IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 6, 2022, accepted April 26, 2022, date of publication May 2, 2022, date of current version May 16, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3171853

Parallel Sponge-Based Authenticated Encryption With Side-Channel Protection and Adversary-Invisible Nonces

MOHAMUD AHMED JIMALE[1], MUHAMMAD REZA Z'ABA[1], MISS LAIHA BINTI MAT KIAH[1], (Senior Member, IEEE), MOHD YAMANI IDNA IDRIS[1], (Member, IEEE), NORZIANA JAMIL[2], MOESFA SOEHEILA MOHAMAD[3], AND MOHD SAUFY ROHMAD[4]

[1]Department of Computer System and Technology, Faculty of Computer Science and Information Technology, Universiti Malaya, Kuala Lumpur 50603, Malaysia
[2]College of Computing and Informatics, Universiti Tenaga Nasional, Kajang, Selangor 43000, Malaysia
[3]Information Security Laboratory, MIMOS Berhad, Kuala Lumpur 57000, Malaysia
[4]Faculty of Electrical Engineering, Universiti Teknologi MARA, Shah Alam, Selangor 40450, Malaysia

Corresponding author: Mohamud Ahmed Jimale (mahamudjimale@gmail.com)

This work was supported by the Fundamental Research Grant Scheme (FRGS) of the Ministry of Higher Education, Malaysia, under Project FP072-2019A (reference code FRGS/1/2019/ICT05/UM/02/1).

ABSTRACT Since its birth in 2000, authenticated encryption (AE) has been a hot research topic, and many new features have been proposed to boost its security or performance. The Block cipher was the dominant primitive in constructing AE schemes, followed by stream ciphers and compression functions until the sponge construction emerged in 2011. 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. Currently, there exist parallel sponge-based AE schemes, but they are not protected against 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.

INDEX TERMS Integrity, authenticated encryption, authentication, confidentiality, CAESAR competition, message authentication code, NIST-LW competition, cryptographic sponge function.

I. INTRODUCTION

A. BACKGROUND

The use of authenticated encryption (AE) schemes is a crucial element for secure communications to protect the confidentiality and integrity of messages. One of the most popular protocols for protecting Internet communications, the Transport Layer Security (TLS), has already phased out non-AE schemes in its current version (1.3), first defined in 2018.

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru.

Encryption primitives such as block and stream ciphers only provide confidentiality, i.e., messages are protected from being viewed by unauthorized entities. Such primitives cannot be simply used in secure communications since an adversary can tamper with the encrypted message (i.e., ciphertext) without detection. In that regard, [1] and [2] came up with the first AE schemes by integrating encryption and message authentication code (MAC) schemes. AE with associated data (AEAD) offers authentication of additional unencrypted chunks of data [3], [4]. A classic case is a network packet header, where only the message must be

encrypted, but the header and the encrypted content must be authenticated.

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 the ''generic compositions'': (1) Encrypt-Then-MAC, in which the message is first encrypted, and then the authenticated tag is generated on encrypted message; (2) Encrypt-and-MAC, in which the encryption and authentication algorithms are independently applied and then combined; and (3) MAC-then-Encrypt in which the authenticated tag is generated on the plaintext, and then the tag and the message are both encrypted.

Researchers proposed Dedicated AE schemes to resolve efficiency problems associated with the generic composition paradigm. Though the idea may have been studied earlier in 1987 by Jansen and Boekee [5], the earliest practical designs came at the beginning of the 21st century by Katz and Yung [2] and then followed by other works [6]–[8]. This new type of dedicated AE scheme utilizes a single key, in contrast to the generic composition approach requiring two separate keys for encryption and authentication [9].

Since the seminal articles of Bellare and Namprempre [1], [10], AE has experienced continuous improvements. The belief that AE schemes could still be refined led to the Competition for AE: Security, Applicability, and Robustness (CAESAR) project, jointly commenced in 2013 by the U.S. National Institute of Standards and Technology (NIST) and Dan Bernstein. NIST declared the final CAESAR portfolio (winners) in 2018, consisting of six schemes [11]–[16]. In the same year, triggered by the upsurge 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). On March 29th, 2021, NIST announced ten finalists out of the 32 candidates from Round 2 as the final portfolio for standardization [17].

AE protects data confidentiality (or privacy) and integrity/authenticity in two forms, each of which branches into two notions of security. Confidentiality protects the secrecy of information in case of the Chosen-plaintext attack (IND-CPA) against passive adversaries and chosen ciphertext attack (IND-CCA) against active adversaries [2], [18], [19]. Integrity guarantees that the messages are from legitimate sources and have not been tampered with during transit at rest. It protects the integrity of plaintext under the INT-PTXT model and that of the ciphertext under the INT-CTXT model.

AE schemes rely on a user-supplied value (a nonce) as an input to the AE scheme that is not supposed to be repeated to encrypt different plaintexts under the same key [8], [19]. Nonces do not have to be random; they must be different for each successive usage. An example is a counter that increases with every new encryption [20].

The way that the nonces are generated or transmitted is the responsibility of Application developers. However, such practice is vulnerable to misuse because reusing nonces (intentionally or otherwise) can have serious consequences. The security of numerous applications and protocols has been abused due to the mishandling of nonces. Examples of violated applications are Wired Equivalent Privacy (WEP) [21], WinZip [22], Wi-Fi-protected access (WPA) 2 [23], and Microsoft Office [24]. Consequently, it is necessary to have AE schemes that provide an acceptable level of protection in the face of such violations. To deal with this issue, Rogaway and Shrimpton [25] proposed the notion of a nonce misuse-resistant AE (MRAE) in 2006. An MRAE scheme guarantees an acceptable level of security even though nonces are reused [25].

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 plain to the receiver along with ciphertext [19], [26], [27]. On the other extreme, [28] stated that sending nonces in clear or using values like DeviceId could compromise security.

To bridge the gap between theory and practice of nonce handling, Bellare *et al.* [28] proposed an approach to alleviate the nonce handling burden and prevent its potential misuse 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, in [28], the authors demonstrated simple ways to turn traditional nonce-based authenticated encryption schemes into nonce-oblivious ones for the block cipher-based AE schemes. However, nonce oblivious AE schemes based on Sponge construction remain an open problem.

In addition to analyzing an AE scheme under the mentioned security models, 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 like IoT devices, sensor network nodes, and smart cards are in the hands of adversaries or are mounted where they are accessible to the general public [29], [30], [31]. Cryptographic algorithms are notably weaker when used in their lightweight form, as Ishai *et al.* [35] stated. Several techniques are in place to prevent SCAs, including hiding [32], masking [30], [33]–[35], and code morphing techniques [36]. However, re-keying is a less resource-intensive way to protect against side-channel attacks (SCAs) [30], [37], [38]. In this approach, the core ciphers are not used in their plain fashion, but they are used with a subkey generation function that uses the master key as input and generates session keys for data processing. The core cipher, a block cipher, for example, itself should be cryptographically strong. It should use the subkeys sparingly (once or a few times) and needs to get protected against Simple Power Analysis (SPA) only. On the other hand, the sub-key generation function needs to be heavily protected against SPA and stronger differential power

analysis (DPA) but does not have to be cryptographically strong. This concept is termed 'leveled implementation' as stated by Abdalla *et al.* [37] and Mennink in [30].

AE schemes are built on some underlying constructions or building blocks. Some of the most used building blocks are block ciphers. Famous block ciphers to create AE schemes include the AES [39], GIFT [40], and SKINNY [41]. Examples of Stream ciphers are in [42]. Permutation-based structures use either dedicated or keyless permutations as building-block primitive. Schemes in this class apply techniques like XOR, Encrypt XOR, Encrypt Mix Encrypt (EME), or variations of the Even-Mansour construction [43] instead of permutations in a sponge-like mode. The sponge construction is the most used form of keyless permutation. Several schemes use 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 [44]. There are also AE that are based on other building blocks like hash function/compression function (CF) like in [45], as there are schemes that use dedicated structures as their underlying structure like those in some other works [46], [47].

Besides the security-related characteristics, other essential attributes boost the performance and efficiency of AE schemes, like the following: Parallelizability, which indicates the ability of a scheme to process the i^{th} block independently of the j^{th} block [48]; Online, which indicates the ability of a scheme to compute the i^{th} ciphertext block having processed the first i plaintext blocks and does not need to know any plaintext beyond that block [49]; Inverse free: A scheme is inverse free if the underlying primitive does not require its inverse to perform encryption or decryption [12], [13]. Incrementality is the ability of a scheme to update parts altered only by the last activity, given a previous ciphertext–tag pair (C, T) [50]. Being single-pass renders a scheme more efficient and indicates that a scheme processes the plaintext only at once to achieve confidentiality and integrity [51], [52]. The lightweight property indicates whether a scheme is intended for use in resource-constrained environments [53], [54]. In the NIST-LW competition dedicated to lightweight AE, several schemes ideal for use with low resource devices were proposed, such as those in [46], [55]. Nonce-obliviousness is another desirable feature in situations where nonce mishandling can be expected.

B. CONTRIBUTIONS

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 [56] and nonce-hiding schemes [57]. 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 [58]. 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); that is, it provides the best security possible in the case where the nonce is repeated. Fifth, PSASPIN provides a security proof based on game-based theory and PRP/PRF switching lemma [25], [46], [59].

C. ORGANIZATION OF THIS WORK

We describe related work in Section II. In section III, we present a model of AE. Section IV is an introduction to the PSASPIN AE scheme and its features. The security analysis and proofs are given in Section V. We present the performance analysis in In Section VI. Section VII is the discussion part, and Section VIII concludes this work.

II. RELATED WORK

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 [30], [60]. Furthermore, adversaries traditionally took advantage of weaknesses in the cryptographic algorithms to breach security. SCAs cast doubt on the trustworthiness of this model [30]. These attacks acquire side-line information about cryptographic functions through passive attacks such as SPA [32], [61]–[63], DPA [32], [64], timing patterns [64], power consumption [36] or electromagnetic emissions [65]. In addition to the traditional types mentioned above, SCAs have been evolving, and recent works show more recent varieties based on techniques like deep learning [60], Artificial Neural networks [66], and thermal sensors [67].

