ISAP

v2.0

Submission to NIST

Christoph Dobraunig   Maria Eichlseder   Stefan Mangard
Florian Mendel   Bart Mennink   Robert Primas
Thomas Unterluggauer

May 17, 2021

isap@iaik.tugraz.at
https://isap.iaik.tugraz.at

Contents

1. Introduction

Isap is a family of nonce-based authenticated ciphers with associated data (AEAD) designed with a focus on robustness against passive side-channel attacks. All Isap family members are permutation-based designs that combine variants of the sponge-based Isap mode with one of several published lightweight permutations.

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 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 was published at FSE 2017 [DEM+17] as an approach to provide inherent security against differential power analysis (DPA) attacks. DPA attacks constitute the most powerful class of passive side-channel attacks in practice and work by accumulating information of the secret key by observing multiple encryptions (or decryptions) of different inputs. By integrating a sponge-based re-keying function in an encrypt-then-MAC construction to always use fresh keys for processing new data, Isap significantly increases the robustness against DPA and related attacks. Compared to the original published proposal [DEM+17], we have improved the mode to address a broader spectrum of implementation attacks, including fault attacks.

We recommend four instances: Isap-A-128a, Isap-K-128a, Isap-A-128 and Isap-K-128. All instances are designed to provide 128-bit security against cryptanalytic attacks as well as inherent security against certain classes of side-channel attacks. Two instances apply the 320-bit Ascon-p permutation [DEMS16] introduced by Ascon, the primary recommendation for lightweight authenticated encryption in the final CAESAR portfolio. We want to emphasize that both Ascon and Isap are Finalists of NIST's lightweight crypto standardization process. So, implementations of the permutation Ascon-p enable the flexibility of using the functionalities provided by Ascon, Isap-A-128a, and Isap-A-128. The other two instances employ the 400-bit Keccak-$p[400]$ permutation [BDPV11e; Nat15], which is a smaller sibling of the SHA-3 permutation used among others by Ketje Sr [BDP+16a], a round-3 candidate in the CAESAR competition.

The security and efficiency of the underlying permutations are critical for the overall design. Both Keccak-$p[400]$ and Ascon-p have been thoroughly analyzed and benchmarked by the cryptographic community in the last years, and both provide a comfortable security margin as well as excellent lightweight implementation characteristics.

2. Specification

Isap is a family of sponge-based authenticated encryption schemes using an n-bit permutation $\mathcal{P}$. The Isap instances are parameterized by the security parameter k, which defines the cryptographic security level of k bits and specifies the size of the key, tag, and nonce. The instances are further parameterized by $(s_{\text{H}}, s_{\text{B}}, s_{\text{E}}, s_{\text{K}})$, which specify the number of rounds the permutation $\mathcal{P}$ is evaluated. We denote the resulting permutations as p_{H}, p_{B}, p_{E}, and p_{K}. In addition, it is parameterized by two rate values r_{H} and r_{B}. Rate r_{H} will be applied for states in the unkeyed sponge and in the keyed sponge that are unlikely to be evaluated more than once, which means that it may be reasonably large as leakage will be limited. Rate r_{B} will be 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 $r_{\text{H}} = n - 2k$ and $r_{\text{B}} = 1$. A member of the Isap family for permutation $\mathcal{P}$ with round parameters $s_{\text{H}}, s_{\text{B}}, s_{\text{E}}, s_{\text{K}}$, rates $r_{\text{H}}, r_{\text{B}}$, and security parameter k is denoted as

$$\text{Isap-}\mathcal{P}\text{-}^{r_{\text{H}},r_{\text{B}}}_{s_{\text{H}},s_{\text{B}},s_{\text{E}},s_{\text{K}}}\text{-}k\,.$$

Authenticated encryption $\mathcal{E}$ and authenticated decryption $\mathcal{D}$ are described in Algorithm 1 and 2 and depicted in Figure 2.1a and 2.1b. Isap can be seen as an encrypt-then-MAC design, where the same k-bit key is used for encryption and message authentication. The individual encryption algorithm IsapEnc and message authentication code IsapMac employed in $\mathcal{E}$ and $\mathcal{D}$ are further discussed in Section 2.2 and Section 2.3. Both internally use a re-keying function IsapRk that will first be discussed in Section 2.1. Table 2.1 summarizes the parameters and notation used in the specification of Isap.

Table 2.1.: Notation used for Isap's interface and mode.

K, N, T	Secret key K, nonce N, and tag T, all of $k = 128$ bits
M, C, A	Plaintext M, ciphertext C, associated data A (in r_{H}-bit blocks M_i, C_i, A_i)
$\perp$	Error, verification of authenticated ciphertext failed
$\lvert x \rvert$	Length of the bitstring x in bits
$x \parallel y$	Concatenation of bitstrings x and y
$x \oplus y$	Xor of bitstrings x and y
$S = S_r \parallel S_c$	The n-bit sponge state S with r-bit outer part S_r and c-bit inner part S_c
$x = \lceil x \rceil^k \parallel \lfloor x \rfloor_k$	Bitstring x split into the first k bits $\lceil x \rceil^k$ (MSB) and last k bits $\lfloor x \rfloor_k$ (LSB)

2.1. 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 flag $f \in \{\textsc{enc}, \textsc{mac}\}$, and a k-bit string Y, and transforms it into a subkey K^* of size z bits. Here, z and the initial value IV are determined by the flag f:

$$(\mathrm{IV}, z) = \begin{cases} (\mathrm{IV}_{\mathrm{KE}}, n-k)\,, & \text{if } f = \textsc{enc}\,, \\ (\mathrm{IV}_{\mathrm{KA}}, k)\,, & \text{if } f = \textsc{mac}\,. \end{cases}$$

The function is described in Algorithm 4 and depicted in Figure 2.1c. 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, which happen at rate r_{B}. We remind the reader of the fact that r_{B} is small.

2.2. 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 2.1d. It first calls IsapRk for encryption using the flag $f = \textsc{enc}$ to select the initial value $\mathrm{IV}_{\mathrm{KE}}$ and $z = n - k$ in order to derive a $(n-k)$-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.

2.3. 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 depicted in Figure 2.1e. 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 $K_{\mathrm{A}}^* \parallel \lfloor S \rfloor_{n-k}$ 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.

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
$T \leftarrow \textsc{IsapMac}(K, N, A, C)$
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
$T' \leftarrow \textsc{IsapMac}(K, N, A, C)$
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 $M \,\|\, 0^{-|M| \bmod r_{\text{H}}}$
$K^*_{\text{E}} \leftarrow \textsc{IsapRk}(K, \textsc{enc}, N)$
$S \leftarrow K^*_{\text{E}} \,\|\, N$
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 $\textsc{IsapRk}(K, f, Y)$

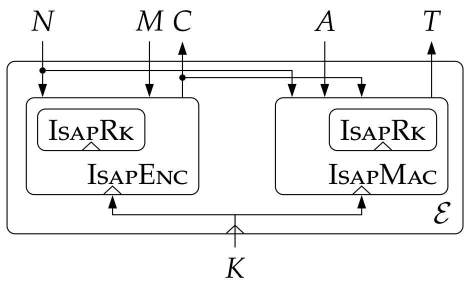
Input: key $K \in \{0,1\}^k$,
flag $f \in \{\textsc{enc}, \textsc{mac}\}$,
string $Y \in \{0,1\}^k$
Output: session key $K^* \in \{0,1\}^z$

Initialization
if $f = \textsc{enc}$ **then**
$(\text{IV}, z) \leftarrow (\text{IV}_{\text{KE}}, n - k)$
else
$(\text{IV}, z) \leftarrow (\text{IV}_{\text{KA}}, k)$
$Y_1 \ldots Y_w \leftarrow r_{\text{B}}$-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^*

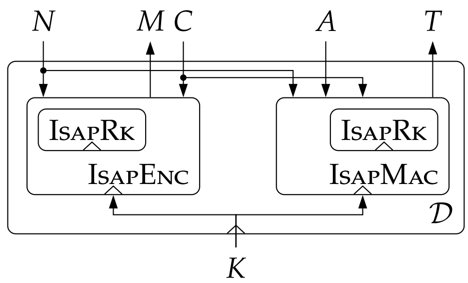
Algorithm 5 $\textsc{IsapMac}(K, N, A, C)$

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 $T \in \{0,1\}^k$

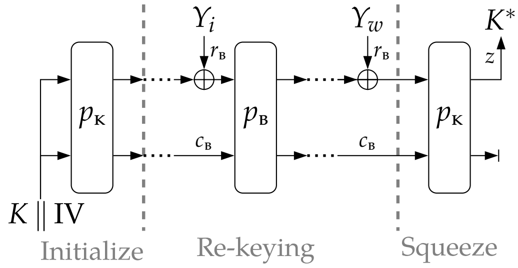
Initialization
$A_1 \ldots A_s \leftarrow r_{\text{H}}$-bit blocks of $A \| 1 \| 0^{-|A|-1 \bmod r_{\text{H}}}$
$C_1 \ldots C_t \leftarrow r_{\text{H}}$-bit blocks of $C \| 1 \| 0^{-|C|-1 \bmod r_{\text{H}}}$
$S \leftarrow N \,\|\, \text{IV}_{\text{A}}$
$S \leftarrow p_{\text{H}}(S)$
Absorbing Associated 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
$K^*_{\text{A}} \leftarrow \textsc{IsapRk}(K, \textsc{mac}, \lceil S \rceil^k)$
$S \leftarrow p_{\text{H}}(K^*_{\text{A}} \,\|\, \lfloor S \rfloor_{n-k})$
$T \leftarrow \lceil S \rceil^k$
return T



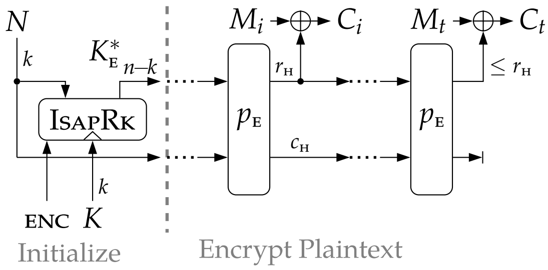
(a) Encryption $\mathcal{E}(K, N, A, M)$



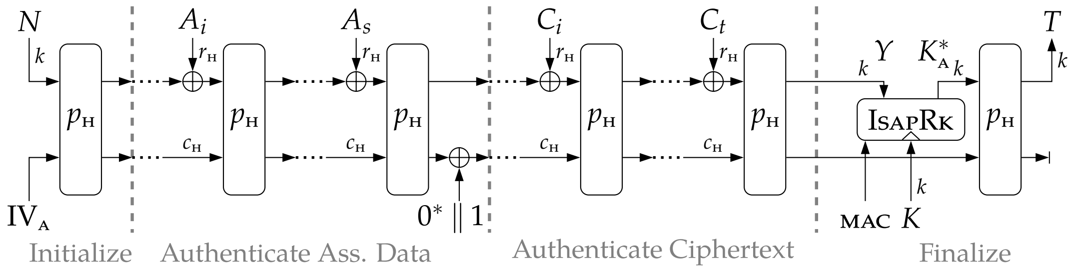
(b) Decryption $\mathcal{D}(K, N, A, C, T)$



(c) IsapRk , with $(\mathrm{IV}, z) = (\mathrm{IV}_{\mathrm{KE}}, n-k)$ if $f = \textsc{enc}$ else $(\mathrm{IV}_{\mathrm{KA}}, k)$ if flag $f = \textsc{mac}$



(d) IsapEnc



(e) IsapMac

Figure 2.1.: Isap's encryption $\mathcal{E}(K, N, A, M)$ and decryption $\mathcal{D}(K, N, A, C, T)$ algorithms composing IsapEnc stream encryption, IsapMac MAC, and IsapRk re-keying.

2.4. Permutations

Isap is instantiated with either the 320-bit permutation used in Ascon [DEMS16; DEMS21] or the 400-bit permutation Keccak-$p[400]$ [BDPV11e; Nat15]. A detailed specification of Ascon-p, including the state layout and specification of the inner and outer state parts, can be found in [DEMS21]. Keccak-$p[400]$ is specified in NIST FIPS PUB 202 [Nat15]. We provide a short description of the two permutations in Appendix A and B.

2.5. Recommended Parameter Sets

Table 2.2 summarizes the recommended parameter sets for Isap. The members Isap-A-128a and Isap-K-128a, where A and K refer to the underlying cryptographic permutation, specify a choice of parameters for fast implementations based on our design rationale and security analysis given in Chapter 4 and Chapter 5. We also specify a more conservative choice of parameters in Isap-A-128 and Isap-K-128. The recommended parameters (Table 2.2) are ordered starting with the primary recommendation, followed by the second, third, and fourth. We recall that $s_{\mathrm{H}}, s_{\mathrm{B}}, s_{\mathrm{E}}, s_{\mathrm{K}}$ denote the number of rounds of permutations $p_{\mathrm{H}}, p_{\mathrm{B}}, p_{\mathrm{E}}, p_{\mathrm{K}}$.

