, M. (2016). COLM v1. *Submission to CAESAR R3.* Retrieved from https://competitions.cr.yp.to/round2/colm.pdf

Ankele, R., & Ankele, R. (2016). Software Benchmarking of the 2nd round CAESAR Candidates. *Cryptology ePrint*. Retrieved from https://eprint.iacr.org/2016/740.pdf

Antognazza, F., Barenghi, A., & Pelosi, G. (2021). Metis: An Integrated Morphing Engine CPU to Protect Against Side Channel Attacks. *IEEE Access, 9*, 69210-69225. doi:10.1109/ACCESS.2021.3077977

Assche, G. B. a. J. D. a. M. P. a. G. V. (2011). Duplexing the sponge: single-pass authenticated encryption and other applications. *Cryptology ePrint Archive*. Retrieved from https://eprint.iacr.org/2011/499.pdf

Aumasson, J.-P., Jovanovic, P., & Neves, S. (2016). NORX v3. *Submission to CAESAR R3.* Retrieved from http://competitions.cr.yp.to/round1/norxv1.pdf

Banik, S., Chakraborti, A., Iwata, T., Minematsu, K., Nandi, M., Peyrin, T., . . . Todo, Y. (2019). GIFT-COFB v1.0. *Submission to the NIST LWC Competition R2.*

Banik, S., Pandey, S. K., Peyrin, T., Sasaki, Y., Sim, S. M., & Todo, Y. (2017). *GIFT: A Small Present.* Paper presented at the Cryptographic Hardware and Embedded Systems (CHES), Cham.

Bao, Z., Chakraborti, A., Datta, N., Guo, J., Nandi, M., Peyrin, T., & Yasuda, K. (2019). PHOTON-Beetle Authenticated Encryption and Hash Family. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/photon-beetle-spec-round2.pdf

Barenghi, A., Breveglieri, L., Koren, I., & Naccache, D. (2012). Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures. *Proceedings of the IEEE, 100*(11), 3056-3076. doi:10.1109/JPROC.2012.2188769

Barwell, G., Martin, D. P., Oswald, E., & Stam, M. (2017). Authenticated Encryption in the Face of Protocol and Side Channel Leakage. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2017/068.pdf

Battistello, A., Coron, J.-S., Prouff, E., & Zeitoun, R. (2016). *Horizontal Side-Channel Attacks and Countermeasures on the ISW Masking Scheme*, Berlin, Heidelberg.

Beierle, C., Biryukov, A., Santos, L. C. d., Großschädl, J., Perrin, L., Udovenko, A., . . . Wang, Q. (2019). Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family. *Submission to the NIST LWC*

Competition R2. Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/sparkle-spec-round2.pdf

Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., . . . Sim, S. M. (2016). *The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS.* Paper presented at the CRYPTO Berlin, Heidelberg.

Belenko, A. (2014). Apple SSL/TLS Bug (CVE-2014-1266). Retrieved from https://www.nowsecure.com/blog/2014/02/23/apple-ssl-tls-bug-cve-2014-1266/

Bellare, M. (1998). Practice-Oriented Provable-Security. Retrieved from https://cseweb.ucsd.edu/~mihir/papers/pops.pdf

Bellare, M., Boldyreva, A., Knudsen, L., & Namprempre, C. (2001). *Online Ciphers and the Hash-CBC Construction.* Paper presented at the CRYPTO Berlin, Heidelberg.

Bellare, M., Desai, A., Jokipii, E., & Rogaway, P. (1997, 20-22 Oct. 1997). *A concrete security treatment of symmetric encryption.* Paper presented at the Proceedings 38th Annual Symposium on Foundations of Computer Science.

Bellare, M., Desai, A., Pointcheval, D., & Rogaway, P. (1998). *Relations among notions of security for public-key encryption schemes*, Berlin, Heidelberg.

Bellare, M., Goldreich, O., & Goldwasser, S. (1994). *Incremental Cryptography: The Case of Hashing and Signing*, Berlin, Heidelberg.

Bellare, M., Kohno, T., & Namprempre, C. (2004). Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm. *ACM Trans. Inf. Syst. Secur., 7*(2), 206–241. doi:10.1145/996943.996945

Bellare, M., & Namprempre, C. (2000a). *Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm.* Paper presented at the ASIACRYPT, Berlin, Heidelberg.

Bellare, M., & Namprempre, C. (2000b). *Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm.* Paper presented at the ASIACRYPT, Berlin, Heidelberg.

Bellare, M., & Namprempre, C. (2008). Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *Journal of Cryptology, 21*(4), 469-491. doi:10.1007/s00145-008-9026-x

Bellare, M., Ng, R., & Tackmann, B. (2019a). *Nonces Are Noticed: AEAD Revisited*, Cham.

