IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2020, No. S1, pp. 390–416. DOI:10.13154/tosc.v2020.iS1.390-416

Isap v2.0

Christoph Dobraunig[1,2], Maria Eichlseder[2], Stefan Mangard[2], Florian Mendel[3], Bart Mennink[1], Robert Primas[2] and Thomas Unterluggauer[2]

[1] Digital Security Group, Radboud University, Nijmegen, The Netherlands
[2] Graz University of Technology, Graz, Austria
[3] Infineon Technologies AG, Neubiberg, Germany
https://isap.iaik.tugraz.at
isap@iaik.tugraz.at

Abstract. We specify Isap v2.0, a lightweight permutation-based authenticated encryption algorithm that is designed to ease protection against side-channel and fault attacks. This design is an improved version of the previously published Isap v1.0, and offers increased protection against implementation attacks as well as more efficient implementations. Isap v2.0 is a candidate in NIST's LightWeight Cryptography (LWC) project, which aims to identify and standardize authenticated ciphers that are well-suited for applications in constrained environments. We provide a self-contained specification of the new Isap v2.0 mode and discuss its design rationale. We formally prove the security of the Isap v2.0 mode in the leakage-resilient setting. Finally, in an extensive implementation overview, we show that Isap v2.0 can be implemented securely with very low area requirements.

Keywords: Authenticated encryption · NIST LWC · Leakage resilience · Sponges

1 Introduction

Ever since the publication of side-channel and fault attacks [Koc96, KJJ99, BDL97, BS97] it has become evident that implementations of cryptographic schemes cannot be considered as a black box, especially in scenarios where an attacker has physical access to the device performing a cryptographic task. However, restricting the access to devices performing cryptographic tasks provides a considerable limitation on the applications in which cryptography can be used at all. As a consequence, shortly after the introduction of side-channel and fault attacks, countermeasures that harden the implementations of cryptographic primitives, such as masking [GP99, CJRR99], have been introduced.

However, cryptographic primitives like the AES [DR02] as well as ARX-based primitives [Ber08, Nat15a] turned out to be costly to protect against implementation attacks, especially considering (higher-order) masking against (higher-order) side-channel attacks. As a consequence, primitives have been introduced that allow for more efficient masking, such as the blockciphers Noekeon [DPVR00], PICARO [PRC12], Zorro [GGNPS13], Robin, or Fantomas [GLSV14] or the permutations used in Ascon [DEMS16], Keccak [BDPV11, Nat15b], or Xoodoo [DHVV18]. Likewise, dedicated modes for symmetric encryption have been introduced that reduce the requirements for countermeasures on the primitive level for protection against side-channel attacks. These modes typically fall in the categories of leakage-resilient cryptography [DP08] or fresh re-keying [MSGR10].

Inspired by these research directions, Isap v1.0 was designed and published at ToSC 2017 [DEM+17], introducing an authenticated encryption scheme focusing on the protection against side-channel attacks. Isap v1.0 is an encrypt-then-MAC [BN00, KJJR11] scheme utilizing the sponge construction [BDPV07, BDPV08]. It combines a sponge-based stream

Received: 2019-12-10, Accepted: 2020-02-15, Published: 2020-06-22

cipher with a suffix keyed sponge acting as MAC in a specific way to provide resistance against side-channel attacks. In particular, both parts derive session keys in a GGM-tree-like [GGM86] manner (similar to [TS14]) in order to harden this key derivation against side-channel attacks. All ingredients combine to a nonce-based authenticated encryption scheme that provides protection against (higher-order) differential power analysis without the need for (higher-order) masking.

1.1 Contributions

In this paper, we present the improved version ISAP v2.0. Its specification is given in Section 2. ISAP v2.0 retains ISAP v1.0's outstanding properties with respect to side-channel protection, but the mode differs subtly from ISAP v1.0 so as to achieve increased protection against other implementation attacks. Additionally, ISAP v2.0 introduces new, more efficient instantiations. The rationale of ISAP v2.0 and its improvements over ISAP v1.0 are detailed in Section 3. In this paper, unless stated otherwise, ISAP always refers to ISAP v2.0.

The main design goal of ISAP v2.0 is to provide out-of-the-box robustness against certain types of implementation attacks while allowing to add additional defense mechanisms at low cost. This is essential whenever cryptographic devices are deployed in locations that are physically accessible by potential attackers – a typical scenario in IoT (Internet of Things) applications. Secure software and firmware updates on such devices in particular are both crucial and challenging.

The ISAP mode of operation can be instantiated with any suitable permutation. We propose four instantiations of ISAP v2.0 (see also Subsection 2.2): two based on the 400-bit permutation KECCAK-p[400] [BDPV11, Nat15b], and two based on the 320-bit permutation used in ASCON [DEMS16, DEMS19], which has recently been selected as first choice for the use case of lightweight applications (resource constrained environments) in the final CAESAR portfolio [CAE14]. The security claims corresponding to these four instantiations are summarized in Subsection 2.3. Note that by implementing either of the two permutations, other cryptographic functionalities can be realized with minimal implementation overhead, including hashing [BDPV11, DEMS19].

Whereas ISAP v1.0 was published without a formal security proof, we complement this paper with a proof of security in the leakage-resilient setting for ISAP v2.0. Although the robustness of the ISAP mode against implementation attacks is rather intuitive (see Section 3), formally proving so turns out to be subtle. The reason for this is that ISAP is built of a sponge-based stream cipher with a suffix-keyed sponge, both of which are, from a generic perspective, modes with structurally different properties. The leakage resilience of these two components has recently been investigated by Dobraunig and Mennink [DM19a, DM20]. In Section 4, we show how these two disjoint leakage resilience results seamlessly fuse together to leakage resilience of the ISAP mode.[1]

In Section 5, we provide an extensive implementation overview for all instances of ISAP. For instance, we demonstrate that one can implement the ASCON-based instance ISAP-A-128A in software to process long messages with up to 21.9 cycles/byte on modern desktop CPUs. We furthermore investigate the cost of hardware implementations, and demonstrate, e.g., that protected implementations are possible with area below 14 kGE. We finalize Section 5 by discussing various aspects of implementation security and the possibility of online implementations.

1.2 Novelty Compared with Previously Published Work

We briefly summarize the novelty of this work compared to previous related publications.

[1] On a related note, Guo et al. [GPPS19] independently constructed a security argument for ISAP.

- Specification: Isap v1.0 was first published in [DEM+17]. Isap v2.0 updates this specification in numerous aspects in order to improve the protection against a wider range of implementation attacks including fault attacks, as we summarize in Section 3. Isap v2.0 is a candidate in the NIST LightWeight Cryptography (LWC) project [DEM+19], but was not published elsewhere.
- Security proof: Isap v1.0 was published without a formal proof. In this paper, we prove the security of Isap v2.0 in a leakage-resilient setting. A preliminary version of the proof given in Section 4 has appeared as a workshop record without proceedings [DM19b].
- Implementation: Section 5 includes new implementation results. We created optimized implementations for all Isap instantiations and for various platforms ranging from high-end 64-bit CPUs to low-end 32-bit microprocessors. We also discuss the implications of implementing Isap in an online instead of a two-pass fashion.

1.3 Related Work

The area of leakage resilient cryptography [DP08] popularized the idea of designing modes of operation that provide some resilience against side-channel attacks. For a long time, the majority of leakage-resilient constructions were based on (tweakable) block ciphers [Pie09, DP10, YSPY10, FPS12, MSJ12, SPY13, PSV15, MSNF16, BPPS17, BGP+19, GSWY20]. However, recently, the focus of leakage-resilient cryptography also started to include permutation-based constructions [DM19a, DM20, GPPS19]. A direction with a similar goal is fresh re-keying [MSGR10], which brought forward many schemes [MPR+11, BDH+14, DKM+15, DFH+16] and served as motivation to start designing Isap v1.0 [DEM+17]. Around the publication of Isap v1.0, several other papers appeared that also focus on authenticated encryption that provides increased resistance against side-channel attacks [BPPS17, BMOS17].

Clearly, the concept of authenticated encryption predates the constructions mentioned before. The first concepts that provided authentication and encryption where typically so-called generic compositions that combine an encryption scheme and a message authentication code (MAC) in one way or another [BN00, Kra01]. More efficient constructions that provide authenticated encryption were introduced by Jutla [Jut01, Jut08]. Later, Rogaway [Rog02] introduced the notion of authenticated encryption with associated data. The research in authenticated encryption led to a competition called CAESAR [CAE14]. Recently, the final portfolio of CAESAR was announced, which includes the authenticated encryption schemes Ascon [DEMS16] and ACORN [Wu15] recommended for lightweight applications, AEGIS-128 [WP16] and OCB [KR16] for high-performance applications, and Deoxys-II [JNPS16] and COLM [ABD+16] for defense in depth.

2 Specification of ISAP

Isap is a family of permutation-based authenticated encryption schemes. The Isap instances are parameterized by the security parameter k, which defines the cryptographic security level of k bits, as well as a set of permutations with different round numbers. The authenticated encryption algorithm $\mathcal{E}$ gets as input a key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^k$, associated data $A \in \{0,1\}^*$, and a message $M \in \{0,1\}^*$. It outputs a ciphertext $C \in \{0,1\}^{|M|}$ and a tag $T \in \{0,1\}^k$, where $|M|$ denotes the bitlength of M. The decryption algorithm $\mathcal{D}$ takes K, N, A, C, and T, and returns either the message M or an error $\perp$.

2.1 Mode

Authenticated encryption $\mathcal{E}$ and authenticated decryption $\mathcal{D}$ are described in Algorithm 1 and 2. ISAP is an encrypt-then-MAC design, where the same k-bit key is used for encryption and message authentication. The encryption ISAPENC is specified in Algorithm 3 and the message authentication ISAPMAC in Algorithm 5. Both functions internally use a re-keying function ISAPRK, which is specified in Algorithm 4. The three functions are specified in more detail below.

In these algorithms, $p_{\mathrm{H}}, p_{\mathrm{B}}, p_{\mathrm{E}}, p_{\mathrm{K}}$ denote permutations updating an n-bit state S. Their instantiations are further elaborated on in Subsection 2.2.

The instances are further parameterized by two rate values r_{H} and r_{B}. The rate r defines how the state S is split into an r-bit outer part S_r and a c-bit inner part S_c as $S = S_r \parallel S_c = \lceil S \rceil^r \parallel \lfloor S \rfloor_c$, where $c = n - r$, $\parallel$ denotes concatenation of bitstrings, $\lceil S \rceil^r$ denotes the first (most significant) r bits of bitstring S, and $\lfloor S \rfloor_c$ denotes the last (least significant) c bits of bitstring S. Rate r_{H} is applied for states in the unkeyed sponge and in the keyed sponge that are unlikely to be evaluated more than once for different outer parts with a fixed inner part, which means that r_{H} may be reasonably large. Rate r_{B} is applied for states in the keyed sponge that may be evaluated more than once, which means that we must bound the amount of leakage by limiting the total number of evaluations that may be made for that state. In each of the members of ISAP, we set $r_{\mathrm{H}} = n - 2k$, $c_{\mathrm{H}} = 2k$ and $r_{\mathrm{B}} = 1$, $c_{\mathrm{B}} = n - 1$ (see also Table 1).