Table 2.2.: 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-A-128a	Ascon-p	128	320	64	1	12	1	6	12
Isap-K-128a	Keccak-$p[400]$	128	400	144	1	16	1	8	8
Isap-A-128	Ascon-p	128	320	64	1	12	12	12	12
Isap-K-128	Keccak-$p[400]$	128	400	144	1	20	12	12	12

The initial values IV_{A}, $\mathrm{IV}_{\mathrm{KA}}$, and $\mathrm{IV}_{\mathrm{KE}}$, which serve as domain separation between the different algorithms, are specified in Table 2.3. 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-A-128 and Isap-A-128a, the resulting IVs have a length of 192 bits, while those for Isap-K-128 and Isap-K-128a are 272 bits long.

Table 2.3.: Initial values for Isap instances in hex notation.

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

2.6. On Hash Functions using Ascon-p or Keccak-$p[400]$

Since Isap is based on either Ascon-p or Keccak-$p[400]$, it lends itself to pairing with already specified hash functions using the same permutations. In the case of Isap-A-128a and Isap-A-128, we suggest a pairing with the hash function AsconHash specified in the Ascon v1.2 design document [DEMS21]. Implementations of AsconHash can be found at `https://github.com/ascon/ascon-c`. In the case of Isap-K-128a, we suggest a pairing with sponge[Keccak-$p[400, 16]$, pad10*1, 144]$(M \| 01, 256)$, and for Isap-K-128 a pairing with sponge[Keccak-$p[400, 20]$, pad10*1, 144]$(M \| 01, 256)$, following the specifications in NIST FIPS PUB 202 [Nat15]. Implementations of Keccak-based hash functions can be found at `https://github.com/XKCP/XKCP`.

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 3.1. Note that as usual, a security loss by a small constant factor is expected.

Table 3.1.: Security claims for recommended parameter configurations of Isap.

Requirement	Security in bits			
	Isap-A-128a	Isap-K-128a	Isap-A-128	Isap-K-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 3.1, 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 secure against simple power analysis (SPA) including template attacks. Furthermore, Isap is designed to improve robustness against other implementation attacks, including certain fault attacks. We provide a more detailed discussion on these non-cryptographic claims in Chapter 6.

4. Design Rationale

The main goal of Isap is to provide robustness against relevant implementation attacks at a very low hardware area footprint. Each variant uses a lightweight permutation as the single primitive, which can be re-used to support hashing and does not require the implementation of an inverse. While mechanisms to counteract side-channel attacks and in particular DPA within the cipher itself (e.g., masking) lead to significant overheads and increase drastically with the protection order, approaches based on fresh re-keying lead to much lower overheads. Isap is designed to provide robustness against DPA during authenticated encryption and authenticated decryption and to provide this increased robustness at a much more lightweight footprint than classical countermeasures. In addition, with the choice of the permutations and details of the mode, we aim to also strengthen robustness against fault attacks and to allow for an easy integration of protection mechanisms against SPA.

In the following, we first recall the design rationale for Isap's mode of operation and its algorithms IsapRk, IsapEnc, and IsapMac as introduced in the FSE 2017 paper [DEM+17] in Section 4.1 to Section 4.5. In Section 4.6, we discuss the choice of the permutations in Isap. Finally, in Section 4.7, we discuss differences of the proposal at hand compared to the original Isap mode.

4.1. Robustness of the Mode against DPA

For discussing the robustness of our scheme against differential power analysis (DPA), we prefer to give a more general, high-level view on our mode in Algorithm 6 to better extract the underlying idea and show the fresh re-keying roots of our scheme. Here, we essentially use the same assumptions and requirements as other re-keying schemes: g_1, g_2 must be two domain-separated (DPA and SPA) secure re-keying functions and the implementations of *ENC*, *DEC*, and *MAC* must be secure against SPA attacks when processing arbitrarily long messages. There are no requirements on the implementation of the hash function H, since it processes only publicly known data.

To achieve robustness against DPA, our authenticated encryption mode in Algorithm 6 incorporates a re-keying approach in an efficient encrypt-then-MAC scheme. The encrypt-then-MAC approach has become increasingly popular for designing authenticated encryptions schemes that withstand side-channel attacks [DEM+17; BPPS17; BMOS17; BGP+20]. While simple re-keying of both a MAC and an encryption scheme can only

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

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
$K_{\mathrm{E}}^* = g_1(N, K)$
$C = ENC_{N,K_{\mathrm{E}}^*}(M)$

Authentication
$Y = H(N, A, C)$
$K_{\mathrm{A}}^* = g_2(Y, K)$
$T = MAC_{K_{\mathrm{A}}^*}(Y)$
return C, T

Authenticated Decryption $\mathcal{D}(K, N, A, C, T)$

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 $M \in \{0,1\}^{|C|}$, or error $\perp$

Verification
$Y = H(N, A, C)$
$K_{\mathrm{A}}^* = g_2(Y, K)$
$T' = MAC_{K_{\mathrm{A}}^*}(Y)$
if $T \neq T'$ **return** $\perp$

Decryption
$K_{\mathrm{E}}^* = g_1(N, K)$
$M = DEC_{N,K_{\mathrm{E}}^*}(C)$
return M

Algorithm 6: Authenticated encryption and decryption procedures.

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. In the following, we give a detailed discussion on the DPA robustness of the two parts encryption/decryption and authentication/verification.

Encryption/Decryption. The encryption and decryption part can be seen as an instance of fresh re-keying [MSGR10; MPR+11]. Such schemes for fresh re-keying combine an SPA-secure encryption scheme *ENC* with a (DPA and SPA) secure re-keying function $g_1 : (N, K) \mapsto K_{\mathrm{E}}^*$. As the nonce N that is used to derive the session key K_{E}^* must not be repeated, fresh session keys are guaranteed and DPA on the encryption scheme *ENC* is effectively prevented.

However, for decryption, there is the threat that an adversary could exploit multiple decryptions with the same nonce N and thus the same session key K_{E}^*, and induce a DPA setting within the decryption *DEC* by using different data. To prevent such a DPA scenario in our mode, verification is performed prior to decryption. Decrypting two different messages (associated data and ciphertext) with the same K_{E}^* indicates either a collision of g_1 for fixed K (the probability of which is negligible in our use case), or two ciphertexts that have been encrypted using the same nonce N. Since we require unique nonces, the latter implies that either the ciphertexts are identical, or one ciphertext has been forged. If a cryptographically secure MAC is used, the probability of a successful forgery is negligible and thus the tag verification will fail for one of the ciphertexts with overwhelming probability.

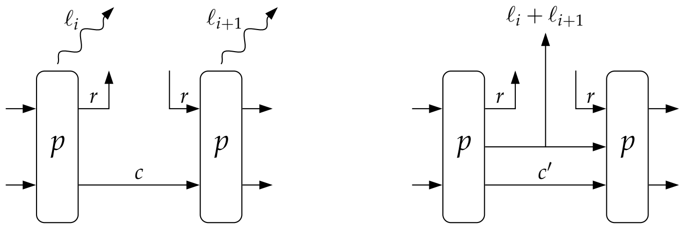
Authenticated ciphers require that no decrypted plaintext is released if tag verification fails. For protection against DPA attacks, we go one step further and require a failed verification to abort the authenticated decryption process, so that the decryption part *DEC* never starts. This ensures that the same session key K_{E}^* is never used to decrypt distinct ciphertexts with *DEC*. Therefore, the verification is responsible for precluding DPA attacks on the decryption.

Authentication/Verification. The authentication/verification shown in Algorithm 6 is based on a hash-then-MAC paradigm. Here, a session key K_{A}^* is first derived via a secure re-keying function g_2 from the hash value Y that is computed from the nonce N, associated data A, and ciphertext C using a cryptographic hash function H. Then, a message authentication code (*MAC*) is used to compute the tag T from the hash value Y and the session key K_{A}^*. This is similar to the construction of Pereira et al. [PSV15], who designed a leakage resilient MAC based on the hash-then-MAC paradigm as well. However, the main difference to our approach is that in [PSV15] a random nonce N is used to derive the session key in the re-keying function. This, however, cannot provide protection against DPA for multiple verifications. Contrary to that, we use the hash of the message $Y = H(N, A, C)$ to derive the session key K_{A}^* in order to securely allow multiple verifications while still providing protection against DPA.

In more detail, the MAC in Algorithm 6 computes the tag T using an independent session key K_{A}^* for every distinct message (N, A, and C). DPA on the *MAC* is thus prevented during the generation of the tag T as the same session key K_{A}^* is never used to authenticate distinct messages.

While the scheme by Pereira et al. [PSV15] also provides side-channel security during tag generation by the use of a unique nonce input N to the re-keying function, tag verification imposes different challenges. In fact, during tag verification one cannot rely on the uniqueness of the nonce anymore, because an attacker can usually modify the message (N, A, and C) to provoke multiple verifications with different data under the same nonce N and thus allow for a DPA scenario. However, the MAC in Algorithm 6 prevents such a DPA scenario on the session key K_{A}^*, since K_{A}^* is bound to the data it processes. Namely, as Y depends on the message (N, A, and C), the MAC session key $K_{\mathrm{A}}^* = g_2(Y, K)$ changes whenever the data changes. Adversaries cannot predictably influence Y due to the use of a cryptographic hash function H. This guarantees that the key K_{A}^* is unique for every new message as long as there is neither a collision in the hash function H nor in the re-keying function g_2. Thus, DPA on the session key K_{A}^* is effectively prevented during verification.

Note that collisions in the re-keying function g_2 or the hash function H may result in the same session key K_{A}^* being used in MAC computations of different messages, thus allowing for a DPA. Yet, collisions in g_2 depend on the secret key K_{A}^* and therefore inputs causing collisions in g_2 cannot be calculated off-line. In contrast, collisions in the hash value Y are directly observable and can be calculated off-line. The complexity of



(a) Leaking permutations (b) Sponge model

Figure 4.1.: Leakage of information in sponge-based constructions.

calculating collisions off-line is determined by the size of the hash. The generic complexity of finding a collision for an n-bit hash function is $2^{n/2}$. Hence, the size of the hash needs to be chosen depending on the potential threat of such an event, which depends on the concrete choice of functions for *MAC* and g_2, as we will see below.

4.2. Sponges and Side-Channel Leakage

While the mode of Section 4.1 ensures protection of the encryption *ENC*, decryption *DEC*, and message authentication code *MAC* against DPA, the primitives implementing *ENC*, *DEC*, and *MAC* still have to withstand SPA attacks. Moreover, SPA protection is also mandatory for the implementations of g_1 and g_2, in addition to the requirement that they provide protection against DPA. Besides dedicated countermeasures like, e.g., shuffling, also the order of the executed instructions and already the choice of the used algorithms for encryption/decryption and MAC play an important role for the robustness of the design against SPA.

Our choice for sponge-based designs is motivated by their suitability to model SPA leakage. Namely, the sponge parameters provide a convenient tool to argue on the side-channel robustness of keyed sponge constructions given bounded side-channel leakage of a single permutation evaluation.

For illustration, we model the leakage from a permutation p by allowing an adversary to learn a certain amount of the state between subsequent permutation calls as depicted in Figure 4.1. Here, we use ℓ to denote the information (in bits) that an attacker can learn about the state from the collected side-channel information. We do not care about how and where the leakage is created within p, but let the adversary account the learned information to either the input or the output state of p. Therefore, given two consecutive permutations p with leakages ℓ_i and ℓ_{i+1}, respectively, the maximum an adversary might learn about the state is $\ell_i + \ell_{i+1}$. This means that if each leakage ℓ_i, ℓ_{i+1} is bounded by λ bits and the adversary can optimally combine these two leakages, the adversary will learn at most 2λ bits of the state between the respective two permutation calls.

One can see the alternative view as a sponge with an adjusted capacity that copes with the leakage generated by the permutation: it has a capacity $c' = c - 2\lambda$ and rate $r' = r + 2\lambda$. The security level corresponding to capacity $c - 2\lambda$ is still guaranteed by the cryptographic properties of the permutation and the associated constrained-input constrained-output (CICO) problem [BDPV11a]. Sponge-based constructions can thus be considered to have bounded security loss for bounded leakage of the permutation.

Clearly, the challenge in practice is to build an implementation that bounds the leakage of p. Especially if many different types of devices have to use the same cryptographic algorithm, it might be infeasible to make any realistic assumptions about the leakage of p. Nevertheless, the advantage of the sponge-based construction is that besides standard SPA countermeasures like hiding and masking, the capacity is an additional and very natural security parameter that helps to increase the ability of a design to withstand side-channel attacks in practice.

Note that loading the same master key for the two IsapRk calls and applying the permutation may directly leak information about the key bits. To prevent this, implementations may store both expanded keys $p_{\mathrm{K}}(K \| \mathrm{IV}_{\mathrm{KA}})$ and $p_{\mathrm{K}}(K \| \mathrm{IV}_{\mathrm{KE}})$ or, alternatively, use slightly larger master keys K.

4.3. Design of IsapRk

Recall that g_1 and g_2 must offer strong DPA protection. In Isap, the role of g_1 and g_2 is taken by IsapRk of Figure 2.1c, which is called for different IVs. When setting the rate to 1 bit, the design is related to the classical GGM construction [GGM86] and can be seen as its sponge-based equivalent, similar to [TS14]. The basic idea in IsapRk is to make DPA infeasible by reducing the input data complexity accordingly. For this purpose, a secret state is constantly updated with small portions of public data by repeating two phases: (i) modifying the secret state according to the public data, and (ii) updating the state such that predictions on the future state based on the absorbed public data become infeasible.