Bellare, M., Ng, R., & Tackmann, B. (2019b). Nonces are Noticed: AEAD Revisited. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2019/624.pdf

Bellare, M., & Rogaway, P. (2005). Introduction to Modern Cryptography. In *Introduction to Modern Cryptography*: Department of Computer Science, Kemper Hall of Engineering, University of California at Davis.

Bellare, M., Rogaway, P., & Wagner, D. (2003). EAX: A Conventional Authenticated-Encryption Mode. *Cryptology ePrint Archive*.

Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., . . . Wiemer, F. (2019a). Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Spook-spec-round2.pdf

Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., . . . Wiemer, F. (2019b). Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Spook-spec-round2.pdf

Benot, O. (2011). Fault Attack. In H. C. A. van Tilborg & S. Jajodia (Eds.), *Encyclopedia of Cryptography and Security* (pp. 452-453). Boston, MA: Springer US.
Bernstein, D. (2013). CAESAR call for submissions, final (2014.01.27). Retrieved from http://competitions.cr.yp.to/caesar-call.html
Bernstein, D. J. (2013). *Failures of secret-key cryptography.* Paper presented at the FSE 2013, Singapore.
Bernstein, D. J. (2016). Supercop. eBACS: ECRYPT Benchmarking of Cryptographic Systems. Retrieved from https://bench.cr.yp.to/supercop.html
Bertoni, G., Daemen, J., Peeters, M., & Assche, G. V. (2006). The RadioGatún Hash Function Family. Retrieved from http://radiogatun.noekeon.org/
Bertoni, G., Daemen, J., Peeters, M., & ASSCHE, G. V. (2011). Cryptographic sponge functions. Retrieved from https://keccak.team/files/CSF-0.1.pdf
Bertoni, G., Daemen, J., Peeters, M., & Assche, G. V. (2012). Permutation-based encryption, authentication and authenticated encryption. Retrieved from https://keccak.team/files/KeccakDIAC2012.pdf
Bertoni, G., Daemen, J., Peeters, M., Assche, G. V., & Keer, R. V. (2016). Keyak v2. *Submission to CAESAR R3.* Retrieved from https://keccak.team/files/Keyakv2-doc2.2.pdf
Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G. (2008). *On the Indifferentiability of the Sponge Construction*, Berlin, Heidelberg.
Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G. (2010). *Sponge-Based Pseudo-Random Number Generators.* Paper presented at the Cryptographic Hardware and Embedded Systems(CHES), Berlin, Heidelberg.
Beyne, T., Chen, Y. L., Dobraunig, C., & Mennink, B. (2019). Elephant v1.1. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/elephant-spec-round2.pdf
Bilgin, B., Bogdanov, A., Knežević, M., Mendel, F., & Wang, Q. (2013). *Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware*, Berlin, Heidelberg.
Biryukov, A., & Khovratovich, D. (2014). *PAEQ: Parallelizable Permutation-Based Authenticated Encryption*, Cham.
Black, J. (2004). The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function. *Cryptology ePrint*. Retrieved from https://eprint.iacr.org/2005/210.pdf
Black, J., Rogaway, P., Shrimpton, T., & Stam, M. (2010). An Analysis of the Blockcipher-Based Hash Functions from PGV. *Journal of Cryptology, 23*(4), 519-545. doi:10.1007/s00145-010-9071-0
Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varıcı, K., & Verbauwhede, I. (2011). *spongent: A Lightweight Hash Function*, Berlin, Heidelberg.
Boldyreva, A., & Kumar, V. (2011). Provable-security analysis of authenticated encryption in Kerberos. *IET Information Security, 5*(4), 207-219. doi:10.1049/iet-ifs.2011.0041
Boldyreva, A., & Taesombut, N. (2004). *Online Encryption Schemes: New Security Notions and Constructions*, Berlin, Heidelberg.
Boorghany, A., Bayat-Sarmadi, S., & Jalili, R. (2016). Efficient Lattice-based Authenticated Encryption: A Practice-Oriented Provable Security Approach. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2016/268.pdf
Borisov, N., Goldberg, I., & Wagner, D. (2001). *Intercepting mobile communications: the insecurity of 802.11*. Paper presented at the Proceedings of the 7th annual international conference on Mobile computing and networking, Rome, Italy. https://doi-org.ezproxy.um.edu.my/10.1145/381677.381695

Brouchier, J., Dabbous, N., Kean, T., Marsh, C., & Naccache, D. (2009). Thermocommunication. *Cryptology ePrint, Paper 2009/002*. Retrieved from https://eprint.iacr.org/2009/002

Brouchier, J., Kean, T., Marsh, C., & Naccache, D. (2009). Temperature Attacks. *IEEE Security & Privacy, 7*(2), 79-82. doi:10.1109/MSP.2009.54

Campbell, T. R. (2020). Daence: Salsa20 and ChaCha in Deterministic Authenticated Encryption with no noNCEnse. *Cryptology ePrint, AP*. Retrieved from https://eprint.iacr.org/2020/067.pdf

Cassiers, G., Guo, C., Pereira, O., Peters, T., & Standaert, F.-X. (2019). *SpookChain: Chaining a Sponge-Based AEAD with Beyond-Birthday Security*, Cham.

