J Cryptol (2021)34:33
https://doi.org/10.1007/s00145-021-09398-9

Journal of
CRYPTOLOGY

ASCON v1.2: Lightweight Authenticated Encryption and Hashing

Christoph Dobraunig
Radboud University, Nijmegen, The Netherlands
Graz University of Technology, Graz, Austria
christoph.dobraunig@iaik.tugraz.at

Maria Eichlseder
Graz University of Technology, Graz, Austria
maria.eichlseder@iaik.tugraz.at

Florian Mendel · Martin Schläffer
Infineon Technologies AG, Neubiberg, Germany
florian.mendel@infineon.com
martin.schlaeffer@infineon.com

Communicated by Tetsu Iwata

Received 2 December 2019 / Revised 28 September 2020 / Accepted 28 September 2020
Online publication 22 June 2021

Abstract. Authenticated encryption satisfies the basic need for authenticity and confidentiality in our information infrastructure. In this paper, we provide the specification of ASCON-128 and ASCON-128a. Both authenticated encryption algorithms provide efficient authenticated encryption on resource-constrained devices and on high-end CPUs. Furthermore, they have been selected as the "primary choice" for lightweight authenticated encryption in the final portfolio of the CAESAR competition. In addition, we specify the hash function ASCON-HASH, and the extendable output function ASCON-XOF. Moreover, we complement the specification by providing a detailed overview of existing cryptanalysis and implementation results.

Keywords. CAESAR competition, Authenticated encryption, Hash function, Extendable output function, Lightweight cryptography, Permutation-based cryptography, ASCON.

1. Introduction

Authenticated encryption schemes protect authenticity and confidentiality of data, two security properties that are crucially needed in our information infrastructure. While the traditional focus of authenticated encryption was in securing the connection between two communicating devices, more and more devices now operate in environments where an attacker may have physical access to the device. Hence, efficient protection against side-channel attacks is crucial and should preferably be implementable efficiently on resource-constrained devices. At the same time, ciphers should also lend themselves to fast software implementations on higher-end CPUs.

In this paper, we present the cipher suite Ascon, which provides authenticated encryption with associated data (AEAD) and hashing functionality. The suite consists of the authenticated ciphers Ascon-128 and Ascon-128a, which have been selected as primary choice for lightweight authenticated encryption in the final portfolio of the CAESAR competition [87], the hash function Ascon-Hash, and the extendable output function Ascon-Xof. All schemes provide 128-bit security and internally use the same 320-bit permutation (with different round numbers) so that a single lightweight primitive is sufficient to implement both AEAD based on the duplex construction [17] and extendable-output hashing using the sponge construction [14].

The Ascon suite and especially the underlying 320-bit permutation have been designed with these challenges of our modern information infrastructure in mind. Ascon is considered highly secure and robust in practice with a very low area footprint in hardware while providing good performance in software and hardware implementations. To provide these properties, the main components of Ascon are inspired from standardized and well-analyzed primitives. The substitution layer uses an affine equivalent of the S-box used in the χ mapping of Keccak [19,27] designed to add diffusion. The permutation layer uses linear functions similar to the Σ functions used in SHA-2 [71,72]. The resulting permutation is defined on 64-bit words using only bitwise Boolean functions (and, not, xor) and rotations within words. Hence, the permutation lends itself well to fast bitsliced implementations on 64-bit platforms, while bit interleaving [13] allows for fast bitsliced implementations on 32-, 16-, and 8-bit platforms. Thus, Ascon is an excellent choice in scenarios where lightweight devices carry out cryptographic operations. Due to the good performance in software, Ascon is a perfect fit in scenarios where lightweight devices communicate with high-end servers. Benchmarks show that Ascon is particularly efficient for short messages [1,45].