Sponge-based constructions are an ideal choice to implement this basic idea as the rate directly influences the input data complexity for each permutation. IsapRk follows this approach and first initializes the internal state by applying the initial permutation p_{K} to the master key K that is padded with the IV. Then, IsapRk repeatedly injects 1 nonce bit into the state, each separated by a permutation call p_{B}. After full absorption of the nonce and finalization using again p_{K}, the session key is output. This working principle is similar to sponge instances of a prefix-MAC. While for general MAC computations the absorption rate can be as big as the state size [BDPV12; MRV15; DMV17], IsapRk uses a small absorption rate of 1 bit to limit the data complexity exploitable in a DPA.

On the other hand, since the rate is highly restricted, we can greatly reduce the number of rounds for p_{B} in the absorbing phase. Nevertheless, since we squeeze more bits and also want to ensure a good diffusion for the keybits, the number of rounds for p_K in the beginning and at the end of IsapRk has to be higher.

In terms of DPA security, permutation p_K and p_B will produce the leakages for two different public inputs, since we use two different IV paired with the master key and set the rate of the further absorption to one. Thus, IsapRk is 2-limiting per permutation call. This results in IsapRk being a secure re-keying function (regarding DPA) under the assumption that the combined leakage resulting from the processing of two different public inputs is bounded such that DPA on the secret state is infeasible. This is a common assumption also used in recent block-cipher based instantiations of the GGM construction by Faust et al. [FPS12] or the 2PRG primitive by Standaert et al. [SPY+10].

4.4. Design of IsapEnc

The encryption algorithm IsapEnc is an instance of fresh re-keying [MSGR10; MPR+11] that combines the secure re-keying function IsapRk in the initialization phase that re-keys a sponge-based stream cipher in the processing phase. To obtain cryptographic security on the processing part of IsapEnc, the nonce N must not be repeated for different plaintexts. This guarantees that the key stream is unpredictable and unique for different encryptions. The encryption part is initialized with a fresh session key paired with a unique nonce. Since the re-keying function IsapRk is hard to invert, even the recovery of the session key by means of implementation attacks does not allow for a recovery of the master key. Moreover, as long as the session key is never paired with different nonces, even DPA on the session key itself is prevented as well.

Furthermore, as a part of the authenticated encryption scheme Isap, IsapEnc remains secure against DPA also for multiple decryption of the same data, since it is guaranteed that this data is always decrypted under the same nonce: following the encrypt-then-MAC design of Isap, decryption only starts after authentication has succeeded. Hence, the authentication part precludes such DPA attacks on the decryption part.

4.5. Design of IsapMac

To get more insight in the design rationale behind IsapMac, consider the alternative illustration in Figure 4.2. The figure shows a normal sponge hash on input of N, A, C, padded in an injective way, followed by the hashing of a keyed value K_A^*. Note that IsapRk does not use frame bits for domain separation, but rather follows the approach of Ascon [DEMS21] and xors a single '1' to the inner part of the state. Although this reduces the capacity by one bit in the worst case, the practical security loss is considered to be negligible.

IsapMac reminds of the sponge-based suffix-MAC of Bertoni et al. [BDPV11a], where one puts a key after the message to obtain a secure MAC function, but the idea is not quite the same: compared to a "standard" sponge-based suffix-MAC, IsapMac uses a secure re-keying function g_2 to absorb the secret key K. While there are several options for g_2, e.g., the polynomial multiplication in [MSGR10], we use the function IsapRk as g_2. Unlike, e.g., polynomial multiplication, IsapRk is not a permutation for a fixed key, but we do not expect any negative consequences due to this, given that IsapRk ideally behaves like a pseudo-random function.

IsapMac prevents DPA on the tag computation in two ways. First, and as shown in Figure 2.1e, the MAC session key K_A^* is derived from the hash value Y and the MAC master key K via the secure re-keying function IsapRk (used as g_2), thus precluding DPA on K. Second, the design prevents DPA on the MAC session key K_A^* by binding it to the data being processed, thus leading to independent MAC session keys K_A^* for different data.

The only problem may pop up if there are collisions in Y: indeed, if two different inputs to IsapMac give the same hash value Y, they have the same MAC session key K_A^*. Yet, to perform a successful DPA, usually more than two traces will be needed to recover one fixed session key K_A^*. The expected number of multicollisions decreases drastically with the number of collisions: the generic complexity for finding a v-collision on a k-bit value is $\sqrt[v]{v! \cdot 2^{k(v-1)}} \approx 2^{k(v-1)/v}$ [STKT06]. The term is already quite high for small values of v, given that $k = 128$.

We remark that even though a DPA attack exploiting multi-collisions might be able to recover the session key K_A^* of IsapMac, this does not imply a key recovery attack on the master key K, since our re-keying function IsapRk is hard to invert.

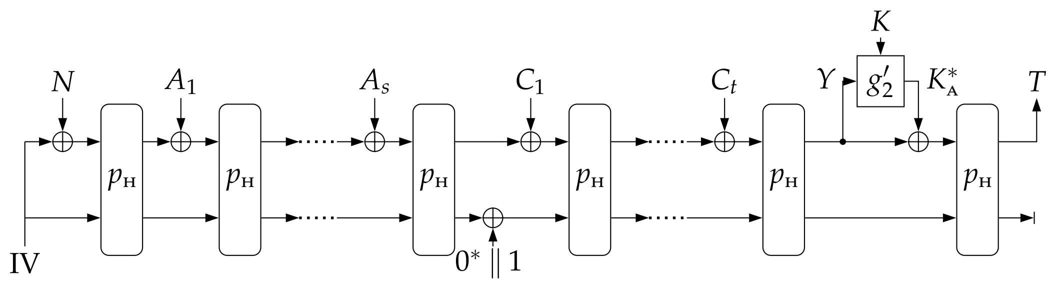


Figure 4.2.: Sketch of authentication/verification just using a sponge-based suffix-MAC. Here, $g_2' = g_2 \oplus \mathrm{id}$.

4.6. Choice of the Permutations

We have decided to use the permutation used in Ascon [DEMS16], together with the permutation Keccak-$p[400]$ [Nat15]. The selection of these permutations is motivated by the fact that:

- Ascon [DEMS16] is the first choice for the use case of lightweight applications (resource constrained environments) in CAESAR [CAE14] and is also a finalist in NIST's ongoing lightweight cryptography standardization process.
- Keccak-$p[400]$ is the smallest permutation specified in NIST's FIPS PUB 202 [Nat15] that allows Isap to be instantiated with a 128-bit security level.

The two permutations share several positive properties and similarities in their design rationale, including the bitsliced design with a weakly aligned linear layer providing strong diffusion, and the 5-bit S-boxes with a low algebraic degree that is useful for efficiently implementing masking countermeasures. There are, however, also differences and more complementary properties. Most notably, Ascon-p operates on a smaller state and uses larger 64-bit words internally instead of the 16-bit words in Keccak-$p[400]$, making it more suitable for efficiently interoperating with higher-end software platforms.

4.6.1. Choice of the Round Numbers for Ascon-p and Keccak-$p[400]$

We emphasize that we do not require ideal properties for the permutations $p_{\text{H}}, p_{\text{B}}, p_{\text{E}}, p_{\text{K}}$. Non-random properties, including but not limited to inside-out zero sum distinguishers, of the permutations p_{H}, p_{E}, p_{K}, and of course p_{B} are known and do not automatically afflict the claimed security properties of the entire encryption algorithm. For a detailed security analysis of Isap, refer to Chapter 5.

Parameters for IsapRk and IsapEnc. Both IsapRk and IsapEnc are keyed sponge-based constructions and according to recent results [BDPV12; GPT15; MRV15], one could apply full-state absorbtion (with empty capacity) in the absorbing phase, while in the squeezing phase a minimum of 128 bits capacity is needed to achieve a security level of 128 bits.

However, since we also have to bear side-channel attacks in mind, we set the rate r_{B} to 1 bit for all instances in the absorption phase for IsapRk, which makes the scheme 2-limiting per permutation call. This, together with the large capacity of the resulting construction, significantly increases the effort for an attacker trying to combine the leakage of two consecutive permutation calls as discussed in Section 4.2.

For IsapEnc, we set the capacity to 256 bits for all instances such that the resulting rate of $n - 256$ bits matches the rate of IsapMac. Thus, we obtain a rate of $r_{\text{H}} = 64$ bits for Ascon-p and of $r_{\text{H}} = 144$ bits for Keccak-$p[400]$ permutation.

Table 4.1.: Summary of parameters for Isap (see also Table 2.2).

(a) Parameters for IsapEnc (specified in Section 2.2, Algorithm 3, Figure 2.1d)

Name	Permutation $\mathcal{P}$	Rate r_{H}	Rounds s_{E}
Isap-A-128a	Ascon-p	64	6
Isap-K-128a	Keccak-$p[400]$	144	8
Isap-A-128	Ascon-p	64	12
Isap-K-128	Keccak-$p[400]$	144	12

(b) Parameters for IsapRk (specified in Section 2.1, Algorithm 4, Figure 2.1c)

Name	Permutation $\mathcal{P}$	Rate r_{B}	Rounds s_{B}	Rounds s_{K}
Isap-A-128a	Ascon-p	1	1	12
Isap-K-128a	Keccak-$p[400]$	1	1	8
Isap-A-128	Ascon-p	1	12	12
Isap-K-128	Keccak-$p[400]$	1	12	12

(c) Parameters for IsapMac (specified in Section 2.3, Algorithm 5, Figure 2.1e)

Name	Permutation $\mathcal{P}$	Rate r_{H}	Rounds s_{H}
Isap-A-128a	Ascon-p	64	12
Isap-K-128a	Keccak-$p[400]$	144	16
Isap-A-128	Ascon-p	64	12
Isap-K-128	Keccak-$p[400]$	144	20

For the number of rounds, Ascon [DEMS16] and the CAESAR candidate Keyak [BDP+16b] serve as orientation for our choices Isap-A-128 and Isap-K-128. We consider 12 rounds to be sufficient to create an unpredictable key-stream during the squeezing phase in the encryption for both variants. Moreover, 12 rounds also provide a clear separation between the single-bit injections during the absorption in IsapRk, so that partially known/leaked information about the internal secret state is hard to combine over consecutive permutation calls.

For the fast versions Isap-K-128a and Isap-A-128a, we follow the inspiration of donkeySponge and monkeyDuplex [BDPV12] and significantly reduce the number of rounds in the re-keying and encryption function of the scheme. The CAESAR candidate Ketje [BDP+16a] serves as inspiration for the version Isap-K-128a. Similar to Ketje Sr, only one round separates the absorption of the one bit elements using Keccak-$p[400]$ in the re-keying function. For the outer part we orient the number of rounds on the "stride" permutation call of Ketje Sr, which has 6 rounds. However, in contrast to Ketje Sr, we decided to use 8 rounds to add an additional security margin of 2 rounds to the scheme, since we return more bits in the end. For a similar reason, we also use 8 rounds of Keccak-$p[400]$ for the encryption function.

For the variant Isap-A-128a, we also reduce the number of rounds in the re-keying function to 1 in the absorbing phase and use 6 rounds in the initialization and finalization. This is the same number of rounds as used in Ascon-128 in the data processing phase. For the same reason we also use 6 rounds for the encryption function, which uses the same rate as Ascon-128.

Parameters for IsapMac. Since we aim for 128-bit security, we use IsapMac in all instances with a capacity of 256 bits, while allowing the remaining $n - 256$ bits as rate. Thus, we have a rate r_{H} of 64 and 144 bits for the Ascon-p and Keccak-$p[400]$ permutation, respectively. The sponge state is initialized with the k-bit nonce N and a fixed $(n-k)$-bit IV_{A}. For Ascon-p, k is larger than the rate r_{H}, but since more than 128 bits of the inner part are fixed to IV_{A}, this does not affect the security level.

For the choice Isap-A-128a and Isap-A-128, we choose the Ascon-p permutation with 12 rounds as specified in [DEMS16] for the use of Ascon-Hash and Ascon-Xof [DEMS21]. We consider this as a very conservative choice and think that this might be improved once more analysis of Ascon-Hash and Ascon-Xof is available.

For the choice Isap-K-128, we choose the Keccak-$p[400]$ permutation with 20 rounds as specified in the the Keccak SHA-3 submission (Version 3.0) [BDPV11b], as the winner of the SHA-3 competition Keccak and its variants are very well analyzed. Since existing analysis is far away from threatening full-round versions of Keccak, we use for our fast variant Isap-K-128a the initial proposal of the designers with 16 rounds as described in the Keccak sponge function family main document (Version 1.2) [BDPV09].

4.7. Updates Compared to the Paper

We have performed small tweaks on Isap compared to the original FSE 2017 publication [DEM+17]. In this section, we provide justification for these tweaks.

