---

"Digital Object Identifier 10.1109/ACCESS.2017.Doi Number"

# Sponge-based Parallel Authenticated Encryption with Variable Tag length and Side-Channel Protection

**Mohamud Ahmed Jimale[1], Nor Aniza Abdullah[1], Miss Laiha Mat Kiah[1], Mohd Yamani Idna Idris[1], Muhammad Reza Z'aba[2], Norziana Jamil[3], Mohd Saufy Rohmad[4]**

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

"Corresponding author: Mohamud Ahmed Jimale (e-mail: mahamudjimale@gmail.com)

This work was supported by the Fundamental Research Grant Scheme (FRGS) of the Ministry of Higher Education, Malaysia, Project Number FP072-2019A, Reference Code FRGS/1/2019/ICT05/UM/02/1"

**ABSTRACT** Authenticated Encryption (AE) protects confidentiality and integrity at the same time. The sponge construction is based on an iterated permutation or transformation that can be used to implement hashing, and AE schemes, among others. Sponge-based AE schemes offer desirable characteristics like parallelizability and incrementality. In addition, they provide security features such as protection against Chosen Plaintext Attacks, Chosen-Ciphertext Attacks, and Side-Channel Attacks (SCAs). Traditionally AE schemes assume the tag length, also called the stretch, as a fixed parameter per key, and the security is proved according to that assumption. However, the variable tag length per key could happen due to misconfiguration or misuse. In that case, the security would be violated, so it is vital to accommodate variable tag length without sacrificing other desirable features. Reyhanitabar et al. proposed Key Equivalent Separation by Stretch feature and concretized it for protection against tag length misuse attacks in block cipher-based AE schemes. However, the problem remains unresolved for sponge-based constructions, where current sponge-based schemes are vulnerable to tag length variation under the same key attacks. This work aims to bridge this gap by proposing a parallel, sponge-based AE scheme with a variable tag length per key that protects against SCAs and suggesting a lower bound for the recommended tag length. Finally, the security of the proposed scheme is discussed, and its performance is analyzed after implementing the proposed AE scheme in the C programming language.

**INDEX TERMS** Authenticated Encryption, Integrity, Message Authentication Code, Nonce-based AE, Parallel AE, Privacy, Side-channel attacks, Sponge-based AE, Tag length, Variable stretch.

## I. INTRODUCTION

### A. BACKGROUND

Secure communication requires Authenticated Encryption (AE) since it guarantees the privacy and authenticity of messages. Simple encryption-only primitives, like block or stream ciphers, protect the confidentiality of messages from being seen by unauthorized parties. However, such primitives, if used on their own without authenticity protection, are insecure about being employed in communications because an active attacker can alter the ciphertext without being noticed. For instance, an eavesdropper of a banking transaction (without authenticity protection) could change bits of the messages (with the help of some prior knowledge or guessing) without even needing to read it and forward it to its source. Still, it could be accepted as a valid transaction. A cryptosystem with such property is termed malleable. On this front, [1] and [2] proposed the first authenticated encryption (AE) schemes, combining individual encryption schemes and message authentication code (MAC) schemes to pave for a new paradigm of protecting privacy and integrity simultaneously. The AE with associated data (AEAD) allows the addition of unencrypted but authenticated pieces of data, such as those used for packet routing [3-5].

AE safeguards confidentiality and integrity by following two different security models. Confidentiality defends data privacy against passive adversaries in the chosen-plaintext attack (IND-CPA) and active attackers in the chosen-ciphertext attack (IND-CCA) [2, 6, 7]. Integrity ensures that communications are authentic and haven't been tampered with, whether in motion or at rest. In addition, AE protects the authenticity of plaintext with the INT-PTXT model and that of ciphertext with the INT-CTXT model.

Most AE structures in the literature assume that the stretch (tag length) is a fixed value per key. Still, a lack of support for variable stretch may render them vulnerable to tag-length variation under the same key attack [8]. For instance, popular standardized AE schemes such as OCB, GCM, and GCM might have their security degraded or could even suffer a complete loss of security if they are misused in this way [9-11], as raised by Manger [9]. The concern was presented several times in CFRG forum discussions about OCB variable tag length[9] and the CAESAR competition mailing list [12]. For example, If the adversary wages that kind of attack under a scheme using different tag lengths under the same key, the adversary needs to break the shortest one, and the whole security is void.

Besides the security perspective, tag length variability is also advantageous in the constrained resources environment, but recalculating parameters the cost for key exchange because of energy and bandwidth limitations Struik [13]. Reyhanitabar et al. (2017) formalized a nonce-based AE with variable stretch (vNBAE) security notion. They proposed a modular approach for defining the key-equivalent separation by stretch (KESS) concept, which, combined with the traditional NAE, implies the vNBAE security notion.

There have been types of attacks in which the attacker exploits background information about the implementation environment of AE schemes instead of analyzing them under the security models previously discussed. These attacks, known as side-channel attacks (SCAs), are especially harmful when chips containing private data are in an adversary's hands or installed in locations where the general public can access them. Smart cards, sensor network nodes, and IoT devices are vulnerable [14], [15]. SCAs can be avoided using several strategies, such as masking [15-17] and hiding[18] [27, 28, 29, 30]. However, rekeying [15, 19, 20], which uses the target cipher plus a subkey generation algorithm that accepts the master key as input, is a less expensive method of obtaining resistance against side-channel attacks.

AE schemes are constructed employing particular underlying building blocks. Block ciphers are the most used building blocks in AE schemes. Famous block ciphers like AES [21], SKINNY [22], and GIFT [23] are used to create AE schemes. An example of stream ciphers is given in [24]. Dedicated and keyless permutations are the fundamental building block of constructions based on permutations. These permutations use Encrypt Mix Encrypt (EME), Encrypt XOR [25], and variations of the Even-Mansour design in place of sponge-like modes [26]. Furthermore, the cryptographic sponge is the most commonly used keyless permutation. Many algorithms, such as the Keccak-f applied in the SHA3 competition winner, employ keyless permutations in the sponge mode of operation. In contrast, others rely on different permutations [27]. Moreover, Some AE schemes use additional building blocks, such as hash and compression functions (CF), as in [28]. Still, others use unique underlying structures, such as those specified in [29, 30] [31].

In addition to security-related attributes, the following key traits also enhance the effectiveness and performance of AE schemes: parallelizability, which measures the capacity of a scheme to handle the $k^{th}$ block separately from the subsequent $j^{th}$ block, given that $k \neq j$[32, 33]; inverse free: An AE scheme is inverse-free if its algorithm does not need it inverse to carry out encryption or decryption operations [44, 45]; Online use, which demonstrates a scheme's capacity to process the kth block of ciphertext after observing the first *k* blocks of plaintext and without knowing any plaintext after the current block [34]. Incrementality is the capacity to update only parts affected by the most recent operation seen in an earlier ciphertext-tag pair (C, T) [35]. The single-pass feature indicates an AE algorithm's ability to process all the plaintext at once to achieve privacy and authenticity in one pass. Being single-pass boosts the efficiency of an AE scheme [8, 36].

## B. CONTRIBUTIONS

This study proposes and implements a Parallel Sponge-based Authentication Encryption with Variable Tag length and Side-channel Protection (PAVTASP). This work is a complementing component of ongoing efforts to improve AE schemes' security and performance and is motivated by PSASPIN [37] and ISAP [38]. But there are three fundamental ways in which the proposed scheme differs from ISAP: first, PAVTASP is parallelizable; it can process several data blocks at a time; second, PAVTASP makes use of the leveled implementation in a different fashion. For example, while PAVTASP utilizes a PRF based on a block cipher or Galois field multiplication in the key-generation part, ISAP uses the sponge construction in the two implementation levels. On the other side, PAVTASP differs from PSASPIN that it allows variable tag length under the same key without losing other desirable features of the scheme. Another contribution of PAVTASP is that it sets a lower bound for the tag length with the help of rekeying that extends the threshold of the number of operations that can be carried out without negotiating for a new key. Finally, the proposed scheme's security is discussed, and its performance is evaluated, compared to other sponge-based AE schemes, after implementation in C programming language.

## C. ORGANIZATION OF THIS WORK

Section II describes the related work. Next, we present a general AE model in section III, introduce the PAVTASP AE scheme and its processes in Section IV, and discuss the security analysis in Section V. The performance analysis is presented in Section VI. Finally, We offer the discussion in Section VII and conclude this work in Section VIII.

## II. RELATED WORK

Protecting integrity necessities an AE scheme to append the authentication tag to the ciphertext. The expanded part of the ciphertext is also referred to as the ciphertext stretch. The difference between the lengths of the plaintext and the produced ciphertext obtains the stretch value. So, AE schemes traditionally have a syntax where the ciphertext is divided into a core ciphertext and a tag which is concatenated to produce the final ciphertext. The Robust AE (RAE) does not use the partitioned ciphertext syntax, so it uses the general term of stretch in that context [39]. The tag length is an essential element of AE authenticity; the cryptographic strength of an authentication strongly depends on the tag size. Therefore, most authors specify the minimum tag length for their schemes to assume security. Still, specific environments tolerate shorter tag lengths according to certain conditions defined by NIST [40].

Most AE schemes (e.g. [6, 7, 27, 41-44]) assume a stretch is a fixed scheme parameter that should be constant per key. The security is proved according to the assumption that different stretch values use distinct keys. However, the variable stretch per key could happen either as a result of misconfiguration or attack; in that case, the security would be violated [39]. Examples of compromised Security because of misuse include attacks on OpenSSH, EAXPrime, and VMWare View remote desktop protocol [45].

The CAESAR [46] and NIST-LW[47] competitions provided guidelines for protecting confidentiality, integrity, robustness, and suitability for use in constrained environments. The robustness discussed until recently mainly focused on some instances of nonce misuse resistance; however, other misuse cases, like tag variation under the same key misuse, have not had enough attention [8].

In addition to its security relevance, tag length variability is desirable in the constrained environment, but negotiating parameters cost is preventively high due to energy and bandwidth limitations. Struik [13] indicated that supporting a variable stretch under the same key would provide a slide scaling for authenticity, extending the lifetime of constrained resources sensors, especially when processed plaintexts are very short. At the same time, only a few packets would need high authenticity.