Chakraborti, A., Chattopadhyay, A., Hassan, M., & Nandi, M. (2015). TriviA: A Fast and Secure Authenticated Encryption Scheme. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2015/590.pdf

Chakraborti, A., & Nandi, M. (2014). TriviA-ck-v1. *Submission to CAESAR R1*. Retrieved from http://competitions.cr.yp.to/round1/triviackv1.pdf

Chakraborty, B., Jha, A., & Nandi, M. (2019). On the Security of Sponge-type Authenticated Encryption Modes. *Cryptology ePrint Archive* doi:https://eprint.iacr.org/2019/1475.pdf

Chang, D., Datta, N., Dutta, A., Mennink, B., Nandi, M., Sanadhya, S., & Sibleyras, F. (2020). Release of Unverified Plaintext: Tight Unified Model and Application to ANYDAE. *IACR Transactions on Symmetric Cryptology, 2019*(4), 119-146. doi:10.13154/tosc.v2019.i4.119-146

Chari, S., Jutla, C. S., Rao, J. R., & Rohatgi, P. (1999). *Towards Sound Approaches to Counteract Power-Analysis Attacks*, Berlin, Heidelberg.

Cheng, P., Bagci, I. E., Roedig, U., & Yan, J. (2020). SonarSnoop: active acoustic side-channel attacks. *International Journal of Information Security, 19*(2), 213-228. doi:10.1007/s10207-019-00449-8

Cheon, J. H., Han, K., Hong, S., Kim, H. J., Kim, J., Kim, S., . . . Song, Y. (2018). Toward a Secure Drone System: Flying With Real-Time Homomorphic Authenticated Encryption. *IEEE Access, 6*, 24325-24339. doi:10.1109/ACCESS.2018.2819189

Cogliani, S., Maimuţ, D.-Ş., Naccache, D., do Canto, R. P., Reyhanitabar, R., Vaudenay, S., & Vizár, D. (2014). *OMD: A Compression Function Mode of Operation for Authenticated Encryption*, Cham.

competitions, C. (2019). CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. Retrieved from https://competitions.cr.yp.to/caesar.html

Daemen, J., Hoffert, S., Peeters, M., Assche, G. V., & Keer, R. V. (2019). Xoodyak, a lightweight cryptographic scheme. *Submission to the NIST LWC Competition R2*. Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Xoodyak-spec-round2.pdf

Daemen, J., & Rijmen, V. (2002). *The Design of Rijndael. AES — The Advanced Encryption Standard*: Springer.

Datta, N., & Nandi, M. (2013). Misuse Resistant Parallel Authenticated Encryptions. *Cryptology ePrint Archive*. Retrieved from https://eprint.iacr.org/2013/767.pdf

Datta, N., & Nandi, M. (2014). *ELmE: A Misuse Resistant Parallel Authenticated Encryption*, Cham.

Degabriele, J. P., Janson, C., & Struck, P. (2019). Sponges Resist Leakage: The Case of Authenticated Encryption. *Cryptology ePrint Archive* doi:https://eprint.iacr.org/2019/1034.pdf

Degabriele, J. P., & Paterson, K. G. (2010). *On the (in)security of IPsec in MAC-then-encrypt configurations*. Paper presented at the Proceedings of the 17th ACM

conference on Computer and communications security, Chicago, Illinois, USA. https://doi-org.ezproxy.um.edu.my/10.1145/1866307.1866363

Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Großschädl, J., & Biryukov, A. (2016). *Design Strategies for ARX with Provable Bounds: Sparx and LAX*, Berlin, Heidelberg.

Do, Q., Martini, B., & Choo, K.-K. R. (2019). The role of the adversary model in applied security research. *Computers & Security, 81*, 156-181. doi:https://doi.org/10.1016/j.cose.2018.12.002

Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., & Unterluggauer, T. (2019a). ISAP v2.0. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/isap-spec-round2.pdf

Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., & Unterluggauer, T. (2019b). ISAP v2.0. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/isap-spec-round2.pdf

Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., & Unterluggauer, T. (2019c). ISAP v2.0. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/isap-spec-round2.pdf

Dobraunig, C., Eichlseder, M., Mendel, F., & Schläffer, M. (2016a). Ascon v2. *Submission to CEASOR R3.* Retrieved from http://competitions.cr.yp.to/round1/asconv1.pdf

Dobraunig, C., Eichlseder, M., Mendel, F., & Schläffer, M. (2016b). Ascon v2. *Submission to CAESAR R3.* Retrieved from http://competitions.cr.yp.to/round1/asconv1.pdf

Dobraunig, C., Eichlseder, M., Mendel, F., & Schläffer, M. (2019). Ascon v1.2. *Submission to the NIST LWC Competition R2.*

Dobraunig, C., Koeune, F., Mangard, S., Mendel, F., & Standaert, F.-X. (2016, 2016//). *Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security.* Paper presented at the Smart Card Research and Advanced Applications, CHAM.