2.1.1 Re-Keying with IsapRK

The re-keying function ISAPRK is called by ISAPENC and ISAPMAC to generate session keys K_{E}^* and K_{A}^* to perform encryption and authentication, respectively. The function gets as input a k-bit key K, a k-bit string Y, a constant IV, and an output size z, where

$$(\mathrm{IV}, z) = \begin{cases} (\mathrm{IV}_{\mathrm{KE}}, n-k)\,, & \text{if called by ISAPENC,} \\ (\mathrm{IV}_{\mathrm{KA}}, k)\,, & \text{if called by ISAPMAC,} \end{cases}$$

and transforms these into a subkey K^* of size z bits. The function is described in Algorithm 4 and illustrated in Figure 1a. It is instantiated using permutations p_{K} and p_{B}: p_{K} is called in the beginning (to process the master key K) and at the end (to generate subkey K^*), and p_{B} is called for all intermediate duplexes using a very small rate r_{B}.

2.1.2 Encryption with IsapEnc

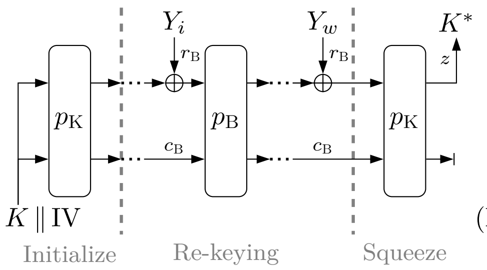
Encryption is performed by using the keyed sponge construction in streaming mode, with the notable difference that, first, ISAPRK is called to generate a subkey K_{E}^*. ISAPENC gets as input a k-bit key K, a k-bit nonce N, and an arbitrarily large message M, and generates a ciphertext C of size $|M|$. The function is described in Algorithm 3 and Figure 1b. It first calls ISAPRK for encryption using the constant initial value $\mathrm{IV} = \mathrm{IV}_{\mathrm{KE}}$ and $z = n - k$ in order to derive an $(n-k)$-bit subkey K_{E}^*. Once this subkey is generated, a regular sponge-based streaming mode using permutation p_{E} is evaluated at high rate r_{H}.

ISAPENC is a streaming mode, so decryption is identical with the roles of M, C swapped.

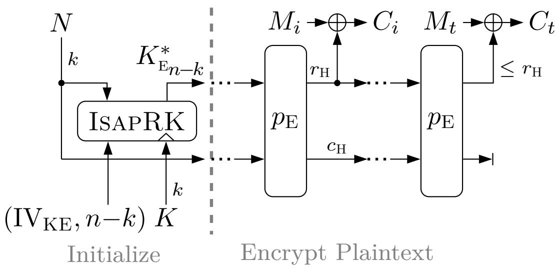
2.1.3 Authentication with IsapMAC

For message authentication, we use a sponge-based hash function to build a suffix-MAC. ISAPMAC gets as input a k-bit key K, a k-bit nonce N, arbitrarily large associated data A, and arbitrarily large ciphertext C, and it outputs a tag T of size k bits. The function is described in Algorithm 5 and Figure 1c. It starts by initializing the state as $N \parallel \mathrm{IV}_{\mathrm{A}}$ and absorbing the non-secret inputs (A, C) in plain sponge mode using permutation p_{H} with high rate r_H. Note that domain separation between A and C is performed using the XOR of a single bit '1' to the inner part of the state. The resulting state S is then split into a k-bit value $\lceil S \rceil^k$ and an $(n-k)$-bit value $\lfloor S \rfloor_{n-k}$. The value $\lceil S \rceil^k$ is fed as input string to IsapRK to generate a subkey K_A^*, and a final call to the permutation p_H is made on input $K_A^* \parallel \lfloor S \rfloor_{n-k}$ to obtain the k-bit tag T.

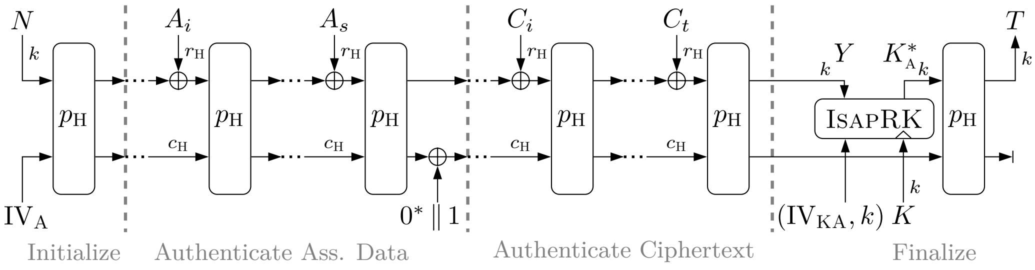
For verification, the tag T' is re-computed in the same way from the received nonce N, associated data A, and ciphertext C, and compared with the received tag T.



(a) IsapRK, with $(IV, z) = (IV_{KE}, n-k)$ in IsapEnc and $(IV, z) = (IV_{KA}, k)$ in IsapMAC



(b) IsapEnc



(c) IsapMAC

Figure 1: Isap authenticated encryption

2.2 Instantiation

Isap is instantiated with either the well-analyzed 400-bit permutation Keccak-p[400] [BDPV11, Nat15b] or the well-analyzed 320-bit permutation used in Ascon [DEMS16, DEMS19]. In total, we specify four instances, which are summarized in Table 1. For each of these instances, we specify the number of rounds of the permutations: s_H, s_B, s_E, s_K for permutations p_H, p_B, p_E, p_K, respectively. All four instances have security level $k = 128$. We provide a short description of the two permutations in Subsection A.1 and A.2.

The initial values IV_A, IV_{KA}, and IV_{KE}, which serve as domain separation between the different algorithms, are specified in Table 2. They are defined as the concatenated 8-bit integer values of all relevant parameters of the instance, plus a constant for the role of each IV. The initial values are then padded with zeros until they reach the required length of $n - k$ bits. For Isap-K-128 and Isap-K-128a, the resulting IVs have a length of 272 bits, while those for Isap-A-128 and Isap-A-128a are 192 bits long.

Algorithm 1 $\mathcal{E}(K, N, A, M)$

```
Input:   key $K \in \{0,1\}^k$,
         nonce $N \in \{0,1\}^k$,
         associated data $A \in \{0,1\}^*$,
         plaintext $M \in \{0,1\}^*$
Output:  ciphertext $C \in \{0,1\}^{|M|}$,
         tag $T \in \{0,1\}^k$

Encryption
  $C \leftarrow \textsc{IsapEnc}(K, N, M)$
Authentication
  $T \leftarrow \textsc{Isap}\text{MAC}(K, N, A, C)$
  return $C, T$
```

Algorithm 2 $\mathcal{D}(K, N, A, C, T)$

```
Input:   key $K \in \{0,1\}^k$,
         nonce $N \in \{0,1\}^k$,
         associated data $A \in \{0,1\}^*$,
         ciphertext $C \in \{0,1\}^*$,
         tag $T \in \{0,1\}^k$
Output:  plaintext $M \in \{0,1\}^*$, or error $\perp$

Verification
  $T' \leftarrow \textsc{Isap}\text{MAC}(K, N, A, C)$
  if $T \neq T'$ return $\perp$
Decryption
  $M \leftarrow \textsc{IsapEnc}(K, N, C)$
  return $M$
```

Algorithm 3 $\textsc{IsapEnc}(K, N, M)$

```
Input:   key $K \in \{0,1\}^k$,
         nonce $N \in \{0,1\}^k$,
         message $M \in \{0,1\}^*$
Output:  ciphertext $C \in \{0,1\}^{|M|}$

Initialization
  $M_1 \ldots M_t \leftarrow r_\text{H}$-bit blocks of $M \| 0^{-|M| \bmod r_\text{H}}$
  $K_\text{E}^* \leftarrow \textsc{Isap}\text{RK}(K, N, \text{IV}_\text{KE}, n-k)$
  $S \leftarrow K_\text{E}^* \| N$
Squeeze
  for $i = 1, \ldots, t$ do
    $S \leftarrow p_\text{E}(S)$
    $C_i \leftarrow S_{r_\text{H}} \oplus M_i$
  $C \leftarrow \lceil C_1 \| \ldots \| C_t \rceil^{|M|}$
  return $C$
```

Algorithm 4 $\textsc{Isap}\text{RK}(K, Y, \text{IV}, z)$

```
Input:   key $K \in \{0,1\}^k$,
         string $Y \in \{0,1\}^k$,
         constant $\text{IV} \in \{\text{IV}_\text{KE}, \text{IV}_\text{KA}\}$,
         output size $z \in \{n-k, k\}$
Output:  session key $K^* \in \{0,1\}^z$

Initialization
  $Y_1 \ldots Y_w \leftarrow r_\text{B}$-bit blocks of $Y \| 0^{-k \bmod r_\text{B}}$
  $S \leftarrow K \| \text{IV}$
  $S \leftarrow p_\text{K}(S)$
Absorb
  for $i = 1, \ldots, w-1$ do
    $S \leftarrow p_\text{B}((S_{r_\text{B}} \oplus Y_i) \| S_{c_\text{B}})$
  $S \leftarrow p_\text{K}((S_{r_\text{B}} \oplus Y_w) \| S_{c_\text{B}})$
Squeeze
  $K^* \leftarrow \lceil S \rceil^z$
  return $K^*$
```

Algorithm 5 $\textsc{Isap}\text{MAC}(K, N, A, C)$

```
Input:   key $K \in \{0,1\}^k$,
         nonce $N \in \{0,1\}^k$,
         associated data $A \in \{0,1\}^*$,
         ciphertext $C \in \{0,1\}^*$
Output:  tag $T \in \{0,1\}^k$

Initialization
  $A_1 \ldots A_s \leftarrow r_\text{H}$-bit bl. of $A \| 1 \| 0^{-|A|-1 \bmod r_\text{H}}$
  $C_1 \ldots C_t \leftarrow r_\text{H}$-bit bl. of $C \| 1 \| 0^{-|C|-1 \bmod r_\text{H}}$
  $S \leftarrow N \| \text{IV}_\text{A}$
  $S \leftarrow p_\text{H}(S)$
Absorbing Associated Data
  for $i = 1, \ldots, s$ do
    $S \leftarrow p_\text{H}((S_{r_\text{H}} \oplus A_i) \| S_{c_\text{H}})$
  $S \leftarrow S \oplus (0^{n-1} \| 1)$
Absorbing Ciphertext
  for $i = 1, \ldots, t$ do
    $S \leftarrow p_\text{H}((S_{r_\text{H}} \oplus C_i) \| S_{c_\text{H}})$
Squeezing Tag
  $K_A^* \leftarrow \textsc{Isap}\text{RK}(K, \lceil S \rceil^k, \text{IV}_\text{KA}, k)$
  $S \leftarrow p_\text{H}(K_A^* \| \lfloor S \rfloor_{n-k})$
  $T \leftarrow \lceil S \rceil^k$
  return $T$
```

Table 1: Recommended parameter configurations for ISAP.

Name	Permutation	Security level	Bit size of			Rounds			
		k	n	r_H	r_B	s_H	s_B	s_E	s_K
ISAP-K-128A	KECCAK-p[400]	128	400	144	1	16	1	8	8
ISAP-A-128A	ASCON-p	128	320	64	1	12	1	6	12
ISAP-K-128	KECCAK-p[400]	128	400	144	1	20	12	12	12
ISAP-A-128	ASCON-p	128	320	64	1	12	12	12	12

Table 2: Initial values for ISAP instances in hex notation.