4.7.1. Absorption of the Nonce N During IsapEnc

Besides side-channel attacks, 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] pose a threat to cryptographic implementations in a hostile environment. To address this threat, 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 2.1d: one can consider IsapRk to serve as re-keying function for a plain sponge-based stream cipher execution with key input $K_{\text{E}}^{*} \parallel N$. 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.

4.7.2. Addition of Ascon-p

We see in Isap a mode of operation that can be instantiated with any suitable permutation. To emphasize this view, we use two different permutations. One is Keccak-$p[400]$, the smallest permutation specified in NIST's FIPS PUB 202 [Nat15] that allows Isap to be instantiated with a 128-bit security level. The other, and new compared to the Isap paper [DEM+17], is the 320-bit permutation of Ascon [DEMS16] that 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 high-end software platforms. Although not standardized, Ascon-p was thoroughly analyzed during the CAESAR competition, and it offers a comfortable security margin.

4.7.3. Single Key

Using a single key for both IsapMac and IsapEnc instead of two independent keys has the advantage that less key material has to be stored compared to the case that different keys are used. It is also sound from a theoretical perspective, given the use of different IV's. On the downside, using the same key exposes this single key to more leakage as discussed in Section 4.2. In a lightweight use case, we think that the savings in storage outweigh this downside and hence, we have decided to use a single key. Additionally, leakage of information about the master key can be compensated for by increasing the size of the master key K.

5. Security Analysis

5.1. Security of the Mode

The mode of Isap combines various ideas and constructions from the unkeyed sponge and the keyed sponge and duplex. The unkeyed sponge was analyzed in [BDPV07; BDPV08], the keyed sponge in [BDPV11d; CDH+12; ADMV15; NY16; JLM14; GPT15; MRV15], and the keyed duplex in [BDPV11c; MRV15; DMV17]. In the remainder of this section, we detail how the security of the Isap mode relies on these results.

The workhorse in Isap is the keyed duplex construction. It operates on a state S, and for a call to the duplex on input block M, it (i) outputs $\lceil S \rceil^r$ and (ii) updates the state to $p(S \oplus M)$. In a very simplified form, the adversary's advantage in the security analysis of the keyed duplex [DMV17] is dominated by

$$\frac{LN}{2^c} + \frac{\mu N}{2^c} + \frac{qN}{2^k},$$

where N is the offline complexity (the number of primitive evaluations of p), L is the number of repeating paths (noting that for every state there is a specific order of queries that lead to it), μ a multicollision term on the outer part of the duplex, and q is the number of state initializations of the duplex. Typically, μ is much smaller than the online complexity M [DMV17], but L may be between 0 and M depending on whether or not the specific use case of the duplex allows for repeating paths. Note that if L is large, one must also take a large capacity c to cope with the loss. If L is small, one can take a smaller capacity c and hence a larger rate r.

The duplex appears in Isap in two different shapes, namely in IsapRk and in IsapEnc. IsapRk is a keyed duplex that might have repeated paths, hence L may be large, but also has a high capacity c_{B}, so the damage of the fact that paths may repeat is limited. Under the assumption that IsapRk is a good duplex, IsapEnc never starts with a repeating state, hence has no repeated paths, thus $L = 0$. This allows us to take a duplex with a lower capacity c_{H} for IsapEnc.

Odd one out is IsapMac: it also relies on IsapRk but is composed of this function and a plain sponge hash function [BDPV07] with rate $\max\{r_{\mathrm{H}}, k\}$ bits. We can in turn rely on the fact that the sponge hash function is indifferentiable from a random oracle [BDPV08]. There is a catch, namely that for Isap-A-128 and Isap-A-128a we have $k > r_{\mathrm{H}}$. In this case, we can rely on the fact that, after the state of a keyless sponge is transformed using the key, one ends up with a keyed sponge for which larger absorption and extraction bits is possible provided that only one permutation call is made.

These generic provable security results, so far, concern black-box security, where the underlying permutations are assumed to be perfectly random and leak-free. Dobraunig and Mennink [DM19a] considered the leakage resilience of the duplex construction, and showed that the full-state keyed duplex is still secure if a limited amount of leakage per duplex call takes place, provided that adversarial state manipulation is restricted wisely. Their result in particular covers the model considered in this work, namely where every evaluation of p may, non-adaptively, leak a limited amount of information. As one of the applications, Dobraunig and Mennink demonstrated that the IsapEnc mode (including the call to IsapRk) is leakage resilient. In a separate work, Dobraunig and Mennink [DM19c] considered leakage resilience of the suffix sponge, a generalization of IsapMac with the function IsapRk abstracted, in a comparable leakage model as before (tightness of their analysis was discussed by Dobraunig and Mennink [DM20]). Dobraunig and Mennink [DM19b] united these two works and showed that they, jointly, imply leakage resilience of the Isap mode. For more information on the leakage resilience of Isap, we refer to the ToSC paper about Isap v2.0 [DEM+20].

Guo et al. [GPPS20] independently considered leakage resilience of the Isap mode. Their approach to treating confidentiality (i.e., the analysis of IsapEnc) is almost identical to the approach suggested by Dobraunig and Mennink [DM19a]. The authenticity of Isap (i.e., the analysis of IsapMac) is only very briefly sketched [GPPS20, Appendix H], and misses the rigor and detail of the suffix keyed sponge analysis of Dobraunig and Mennink [DM19c]. Note that GPPS indicate that their proof is a sketch and thus, their bounds are described in big $\mathcal{O}$ notation without supporting computation. On the other hand, Guo et al. consider a more generous leakage assumption, namely that leakages are hard-to-invert. Overall, one can say that the approaches of Dobraunig and Mennink [DM19b] and Guo et al. [GPPS20] are different and complementary.

5.1.1. List of Published Analysis

- Christoph Dobraunig and Bart Mennink. “Leakage Resilient Value Comparison With Application to Message Authentication”. In: EUROCRYPT 2021. LNCS. to appear. Springer, 2021. URL: https://eprint.iacr.org/2021/402.

- Christoph Dobraunig and Bart Mennink. “Tightness of the Suffix Keyed Sponge Bound”. In: IACR Transactions of Symmetric Cryptology 2020.4 (2020), pp. 195–212. DOI: 10.46586/tosc.v2020.i4.195-212.

- Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. “Isap v2.0”. In: IACR Transactions of Symmetric Cryptology 2020.S1 (2020), pp. 390–416. DOI: 10.13154/tosc.v2020.iS1.390-416.

Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. "Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography – A Practical Guide Through the Leakage-Resistance Jungle". In: CRYPTO 2020. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12170. LNCS. Springer, 2020, pp. 369–400. DOI: 10.1007/978-3-030-56784-2_13.

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

Christoph Dobraunig and Bart Mennink. "Security of the Suffix Keyed Sponge". In: IACR Transactions of Symmetric Cryptology 2019.4 (2019), pp. 223–248. DOI: 10.13154/tosc.v2019.i4.223-248.

Jean Paul Degabriele, Christian Janson, and Patrick Struck. "Sponges Resist Leakage: The Case of Authenticated Encryption". In: ASIACRYPT 2019. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11922. LNCS. Springer, 2019, pp. 209–240. DOI: 10.1007/978-3-030-34621-8_8.

Christoph Dobraunig and Bart Mennink. "Leakage Resilience of the Duplex Construction". In: ASIACRYPT 2019. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11923. LNCS. Springer, 2019, pp. 225–255. DOI: 10.1007/978-3-030-34618-8_8.

Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. "Towards Low-Energy Leakage-Resistant Authenticated Encryption from the Duplex Sponge Construction". In: IACR Transactions of Symmetric Cryptology 2020.1 (2020), pp. 6–42. DOI: 10.13154/tosc.v2020.i1.6-42.

Joan Daemen, Bart Mennink, and Gilles Van Assche. "Full-State Keyed Duplex with Built-In Multi-user Support". In: ASIACRYPT 2017. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10625. LNCS. Springer, 2017, pp. 606–637. DOI: 10.1007/978-3-319-70697-9_21.

Yusuke Naito and Kan Yasuda. "New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length". In: FSE 2016. Ed. by Thomas Peyrin. Vol. 9783. LNCS. Springer, 2016, pp. 3–22. DOI: 10.1007/978-3-662-52993-5_1.

Bart Mennink, Reza Reyhanitabar, and Damian Vizár. "Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption". In: ASIACRYPT 2015. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Springer, 2015, pp. 465–489. DOI: 10.1007/978-3-662-48800-3_19.

Peter Gaži, Krzysztof Pietrzak, and Stefano Tessaro. “The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC”. In: CRYPTO 2015. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9215. LNCS. Springer, 2015, pp. 368–387. DOI: 10.1007/978-3-662-47989-6_18.

Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. “Security of Keyed Sponge Constructions Using a Modular Proof Approach”. In: FSE 2015. Ed. by Gregor Leander. Vol. 9054. LNCS. Springer, 2015, pp. 364–384. DOI: 10.1007/978-3-662-48116-5_18.

Yu Sasaki and Kan Yasuda. “How to Incorporate Associated Data in Sponge-Based Authenticated Encryption”. In: CT-RSA 2015. Ed. by Kaisa Nyberg. Vol. 9048. LNCS. Springer, 2015, pp. 353–370. DOI: 10.1007/978-3-319-16715-2_19.

Philipp Jovanovic, Atul Luykx, and Bart Mennink. “Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes”. In: ASIACRYPT 2014. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, 2014, pp. 85–104. DOI: 10.1007/978-3-662-45611-8_5.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. “Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications”. In: SAC 2011. Ed. by Ali Miri and Serge Vaudenay. Vol. 7118. LNCS. Springer, 2011, pp. 320–337. DOI: 10.1007/978-3-642-28496-0_19.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. “On the Indifferentiability of the Sponge Construction”. In: EUROCRYPT 2008. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, 2008, pp. 181–197. DOI: 10.1007/978-3-540-78967-3_11.

5.2. Security of the Keccak-$p[400]$ Instance

Due to the prominence of Keccak [BDPV11b] as winner of the SHA-3 competition [Nat12], and Keyak [BDP+16b] and Ketje [BDP+16a] as submissions to CAESAR [CAE14], a plethora of cryptanalytic results for keyed and unkeyed sponge and duplex constructions using round reduced versions of the Keccak-f permutations, as well as on the permutations exist. While arguably the majority of the analyses focuses on the 1600-bit variant of the Keccak-f permutation, the similarity in structure of the permutation usually allows to apply variations of the same techniques on smaller permutation variants. A good overview on existing analysis results on Keccak can be found in [JN15]. In this section, we recapitulate the from our point of view most relevant attacks on Keccak and discuss the applicability to our schemes.

5.2.1. Permutation

Zero-sum distinguishers [AM09; BC10] are the permutation distinguishers covering the highest number of rounds. They exploit the low algebraic degree of the Keccak-f permutations creating sets of inputs and outputs, which sum to zero. Guo et al. [GLS16] present zero-sum distinguishers for 12 rounds of Keccak-f[1600] with a complexity of 2^{65} using a 3-round linear structure in the middle of the permutation, while achieving 2^{82} using a 2-round linear structure. They also claim for the 12-round 400-bit permutation Keccak-p[400,12] zero-sum distinguishers with a complexity 2^{82} using a 2-round linear structure, while 3-round structures seem to be inapplicable. However, to mount an attack using zero-sum distinguishers on sponges, an attacker would have to be able to choose inputs in the middle of the permutation. Thus, no attacks on Keyak and Ketje with the 12-round Keccak-p permutations are known that exploit zero-sum distinguishers. Therefore, we conclude that the same is true for Isap-K-128, which also uses 12 rounds for IsapEnc and IsapRk.

5.2.2. IsapRk and IsapEnc

IsapRk and IsapEnc are sponge-based constructions where the secret key is injected during the beginning of the absorption phase, similar to a Keccak prefix-MAC, Keyak, or Ketje. The attacks covering the highest number of rounds for keyed sponges exploit the low algebraic degree of the Keccak-f permutations. This includes the cube-like attacks by Dinur et al. [DMP+15], who present amongst others a keystream prediction for a Keccak-based stream cipher which uses 9 rounds of the 1600-bit permutation to achieve 512-bit security with time complexity 2^{256}. Huang et al. [HWX+17] present conditional cube attacks, including a key-recovery attack on 8 rounds of Keyak with a time complexity of 2^{74}.

In the case of Isap-K-128, two factors preclude those attacks. First of all, the permutation has 12 rounds, whereas the attacks are only capable of covering at most 9 rounds. Second, the nonce N or the hash value Y are absorbed bitwise separated by 12 rounds of the permutation, which significantly reduces the ability of an attacker to exploit cubes in the first place. For Isap-K-128a, the number of rounds between the bitwise injections of the nonce N or the hash value Y is reduced to one. Still, this means having at least 128 rounds from the point where the key is introduced up to the point when a part of the state is leaked. Hence, we expect that conditional cube and cube-like attacks do not work on Isap-K-128a.

Another important attack vector are linear and differential attacks. These are especially relevant in the case of Isap-K-128a, where only the 1-round permutation is used for absorption and the 8-round permutation is used for squeezing the sponge. While having, e.g., colliding differential trails during absorption would also imply problems for Ketje, the situation changes for the squeezing phase. Due to the increased rate used in

Isap-K-128a compared to Ketje, an attacker has more freedom. For this reason, we have increased the number of rounds to 8 for p_E.