The issue was raised several times in CFRG forum discussions about OCB variable tag length[9] and the CAESAR competition mailing list [12]. The discussions motivated the modification of several second candidate schemes[12, 48, 49] to be modified for some heuristic solutions to the problem to accommodate variable tag length under the same key. The absence of variable tag length support is not just a theoretical concern because widely deployed schemes such as OCB, CCM, and GCM malfunction in one way or another once misused in this way [9-11]. That misuse may cause degraded security to complete loss of security, as raised by Manger [9]. For instance, if those schemes use different tag lengths under the same keys, if the attackers have a 128-bit tag, it's trivial for them to produce a valid output with a 64-bit tag under the same key by dropping the last 64 bits because shorter tags are simply the truncation of longer ones.

Reyhanitabar et al.(2017) discussed the issue in detail, formalized a security notion vNBAE, then came up with an all-in-one security definition for it. Then the authors proposed a modular approach for defining the concept called key-equivalent separation by stretch (KESS), which, combined with the traditional Nonce-based AE (NAE), implies the vNBAE security notion. Finally, the authors proved that the vNBAE goal was efficient and provably achievable, applying simple tweaks to existing schemes by concretizing it with the modification of OCB without sacrificing its desirable features, such as the online processing of data blocks [8]. Finally, the authors proved that the vNBAE goal was efficient and provably achievable, applying simple tweaks to existing schemes by concretizing it with the modification of OCB based on a tweakable block cipher [8]

Side-channel attacks (SCAs) are implementation-based security threats that exploit the connection between cryptographic algorithms and the emission patterns of implementation environments, such as electromagnetic emissions, radiation emissions, and power consumption traces. The essential idea behind these attacks is to infer a secret key from how the side-channel signal pattern is related to it [18, 50]. When cryptographic equipment is mounted in a location where attackers can physically reach it, SCAs are more dangerous. Therefore, many sources in the literature advocated several countermeasures against SCAs, such as masking [15-17, 51] and hiding [18]. However, these solutions come with unsustainable performance costs in situations with limited resources, like IoT devices and smart cards. Therefore, fresh rekeying [19, 20] is a more affordable option to get SCA protection compared to the other methods listed. Furthermore, by limiting the usage of each session key to just a single or limited number of times, fresh rekeying protects against SCAs making more difficult for adversaries to collect intermediate key related values [19, 20, 52, 53].

The sponge construction is an iterative cryptographic primitive for creating a function $f$ that receives inputs of any length and produces outputs of any size using transformations or permutations of fixed length. Bertoni et al. initially proposed sponge construction [54]. It acts on a state $b$ that consists of a rate part $r$ and a capacity part $c$, where $b = r + c$ bits [54]. The sponge absorbs its input blocks first, then processes and squeezes them out as an output truncated to the desired length. The keyless permutation method most frequently employed in AE is the sponge structure. Stream ciphers and re-seedable pseudorandom generators are two further cryptographic applications of Sponges in addition to AE [27]. Depending on the functionality needed, there are different ways to employ the sponge function. For instance, online, single-pass AE methods are typically implemented using the Duplex modes and their variation MonkeyDuplex *[27, 55]* modes. Therefore, we can use the Sponge/Duplex lemma *[54]* to prove Duplex mode of operation is as secure as the sponge construction. Ascon [56], one of the winners in the CAESAR competition as well as five of the ten finalists in the NIST competition for lightweight AE, including Ascon *[57]*, Elephant *[58]*, ISAP *[59]*, Photon-Beetle *[60]*, and Xoodyak *[61]*, were based on sponge construction.

Several parallelizable AE schemes based on sponge construction have been proposed. For instance, the AE schemes in the works [62-64] are incremental and parallelizable to varying degrees, but they are not protected against SCAs. Other sponge-based AE schemes are protected against side-channel attacks but are not parallelizable, incremental, or single-pass [59, 65, 66]. PSASPIN AE is a parallel, sponge-based AE and is defended against Differential Power Analysis (DPA) and Simple Power Analysis (SPA), but it does not support a variable tag length under the same key. This study proposes a parallel sponge-based AE that supports variable tag length and protects against SPAs and DPAs. See Table 1 for a comparison of the proposed solution and other AE schemes based on the sponge construction and its operation modes.

**Table 1: PAVTASP compared to other AE schemes.**

| Scheme | Parallel | Incremental | Single-pass | Online | SCA Protection | Variable stretch |
|---|---|---|---|---|---|---|
| A Parallel AE based on the duplex mode of construction (Morawiecki & Pieprzyk, 2013) | Yes | Yes | Yes | Yes | No | No |
| $\pi$–Cipher v11 (Gligoroski *et al.*, 2014) | Yes | Yes | No | Yes | No | No |
| Spook (Bellizia *et al.*, 2019) | No | No | No | No | Yes | No |
| ISAP ((Dobraunig *et al.*, 2019) | No | No | No | No | Yes | No |
| SALE (Degabriele, Janson, Struck, 2019) | No | No | No | No | Yes | No |
| NORX[62] | Yes | Yes | Yes | Yes | No | No |
| ASCON[56] | No | No | Yes | Yes | No | No |
| PSASPIN[37] | Yes | Yes | Yes | Yes | Yes | No |
| PAVTASP (the proposed scheme) | Yes | Yes | Yes | Yes | Yes | Yes |

## III- MODELING AUTHENTICATED ENCRYPTION

We can think of the AEAD as a function that takes in a secret key *K*, plaintext *M*, associated data AD, a nonce (*N*) and outputs a ciphertext *C*, and an authentication tag *T*. its encryption algorithm can be modeled as $E: K \times N \times AD \times M \rightarrow C|T$—, and its decryption can be modeled as $D: K \times N \times AD \times C \rightarrow P\{\perp\}$. Separated AE supports a verification algorithm, $V: K \times N \times AD \times C \times T \rightarrow M\{\top, \perp\}$, in addition. The encryption algorithm, $E_K(N, AD, M) = (C, T)$, and the decryption algorithm, $D_K(N, AD, C) = M$ if *(C, T)* is valid; otherwise, it yields $\perp$; the verification algorithm $V_K(N, AD, C, T) = \perp (failure\ symbol)$ if a forgery is discovered [67-69].

## IV. SPONGE-BASED PARALLEL AUTHENTICATED ENCRYPTION WITH VARIABLE TAG LENGTH AND SIDE-CHANNEL PROTECTION (PAVTASP)

The proposed scheme, PAVTASP, is an AEAD sponge-based sponge construction with a state width ***b*** of 320 bits, a rate part ***r*** of 128 bits, and a capacity ***c*** part of 192 bits, where ***b=c+r***. The scheme prevents SPAs and DPAs by generating a fresh session key in every process using parallel fresh re-keying and has the following necessary properties that are critical for the performance and security: Side-channel protected, parallel, single-pass, incremental, and online. In addition, it supports variable tag length without compromising security or losing other desirable features.

### A. NOTATIONS

Here are definitions for the notations in this paper. The letters ***K, N, T,*** and ***IV*** are used respectively to represent the key, the nonce, the authentication tag, and the initialization vector. The plaintext message, the ciphertext, and the associated data are each denoted by ***M, C***, and ***A***, respectively. A failure of verification or an error is what we mean by ($\perp$). ***S*** stands for the 320-bit state of the sponge construction. $\boldsymbol{S_r}$ represents the rate part of the state, whereas $\boldsymbol{S_c}$ represents the capacity part of the state. The length of the string *'X'* is denoted by the symbol |*X*|, whereas the text *'X'* concatenated with the string *'Y'* is designated as ***X*||*Y***. ***P*** is the sponge permutation, $\boldsymbol{0^k}$ stands for an entirely 0-bit string of length ***k***, and |***X***| is the symbol for the length of the string '***X***.' We represent the XOR of the strings ***'X'*** and ***'Y'*** as $X \oplus Y$. By $\lfloor X \rfloor_k$, we refer to a bitstring ***X*** that has been truncated to its last ***k*** most significant bits. We refer to a bitstring ***X*** that has been truncated to the first k least significant bits as $\lfloor X \rceil_k$. Finally, we denote $\boldsymbol{\tau}$ by the tag length(stretch).

### B. PARAMETERS

PAVTASP is an AE scheme using a 320-bit permutation ***P*** that applies eight rounds of ASCON permutations [57]. It requires five inputs: a 128-bit secret master ***K*** from which a session key $\boldsymbol{K_S}$ is obtained, a variable length tag length (stretch) $\boldsymbol{\tau}$, a variable length plaintext ***M***, a variable length associated data ***AD***, and a 128-bit nonce ***N***. Decryption requires a 128-bit session key $\boldsymbol{K_S}$ obtained from ***K***, a ciphertext

***C***, a nonce ***N***, an authentication tag ***T*****,** and a variable length stretch value $\boldsymbol{\tau}$. For protection against SCAs, PAVTASP employs fresh rekeying to obtain a new key each time encryption/decryption and authentication operations are invoked. Figure 1 and Figure 2 respectively show a schematic view of the PAVTASP encryption and decryption processes. The initialization, encryption, and decryption/verification processes are described below. PAVTASP is a single-pass scheme that performs encryption and authentication in a single pass over the duplex structure. In addition, the scheme allows variable length and protects against SPAs and DPAs.

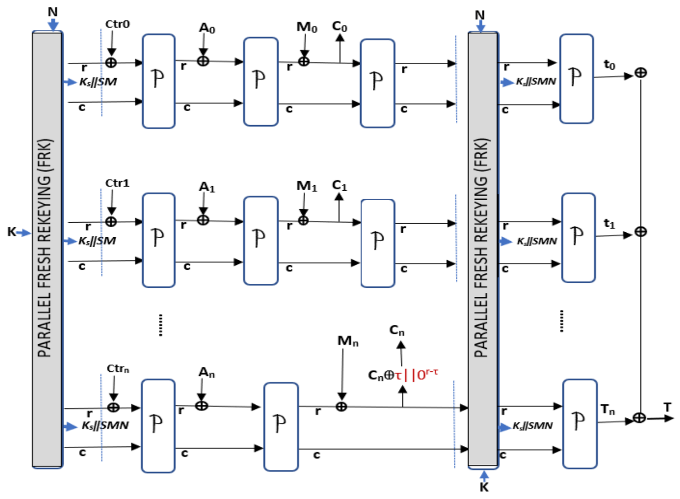


Figure 1: Schematic view of PAVTASP encryption.