ISAP	$\mathrm{IV_A}$	$1 \,\|\, k \,\|\, r_H \| r_B \,\|\, s_H \| s_B \| s_E \| s_K \,\|\, 0^*$
	$\mathrm{IV_{KA}}$	$2 \,\|\, k \,\|\, r_H \| r_B \,\|\, s_H \| s_B \| s_E \| s_K \,\|\, 0^*$
	$\mathrm{IV_{KE}}$	$3 \,\|\, k \,\|\, r_H \| r_B \,\|\, s_H \| s_B \| s_E \| s_K \,\|\, 0^*$
ISAP-K-128A	$\mathrm{IV_A}$	`01 80 9001 10010808 00`*
	$\mathrm{IV_{KA}}$	`02 80 9001 10010808 00`*
	$\mathrm{IV_{KE}}$	`03 80 9001 10010808 00`*
ISAP-A-128A	$\mathrm{IV_A}$	`01 80 4001 0C01060C 00`*
	$\mathrm{IV_{KA}}$	`02 80 4001 0C01060C 00`*
	$\mathrm{IV_{KE}}$	`03 80 4001 0C01060C 00`*
ISAP-K-128	$\mathrm{IV_A}$	`01 80 9001 140C0C0C 00`*
	$\mathrm{IV_{KA}}$	`02 80 9001 140C0C0C 00`*
	$\mathrm{IV_{KE}}$	`03 80 9001 140C0C0C 00`*
ISAP-A-128	$\mathrm{IV_A}$	`01 80 4001 0C0C0C0C 00`*
	$\mathrm{IV_{KA}}$	`02 80 4001 0C0C0C0C 00`*
	$\mathrm{IV_{KE}}$	`03 80 4001 0C0C0C0C 00`*

2.3 Security Claims

All ISAP family members provide 128-bit security against cryptographic attacks in the notion of nonce-based authenticated encryption with associated data (AEAD): they protect the confidentiality of the plaintext (except its length) and the integrity of ciphertext including the associated data (under adaptive forgery attempts). See also Table 3. Note that, as usual, a security loss by a small constant factor is expected.

Table 3: Security claims for recommended parameter configurations of ISAP.

Requirement	Security in bits			
	ISAP-K-128A	ISAP-A-128A	ISAP-K-128	ISAP-A-128
Confidentiality of plaintext	128	128	128	128
Integrity of plaintext	128	128	128	128
Integrity of associated data	128	128	128	128
Integrity of nonce	128	128	128	128

In order to fulfill the security claims stated in Table 3, implementations must ensure that the nonce is never repeated for two encryptions under the same key, and that the decryption process is only started after successful verification of the final tag. Except for the single-use requirement, there are no constraints on the choice of the nonce. It is possible to use a simple counter. It is beneficial that a system or protocol implementing the algorithm monitors and, if necessary, limits the number of tag verification failures per key. After reaching this limit, the decryption algorithm rejects all tags. Such a limit is not required for the security claims above, but may be reasonable in practice to increase the robustness against certain implementation attacks.

All algorithms are designed to achieve practical security against recovery of the secret master key by passive side-channel attacks assuming an implementation that is secured against simple power analysis (SPA) including template attacks. Furthermore, Isap is designed to improve robustness against other implementation attacks, including certain fault attacks.

3 Rationale

The main goal of Isap v1.0 [DEM+17] was to provide protection against differential power analysis (DPA) [KJJ99]. For Isap v2.0 [DEM+19], we provided several modifications on the mode to achieve additional robustness against other implementation attacks, such as fault attacks.

To achieve robustness against DPA, Isap v1.0 and Isap v2.0 incorporate a re-keying approach [MSGR10] in an efficient encrypt-then-MAC [BN00, KJJR11] scheme. While simple re-keying of both a MAC and an encryption scheme can only provide side-channel robustness for the encryption process, our scheme achieves side-channel robustness for multiple decryptions as well. Namely, the verification provides security of the decryption part in case of maliciously modified ciphertexts, while the MAC is protected by making its session key depend on the authenticated message itself.

The most significant difference of Isap v2.0 versus Isap v1.0 lies in the absorption of the nonce N during IsapEnc. In Isap v2.0, we decided to make the re-keying performed during IsapEnc hard to invert by overwriting part of the state with the nonce N. The change is clearly visible in Figure 1b: one can consider IsapRK to serve as re-keying function for a plain sponge-based stream cipher execution with key input $K^*_\mathrm{E} \| N$. The change implies that an attacker who is able to recover the state during the generation of the keystream cannot recover the master key K. As a result, neither the knowledge of the session key K^*_A nor of the session key K^*_E leads to a recovery of the master key K. This change results in an increased robustness of Isap v2.0 against active implementations attacks like Differential Fault Analysis (DFA) [BS97], Statistical Fault Attacks (SFA) [FJLT13, DEK+16], or Statistical Ineffective Fault Attacks (SIFA) [DEK+18, DMMP18, DEG+18].

Other changes include the use of a single key for both IsapMAC and IsapEnc instead of two independent keys. This has the advantage that less key material has to be stored than before. Furthermore, as Isap is a mode of operation that can be instantiated with any suitable permutation, we specify additional instances that use the 320-bit permutation of Ascon [DEMS16], which has recently been announced as the first choice for the use case of lightweight applications (resource constrained environments) in the final CAESAR portfolio [CAE14]. Compared to Keccak-p[400], Ascon-p maintains a smaller state size and is furthermore better suited for implementation on modern 64-bit CPUs.

4 Security of the ISAP Mode

In essence, the components of Isap follow two different permutation-based design strategies. IsapRK and IsapEnc are instances of a keyed duplex that *initialize the state with a key* and subsequently evolve the state by duplexing calls with extraction or absorption. The function IsapMAC, on the other hand, first absorbs data and *finalizes the state with a key.*

In two recent articles, Dobraunig and Mennink (DoMe) set out to perform a leakage resilience analysis of these two components. In [DM19a], DoMe proved leakage resilience of the generalized keyed duplex mode. This mode in particular covers IsapRK and the stream encryption within IsapEnc. DoMe showed how these two can be combined to obtain confidentiality of a variant of Isap [DM19a, Section 7]. In [DM20], DoMe introduced and formalized the suffix keyed sponge and proved its leakage resilience. The authentication part of Isap, IsapMAC, is a special type of suffix keyed sponge. These two works [DM19a, DM20] lead to the leakage resilience of Isap, with two caveats:

- The demonstration of how the duplex can be used to achieve confidentiality in [DM19a] is slightly different from how Isap performs encryption. The composition has yet to be described in detail.

- The security proof of the suffix keyed sponge abstracts the key absorption. In Isap, this key absorption is done by IsapRK, which is also called by IsapEnc. This means that we cannot directly conclude security of Isap from the disjoint results of [DM19a] and [DM20], but the combination must be spelled out.

Next, we show how the leakage resilience of the keyed duplex and the leakage resilience of the suffix keyed sponge accumulate to the leakage resilience of the Isap mode. (A preliminary version of this section appeared before as workshop records without proceedings [DM19b]). We want to note that we consider further cryptanalysis and also the evaluation of implementations of Isap to be crucial to get a deeper insight in the security of Isap.

4.1 Security Model

We consider security of Isap $= (\mathcal{E}, \mathcal{D})$ in the random permutation model. We consider a simplified setting where $p_1 := p_{\mathrm{K}} = p_{\mathrm{B}}$, $p_2 := p_{\mathrm{E}}$, and $p_3 := p_{\mathrm{H}}$ are uniformly randomly drawn from the set of all n-bit permutations: $p_1, p_2, p_3 \xleftarrow{\$} \mathsf{perm}(n)$. Let $K \xleftarrow{\$} \{0,1\}^k$. Let $\$_{*+k}$ be a function that for each (N, A, M) outputs a uniform random string of length $|M| + k$ bits (noting that a nonce should never be repeated), and let $\perp$ be a function that always returns $\perp$.

In the black-box security model, one would consider an adversary that has access to either $(\mathcal{E}_K^{\boldsymbol{p}}, \mathcal{D}_K^{\boldsymbol{p}}, \boldsymbol{p}^{\pm})$ in the real world or $(\$_{*+k}, \perp, \boldsymbol{p}^{\pm})$ in the ideal world, where $\boldsymbol{p} = (p_1, p_2, p_3)$ and where "$^{\pm}$" stands for bi-directional query access:

$$\mathbf{Adv}_{\mathrm{ISAP}}^{\mathrm{ae}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\mathcal{E}_K^{\boldsymbol{p}}, \mathcal{D}_K^{\boldsymbol{p}}, \boldsymbol{p}^{\pm} \ ; \ \$_{*+k}, \perp, \boldsymbol{p}^{\pm}\right) .$$

In case of leakage resilience, we adopt the conventional approach of non-adaptive leakage resilience, e.g., [Pie09, YSPY10, FPS12, SPY$^+$10, DP10], where the adversary has access to a leak-free version of the construction, which it has to distinguish from random, and a leaky version, which it may use to gather information. We assume that, a priori, any permutation evaluation within the leaky construction may leak information.

Formally, we obtain the following model, which follows Barwell et al. [BMOS17] with the difference that we consider security in the ideal permutation model. Let $\boldsymbol{p}, K, \$_{*+k}$ be as above. Let $\mathcal{L} = \{L : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\lambda\}$ be a class of leakage functions independent of the permutations, and for any leakage function $L \in \mathcal{L}$, define by $[\mathcal{E}_K^{\boldsymbol{p}}]_L$ (resp., $[\mathcal{D}_K^{\boldsymbol{p}}]_L$) an evaluation of $\mathcal{E}_K^{\boldsymbol{p}}$ (resp., $\mathcal{D}_K^{\boldsymbol{p}}$) where each permutation call within leaks λ bits of its input plus output. We now consider an adversary that *in addition* to the oracles in the black-box model has access to $[\mathcal{E}_K^{\boldsymbol{p}}]_L$ and $[\mathcal{D}_K^{\boldsymbol{p}}]_L$:

$$\mathbf{Adv}_{\mathrm{ISAP}}^{\mathrm{nalr\text{-}ae}}(\mathcal{A}) = \max_{L \in \mathcal{L}} \Delta_{\mathcal{A}}\left([\mathcal{E}_K^{\boldsymbol{p}}]_L, [\mathcal{D}_K^{\boldsymbol{p}}]_L, \mathcal{E}_K^{\boldsymbol{p}}, \mathcal{D}_K^{\boldsymbol{p}}, \boldsymbol{p}^{\pm} \ ; \ [\mathcal{E}_K^{\boldsymbol{p}}]_L, [\mathcal{D}_K^{\boldsymbol{p}}]_L, \$_{*+k}, \perp, \boldsymbol{p}^{\pm}\right) . \quad (1)$$

The adversary is not allowed to make an encryption query (to the leaky or leak-free oracle) under a repeated nonce.