Dobraunig, C., & Mennink, B. (2019, 2019//). *Leakage Resilience of the Duplex Construction.* Paper presented at the Advances in Cryptology – ASIACRYPT 2019, Cham.

Dolev, D., Dwork, C., & Naor, M. (1991). *Non-malleable cryptography*. Paper presented at the Proceedings of the twenty-third annual ACM symposium on Theory of Computing, New Orleans, Louisiana, USA. http://delivery.acm.org/10.1145/110000/103474/p542-dolev.pdf?ip=103.18.0.18&id=103474&acc=ACTIVE%20SERVICE&key=69AF3716A20387ED%2EE7759EC8BE158239%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1569722055_2a8fb49f45cb38a6e75564b255da9070

Duc, A., Faust, S., & Standaert, F.-X. (2015). *Making Masking Security Proofs Concrete.* Paper presented at the EUROCRYPT Berlin, Heidelberg.

Dworkin, M. (2007). *Recommendation for Block Cipher Modes of Operation-Galois/Counter Mode (GCM) and GMAC*. Retrieved from

Dziembowski, S., & Pietrzak, K. (2008, 25-28 Oct. 2008). *Leakage-Resilient Cryptography.* Paper presented at the 2008 49th Annual IEEE Symposium on Foundations of Computer Science.

Engels, D., Saarinen, M.-J. O., Schweitzer, P., & Smith, E. M. (2011). The Hummingbird-2 Lightweight Authenticated Encryption Algorithm. *Cryptology ePrint Archive*. Retrieved from https://eprint.iacr.org/2011/126.pdf

Ferguson, N., Whiting, D., Schneier, B., Kelsey, J., Lucks, S., & Kohno, T. (2003). *Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive*, Berlin, Heidelberg.

FIPS. (2015). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. In.

Fleischmann, E., Forler, C., Lucks, S., & Wenzel, J. (2013). McOE- A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. Retrieved from https://eprint.iacr.org/2011/644.pdf

Fouque, P.-A., Joux, A., Martinet, G., & Valette, F. (2004). *Authenticated On-Line Encryption*, Berlin, Heidelberg.

Furuya, S., & Sakurai, K. (2003). *Single-Path Authenticated-Encryption Scheme Based on Universal Hashing*, Berlin, Heidelberg.

G. R, J., & Ganesh, R. S. (2018, 21-22 Dec. 2018). *Review of Recent Strategies in Cryptography-Steganography Based Security Techniques.* Paper presented at the 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET).

Gligor, V. D., & Donescu, P. (2002). *Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes.* Paper presented at the FSE, Berlin, Heidelberg.

Gligoroski, D., Mihajloska, H., Samardjiska, S., Jacobsen, H., El-Hadedy, M., & Jensen, R. E. (2014). π–Cipher v11. *Submission to CAESAR R1*. Retrieved from http://competitions.cr.yp.to/round1/picipherv1.pdf

Gligoroski, D., Mihajloska, H., Samardjiska, S., Jacobsen, H., Jensen, R. E., & El-Hadedy, M. (2014, 2014//). *π-Cipher: Authenticated Encryption for Big Data.* Paper presented at the Secure IT Systems, Cham.

Gueron, S., & Lindell, Y. (2015a). GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2015/102.pdf

Gueron, S., & Lindell, Y. (2015b). *GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte*. Paper presented at the Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, Colorado, USA. https://doi-org.ezproxy.um.edu.my/10.1145/2810103.2813613

Guo, C., Pereira, O., Peters, T., & Standaert, F. c.-X. (2019). Towards Low-Energy Leakage-Resistant Authenticated Encryption from the Duplex Sponge Construction. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2019/193.pdf

Harris, S. (2014). The Enchilada authenticated ciphers, v1. *Submission to CAESAR R1*. Retrieved from The Enchilada authenticated ciphers, v1

Hawkes, P., & Rose, G. G. (2003). A Mode of Operation with Partial Encryption and Message Integrity. *Cryptology ePrint*. Retrieved from https://eprint.iacr.org/2003/001.pdf

Hell, M., Johansson, T., Meier, W., Sönnerup, J., & Yoshida, H. (2019a). Grain-128AEAD - A lightweight AEAD stream cipher. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/grain-128aead-spec-round2.pdf

Hell, M., Johansson, T., Meier, W., Sönnerup, J., & Yoshida, H. (2019b). Grain-128AEAD - A lightweight AEAD stream cipher. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/grain-128aead-spec-round2.pdf

Hoang, V. T., Krovetz, T., & Rogaway, P. (2014). Robust Authenticated-Encryption: AEZ and the Problem that it Solves. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2014/793.pdf