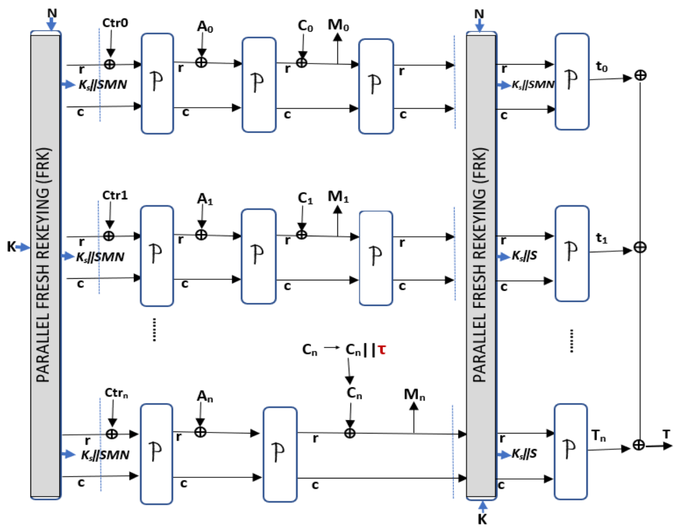


Figure 2: Schematic view of PAVTASP decryption.

### *C. PAVTASP PROCESSES*

#### (1) INITIALIZATION

In the initialization stage, the parallel fresh re-keying function (PFRK) is called. To protect the parallel threads of the system against SCAs, it requires a master ***K*** and a nonce ***N*** and generates new session keys $\boldsymbol{K_{S,\,i}}$. The first ***N*** is produced at the source, shared in a secure manner, and incremented in every process to maintain synchronization between the communicating sides. This work assumes the existence of a secure way of making the keys and nonces available to the parties. For instance, key distribution mechanisms such as key wrapping schemes [70], could apply to the nonces. Another possibility is hiding the nonce in the ciphertext and extracting it at the destination using existing hide-nonce-transforms [37]. See Algorithm 3 for details on FRK. Once the secret session key is generated, the state ***S*** is updated by feeding the session key concatenated with a Secret Message number (***SMN***) to permutation ***P***. The system generates a secure counter *Ctr* to monitor the number of parallel lanes and is incremented by 1 with every thread. The state is updated by XORing it with the *Ctr* variable before processing the AD part*;* $S \leftarrow (Ctr \oplus S_r) \parallel S_c$.

#### (2) ASSOCIATED DATA (AD) PROCESSING

PAVTASP first divides the ***A*** into ***r***-bit blocks and pads it with '1' and the list number of '0's to make the length of ***A*** a multiple of ***r***. There is no need for padding if the AD block is empty. Associated data is processed one block at a time, $A_0||A_1||\ldots\ldots A_i||$, $|A|=r$, then, each block of *A* is XORed with $\boldsymbol{S_r}$ (the rate part), and after concatenating with $S_c$ (the capacity part), the shared state ***S*** is updated by the permutation ***P*** as following: $S \leftarrow ((S_r \oplus A_i) \parallel S_c)$. After processing the last block of associated data $\boldsymbol{A_i}$, a domain separator of 1-bit is XORed with the state *S*: $S \leftarrow S \oplus (0^{319} \parallel 1)$.

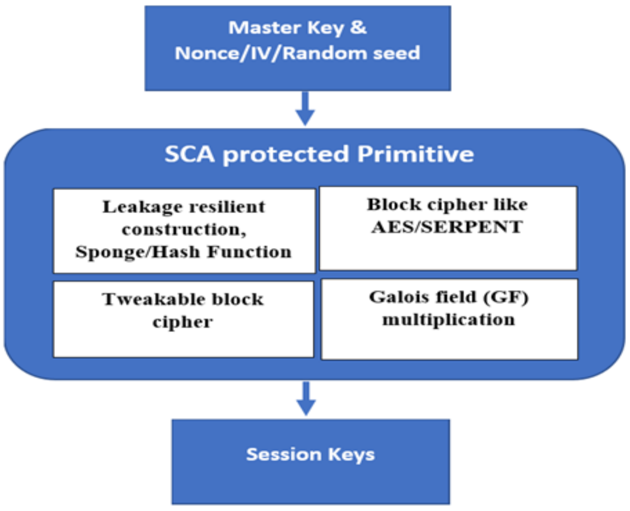

Master Key &
Nonce/IV/Random seed
SCA protected Primitive
Leakage resilient construction, Sponge/Hash Function
Block cipher like AES/SERPENT
Tweakable block cipher
Galois field (GF) multiplication
Session Keys

Figure 3: Rekeying function options that can be used in PAVTASP[37].

#### (3) ENCRYPTION

The padded plaintext (*M*) is broken into *r*-bit blocks and processed block by block: $M||1||0^{r-1-(|M| \bmod r)} = M_0||M_1||\ldots||M_n$. For all plaintext blocks except the last, each block of plaintext ($M_i$) is XORed with the outer part of the state ($S_r$): $C_i \leftarrow S_r \oplus M_i$, to output the ciphertext block ($C_i$), after which the state is updated: $S \leftarrow (C_i \parallel S_c)$. The resulting state is truncated so that the total length of the ciphertext is equal to that of the original unpadded length of the plaintext: $C_z \leftarrow \lfloor S_r \rfloor_{|M| mod\ r}$. The resulting ciphertext block is then XORed with the padded stretch value to produce the final text in the following manner: $C_z \leftarrow \tau||0^{r-\tau} \oplus C_z$. See algorithm 1 for details.

### (4) DECRYPTION

The padded ciphertext ($C$) is split into $r$-bit blocks and processed block by block: $C||1||0^{r-1-(|M| \bmod r)} = C_0||C_1||....||C_z$. The last ciphertext block ($C_z$) is parsed into a ciphertext block and the stretch part in the following manner: $\tau||C_z \leftarrow C_z$. The rest of the process is identical to encryption; the two processes differ only in that plaintext and ciphertext are swapped. See algorithm 2 for details.

### (5) FINALIZATION

In the finalization and authentication state, the fresh re-keying function is again called for protection against SCA and forgery attacks. Once the session key $K_s$ is generated and concatenated with ***SMN***, the shared state is updated by transforming the XOR of state $S$ with the padded fresh session key $K_S$, $S \leftarrow P(S \oplus (0^r||K_s||0^{c-r-k})$. For all the parallel threads except the last one, intermediate tags $t_i$s are generated by truncating the state XORed with the session key to the last $\boldsymbol{\tau}$ bits, and the intermediate tags are combined. In the last threat (once the counter $Ctr$ is equal to $n$), the final intermediate tag $T$ is generated and concatenated to the ciphertext blocks, $T \leftarrow t_0 \oplus t_1 \ldots . \oplus t_{n-1}$. In the finalization of the decryption process, the plaintext is returned if the generated tag length is equal to the stretch value $\boldsymbol{\tau}$, and the generated tag $\boldsymbol{T''}$ is qual the $\boldsymbol{T}$ parameter $T$ of the decryption; otherwise, the process aborts. *If* $T''=T$ *and* $|T|=\tau$ *Return* $M_1||\ldots||M_i||M_z$.

### (6) THE REKEYING FUNCTION

Figure 3 illustrates several implementation options for the rekeying function. The leveled implementation used in this work is adapted from the method proposed by Abdalla and Bellare [20] and implemented by [19, 71]. The structure is split up into a rekeying function and a data processing (rekeyed) component. Several alternatives for implementing the data processing part include block ciphers like AES, sponge construction and its operation modes, and tweakable block ciphers [66]. On the other hand, a PRF serving as a pseudorandom generator ($G$) can be used to implement the rekeying function with different options, including the following: Constructions based on Galois field (GF) multiplication, leakage-resistant primitives like duplex sponges [38, 66], protected block ciphers like SERPENT [72] and AEAS, tweakable block ciphers or traditional block ciphers strengthened with countermeasures like hiding and masking [52, 65, 73].