Ciphers have to withstand real-world threats. Therefore, Ascon's permutation and authenticated encryption mode have been designed to provide robustness against certain implementation mistakes and attacks and to facilitate efficient protected implementations. For example, even if an attacker somehow manages to recover an internal state during data processing (e.g., due to side-channel attacks), this does not directly lead to the recovery of the secret key or to constructing forgeries without significant additional computations. Besides increasing the robustness of any implementation, this also allows more efficient protection against side-channel attacks such as differential power analysis (DPA) attacks with leveled implementations [4,7]: In masked implementations, it can be sufficient that the initialization and finalization provides high robustness against side-channel analysis, whereas the bulk data can be processed at higher speed with a lower protection level. Thanks to Ascon's low-degree S-box, masked implementations induce only a relatively small overhead in hardware and software [75], so it is feasible to include protection on constrained devices.

Ascon-128 and Ascon-128a have been selected as the "primary choice" for lightweight authenticated encryption in the final portfolio of the CAESAR competition. Of the initial 57 submissions, six were selected for this portfolio in three use-cases. During this competition, Ascon and its permutation have undergone a thorough public evaluation.

This has resulted in numerous publications giving insight in the security of Ascon, and many more that discuss Ascon more generally. All existing analysis shows a comfortable security margin, with no indication of weaknesses regarding Ascon-128 and Ascon-128a.

History Ascon was first published as a candidate in Round 1 [32] of the CAESAR competition [87]. This original design (version v1) specified the permutation as well as the mode for authenticated encryption with two recommended family members: The primary recommendation Ascon-128 as well as a variant Ascon-96 with 96-bit key. For the subsequent versions v1.1 for Round 2 [33] and v1.2 for Round 3 [35], minor functional tweaks were applied, including a reordering of the round constants and the modification of the secondary recommendation to the current Ascon-128a.

At the time of writing, Ascon is competing in Round 2 [36] of the NIST Lightweight Cryptography (LWC) project [74]. The submission to NIST includes not only the authenticated cipher family, but also introduces modes of operation for hashing: Ascon-Hash (with fixed output size) and Ascon-Xof (with variable output size), as well as a third parameterization for authenticated encryption: Ascon-80pq.

The present paper describes Ascon v1.2 as selected in the CAESAR competition including its additions for NIST LWC competition.

Outline We specify the Ascon cipher suite with its modes for authenticated encryption and hashing as well as the core permutation in Sect. 2. We summarize the security claim for the Ascon family in Sect. 3 and give an overview of Ascon's main features in Sect. 4. In Sect. 5, we provide details on the design rationale for the modes and the permutation. In Sect. 6, we analyze the security of the Ascon cipher suite and provide an overview of third-party cryptanalysis results. We summarize and discuss implementation security and efficient implementations in Sect. 7. Finally, we conclude in Sect. 8.

2. Specification of Ascon

This section provides a complete and self-contained specification of the Ascon cipher suite, starting with an overview of the algorithms in Sect. 2.1, the individual recommended parameter sets in Sect. 2.2, and the notation in Sect. 2.3. Afterward, the authenticated encryption modes are specified in Sect. 2.4, the hashing mode in Sect. 2.5, and the underlying permutation in Sect. 2.6.

2.1. *Algorithms in the* Ascon *Cipher Suite*

The Ascon cipher suite consists of a family of authenticated encryption designs Ascon together with the hash function Ascon-Hash that builds upon the extendable output function Ascon-Xof.

Authenticated encryption For the authenticated encryption designs Ascon, the family members are parameterized by the key length $k \leq 160$ bits, the rate (data block size) r and internal round numbers a and b. Each design specifies an authenticated encryption algorithm $\mathcal{E}_{k,r,a,b}$ and a decryption algorithm $\mathcal{D}_{k,r,a,b}$. The authenticated encryption procedure $\mathcal{E}_{k,r,a,b}$ takes as inputs a secret key K with k bits, a nonce (public message number) N with 128 bits, associated data A of arbitrary length and a plaintext P of

Table 1. Parameters for recommended authenticated encryption schemes.