4.2 Multicollision Limit Function

Daemen et al. [DMV17] introduced the multicollision limit function in the context of keyed sponge proofs. Let $q, n, k \in \mathbb{N}$ such that $k \leq n$. Consider the experiment of throwing q balls uniformly at random in 2^{n-k} bins, and denote by μ the maximum number of balls in any single bin. The multicollision limit function $\mu^q_{n-k,k}$ is defined as the smallest natural number x that satisfies

$$\mathbf{Pr}\left(\mu > x\right) \leq \frac{x}{2^k} .$$

Daemen et al. [DMV17] also gave an in-depth analysis of the term $\mu^q_{n-k,k}$. The analysis is tedious, but the conclusion is that the term behaves as follows:

$$\mu^q_{n-k,k} \lesssim \begin{cases} n/\log_2\left(\frac{2^{n-k}}{q}\right) , & \text{for } q \lesssim 2^{n-k} , \\ n \cdot \frac{q}{2^{n-k}} , & \text{for } q \gtrsim 2^{n-k} . \end{cases}$$

4.3 Main Result

We present the main result on the leakage resilience of the Isap mode. The result is stated with respect to the formalism of Subsection 4.1.

Theorem 1. *Assume that $4 \leq k \leq n$, and $1 \leq \lambda \leq 2n$. Let $\boldsymbol{p} = (p_1, p_2, p_3) \xleftarrow{\$} \mathsf{perm}(n)^3$ and $K \xleftarrow{\$} \{0,1\}^k$. Let $\mathcal{L} = \{L : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\lambda\}$ be a class of leakage functions. For any adversary making $q_e \geq 2$ encryption queries with unique nonces and q_v verification queries (writing $q = q_e + q_v$) with a total amount of Q plaintext blocks, and $P \leq 2^{n-1}$ primitive queries to each of p_1, p_2, p_3,*

$$\begin{aligned}\mathbf{Adv}^{\text{nalr-ae}}_{\textit{ISAP}}(\mathcal{A}) \leq\ & \frac{4\binom{4+2kq+P}{2} + 2\binom{Q+P}{2} + 6\binom{P}{2}}{2^n} + \frac{2\binom{Q}{2}}{2^{n-\lambda}} + \frac{32kqP + 16k^2q^2}{2^{n-4\lambda}} \\ & + \frac{2\mu^{2(P-q)}_{k,n-k}}{2^{n-k}} + \frac{2\mu^{2q}_{k,n-k} \cdot P}{2^{n-k-\lambda}} + \frac{2P + 2qQ}{2^{n-k-2\lambda}} \\ & + \frac{8P^2}{2^{2k}} + \frac{4\mu^{Q}_{n-2k,2k} \cdot (P+1)}{2^{2k-2\lambda}} + \frac{q_v + 4}{2^k} + \frac{8P}{2^{k-2\lambda}} + \frac{2\mu^{2(P-q)}_{n-k,k} \cdot P}{2^{k-\lambda-\mu^{2(P-q)}_{k,n-k}\lambda}} .\end{aligned}$$

The proof is included in Subsection 4.4.

4.4 Proof of Theorem 1

Note that both encryption $\mathcal{E}$ and decryption $\mathcal{D}$ of Isap can be specified as function of IsapRK =: IR, IsapEnc =: IE, and IsapMAC =: IM:

$$\mathcal{E}^{\boldsymbol{p}}_K = \mathcal{E}^{\mathrm{IR}^{p_1}_K, \mathrm{IE}^{p_2}_{\boldsymbol{K}^\star_{\mathrm{KE}}}, \mathrm{IM}^{p_3}_{\boldsymbol{K}^\star_{\mathrm{KA}}}} ,$$
$$\mathcal{D}^{\boldsymbol{p}}_K = \mathcal{D}^{\mathrm{IR}^{p_1}_K, \mathrm{IE}^{p_2}_{\boldsymbol{K}^\star_{\mathrm{KE}}}, \mathrm{IM}^{p_3}_{\boldsymbol{K}^\star_{\mathrm{KA}}}} ,$$

where $\boldsymbol{K}^\star_{\mathrm{KE}}$ is defined as the output states of $\mathrm{IR}^{p_1}_K$ for $\mathrm{IV} = \mathrm{IV}_{\mathrm{KE}}$, and $\boldsymbol{K}^\star_{\mathrm{KA}}$ the output states of $\mathrm{IR}^{p_1}_K$ for $\mathrm{IV} = \mathrm{IV}_{\mathrm{KA}}$. Here, the * is used to explicitly remind of the fact that the keys come from $\mathrm{IR}^{p_1}_K$. Note that these values are, in particular, defined by the inputs to IE^{p_2} and IM^{p_3}.

Let $L \in \mathcal{L}$ be any leakage and $\mathcal{A}$ be any adversary. Our goal is to bound

$$
\begin{aligned}
&\Delta_{\mathcal{A}}\left([\mathcal{E}_K^{\boldsymbol{p}}]_L, [\mathcal{D}_K^{\boldsymbol{p}}]_L, \mathcal{E}_K^{\boldsymbol{p}}, \mathcal{D}_K^{\boldsymbol{p}}, \boldsymbol{p}^{\pm} \;;\; [\mathcal{E}_K^{\boldsymbol{p}}]_L, [\mathcal{D}_K^{\boldsymbol{p}}]_L, \$_{*+k}, \perp, \boldsymbol{p}^{\pm}\right)\\
=&\Delta_{\mathcal{A}}\left(\left[\mathcal{E}^{\mathrm{IR}_K^{p_1}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L, \left[\mathcal{D}^{\mathrm{IR}_K^{p_1}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L, \mathcal{E}^{\mathrm{IR}_K^{p_1}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}, \mathcal{D}^{\mathrm{IR}_K^{p_1}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}, \boldsymbol{p}^{\pm};\right.\\
&\left.\left[\mathcal{E}^{\mathrm{IR}_K^{p_1}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L, \left[\mathcal{D}^{\mathrm{IR}_K^{p_1}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L, \$_{*+k}, \perp, \boldsymbol{p}^{\pm}\right). \qquad (2)
\end{aligned}
$$

4.4.1 Eliminating IR^{p_1}

The function $\mathrm{IR}_K^{p_1}$ is called a total amount of at most $2q = 2(q_e + q_v)$ times: q times for $\mathrm{IV} = \mathrm{IV}_{\mathrm{KE}}$ with a requested output of $n-k$ bits, and q times for $\mathrm{IV} = \mathrm{IV}_{\mathrm{A}}$ with a requested output of k bits. Note that repeated evaluations of $\mathrm{IR}_K^{p_1}$ for the same nonce give the same output and are not counted doubly. It is a duplex construction, and we can rely on the leakage resilience of the duplex [DM19a]. Concretely, we can view it as an idealized duplex function $\mathsf{AIXIF1}^{ro}$ based on a random oracle. Details about this idealized duplex function can be found in [DM19a], but for the specific case of $\mathrm{IR}_K^{p_1}$, one can think of it as a random function that simply absorbs all data (K, Y, IV) and outputs either $n-k$ or k bits of output, and that outputs dummy leakages for each duplexing call. The following Proposition 1 is very similar to [DM19a, Corollary 1]: it is based on slightly different parametrization, but we have performed the same simplifications on the bound.

Proposition 1. *Assume that $4 \le k \le n$, and $1 \le \lambda \le 2n$. Let $p_1 \xleftarrow{\$} \mathsf{perm}(n)$ and $K \xleftarrow{\$} \{0,1\}^k$. Let $\mathsf{AIXIF1}^{ro}$ be an idealized duplex function based on a random oracle (details can be found in [DM19a]). Let $\mathcal{L} = \{L : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\lambda\}$ be a class of leakage functions. For any adversary $\mathcal{A}'$ making $q \ge 2$ queries for IV_{KE} and $q \ge 2$ queries for IV_{KA}, all of length at most k bits, and P primitive queries to p_1,*

$$
\begin{aligned}
\mathbf{Adv}_{IR}^{\text{nalr-duplex}}(\mathcal{A}') &= \max_{L \in \mathcal{L}} \Delta_{\mathcal{A}'}\left([IR_K^{p_1}]_L, p_1^{\pm} \;;\; [\mathsf{AIXIF1}_K^{ro}]_L, p_1^{\pm}\right)\\
&\le \frac{8kqP + 4k^2q^2}{2^{n-4\lambda}} + \frac{\binom{4+2kq+P}{2} + \binom{P}{2}}{2^n} + \frac{2P}{2^{k-2\lambda}} + \frac{1}{2^k}. \qquad (3)
\end{aligned}
$$

In addition, except with probability at most the same bound, all output states after absorption have min-entropy at least $n - \lambda$.

A simple hybrid reduction allows us to replace $\mathrm{IR}_K^{p_1}$ by $\mathsf{AIXIF1}_K^{ro}$ in (2):

$$
\begin{aligned}
(2) \le\ &\Delta_{\mathcal{A}}\left(\left[\mathcal{E}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L, \left[\mathcal{D}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L,\right.\\
&\quad \mathcal{E}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}, \mathcal{D}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}, \boldsymbol{p}^{\pm} \;;\\
&\quad \left.\left[\mathcal{E}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L, \left[\mathcal{D}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L, \$_{*+k}, \perp, \boldsymbol{p}^{\pm}\right)\\
&+ 2 \cdot \Delta_{\mathcal{A}'}\left([\mathrm{IR}_K^{p_1}]_L, p_1^{\pm} \;;\; [\mathsf{AIXIF1}_K^{ro}]_L, p_1^{\pm}\right)\\
\le\ &\Delta_{\mathcal{A}}\left(\left[\mathcal{E}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L, \left[\mathcal{D}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L,\right.\\
&\quad \mathcal{E}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}, \mathcal{D}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}, \boldsymbol{p}^{\pm} \;;\\
&\quad \left.\left[\mathcal{E}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L, \left[\mathcal{D}^{\mathsf{AIXIF1}_K^{ro}, \mathrm{IE}_{\boldsymbol{K}^\star_{\mathrm{KE}}}^{p_2}, \mathrm{IM}_{\boldsymbol{K}^\star_{\mathrm{KA}}}^{p_3}}\right]_L, \$_{*+k}, \perp, \boldsymbol{p}^{\pm}\right) + 2 \cdot (3). \qquad (4)
\end{aligned}
$$

4.4.2 Towards mutually independent IE^{p_2} and IM^{p_3}

The function $\mathsf{AIXIF1}_K^{ro}$ is independent of all other functions in the oracles, and the adversary never gets its outcomes. This means that we can basically plainly replace $\boldsymbol{K}_{\mathrm{KE}}^{\star}$ by a dummy $\boldsymbol{K}_{\mathrm{KE}} \xleftarrow{\mathcal{D}_{\boldsymbol{K}}} (\{0,1\}^{n-k})^{2^k}$ consisting of keys with min-entropy $n-k-\lambda$. Note that $\mathsf{AIXIF1}_K^{ro}$ is called by IE^{p_2} for at most q different values, namely the nonces, and each nonce henceforth lets $\mathrm{IE}^{p_2}_{\boldsymbol{K}_{\mathrm{KE}}}$ select the resulting key. Likewise, we can replace $\boldsymbol{K}_{\mathrm{KA}}^{\star}$ by a dummy $\boldsymbol{K}_{\mathrm{KA}} \xleftarrow{\mathcal{D}_{\boldsymbol{K}}} (\{0,1\}^{k})^{2^k}$ consisting of keys with min-entropy $k-\lambda$, with the remark that identical evaluations of $\mathsf{AIXIF1}_K^{ro}$ by IM^{p_3} yield identical outputs and thus identical selections from $\boldsymbol{K}_{\mathrm{KA}}$. Here, distribution $\mathcal{D}_{\boldsymbol{K}}$ takes the leakage into account, but details of $\mathcal{D}_{\boldsymbol{K}}$ are irrelevant: all that matters is that the resulting values have a min-entropy $n-k-\lambda$ resp. $k-\lambda$ after leakage. The step is done at the price of the bound of Proposition 1, noting that except with that bound the output states of $\mathsf{AIXIF1}_K^{ro}$ have min-entropy at least $n-k-\lambda$ resp. $k-\lambda$. Now, there is no need to keep "$\mathsf{AIXIF1}_K^{ro}$" in the equation anymore, and we obtain from (4):

$$\begin{aligned}(2) \leq \Delta_{\mathcal{A}}\Big(&\Big[\mathcal{E}^{\mathrm{IE}^{p2}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \Big[\mathcal{D}^{\mathrm{IE}^{p2}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \mathcal{E}^{\mathrm{IE}^{p2}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}, \mathcal{D}^{\mathrm{IE}^{p2}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}, \boldsymbol{p}^{\pm}\ ;\\ &\Big[\mathcal{E}^{\mathrm{IE}^{p2}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \Big[\mathcal{D}^{\mathrm{IE}^{p2}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \$_{*+k}, \perp, \boldsymbol{p}^{\pm}\Big)\\ &+ 4\cdot(3)\,. \end{aligned} \tag{5}$$