In this work, the rekeying function can be constructed in one of two ways using the leveled implementation. Using a rekeying function built on a GF multiplication field is the first option similar to those in [71, 74]; however, some modifications are necessary because those implementations only protect against lower-order DPAs. On the other hand, this work integrates the defense mechanisms to protect against higher-order DPAs. A leakage-resilient block cipher is an additional option with a more complex design than the first one but is preferable for hardware implementations. The function ***G*** based on the GF multiplication is implemented in Algorithm 3 using combined shuffling and masking for defense against higher-order DPAs.

```
Algorithm 1: Encryption
E (A,M,K,N,$\tau$)
Input: Key $K \in \{0,1\}^k$, $K \le 128$,
    Plaintext $M \in \{0,1\}^*$,
    Nonce $N \in \{0,1\}^{128}$,
    Associated Data $A \in \{0,1\}^*$,
    Stretch $\tau \in \mathbb{N}$,
Output: Ciphertext $C \in \{0,1\}^{|M|}$,
    Tag $T \in \{0,1\}\ |\tau|$
Initialization
    ///A new counter value
    $Ctr = \lceil K \rceil_{|r|}$
    //Call the fresh rekeying function
    $K_s \leftarrow PFRK(K,N)$ ;$S \leftarrow K_s||SMN$
Processing the Associated Data
    Let $A = A_0||A_1||....||A_x$, $|A_i| = r$, for $i<x$, $A_y \le r$ and $|A_x|>0$
    if $x>0$
    for $i = 1, ..., x$ do
        $S \leftarrow P((Sr \oplus A_i)||S_c)$
    $S \leftarrow S \oplus (0^{319}||1)$
Processing Plaintext
    Let $M = A_0||M_1||....||M_n$, $|M_i| = r$, for $i<n$, $M_n \le r$ and
    $|M_n|>0$ if $n>0$
    for $I = 1,...,n-1$ do
        $Sr \leftarrow Sr \oplus Mi$ ;$CM \leftarrow Sr$ ;$S \leftarrow P(Sr||Sc)$
        $Sr \leftarrow Sr \oplus M_{nn}$
    $C_n \leftarrow \lfloor S_r \rfloor_{|M| mod\ r}$
    $Cz \leftarrow \tau||0^{r-\tau} \oplus C_n$
Finalization
    $Ks \leftarrow PFRK(K,N)$
    $S \leftarrow P(S \oplus (0_r||K_s||0^{c-r-k})$
    $t_i \leftarrow \lceil S \otimes K_s \rceil_{|\tau|}$
    //if final lane
    If $Ctr = n$
        $T \leftarrow t_0 \oplus t_1 \ldots . \oplus t_{n-1}$
    Return $C_1||...||C_i||C_n||T$
Algorithm2 : Decryption
D (A,C,K,N,T, $\tau$)
Input: Key $K \in \{0,1\}^k$, $K \le 128$,
    Ciphertext $C \in \{0,1\}|M|$,
    Associated Data $A \in \{0,1\}^*$,
    Nonce $N \in \{0,1\}128$,
    Stretch $\tau \in \mathbb{N}$,
Tag $T \in \{0,1\}\ |\tau|$
Output: Plaintext $M \in \{0,1\}^{*\ or} \perp$
```

```
Initialization
        $Ctr = Ctr = \lceil K \rceil_{|r|}$
        //Generate a fresh subkey
        $Ks \leftarrow PFRK(K,N)$
        //Call the fresh rekeying function
        $S \leftarrow Ks||SMN$
Processing the Associated Data
        Let $A = A_0||A_1||....||A_x$, $|A_i| = r$, for $i<x$, $Ay \le r$ and $|A_x|>0$
        if $x>0$
        for $i = 1, ..., x$ do
        $S \leftarrow P((Sr \oplus A_i)||Sc)$
        $S \leftarrow S \oplus (0319||1)$
Processing Ciphertext
        Let $C = A_0||C_1||....||C_n$, $|C_i| = r$, for $i<n$, $C_n \le r$ and $|C_n|>0$
        if $n>0$
        for $i = 1,...,n-1$ do
        $M_i \leftarrow Sr \oplus C_i$   ;$S \leftarrow CM_i||S_c$
        $S \leftarrow P(S)$
        $\tau||C_n \leftarrow C_n$
        $M_n \leftarrow \lfloor S_r \rfloor_{n|} \oplus C_n$
        $S_r \leftarrow S_r \oplus (M_n||1||0^*)$
Finalization
        $Ks \leftarrow PFRK(K,N)$
        $S \leftarrow P(S \oplus (0^r||K_{s||0c-r-k})$
        $t_i \leftarrow \lceil S \otimes K_s \rceil_{128}$
        //in the final lane
        $If\ Ctr = n$
                $T" \leftarrow t_0 \oplus t_1 \dots . \oplus t_{n-1}$
        If $T" = T$ and $|T| = \tau$
                Return $M_1||...||M_i||M_n$
```

"*Algorithm 3: Parallel Fresh Rekeying (PFRK)*

```
            $Requires:\ a, b \in GF(2^2)[y]/y^d + 1$
 $Ensures: c = b * b \in GF(2^2)[y]/y^d + 1$
$x \leftarrow rand(), j \leftarrow x, k \leftarrow x, with\ h = 1 - m, 0 \leftarrow st$
$while\ k\ \#\ x - 1\ mod\ d\ \boldsymbol{do}$
    $k_b \leftarrow k$
    $for\ h = 1\ to\ m\ \boldsymbol{do}$
        $kb_h \leftarrow kb_h \oplus b_j$
        $j \leftarrow j + 1\ mod\ d$
        $End\ for$
        $k_s \leftarrow N.bk_h$
        $for\ h = 1\ to\ m\ \boldsymbol{do}$
        $k_s \leftarrow N \cdot (b_j \oplus bk_h)$
    $End\ for$
$End\ while$
$Return\ (k_{s_h}, st + 1)$"[37]
```

## V. SECURITY ANALYSIS

We evaluate PAVTASP security in terms of its two implementation levels. The first level is the rekeying function, which protects against DPA and SPA and generates session keys. The second level is the sponge function, based on the duplex construction, which protects against SPA. We can measure the overall adversarial advantage according to its ability to compromise the key generation function and its ability to compromise the base rekeyed scheme. Furthermore, there are various countermeasures for protection against SCAs. Examples include employing session keys for one or more tasks, hiding, masking, and applying logic styles. However, [18] indicated that the most effective way is to combine countermeasures rather than using them separately. For example, we can combine shuffling and masking for defense against higher-order DPAs.

### *A. Security of Parallel Fresh Rekeying Function (G).*

The parallel rekeying function creates session keys for use in the AE schemes' encryption component using an initial master key [15, 20]. This method enables us to encrypt more data under the same key, increasing the key's lifetime. Rekeying functions can be divided into two categories: parallel rekeying, which generates session keys, all at once, separately, and serial rekeying, in which the generated session keys depend on the prior states and are updated continuously [20]. According to [53], when concurrent access to data is implemented, parallel rekeying is required.

According to [20], a stateful generator's pseudo-randomness is defined as follows: Consider the following experiment taking into account $G = (K, N)$ as a stateful generator with a block size of ***k***, ***n*** as an integer, and ***A*** as an adversary:

```
$Experiment\ EXP_{G,n,A}^{prg-real}$
  $for\ i = 1, \dots., n\ \boldsymbol{do}$
  $(Out_i, St_i) \leftarrow N(St_{i-1}); \leftarrow st \parallel Out_i$
    $g_n \leftarrow A(st)$
    $return\ g_n$
$Experiment\ EXP_{G,n,A}^{prg-rand}$
$st \leftarrow \{0,1\}^{n.k}$
$g_n \leftarrow A(st)$
$return\ g_n$
```

The security evaluation of the proposed parallel fresh rekeying function can be described concerning the security notions of pseudorandom generators. The approach proposed in [20] is followed in this work, but their schemes protect a block cipher, whereas that proposed here protects a parallel sponge-based AE scheme. Pseudorandomness, which represents adversary *A*'s inability to differentiate the generator's output from a random string of identical length, is the desirable property of the generator. Real and random experiments define adversary *A*'s advantage and the generating function's advantage *(ADV)* in the following way:

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

$ADV_{G,n,A}^{prg}(t) = max_A\left\{ADV_{G,n,A}^{prg}\right\}$. The advantage function quantifies the likelihood that *A* can compromise function *G* using the abovementioned resources. The maximum is overall *A* running in *t*, the time complexity, and the total time complexity is the running time of the two experiments, in addition to the size of adversary *A*'s code. The underlying PRF is what ensures the key generation function security $F: \{0,1\}^l \times \{0,1\}^l \times \{0,1\}^l \to \{0,1\}^l$. Let $\{0,1\}^l\ to$ be a function family that maps an *l*-bit string to an *l*-bit string assuming a uniform distribution. If **D** is a distribution with Oracle access, then

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

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

The overall *A*'s advantage is the maximum, *t* denotes the time complexity, and *q* represents the oracle queries executed. In the following, we demonstrate how the underlying PRF affects the pseudorandom of the parallel fresh rekeying function.

Theorem 1

Let $F: \{0,1\}^l \times \{0,1\}^l \to \{0,1\}^l$ be a PRF, and let *G*[*F*] be a parallel key generator; then $ADV_{G[F\},l}^{prg}(t) \leq ADV_F^{prf}(t,l)$.

Proof: Assume that adversary A is trying to defeat the pseudo-randomness of *G*[*F*], and assume *t* to be the running time of $EXP_{G[F],l,A}^{prg-real}$. An upper bound is determined for $ADV_{G[F],l,A}^{prg}$. Then a distinguisher *D* for *F* is constructed whose advantage is related to that of *A*. The distinguisher *D* interacts with an oracle *B* that calculates *s=B(1)*||.....||*B(n)* and produces the same guess of *A* on input *s*. When *B* is randomly drawn from *F*, the likelihood that the distinguisher *D* produces 1 is equal to the likelihood that $EXP_{G[F],n,A}^{prg-real}$ produces 1. Alternatively, the likelihood that $EXP_{G[F],n,A}^{prg-rand}$ produces 1 is equal to that of *D* producing 1 assuming that *B* is drawn randomly from a random function family $R^n$. Because *D's* running is *t*, makes at most *l* queries to its oracle, it follows that $ADV_{G[F],l,A}^{prg} \leq ADV_F^{prf}(t,l)$. *A* is an adversary, and the combined maximum time of the two experiments is *t*, and that completes the proof. The Security of the PRF (*F*) under *l* queries determines, quantitatively, the pseudo-randomness of the fresh rekeying function *(G)*. When *F* is a PRF, then $ADV_{G[F],l}^{prg}(t) \approx \frac{l+t}{2^k}$.

## *B. The Base AE Scheme Security.*

The sponge-based AE scheme, PAVTASP, is based on the duplex mode of operation. It receives a plaintext *M*, associated data AD, a nonce *N*, a stretch value *τ*, and a master key *K* fed into the pseudorandom generator to produce subkeys to process data blocks in a parallel fashion. At the core of AE security, we consider two notions of security for sponge construction, confidentiality, and integrity [2, 6, 7]. Finally, the method put forward by Jovanovic *et al*. [75], Andreeva *et al*. [30], and Mihajloska *et al*. [76] is used to prove the security of the rekeyed part of PAVTASP.

## *C. Confidentiality (or Privacy)*

Confidentiality ensures that only legitimate parties can view the messages in the IND-CPA model against passive attackers and the IND-CCA model against active attackers. The attacker is granted access to an encryption oracle in the first model and a decryption oracle in the second. The adversary's advantage must always be insignificant for an AE scheme to be secure [2, 6, 7].

Consider *P* a collection of idealized permutations of an AE scheme *II*. The following formula describes *A's* advantage (*ADV*) while having access to both forward and inverse permutations in breaching scheme *II*'s privacy:

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

The $P^{\pm}$ denotes that *A* can make queries to forward and inverse permutations. Assuming that *A* does not call $E_k$ and *$* using the same nonces, $ADV_{II}^{priv}(q_p, q_e, \lambda_e)$ represents all adversaries the maximum advantages making queries to $E_k$ or *$*.

## *D. Integrity or Authenticity*

Integrity guarantees that communications come from reliable parties and haven't been changed while in motion or at rest. AE provides plaintext integrity under the INT-PTXT paradigm and ciphertext integrity under the INT-CTXT paradigm. The former assures that the attacker cannot forge ciphertext decryption of data that the sender did not previously encrypt. The latter guarantees that the adversary cannot come up with a ciphertext that the sender had not created, regardless of whether the plaintext is new [2, 6].