5.2.3. IsapMac

Since IsapMac is a suffix-MAC, attacks when unkeyed sponges are used as hash functions are also of concern. For instance, collision attacks on the hashing part of IsapMac have the potential to allow for forgeries. For Keccak, collision attacks for up to 5 rounds were proposed by Dinur et al. [DDS13]. Recently, the 5-round challenges for 1600-bit and 800-bit permutations of the Keccak crunchy crypto collision contest [BDPV14] have been solved, while the 5-round challenge for the 400-bit permutation is still open. Regarding pre-image attacks, attacks for up to 4 rounds for variants of Keccak exist [MPS13; GLS16]. Taking these results together with the result for keyed sponges of Section 5.2.2, we conclude that having 20 rounds in the case of Isap-K-128 and even 16 rounds in the case of Isap-K-128a provide a sufficient security margin for IsapMac.

5.2.4. List of Published Analysis

As the winner of the NIST SHA-3 competition [Kay07], Keccak has received a lot of attention and several results regarding its security have been published. The following list contains both results evaluating the permutation and evaluation of the security of the hash function and the authenticated encryption schemes Keyak [BDP+16b] and Ketje [BDP+16a], both CAESAR round 3 candidates based on the Keccak permutation.

- Rachelle Heim Boissier, Camille Noûs, and Yann Rotella. "Algebraic Collision Attacks on Keccak". In: IACR Transactions of Symmetric Cryptology 2021.1 (2021), pp. 239–268. DOI: 10.46586/tosc.v2021.i1.239-268.

- Le He, Xiaoen Lin, and Hongbo Yu. "Improved Preimage Attacks on 4-Round Keccak-224/256". In: IACR Transactions of Symmetric Cryptology 2021.1 (2021), pp. 217–238. DOI: 10.46586/tosc.v2021.i1.217-238.

- Fukang Liu, Takanori Isobe, Willi Meier, and Zhonghao Yang. "Algebraic Attacks on Round-Reduced Keccak/Xoodoo". Cryptology ePrint Archive, Report 2020/346. 2020. URL: https://eprint.iacr.org/2020/346.

- Sahiba Suryawanshi, Dhiman Saha, and Satyam Sachan. "New Results on the SymSum Distinguisher on Round-Reduced SHA3". In: AFRICACRYPT. Ed. by Abderrahmane Nitaj and Amr M. Youssef. Vol. 12174. LNCS. Springer, 2020, pp. 132–151. DOI: 10.1007/978-3-030-51938-4_7.

- Guozhen Liu, Weidong Qiu, and Yi Tu. "New Techniques for Searching Differential Trails in Keccak". In: IACR Transactions of Symmetric Cryptology 2019.4 (2019), pp. 407–437. DOI: 10.13154/tosc.v2019.i4.407-437.
- Mahesh Sreekumar Rajasree. "Cryptanalysis of Round-Reduced KECCAK Using Non-linear Structures". In: INDOCRYPT. Ed. by Feng Hao, Sushmita Ruj, and Sourav Sen Gupta. Vol. 11898. LNCS. Springer, 2019, pp. 175–192. DOI: 10.1007/978-3-030-35423-7_9.
- Zheng Li, Xiaoyang Dong, Wenquan Bi, Keting Jia, Xiaoyun Wang, and Willi Meier. "New Conditional Cube Attack on Keccak Keyed Modes". In: IACR Transactions of Symmetric Cryptology 2019.2 (2019), pp. 94–124.
- Ting Li and Yao Sun. "Preimage Attacks on Round-Reduced Keccak-224/256 via an Allocating Approach". In: EUROCRYPT (3). Vol. 11478. LNCS. Springer, 2019, pp. 556–584. DOI: 10.1007/978-3-030-17659-4_19.
- Wenquan Bi, Xiaoyang Dong, Zheng Li, Rui Zong, and Xiaoyun Wang. "MILP-aided cube-attack-like cryptanalysis on Keccak Keyed modes". In: Des. Codes Cryptography 87.6 (2019), pp. 1271–1296. DOI: 10.1007/s10623-018-0526-x.
- Fukang Liu, Zhenfu Cao, and Gaoli Wang. "Finding Ordinary Cube Variables for Keccak-MAC with Greedy Algorithm". In: IWSEC. Vol. 11689. LNCS. Springer, 2019, pp. 287–305. DOI: 10.1007/978-3-030-26834-3_17.
- Haibo Zhou, Zheng Li, Xiaoyang Dong, Keting Jia, and Willi Meier. "Practical Key-recovery Attacks on Round-Reduced Ketje Jr, Xoodoo-AE and Xoodyak". Cryptology ePrint Archive, Report 2019/447. 2019. URL: https://eprint.iacr.org/2019/447.
- Ling Song and Jian Guo. "Cube-Attack-Like Cryptanalysis of Round-Reduced Keccak Using MILP". In: IACR Transactions of Symmetric Cryptology 2018.3 (2018), pp. 182–214. DOI: 10.13154/tosc.v2018.i3.182-214.
- Ling Song, Jian Guo, Danping Shi, and San Ling. "New MILP Modeling: Improved Conditional Cube Attacks on Keccak-Based Constructions". In: ASIACRYPT 2018. Ed. by Thomas Peyrin and Steven D. Galbraith. Vol. 11273. LNCS. Springer, 2018, pp. 65–95. DOI: 10.1007/978-3-030-03329-3_3.
- Ting Li, Yao Sun, Maodong Liao, and Dingkang Wang. "Preimage Attacks on the Round-reduced Keccak with Cross-linear Structures". In: IACR Transactions of Symmetric Cryptology 2017.4 (2017), pp. 39–57. DOI: 10.13154/tosc.v2017.i4.39-57.

- Silvia Mella, Joan Daemen, and Gilles Van Assche. "New techniques for trail bounds and application to differential trails in Keccak". In: IACR Transactions of Symmetric Cryptology 2017.1 (2017), pp. 329–357. DOI: 10.13154/tosc.v2017.i1.329-357.

- Zheng Li, Wenquan Bi, Xiaoyang Dong, and Xiaoyun Wang. "Improved Conditional Cube Attacks on Keccak Keyed Modes with MILP Method". In: ASIACRYPT 2017. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. LNCS. Springer, 2017, pp. 99–127. DOI: 10.1007/978-3-319-70694-8_4.

- Maolin Li and Lu Cheng. "Distinguishing Property for Full Round KECCAK-f Permutation". In: CISIS 2017. Ed. by Leonard Barolli and Olivier Terzo. Vol. 611. Advances in Intelligent Systems and Computing. Springer, 2017, pp. 639–646. DOI: 10.1007/978-3-319-61566-0_59.

- Ling Song, Guohong Liao, and Jian Guo. "Non-full Sbox Linearization: Applications to Collision Attacks on Round-Reduced Keccak". In: CRYPTO 2017. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, 2017, pp. 428–451. DOI: 10.1007/978-3-319-63715-0_15.

- Senyang Huang, Xiaoyun Wang, Guangwu Xu, Meiqin Wang, and Jingyuan Zhao. "Conditional Cube Attack on Reduced-Round Keccak Sponge Function". In: EUROCRYPT 2017. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. LNCS. 2017, pp. 259–288. DOI: 10.1007/978-3-319-56614-6_9.

- Kexin Qiao, Ling Song, Meicheng Liu, and Jian Guo. "New Collision Attacks on Round-Reduced Keccak". In: EUROCRYPT 2017. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10212. LNCS. 2017, pp. 216–243. DOI: 10.1007/978-3-319-56617-7_8.

- Jian Guo, Meicheng Liu, and Ling Song. "Linear Structures: Applications to Cryptanalysis of Round-Reduced Keccak". In: ASIACRYPT 2016. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. 2016, pp. 249–274. DOI: 10.1007/978-3-662-53887-6_9.

- Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. "Cube Attacks and Cube-Attack-Like Cryptanalysis on the Round-Reduced Keccak Sponge Function". In: EUROCRYPT 2015. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 733–761. DOI: 10.1007/978-3-662-46800-5_28.

- Jérémy Jean and Ivica Nikolic. "Internal Differential Boomerangs: Practical Analysis of the Round-Reduced Keccak-f Permutation". In: FSE 2015. Ed. by Gregor Leander. Vol. 9054. LNCS. Springer, 2015, pp. 537–556. DOI: 10.1007/978-3-662-48116-5_26.

- Sourav Das and Willi Meier. "Differential Biases in Reduced-Round Keccak". In: AFRICACRYPT 2014. Ed. by David Pointcheval and Damien Vergnaud. Vol. 8469. LNCS. Springer, 2014, pp. 69–87. DOI: 10.1007/978-3-319-06734-6_5.

- Sukhendu Kuila, Dhiman Saha, Madhumangal Pal, and Dipanwita Roy Chowdhury. "Practical Distinguishers against 6-Round Keccak-f Exploiting Self-Symmetry". In: AFRICACRYPT 2014. Ed. by David Pointcheval and Damien Vergnaud. Vol. 8469. LNCS. Springer, 2014, pp. 88–108. DOI: 10.1007/978-3-319-06734-6_6.

- Pawel Morawiecki and Marian Srebrny. "A SAT-based preimage analysis of reduced Keccak hash functions". In: Information Processing Letters 113.10-11 (2013), pp. 392–397. DOI: 10.1016/j.ipl.2013.03.004.

- Pawel Morawiecki, Josef Pieprzyk, and Marian Srebrny. "Rotational Cryptanalysis of Round-Reduced Keccak". In: FSE 2013. Ed. by Shiho Moriai. Vol. 8424. LNCS. Springer, 2013, pp. 241–262. DOI: 10.1007/978-3-662-43933-3_13.

- Stefan Kölbl, Florian Mendel, Tomislav Nad, and Martin Schläffer. "Differential Cryptanalysis of Keccak Variants". In: IMA 2013. Ed. by Martijn Stam. Vol. 8308. LNCS. Springer, 2013, pp. 141–157. DOI: 10.1007/978-3-642-45239-0_9.

- Joan Daemen and Gilles Van Assche. "Differential Propagation Analysis of Keccak". In: FSE 2012. Ed. by Anne Canteaut. Vol. 7549. LNCS. Springer, 2012, pp. 422–441. DOI: 10.1007/978-3-642-34047-5_24.

- Itai Dinur, Orr Dunkelman, and Adi Shamir. "New Attacks on Keccak-224 and Keccak-256". In: FSE 2012. Ed. by Anne Canteaut. Vol. 7549. LNCS. Springer, 2012, pp. 442–461. DOI: 10.1007/978-3-642-34047-5_25.

- Alexandre Duc, Jian Guo, Thomas Peyrin, and Lei Wei. "Unaligned Rebound Attack: Application to Keccak". In: FSE 2012. Ed. by Anne Canteaut. Vol. 7549. LNCS. Springer, 2012, pp. 402–421. DOI: 10.1007/978-3-642-34047-5_23.

- Christina Boura, Anne Canteaut, and Christophe De Cannière. "Higher-Order Differential Properties of Keccak and Luffa". In: FSE 2011. Ed. by Antoine Joux. Vol. 6733. LNCS. Springer, 2011, pp. 252–269. DOI: 10.1007/978-3-642-21702-9_15.

- María Naya-Plasencia, Andrea Röck, and Willi Meier. "Practical Analysis of Reduced-Round Keccak". In: INDOCRYPT 2011. Ed. by Daniel J. Bernstein and Sanjit Chatterjee. Vol. 7107. LNCS. Springer, 2011, pp. 236–254. DOI: 10.1007/978-3-642-25578-6_18.

- Christina Boura and Anne Canteaut. "A zero-sum property for the Keccak-f permutation with 18 rounds". In: ISIT 2010. IEEE, 2010, pp. 2488–2492. DOI: 10.1109/ISIT.2010.5513442.

- Christina Boura and Anne Canteaut. "Zero-Sum Distinguishers for Iterated Permutations and Application to Keccak-f and Hamsi-256". In: SAC 2010. Ed. by Alex Biryukov, Guang Gong, and Douglas R. Stinson. Vol. 6544. LNCS. Springer, 2010, pp. 1–17. DOI: 10.1007/978-3-642-19574-7_1.

- Wenquan Bi, Zheng Li, Xiaoyang Dong, Lu Li, and Xiaoyun Wang. "Conditional cube attack on round-reduced River Keyak". In: Designs, Codes and Cryptography 86.6 (2018), pp. 1295–1310. DOI: 10.1007/s10623-017-0396-7.

- Xiaoyang Dong, Zheng Li, Xiaoyun Wang, and Ling Qin. "Cube-like Attack on Round-Reduced Initialization of Ketje Sr". In: IACR Transactions of Symmetric Cryptology 2017.1 (2017), pp. 259–280. DOI: 10.13154/tosc.v2017.i1.259-280.

- Thomas Fuhr, María Naya-Plasencia, and Yann Rotella. "State-Recovery Attacks on Modified Ketje Jr". In: IACR Transactions of Symmetric Cryptology 2018.1 (2018), pp. 29–56. DOI: 10.13154/tosc.v2018.i1.29-56.