SCAs take advantage of the relationship between the cryptographic algorithms and the patterns of radiations from the implemented devices. The main philosophy of these attacks is deducing the secret key from their relationships with the side channel signal behavior [32], [61]. SCAs are especially dangerous when cryptographic devices are placed where they can be physically accessible by adversaries. Several mechanisms have been proposed in the literature for protection against SCAs, including hiding [32] and masking [30], [33]–[35]. However, these countermeasures imply heavy performance penalties that are unbearable in resource-constrained environments such as IoT devices and smart cards. Fresh rekeying [37], [38] is a less resource-intensive way to obtain SCA protection than other mentioned ways. Furthermore, fresh rekeying provides protection against SCA by preventing the attackers from obtaining the intermediate

key materials by confining the use of every session key to once or few times [37], [38], [58], [68].

Nonces are used in AE schemes to prevent predictability and thus boost security as long as they are unique. The Ideal way of using nonces (or IVs) is to make sure that they are not repeated. Still, nonce repetition may occur accidentally, deliberately, or due to device malfunctioning, virtual machine cloning, or resetting source of generations [69], [70]. Rogaway and Shrimpton in [25] proposed the SIV model of operation. They came up with the idea of Nonce Misuse Resistance Authenticated Encryption (NMRAE) which provides the best security possible such that the authenticity remains protected. Privacy is protected only to the extent that some minimal information may be leaked, whether two plaintexts are equal and revealed if the message M, the header H, and the particular Nonce (IV) are repeated together, which can happen with negligible probability.

In addition to their misuse through repetition, how nonces are communicated stirred a hot debate within researchers. The gap between the theory and practice of using nonces and the concern of security breaches caused by wrongly handling them was first raised in 2019 by Bellare, NG, and Tackman [57]. They suggested a nonce hiding (NH) syntax for AE schemes and concretized it 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 (NH1 to NH5 transforms).

The sponge construction, first proposed by Bertone *et al.* in [71], is an iterated cryptographic primitive for building a function **f** with variable-length input and arbitrary output length based on a fixed-length transformation or permutation [71]. The sponge construction operates on a state of $\mathbf{b} = (\mathbf{r} + \mathbf{c})$ bits where **r** is called the bitrate and **c** is the capacity [71]. The sponge first absorbs its input block by block before processing and squeezing them out afterward. The sponge construction is the most used form of keyless permutation in AE. Sponges are also used for other cryptographic purposes like re-seedable pseudorandom generators and stream ciphers [44]. The sponge function is used in several modes based on the functionality required; for instance, the Duplex and its variant MonkeyDuplex [44], [72] modes are mainly used to implement online, single-pass AE schemes. The security of the Duplex construction can be proved to be equivalent to the Sponge construction through the Sponge/Duplex lemma in [71]. One of the winners in the CAESAR competition, namely Ascon [12], was based on Sponge construction, while five out of the ten finalists in the NIST lightweight competition were based on sponge construction, namely: Ascon [73], Elephant [74], ISAP [75], Photon-Beetle [76], Xoodyak [77].

Besides the security features, functional characteristics like parallelizability, incrementality, and being single-pass are indispensable for AE schemes because they contribute to the performance and efficiency of the schemes. Most of the sponge-based AE schemes are serial in nature since the original construction can not be parallelized at the algorithmic level. Still, several parallel AE sponge-based AE schemes have been proposed based on the Duplex construction. For instance, the AE schemes [78]–[80] provide various degrees of parallelizability and incrementality, but they are not protected against SCAs. Nonce obliviousness, proposed by Bellare *et al.* [57], obviates program designers from the burden of taking care of nonce transmission by modifying the traditional syntax of Nonce-Based Authenticated Encryption (NBAE) so that the nonce is integrated into the ciphertext and recovered at the destination. The nonce-hiding transforms were proposed and concretized for block ciphers in [57].

TABLE 1. Comparison of PSASPIN features to similar AE schemes (Y = Yes, N = No).

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

Cryptographic sponges offer a promising solution for protection against SCAs. Furthermore, the capacity parameter of the sponge construction helps withstand SCAs [56], [81]; however, the Sponge-based AE schemes that are incremental and single-pass do not provide protection mechanisms against SPA and DPA [79], [80]. On the other side, Sponge-based AE schemes that offer protection against SPA and DPA are neither parallelizable nor incremental. Examples of such schemes are [75], [81], [82]. In addition to that, as far as the authors know, there are no sponge-based AE schemes that are nonce-oblivious and NMR. Table 1 compares existing Sponge-based solutions to the proposed one (PSASPIN).

III. 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

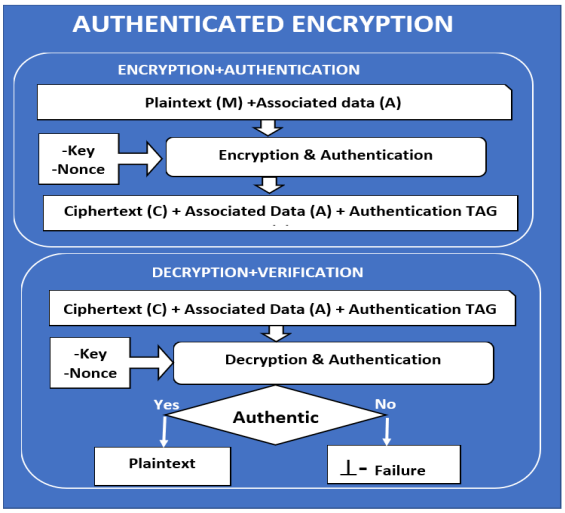

FIGURE 1. A schematic structure of an authenticated encryption (AE) scheme.

$-E : K \times N \times H \times P \rightarrow C|T$—along with a decryption $D : K \times N \times H \times C \rightarrow P\{\perp\}$. Separated AE with associated data also features a verification algorithm $V : K \times N \times H \times C \times T \rightarrow P\{\top, \perp\}c$. 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 [7], [8], [83]. Figure 1 illustrates a schematic structure of an AE scheme.

IV. PARALLEL SPONGE-BASED AE WITH SIDE-CHANNEL PROTECTION AND ADVERSARY INVISIBLE NONCES (PSASPIN)

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. It hides nonces from the adversary by taking a nonce as part of the input to the encryption but omitting it from the decryption. Nonces are encrypted, together 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.

PSASPIN has the following desirable properties important for AE schemes in terms of security, performance, and efficiency: parallelizable, incremental, single-pass, side-channel protected, and nonce-oblivious (preventing the adversary from viewing/ accessing the nonces).

A. 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.

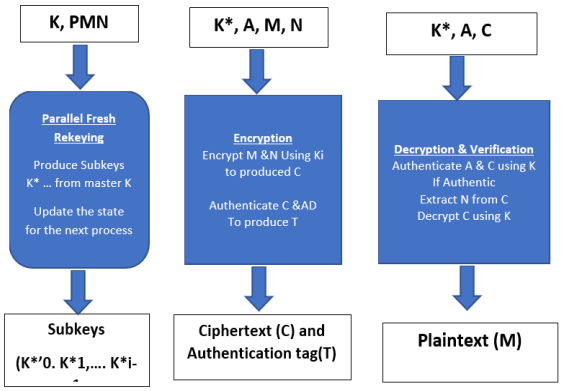

FIGURE 2. A high-level view of PSASPIN structure.

B. NOTATIONS

Here we introduce the notations used in this work. By **K, T, N, IV**, we denote the key, the authentication tag, the nonce, and the initialization vector, respectively. By **M, C, 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 $\mathbf{0}^k$ we mean an all 0-bit string of length k. By $|\mathbf{X}|$, we indicate the length of string X. By $\mathbf{X}||\mathbf{Y}$; we denote the string 'X' concatenated to the string 'Y.' By $\boldsymbol{X} \oplus \boldsymbol{Y}$ we denote the XOR of 'X' and 'Y' strings. By $\lceil \boldsymbol{X} \rceil^k$ We denote a bitstring X truncated to the most significant (last) k bits. By $\lfloor \boldsymbol{X} \rfloor_k$, we denote a bitstring X truncated to the least significant bit (first) k bits.

C. 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. A general scheme view is depicted in figure 2. The encryption and decryption process of PSASPIN AE are shown in figure 4 and figure 5, respectively.

D. 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.

1) 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 in Appendix A for details of FRK. After generating the session key, the shared state **S** is updated. The first Random IV is generated at the source and securely shared after that and will be incremented to keep the sides synchronized to the sponge permutation P. A counter (*Ctr)* is generated to keep track of the number of parallel threads incrementing by 1 with every lane.

2) 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. $A_0||A_1||\ldots\ldots..A_i||$, $|A| = r$. 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: $S \leftarrow P((S_r \oplus A_i)\,\|S_c)$. After processing the last A_i, a 1-bit domain separator is XORed with the state S: $S \leftarrow S \oplus (0^{319}\|1)$ 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 [73].

3) 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, $S_r \leftarrow S_r \oplus (N\|A_0\|M_0)$, 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←Sr⊕(N_1); S←P((Sr⊕N1)||Sc). The new nonce N1 is a ciphertext of the original nonce and will be XORed with the first block of the ciphertext in the next stage.

4) 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: $C_0 \leftarrow N_1\|C_{M0}$; S←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←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: $S_r \leftarrow S_r \oplus M_z$;$C_z \leftarrow \lfloor S_r \rfloor_{|M| mod\ r}$.

5) 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});$N_1\|C_{M0} \leftarrow C_0$; S←P(($S_r$⊕$N_1$)||$S_c$); $M_0 \leftarrow S_r \oplus C_{M0}$. 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), the state is updated; $M_i \leftarrow S_r \oplus C_i$; S←$C_{Mi}$||Sc; S←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.

6) 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: $C_1||\ldots||C_i||C_z||T$. See figures 4.4 and 4.5 for the schematic view of PSASPIN and algorithm later in this section for its processes.