Hoang, V. T., Reyhanitabar, R., Rogaway, P., & Vizár, D. (2015). Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2015/189.pdf

Horster, P., Michels, M., & Petersen, H. (1994). Authenticated encryption schemes with low communication costs. *Electronics Letters, 30*(15), 1212-1213. doi:10.1049/el:19940856

Hsu, C. L., & Wu, T. (1998). Authenticated encryption scheme with (t, n) shared verification. *IEE Proceedings - Computers and Digital Techniques, 145*(2), 117-120. doi:10.1049/ip-cdt:19981905

Hutter, M., & Schmidt, J. o.-M. (2014). The Temperature Side Channel and Heating Fault Attacks. *Cryptology ePrint*. Retrieved from https://eprint.iacr.org/2014/190.pdf

Inoue, A., Iwata, T., Minematsu, K., & Poettering, B. (2020). Cryptanalysis of OCB2: Attacks on Authenticity and Confidentiality. *Journal of Cryptology, 33*(4), 1871-1913. doi:10.1007/s00145-020-09359-8

IRTF, I. R. T. F. (2019). Re-keying Mechanisms for Symmetric Keys. In (Vol. Request for Comments: 8645): IETF.

Ishai, Y., Sahai, A., & Wagner, D. (2003). *Private Circuits: Securing Hardware against Probing Attacks.* Paper presented at the CRYPTO Berlin, Heidelberg.

ISO. (2019). Information security - Lightweight cryptography - Part 2: Block ciphers. In (2.0 ed.).

ISO. (2022). Information security — Encryption algorithms — Part 7: Tweakable block ciphers. In (1 ed., pp. 18).

Iwata, T. (2008). *Authenticated Encryption Mode for Beyond the Birthday Bound Security*, Berlin, Heidelberg.

Iwata, T., Khairallah, M., Minematsu, K., & Peyrin, T. (2019). Romulus v1.2. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Romulus-spec-round2.pdf

Iwata, T., Minematsu, K., Guo, J., & Morioka, S. (2017). *CLOC: Compact Low-Overhead CFB* Submission to CEASOR R3_REM. Retrieved from http://competitions.cr.yp.to/round1/clocv1.pdf

Jansen, C. J. A., & Boekee, D. E. (1988). *Modes of Blockcipher Algorithms and Their Protection Against Active Eavesdropping.* Paper presented at the EUROCRYPT' 87, Berlin, Heidelberg.

Jean, J., Nikolić, I., & Peyrin, T. (2016). Deoxys v1.41. *Submission to CAESAR R3.* Retrieved from http://competitions.cr.yp.to/round1/deoxysv1.pdf

Jian-Zhu, L., & HuoYan, C. (2000, 25-27 Oct. 2000). *Improvement of authenticated encryption scheme with (t, n) shared verification.* Paper presented at the Proceedings 24th Annual International Computer Software and Applications Conference. COMPSAC2000.

Jimale, M., Zaba, M., Mat Kiah, M. L., Idris, M., Jamil, N., Mohamad, M., & Rohmad, M. (2022). Authenticated Encryption Schemes: A Systematic Review. *IEEE Access*, 1-1. doi:10.1109/ACCESS.2022.3147201

Jimale, M. A., M. R, Z., x, aba, Kiah, M. L. B. M., Idris, M. Y. I., . . . Rohmad, M. S. (2022). Parallel Sponge-Based Authenticated Encryption With Side-Channel Protection and Adversary-Invisible Nonces. *IEEE Access, 10*, 50819-50838. doi:10.1109/ACCESS.2022.3171853

Jonsson, J. (2003). *On the Security of CTR + CBC-MAC.* Paper presented at the Selected Areas in Cryptography(SAC), Berlin, Heidelberg.

Jovanovic, P., Luykx, A., & Mennink, B. (2014a). Beyond 2c/2 Security in Sponge-Based Authenticated Encryption Modes. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2014/373.pdf

Jovanovic, P., Luykx, A., & Mennink, B. (2014b). *Beyond 2c/2 Security in Sponge-Based Authenticated Encryption Modes.* Paper presented at the ASIACRYPT Berlin, Heidelberg.

Jutla, C. S. (2001). *Encryption Modes with Almost Free Message Integrity.* Paper presented at the EUROCRYPT Berlin, Heidelberg.

Kalai, Y. T., & Reyzin, L. (2019). A Survey of Leakage-Resilient Cryptography. *Cryptology ePrint*. Retrieved from https://eprint.iacr.org/2019/302.pdf

Kamara, S. (2017). Lectures 2+3: Provable Security. *Topics in Applied Cryptography.* Retrieved from https://cs.brown.edu/~seny/2950-v/

Kaplan, M., Leurent, G., Leverrier, A., & Naya-Plasencia, M. (2016). Breaking Symmetric Cryptosystems using Quantum Period Finding. *Cryptography and Security (cs.CR)*.