Name	Algorithms	Bit size of				Rounds	
		Key	Nonce	Tag	Data block	p^a	p^b
Ascon-128	$\mathcal{E}, \mathcal{D}_{128,64,12,6}$	128	128	128	64	12	6
Ascon-128a	$\mathcal{E}, \mathcal{D}_{128,128,12,8}$	128	128	128	128	12	8

arbitrary length. It produces an output consisting of the authenticated ciphertext C of exactly the same length as the plaintext P plus an authentication tag T of size 128 bits, which authenticates both the associated data and the encrypted message:

$$\mathcal{E}_{k,r,a,b}(K, N, A, P) = (C, T).$$

The decryption and verification procedure $\mathcal{D}_{k,r,a,b}$ takes as input the key K, nonce N, associated data A, ciphertext C and tag T, and outputs either the plaintext P if the verification of the tag is correct or an error $\perp$ if the verification of the tag fails:

$$\mathcal{D}_{k,r,a,b}(K, N, A, C, T) \in \{P, \perp\}.$$

Hashing The extendable output function is parameterized by the rate (data block size) r, a round number a, and an output length limit h ($h = 0$ for unlimited output). The extendable output function $\mathcal{X}_{h,r,a}$ maps the input message M of arbitrary length to a hash output H of arbitrary specified length $\ell \leq h$:

$$\mathcal{X}_{h,r,a}(M, \ell) = H.$$

Both Ascon-Hash and Ascon-Xof use this algorithm: Ascon-Hash with $h = \ell = 256$, Ascon-Xof with $h = 0$ for unlimited output. The parameter ℓ solely influences the bit length of H. So, calls to $\mathcal{X}_{h,r,a}(M, \ell') = H'$ and $\mathcal{X}_{h,r,a}(M, \ell'') = H''$, having the same parameters and inputs except $\ell' < \ell''$ result in H' and H'' having the same value for the first ℓ' bits.

2.2. *Recommended Parameter Sets*

Authenticated Encryption Table 1 lists our recommended instances for authenticated encryption and specifies their parameters, including the key size k, the fixed nonce and tags sizes, the rate r, and the number of rounds a for the initialization and finalization permutation p^a and b for the intermediate permutation p^b processing the associated data and plaintext. The list is sorted by priority: the primary recommendation is Ascon-128 and the secondary recommendation is Ascon-128a. Both schemes are identical to the CAESAR candidates [35] selected as primary choices for lightweight use-cases in the final CAESAR portfolio [87].

Hashing Table 2 lists our recommended instance for hashing and specifies its parameters, including the size of the hash output h, the rate r, as well as the number of rounds a for

Table 2. Parameters for recommended hashing algorithms.

Name	Algorithm	Bit size of		Rounds
		Hash	Data block	p^a
ASCON-HASH	$\mathcal{X}_{256,64,12}$ with $\ell = 256$	256	64	12

Table 3. Parameters for extendable output function.

Name	Algorithm	Bit size of		Rounds
		Hash	Data block	p^a
ASCON-XOF	$\mathcal{X}_{0,64,12}$ with arbitrary ℓ	ℓ	64	12

Table 4. Parameters for authenticated encryption with increased key size.

Name	Algorithms	Bit size of				Rounds	
		Key	Nonce	Tag	Data block	p^a	p^b
ASCON-80pq	$\mathcal{E}, \mathcal{D}_{160,64,12,6}$	160	128	128	64	12	6

the permutation p^a. The list is sorted by priority: the primary and only recommendation is ASCON-HASH.

Further constructions based on ASCON*'s permutation.* Besides these main recommendations listed above, it is also possible to use ASCON's permutation for other purposes and in different parameter configurations.

We define an extendable output function ASCON-XOF which uses the algorithm $\mathcal{X}_{0,64,12}$ with a rate of 64 bits and 12 rounds for p^a to produce a hash output of arbitrary length (see Table 3).

Furthermore, we define a new authenticated encryption scheme ASCON-80pq which uses the algorithms $\mathcal{E}, \mathcal{D}_{160,64,12,6}$ with an increased key size of 160 bits, a nonce and tag size of 128 bits, a rate of 64 bits, 12 rounds for p^a and 6 rounds for p^b (see Table 4).