7) 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 [38] and adopted by [37], [84], 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 [81]. 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 [56], [81], (2) protected block cipherslike AEAS and SERPENT [85], (3) Tweakable block ciphers(4) Algebraic construction-based Galois field (GF) multiplication, or (5) Traditional block ciphers with countermeasures like masking and hiding [68], [82], [86].

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

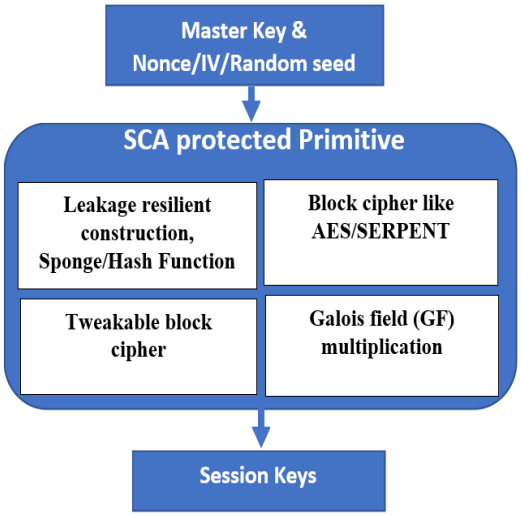

FIGURE 3. Options for implementation of rekeying function.

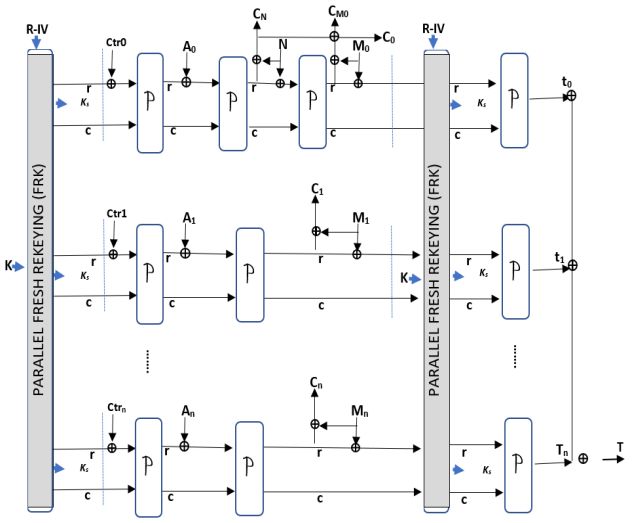

FIGURE 4. PSASPIN encryption.

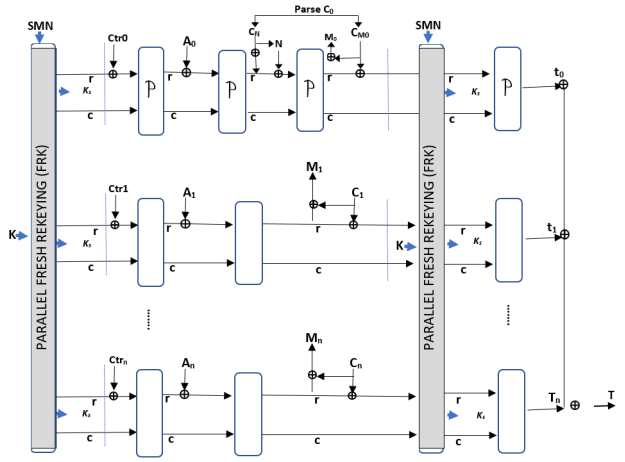

FIGURE 5. PSASPIN decryption.

multiplication field as in [84], [87], 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 1 in appendix A, 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 the preferred alternative in hardware implementations. See Algorithm 1 in appendix A for details of the $\boldsymbol{G}$ function based on the GF multiplication field using masking and shuffling combined to protect against higher-order DPA attacks.

V. SECURITY MODEL

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 generate session keys to protect the core scheme. 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 [32], 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 [30], [38]. 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 [38]. According to [58], parallel rekeying is necessary when parallel access to data is used.

Following [38], we define the pseudorandomness of a stateful generator: Let G = (K, N) be a stateful generator with a block length $\boldsymbol{k}$, let $\boldsymbol{n}$ be an integer, and Let $\mathbf{A}$ be an adversary.

Consider the following experiment:

$$\begin{aligned}
&\textit{Experiment } EXP^{prg-real}_{G,n,A}\\
&\quad \textit{for } i = 1, \ldots, n \textbf{ do}\\
&\quad (Out_i, St_i) \leftarrow N\,(St_{i-1})\,; \leftarrow s \| Out_i\\
&\quad g \leftarrow A(s)\\
&\quad \textit{return } g\\
&\textit{Experiment } EXP^{prg-rand}_{G,n,A}\\
&\quad s \leftarrow \{0,1\}^{n.k}\\
&\quad g \leftarrow A(s)\\
&\quad \textit{return } g
\end{aligned}$$

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 [38], 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 the real and random experiments in the following manner.

$$\begin{aligned}
ADV^{prg}_{G,n,A} &= \Pr\left[EXP^{prg-Real}_{G,n,A} = 1\right]\\
&\quad - \Pr\left[EXP^{prg-rand}_{G,n,A} = 1\right]\\
ADV^{prg}_{G,n,A}(t) &= max_A\left\{ADV^{prg}_{G,n,A}\right\}.
\end{aligned}$$

The maximum is over **A** with time complexity $\boldsymbol{t}$, and the time complexity is the execution time of the two experiments added to the size of the code of the adversary $\boldsymbol{A}$. 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 $\boldsymbol{n}$ bit string to $\boldsymbol{n}$ bit string under a uniform distribution, if $\boldsymbol{D}$ is a distribution that has an oracle access, then

$$\begin{aligned}
ADV^{prf}_{F,D} &= \Pr\left[D^{(F,K)} = 1 : K \underleftarrow{R} \{0,1\}^n\right] - \Pr\left[D^{f}(.)\right.\\
&\quad \left. = 1 : f \overleftarrow{\$} R^n\right]
\end{aligned}$$

Is the advantage of the distinguisher **D**, the advantage of **F** is: $ADV^{prf}_{F}(t,q) = Max_D\left\{ADV^{prf}_{F,D}\right\}$,

The maximum is over all **A**, the time complexity is $\boldsymbol{t}$, and making $\boldsymbol{q}$ oracle queries.

The following theorem shows how the pseudorandomness of 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: $ADV^{prg}_{G[F\},n}(t) \leq ADV^{prf}_{F}(t,n)$.

Proof: Let **A** be an adversary trying to compromise the pseudorandomness of G[F], and let $\boldsymbol{t}$ be its running time of $EXP^{prg-real}_{G[F],n,A}$ and $EXP^{prg-rand}_{G[F],\ n,A}$. We upper bound the advantage of **A** $ADV^{prg}_{G[F],n,A}$. We construct a distinguisher $\boldsymbol{D}$ for **F** and relate its advantage to **A**'s advantage. The distinguisher $\boldsymbol{D}$ has access to an oracle $\boldsymbol{O}$. It computes $s = \boldsymbol{O}(1)||\ldots..||\boldsymbol{O}(n)$ and outputs the same guess $\boldsymbol{A}$ on its input **s**. we could say that when the Oracle $\boldsymbol{O}$ is drawn at random from **F**, the probability that the distinguisher $\boldsymbol{D}$ return 1 equals the probability that $EXP^{prg-real}_{G[F],n,A}$ returns 1. On the other hand, the probability that $EXP^{prg-rand}_{G[F],n,A}$ return 1 equals that of $\boldsymbol{D}$ returning 1 when $\boldsymbol{O}$ is drawn randomly from the family of random functions R^n. As **D** runs in time at most $\boldsymbol{t}$ and makes at most $\boldsymbol{n}$ queries to its oracle, we obtain that: $ADV^{prg}_{G[F],n,A} \leq ADV^{prf}_{F}(t,n)$. If **A** is an arbitrary adversary and the maximum time of the two experiments is $\boldsymbol{t}$, that concludes the proof of the theorem.

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

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 $\boldsymbol{M}$, associated data $\boldsymbol{A}$, a PMN nonce $\boldsymbol{N}$, 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 [2], [18], [19] 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.* [88], Andreeva *et al.* [46], and Mihajloska *et al.* [89]

C. CONFIDENTIALITY (OR PRIVACY)

Confidentiality protects the secrecy of information in the case of the Chosen Plaintext Attack (IND-CPA) against eavesdropping adversaries and Chosen Ciphertext Attack (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. [2], [18], [19]

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

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

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 which means it can use the same nonces in calling E_k and \$. $ADV^{priv}_{II}(q_p, q_e, \lambda_e)$ 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 that they have not been tampered with during transit or while 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 is unable to produce ciphertext decryption of a message that the sender had never encrypted, and the latter ensures that the attacker is not able to create a ciphertext that the sender has not previously produced, whether the plaintext is new or not [2], [18].