Assume that *P* is a collection of the AE scheme *II*'s underpinning idealized permutations. Then, we can describe the integrity-related goals of AE are defined as demonstrated by the adversary *A*'s failure to produce a new plaintext that is not the outcome of a valid decryption *(D*$_k$*(C))* process under a valid key *K*:

$$ADV_{II}^{auth} = Pr_{P,K}\left(A^{P^{\pm},E_K,D_K}\ Forges\right),$$

The probability is taken over *A*, and *K*, assuming *P* has been chosen randomly. We say that adversary *A* wins in making a forgery if $D_k$ produces a message that is different from ⊥ on receiving an input *(A, C, N, T)*, and (*A, C*) have not been created by $E_k$ after taking *(N, A, M)* as input. The adversary is also assumed to be nonce-respecting in that it does not repeat the same nonces, as in the privacy case. Let us represent authenticity as $ADV_{II}^{auth}(q_p, q_E, \lambda_E, q_D, \lambda_D)$. We can determine the maximum advantage over all adversaries by querying $P^{\pm}$ at most $q_p$ times making at most $q_E$ queries of total length at most $\lambda_E$ blocks to $E_K$ and at most $q_D$ queries of the total length $\lambda_D$ to $D_K/\perp$.

For the proof of PAVTASP's privacy, we consider an adversary *A* making $q_P$ permutation queries and $q_E$ encryption queries whose total length is $\lambda_E$. For integrity proof, we consider adversary *A* making $q_D$ decryption queries totaling a length of $\lambda_D$. The number of permutation calls is determined by $q_E$ encryption queries, and the same procedure is repeated for encryption queries with similar parameters. Considering $q_E$, of *c* associated data blocks and *f* message blocks, and *T* intermediate tags, the equivalent *n* state values can be described as follows:

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

The number of state values $\sigma_{e,j}$ is *c*+*f*+4, assuming that the *j*th query is *c*+*f* blocks, and which results in the number of Π evaluations using the encryption query:

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

We calculate for $\sigma_D$ and $\sigma_{j,D}$ in the same manner.

## E. Syntax of NBAE with variable tag length

We can expand the syntax of Nonce-Based Authenticated Encryption (NBAE) schemes to include a variable tag length as proposed by [8]. An AE scheme with variable stretch consists of a triplet $\Pi = (K, E, D)$ where $K \subseteq \{1,0\}^*$ a set of keys with a uniform distribution, and $E: K \times N \times \mathcal{T}_T \times M \to C$ and $D: K \times A \times N \times \mathbb{N} \times C \to M \cup \{\perp\}$ are encryption and decryption algorithms in that order. In this context, we call $N$ the nonce space, A the Associated Data space, M the plaintext space, C the ciphertext space, and $\mathcal{T}$ the stretch space of the scheme (Π). We assume that $N \subseteq \{1,0\}^*, M \subseteq \{1,0\}^*, A \subseteq \{1,0\}^*, C \subseteq \{1,0\}^*, and\ \mathcal{T}_T \subseteq \mathbb{N}$. We also assume that if $M \in M\ then\ \{0,1\}^{|M|} \subseteq M.$

Reyhanitabar, Vaudenay [8] proposed a modular syntax for achieving the vNBAE, key-equivalent separation by stretch (KESS). The assumption is that the scheme should behave as having a fresh independent key with each value of a separate tag length value. In addition, they stated that KESS ensures that the scheme instances using different tag lengths be independent and inaccessible to one another rather than encouraging short tag lengths or claiming particular robustness.

Let $\Pi = (K, E, D)$ be and vNBAE scheme. Let A be an adversary that tries to break KESS of Π by distinguishing two games, one containing the encryption and decryption oracles of the Real schemes and another of an Ideal scheme with similar parameters. The advantage of A in breaking the scheme is measured by $ADV_{\Pi}^{KESS}(A) = \Pr[A^{KESS-Real_{\Pi}} \Rightarrow] - \Pr\,[A^{KESS-Ideal_{\Pi}} \Rightarrow 1].$ When combined with NBAE security, KESS implies vNBAE Security. KESS's role is merely to handle the interaction between queries with different tag lengths so that queries of $\tau$ bit of stretch are independent of one another. So we have: $(KESS \wedge NBAE) \Rightarrow vNBAE. Let\ \Pi = (K, E, D)$ be a nonce-based AE scheme with a variable stretch; we have that:

$$\begin{aligned} ADV_{\Pi}^{vNBAE\ (\tau_c)}(t, q_E, q_D, \sigma) \\ &\leq ADV_{\Pi}^{KESS}(t', q_E, q_D, \sigma) \\ &+ ADV_{\Pi[c]}^{NBAE}\left(t'', q_E^{\tau_c}, q_D^{\tau_c}, \sigma^{\tau_c}\right), \\ with\ t' = t + O(q)\ and\ t'' = t + O(\sigma)\ where\ q \\ &= \sum\nolimits_{\tau \in \mathcal{T}_T} (q_E^{\tau} + q_D^{\tau})\ and\ \sigma \\ &= \sum\nolimits_{\tau \in \mathcal{T}_T} (\sigma_E^{\tau} + \sigma_D^{\tau}). \end{aligned}$$

The adversarial resources are $(t, q_e, q_d, \sigma)$ where $t$ is the adversarial running time $q_D = (q_E^{\tau} \backslash \tau \in \mathcal{T}_T)$ stands for the vector of the number of encryption queries made with the stretch value $\tau$ for every stretch $\tau \in \mathcal{T}_T$, and $q_D = (q_D^{\tau} \backslash \tau \in \mathcal{T}_T)$ denotes the number of decryption queries, and $\sigma = (\sigma^{\tau} \backslash \tau \in \mathcal{T}_T)$ stands for the vector of the total amount of data processed by all queries with stretch value $\tau$ for every $\tau \in \mathcal{T}_T$. For a resources parameterized function of an AE scheme Π for a given tag length value $\tau_c$ the adversarial advantage can be defined as $ADV_{\Pi}^{vNBAE(\tau_c)}(r_{\tau_c}) = Max_A\{ADV_{\Pi}^{\tau_c}(A)\}$. The maximum is taken over all adversaries with $r_{\tau_c}$ bound resources, the scheme is secure if, for all practical adversaries, with the resources mentioned above, the advantage is negligible. Thus a scheme Π is a vNBAE-secure if for every stretch value $\tau_c \in \mathcal{T}_T$, for all practical adversaries with the specified resources, the advantage $ADV_{\Pi}^{vNBAE(\tau_c)}(r_{\tau_c})$ is small, keeping in mind that the advantage $ADV_{\Pi}^{vNBAE(\tau_c)}(r_{\tau_c})$ will be inevitably high if the stretch value $\tau_c$ is small.

## *F. A lower limit for tag lengths*

As more data is processed with a single key, the security assurance provided by cryptographic systems deteriorates. Therefore, limiting the amount of plaintext and associated data blocks protected by calls to the authenticated encryption function during the key lifetime is recommended. For instance, $2^{64}$ would be a reasonable limit for most applications. Jovanovic, Luykx [75] set the integrity bound (tag length) of sponge-based schemes to $2^{c/2}$ where c is the capacity. As soon as the lower bound limit approaches, a key exchange should be negotiated for the security to hold. Abdalla and Bellare [20] and Mennink [15] stated that with fresh rekeying, the security bounds of schemes could be enhanced, for instance, from $2^{k/2}$ to $2^{k/3}$. For that reason, we claim that PAVTASP enhances the minimum recommended tag length of $2^{64}$ for the equal key length and capacity of 128 bits to $2^{128/3}$ or approximately $2^{42}$ bits. NIST standard on this issue requires the implementing parties to be careful when using shorter tag lengths [40]. For instance, they recommend that packets that fail the integrity should be discarded silently to prevent them from giving useful information to the potential attackers and limiting the associated data packet to contain only the necessary header information.

## *G. PAVTASP adversary model*

In this study, adversary A is considered powerful, with full access to the communication medium, intent on compromising privacy and integrity, access to encryption and decryption oracles, and the ability to employ various tag lengths while using the same key. Figure 4 depicts the PAVTASP adversarial model following the approach of Do et al. [77] and Jimale, M. R [37].

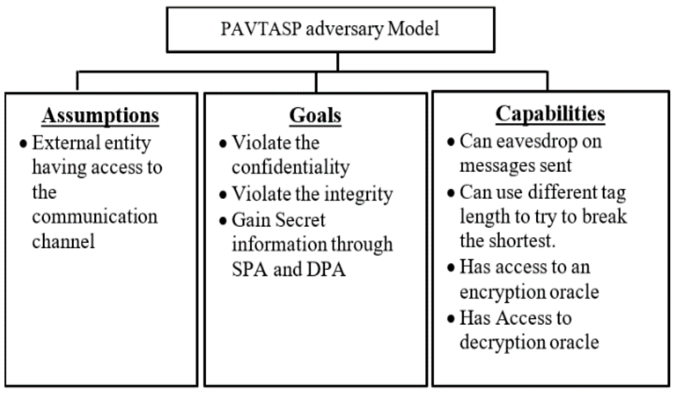

PAVTASP adversary Model
Assumptions
• External entity having access to the communication channel
Goals
• Violate the confidentiality
• Violate the integrity
• Gain Secret information through SPA and DPA
Capabilities
• Can eavesdrop on messages sent
• Can use different tag length to try to break the shortest.
• Has access to an encryption oracle
• Has Access to decryption oracle

Figure 4: PAVTASP adversary model

If adversary A cannot breach PAVTASP's Security with a non-trivial probability under the specified assumptions, capabilities, and goals, then PAVTASP is assumed secure. For example, although A can utilize different values of stretch (tag lengths) under the same key, the KESS property instances of the schemes using stretch values are separated and cannot interact, so that will not increase the success probability of A**.**

## VI. PERFORMANCE ANALYSIS

Although security is the most decisive factor for cryptographic algorithms, performance is equally vital because of the continuously shifting information processing paradigms and diverse implementation platform requirements. For instance, the ever-increasing reliance on online data processing necessitates speedy accessibility of information for better user experiences. Therefore, we implemented PAVTASP to evaluate its performance and measure it against three other Sponge-based AE schemes using similar parameters.