In both worlds, the encryption and authentication are mutually independent: the former is instantiated with $p_2 \xleftarrow{\$} \mathsf{perm}(n)$ and $\boldsymbol{K}_{\mathrm{KE}} \xleftarrow{\mathcal{D}_{\boldsymbol{K}}} (\{0,1\}^{n-k})^{2^k}$ and the latter is instantiated with $p_3 \xleftarrow{\$} \mathsf{perm}(n)$ and $\boldsymbol{K}_{\mathrm{KA}} \xleftarrow{\mathcal{D}_{\boldsymbol{K}}} (\{0,1\}^{k})^{2^k}$. We can therefore cleanly replace both functionalities independently.

4.4.3 Individual results on IE^{p_2} and IM^{p_3}

For the encryption $\mathrm{IE}^{p_2}_{\boldsymbol{K}_{\mathrm{KE}}}$, we consider it to be a duplex construction, and derive a leakage resilience bound from [DM19a, Theorem 1]. For the specific case of $\mathrm{IE}^{p_2}_{\boldsymbol{K}_{\mathrm{KE}}}$, the idealized duplex function can be thought of as simply absorbing (sub-)key K_{E}^* and nonce N and outputting a sufficiently large keystream. Note that, in fact, $\mathrm{IE}^{p_2}_{\boldsymbol{K}_{\mathrm{KE}}}$ is only slightly different from the construction considered in [DM19a, Corollary 2] and we can follow a comparable line of reasoning, but now with the differences that, if nonces are not repeated, Ω and ν_{fix} (both related to the number of evaluations where an adversary can set the outer part of the state to a certain value) now equal 0, and q_{IV} (the maximum number of queries for a single nonce) equals 1. Here, it is important to note that $\mathrm{IE}^{p_2}_{\boldsymbol{K}_{\mathrm{KE}}}$ gets called at most $q = q_e + q_v$ times: the q_e encryption calls are always for different nonces, the q_v verification calls might be for repeated nonces. However, if this happens, also the key under which it is queried is identical, and this means that the evaluation of $\mathrm{IE}^{p_2}_{\boldsymbol{K}_{\mathrm{KE}}}$ is a repeated query and is not counted doubly. In addition, theoretically $\mathrm{IE}^{p_2}_{\boldsymbol{K}_{\mathrm{KE}}}$ might be called multiple times for the same key with different nonces. This, however, only happens with a small probability. Therefore, we also bound the term q_δ, the maximum number of initialization calls for single key, probabilistically by 1. This incurs an extra term $\frac{\binom{q}{2}}{2^{n-k-2\lambda}}$. That term is, eventually, absorbed in the simplification performed on the bound.

Concretely, we take the general bound of [DM19a, Theorem 1] for the following parameters:

- State, capacity, and rate are n, $2k$, and $n-2k$, respectively. Key size is $n-k$, and the min-entropy of the key is $n-k-\lambda$.
- The parameter α, that rotates the input to the initialization of the duplex, equals 0.

- The number of instances, or “users”, equals the number of initialization calls: q. The online complexity is Q and the offline complexity P.
- As nonces are unique, $q_{\mathrm{IV}} = 1$. The parameter q_δ, that measures the number of initialization calls for a single key (with different nonces), is bounded to equal 1. As explained above, this bound incurs an extra term $\frac{\binom{q}{2}}{2^{n-k-2\lambda}}$.
- Parameters L, Ω, and ν_{fix}, that measure the number of duplexing calls for which the adversary knows/influences the outer part input, equal 0. Additionally, paths in the duplex do not repeat and we have $R = 1$.

With these parameters, we obtain below Proposition 2 from [DM19a, Theorem 1].

Proposition 2. *Assume that $4 \le k \le n$, and $1 \le \lambda \le 2n$. Let $p_2 \xleftarrow{\$} \mathsf{perm}(n)$ and $\boldsymbol{K}_{\text{KE}} \xleftarrow{\mathcal{D}_{\boldsymbol{K}}} (\{0,1\}^{n-k})^q$ be a random array of keys each with min-entropy at least $n-k-\lambda$. Let $\mathsf{AIXIF2}^{ro}$ be an idealized duplex function based on a random oracle (details can be found in [DM19a]). Let $\mathcal{L} = \{L : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\lambda\}$ be a class of leakage functions. For any adversary $\mathcal{A}''$ making $q \ge 2$ queries with unique nonces and Q plaintext blocks, and $P \le 2^{n-1}$ primitive queries to p_2,*

$$\begin{aligned}\mathbf{Adv}_{IE}^{\text{nalr-duplex}}(\mathcal{A}'') &= \max_{L\in\mathcal{L}} \Delta_{\mathcal{A}''}\left(\left[IE^{p_2}_{\boldsymbol{K}_{\text{KE}}}\right]_L, p_2^{\pm}\ ;\ \left[\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\text{KE}}}\right]_L, p_2^{\pm}\right) + \frac{\binom{q}{2}}{2^{n-k-2\lambda}} \\ &\le \frac{2\mu^{Q}_{n-2k,2k}\cdot(P+1)}{2^{2k-2\lambda}} + \frac{\binom{Q}{2}}{2^{n-\lambda}} + \frac{P+qQ}{2^{n-k-2\lambda}} + \frac{\binom{Q+P}{2}+\binom{P}{2}}{2^n}\,. \end{aligned} \tag{6}$$

For the message authentication $\mathrm{IM}^{p_3}_{\boldsymbol{K}_{\text{KA}}}$, this is basically a suffix keyed sponge with properly protected key absorption function G that is $2^{-(k-\lambda)}$-uniform and $2^{-(k-\lambda)}$-universal. It operates on capacity $c = 2k-1$, noting that the addition of $0^* \,\|\, 1$ as domain separator between the hashing of associated data and ciphertext reduces the capacity by 1. We obtain below Proposition 3 immediately from [DM20, Theorem 3].

Proposition 3. *Let $p_3 \xleftarrow{\$} \mathsf{perm}(n)$ and $\boldsymbol{K}_{\text{KE}} \xleftarrow{\mathcal{D}_{\boldsymbol{K}}} (\{0,1\}^k)^q$ be a random array of keys each with min-entropy at least $k-\lambda$. Let $\$_k$ be a function that outputs random k-bit strings for each new arbitrarily-long input. Let $\mathcal{L} = \{L : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\lambda\}$ be a class of leakage functions. For any adversary $\mathcal{A}'''$ making $q \ge 2$ queries, all of length at most k bits, and $P \le 2^{n-1}$ primitive queries to p_3,*

$$\begin{aligned}\mathbf{Adv}_{IM}^{\text{nalr-prf}}(\mathcal{A}''') &= \max_{L\in\mathcal{L}} \Delta_{\mathcal{A}'''}\left(\left[IM^{p_3}_{\boldsymbol{K}_{\text{KA}}}\right]_L, IM^{p_3}_{\boldsymbol{K}_{\text{KA}}}, p_3^{\pm}\ ;\ \left[IM^{p_3}_{\boldsymbol{K}_{\text{KA}}}\right]_L, \$_k, p_3^{\pm}\right) \\ &\le \frac{2P^2}{2^{2k-1}} + \frac{\mu^{2(P-q)}_{k,n-k}}{2^{n-k}} + \frac{\mu^{2(P-q)}_{n-k,k}\cdot P}{2^{k-\lambda-\mu^{2(P-q)}_{k,n-k}\lambda}} + \frac{\mu^{2q}_{k,n-k}\cdot P}{2^{n-k-\lambda}}\,. \end{aligned} \tag{7}$$

4.4.4 Completing the proof

We will apply above propositions to (5) to conclude the proof. Informally, we will first replace $\mathrm{IE}^{p_2}_{\boldsymbol{K}_{\text{KE}}}$ with $\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\text{KE}}}$ at the cost of $\mathbf{Adv}_{\mathrm{IE}}^{\text{nalr-duplex}}(\mathcal{A}'')$ of Proposition 2. Then, we replace $\mathrm{IM}^{p_3}_{\boldsymbol{K}_{\text{KA}}}$ with $\$_k$ at the cost of $\mathbf{Adv}_{\mathrm{IM}}^{\text{nalr-prf}}(\mathcal{A}''')$ of Proposition 3. These transitions have to be performed in both worlds of (5), yielding a factor 2.

Formally, we obtain from (5), for adversaries $\mathcal{A}''$ and $\mathcal{A}'''$ as quantified in Proposition 2 and Proposition 3, respectively:

$$\begin{aligned}
(2) \leq & \Delta_{\mathcal{A}} \Big(\Big[\mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \Big[\mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}, \mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}, \boldsymbol{p}^{\pm}; \\
& \qquad \Big[\mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \Big[\mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \$_{*+k}, \perp, \boldsymbol{p}^{\pm} \Big) \\
& + 4 \cdot (3) + 2 \cdot \mathbf{Adv}^{\text{nalr-duplex}}_{\mathrm{IE}}(\mathcal{A}'') \\
\leq & \Delta_{\mathcal{A}} \Big(\Big[\mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \Big[\mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}, \mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}, \boldsymbol{p}^{\pm}; \\
& \qquad \Big[\mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \Big[\mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \$_{*+k}, \perp, \boldsymbol{p}^{\pm} \Big) \\
& + 4 \cdot (3) + 2 \cdot (6) \\
\leq & \Delta_{\mathcal{A}} \Big(\Big[\mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \Big[\mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\$_k}, \mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\$_k}, \boldsymbol{p}^{\pm} ; \\
& \qquad \Big[\mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \Big[\mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \$_{*+k}, \perp, \boldsymbol{p}^{\pm} \Big) \\
& + 4 \cdot (3) + 2 \cdot (6) + 2 \cdot \mathbf{Adv}^{\text{nalr-prf}}_{\mathrm{IM}}(\mathcal{A}''') \\
\leq & \Delta_{\mathcal{A}} \Big(\Big[\mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \Big[\mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\$_k}, \mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\$_k}, \boldsymbol{p}^{\pm} ; \\
& \qquad \Big[\mathcal{E}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \Big[\mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\mathrm{IM}^{p3}_{\boldsymbol{K}_{\mathrm{KA}}}}\Big]_L, \$_{*+k}, \perp, \boldsymbol{p}^{\pm} \Big) \\
& + 4 \cdot (3) + 2 \cdot (6) + 2 \cdot (7) \,. \qquad (8)
\end{aligned}$$

The remaining distance of (8) boils down to forging a tag for $\mathcal{D}^{\mathsf{AIXIF2}^{ro}_{\boldsymbol{K}_{\mathrm{KE}}},\$_k}$, in which the adversary succeeds with probability at most $\frac{q_v}{2^k}$:

$$(2) \leq 4 \cdot (3) + 2 \cdot (6) + 2 \cdot (7) + \frac{q_v}{2^k} \,. \qquad (9)$$

5 Implementation

The main design goal of Isap is to provide out-of-the-box robustness against certain types of implementation attacks while allowing to add additional defense mechanisms at low cost. This is essential in situations where cryptographic devices are deployed in locations where they are physically accessible by potential attackers. The area requirements of Isap are very low even with integrated countermeasures against side-channel attacks, so the scheme is suitable for deployment in software or hardware on very constrained devices that are exposed to adversarial access. These features make Isap an excellent choice for a variety of applications on constrained devices in the IoT (Internet of Things), particularly for highly sensitive processes with bulk data, such as software and firmware updates.