Karaklajić, D., Schmidt, J., & Verbauwhede, I. (2013). Hardware Designer's Guide to Fault Attacks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 21*(12), 2295-2306. doi:10.1109/TVLSI.2012.2231707

Katz, J., & Yung, M. (2001). *Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation.* Paper presented at the FSE, Berlin, Heidelberg.

Kefei, C. (1998). Authenticated encryption scheme based on quadratic residue. *Electronics Letters, 34*(22), 2115-2116. doi:10.1049/el:19981408

Khan, A. N., Kiah, M. L. M., Khan, S. U., Madani, S. A., & Khan, A. R. (2013, 22-25 Sept. 2013). *A study of incremental cryptography for security schemes in mobile cloud computing environments.* Paper presented at the 2013 IEEE Symposium on Wireless Technology & Applications (ISWTA).

Kim, T., & Shin, Y. (2022). ThermalBleed: A Practical Thermal Side-Channel Attack. *IEEE Access, 10*, 25718-25731. doi:10.1109/ACCESS.2022.3156596

Kocher, P., Jaffe, J., & Jun, B. (1999). *Differential Power Analysis*, Berlin, Heidelberg.

Kocher, P. C. (1996). *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, Berlin, Heidelberg.

Kohno, T. (2004). *Attacking and repairing the winZip encryption scheme*. Paper presented at the Proceedings of the 11th ACM conference on Computer and communications security, Washington DC, USA. https://doi-org.ezproxy.um.edu.my/10.1145/1030083.1030095

Krämer, J., & Struck, P. (2020). Leakage-Resilient Authenticated Encryption from Leakage-Resilient Pseudorandom Functions. *Cryptology ePrint, AP*. Retrieved from https://eprint.iacr.org/2020/280.pdf

Krawczyk, H. (2001). *The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)*, Berlin, Heidelberg.

Krovetz, T. (2014). HS1-SIV (Draft v2). *Submission to CAESAR R1.* Retrieved from http://competitions.cr.yp.to/round1/hs1siv-nh.pdf

Krovetz, T., & Rogaway, P. (2011). *The Software Performance of Authenticated-Encryption Modes*, Berlin, Heidelberg.

Krovetz, T., & Rogaway, P. (2016). OCB (v1.1). *Submission to CAESAR R3.* Retrieved from http://competitions.cr.yp.to/round1/ocbv1.pdf

Kuhn, M. G., & Anderson, R. J. (1998). *Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations*, Berlin, Heidelberg.

Kwon, D., Hong, S., & Kim, H. (2022). Optimizing Implementations of Non-Profiled Deep Learning-Based Side-Channel Attacks. *IEEE Access, 10*, 5957-5967. doi:10.1109/ACCESS.2022.3140446

Kwon, D., Kim, H., & Hong, S. (2021). Non-Profiled Deep Learning-Based Side-Channel Preprocessing With Autoencoders. *IEEE Access, 9*, 57692-57703. doi:10.1109/ACCESS.2021.3072653

Lee, J., & Han, D.-G. (2020). Security analysis on dummy based side-channel countermeasures—Case study: AES with dummy and shuffling. *Applied Soft Computing, 93*, 106352. doi:https://doi.org/10.1016/j.asoc.2020.106352

Li, S. (2017). Chapter 1 - Introduction: Securing the Internet of Things. In S. Li & L. D. Xu (Eds.), *Securing the Internet of Things* (pp. 1-25). Boston: Syngress.

Lucks, S. (2005). *Two-Pass Authenticated Encryption Faster Than Generic Composition*, Berlin, Heidelberg.

Mangard, S., Oswald, E., & Popp, T. (2007). *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Boston, MA: Springer US.

Martin, K. (2012). *Everyday Cryptography*: Oxford Press.

McGrew, D. (2008). An Interface and Algorithms for Authenticated Encryption. In (Vol. Request for Comments: 5116): IETF Network Working Group.

Medwed, M., Petit, C., Regazzoni, F., Renauld, M., & Standaert, F.-X. (2011). *Fresh Re-keying II: Securing Multiple Parties against Side-Channel and Fault Attacks.* Paper presented at the CARDIS, Berlin, Heidelberg.

Medwed, M., Standaert, F.-X., Großschädl, J., & Regazzoni, F. (2010). *Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices.* Paper presented at the AFRICACRYPT Berlin, Heidelberg.

Mennink, B. (2020). Beyond Birthday Bound Secure Fresh Rekeying: Application to Authenticated Encryption. *Cryptology ePrint, AP*. Retrieved from https://eprint.iacr.org/2020/1082.pdf

Mihajloska, H., Mennink, B., & Gligoroski, D. (2016). π-Cipher with Intermediate Tags. Retrieved from http://pi-cipher.org/upload/Pi-Cipher%20with%20intermediate%20tags.pdf

Minematsu, K. (2009). *Beyond-Birthday-Bound Security Based on Tweakable Block Cipher*, Berlin, Heidelberg.