We used Visual Studio code version 1.74.0 IDE, installed on Dell Spectre x360 Convertible laptop with Intel Core i7-1065G7 CPU/1.30GHz 1.50 GHz processor and 16 GB memory, running Microsoft Windows 10 version 22H2 (OS built 190452251). We ran the C language implementations of PAVTASP with six other sponge-based schemes NORX *[62]*, PSASPIN *[37]*, ASCON *[56]*, *ISAP[59]*, π–CIPHER[63], and SPOOK [78]. We recorded the performance metrics and compared the performance results, as shown in Table 2. We used the framework by Dobraunig et al. [57], enabling the GCC compiler flags: -O3 -march=native -Wall for compile-time optimization. The performance metrics are presented in cycle per byte (cpb), following the approach of NORX [56, 62], ASCON [56], and ISAP [59].

Table *2*. Comparison of PAVTASP against other AE schemes (in cycles per byte)

| **Message size** / **Scheme** | **1** | **8** | **16** | **32** | **64** | **1536** | **32768** |
|---|---|---|---|---|---|---|---|
| ISAP | 2842 | 368 | 192 | 104 | 59 | 16.5 | 14.8 |
| π–CIPHER | 2178 | 272 | 136 | 68 | 42.1 | 9.8 | 8.4 |
| NORX | 1065 | 133 | 67 | 34 | 16.6 | 3.2 | 2.5 |
| ASCON | 174 | 20 | 15 | 9 | 6.2 | 3.3 | 3.1 |
| **PAVTASP** | **71** | **9** | **8** | **6** | **4.5** | **3.4** | **3.7** |
| PSASPIN | 56 | 8 | 12 | 8 | 5.3 | 3.1 | 3 |
| SPOOK | 29 | 4 | 2 | 1 | 0.5 | 0.1 | 0.1 |

Table 2 compares the performance of PAVTASP against other similar AE schemes based on sponge construction. The entries in the table show the number of cycles per byte (cpb) when processing messages of different lengths: from 1 byte, 8 bytes, and up to 32768 bytes—the higher the cpb, the more efficient the scheme.

The results demonstrate that ISAP and π–CIPHER outperform PAVTASP in every message length. Although ISAP provides protection against SCA, it lacks many other features that PAVTASP has, e.g., parallelizability, incremental, single pass, online, and variable stretch (refer to Table 1). On the other hand, π–CIPHER does not offer protection against SCAs and lacks the support of variable stretch features.

Furthermore, PAVTASP generally outperforms PSASPIN and SPOOK in processing any message length. For example, PAVTASP gives 71 cpb when encrypting one byte, whereas PSASPIN requires 56 cpb and SPOOK 29 cpb. Note that the only difference between PSASPIN and PAVTASP is that the former does not provide a variable stretch tag feature, whereas the latter does; in addition, PAVTASP does not support the nonce-hiding features. Therefore, PAVTASP has a slight advantage over PSASPIN in performance.

NORX and ASCON seem to excel in processing short messages. However, these two schemes are not protected against SCA and do not have the same number of features that PAVTASP has. As in any cryptographic scheme, there are tradeoffs between security and performance. Generally, an AE scheme that provides more security features is not necessarily the most efficient. Therefore, applications that require the set of features offered by PAVTASP would likely benefit from using it. Furthermore, as far as we know, PAVTASP is the only sponge-based AE scheme that allows the use of variable tag length under the same key without compromising other critical security and performance features.

## VII. DISCUSSION

The cryptographic sponge function was first proposed by Bertoni *et al.* [54] and gained popularity after NIST declared Keccak as the SHA3 competition winner in October 2012 [79]. Sponge-based variants like Duplex, MonkeyDuplex, SpongeWrap, and DonkeySponge [27, 55] and their innovative design philosophies have eliminated the complications of key scheduling in other constructions like block ciphers. Our sponge-based scheme, PAVTASP, protects against SPAs and DPAs in addition to being parallel.

The first parallelizable AE construction based on the duplex mode of the sponge function was presented by Morawiecki et al. [64] and, then other works followed like those in [30, 62, 63]. However, the main problem with these early efforts was that the resulting schemes did not protect against side-channel attacks, especially against SPAs and DPAs [18, 50].

Furthermore, ISAP [38], SALE [38], and SPOOK [65] sponge-based AE schemes were proposed to defend against SPA and DPA attacks using different approaches. For example, some AE schemes [38, 66] used sponge-based constructions to defend against SCA, followed by [65], who based their scheme on a tweakable block cipher. These studies used the leveled implementation approach [18, 20]. But these constructions lacked parallelizability, which is an essential performance feature. The scheme proposed in this work combines the merits of parallelizability, and protection against SCAs particularly against SPAs and DPAs.

Several countermeasures to protect against SCAs, like hiding, shuffling [18], and masking [15, 16], but fresh rekeying achieves the same goal less resource-intensively. Abdalla &

Bellare [20] first proposed fresh rekeying. This method uses the master key $K$ as an input to a pseudorandom generator that generates session keys which are then used to preserve systems' privacy and integrity; thus, it does not use the master key directly in the schemes. Furthermore, fresh re-keying increases the key lifetime, the number of times a specific key can be used to encrypt messages before needing to be changed. Abdalla and Bellare first suggested a rekeying scheme in a leveled implementation fashion [20]. They stated that the rekeying component should be protected against both SPA and DPA but itself does not need to be cryptographically strong . The main construction (the rekeyed part), on the other hand, must be protected only against SPA, but must be cryptographically strong.

Medwed *et al*. [19, 71] developed a rekeying scheme following the leveled implementation approach using the AES block cipher for the base scheme component and a PRF based on modular multiplication GF($2^8$) for the rekeying part. However, their scheme is susceptible to attacks suggested by Black *et al*. in [80], because of their way of intermediate processing states of the block cipher, as mentioned by Dobraunig, Koeune [81]. Our work uses a rekeying function on the GF($2^8$) multiplication field that is protected against by a combination of masking and hiding to defend it against higher-order DPAs.

On the other side, the leveled implementation of SALE [66] and ISAP [38] used sponge-based structures in both levels of rekeying component, and the core rekeyed data processing component. Although using the same primitive for the two levels is preferable for reducing the code size of the scheme, the possibility of enabling CPAs may lead to compromising the subsequent keys [53]. Other works, notably Spook [65], employed a tiered implementation in which the data processing component is based on a sponge construction (T-Sponge) and the rekeying generation on a tweakable block cipher. However, a Galois field multiplication GF($2^8$) based algebraic construct is lighter and easier to protect against SPAs and DPAs. PSASPIN [37] proposed a parallel, sponge-based AE scheme with SPA and DPA protection and hidden nonces from adversaries. Still, it assumed the tag length is a fixed parameter under the same key and thus might be vulnerable to tag length variation attacks. Remember that these AE schemes, except PSASPIN, are serial and do not provide parallelism, which is a crucial property for AE schemes. Our scheme in this study, PAVTASP, allows the use of variable stretch while using the same key, in addition to parallelizability and protections against SCAs. No other sponge-based AE scheme provides the combination of those properties to the best of the author's knowledge.

Most AE schemes, including [6, 7, 27, 41-44], consider the stretch a fixed scheme parameter per key, and the security is proved accordingly, assuming that different stretch values use distinct keys. However, using variable tag lengths under the same key could happen either because of misconfiguration or attack, and the security would be violated[39]. In addition to its security relevance, tag length variability is desirable in constrained resource environments. Still, negotiating parameters cost is preventively high due to resource limitations, according to Struik [13].

Reyhanitabar et al. (2017) discussed the issue in detail and formalized a security notion for the nonce-based AE schemes vNBAE. Furthermore, the authors proposed a modular approach for defining the key-equivalent separation by stretch (KESS) concept, which, combined with the traditional NBAE implies the vNBAE security notion. Finally, the authors proved that the vNBAE goal was efficient and provably achievable, concretizing it with the modification of OCB without sacrificing its desirable features, such as the online processing of data blocks [8]. Finally, the authors outlined some open problems indicating possible ways to extend their work, including the possibility of describing transformations that apply to large subsets of NBAE secure schemes encoding the stretch value $\tau$ in the input of sponge-based modes. This work fills the gap by proposing and implementing a sponge-based AE scheme that encodes the stretch value $\tau$ in the encryption and decryption processes to allow the secure use of variable tag lengths under the same key.

There should be an upper limit on the amount of plaintext and associated data blocks protected by calls to the authenticated encryption function over the key lifetime. According to NIST, Special Publication 800-38D [40], $2^{64}$ would be a fair upper limit for the majority of applications. Jovanovic, Luykx [75] set sponge-based schemes' integrity bound (tag length) to $2^{c/2,}$ where c is the capacity. As soon as the lower bound limit approaches, a key exchange should be negotiated to the security to hold. Abdalla and Bellare [20] and Mennink [15] stated that with fresh rekeying, the security bounds of schemes could be enhanced, for instance, from $2^{k/2}$ to $2^{k/3}$. Our work, PAVTASP, enhances the minimum recommended tag length of $2^{64}$ for the equal key length and capacity of 128 bits to $2^{128/3}$ or approximately $2^{42}$ bits. This fact is supported by the fact that fresh rekeying can enhance to increase the traditional bounder of security limit.

This work is inspired by ISAP [38] and PSASPIN[37] but differs from ISAP in two ways: first, the proposed scheme PAVTASP is parallelizable. Second, it follows a different implementation approach for key generation and data processing to protect against the weaknesses indicated in [53]. Our implementation consisted of two layers, and The rekeying layer is based on Galois Field multiplication using a PRF, following the design proposed by Medwed, Standaert [19], Medwed, Petit [71]. Moreover, the related key attack concern raced by Dobraunig *et al*. [81] is not relevant in the case of the sponge-based schemes because those attacks exploit the partial key processing values of key scheduling, which does not exist for the sponge functions.