2.3. *State and Notation*

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 (Sect. 2.6), the 320-bit state S is split into five 64-bit registers words x_i, as illustrated in Fig. 3a:

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

Table 5. Notation used for Ascon's interface, mode, and permutation.

K	Secret key K of $k \leq 160$ bits
N, T	Nonce N, tag T, all of 128 bits
P, C, A	Plaintext P, ciphertext C, associated data A (in r-bit blocks after padding P_i, C_i, A_i)
M, H	Message M, hash value H (in r-bit blocks M_i, H_i)
$\perp$	Error, verification of authenticated ciphertext failed
S	The 320-bit state S of the sponge function
S_r, S_c	The r-bit outer and c-bit inner part of the state S
p, p^a, p^b	Permutations p^a, p^b consisting of a, b update rounds p, respectively
$x \in \{0, 1\}^k$	Bitstring x of length k (variable if $k = *$)
0^k	Bitstring of k bits (variable length if $k = *$), all 0
$\|x\|$	Length of the bitstring x in bits
$\lfloor x \rfloor_k$	Bitstring x truncated to the first k bits
$\lceil x \rceil^k$	Bitstring x truncated to the last k bits
$x \\| y$	Concatenation of bitstrings x and y
$x \oplus y$	Xor of bitstrings x and y
$x \bmod y$	Remainder in integer division of x by y
$\lceil x \rceil$	Ceiling function, smallest integer larger than x
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)
$x \ggg i$	Right-rotation (circular shift) by i bits of 64-bit word x

Whenever S needs to be interpreted as a byte-array (or bitstring), it 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 5 lists the notation and symbols used in this document.

2.4. *Authenticated Encryption*

The mode of operation of Ascon for authenticated encryption is based on duplex modes like MonkeyDuplex [20], but uses a stronger keyed initialization and keyed finalization function. The encryption and decryption operations are illustrated in Fig. 1a, b and specified in Algorithm 1.

Algorithm 1: Authenticated encryption and decryption procedures

```
Authenticated Encryption
$\mathcal{E}_{k,r,a,b}(K, N, A, P)$

**Input:** key $K \in \{0,1\}^k$, $k \leq 160$,
  nonce $N \in \{0,1\}^{128}$,
  associated data $A \in \{0,1\}^*$,
  plaintext $P \in \{0,1\}^*$
**Output:** ciphertext $C \in \{0,1\}^{|P|}$,
  tag $T \in \{0,1\}^{128}$

**Initialization**
  $S \leftarrow \mathrm{IV}_{k,r,a,b} \| K \| N$
  $S \leftarrow p^a(S) \oplus (0^{320-k} \| K)$
**Processing Associated Data**
  **if** $|A| > 0$ **then**
    $A_1 \ldots A_s \leftarrow r$-bit blocks of $A\|1\|0^*$
    **for** $i = 1, \ldots, s$ **do**
      $S \leftarrow p^b((S_r \oplus A_i) \| S_c)$
  $S \leftarrow S \oplus (0^{319} \| 1)$
**Processing Plaintext**
  $P_1 \ldots P_t \leftarrow r$-bit blocks of $P\|1\|0^*$
  **for** $i = 1, \ldots, t-1$ **do**
    $S_r \leftarrow S_r \oplus P_i$
    $C_i \leftarrow S_r$
    $S \leftarrow p^b(S)$
  $S_r \leftarrow S_r \oplus P_t$
  $\tilde{C}_t \leftarrow \lfloor S_r \rfloor_{|P| \bmod r}$
**Finalization**
  $S \leftarrow p^a(S \oplus (0^r \| K \| 0^{320-r-k}))$
  $T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$
  **return** $C_1 \| \ldots \| C_{t-1} \| \tilde{C}_t, T$
```

```
Verified Decryption
$\mathcal{D}_{k,r,a,b}(K, N, A, C, T)$

**Input:** key $K \in \{0,1\}^k$, $k \leq 160$,
  nonce $N \in \{0,1\}^{128}$,
  associated data $A \in \{0,1\}^*$,
  ciphertext $C \in \{0,1\}^*$,
  tag $T \in \{0,1\}^{128}$
**Output:** plaintext $P \in \{0,1\}^{|C|}$ or $\perp$

**Initialization**
  $S \leftarrow \mathrm{IV}_{k,r,a,b} \| K \| N$
  $S \leftarrow p^a(S) \oplus (0^{320-k} \| K)$
**Processing Associated Data**
  **if** $|A| > 0$ **then**
    $A_1 \ldots A_s \leftarrow r$-bit blocks of $A\|1\|0^*$
    **for** $i = 1, \ldots, s$ **do**
      $S \leftarrow p^b((S_r \oplus A_i) \| S_c)$
  $S \leftarrow S \oplus (0^{319} \| 1)$
**Processing Ciphertext**
  $C_1 \ldots C_{t-1}\tilde{C}_t \leftarrow r$-bit blocks of $C$, $0 \leq |\tilde{C}_t| < r$
  **for** $i = 1, \ldots, t-1$ **do**
    $P_i \leftarrow S_r \oplus C_i$
    $S \leftarrow C_i \| S_c$
    $S \leftarrow p^b(S)$
  $\tilde{P}_t \leftarrow \lfloor S_r \rfloor_{|\tilde{C}_t|} \oplus \tilde{C}_t$
  $S_r \leftarrow S_r \oplus (\tilde{P}_t \| 1 \| 0^*)$
**Finalization**
  $S \leftarrow p^a(S \oplus (0^r \| K \| 0^{320-r-k}))$
  $T^* \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$
  **if** $T = T^*$ **return** $P_1 \| \ldots \| P_{t-1} \| \tilde{P}_t$
  **else return** $\perp$
```