Morawiecki, P., & Pieprzyk, J. (2013). Parallel authenticated encryption with the duplex construction. *Cryptology ePrint Archive*. Retrieved from https://eprint.iacr.org/2013/658.pdf

Morris Dworkin. (2007). *Recommendation for Block Cipher Modes of Operation- The CCM Mode for Authentication and Confidentiality*. Retrieved from

Mukhtar, N., Fournaris, A. P., Khan, T. M., Dimopoulos, C., & Kong, Y. (2020). Improved Hybrid Approach for Side-Channel Analysis Using Efficient Convolutional Neural Network and Dimensionality Reduction. *IEEE Access, 8*, 184298-184311. doi:10.1109/ACCESS.2020.3029206

Naito, Y., & Sugawara, T. (2019). Lightweight Authenticated Encryption Mode of Operation for Tweakable Block Ciphers. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2019/339.pdf

Namprempre, C., Rogaway, P., & Shrimpton, T. (2014). *Reconsidering Generic Composition*, Berlin, Heidelberg.

NIST. (2020). Lightweight Cryptography. Retrieved from https://www.nist.gov/programs-projects/lightweight-cryptography

Nyberg, K., & Rueppel, R. A. (1995). *Message recovery for signature schemes based on the discrete logarithm problem*, Berlin, Heidelberg.

Paar, C., & Pelzl, J. (2010). *Understanding Cryptography*. Verlag Berlin Heidelberg: Springer.

Paterson, K. G., & Watson, G. J. (2012). Authenticated-Encryption with Padding: A Formal Security Treatment. In D. Naccache (Ed.), *Cryptography and Security: From Theory to Applications: Essays Dedicated to Jean-Jacques Quisquater on*

the Occasion of His 65th Birthday (pp. 83-107). Berlin, Heidelberg: Springer Berlin Heidelberg.

Peña, P. I. S., & Torres, R. E. G. (2016, 26-30 Sept. 2016). *Authenticated Encryption based on finite automata cryptosystems.* Paper presented at the 2016 13th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE).

Pereira, O., Standaert, F.-X., & Vivek, S. (2015). *Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives*. Paper presented at the Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, Colorado, USA. https://doi-org.ezproxy.um.edu.my/10.1145/2810103.2813626

Peyrin, T. I. a. M. K. a. K. M. a. T. (2019). Duel of the Titans: The Romulus and Remus Families of Lightweight AEAD Algorithms. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2019/992.pdf

Picek, S., Heuser, A., Jovic, A., Ludwig, S. A., Guilley, S., Jakobovic, D., & Mentens, N. (2017, 14-19 May 2017). *Side-channel analysis and machine learning: A practical perspective.* Paper presented at the 2017 International Joint Conference on Neural Networks (IJCNN).

Popp, T. (2009). *An introduction to implementation attacks and countermeasures*. Paper presented at the Proceedings of the 7th IEEE/ACM international conference on Formal Methods and Models for Codesign, Cambridge, Massachusetts.

Prouff, E., & Rivain, M. (2013). *Masking against Side-Channel Attacks: A Formal Security Proof.* Paper presented at the EUROCRYPT Berlin, Heidelberg.

Quisquater, J.-J., & Samyde, D. (2002). Eddy current for magnetic analysis with active sensor.

Rackoff, C., & Simon, D. R. (1992). *Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack*, Berlin, Heidelberg.

Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3. In: IETF.

Reyhanitabar, R., Vaudenay, S., & Vizár, D. (2016a). Authenticated Encryption with Variable Stretch. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2016/463.pdf

Reyhanitabar, R., Vaudenay, S., & Vizár, D. (2016b, 2016//). *Authenticated Encryption with Variable Stretch.* Paper presented at the Advances in Cryptology – ASIACRYPT 2016, Berlin, Heidelberg.

Riou, S. (2019). DryGASCON, Lightweight Cryptography Standardization Process round 1 submission. *Submission to NIST.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/drygascon-spec-round2.pdf

Rogaway, P. (2002a). *Authenticated-encryption with associated-data*. Paper presented at the Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA. https://doi-org.ezproxy.um.edu.my/10.1145/586110.586125

Rogaway, P. (2002b). *Authenticated-encryption with associated-data*. Paper presented at the Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA. https://dl.acm.org/citation.cfm?doid=586110.586125

Rogaway, P. (2004a). *Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC*, Berlin, Heidelberg.

Rogaway, P. (2004b). *Nonce-Based Symmetric Encryption*, Berlin, Heidelberg.

Rogaway, P., Bellare, M., & Black, J. (2003). OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur., 6*(3), 365–403. doi:10.1145/937527.937529

Rogaway, P., Bellare, M., Black, J., & Krovetz, T. (2001). *OCB: a block-cipher mode of operation for efficient authenticated encryption*. Paper presented at the Proceedings of the 8th ACM conference on Computer and Communications Security, Philadelphia, PA, USA.