Let us denote $\boldsymbol{P}$ as a set of underlying idealized permutations of AE scheme II. Then, we define integrity-related goals of AE as captured by the inability of adversary $\boldsymbol{A}$ to come up with a new plaintext that had not been produced by a valid decryption (D_k ($\boldsymbol{C}$) 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 $\boldsymbol{A}$, K, and P. The adversary succeeds in forging if $\boldsymbol{D}_k$ returns a message that is different from $\perp$ 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 $ADV_{II}^{auth}\left(q_p, q_E, \lambda_E, q_D, \lambda_D\right)$. We denote the maximum advantage taken over all adversaries that 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 $\boldsymbol{n}$ state values in the following manner:

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

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 $\Pi-$function evaluations via the encryption query is calculated as follows:

$$\sigma_E := \sum_{j=1}^{q_E} \sigma_{j,E} \leq q_E\,(c+f+4) = \lambda_E + 4q_E \tag{2}$$

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 [57], 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. The authors in [57]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 Nonce Misused Resistance (NMR) and nonce hiding, and we intend to concretize it for sponge-based AE schemes. In HN4 [57], A PRF F is applied to the triple (M, N, H), as in SIV[70], to produce a synthetic nonce N_1, which is sent as part of the Ciphertext C_2. As with SIV [70], the security of HN4 assumes tidiness [90]. For all K, N, C1, H if S1.DEC(K,N,C1,H) = M $\not=\perp$, 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: $ADV^{ae2}_{SE2} \leq ADV^{prf}_{F} + \frac{q_n q_v}{2^{SE1.nl}}$. 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 [21]–[24]. Rogaway and Shrimpton Proposed SIV construction in [25] as NMRAE that provides the best security possible if nonces are repeated. In SIV, the nonce is constructed by applying a Pseudorandom Function (PRF) to

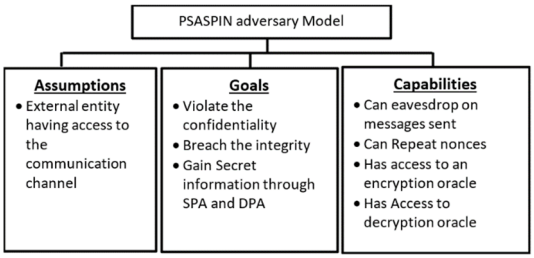

FIGURE 6. PSASPIN adversary model.

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 $\boldsymbol{n}$. Let $\tilde{\Pi} = SIV[F, \Pi]$. Let **A** be an adversary (for attaching ($\tilde{\Pi}$) with running time $\boldsymbol{t}$ and asking $\boldsymbol{q}$ queries with a total length of μ. Then there exists an adversary **B** and **D** such that:

$ADV^{priv\backslash\$}_{\Pi}(B) + ADV^{prf}_{F}(D) \geq ADV^{dae}_{\tilde{\Pi}}(A) - \frac{q}{2^n}$;

Furthermore, B and D run in time $\acute{\imath} = t + Time_{\Pi}(\mu) + c\mu$ for some absolute constant $\boldsymbol{c}$ and ask at most $\boldsymbol{q}$ queries with total length μ.

G. PSASPIN ADVERSARY MODEL

The adversary $\boldsymbol{A}$ 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 [91].

PSASPIN is secure as long as $\boldsymbol{A}$ with the defined assumptions, capabilities, and goals cannot violate its security with a non-negligible probability. It is worth noting that although $\boldsymbol{A}$ can use the same nonce to encrypt several messages, it cannot access the nonce (which is 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 [88], [46], [89], under the adversary model described in section 5. G.

1) PRIVACY OF PSASPIN

Theorem 1: Let $\Pi = (E, D)$ be a sponge-based AEAD based on an ideal permutation $\mathcal{P}$, then:

$$ADV^{priv}_{\Pi}(q_p, q_E, \lambda_E) \leq \frac{3(q_P + \sigma_E)^2}{2^{b+1}} + \left(\frac{8eq_P\sigma_E}{2^b}\right)^{\frac{1}{2}} + \frac{rq_P}{2^c} + \frac{q_P + \sigma_E}{2^k},$$

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 $\mathcal{P}$ do not collide. Because of the use of 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 $\mathcal{P}$ 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 [92], 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 $(p^{\pm}, \backslash\$)$ W hose challenge is to distinguish between these two views. Here, the advantage can be expressed as:

$$ADV^{priv}_{\Pi}(A) = \Delta_A\left(p^{\pm}, E_K; p^{\pm}, \backslash\$\right). \tag{3}$$

To make the analysis simpler, $p^{\pm}$ is replaced by a random function $\boldsymbol{f}$ following the PRP/PRF switch lemma [88], moving from $p^{\pm}$ to $\boldsymbol{f}^{\pm}$ 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 $\frac{(q_p+q_E 2)}{2^b} \leq \frac{(q_p+q_E)^2}{2^{b+1}}$ Applying the PRF/ PRF switch to both sides of the inequality:

$$\Delta_A\left(p^{\pm}, E_K; p^{\pm}, \backslash\$\right) \leq \Delta_A\left(f^{\pm}, E_K; f^{\pm}, \backslash\$\right) + \frac{(q_p + \sigma_E)^2}{2^b}. \tag{4}$$

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

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

$$\left(init.S_0 \begin{bmatrix} A_{S1,0} & M_{S1,0} & t_{s1,0} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ A_{Sn+1,c} & M_{Sn+1,f} & t_{sn+1,t} \end{bmatrix}\right). \tag{5}$$

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 $i \in \{1 \ldots q_{p\}, j,j' \in \{1 \ldots q_E,\ k\{1,\ldots,\sigma_{E,j}\},\ k' \in \left\{1,\ldots,\sigma_{E,j'}\right\}$:

$$\begin{aligned} \textbf{\textit{guess}}\ (i;j,k) &\equiv x_i = s_{j,k},\ \textit{hit}\ \left(j,k;j',k'\right) \\ &\equiv \textbf{\textit{parent}}\ \left(s_j,k\right) \neq \textbf{\textit{parent}}\ \left(s_{j'},k'\right) \wedge s_j, \\ &\quad k = s_{j'},k'. \end{aligned}$$

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

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

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

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

The exception is when a collision occurs and ***guess*** occurs; however, this event is already excluded by assuming $\neg$***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 ***guess*** $\vee$ ***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, $\neg$***event*** would have been triggered. Because $M_{Sj,0} = f\left(A_{Sj,c}\right) \oplus Ctr_0$, 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 $\boldsymbol{\mathcal{P}}$ 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

$$\Pr\left(A^{f^{\pm},E_K}\ \text{sets}\ \textbf{\textit{event}}\right) = \leq \frac{q_p q_E + \frac{\sigma_E^2}{2}}{2^b} + \left(\frac{8eq_p\sigma_E}{2^b}\right)^{\frac{1}{2}} + \frac{rq_p}{2^c} + \frac{q_p+\sigma_E}{2^k}. \quad (6)$$

Proof: Let **A** be an adversary interacting with oracles $(f^{\pm},\ E_K)$, and let $\Pr(\textbf{\textit{guess}} \vee \textbf{\textit{hit}})$ be the probability to be bounded. For $i \in \{1,\ldots,q_p\}$, 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 $j \in \{1,\ldots,q_E\}$ *and* $k \in \left\{1,\ldots,\sigma_{E,j}\right\}$, and considering a threshold $\rho \geq 1$, the following is defined:

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

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:

$$\mathbf{P\,r}\ (guess \vee hit) \leq \mathbf{P\,r}\left(guess \vee hit \,\middle|\, \begin{matrix} \neg\ (key \vee multi) \\ +\boldsymbol{P\,r}\ (key \vee multi) \end{matrix}\right) \quad (7)$$

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 ***guess*** $\vee$ ***hit***, under the assumptions that this query does not set (***key*** $\vee$ ***multi***) and also that (***guess*** $\vee$ ***hit*** $\vee$ ***key*** $\vee$ ***multi***) has not been triggered. For the analysis of (***key*** $\vee$ ***multi***) a similar method can be used.

Event Guess: this event can be triggered by the i^{th} permutation query (for $i = 1,\ldots,q_p$) or in evaluating any state value of the j^{th} construction query (for $j = 1,\ldots,q_E$). 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 ***key*** $\vee$ ***multi*** and also assumes that ***guess*** $\vee$ ***hit*** $\vee$ ***key*** $\vee$ ***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 ***guess***. Let $j_i \in \{1,\ldots,q_E\}$ for $i = 1,\ldots,\ q_p$ be the number of encryption queries before the i^{th} primitive query. Likewise, $e_j \in \left\{1,\ldots,q_p\right\}$ for $j = 1,\ldots,q_E$ 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 $i \in \left\{1,\ldots,q_p\right\}$, 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 ***guess*** for $j \in \{1, \ldots, q_E\}$. 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:

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

$$= f(A_{S_c}) \oplus \begin{pmatrix} Ctr_0 \\ \vdots \\ \vdots \\ Ctr_{n-1} \end{pmatrix},$$

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 $\left(\frac{\sigma_E - q_E}{2}\right)/2^b$. 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 $\binom{\sigma E}{2}\left(\frac{2}{2^r}\right)^\rho \leq \left(\frac{2e\sigma_E}{\rho 2^r}\right)^\rho$ 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, $\Pr(multi) \leq 2^r\left(\frac{2e\sigma_E}{\rho 2^r}\right)^\rho$.

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

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

Replacing $\rho = \max\{r, \left(\frac{2e\sigma_E 2^c}{q_p 2^r}\right)^{1/2}\}$ gives: $\Pr(guess \vee hit) \leq \frac{q_p\sigma_E + \frac{\sigma_E^2}{2}}{2^b} + 2\left(\frac{2eq_p\sigma_E}{2^b}\right)^{\frac{1}{2}} + \frac{rq_p}{2^c} + \frac{q_p+\sigma_E}{2^k}$, 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:

$$ADV_\Pi^{auth}(A) = \Pr\left(A^{p^{\pm}, E_K, D_K} forges\right) \leq \Pr\left(A^{f^{\pm}, E_K, D_K} forges\right) + \frac{(q_p + \sigma_E + \sigma_D)^2}{2^{b+1}}. \quad (8)$$

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 $i \in \{1, \ldots, q_p\}$, (D, j, k) be a decryption query index and (δ', j', k') be an encryption or decryption query index:

$$\begin{aligned} D_{guess}(i; j, k) &\equiv x_i = S_{D,j,k}, D_{hit}\left(j, k; \vartheta', j', k'\right) \\ &\equiv parent\left(S_{D,j,k}\right) \\ &\neq parent\left(S_{\vartheta',j',k'}\right) \wedge \left(S_{D,j,j}\right) = S_{\vartheta',j',k'}, \end{aligned}$$

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):

$$\begin{aligned} &\Pr\left(A^{f^{\pm}, E_K, D_K} forges\right) \\ &\quad \leq \Pr\left(A^{f^{\pm}, E_K, D_K} forges | \neg event\right) \\ &\qquad + \Pr\left(A^{f^{\pm}, E_K, D_K} sets\ event.\right) \end{aligned} \tag{9}$$

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 a 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:

$$\begin{aligned} &\Pr\left(A^{f^{\pm}, E_K, D_K} setseven\right) \\ &\quad \leq \frac{q_p \sigma_E + \frac{\sigma_E^2}{2}}{2^b} + \left(\frac{8_e q_p \sigma_E}{2^b}\right)^{\frac{1}{2}} \\ &\qquad + \frac{r q_p}{2^c} + \frac{q_p + \sigma_E + \sigma_D}{2^k} \\ &\qquad + \frac{\left(q_p + \sigma_E + \sigma_D\right)\sigma_D}{2^c} + \frac{\left(q_p + \sigma_E\right)\sigma_D + \frac{\sigma_D^2}{2}}{2^c}. \end{aligned}$$

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

$$\begin{aligned} &\Pr\left(guess \vee hit \vee D_{guess} \vee D_{hit}\right) \\ &\quad \leq \Pr\left(guess \vee hit \vee D_{guess} \vee D_{hit} | \neg\left(key \vee multi\right)\right) \\ &\qquad + \Pr\left(key \vee multi\right). \end{aligned} \tag{10}$$

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 $T /= T_{\vartheta,j}$. 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.