This section covers implementation aspects of Isap. We first provide an overview of Isap's performance in software and hardware in Subsection 5.1 and Subsection 5.2, respectively. We then discuss the robustness of Isap against implementation attacks and specific aspects to consider for securely implementing Isap in Subsection 5.3. Finally, in Subsection 5.4 we discuss what happens if implementations break with the strict structure of Isap that the verification has to be executed before the decryption. A simultaneous processing of ciphertext blocks in the verification and decryption is likely to provide a similar protection of cryptographic keys against implementation attacks but loses side-channel protection of the plaintext during decryption. Up-to-date implementations of Isap can be found on `https://isap.iaik.tugraz.at/implementations`.

5.1 Software Implementations

We developed generic and platform-optimized implementations of all Isap instantiations for various CPU architectures such as x64, ARMv6, and ARMv7. The codebase thus covers high performance scenarios like 64-bit CPUs, as well as more constrained devices such as 32-bit ARM Cortex-A application processors and Cortex-M microprocessors, where implementation security is often of particular interest.

We benchmarked our implementations on various platforms, covering scenarios from high-end desktop CPUs (such as AMD Ryzen 7 1700) to low end microprocessors (such as STM32F405). The benchmarked scenarios include authenticated encryption of relatively small messages (64 bytes), typical Ethernet II frame sizes (1536 bytes), and very large messages (long[2]). The resulting performance metrics are listed in Table 4.

Table 4: Encryption performance of Isap in cycles/byte for different message lengths.

(a) Instances based on Ascon-p

	Isap-A-128a			Isap-A-128		
Message length in Bytes	64 B	1536 B	long	64 B	1536 B	long
AMD Ryzen 7 1700 (x64)	85.7	24.5	21.9	511.0	48.9	29.8
Intel i5-6200U (x64)	104.0	34.3	31.4	698.0	68.1	42.0
Raspberry Pi 1B (ARMv6M)	966.0	190.0	161.0	3771.0	347.0	211.0
STM32F405 (ARMv7M)	1223.0	352.0	311.0	6818.0	675.0	406.0

(b) Instances based on Keccak-p[400]

	Isap-K-128a			Isap-K-128		
Message length in Bytes	64 B	1536 B	long	64 B	1536 B	long
AMD Ryzen 7 1700 (x64)	295.0	64.1	54.3	2108.0	156.0	75.0
Intel i5-6200U (x64)	342.0	72.8	61.3	2318.0	173.0	84.0
Raspberry Pi 1B (ARMv6M)	3464.0	743.0	635.0	23 917.0	1790.0	878.0
STM32F405 (ARMv7M)	4007.0	808.0	658.0	25 472.0	1909.0	869.0

When compared to the reported performance numbers of the original NIST submission document [DEM+19], we can see performance improvements in all scenarios. For small messages the runtime of Isap is dominated by the re-keying operation IsapRK while the runtime of hashing and encrypting dominates for processing longer messages. The high speed-up of Isap-A-128a and Isap-K-128a, when compared to their conservative counterparts, is due to the parametrization of the initialization for short messages, and due to parametrization of encryption and authentication for long messages.

5.2 Hardware Implementations

We provide estimations for the performance and area requirements of Isap in hardware. The numbers are based on concrete ASIC implementations of Isap from the original FSE paper [DEM+17] (Isap v1.0 in the following), but accommodate for the modifications in this proposal.

The Isap family members differ in the permutation's round function, the rate, and the number of rounds. The implementations employ a single instance of the permutation

[2]Long messages represent 1/512 of the difference in cycle counts between processing 2048-byte and 1536-byte inputs.

(Keccak-p[400] or Ascon-p) that performs one round per cycle. The number of rounds performed is chosen at runtime depending on the executed algorithm, i.e., IsapEnc, IsapMAC, or IsapRK. Table 5 shows the synthesis results for Isap v1.0 based on 130 nm UMC technology. We expect these numbers to be quite accurate for Isap's Keccak-based instances since area requirements have not changed compared to Isap v1.0 and the impact of different round numbers on the runtime is easy to estimate. For the Ascon-based instances we refer to synthesis results of the fast one-round permutation in 90 nm UMC technology by [GWDE15]. We combine these results with the existing design but replace the area/performance metrics for the permutation. These numbers are thus rougher and more pessimistic estimates and will be refined once dedicated ASIC implementations are available.

Table 5: Hardware characteristics of all Isap instantiations.

Function	Area [kGE]	Frequency [MHz]	Initialization [cycles]	[µs]	Runtime per byte [cycles]	[ns]
Isap-A-128a	$\leq$ 12.78	$\geq$ 169	556	4	2.75	$\leq$ 15.00
Isap-A-128	$\leq$ 12.78	$\geq$ 169	3 374	21	3.50	$\leq$ 20.00
Isap-K-128a	14.00	169	580	4	1.55	8.88
Isap-K-128	14.00	169	3 406	21	2.00	11.11

5.2.1 Area

As Isap-K-128 and Isap-K-128a use the same implementation design, they consume roughly the same chip area if the same permutation is used. Most of the chip area is due to the permutation core, which consumes 8.3 kGE (Keccak-p[400]) or $\leq$ 7.08 kGE (full Ascon scheme including the mode [GWDE15]). The remaining logic of about 5.7 kGE is required for multiplexing and a temporary state register to hold the hash value within IsapMAC while executing the re-keying function IsapRK. A standalone implementation of IsapRK yields roughly the same size as the Keccak core itself and is thus smaller than other re-keying functions like a masked polynomial multiplication [MSGR10] or an implementation of the GGM tree using an AES core computing 1 round per cycle [SPY+10].

5.2.2 Runtime

The runtime can be divided into two parts: the time for performing initialization/finalization and the time for processing data blocks. The initialization/finalization runtime is dominated by the re-keying operations in both IsapEnc and IsapMAC and is independent of the length of the message. Its impact on runtime thus vanishes for long messages. The runtime for processing a single block is also independent of the length of the message, but defines the overall runtime for long messages.

Compared to the conservative parameterization in Isap-K-128, Isap-K-128a yields a speed-up of 83 % for initialization and 22 % for the processing of a message block. The substantial speed-up during initialization is highly relevant for short messages, while the speed-up observed for encryption and authentication of a 144-bit message block dominates for long messages.

5.2.3 Comparison

Isap is an efficient authenticated encryption scheme with low hardware footprint that prevents DPA by design. Isap can be implemented securely using a standard implementation of the 400-bit Keccak permutation and adds only a small hardware overhead, while a first-order secure threshold implementation to achieve DPA protection on the primitive level would increase the area by a factor of 3 to 4 [BDN$^+$13]. For other cryptographic primitives such as the AES, the area overhead for first-order secure masked implementations is similar or even worse [DRB$^+$16, GMK17]. When higher-order DPA robustness is required, the hardware overhead of masking rises even more [GMK17]. Consequently, the implementation cost of standard authenticated encryption modes for AES such as AES-CCM and AES-GCM secured via masking rises accordingly.

5.3 Implementation Security

Two main directions in counteracting implementation attacks exist. The first approach works by hardening the implementation of cryptographic algorithms with techniques like hiding or masking [GP99, CJRR99]. The second approach to counteract implementation attacks is to use cryptographic protocols that ensure that certain types of attacks cannot be performed at all on the underlying cryptographic primitive [DP08, Pie09, MSGR10, MPR$^+$11, DKM$^+$15].

Isap combines both approaches by providing a mode of operation that increases resistance against implementation attacks, which is instantiated with permutations that lend themselves to efficient countermeasures on the primitive level. While the original proposal Isap v1.0 at FSE 2017 [DEM$^+$17] already provides robustness against DPA attacks by design [DMP20], the additional modifications in current proposal Isap v2.0 also provide hardening against several types of fault attacks such as DFA [BS97], SFA [FJLT13, DEK$^+$16], or SIFA [DEK$^+$18, DMMP18, DEG$^+$18], the last of which is especially hard to prevent on a primitive level. As a consequence, most parts of the underlying cryptographic primitive only need to be secured against passive attacks that can extract information about the key by observing cryptographic operations for a single fixed input, i.e., SPA. This induces a significantly lower implementation overhead of the protected primitive compared to implementations that need protection against DPA attacks on a primitive-level.

In summary, Isap's robustness against passive implementation attacks rests on the following pillars:

1. IsapEnc and IsapMAC, the encryption/decryption and authentication procedures, are inherently protected against DPA by Isap's Encrypt-then-MAC mode with its re-keying function, which guarantees that fresh keys are used whenever processing new data.

2. IsapEnc's and IsapMAC's robustness against SPA follows directly from the underlying sponge construction under a generous bounded-leakage assumption.

3. IsapRK, the re-keying procedure called internally by IsapEnc and IsapMAC, is the sponge-based equivalent of the 2-limiting GGM construction and thus protected against DPA.

4. IsapRK's robustness against SPA follows from the same model and assumptions as IsapEnc's.

5.3.1 SPA Leakage

Isap has primarily been designed to be robust against DPA attacks. Furthermore, the design of Isap's components IsapMAC, IsapRK, and IsapEnc have an increased capacity in order to better withstand SPA attacks. Still, like for any scheme, robustness against SPA attacks such as template attacks relies on limiting the leakage per execution, which may require additional implementation countermeasures such as hiding. This applies in particular for the decryption, where an attacker may obtain several measurements for the same data.

As pointed out by Medwed et al. [MSJ12], the concrete security of a construction against side-channel attacks highly depends on the way it is implemented and on the platform on which it is executed. For instance, they show that an implementation of the GGM construction using AES-128 on an 8-bit microcontroller can be broken by using template attacks. By making assumptions on the implementation, e.g., parallel execution of the S-boxes, Medwed et al. [MSJ12, MSNF16] were able to provide security guarantees with respect to side-channel attacks for their constructions. In contrast, in this work we do not make any assumption on the way Isap is implemented and on the countermeasures used to protect the implementations. Clearly, an 8-bit microcontroller implementation needs more sophisticated SPA countermeasures than a parallel implementation of the round function. We consider the evaluation of the SPA robustness of various implementation strategies for Isap to be an interesting topic for further research.