- Itai Dinur, Orr Dunkelman, and Adi Shamir. "Collision Attacks on Up to 5 Rounds of SHA-3 Using Generalized Internal Differentials". In: FSE 2013. Ed. by Shiho Moriai. Vol. 8424. LNCS. Springer, 2013, pp. 219–240. DOI: 10.1007/978-3-662-43933-3_12.

5.3. Security of the Ascon-p Instance

Ascon [DEMS16] has been selected as the primary recommendation for lightweight use-cases in the final portfolio of the CAESAR [CAE14] competition. As CAESAR [CAE14] was a competition lasting for nearly 5 years, Ascon has received a large amount of analysis and we discuss the most important results for the decision of our parameters.

5.3.1. Permutation

Similar as for Keccak, the best known distinguishing property for Ascon's permutation are zero-sum distinguishers. In the case of Ascon, such distinguishers exist for 12 rounds with a complexity of 2^{130} [DEMS15]. As remarked before, to mount an attack using zero-sum distinguishers on sponges, an attacker would have to be able to choose inputs in the middle of the permutation. Hence, no attacks exploiting this property on Ascon are known. For a detailed discussion of the cryptographic properties of the Ascon permutation we refer to [DEMS21].

5.3.2. IsapRk and IsapEnc

The authenticated encryption scheme Ascon-128 has a rate of 64 bits and uses 12 round of Ascon's permutation during the initialization and finalization and 6 rounds during the processing of the data. The best known attacks on Ascon-128 cover 7 out of 12 rounds of the initialization [LDW17; RHSS21] and 4 out of 12 rounds during the finalization [DEMS15]. Hence, we use 12 rounds of the permutation in IsapEnc and IsapRk for Isap-A-128.

For the fast variant Isap-A-128a, we reduce the number of rounds for the permutation p_E used to generate the keystream to 6, which matches the number of rounds of the data processing of Ascon-128. Furthermore, we reduce the number of rounds of p_B used in IsapRk to 1. Using the same tools as [DEMS15] we searched for differential trails for this construction and could show that no collision producing trails for less than 5 iterations exist and any that trail spanning more rounds has at least 64 active S-boxes (probability $\ll 2^{-128}$).

5.3.3. IsapMac

Recently, a hash function based on Ascon's permutation has been introduced [DEMS21]. Since this construction is rather new, we have decided to base IsapMac for both Isap-A-128 and Isap-A-128a on Ascon-Hash. Thus p_H has 12 rounds in both cases. As Ascon-Hash has a 64-bit rate, also IsapMac has a 64-bit rate for absorbing the associated data and the ciphertext. The state is initialized to the 128-bit nonce N and a 192-bit fixed IV, similar to the keyed initial state of the re-keying function.

5.3.4. List of Published Analysis

As the primary selection for lightweight applications in the CAESAR competition [CAE14], Ascon has received a lot of attention and several results regarding its security have been published. The following list contains both results evaluating the permutation and evaluation of the security of the authenticated encryption and hashing, either using variants of Ascon's permutation, or idealized versions of it.

- Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. "Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round Ascon". In: IACR Transactions of Symmetric Cryptology 2021.1 (2021), pp. 130–155. doi: 10.46586/tosc.v2021.i1.130-155.

- Cihangir Tezcan. "Analysis of Ascon, DryGASCON, and Shamash Permutations". In: International Journal of Information Security Science 9.3 (2020), pp. 172–187. url: https://eprint.iacr.org/2020/1458.

Rui Zong, Xiaoyang Dong, and Xiaoyun Wang. "Collision Attacks on Round-Reduced Gimli-Hash/Ascon-Xof/Ascon-Hash". Cryptology ePrint Archive, Report 2019/1115. 2019. URL: https://eprint.iacr.org/2019/1115.

Hailun Yan, Xuejia Lai, Lei Wang, Yu Yu, and Yiran Xing. "New zero-sum distinguishers on full 24-round Keccak-f using the division property". In: IET Information Security 13.5 (2019), pp. 469–478. DOI: 10.1049/iet-ifs.2018.5263.

Achiya Bar-On, Orr Dunkelman, Nathan Keller, and Ariel Weizman. "DLCT: A New Tool for Differential-Linear Cryptanalysis". In: EUROCRYPT 2019. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. LNCS. Springer, 2019, pp. 313–342. DOI: 10.1007/978-3-030-17653-2_11.

Gregor Leander, Cihangir Tezcan, and Friedrich Wiemer. "Searching for Subspace Trails and Truncated Differentials". In: IACR Transactions on Symmetric Cryptology 2018.1 (2018), pp. 74–100. DOI: 10.13154/tosc.v2018.i1.74-100.

Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. "Conditional Cube Attack on Round-Reduced ASCON". In: IACR Transactions on Symmetric Cryptology 2017.1 (2017), pp. 175–202. DOI: 10.13154/tosc.v2017.i1.175-202.

Ashutosh Dhar Dwivedi, Miloš Klouček, Pawel Morawiecki, Ivica Nikolič, Josef Pieprzyk, and Sebastian Wójtowicz. "SAT-based Cryptanalysis of Authenticated Ciphers from the CAESAR Competition". In: SECRYPT ICETE 2017. Ed. by Pierangela Samarati, Mohammad S. Obaidat, and Enrique Cabello. SciTePress, 2017, pp. 237–246. DOI: 10.5220/0006387302370246.

Cihangir Tezcan. "Truncated, Impossible, and Improbable Differential Analysis of Ascon". In: ICISSP 2016. Ed. by Olivier Camp, Steven Furnell, and Paolo Mori. SciTePress, 2016, pp. 325–332. DOI: 10.5220/0005689903250332.

Faruk Göloğlu, Vincent Rijmen, and Qingju Wang. "On the division property of S-boxes". Cryptology ePrint Archive, Report 2016/188. 2016. URL: https://eprint.iacr.org/2016/188.

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates". In: ASIACRYPT 2015. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Springer, 2015, pp. 490–509. DOI: 10.1007/978-3-662-48800-3_20.

Yosuke Todo. "Structural Evaluation by Generalized Integral Property". In: EUROCRYPT 2015. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 287–314. DOI: 10.1007/978-3-662-46800-5_12.

Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Cryptanalysis of Ascon". In: CT-RSA 2015. Ed. by Kaisa Nyberg. Vol. 9048. LNCS. Springer, 2015, pp. 371–387. DOI: 10.1007/978-3-319-16715-2_20.

6. 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 physical 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 secure software and firmware updates.

This chapter covers the implementation of ISAP. We first discuss the robustness of ISAP against implementation attacks in Section 6.1, including a summary of the general rationale introduced in Chapter 4 as well as specific aspects to consider for securely implementing ISAP. Then, we provide an overview of ISAP's performance in software in Section 6.2. With respect to hardware, we discuss dedicated ASIC implementations in Section 6.3, dedicated FPGA implementations in Section 6.4, and a flexible hardware accelerator design for both ASCON/ISAP in Section 6.5.

6.1. Implementation Security

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 (see Section 4.1).

2. ISAPENC's and ISAPMAC's robustness against SPA follows directly from the underlying sponge construction under a generous bounded-leakage assumption (see Section 4.2).

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 (see Section 4.3).

4. IsapRk's robustness against SPA follows from the same model and assumptions as IsapEnc's except for the initialization step, which can be protected by either storing the expanded key or increasing the key size.

6.1.1. Efficiency of Secure Implementations

Protecting cryptographic implementations against implementation attacks has been a hot research topic during the last two decades. Today, there exist two directions in counteracting the attacks.

The first approach works by hardening the implementation of cryptographic algorithms with techniques like hiding or masking. The drawback of this approach is that the overhead for securing a cryptographic primitive against implementation attacks can be very high and is not generic as it depends on the cryptographic primitive itself. In particular, these countermeasures typically imply a significant increase in area requirements. On top of that, implementing countermeasures for specific primitives can be error-prone and difficult to verify for correctness in practice.

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. Isap mainly follows this second approach but also allows for additional countermeasures on primitive-level at low cost. While the original proposal at FSE 2017 [DEM+17] already provides robustness against DPA attacks by design, the additional modifications in this proposal 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, the second approach as implemented by Isap provides secure implementations with a significantly lower area overhead and a very low runtime overhead for processing bulk data. This is ideal for applications that require strong protection while encrypting or decrypting long messages, such as software or firmware updates on embedded devices or bitstream files for FPGAs. Preventing implementation attacks at the protocol level does come with a certain runtime overhead for short messages, however: The protection overhead is shifted to the initialization phase, which only dominates the runtime for short messages. If we consider the usage of masking for additional SPA protection Isap profits from the fact that, for longer messages, the majority of the runtime is spent within IsapMac which is mostly unkeyed (execpt the final tag generation) and hence does not need to use masked permutations during the absorbtion of A and C. In other words, the higher the desired masking order, the better the runtime of Isap will be compared to other single-pass AE schemes that would likely still need to integrate even higher masking orders to cope with DPA attacks.

We provide a comparison of Isap's performance for short and long messages on different platforms in Section 6.2 (software) and Section 6.3 (hardware).

6.1.2. 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 (see Section 4.2). 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] and follow-up work [MSNF16] are 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 research topic.

6.1.3. 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 SFA and SIFA on K. 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, 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. To mitigate this, Isap's re-keying functions are 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 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.

6.1.4. 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+20], the comparison of the tag should be done in a side-channel secure 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 $\lfloor p_{\text{H}}(T'\|0^*)\rfloor_k \stackrel{?}{=} \lfloor p_{\text{H}}(T\|0^*)\rfloor_k$, and the comparison of the actual tags T, T' is only executed if the first comparison was successful. In [DM21], the security of this method is analyzed and an improved variant is proposed. This variant also uses the publicly known value Y that forms the input to IsapRk in IsapMac, and uses it in the comparison: $\lfloor p_{\text{H}}(Y\|T'\|0^*)\rfloor_k \stackrel{?}{=} \lfloor p_{\text{H}}(Y\|T\|0^*)\rfloor_k$.

6.2. Software Implementations

In the following, we present performance metrics of platform-optimized software implementations of Isap-A-128a and Isap-K-128a for various CPU architectures such as x64, ARMv8-A, and ARMv7-M. These architectures thus cover high performance scenarios like 64-bit desktop CPUs, as well as more constrained devices such as 32-bit ARM Cortex-A application processors and ARM 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). We also present some numbers from eBACS [BL] for comparison. The benchmarked scenarios include authenticated encryption of relatively small messages (64 bytes), typical Ethernet II frame sizes (1536 bytes), and very large messages (long). The resulting performance metrics are listed in Table 6.1. 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.

Table 6.1.: Software performance of Isap instances in cycles/byte (x+0 enc.). We refer to our website for up to date implementations and performance numbers.

Processor	Architecture	Isap-A-128a			Isap-K-128a		
		64 B	1536 B	long	64 B	1536 B	long
AMD Ryzen 7 1700[1]	x64	85.7	24.5	21.9	295.0	64.1	54.3
AMD Ryzen 5 1600[2]	x64	-	-	26.9	-	-	67.9
ARM Cortex-A53[2]	ARMv8A	-	-	41.5	-	-	162.0
STM32F303[1]	ARMv7M	542.0	168.0	152.0	2135.0	415.0	338.0
ARM Cortex-M4F[3]	ARMv7M	614.0	-	-	2163.6	-	-
ATmega328P[3]	AVR	450.0	-	-	543.0	-	-

[1]From the ISAP Code Package [Teaa].

[2]From eBACS [BL].

[3]From the NIST LWC team [Teac].

6.3. ASIC Implementations

In this section, we discuss various performance and area metrics of Isap-A-128a and Isap-K-128a in hardware. The presented numbers are either based on the comprehensive ASIC benchmarking report of round 2 candidates in the NIST Lightweight Cryptography Standardization Process [AZ21], or derived from our hardware reference implementations [Teab].

Area. When compared to AesGcm, Isap enables AEAD with just about half the area at similar throughputs (Table 6.2). Besides that, Isap's mode-level features additionally provide protection/hardening against various implementation attacks like DPA. Adding, e.g., mode-level DPA protection to AesGcm would require the additional cost of 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].

Since Isap-A-128a and Isap-K-128a implement the same mode, their area difference is mostly determined by permutation type and state size. Consequently, the Keccak-$p[400]$-based Isap-K-128a does require slightly more area than the Ascon-p-based Isap-A-128a.

Runtime. The runtime of Isap can be divided into two parts: the time for performing initialization/finalization (latency) and the time for processing data. The latency 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 byte is also independent of message length, but defines the overall runtime for long messages. In Table 6.3 we report the corresponding cycle counts of our hardware reference implementations [Teab] for AEAD (long+0 enc) and

Table 6.2.: Hardware metrics for Isap-A-128a and Isap-K-128a for AEAD (long+0 enc.) at 50 MHz as reported in [AZ21]. We refer to our website for up to date implementations and metrics.