Initialization The 320-bit initial state of Ascon is formed by the secret key K of k bits and nonce N of 128 bits, as well as an IV specifying the algorithm (including the key size k, the rate r, the initialization and finalization round number a, and the intermediate round number b, each written as an 8-bit integer):

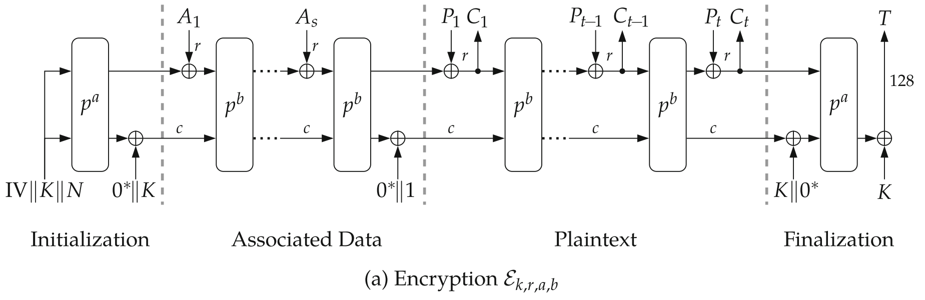
$$\mathrm{IV}_{k,r,a,b} \leftarrow k\|r\|a\|b\|0^{160-k} = \begin{cases} \texttt{80400c0600000000} & \text{for Ascon-128} \\ \texttt{80800c0800000000} & \text{for Ascon-128a} \\ \texttt{a0400c06} & \text{for Ascon-80pq} \end{cases}$$

$$S \leftarrow \mathrm{IV}_{k,r,a,b}\|K\|N$$

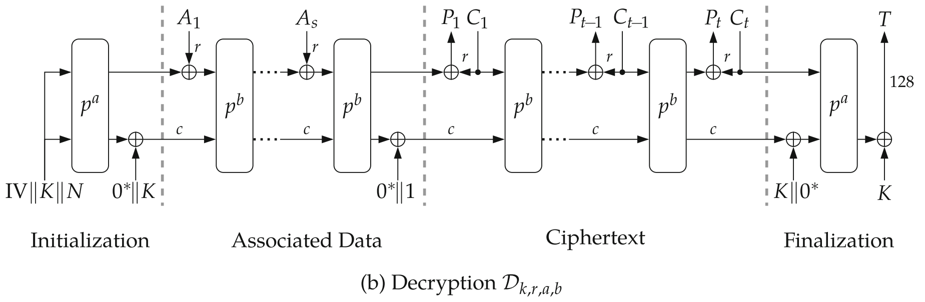
In the initialization, a rounds of the round transformation p are applied to the initial state, followed by an XOR of the secret key K:

$$S \leftarrow p^a(S) \oplus (0^{320-k}\|K)$$

Processing Associated Data Ascon processes the associated data A in blocks of r bits. It appends a single 1 and the smallest number of 0s to A to obtain a multiple of r bits