Finally, PAVTASP differs from PSASPIN because it permits using variable tag lengths under the same key in a secure manner, protecting against misuse attacks related to instances of the same AE schemes using different stretch values under the same secret key. PAVTASP follows the KESS approach proposed by Reyhanitabar et al.(2017) to achieve this goal. Thus PAVTASP has a different syntax that adapts the tag length (stretch) value as an input parameter in encryption and decryption processes. Furthermore, PAVTASP performs better than PSASPIN after testing their implementation in the C programming language.

## III. CONCLUSION

This paper proposed and implemented a side-channel attack-resistant sponge-based, parallel AE scheme that permits using variable tag lengths under the same key, PAVTASP. Our implementation consisted of two layers. The rekeying layer is based on Galois Field multiplication, while the base scheme layer is based on the sponge construction in the duplex mode. The proposed scheme is advantageous over similar sponge-based AE schemes because it allows variable tag lengths under the same key without sacrificing other valuable features like online and parallelizability. Finally, the security of the proposed scheme is evaluated, and its performance is analyzed and compared to similar AE schemes after implementing it in the C programming language.

## ACKNOWLEDGMENT

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

## REFERENCES

1. Bellare, M. and C. Namprempre. *Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm*. in *ASIACRYPT*. 2000. Berlin, Heidelberg: Springer Berlin Heidelberg.
2. Katz, J. and M. Yung. *Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation*. in *FSE*. 2001. Berlin, Heidelberg: Springer Berlin Heidelberg.
3. Hawkes, P. and G.G. Rose, *A Mode of Operation with Partial Encryption and Message Integrity*. Cryptology ePrint, 2003.
4. Jonsson, J. *On the Security of CTR + CBC-MAC*. in *Selected Areas in Cryptography(SAC)*. 2003. Berlin, Heidelberg: Springer Berlin Heidelberg.
5. Riou, S. *DryGASCON, Lightweight Cryptography Standardization Process round 1 submission*. Submission to NIST 2019 17/01/2021]; Available from: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/drygascon-spec-round2.pdf.
6. Bellare, M. and C. Namprempre. *Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm*. in *ASIACRYPT*. 2000. Berlin, Heidelberg: Springer Berlin Heidelberg.
7. Rogaway, P., *Authenticated-encryption with associated-data*, in *Proceedings of the 9th ACM conference on Computer and communications security*. 2002, Association for Computing Machinery: Washington, DC, USA. p. 98–107.
8. Reyhanitabar, R., S. Vaudenay, and D. Vizár. *Authenticated Encryption with Variable Stretch*. in *Advances in Cryptology – ASIACRYPT 2016*. 2016. Berlin, Heidelberg: Springer Berlin Heidelberg.
9. Manger, J.H. *[Cfrg] Attacker changing tag length in OCB*. 2013 11/07/2022]; Available from: https://mailarchive.ietf.org/arch/msg/cfrg/8gF2RgsjjK5iE8olDihu8SNRNt4/.
10. Rogaway, P. *[Cfrg] Attacker changing tag length in OCB*. 2013 07/11/2022]; Available from: https://mailarchive.ietf.org/arch/msg/cfrg/8gF2RgsjjK5iE8olDihu8SNRNt4/.
11. Wagner, P.R.a.D., *A Critique of CCM*. Cryptology ePrint Archive, 2003.
12. Iwata, T. *CLOC and SILC will be tweaked*. 2015 [cited 2022 07/11/2022]; Available from: https://competitions.cr.yp.to/round3/clocsilcv3.pdf.
13. Struik, R., *AEAD Ciphers for Highly Constrained Networks*, in *DIAC 2013*. 2013.
14. Dobraunig, C., M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer. *ISAP v2.0*. Submission to the NIST LWC Competition R2 2019 10/03/2021]; Available from: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/isap-spec-round2.pdf.
15. Mennink, B., *Beyond Birthday Bound Secure Fresh Rekeying: Application to Authenticated Encryption*. Cryptology ePrint, 2020. **AP**.
16. Duc, A., S. Faust, and F.-X. Standaert. *Making Masking Security Proofs Concrete*. in *EUROCRYPT* 2015. Berlin, Heidelberg: Springer Berlin Heidelberg.
17. Ishai, Y., A. Sahai, and D. Wagner. *Private Circuits: Securing Hardware against Probing Attacks*. in *CRYPTO* 2003. Berlin, Heidelberg: Springer Berlin Heidelberg.
18. Mangard, S., E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Power Analysis Attacks: Revealing the Secrets of Smart Cards. 2007, Boston, MA: Springer US.
19. Medwed, M., F.-X. Standaert, J. Großschädl, and F. Regazzoni. *Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices*. in *AFRICACRYPT* 2010. Berlin, Heidelberg: Springer Berlin Heidelberg.
20. Abdalla, M. and M. Bellare. *Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques*. in *ASIACRYPT* 2000. Berlin, Heidelberg: Springer Berlin Heidelberg.
21. Daemen, J. and V. Rijmen, *The Design of Rijndael. AES — The Advanced Encryption Standard*. 2002: Springer.
22. Beierle, C., 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 *CRYPTO* 2016. Berlin, Heidelberg: Springer Berlin Heidelberg.
23. Banik, S., S.K. Pandey, T. Peyrin, Y. Sasaki, S.M. Sim, and Y. Todo. *GIFT: A Small Present*. in *Cryptographic Hardware and Embedded Systems (CHES)*. 2017. Cham: Springer International Publishing.
24. Tahir, R., M.Y. Javed, and A.R. Cheema. *Rabbit-MAC: Lightweight Authenticated Encryption in Wireless Sensor Networks*. in *2008 International Conference on Information and Automation*. 2008.
25. Hoang, V.T., T. Krovetz, and P. Rogaway, *Robust Authenticated-Encryption: AEZ and the Problem that it Solves*. Cryptology ePrint Archive 2014.
26. Alizadeh, J., M.R. Aref, and N. Bagheri, *JHAE: A Novel Permutation-Based Authenticated Encryption Mode Based on the Hash Mode JH*. Cryptology ePrint Archive 2014.
27. Assche, G.B.a.J.D.a.M.P.a.G.V., *Duplexing the sponge: single-pass authenticated encryption and other applications*. Cryptology ePrint Archive, 2011.
28. Cogliani, S., D.-Ş. Maimu, D. Naccache, R.P.d. Canto, R. Reyhanitabar, S. Vaudenay, and D. Vizár. *Offset Merkle-Damgård (OMD) version 1.0*. Submission to CAESAR R1 2014 12/08/2020]; Available from: http://competitions.cr.yp.to/round1/omdv10.pdf.
29. Peña, P.I.S. and R.E.G. Torres. *Authenticated Encryption based on finite automata cryptosystems*. in *2016 13th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. 2016.
30. Andreeva, E., B. Bilgin, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha, and K. Yasuda, *APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography*. Cryptology ePrint Archive 2013.
31. Xin, L., M. Zhi, and F. Deng-Guo. *A quantum authenticated encryption scheme*. in *Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04. 2004*. 2004.
32. Iwata, T., K. Minematsu, J. Guo, S. Morioka, and E. Kobayashi. *CLOC and SILC*. Submission to CAESAR R3 2016 03/02/2021]; Available from: http://competitions.cr.yp.to/round3/clocsilcv3.pdf.