Scheme	Interface	Cell Lib	Synthesizer*	Throughput [cycles/byte]	Area [kGE]
AesGcm	32-bit	STM65nm	SDC + CE	2.1	53.0
AesGcm	32-bit	STM65nm	CG + CI	2.1	27.0
AesGcm	32-bit	TSMC65nm	SDC + CE	2.1	25.8
AesGcm	32-bit	TSMC65nm	CG + CI	2.1	26.2
Isap-A-128a	32-bit	STM65nm	SDC + CE	3.2	17.2
Isap-A-128a	32-bit	STM65nm	CG + CI	3.2	12.9
Isap-A-128a	32-bit	TSMC65nm	SDC + CE	3.2	11.4
Isap-A-128a	32-bit	TSMC65nm	CG + CI	3.2	12.0
Isap-K-128a	16-bit	STM65nm	SDC + CE	2.3	19.6
Isap-K-128a	16-bit	STM65nm	CG + CI	2.3	14.0
Isap-K-128a	16-bit	TSMC65nm	SDC + CE	2.3	13.4
Isap-K-128a	16-bit	TSMC65nm	CG + CI	2.3	13.0

*SPC = Synopsys Design Compiler vP-2019.03, CE = Cadence Encounter v14.13, CG = Cadence Genus v18.10, CI = Cadence Innovus v18.10.

MAC-only (0+long enc) operation. Quite noticeably, MAC-only operation results in a faster initialization/finalization and higher throughput. This is mainly due to the fact that IsapEnc can be entirely skipped if the plain/ciphertext length is zero. In general, the throughput of Isap-K-128a is higher than in Isap-A-128a due to the larger rate.

Comparison. Isap is an efficient authenticated encryption scheme with low hardware footprint that prevents implementation attacks like DPA by design. Isap can be implemented securely using a standard implementation of the Ascon-p or Keccak-$p[400]$ 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.

Table 6.3.: Performance of ISAP-A-128A and ISAP-K-128A hardware reference implementations from [Teab].

Instance	Interface	Scenario	Latency [cycles]	Throughput [cycles/byte]
ISAP-A-128A	32-bit	AEAD (long+0 enc.)	368	2.75
ISAP-A-128A	32-bit	MAC (0+long enc.)	215	1.75
ISAP-K-128A	16-bit	AEAD (long+0 enc.)	378	2.33
ISAP-K-128A	16-bit	MAC (0+long enc.)	233	1.38

6.4. FPGA Implementations

In the following, we compare the performance and area of ISAP to AESGCM on 7-series Xilinx FPGA platforms. As can be seen in Table 6.4, area and performance of unprotected AESGCM implementations [Hel] are roughly on par with ISAP, which does offer protection/hardening against side-channel/fault attacks out of the box. However, even if we only consider the overhead of 1st-order Threshold Implementations (TI) for AESGCM, the area increases significantly while the throughput drops. Note that the *fast* version of AESGCM TI does reach very high throughput numbers, however only if combined with an RNG (cost not included in the area) that can deliver randomness at a rate of up to 175.24 Gbit/s, which is impractical [MMNV18].

Table 6.4.: FPGA metrics of ISAP compared to the NIST standardized AESGCM mode for the scenario AEAD (long+0 enc.) The columns SCA and FI indicate if the designs offer some protection against side-channel/fault-injection attacks.

Design	FPGA	Slices	SCA	FI	Throughput [cycles/byte]
AESGCM [MHN+20]	Artix-7	1 008	✗	✗	0.68
AESGCM [MHN+20]	Artix-7	810	✗	✗	2.06
AESGCM TI [MMNV18]*	Virtex-7	3 433	✓	✗	11.82
ISAP-A-128A [MHN+20]	Artix-7	618	✓	✓	2.75
ISAP-K-128A [MHN+20]	Artix-7	655	✓	✓	2.33

*Reported area does not include the RNG.

6.5. Hardware Accelerators

We discuss a hardware accelerator design for ASCON-p, presented at CARDIS 2020 [SP20], that utilizes tight integration into a processors register file to significantly speed up various ASCON/ISAP computations at a comparably low cost.

The key idea behind this accelerator is to couple the combinatorial logic for one round of ASCON-p to certain words in the processors register file that can then be updated within one clock cycle via a special assembly instruction. This way, the cryptographic mode remains flexible in software while the computationally expensive permutation profits from hardware acceleration. When integrated into the 32-bit RI5CY core [Gro], this accelerator can be realized with about 4.7 kGE, or about half the area of dedicated co-processor designs (see Table 6.5). At the same time, authenticated encryption and hashing see speed-up factors of about 50 to 80, when compared to corresponding pure software implementations (see Table 6.6).

Table 6.5.: Comparison between the RI5CY core with/without 1-round ASCON-p accelerator and dedicated co-processor designs of ASCON and ISAP.

Design	Area [kGE]
RI5CY base design	45.6
ASCON-p accelerator	4.7
ASCON co-processor [GWDE15]	7.1
ASCON co-processor [DEMS]	9.4
ISAP co-processor [Table 6.2]	$\geq$ 11.4

Table 6.6.: Runtime and code size comparison of ASM implementations of ASCON and ISAP with 1-round ASCON-p hardware acceleration (x+0 enc.)

Scheme	Cycles/Byte			Code Size (B)
	64 B	1536 B	long	
ASCON128	4.2	2.2	2.1	888
ASCONHASH	4.6	2.6	2.5	484
ISAP-A-128A	29.1	5.2	4.2	1 844
ISAP-A-128	73.6	7.7	5.0	2 552

A. Specification of Ascon-p

The following description of the Ascon-p permutation is adapted from the Ascon specification [DEMS16; DEMS21].

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:

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

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 A.1 lists the notation and symbols used in the following description.

Table A.1.: Notation used for Ascon's permutation.

p_C, p_S, p_L	constant-addition, substitution and linear layer of $p = p_L \circ p_S \circ p_C$
$x_0, \ldots, x_4$	The five 64-bit words of the state S
$x_{0,i}, \ldots, x_{4,i}$	Bit i, $0 \leq i < 64$, of words $x_0, \ldots, x_4$, 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, $S = x_0 \parallel x_1 \parallel x_2 \parallel x_3 \parallel x_4$.

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 A.2):

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

Table A.2.: 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`

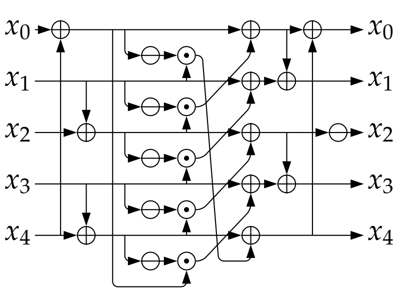
Substitution Layer

The substitution layer p_S updates the state S with 64 parallel applications of the 5-bit S-box $\mathcal{S}(x)$ defined in Figure A.1a to each bit-slice of the five registers $x_0 \ldots x_4$. 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 Figure A.1b 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 A.1.: Ascon's substitution layer and linear diffusion layer.

B. Specification of Keccak-p[400]

Keccak-p[400] is specified in [BDPV11e; Nat15]. 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

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

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 [BDPV11e; Nat15].

Acknowledgments

The authors would like to thank Mario Werner for many helpful discussions and providing his hardware description of Keccak.

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644052 (HECTOR) and agreement No 681402 (SOPHIA). Furthermore, this work has been supported in part by the Austrian Research Promotion Agency (FFG) under grant number 845589 (SCALAS) and grant ESPRESSO as well as by the Austrian Science Fund (FWF): P26494-N15 and J 4277-N38. Bart Mennink is supported by a postdoctoral fellowship from the Netherlands Organisation for Scientific Research (NWO) under Veni grant 016.Veni.173.017.

Bibliography

[ADMV15] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. "Security of Keyed Sponge Constructions Using a Modular Proof Approach". In: FSE 2015. Ed. by Gregor Leander. Vol. 9054. LNCS. Springer, 2015, pp. 364–384. DOI: 10.1007/978-3-662-48116-5_18 (p. 22).

[AM09] Jean-Philippe Aumasson and Willi Meier. "Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi". 2009. URL: https://131002.net/data/papers/AM09.pdf (p. 26).

[AZ21] Mark D. Aagaard and Nusa Zidaric. "ASIC Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process". In: IACR Cryptol. ePrint Arch. 2021 (2021), p. 49. URL: https://eprint.iacr.org/2021/049 (pp. 39, 40).

[BC10] Christina Boura and Anne Canteaut. "A zero-sum property for the Keccak-f permutation with 18 rounds". In: ISIT 2010. IEEE, 2010, pp. 2488–2492. DOI: 10.1109/ISIT.2010.5513442 (p. 26).

[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: CARDIS 2013. Ed. by Aurélien Francillon and Pankaj Rohatgi. Vol. 8419. LNCS. Springer, 2013, pp. 187–199. DOI: 10.1007/978-3-319-08302-5_13 (p. 40).

[BDP+16a] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. "Ketje v2". Submission to Round 3 of the CAESAR competition. 2016. URL: https://competitions.cr.yp.to/round3/ketjev2.pdf (pp. 3, 19, 25, 27).

[BDP+16b] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. "Keyak v2.2". Submission to Round 3 of the CAESAR competition. 2016. URL: https://competitions.cr.yp.to/round3/keyakv22.pdf (pp. 19, 25, 27).

[BDPV07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Sponge functions". Ecrypt Hash Workshop 2007. 2007. URL: http://sponge.noekeon.org/SpongeFunctions.pdf (p. 22).

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "On the Indifferentiability of the Sponge Construction". In: EUROCRYPT 2008. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, 2008, pp. 181–197. DOI: 10.1007/978-3-540-78967-3_11 (p. 22).

[BDPV09] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. "Keccak sponge function family main document (Version 1.2)". 2009. URL: https://keccak.noekeon.org/Keccak-main-1.2.pdf (p. 20).

[BDPV11a] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. "Cryptographic sponge functions (Version 0.1)". 2011. URL: https://sponge.noekeon.org/ (pp. 15, 17).

[BDPV11b] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. "The Keccak SHA-3 submission (Version 3.0)". 2011. URL: https://keccak.noekeon.org/Keccak-submission-3.pdf (pp. 20, 25).

[BDPV11c] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications". In: SAC 2011. Ed. by Ali Miri and Serge Vaudenay. Vol. 7118. LNCS. Springer, 2011, pp. 320–337. DOI: 10.1007/978-3-642-28496-0_19 (p. 22).

[BDPV11d] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "On the security of the keyed sponge construction". In: SKEW 2011. 2011. URL: http://sponge.noekeon.org/SpongeKeyed.pdf (p. 22).

[BDPV11e] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "The Keccak reference, Version 3.0". SHA-3 competition (round 3). 2011. URL: https://keccak.team/files/Keccak-reference-3.0.pdf (pp. 3, 8, 45).

[BDPV12] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Permutation-based Encryption, Authentication and Authenticated Encryption". DIAC Workshop. 2012. URL: https://www.hyperelliptic.org/djb/diac/record.pdf (pp. 15, 18, 19).

[BDPV14] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Keccak Crunchy Crypto Collision and Pre-image Contest". 2014. URL: https://keccak.noekeon.org/crunchy_contest.html (p. 27).

[BGP+20] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. "TEDT, a Leakage-Resilient AEAD mode for High (Physical) Security Applications". In: IACR Transactions on Cryptographic Hardware and Embedded Systems 2020.1 (2020), pp. 6–42. DOI: 10.13154/tches.v2020.i1.256-320 (pp. 11, 38).

[BL] Daniel J. Bernstein and Tanja Lange, eds. "eBACS: ECRYPT Benchmarking of Cryptographic Systems". URL: https://bench.cr.yp.to/results-nistlwc-aead.html (visited on 05/05/2021) (pp. 38, 39).

[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. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. LNCS. Springer, 2017, pp. 693–723. DOI: 10.1007/978-3-319-70694-8_24 (p. 11).

[BPPS17] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. "On Leakage-Resilient Authenticated Encryption with Decryption Leakages". In: IACR Transactions of Symmetric Cryptology 2017.3 (2017), pp. 271–293. DOI: 10.13154/tosc.v2017.i3.271-293 (p. 11).

[BS97] Eli Biham and Adi Shamir. "Differential Fault Analysis of Secret Key Cryptosystems". In: CRYPTO '97. Ed. by Burton S. Kaliski Jr. Vol. 1294. LNCS. Springer, 1997, pp. 513–525. DOI: 10.1007/BFb0052259 (pp. 20, 36).

[CAE14] CAESAR committee. "CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness". 2014. URL: https://competitions.cr.yp.to/ (pp. 18, 21, 25, 31, 32).

[CDH+12] Donghoon Chang, Morris Dworkin, Seokhie Hong, John Kelsey, and Mridul Nandi. "A keyed sponge construction with pseudorandomness in the standard model". The Third SHA-3 Candidate Conference (March 2012). 2012 (p. 22).

[DDS13] Itai Dinur, Orr Dunkelman, and Adi Shamir. "Collision Attacks on Up to 5 Rounds of SHA-3 Using Generalized Internal Differentials". In: FSE 2013. Ed. by Shiho Moriai. Vol. 8424. LNCS. Springer, 2013, pp. 219–240. DOI: 10.1007/978-3-662-43933-3_12 (p. 27).

[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: ASIACRYPT 2018. Ed. by Thomas Peyrin and Steven D. Galbraith. Vol. 11273. LNCS. Springer, 2018, pp. 315–342. DOI: 10.1007/978-3-030-03329-3_11 (pp. 20, 36, 37).