Rogaway, P., & Shrimpton, T. (2006). Deterministic Authenticated-Encryption: A Provable-Security Treatment of the Key-Wrap Problem. *Cryptology ePrint Archive*. Retrieved from https://eprint.iacr.org/2006/221.pdf

Saarinen, M.-J. O. (2013). CBEAM: Efficient Authenticated Encryption from Feebly One-Way ϕ Functions. *Cryptology ePrint Archive*. Retrieved from https://eprint.iacr.org/2013/773.pdf

Saarinen, M.-J. O. (2014). The BRUTUS automatic cryptanalytic framework: Testing CAESAR authenticated encryption candidates for weaknesses. *Cryptology ePrint Archive* Retrieved from https://eprint.iacr.org/2014/850.pdf

Samyde, D., Skorobogatov, S., Anderson, R., & Quisquater, J. (2002, 11-11 Dec. 2002). *On a new way to read data from memory.* Paper presented at the First International IEEE Security in Storage Workshop, 2002. Proceedings.

Santoli, T., & Schaffner, C. (2016). Using Simon's Algorithm to Attack Symmetric-Key Cryptographic Primitives. *Quantum Information & Computation, volume 17 no.1&2*, pages 65-78, 2017.

Sasaki, Y., & Yasuda, K. (2016). *A New Mode of Operation for Incremental Authenticated Encryption with Associated Data.* Paper presented at the SAC Cham.

SHA zoo. (2013). Retrieved from https://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo

Shrimpton, T. (2004). A Characterization of Authenticated-Encryption as a Form of Chosen-Ciphertext Security. *Cryptology ePrint Archive*. Retrieved from https://eprint.iacr.org/2004/272.pdf

Stallings, W. (2014). *Cryptography and Network Security Principles and Practice* (4th ed.): Pearson Higher Education.

Tahir, R., Javed, M. Y., & Cheema, A. R. (2008, 20-23 June 2008). *Rabbit-MAC: Lightweight Authenticated Encryption in Wireless Sensor Networks.* Paper presented at the 2008 International Conference on Information and Automation.

Tsang, P. P., & Smith, S. W. (2008). *Secure Cryptographic Precomputation with Insecure Memory*, Berlin, Heidelberg.

Vanhoef, M., & Piessens, F. (2017). *Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2*. Paper presented at the Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, Texas, USA. https://doi-org.ezproxy.um.edu.my/10.1145/3133956.3134027

Vaudenay, S. (2002). *Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS*, Berlin, Heidelberg.

Walter, C. D. (2001). *Sliding Windows Succumbs to Big Mac Attack*, Berlin, Heidelberg.

Whiting, D., Housley, R., & Ferguson, N. (2003). *Counter with CBC-MAC (CCM)*: RFC Editor.

Wu, H. (2005). The Misuse of RC4 in Microsoft Word and Excel. *Cryptology ePrint*. Retrieved from https://eprint.iacr.org/2005/007

Wu, H. (2016). ACORN- A Lightweight Authenticated Cipher (v3). *Submission to CAESAR R3.* Retrieved from http://competitions.cr.yp.to/round1/acornv1.pdf

Wu, H., & Huang, T. (2019). TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms. *Submission to the NIST LWC Competition R2.* Retrieved from https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/TinyJAMBU-spec-round2.pdf

Wu, H., & Preneel, B. (2016). AEGIS- A Fast Authenticated Encryption Algorithm (v1.1). *Submission to CAESAR R3.* Retrieved from http://competitions.cr.yp.to/round1/aegisv1.pdf

Xin, L., Zhi, M., & Deng-Guo, F. (2004, 31 Aug.-4 Sept. 2004). *A quantum authenticated encryption scheme.* Paper presented at the Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04. 2004.

Zhang, F., Liang, Z.-y., Yang, B.-l., Zhao, X.-j., Guo, S.-z., & Ren, K. (2018). Survey of design and security evaluation of authenticated encryption algorithms in the CAESAR competition. *Frontiers of Information Technology & Electronic Engineering, 19*(12), 1475-1499. doi:10.1631/FITEE.1800576

Zhang, L., Wu, W., Sui, H., & Wang, P. (2014). iFeed[AES] v1. *Submission to CAESAR R1.* Retrieved from http://competitions.cr.yp.to/round1/ifeedaesv1.pdf

Zheng, Y. (1997). *Digital signcryption or how to achieve cost(signature & encryption) ≪ cost(signature) + cost(encryption)*, Berlin, Heidelberg.

Zoltak, B. (2004). VMPC-MAC: A Stream Cipher Based Authenticated Encryption Scheme. *Cryptology ePrint Archive*. Retrieved from https://eprint.iacr.org/2004/301.pdf