5.3.2 Tag Comparison

Special care has to be taken for tag comparison. On the one hand, an active attacker performing fault attacks to skip the tag comparison will be able to break the authenticity of the scheme. Therefore, additional implementation countermeasures are needed to prevent this. On the other hand, as observed by Berti et al. [BGP+19], the comparison of the tag should be done in a side-channel secured manner to minimize the leakage of the correctly computed tag. One option to do this is to mask the comparison. Another option is to do the comparison after another permutation call: the computed tag T' and the transmitted tag T are compared by first looking at k bits of $\lfloor p_{\mathrm{H}}(T'\|0^*)\rfloor_k \oplus \lfloor p_{\mathrm{H}}(T\|0^*)\rfloor_k$, and the comparison of T, T' is only executed if the first comparison was successful. Learning k bits of information of the output $p_{\mathrm{H}}(T'\|0^*)$ is of no help in the quest of recovering T' in order to mount a forgery.

5.3.3 Fault Attacks

Isap's updated mode also provides robustness against certain fault attacks. Since the nonce changes for each authenticated encryption call, so do K_{A}^* and K_{E}^*, which renders classical fault attacks like DFA impractical against the authenticated encryption. Other fault attacks like SFA [FJLT13, DEK+16] or SIFA [DEK+18, DMMP18, DEG+18] might still be applicable in this setting, but we expect that the SPA countermeasures that are typically in place to cope with SPA attacks will drastically increase the complexity of these attacks. In particular, the extremely small rate and the resulting data complexity of 2 in the re-keying function of Isap will significantly increase the complexity of extracting Kusing SFA or SIFA. Additionally, in case an attacker manages to obtain one of the two session keys K_{A}^* or K_{E}^*, it is infeasible to recover the master key K without performing additional implementation attacks, since the re-keying function IsapRK is hard to invert.

During authenticated decryption, the nonce can be kept constant for multiple computations, which potentially enables DFA on the decryption. As mitigation, Isap's re-keying function is hard to invert, forcing an attacker to mount the attack on the re-keying function itself. However, since the session keys produced by the re-keying function can typically not be simply observed by the attacker, DFA attacks on the scheme are significantly more complicated. Additionally, we can track the number of failed verifications and halt the device after a few verification failures. This will significantly increase the robustness of the implementation against fault attacks.

5.4 Online Implementations of ISAP

Considering the authenticated encryption mode of Isap, it appears that it does not make a big difference on its security against side-channel attacks if first all ciphertext blocks are produced and then absorbed by IsapMAC, or if block are absorbed by IsapMAC when they are ready, as long as the nonce is unique and an attacker does not act in an adaptive way. Hence, it is possible to implement the authenticated encryption of Isap online. However, this is not true for the decryption. In this section, we discuss the resulting implications of violating the necessity for Isap that the verification of the ciphertext has to be performed before the decryption of it starts.

5.4.1 Implementation Security

As discussed in Subsection 5.3, Isap's hardening/protection against key extraction via implementation attacks mainly relies on the usage of a hard to invert and leakage-resilient re-keying function during encryption and tag generation. In this scenario, IsapEnc and IsapMAC are not necessarily required to be called in order, and consequently, an online implementation of Isap also retains these properties. However, all confidentiality of decrypted plaintexts in the presence of implementation attacks can be lost if the tag verification is not completed before the decryption of the ciphertext starts.

5.4.2 State Size

Implementing Isap in an online manner requires that the states for IsapEnc and IsapMAC are instantiated simultaneously. While this is not an issue for software implementations, it technically could result in a noticeable area increase for hardware implementations. However, when taking a closer look at the finalization of IsapMAC in Figure 1, one can observe that the two-pass version of Isap already needs to temporarily store an additional $n - k$ bits during the re-keying operation that could be reused to hold a second state during plain-/ciphertext processing. On top of that, an online implementation of Isap does not need to store the nonce. This saves an additional k bits of registers and thus, roughly, evens out the register requirements of both implementation options.

5.4.3 Runtime

Software implementations of Isap's Keccak instantiations can suffer from the fact that the algorithmic description of Keccak-p[400] is based on 16-bit lanes. 32-bit and 64-bit CPUs can hence not always fully utilize their larger registers, which leads to a performance degradation on these platforms. However, performing IsapEnc in parallel with IsapMAC allows for two permutation calls on both states in parallel. More concretely, one could instantiate both Keccak-p[400] states (each $5 \times 5 \times 16$ bits) as one larger state ($5 \times 5 \times 32$ bits) and then perform the encryption of M_i to C_i, as well as the absorption of C_{i-1} concurrently. This could result in a speedup of about 50 % for the Keccak instantiations of Isap on 32- or 64-bit processors, and in a similar spirit, to a 50 % speedup for the Ascon instantiations of Isap on more powerful CPUs that can operate on 128-bit registers.

Acknowledgments. The authors want to thank Mario Werner and Robert Schilling for helpful discussions and providing their hardware descriptions of Keccak.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 681402), the Austrian Research Promotion Agency (FFG) via the K-project DeSSnet, which is funded in the context of COMET – Competence Centers for Excellent Technologies by BMVIT, BMWFW, Styria and Carinthia, the Austrian Research Promotion Agency (FFG) via the project ESPRESSO, which is funded by the province of Styria and the Business Promotion Agencies of Styria and Carinthia, and the Austrian Science Fund (FWF): J 4277-N38. Additionally, Bart Mennink is supported by a postdoctoral fellowship from the Netherlands Organisation for Scientific Research (NWO) under Veni grant 016.Veni.173.017.

References

[ABD$^+$16] Elena Andreeva, Andrey Bogdanov, Nilanjan Datta, Atul Luykx, Bart Mennink, Mridul Nandi, Elmar Tischhauser, and Kan Yasuda. COLM v1. CAESAR, second choice for defense in depth, https://competitions.cr.yp.to/caesar-submissions.html, 2016.

[BDH$^+$14] Sonia Belaïd, Fabrizio De Santis, Johann Heyszl, Stefan Mangard, Marcel Medwed, Jörn-Marc Schmidt, François-Xavier Standaert, and Stefan Tillich. Towards fresh re-keying with leakage-resilient PRFs: cipher design principles and analysis. *J. Cryptographic Engineering*, 4(3):157–171, 2014.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *EUROCRYPT '97*, volume 1233 of *LNCS*, pages 37–51. Springer, 1997.

[BDN$^+$13] Begül Bilgin, Joan Daemen, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Gilles Van Assche. Efficient and first-order DPA resistant implementations of Keccak. In Aurélien Francillon and Pankaj Rohatgi, editors, *CARDIS 2013*, volume 8419 of *LNCS*, pages 187–199. Springer, 2013.

[BDPV07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. Ecrypt Hash Workshop 2007, May 2007.

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, 2008.

[BDPV11] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The Keccak reference (version 3.0). https://keccak.team/files/Keccak-reference-3.0.pdf, 2011.

[Ber08] Daniel J. Bernstein. ChaCha, a variant of Salsa20, January 2008.

[BGP$^+$19] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT, a leakage-resist AEAD mode for high physical security applications, Nov. 2019.

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In *ASIACRYPT 2017*, volume 10624 of *LNCS*, pages 693–723. Springer, 2017.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, 2000.

[BPPS17] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On leakage-resilient authenticated encryption with decryption leakages. *IACR Transactions on Symmetric Cryptology*, 2017(3):271–293, Sep. 2017.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO '97*, volume 1294 of *LNCS*, pages 513–525. Springer, 1997.

[CAE14] CAESAR committee. CAESAR: Competition for authenticated encryption: Security, applicability, and robustness, 2014.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO '99*, volume 1666 of *LNCS*, pages 398–412. Springer, 1999.

[DEG$^+$18] Christoph Dobraunig, Maria Eichlseder, Hannes Gross, Stefan Mangard, Florian Mendel, and Robert Primas. Statistical ineffective fault attacks on masked AES with fault countermeasures. In Thomas Peyrin and Steven D. Galbraith, editors, *ASIACRYPT 2018*, volume 11273 of *LNCS*, pages 315–342. Springer, 2018.

[DEK$^+$16] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Victor Lomné, and Florian Mendel. Statistical fault attacks on nonce-based authenticated encryption schemes. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 369–395, 2016.

[DEK$^+$18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: exploiting ineffective fault inductions on symmetric cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):547–572, 2018.

[DEM$^+$17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP – towards side-channel secure authenticated encryption. *IACR Transactions on Symmetric Cryptology*, 2017(1):80–105, 2017.

[DEM$^+$19] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP v2.0. Submission to NIST Lightweight Cryptography, 2019.

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. CAESAR, first choice for lightweight applications (resource constrained environments), https://competitions.cr.yp.to/caesar-submissions.html, 2016.

[DEMS19] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Submission to NIST Lightweight Cryptography, 2019.

[DFH$^+$16] Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards sound fresh re-keying with hard (physical) learning problems. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016*, volume 9815 of *LNCS*, pages 272–301. Springer, 2016.

[DHVV18] Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. The design of xoodoo and xoofff. *IACR Transactions on Symmetric Cryptology*, 2018(4):1–38, Dec. 2018.

[DKM$^+$15] Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. Towards fresh and hybrid re-keying schemes with beyond birthday security. In Naofumi Homma and Marcel Medwed, editors, *CARDIS 2015*, volume 9514 of *LNCS*, pages 225–241. Springer, 2015.

[DM19a] Christoph Dobraunig and Bart Mennink. Leakage resilience of the duplex construction. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019*, volume 11923 of *LNCS*, pages 225–255. Springer, 2019.

[DM19b] Christoph Dobraunig and Bart Mennink. Leakage Resilience of the ISAP Mode: a Vulgarized Summary, 2019. NIST Lightweight Cryptography Workshop 2019.

[DM20] Christoph Dobraunig and Bart Mennink. Security of the suffix keyed sponge. *IACR Transactions on Symmetric Cryptology*, 2019(4):223–248, Jan. 2020.

[DMMP18] Christoph Dobraunig, Stefan Mangard, Florian Mendel, and Robert Primas. Fault attacks on nonce-based authenticated encryption: Application to Keyak and Ketje. In Carlos Cid and Michael J. Jacobson Jr., editors, *SAC 2018*, volume 11349 of *LNCS*, pages 257–277. Springer, 2018.

[DMP20] Christoph Dobraunig, Bart Mennink, and Robert Primas. Exploring the golden mean between leakage and fault resilience and practice. Cryptology ePrint Archive, Report 2020/200, 2020. https://eprint.iacr.org/2020/200.

[DMV17] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-state keyed duplex with built-in multi-user support. In *ASIACRYPT 2017*, volume 10625 of *LNCS*, pages 606–637. Springer, 2017.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS 2008*, pages 293–302, 2008.

[DP10] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In *CRYPTO 2010*, volume 6223 of *LNCS*, pages 21–40. Springer, 2010.

[DPVR00] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. The NOEKEON block cipher, 2000. Nessie Proposal.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[DRB$^+$16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with $d+1$ shares in hardware. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 194–212. Springer, 2016.

[FJLT13] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. Fault attacks on AES with faulty ciphertexts only. In Wieland Fischer and Jörn-Marc Schmidt, editors, *FDTC 2013*, pages 108–118. IEEE Computer Society, 2013.

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In *CHES 2012*, volume 7428 of *LNCS*, pages 213–232. Springer, 2012.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GGNPS13] Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert. Block ciphers that are easier to mask: How far can we go? In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES 2013*, volume 8086 of *LNCS*, pages 383–399. Springer, 2013.