– $(N; A) = \left(N_{\vartheta,j}, A_{\vartheta,j}\right)$, but $C \neq C_{\vartheta,j}$. If the ciphertexts $C / = C_{\delta,j}$ 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, theoverline (ϑth decryption query triggers the event D_{hit} with probability $\sum_{k=1}^{\sigma_{D\vartheta}} \frac{\sigma_E + \sigma_{D,1} + \cdots + \sigma_{D\vartheta-1} + (k-1)}{2^c}$;

– 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.

– $(N) = \left(N_{\vartheta,j}\right)$, 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.

– $(N) /= \left(N_{\vartheta,j}\right)$: 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:
– $\Pr\left(D_{hit} | \neg(key \vee multi)\right) \leq$

$$\sum_{\frac{\sigma_D}{2^k}}^{q_D} \sum_{j=1}^{\sigma_{D\nabla j}} \sum_{k=1} \frac{\sigma_E + \sigma_{D,j} + \ldots + \sigma_{D\nabla j-1} + (k-1)}{2^c} + \frac{\sigma_D}{2^k} \leq \frac{\sigma_E \sigma_D \left(\frac{\sigma_D}{2}\right)}{2^c} +$$

Lemma 2 and (1) lead to:

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

This completes the proof.

VI. PERFORMANCE ANALYSIS

A. BACKGROUND

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 [73], [93]

The need for measuring the performance of a cryptographic algorithm usually stems from the requirement to compare several algorithms or the need to know how well a specific algorithm performs for applicability to specific use cases. Therefore, it is an essential deciding factor in including such algorithms in real-world protocols [93].

Several benchmarking frameworks for AE schemes exist in the literature, such as SUPERCORP [94] and BRUTUS [95]. In addition, Ankele & Ankele [93] proposed their framework for evaluating the software performance of 2^{nd} round candidates of the CAESAR competition. In this work, we follow the benchmarking schemes followed by ASCON developers [73] and Krovetz & Rogaway [96].

To evaluate its performance and compare it with similar schemes, PSASPIN was implemented in C language following the steps of Dobraunig *et al.* [73], despite the PSASPIN providing more features such as parallelizability protection against SPA and DPA, adversary invisible nonces, and Nonce Misuse Resistance (NMR).

B. 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,

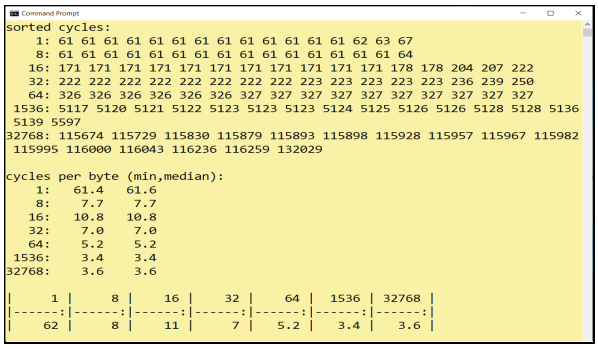

FIGURE 7. Measuring the performance of PSASPIN.

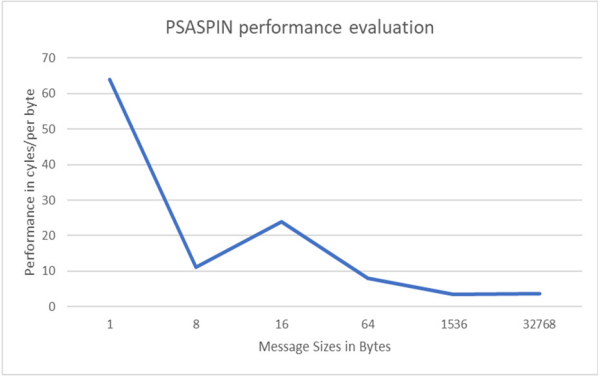

FIGURE 8. PSASPIN performance with different message sizes.

we enabled the GCC compiler flags: -O3 -march = native -Wall with the framework used by Dobraunig *et al.* [73].

C. THE RESULT

PSASPIN uses the inverse-free ASCON permutation [73], 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 2 and figure 7. 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 & Ankle in [93]. Figure 8 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 2 and figure 9 for a comparison of PSASPIN with other AE schemes.

Figure 7 shows that PSASPIN is suited for longer messages more than shorter messages, the penalty coming from calculations at the initial stages of the process. Table 2 and figure 8 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

TABLE 2. Comparing PSASPIN with other sponge-based AE schemes.

MESSAGE SIZE / SCHEME	1	8	16	32	64	1536	32768
NORX	1013	131	65	30	16.2	2.8	2.3
PI-CHIPHER	2055	250	123	63	36.9	8.5	7.3
ISAP	2614	337	173	93	52.9	14.4	12.9
ASCON	174	20	15	9	6.2	3.3	3.1
PSASPIN	61.4	11	24	8	5.4	3.5	3.6

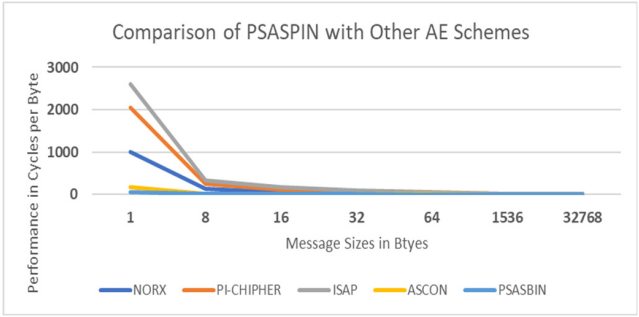

FIGURE 9. Comparison of PSASPIN and other sponge-bases AE schemes.

are sponge-based parallel AE schemes that do not protect against SPA/DPA. Finally, table 2 and figure 8 show that PSABBIN 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.

VII. DISCUSSION

Sponge-based cryptography appeared in 2011 with Bertoni *et al.* [71] took momentum after the sponge-based hash function Keccak was selected by NIST as the basis for SHA3 in October 2012 [97]. Block ciphers were the dominant construction for AE until the first sponge-based AE was proposed by [44], followed by other researchers, including [46], [79], [80], [98]. As with other underlying constructions of AE sponge-based backed by its modes of operations like Duplex and its variants monkeyDuplex, SpongeWrap, and DonkeySponge, [44], [72], 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 [32], [61], [99]. Moreover, protecting against SCAs is not trivial when devices with resource constraints like IoT and low memory smart cards are used as cryptographic devices [56], [100].

Sponge-based AE encryption schemes provide functional features that boost performance like parallelizability, incrementality [78], [79], single-pass, and online [73]. Morawiecki and Pieprzyk [80] proposed the first parallelizable AE scheme based on the duplex mode of the sponge construction, followed by [46], [78], [79]. 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 [32], [61]. 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 [57]. In this work, we aim to propose PSASPIN, a sponge-based AE that is protected against SPA and DPA and handles the nonces in a way invisible to the adversaries.

Regarding the protection against SCAs, the authors of ISAP [56], SALE [56], and SPOOK [82] proposed sponge-based AE schemes that are fortified against certain types of SCAs using different countermeasures.

For instance, AE schemes in [56], [81] employed sponge-based structures to protect against SPA and DPA, followed by [82], which used a construction based on a tweakable block cipher for the same purpose. In addition, these works used leveled implantation, first proposed by [32], [38]. 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 [32] and masking [30], [33]–[35], fresh rekeying is a cheaper way to achieve the same goal. First proposed by Abdalla & Bellare [38], 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) used to protect confidentiality and integrity. Rekeying increases key lifetime, meaning the number of times the same key can be used to process data before being replaced. Abdalla & Bellare [38], [85] 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 [101], [102].

Medwed *et al.* [37], [84] 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 *et al.* in [103] due to the weakness in key processing intermediate states of the block cipher as indicated by [104]. The authors used masking and shuffling to protect the rekeying part but applied them separately, so their scheme is protected against first-order DPA attacks. The proposed scheme is not vulnerable Chosen plaintext Attack (CPA) mentioned in [104] because it uses two different constructions. In addition, PSASPIN combines

masking and shuffling to protect against higher-order DPA attacks.

The authors of ISAP [56] and SALE [81] used a leveled implementation (although in different ways), where the rekeying part and the core rekeyed part use sponge-based constructs. According to [58], although using the same primitive for rekeying and data processing can reduce the code size, the possibility of mounting CPA attack may lead to compromise of the following keys. On the other side, the authors of Spook [82] 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, which is 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 [58].

Regarding the nonce-obliviousness feature, Bellare *et al.* [57] 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 (NH) 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. The authors in [57] concretized their proposal for the Block cipher-based AE schemes, but it is an open problem for the sponge-based scenarios. Our scheme concretizes the nonce-hiding syntax for the Sponge-based AE schemes.