33. Jimale, M., M. Zaba, M.L. Mat Kiah, M. Idris, N. Jamil, M. Mohamad, and M. Rohmad, *Authenticated Encryption Schemes: A Systematic Review*. IEEE Access, 2022: p. 1-1.
34. Bellare, M., A. Boldyreva, L. Knudsen, and C. Namprempre. *Online Ciphers and the Hash-CBC Construction*. in *CRYPTO* 2001. Berlin, Heidelberg: Springer Berlin Heidelberg.
35. Sasaki, Y. and K. Yasuda. *A New Mode of Operation for Incremental Authenticated Encryption with Associated Data*. in *SAC* 2016. Cham: Springer International Publishing.
36. Boorghany, A., S. Bayat-Sarmadi, and R. Jalili, *Efficient Lattice-based Authenticated Encryption: A Practice-Oriented Provable Security Approach*. Cryptology ePrint Archive 2016.
37. Jimale, M.A., Z. M. R, x, aba, M.L.B.M. Kiah, M.Y.I. Idris, N. Jamil, M.S. Mohamad, and M.S. Rohmad, *Parallel Sponge-Based Authenticated Encryption With Side-Channel Protection and Adversary-Invisible Nonces*. IEEE Access, 2022. **10**: p. 50819-50838.
38. Dobraunig, C., M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer. *ISAP v2.0*. Submission to the NIST LWC Competition R2 2019; Available from: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/isap-spec-round2.pdf.
39. Reyhanitabar, R., S. Vaudenay, and D. Vizár. *Authenticated Encryption with Variable Stretch*. 2016. Berlin, Heidelberg: Springer Berlin Heidelberg.
40. Dworkin, M., *Recommendation for Block Cipher Modes of Operation- Galois/Counter Mode (GCM) and GMAC*. 2007, NIST.
41. Bellare, M., P. Rogaway, and D. Wagner, *A Conventional Authenticated-Encryption Mode*. 2003.
42. Rogaway, P. and T. Shrimpton, *Deterministic Authenticated-Encryption: A Provable-Security Treatment of the Key-Wrap Problem*. Cryptology ePrint Archive, 2006.
43. Kandele, S. and S. Paul, *Key Assignment Scheme with Authenticated Encryption*. Cryptology ePrint Archive 2018.
44. Bao, Z., J. Guo, T. Iwata, and K. Minematsu, *ZOCB and ZOTR: Tweakable Blockcipher Modes for Authenticated Encryption with Full Absorption*. Cryptology ePrint Archive 2019.
45. Bernstein, D.J. *Cryptographic competitions: Disasters*. 2014; Available from: https://competitions.cr.yp.to/disasters.html.
46. Bernstein, D. *CAESAR call for submissions, final (2014.01.27)*. 2013 06/12/2019]; Available from: http://competitions.cr.yp.to/caesar-call.html.
47. NIST. *Lightweight Cryptography*. 2020 04/12/2020]; Available from: https://www.nist.gov/programs-projects/lightweight-cryptography.
48. Minematsu, K. *AES-OTR v3.1*. Submission to CAESAR R3 2016 14/01/2021]; Available from: http://competitions.cr.yp.to/round1/aesotrv1.pdf.
49. Reyhanitabar, R., S. Vaudenay, and D. Vizár. *Misuse-Resistant Variants of the OMD Authenticated Encryption Mode*. 2014. Cham: Springer International Publishing.
50. Popp, T., *An introduction to implementation attacks and countermeasures*, in *Proceedings of the 7th IEEE/ACM international conference on Formal Methods and Models for Codesign*. 2009, IEEE Press: Cambridge, Massachusetts. p. 108–115.
51. Prouff, E. and M. Rivain. *Masking against Side-Channel Attacks: A Formal Security Proof*. in *EUROCRYPT* 2013. Berlin, Heidelberg: Springer Berlin Heidelberg.
52. Krämer, J. and P. Struck, *Leakage-Resilient Authenticated Encryption from Leakage-Resilient Pseudorandom Functions*. Cryptology ePrint, 2020. **AP**.
53. IRTF, I.R.T.F., *Re-keying Mechanisms for Symmetric Keys*. 2019, IETF.
54. Bertoni, G., J. Daemen, M. Peeters, and G.V. ASSCHE. *Cryptographic sponge functions*. 2011; Available from: https://keccak.team/files/CSF-0.1.pdf.
55. Bertoni, G., J. Daemen, M. Peeters, and G.V. Assche. *Permutation-based encryption, authentication and authenticated encryption*. 2012 29/04/2021]; Available from: https://keccak.team/files/KeccakDIAC2012.pdf.
56. Dobraunig, C., M. Eichlseder, F. Mendel, and M. Schläffer. *Ascon v2*. Submission to CAESAR R3 2016 03/02/2021]; Available from: http://competitions.cr.yp.to/round1/asconv1.pdf.
57. Dobraunig, C., M. Eichlseder, F. Mendel, and M. Schläffer. *Ascon v1.2*. Submission to the NIST LWC Competition R2 2019 10/03/2021].
58. Beyne, T., Y.L. Chen, C. Dobraunig, and B. Mennink. *Elephant v1.1*. Submission to the NIST LWC Competition R2 2019 10/03/2021]; Available from: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/elephant-spec-round2.pdf.
59. Dobraunig, C., M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer. *ISAP v2.0*. Submission to the NIST LWC Competition R2 2019 [cited 10/03/2021; Available from: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/isap-spec-round2.pdf.
60. Bao, Z., A. Chakraborti, N. Datta, J. Guo, M. Nandi, T. Peyrin, and K. Yasuda. *PHOTON-Beetle Authenticated Encryption and Hash Family*. Submission to the NIST LWC Competition R2 2019 10/03/2021]; Available from: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/photon-beetle-spec-round2.pdf.
61. Daemen, J., S. Hoffert, M. Peeters, G.V. Assche, and R.V. Keer. *Xoodyak, a lightweight cryptographic scheme*. Submission to the NIST LWC Competition R2 2019 10/03/2021]; Available from: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Xoodyak-spec-round2.pdf.
62. Aumasson, J.-P., P. Jovanovic, and S. Neves. *NORX v3*. Submission to CAESAR R3 2016 03/02/2021]; Available from: http://competitions.cr.yp.to/round1/norxv1.pdf.
63. Gligoroski, D., H. Mihajloska, S. Samardjiska, H. Jacobsen, M. El-Hadedy, and R.E. Jensen. *π–Cipher v11*. Submission to CAESAR R1 2014 12/08/2020]; Available from: http://competitions.cr.yp.to/round1/piciphervl.pdf.
64. Morawiecki, P. and J. Pieprzyk, *Parallel authenticated encryption with the duplex construction*. Cryptology ePrint Archive, 2013.
65. Bellizia, D., 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. *Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher*. Submission to the NIST LWC Competition R2 2019; Available from: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Spook-spec-round2.pdf.
66. Degabriele, J.P., C. Janson, and P. Struck, *Sponges Resist Leakage: The Case of Authenticated Encryption*. Cryptology ePrint Archive 2019.
67. Rogaway, P., M. Bellare, J. Black, and T. Krovetz, *OCB: a block-cipher mode of operation for efficient authenticated encryption*, in *Proceedings of the 8th ACM conference on Computer and Communications Security*. 2001, ACM: Philadelphia, PA, USA. p. 196-205.
68. Gligor, V.D. and P. Donescu. *Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes*. in *FSE*. 2002. Berlin, Heidelberg: Springer Berlin Heidelberg.
69. Whiting, D., R. Housley, and N. Ferguson, *Counter with CBC-MAC (CCM)*. 2003: RFC Editor.
70. Lee, B., *The Key Distribution Pr y Distribution Problem: Prior Adv oblem: Prior Advances and F ances and Future Challenges*, in *Trinity College Digital Repository* 2020.
71. Medwed, M., C. Petit, F. Regazzoni, M. Renauld, and F.-X. Standaert. *Fresh Re-keying II: Securing Multiple Parties against Side-Channel and Fault Attacks*. in *CARDIS*. 2011. Berlin, Heidelberg: Springer Berlin Heidelberg.
72. Pereira, O., F.-X. Standaert, and S. Vivek, *Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives*, in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, Association for Computing Machinery: Denver, Colorado, USA. p. 96–108.
73. Barwell, G., D.P. Martin, E. Oswald, and M. Stam, *Authenticated Encryption in the Face of Protocol and Side Channel Leakage*. Cryptology ePrint Archive 2017.
74. Abdalla, M., S. Belaïd, and P.-A. Fouque. *Leakage-Resilient Symmetric Encryption via Re-keying*. in *Cryptographic Hardware and*

*Embedded Systems - CHES*. 2013. Berlin, Heidelberg: Springer Berlin Heidelberg.

75. Jovanovic, P., A. Luykx, and B. Mennink, *Beyond 2c/2 Security in Sponge-Based Authenticated Encryption Modes.* Cryptology ePrint Archive 2014.
76. Mihajloska, H., B. Mennink, and D. Gligoroski. *π-Cipher with Intermediate Tags*. 2016 01/09/2021]; Available from: http://pi-cipher.org/upload/Pi-Cipher%20with%20intermediate%20tags.pdf.
77. Do, Q., B. Martini, and K.-K.R. Choo, *The role of the adversary model in applied security research.* Computers & Security, 2019. **81**: p. 156-181.
78. Bellizia, D., 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. *Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher*. Submission to the NIST LWC Competition R2 2019 21/02/2021]; Available from: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Spook-spec-round2.pdf.
79. *SHA zoo*. 2013 21/09/2021]; Available from: https://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo.
80. Black, J., P. Rogaway, T. Shrimpton, and M. Stam, *An Analysis of the Blockcipher-Based Hash Functions from PGV.* Journal of Cryptology, 2010. **23**(4): p. 519-545.
81. Dobraunig, C., 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*. 2016. CHAM: Springer International Publishing.

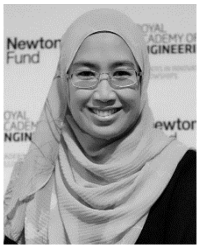
**MOHAMUD AHMED JIMALE** Earned his bachelor's degree in Information Technology from SIMAD University, Mogadishu, Somalia, his master's degree in computer science from the University Of Malaya, and is currently pursuing his Ph.D. at the University Of Malaya, Malaysia. His research interests include Cryptography, Blockchain, Cloud computing, and the Internet of Everything (IoE). He held different academic and administrative positions, including the founding present of Jamhuriya University of Science and Technology (JUST) and the Deputy head of the Information Technology department of Dahabshil International Co. in Somalia.

**NOR ANIZA ABDULLAH** is an Associate Professor at the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. She has a bachelor's degree in computer science from University Malaya, and a Master's degree in interactive multimedia from Westminster University, London. Her PhD degree is in computer science from Southampton University, United Kingdom. Her research interests include personalized learning, adaptive multimedia, recommender systems, machine learning, image processing and mobile crowdsourcing..

**MISS LAIHA BINTI MAT KIAH** received her PhD degree in Information Security from Royal Holloway, University of London, United Kingdom in 2007, and since then she is an active researcher at Faculty of Computer Science & Information Technology, UM in her Computer Science field particularly in security. She was promoted to Professorship in 2015, and is an active member of IEEE as Senior Member, EC Council, Malaysian Society for Cryptology Research (MSCR) and Malaysia Board of Technologists (Ts.). Her main research interest will always be in the 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. Current research interests include Cyber Security, Blockchain Technology, IoT and Health Information Exchange.

**MUHAMMAD REZA Z'ABA** is a Staff Researcher at MIMOS Berhad, which is a research arm under the purview of the Ministry of Science and Technology, Malaysia. He was previously a Senior Lecturer at the Faculty of Computer Science and Information Technology, Universiti Malaya. Prior to that, he was a researcher at MIMOS Berhad. He received his Bachelor of Science (Computer) in 2004 from Universiti Teknologi Malaysia (UTM) and PhD from Queensland University of Technology, Australia in 2010. His main research interests are in the area of symmetric cryptography (authenticated encryption, block cipher and hash function). He is also interested in digital identity, blockchain-related technologies and other areas of information security.

**MOHD YAMANI IDNA IDRIS** Received the B.E. and Ph.D. degrees in Electrical Engineering and M.Sc. in Computer Science 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, Malaysia. He has published many articles in reputable journals, and has received many awards for his inventions. His research interests include Internet of Things (IoT), Information Security, Embedded System, Image Processing and Computer Vision, and Wireless Sensor Network.


**NORZIANA JAMIL** Received Her PhD in Security in Computing in 2013. She is now an Associate Professor at the Universiti Tenaga Nasional, Malaysia. Her area of research specialization and interest includes Cryptography, Security for Cyber-Physical Systems, security analytics and intelligent system. She is an alumni of Leadership in Innovation Fellowship by UK 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.

**MOHD SAUFY ROHMAD** is the Head of Robotic, IoT and Big Data in Smart Manufacturing Research Institute, UiTM Shah Alam. He is a full time Senior Lecturer in College of Engineering, UiTM Shah Alam. He was previously a Researcher in Telekom Malaysia R&D and MIMOS Berhad. He also was a SoC Design Engineering in Intel Penang. He received his first degree from Universiti Teknologi PETRONAS in Information Technology and Master of Science in Computer Science in UiTM. Now he is finishing his PhD in Embedded Cryptography in UiTM. He also involved in a few Industrial IoT projects for livestock and agriculture.