[DEK+16] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Victor Lomné, and Florian Mendel. "Statistical Fault Attacks on Nonce-Based Authenticated Encryption Schemes". In: ASIACRYPT 2016. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. 2016, pp. 369–395. DOI: 10.1007/978-3-662-53887-6_14 (pp. 20, 36, 37).

[DEK+18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. "SIFA: Exploiting Ineffective Fault Inductions on Symmetric Cryptography". In: IACR Transactions on Cryptographic Hardware and Embedded Systems 2018.3 (2018), pp. 547–572. DOI: 10.13154/tches.v2018.i3.547-572 (pp. 20, 36, 37).

[DEM+17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. "ISAP – Towards Side-Channel Secure Authenticated Encryption". In: IACR Transactions on Symmetric Cryptology 2017.1 (2017), pp. 80–105. DOI: 10.13154/tosc.v2017.i1.80-105 (pp. 3, 11, 20, 21, 36).

[DEM+20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. "Isap v2.0". In: IACR Transactions of Symmetric Cryptology 2020.S1 (2020), pp. 390–416. DOI: 10.13154/tosc.v2020.iS1.390-416 (p. 23).

[DEMS] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer, eds. "Ascon Hardware Implementations". URL: https://ascon.iaik.tugraz.at/implementations.html (visited on 04/26/2021) (p. 42).

[DEMS15] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Cryptanalysis of Ascon". In: CT-RSA 2015. Ed. by Kaisa Nyberg. Vol. 9048. LNCS. Springer, 2015, pp. 371–387. DOI: 10.1007/978-3-319-16715-2_20 (pp. 31, 32).

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon v1.2". Submission to CAESAR: Competition for Authenticated Encryption. Security, Applicability, and Robustness. 2016. URL: https://ascon.iaik.tugraz.at/ (pp. 3, 8, 18–21, 31, 43).

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon v1.2". Submission to NIST. 2019 – 2021. URL: https://ascon.iaik.tugraz.at (pp. 8, 9, 16, 20, 31, 32, 43).

[DM19a] Christoph Dobraunig and Bart Mennink. "Leakage Resilience of the Duplex Construction". In: ASIACRYPT 2019. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11923. LNCS. Springer, 2019, pp. 225–255. DOI: 10.1007/978-3-030-34618-8_8 (p. 23).

[DM19b] Christoph Dobraunig and Bart Mennink. "Leakage Resilience of the ISAP Mode: a Vulgarized Summary". NIST Lightweight Cryptography Workshop 2019. 2019 (p. 23).

[DM19c] Christoph Dobraunig and Bart Mennink. "Security of the Suffix Keyed Sponge". In: IACR Transactions of Symmetric Cryptology 2019.4 (2019), pp. 223–248. DOI: 10.13154/tosc.v2019.i4.223-248 (p. 23).

[DM20] Christoph Dobraunig and Bart Mennink. "Tightness of the Suffix Keyed Sponge Bound". In: IACR Transactions of Symmetric Cryptology 2020.4 (2020), pp. 195–212. DOI: 10.46586/tosc.v2020.i4.195-212 (p. 23).

[DM21] Christoph Dobraunig and Bart Mennink. "Leakage Resilient Value Comparison With Application to Message Authentication". In: EUROCRYPT 2021. LNCS. to appear. Springer, 2021. URL: https://eprint.iacr.org/2021/402 (p. 38).

[DMMP18] Christoph Dobraunig, Stefan Mangard, Florian Mendel, and Robert Primas. "Fault Attacks on Nonce-Based Authenticated Encryption: Application to Keyak and Ketje". In: SAC 2018. Ed. by Carlos Cid and Michael J. Jacobson Jr. Vol. 11349. LNCS. Springer, 2018, pp. 257–277. DOI: 10.1007/978-3-030-10970-7_12 (pp. 20, 36, 37).

[DMP+15] Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. "Cube Attacks and Cube-Attack-Like Cryptanalysis on the Round-Reduced Keccak Sponge Function". In: EUROCRYPT 2015. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 733–761. DOI: 10.1007/978-3-662-46800-5_28 (p. 26).

[DMV17] Joan Daemen, Bart Mennink, and Gilles Van Assche. "Full-State Keyed Duplex with Built-In Multi-user Support". In: ASIACRYPT 2017. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10625. LNCS. Springer, 2017, pp. 606–637. DOI: 10.1007/978-3-319-70697-9_21 (pp. 15, 22).

[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: CHES 2016. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. LNCS. Springer, 2016, pp. 194–212. DOI: 10.1007/978-3-662-53140-2_10 (p. 40).

[FJLT13] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. "Fault Attacks on AES with Faulty Ciphertexts Only". In: FDTC 2013. Ed. by Wieland Fischer and Jörn-Marc Schmidt. IEEE Computer Society, 2013, pp. 108–118. DOI: 10.1109/FDTC.2013.18 (pp. 20, 36, 37).

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. "Practical Leakage-Resilient Symmetric Cryptography". In: CHES 2012. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. LNCS. Springer, 2012, pp. 213–232. DOI: 10.1007/978-3-642-33027-8_13 (p. 16).

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: Journal of the ACM 33.4 (1986), pp. 792–807. DOI: 10.1145/6490.6503 (p. 15).

[GLS16] Jian Guo, Meicheng Liu, and Ling Song. "Linear Structures: Applications to Cryptanalysis of Round-Reduced Keccak". In: ASIACRYPT 2016. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. 2016, pp. 249–274. DOI: 10.1007/978-3-662-53887-6_9 (pp. 26, 27).

[GMK17] Hannes Gross, Stefan Mangard, and Thomas Korak. "An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order". In: CT-RSA 2017. Ed. by Helena Handschuh. Vol. 10159. LNCS. Springer, 2017, pp. 95–112. DOI: 10.1007/978-3-319-52153-4_6 (p. 40).

[GPPS20] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. "Towards Low-Energy Leakage-Resistant Authenticated Encryption from the Duplex Sponge Construction". In: IACR Transactions of Symmetric Cryptology 2020.1 (2020), pp. 6–42. DOI: 10.13154/tosc.v2020.i1.6-42 (p. 23).

[GPT15] Peter Gaži, Krzysztof Pietrzak, and Stefano Tessaro. “The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC”. In: CRYPTO 2015. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9215. LNCS. Springer, 2015, pp. 368–387. DOI: 10.1007/978-3-662-47989-6_18 (pp. 18, 22).

[Gro] OpenHW Group, ed. “OpenHW Group CORE-V CV32E40P RISC-V IP”. URL: https://github.com/openhwgroup/cv32e40p (visited on 04/26/2021) (p. 42).

[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. URL: https://eprint.iacr.org/2015/034 (p. 42).

[Hel] Helion Technology Limited. “AES-GCM cores”. (accessed: 09/2020). URL: https://www.heliontech.com/aes_gcm.htm (p. 41).

[HWX+17] Senyang Huang, Xiaoyun Wang, Guangwu Xu, Meiqin Wang, and Jingyuan Zhao. “Conditional Cube Attack on Reduced-Round Keccak Sponge Function”. In: EUROCRYPT 2017. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. LNCS. 2017, pp. 259–288. DOI: 10.1007/978-3-319-56614-6_9 (p. 26).

[JLM14] Philipp Jovanovic, Atul Luykx, and Bart Mennink. “Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes”. In: ASIACRYPT 2014. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, 2014, pp. 85–104. DOI: 10.1007/978-3-662-45611-8_5 (p. 22).

[JN15] Jérémy Jean and Ivica Nikolic. “Internal Differential Boomerangs: Practical Analysis of the Round-Reduced Keccak-f Permutation”. In: FSE 2015. Ed. by Gregor Leander. Vol. 9054. LNCS. Springer, 2015, pp. 537–556. DOI: 10.1007/978-3-662-48116-5_26 (p. 25).

[Kay07] Richard F. Kayser. “Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family”. In: Federal Register Notice 72.212 (2007), pp. 62212–62220. URL: http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf (p. 27).

[LDW17] Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. “Conditional Cube Attack on Round-Reduced ASCON”. In: IACR Transactions on Symmetric Cryptology 2017.1 (2017), pp. 175–202. DOI: 10.13154/tosc.v2017.i1.175-202 (p. 32).

[MHN+20] Kamyar Mohajerani, Richard Haeussler, Rishub Nagpal, Farnoud Farahmand, Abubakr Abdulgadir, Jens-Peter Kaps, and Kris Gaj. “FPGA Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process: Methodology, Metrics, Tools, and Results”. In: IACR Cryptol. ePrint Arch. 2020 (2020), p. 1207. URL: https://eprint.iacr.org/2020/1207 (p. 41).

[MMNV18] Nele Mentens, Vojtech Miskovsky, Martin Novotny, and Jo Vliegen. "High-speed Side-channel-protected Encryption and Authentication in Hardware". Cryptology ePrint Archive, Report 2018/1088. 2018. URL: https://eprint.iacr.org/2018/1088 (p. 41).

[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: CARDIS 2011. Ed. by Emmanuel Prouff. Vol. 7079. LNCS. Springer, 2011, pp. 115–132. DOI: 10.1007/978-3-642-27257-8_8 (pp. 12, 16).

[MPS13] Pawel Morawiecki, Josef Pieprzyk, and Marian Srebrny. "Rotational Cryptanalysis of Round-Reduced Keccak". In: FSE 2013. Ed. by Shiho Moriai. Vol. 8424. LNCS. Springer, 2013, pp. 241–262. DOI: 10.1007/978-3-662-43933-3_13 (p. 27).

[MRV15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. "Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption". In: ASIACRYPT 2015. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Springer, 2015, pp. 465–489. DOI: 10.1007/978-3-662-48800-3_19 (pp. 15, 18, 22).

[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: AFRICACRYPT 2010. Ed. by Daniel J. Bernstein and Tanja Lange. Vol. 6055. LNCS. Springer, 2010, pp. 279–296. DOI: 10.1007/978-3-642-12678-9_17 (pp. 12, 16, 17, 39).

[MSJ12] Marcel Medwed, François-Xavier Standaert, and Antoine Joux. "Towards Super-Exponential Side-Channel Security with Efficient Leakage-Resilient PRFs". In: CHES 2012. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. LNCS. Springer, 2012, pp. 193–212. DOI: 10.1007/978-3-642-33027-8_12 (p. 37).

[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: ASIACRYPT 2016. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. 2016, pp. 602–623. DOI: 10.1007/978-3-662-53887-6_22 (p. 37).

[Nat12] National Institute of Standards and Technology. "SHA-3 Competition". 2007–2012. URL: https://csrc.nist.gov/groups/ST/hash/sha-3/index.html (p. 25).

[Nat15] 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. Aug. 2015. DOI: 10.6028/NIST.FIPS.202 (pp. 3, 8, 9, 18, 21, 45).

[NY16] Yusuke Naito and Kan Yasuda. "New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length". In: FSE 2016. Ed. by Thomas Peyrin. Vol. 9783. LNCS. Springer, 2016, pp. 3–22. DOI: 10.1007/978-3-662-52993-5_1 (p. 22).

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. "Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives". In: ACM SIGSAC 2015. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel. ACM, 2015, pp. 96–108. DOI: 10.1145/2810103.2813626 (p. 13).

[RHSS21] Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. "Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round Ascon". In: IACR Transactions of Symmetric Cryptology 2021.1 (2021), pp. 130–155. DOI: 10.46586/tosc.v2021.i1.130-155 (p. 32).

[SP20] Stefan Steinegger and Robert Primas. "A Fast and Compact RISC-V Accelerator for Ascon and Friends". In: CARDIS. Vol. 12609. Lecture Notes in Computer Science. Springer, 2020, pp. 53–67 (p. 41).

[SPY+10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. "Leakage Resilient Cryptography in Practice". In: Towards Hardware-Intrinsic Security. Ed. by Ahmad-Reza Sadeghi and David Naccache. Information Security and Cryptography. Springer, 2010, pp. 99–134. DOI: 10.1007/978-3-642-14452-3_5 (pp. 16, 39).

[STKT06] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. "Birthday Paradox for Multi-collisions". In: ICISC 2006. Ed. by Min Surp Rhee and Byoungcheon Lee. Vol. 4296. LNCS. Springer, 2006, pp. 29–40. DOI: 10.1007/11927587_5 (p. 17).

[Teaa] ISAP Team. "ISAP Code Package". (accessed: 05/2021). URL: https://github.com/isap-lwc/isap-code-package (p. 39).

[Teab] ISAP Team. "ISAP Hardware Package". (accessed: 05/2021). URL: https://github.com/isap-lwc/isap-hardware-package (pp. 39, 41).

[Teac] NIST LWC Team. "Benchmarking of Lightweight Cryptographic Algorithms on Microcontrollers". (accessed: 05/2021). URL: https://github.com/usnistgov/Lightweight-Cryptography-Benchmarking (p. 39).

[TS14] Mostafa M. I. Taha and Patrick Schaumont. "Side-channel countermeasure for SHA-3 at almost-zero area overhead". In: HOST 2014. IEEE Computer Society, 2014, pp. 93–96. DOI: 10.1109/HST.2014.6855576 (p. 15).