This work is motivated by ISAP [56] 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 [58]. Third, our scheme provides security proof using game-playing theory based on PRP/PFR switching lemma [25], [46], [59] for the sponge-based data processing part based on the work in[38]. 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 GF(2^8) field as in [37], [84]. Dobraunig *et al.* [104] stated that the schemes that use Galois field multiplication like [37], [84] 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.

VIII. CONCLUSION

This work proposes a sponge-based nonce-oblivious, NMR, parallel, online and single-pass AE scheme protected against side-channel attacks. We used a leveled implementation approach where the key generation part used 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.

APPENDIX A

Algorithm 1: : PSASPIN Processes

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

Input: Key Kϵ 0,1^k, K$\leq$128,
Nonce N ϵ 0,1^{128},
Associated Data Aϵ0,1*,
Plaintext Mϵ0,1*
Output: Ciphertext Cϵ0,1$^{|M|}$,
Tag Tϵ0,1^{128}

Initialization
//Initialize a counter
Ctr=$Ctr = \lceil K \rceil_{64}$
Split A into A||1||* r-bit blocks of
$A_1 \ldots . A_n$
Split M into M||1||* r-bit blocks of
$M_1 \ldots . M_{n-1}$
//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
for i = 1, . . . ,x do
S$\leftarrow \mathcal{P}((S_r \oplus A_i)||S_c)$
S$\leftarrow$S$\oplus(0^{399}1)$
Processing Nonce
//in the first thread, process on block nonce N of 128 bits.
If Ctr=0
$S_r \leftarrow S_r \oplus (N\|A_0\|M_0)$
$N_1 \leftarrow S_r$
$S_r \leftarrow S_r \oplus (N_1)$
S$\leftarrow \mathcal{P}((S_r \oplus N_1)||S_c)$
Processing Plaintext
If Ctr=0
$S_r \leftarrow S_r \oplus M_0$
$C_{M0} \leftarrow S_r$
$C_0 \leftarrow N_1 \| C_{M0}$
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 $K \epsilon\ 0,1^k$, $K \leq 128$,

Nonce $N \epsilon\ 0,1^{128}$,

Associated Data $A \epsilon 0,1^*$,

Plaintext $M \epsilon 0,1^*$

Output: Ciphertext $C \epsilon 0,1^{|M|}$,

Tag $T \epsilon 0,1^{128}$

Initialization

//Initialize a counter

$Ctr=Ctr = \lceil K \rceil_{64}$

Split A into A||1||* r-bit blocks of

$A_1 \ldots . A_n$

Split M into M||1||* r-bit blocks of

$M_1 \ldots . M_{n-1}$

//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

$A_1 \ldots . A_x$

for i = 1, . . . ,x do

$S \leftarrow P((S_r \oplus A_i) \| S_c)$

$S \leftarrow S \oplus (0^{399}1)$

// No nonce input

Processing Ciphertext

Split C into C||1||* r-bit blocks of

$C_1 \ldots . C_{z-1}$

// parse C_0into $C_N$and C_{M0}

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

$S \leftarrow \mathcal{P}((S_r \oplus N_1) \| S_c)$

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

for I = 1,. . . ,z-1 do

$M_i \leftarrow S_r \oplus C_i$

$S \leftarrow C_{Mi} \| Sc$

$S \leftarrow P(S)$

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

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

Finalization

$K_s \leftarrow$FRK(K,IV)

$S \leftarrow P(S \oplus (0^r \| K_s \| 0^{c-r-k})$

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

//in the last thread

If $Ctr=n$

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

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 $k \# x - 1\ mod\ d$ ***do***

$k_b \leftarrow k$

for $i = 1$ *to* m ***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 ***do***

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

End for

End while

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

REFERENCES

[1] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Advances in Cryptology*. Berlin, Germany: Springer, 2000.

[2] J. Katz and M. Yung, "Unforgeable encryption and chosen ciphertext secure modes of operation," in *Fast Software Encryption*. Berlin, Germany: Springer, 2001.

[3] S. Riou. (Jan. 17, 2021). *DryGASCON, Lightweight Cryptography Standardization Process Round 1 Submission*. Submission to NIST 2019. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/drygascon-spec-round2.pdf

[4] M. Dworkin, "Recommendation for block cipher modes of operation—The CCM mode for authentication and confidentiality," NIST, Gaithersburg, MD, USA, Tech. Rep. 800-38C, 2007.

[5] C. J. A. Jansen and D. E. Boekee, "Modes of blockcipher algorithms and their protection against active eavesdropping," in *Advances in Cryptology*. Berlin, Germany: Springer, 1988.

[6] C. S. Jutla, "Encryption modes with almost free message integrity. in *Advances in Cryptology*. Berlin, Germany: Springer, 2001.

[7] V. D. Gligor and P. Donescu, "Fast encryption and authentication: XCBC encryption and XECB authentication modes," in *Fast Software Encryption*. Berlin, Germany: Springer, 2002.

[8] P. Rogaway, M. Bellare, and J. Black, "OCB: A block-cipher mode of operation for efficient authenticated encryption," in *Proc. 8th ACM Conf. Comput. Commun. Security.*, Philadelphia, PA, USA, 2001, pp. 196–205.

[9] K. Martin, *Everyday Cryptography*. Oxford, U.K.: Oxford Press, 2012.

[10] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and Analysis of the generic composition paradigm," *J. Cryptol.*, vol. 21, no. 4, pp. 469–491, Oct. 2008.

[11] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, K. Yasuda, N. Datta, and M. Nandi. (2016). *COLM v1. Submission to CAESAR R3*. (Mar. 2, 2021). [Online]. Available: https://competitions.cr.yp.to/round2/colm.pdf