[GLSV14] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-designs: Bitslice encryption for efficient masked software implementations. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 18–37. Springer, 2014.

[GMK17] Hannes Gross, Stefan Mangard, and Thomas Korak. An efficient side-channel protected AES implementation with arbitrary protection order. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 95–112. Springer, 2017.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In *CHES'99*, volume 1717 of *LNCS*, pages 158–172. Springer, 1999.

[GPPS19] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards lightweight side-channel security and the leakage-resilience of the duplex sponge. Cryptology ePrint Archive, Report 2019/193, 2019.

[GSWY20] Chun Guo, François-Xavier Standaert, Weijia Wang, and Yu Yu. Efficient side-channel secure message authentication with better bounds. *IACR Transactions on Symmetric Cryptology*, 2019(4):23–53, Jan. 2020.

[GWDE15] Hannes Gross, Erich Wenger, Christoph Dobraunig, and Christoph Ehrenhöfer. Suit up! made-to-measure hardware implementations of ascon. Cryptology ePrint Archive, Report 2015/034, 2015.

[JNPS16] Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. Deoxys v1.41. CAESAR, first choice for defense in depth, https://competitions.cr.yp.to/caesar-submissions.html, 2016.

[Jut01] Charanjit S. Jutla. Encryption modes with almost free message integrity. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 529–544. Springer, 2001.

[Jut08] Charanjit S. Jutla. Encryption modes with almost free message integrity. *J. Cryptology*, 21(4):547–578, 2008.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.

[KJJR11] Paul C. Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *J. Cryptographic Engineering*, 1(1):5–27, 2011.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO '96*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

[KR16] Ted Krovetz and Phillip Rogaway. OCB (v1.1). CAESAR, for high-performance applications, https://competitions.cr.yp.to/caesar-submissions.html, 2016.

[Kra01] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 310–331. Springer, 2001.

[MPR$^+$11] Marcel Medwed, Christophe Petit, Francesco Regazzoni, Mathieu Renauld, and François-Xavier Standaert. Fresh re-keying II: securing multiple parties against side-channel and fault attacks. In Emmanuel Prouff, editor, *CARDIS 2011*, volume 7079 of *LNCS*, pages 115–132. Springer, 2011.

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 279–296. Springer, 2010.

[MSJ12] Marcel Medwed, François-Xavier Standaert, and Antoine Joux. Towards super-exponential side-channel security with efficient leakage-resilient prfs. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 193–212. Springer, 2012.

[MSNF16] Marcel Medwed, François-Xavier Standaert, Ventzislav Nikov, and Martin Feldhofer. Unknown-input attacks in the parallel setting: Improving the security of the CHES 2012 leakage-resilient PRF. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 602–623, 2016.

[Nat15a] National Institute of Standards and Technology. FIPS PUB 180-4: Secure hash standard (SHS). Federal Information Processing Standards Publication 180-4, August 2015.

[Nat15b] National Institute of Standards and Technology. FIPS PUB 202: SHA-3 Standard: Permutation-based hash and extendable-output functions. Federal Information Processing Standards Publication 202, U.S. Department of Commerce, 8 2015.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 462–482. Springer, 2009.

[PRC12] Gilles Piret, Thomas Roche, and Claude Carlet. PICARO - A block cipher allowing efficient higher-order side-channel resistance. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS 2012*, volume 7341 of *LNCS*, pages 311–328. Springer, 2012.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *CCS 2015*, pages 96–108. ACM, 2015.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *CCS 2002*, pages 98–107. ACM, 2002.

[SPY$^+$10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 99–134. Springer, 2010.

[SPY13] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013*, volume 8042 of *LNCS*, pages 335–352. Springer, 2013.

[TS14] Mostafa M. I. Taha and Patrick Schaumont. Side-channel countermeasure for SHA-3 at almost-zero area overhead. In *HOST 2014*, pages 93–96. IEEE Computer Society, 2014.

[WP16] Hongjun Wu and Bart Preneel. AEGIS: a fast authenticated encryption algorithm (v1.1). CAESAR, for high-performance applications, https://competitions.cr.yp.to/caesar-submissions.html, 2016.

[Wu15] Hongjun Wu. ACORN: A lightweight authenticated cipher (v2). CAESAR, second choice for lightweight applications (resource constrained environments), https://competitions.cr.yp.to/caesar-submissions.html, 2015.

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In *CCS 2010*, pages 141–151. ACM, 2010.

A Specification of Permutations

A.1 Specification of Keccak-$p[400]$

Keccak-$p[400]$ is specified in [BDPV11, Nat15b]. In the following, we briefly recall the permutation's state geometry and the round function's five steps:

$$\mathrm{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta \,.$$

The 400-bit state of Keccak-$p[400]$ is labeled as a three-dimensional bit array $a[x][y][z]$ with $0 \leq x < 5$, $0 \leq y < 5$, and $0 \leq z < 16$. This state is mapped to the bitstring S as $S[16(5y + x) + z] = a[x][y][z]$, where the outer part for rate r corresponds to the bit positions $S[0, \ldots, r - 1]$.

The steps are defined by

$$\begin{aligned}
\theta: &\quad a[x][y][z] \leftarrow a[x][y][z] \oplus \bigoplus_{y'=0}^{4} a[x-1][y'][z] \oplus \bigoplus_{y'=0}^{4} a[x+1][y'][z-1]\,,\\
\rho: &\quad a[x][y][z] \leftarrow a[x][y][z - (t+1)(t+2)/2], \text{ with } t < 24 \text{ s.t. } \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^{t} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}\\
&\qquad\qquad \text{or } t = -1 \text{ if } x = y = 0,\\
\pi: &\quad a[x][y] \leftarrow a[x + 3y][x],\\
\chi: &\quad a[x] \leftarrow a[x] \oplus (a[x+1] \oplus 1) \cdot a[x+2]\,,\\
\iota: &\quad a \leftarrow a \oplus \mathrm{RC}[i_\mathrm{r}]\,,
\end{aligned}$$

where multiplications are over $\mathbb{F}_2$ (bitwise AND) and all index computations are modulo 5 (for x, y) or modulo 16 (for z). The round constants are $\mathrm{RC}[i_\mathrm{r}][x][y] = 0$ except for $\mathrm{RC}[i_\mathrm{r}][0][0][z] = \mathrm{rc}[j + 7i_\mathrm{r}]$ for all $z = 2^j - 1$, $0 \leq j \leq 4$, where $\mathrm{rc}[i]$ is specified by an LFSR with the primitive monomial $p(X) = X^8 + X^6 + X^5 + X^4 + 1$ and i gives the cycles starting from an initialized binary value of '1000000'. If Keccak-$p[400]$ is instantiated with n_r rounds, i_r ranges from $20 - n_\mathrm{r}$ to 19. For a more detailed description, we refer to [BDPV11, Nat15b].

A.2 Specification of Ascon-p

The following description of the Ascon-p permutation is adapted from the Ascon specification [DEMS16, DEMS19].

All members of the Ascon cipher suite operate on a state of 320 bits which they update with permutations p^a (a rounds) and p^b (b rounds). The 320-bit state S is divided into an outer part S_r of r bits and an inner part S_c of c bits, where the rate r and capacity $c = 320 - r$ depend on the Ascon variant.

For the description and application of the round transformations, the 320-bit state S is split into five 64-bit registers words x_i:

$$S = S_r \,\|\, S_c = x_0 \,\|\, x_1 \,\|\, x_2 \,\|\, x_3 \,\|\, x_4 \,.$$

Whenever S needs to be interpreted as a byte-array (or bitstring) for the sponge interface, this starts with the most significant byte (or bit) of x_0 as byte 0 and ends with the least significant byte (or bit) of x_4 as byte 39.

Table 6 lists the notation and symbols used in the following description.

Table 6: Notation used for Ascon's permutation.

p_C, p_S, p_L	constant-addition, substitution and linear layer of $p = p_L \circ p_S \circ p_C$
$x_0, \dots, x_4$	The five 64-bit words of the state S
$x_{0,i}, \dots, x_{4,i}$	Bit i, $0 \leq i < 64$, of words $x_0, \dots, x_4$, with $x_{\cdot,0}$ the rightmost bit (LSB)
$x \oplus y$	Bitwise XOR of 64-bit words or bits x and y
$x \odot y$	Bitwise AND of 64-bit words or bits x and y (denoted $x\,y$ in the ANF)
$\ominus x$	Bitwise NOT of 64-bit word or bit x
$x \ggg i$	Right-rotation (circular shift) by i bits of 64-bit word x

Isap uses Ascon's two 320-bit permutations p^a and p^b, as well as an additional variant reduced to one round, p^1. The permutations iteratively apply an SPN-based round transformation p that in turn consists of three steps p_C, p_S, p_L and differ only in the number of rounds:

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

For the description and application of the round transformations, the 320-bit state S is split into five 64-bit registers words x_i, $S = x_0 \,\|\, x_1 \,\|\, x_2 \,\|\, x_3 \,\|\, x_4$.

Addition of Constants

The constant addition step p_C adds a round constant c_r to register word x_2 of the state S in round i. Both indices r and i start from zero and we use $r = i$ for p^a and $r = i + a - b$ for p^b (see Table 7):

$$x_2 \leftarrow x_2 \oplus c_r \,.$$

Table 7: The round constants c_r used in each round i of p^a and p^b.

p^{12}	p^8	p^6	Constant c_r	p^{12}	p^8	p^6	Constant c_r
0			`00000000000000f0`	6	2	0	`0000000000000096`
1			`00000000000000e1`	7	3	1	`0000000000000087`
2			`00000000000000d2`	8	4	2	`0000000000000078`
3			`00000000000000c3`	9	5	3	`0000000000000069`
4	0		`00000000000000b4`	10	6	4	`000000000000005a`
5	1		`00000000000000a5`	11	7	5	`000000000000004b`

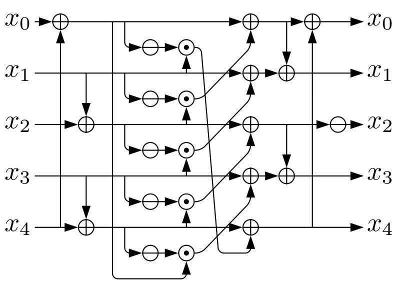
Substitution Layer

The substitution layer p_S updates the state S with 64 parallel applications of the 5-bit S-box $\mathcal{S}(x)$ defined in 2a to each bit-slice of the five registers $x_0 \dots x_4$. It is typically implemented in bitsliced form with operations performed on the 64-bit words.

Linear Diffusion Layer

The linear diffusion layer p_L provides diffusion within each 64-bit register word x_i. It applies a linear function $\Sigma_i(x_i)$ defined in 2b to each word x_i:

$$x_i \leftarrow \Sigma_i(x_i), \quad 0 \leq i \leq 4.$$



(a) Ascon's 5-bit S-box $\mathcal{S}(x)$

$$\begin{aligned}
x_0 &\leftarrow \Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)\\
x_1 &\leftarrow \Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)\\
x_2 &\leftarrow \Sigma_2(x_2) = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6)\\
x_3 &\leftarrow \Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)\\
x_4 &\leftarrow \Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)
\end{aligned}$$

(b) Ascon's linear layer with 64-bit functions $\Sigma_i(x_i)$

Figure 2: Ascon's substitution layer and linear diffusion layer.