[12] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer. (2016). *ASCON v2. Submission to CAESAR R3*. (Mar. 2, 2021). [Online]. Available: http://competitions.cr.yp.to/round1/asconv1.pdf
[13] J. Jean, I. Nikolić, and T. Peyrin. (2016). *Deoxys v1.41. Submission to CAESAR R3*. (Jan. 17, 2021). [Online]. Available: http://competitions.cr.yp.to/round1/deoxysv1.pdf
[14] T. Krovetz and P. Rogaway. (2016). *OCB (v1.1). Submission to CAESAR R3*. (Jan. 14, 2021). [Online]. Available: http://competitions.cr.yp.to/round1/ocbv1.pdf
[15] H. Wu, and B. Preneel. (2016). *AEGIS—A Fast Authenticated Encryption Algorithm (v1.1). Submission to CAESAR R3*. (Jan. 14, 2021). [Online]. Available: http://competitions.cr.yp.to/round1/aegisv1.pdf.
[16] H. Wu. (2016). *ACORN—A Lightweight Authenticated Cipher (v3). Submission to CAESAR R3*. (Feb. 2, 2021). [Online]. Available: http://competitions.cr.yp.to/round1/acornv1.pdf
[17] M. A. Jimale, M. R. Z'aba, M. L. B. M. Kiah, M. Y. I. Idris, N. Jamil, M. S. Mohamad, and M. S. Rohmad, "Authenticated encryption schemes: A systematic review," *IEEE Access*, vol. 10, pp. 14739–14766, 2022.
[18] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Advances in Cryptology*. Berlin, Germany: Springer, 2000.
[19] P. Rogaway, "Authenticated-encryption with associated-data," in *Proc. 9th ACM Conf. Comput. Commun. Secur. (CCS)*, Washington, DC, USA, 2002, pp. 98–107.
[20] S. Gueron and Y. Lindell, "GCM-SIV: Full nonce misuse-resistant authenticated encryption at under one cycle per byte," *22nd ACM Conf. Comput. Commun. Secur.*, Denver, CO, USA: Denver Marriot City Center, 2015, pp. 12–16.
[21] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile communications: The insecurity of 802.11," in *Proc. 7th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, Rome, Italy, 2001, pp. 180–189.
[22] T. Kohno, "Attacking and repairing the winZip encryption scheme," in *Proc. 11th ACM Conf. Comput. Commun. Secur. (CCS)*, Washington DC, USA, 2004, pp. 72–81.
[23] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dallas, TX, USA, Oct. 2017, pp. 1313–1328.
[24] H. Wu, "The misuse of RC4 in Microsoft word and excel," Cryptology ePrint, Tech. Rep. 2005/007, 2005. [Online]. Available: https://eprint.iacr.org/2015/102.pdf
[25] P. Rogaway and T. Shrimpton, "Deterministic authenticated-encryption: A provable-security treatment of the key-wrap problem," Cryptology ePrint Archive, Dept. Comput. Sci., Univ. California, Davis, CA 95616, USA, Tech. Rep. 2006/221, 2006. [Online]. Available: https://www.iacr.org/archive/eurocrypt2006/40040377/40040377.pdf
[26] P. Rogaway, *Nonce-Based Symmetric Encryption*. Berlin, Germany: Springer, 2004.
[27] P. Rogaway, M. Bellare, and J. Black, "OCB: A block-cipher mode of operation for efficient authenticated encryption," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 3, pp. 365–403, Aug. 2003.
[28] M. Bellare, R. Ng, and B. Tackmann, "Nonces are noticed: AEAD revisited," Cryptology ePrint Archive, IACR-CRYPTO-2019, Santa Barbara, CA, USA, Tech. Rep. 2019/624, Aug. 2019. [Online]. Available: https://eprint.iacr.org/2019/624
[29] C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer. (2019). *ISAP v2.0. Submission to the NIST LWC Competition R2*. (Mar. 10, 2021). [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/isap-spec-round2.pdf
[30] B. Mennink, "Beyond birthday bound secure fresh rekeying: Application to authenticated encryption," in *Proc. ASIACRYPT*, Dec. 2020. [Online]. Available: https://eprint.iacr.org/2020/1082.pdf
[31] S. Picek, A. Heuser, A. Jovic, S. A. Ludwig, S. Guilley, D. Jakobovic, and N. Mentens, "Side-channel analysis and machine learning: A practical perspective," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 4095–4102.
[32] S. Mangard and E. T. O. Popp, "Power analysis attacks: Revealing the secrets of smart cards," in *Power Analysis Attacks: Revealing the Secrets of Smart Card*. Boston, MA, USA: Springer, 2007.
[33] A. Duc, S. Faust, and F.-X. Standaert, "Making masking security proofs concrete," in *Advances in Cryptology*. Berlin, Germany: Springer, 2015.
[34] E. Prouff and M. Rivain, "Masking against side-channel attacks: A formal security proof," in *Advances in Cryptology*. Berlin, Germany: Springer, 2013.
[35] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology*. Berlin, Germany: Springer, 2003.
[36] F. Antognazza, A. Barenghi, and G. Pelosi, "Metis: An integrated morphing engine CPU to protect against side channel attacks," *IEEE Access*, vol. 9, pp. 69210–69225, 2021.
[37] M. Medwed, F.-X. Standaert, J. Großschädl, and F. Regazzoni, "Fresh re-keying: Security against side-channel and fault attacks for low-cost devices," in *Progress in Cryptology*. Berlin, Germany: Springer, 2010.
[38] M. Abdalla and M. Bellare, "Increasing the lifetime of a key: A comparative analysis of the security of re-keying techniques," in *Advances in Cryptology*. Berlin, Germany: Springer, 2000.
[39] J. Daemen and V. Rijmen, "The design of Rijndael," in *The Advanced Encryption Standard*. Springer-Verlag, Berlin, Germany, 2002.
[40] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT: A small present," in *Cryptographic Hardware and Embedded Systems*. Cham, Switzerland: Springer, 2017.
[41] C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. M. Sim, "The *SKINNY* family of block ciphers and its low-latency variant *MANTIS*," in *Advances in Cryptology*. Berlin, Germany: Springer, 2016.
[42] R. Tahir, M. Y. Javed, and A. R. Cheema, "Rabbit-MAC: Lightweight authenticated encryption in wireless sensor networks," in *Proc. Int. Conf. Inf. Autom.*, Jun. 2008, pp. 573–577.
[43] J. Alizadeh, M. R. Aref, and N. Bagheri, "JHAE: A novel permutation-based authenticated encryption mode based on the hash mode JH," Tech. Rep. 2014/193, Cryptology ePrint Archive, 2014. [Online]. Available: https://eprint.iacr.org/2014/193
[44] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "Duplexing the sponge: Single-pass authenticated encryption and other applications," SAC Taichung, Taiwan, Taipei City, Tech. Rep. 2011/499, 2011. [Online]. Available: https://eprint.iacr.org/2011/499.pdf
[45] S. Cogliani, D.-Ş. Maimu, D. Naccache, R. P. D. Canto, R. Reyhanitabar, S. Vaudenay, and D. Vizár. (2014). *Offset Merkle-Damgård (OMD) Version 1.0. Submission to CAESAR R1*. (Aug. 12, 2020). [Online]. Available: http://competitions.cr.yp.to/round1/omdv10.pdf
[46] E. Andreeva, B. Bilgin, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha, and K. Yasuda, "APE: Authenticated permutation-based encryption for lightweight cryptography," in *Proc. FSE*, London, U.K., Mar. 2014. [Online]. Available: https://eprint.iacr.org/2013/791.pdf
[47] X. Lu, Z. Ma, and D.-G. Feng, "A quantum authenticated encryption scheme," in *Proc. 7th Int. Conf. Signal Process. (ICSP)*, Aug. 2004, pp. 2306–2309.
[48] T. Iwata, K. Minematsu, J. Guo, S. Morioka, and E. Kobayashi. (2016). *CLOC and SILC. Submission to CAESAR R3*. (Mar. 2, 2021). [Online]. Available: http://competitions.cr.yp.to/round3/clocsilcv3.pdf
[49] M. Bellare, A. Boldyreva, L. Knudsen, and C. Namprempre, "Online ciphers and the hash-CBC construction," in *Advances in Cryptology*. Berlin, Germany: Springer, 2001.
[50] Y. Sasaki and K. Yasuda, "A new mode of operation for incremental authenticated encryption with associated data," in *Selected Areas in Cryptography*. Cham, Switzerland: Springer, 2016.
[51] A. Boorghany, S. Bayat-Sarmadi, and R. Jalili, "Efficient lattice-based authenticated encryption: A practice-oriented provable security approach," Cryptology ePrint Archive, Tech. Rep. 2016/268, 2016. [Online]. Available: https://eprint.iacr.org/2016/268.pdf
[52] R. Reyhanitabar, S. Vaudenay, and D. Vizár, "Authenticated encryption with variable stretch," in *Advances in Cryptology*. Berlin, Germany: Springer, 2016.
[53] A. Chakraborti, A. Chattopadhyay, M. Hassan, and M. Nandi, "TriviA: A fast and secure authenticated encryption scheme," in *Proc. CHES*, Saint Malo, France, Sep. 2015. [Online]. Available: https://eprint.iacr.org/2015/590.pdf
[54] M. Agrawal, D. Chang, and S. Sanadhya, "A new authenticated encryption technique for handling long ciphertexts in memory constrained devices," in *Proc. ACISP*, Brisbane, Australia, 2015. [Online]. Available: https://eprint.iacr.org/2015/331.pdf
[55] D. Engels, M.-J. O. Saarinen, P. Schweitzer, and E. M. Smith, "The hummingbird-2 lightweight authenticated encryption algorithm," in *Proc. 7th Int. Workshop, RFIDSec*, Amherst, USA, Jun. 2011. [Online]. Available: https://eprint.iacr.org/2011/126.pdf
[56] C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer. (2019). *ISAP v2.0. Submission to the NIST LWC Competition R2*. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/isap-spec-round2.pdf

[57] M. Bellare, R. Ng, and B. Tackmann, *Nonces Are Noticed: AEAD Revisited*. Cham, Switzerland: Springer, 2019.
[58] *Re-keying Mechanisms for Symmetric Keys*. Wilmington, DE, USA: IETF Administration LLC N West Street, Suite, 2019.
[59] P. Jovanovic, A. Luykx, and B. Mennink, "Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes," in *Advances in Cryptology*. Berlin, Germany: Springer, 2014.
[60] D. Kwon, H. Kim, and S. Hong, "Non-profiled deep learning-based side-channel preprocessing with autoencoders," *IEEE Access*, vol. 9, pp. 57692–57703, 2021.
[61] T. Popp, "An introduction to implementation attacks and countermeasures," in *Proc. 7th IEEE/ACM Int. Conf. Formal Methods Models Co-Design*, Cambridge, MA, USA, Jul. 2009, pp. 108–115.
[62] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Advances in Cryptology*. Berlin, Germany: Springer, 1999.
[63] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology*. Berlin, Germany: Springer, 1999.
[64] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology*. Berlin, Germany: Springer, 1996.
[65] M. G. Kuhn and R. J. Anderson, "Soft tempest: Hidden data transmission using electromagnetic emanations," *Information Hiding*. Berlin, Germany: Springer, 1998.
[66] N. Mukhtar, A. P. Fournaris, T. M. Khan, C. Dimopoulos, and Y. Kong, "Improved hybrid approach for side-channel analysis using efficient convolutional neural network and dimensionality reduction," *IEEE Access*, vol. 8, pp. 184298–184311, 2020.
[67] T. Kim and Y. Shin, "ThermalBleed: A practical thermal side-channel attack," *IEEE Access*, vol. 10, pp. 25718–25731, 2022.
[68] J. Krämer and P. Struck, "Leakage-resilient authenticated encryption from leakage-resilient pseudorandom functions," Cryptology ePrint, Tech. Rep. 2020/280, 2020. [Online]. Available: https://eprint.iacr.org/2020/280.pdf
[69] A. Belenko, "Apple SSL/TLS bug (CVE-2014–1266)," NowSecure, Tech. Rep., 2014. [Online]. Available: https://www.nowsecure.com/blog/2014/02/23/apple-ssl-tls-bug-cve-2014-1266/
[70] S. Gueron and Y. Lindell, "GCM-SIV: Full nonce misuse-resistant authenticated encryption at under one cycle per byte," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Denver, CO, USA, Oct. 2015, pp. 109–119.
[71] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. (2011). *Cryptographic Sponge Functions*. [Online]. Available: https://keccak.team/files/CSF-0.1.pdf
[72] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. (2012). *Permutation-Based Encryption, Authentication and Authenticated Encryption*. (Apr. 29, 2021). [Online]. Available: https://keccak.team/files/KeccakDIAC2012.pdf
[73] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "ASCON v1.2. submission to the NIST LWC competition R2," National Institute standards Technology (NIST), Tech. Rep. NISTIR 8369, 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8369.pdf
[74] T. Beyne, Y. L. Chen, C. Dobraunig, and B. Mennink. (2019). *Elephant v1.1. Submission to the NIST LWC Competition R2*. (Mar. 10, 2021). [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/elephant-spec-round2.pdf
[75] C. Dobraunig,M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer. (2019). *ISAP v2.0. Submission to the NIST LWC Competition R2*. (Mar. 10, 2021). [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/isap-spec-round2.pdf
[76] Z. Bao, A. Chakraborti, N. Datta, J. Guo, M. Nandi, T. Peyrin, and K. Yasuda. (2019). *PHOTON-Beetle Authenticated Encryption and Hash Family. Submission to the NIST LWC Competition R2*. (Mar. 10, 2021). [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/photon-beetle-spec-round2.pdf
[77] J. Daemen, S. Hoffert, M. Peeters, G. V. Assche, and R. V. Keer. (2019). *Xoodyak, a Lightweight Cryptographic Scheme. Submission to the NIST LWC Competition R2*. (Mar. 10, 2021).[Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Xoodyak-spec-round2.pdf.
[78] J.-P. Aumasson, P. Jovanovic, and S. Neves. (2016). *NORX v3. Submission to CAESAR R3*. (Mar. 2, 2021). [Online]. Available: http://competitions.cr.yp.to/round1/norxv1.pdf
[79] D. Gligoroski, H. Mihajloska, S. Samardjiska, H. Jacobsen, M. El-Hadedy, and R. E. Jensen. (2014). *Π-Cipher v11. Submission to CAESAR R1*. (Dec. 8, 2020). [Online]. Available: http://competitions.cr.yp.to/round1/picipherv1.pdf
[80] P. Morawiecki and J. Pieprzyk, "Parallel authenticated encryption with the duplex construction," Cryptology ePrint Archive, Tech. Rep. 2013/658, 2013. [Online]. Available: https://eprint.iacr.org/2013/658.pdf
[81] J. P. Degabriele, C. Janson, and P. Struck, "Sponges resist leakage: The case of authenticated encryption," in *Proc. ACR-ASIACRYPT*, Kobe, Japan, Dec. 2019. [Online]. Available: https://eprint.iacr.org/2019/1034.pdf
[82] D. Bellizia, F. Berti, O. Bronchain, G. Cassiers, S. Duval, C. Guo, G. Leander, G. Leurent, I. Levi, C. Momin, O. Pereira, T. Peters, F.-X. Standaert, and F. Wiemer. (2019). *Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher. Submission to the NIST LWC Competition R2* [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Spook-spec-round2.pdf
[83] D. Whiting, R. Housley, and N. Ferguson, "Counter with CBC-MAC (CCM)," RFC Editor, Tech. Rep., 2003.
[84] M. Medwed, C. Petit, F. Regazzoni, M. Renauld, and F.-X. Standaert, "Fresh re-keying II: Securing multiple parties against side-channel and fault attacks," in *Proc. CARDIS*. Berlin, Germany: Springer-Verlag, 2011.
[85] O. Pereira, F.-X. Standaert, and S. Vivek, "Leakage-resilient authentication and encryption from symmetric cryptographic primitives," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Denver, CO, USA, Oct. 2015, pp. 96–108.
[86] G. Barwell, D. P. Martin, E. Oswald, and M. Stam, "Authenticated encryption in the face of protocol and side channel leakage," in *Proc. IACR-ASIACRYPT*, Hong Kong, Dec. 2017. [Online]. Available: https://eprint.iacr.org/2017/068.pdf
[87] M. Abdalla, S. Belaïd, and P.-A. Fouque, "Leakage-resilient symmetric encryption via re-keying," in *Cryptographic Hardware and Embedded Systems*. Berlin, Germany: Springer, 2013.
[88] P. Jovanovic, A. Luykx, and B. Mennink, "Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes," in *Proc. IACR-ASIACRYPT*, Kaoshiung, Taiwan, Dec. 2014. [Online]. Available: https://eprint.iacr.org/2014/373.pdf
[89] H. Mihajloska, B. Mennink, and D. Gligoroski. (2016). *Π-Cipher with Intermediate Tags*. (Jan. 9, 2021). [Online]. Available: http://pi-cipher.org/upload/Pi-Cipher%20with%20intermediate%20tags.pdf
[90] C. Namprempre, P. Rogaway, and T. Shrimpton, "Reconsidering generic composition," in *Advances in Cryptology*. Berlin, Germany: Springer, 2014.
[91] Q. Do, B. Martini, and K.-K. R. Choo, "The role of the adversary model in applied security research," *Comput. Secur.*, vol. 81, pp. 156–181, Mar. 2019.
[92] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Sponge-based pseudo-random number generators," in *Cryptographic Hardware and Embedded Systems*. Berlin, Germany: Springer, 2010.
[93] R. Ankele and R. Ankele, "Software benchmarking of the 2nd round CAESAR candidates," Cryptology ePrint, Tech. Rep. 2016/740, 2016. [Online]. Available: https://eprint.iacr.org/2016/740
[94] D. J. Bernstein. (2016). *Supercop. eBACS: ECRYPT Benchmarking of Cryptographic Systems*. (Jan. 26, 2022). [Online]. Available: https://bench.cr.yp.to/supercop.html
[95] M.-J. O. Saarinen, "The BRUTUS automatic cryptanalytic framework: Testing CAESAR authenticated encryption candidates for weaknesses," *J. Cryptograph. Eng.*, 2014. [Online]. Available: https://eprint.iacr.org/2014/850.pdf, doi: 10.1007/s13389-015-0114-1.
[96] T. Krovetz and P. Rogaway, "The software performance of authenticated-encryption modes," in *Fast Software Encryption*. Berlin, Germany: Springer, 2011.
[97] (Sep. 21, 2021). *SHA Zoo*. [Online]. Available: https://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo
[98] M.-J. O. Saarinen, "CBEAM: Efficient authenticated encryption from feebly one-way ϕ functions," in *Proc. CT-RSA Conf.*, San Francisco, CA, USA, Feb. 2014. [Online]. Available: https://eprint.iacr.org/2013/773.pdf
[99] S. Dziembowski and K. Pietrzak, "Leakage-resilient cryptography," in *Proc. 49th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2008, pp. 293–302.
[100] C. Guo, O. Pereira, T. Peters, and F. C.-X. Standaert, "Towards low-energy leakage-resistant authenticated encryption from the duplex sponge construction," Cryptology ePrint Archive, Tech. Rep. 2019/193, 2019. [Online]. Available: https://eprint.iacr.org/2019/193.pdf

[101] A. Battistello, J.-S. Coron, E. Prouff, and R. Zeitoun, "Horizontal side-channel attacks and countermeasures on the ISW masking scheme," in *Cryptographic Hardware and Embedded Systems*. Berlin, Germany: Springer, 2016.
[102] C. D. Walter, "Sliding windows succumbs to big mac attack," in *Cryptographic Hardware and Embedded Systems*. Berlin, Germany: Springer, 2001.
[103] J. Black, P. Rogaway, T. Shrimpton, and M. Stam, "An analysis of the blockcipher-based hash functions from PGV," *J. Cryptol.*, vol. 23, no. 4, pp. 519–545, Oct. 2010.
[104] C. Dobraunig, F. Koeune, S. Mangard, F. Mendel, and F.-X. Standaert, "Towards fresh and hybrid re-keying schemes with beyond birthday security," in *Smart Card Research and Advanced Applications*. Cham, Switzerland: Springer, 2016.

MOHAMUD AHMED JIMALE received the bachelor's degree in information technology from SIMAD University, Mogadishu, Somalia, the master's degree in computer science from the University of Malaya, Malaysia, where he is currently pursuing the Ph.D. degree. His research interests include cryptography, blockchain, cloud computing, and the Internet of Everything (IoE). He held different academic and administrative positions, including the founding present of the Jamhuriya University of Science and Technology (JUST) and the Deputy Head of the Information Technology Department, Dahabshil International Company, Somalia.

MUHAMMAD REZA Z'ABA received the Bachelor of Science (computer) degree from Universiti Teknologi Malaysia (UTM), in 2004, and the Ph.D. degree from the Queensland University of Technology, Australia, in 2010.

He is currently a Senior Lecturer at the Department of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya. He was previously a Researcher at MIMOS Berhad (from 2010 until February 2018), which is a research arm under the purview of the Ministry of Science, Technology and Innovation Malaysia. His main research interests include symmetric cryptography (block ciphers and hash functions) and blockchain-related technologies, including digital currencies and other areas of information security.

MISS LAIHA BINTI MAT KIAH (Senior Member, IEEE) received the Ph.D. degree in information security from Royal Holloway, University of London, U.K., in 2007. Since then, she is an active Researcher at the Faculty of Computer Science and Information Technology, UM, in computer science field particularly in security. She was promoted to professorship, in 2015. Her main research interest includes security aspect of computing and technology fields with variation of applications in multi and/or trans disciplinary projects. This is evidenced by her publications and research projects in which she is/was the principal investigator (PI) as well as co-PIs. As a professional technologist (Ts.), keeping up with the current trend and demand of ever evolving computing technology field is crucial to ensure the quality and the impact of her research work. Her current research interests include cyber security, blockchain technology, the IoT, and health information exchange. She is an active member of EC Council, the Malaysian Society for Cryptology Research (MSCR), and the Malaysia Board of Technologists (Ts.).

MOHD YAMANI IDNA IDRIS (Member, IEEE) received the B.E. degree in electrical engineering, the M.Sc. degree in computer science, and the Ph.D. degree in electrical engineering from the Universiti Malaya, Kuala Lumpur, Malaysia. He is currently an Associate Professor with the Department of Computer System and Technology, Faculty of Computer Science and Information Technology, Universiti Malaya. He has published many articles in reputable journals and has received many awards for his inventions. His research interests include the Internet of Things (IoT), information security, embedded systems, image processing and computer vision, and wireless sensor networks.

NORZIANA JAMIL received the Ph.D. degree in security in computing, in 2013. She is currently an Associate Professor at the Universiti Tenaga Nasional, Malaysia. Her research interests include cryptography, security for cyber-physical systems, security analytics, and intelligent systems. She is an alumni of leadership in innovation fellowship by U.K. Royal Academy of Engineering, a Project Leader of various cryptography and cyber security related research and consultancy projects, has been actively involving in advisory for cryptography and cyber security projects, and works with several international prominent researchers and professors.

MOESFA SOEHEILA MOHAMAD received the Bachelor of Arts degree in mathematics and computation from Oxford University, in 1997, and the Master of Science degree in mathematics for cryptography and communications from the Royal Holloway College, University of London, in 2011. She is currently pursuing the Ph.D. degree in IT with Multimedia University, Cyberjaya, Malaysia. She has been a Break Researcher at MIMOS Berhad, since 1997.

MOHD SAUFY ROHMAD received the bachelor's degree in information technology from the Universiti Teknologi PETRONAS and the Master of Science degree in computer science from UiTM, where he is currently pursuing the Ph.D. degree in embedded cryptography. He is currently the Head of robotic, the IoT, and big data with the Smart Manufacturing Research Institute, UiTM Shah Alam, where he is also a full-time Senior Lecturer with the College of Engineering. He was previously a Researcher at Telekom Malaysia Research and Development and MIMOS Berhad. He was a SoC Design Engineering at Intel, Penang. He also involved in a few Industrial IoT projects for livestock and agriculture.