(a) Encryption $\mathcal{E}_{k,r,a,b}$



(b) Decryption $\mathcal{D}_{k,r,a,b}$

Fig. 1. Ascon's mode of operation.

and split it into s blocks of r bits, $A_1 \| \dots \| A_s$. In case A is empty, no padding is applied and $s = 0$:

$$A_1, \dots, A_s \leftarrow \begin{cases} r\text{-bit blocks of } A\|1\|0^{r-1-(|A| \bmod r)} & \text{if } |A| > 0 \\ \varnothing & \text{if } |A| = 0 \end{cases}$$

Each block A_i with $i = 1, \dots, s$ is XORed to the first r bits S_r of the state S, followed by an application of the b-round permutation p^b to S:

$$S \leftarrow p^b((S_r \oplus A_i)\|S_c), \qquad 1 \leq i \leq s$$

After processing A_s (also if $s = 0$), a 1-bit domain separation constant is XORed to S:

$$S \leftarrow S \oplus (0^{319}\|1)$$

Processing Plaintext/Ciphertext Ascon processes the plaintext P in blocks of r bits. The padding process appends a single 1 and the smallest number of 0s to the plaintext P such that the length of the padded plaintext is a multiple of r bits. The resulting padded plaintext is split into t blocks of r bits, $P_1 \| \dots \| P_t$:

$$P_1, \dots, P_t \leftarrow r\text{-bit blocks of } P\|1\|0^{r-1-(|P| \bmod r)}$$

Encryption In each iteration, one padded plaintext block P_i with $i = 1, \dots, t$ is XORed to the first r bits S_r of the internal state S, followed by the extraction of one ciphertext block C_i. For each block except the last one, the whole internal state S is transformed by the permutation p^b using b rounds:

$$\begin{aligned} C_i &\leftarrow S_r \oplus P_i \\ S &\leftarrow \begin{cases} p^b(C_i \| S_c) & \text{if } 1 \le i < t \\ C_i \| S_c & \text{if } 1 \le t \end{cases} \end{aligned}$$

The last ciphertext block C_t is then truncated to the length of the unpadded last plaintext block-fragment so that its length is between 0 and $r - 1$ bits, and the total length of the ciphertext C is exactly the same as for the original plaintext P:

$$\tilde{C}_t \leftarrow \lfloor C_t \rfloor_{|P| \bmod r}$$

Decryption In each iteration except the last one, the plaintext block P_i is computed by XORing the ciphertext block C_i with the first r bits S_r of the internal state. Then, the first r bits of the internal state, S_r, are replaced by C_i. Finally, for each ciphertext block except the last one, the internal state is transformed by the b-round permutation p^b:

$$\begin{aligned} P_i &\leftarrow S_r \oplus C_i \\ S &\leftarrow p^b(C_i \| S_c), \qquad 1 \le i < t \end{aligned}$$

For the last, truncated ciphertext block $\tilde{C}_t$ with $0 \le \ell < r$ bits, the procedure differs:

$$\begin{aligned} \tilde{P}_t &\leftarrow \lfloor S_r \rfloor_\ell \oplus \tilde{C}_t \\ S &\leftarrow (S_r \oplus (\tilde{P}_t \| 1 \| 0^{r-1-\ell})) \| S_c \end{aligned}$$

Finalization In the finalization, the secret key K is XORed to the internal state and the state is transformed by the permutation p^a using a rounds. The tag T consists of the last 128 bits of the state XORed with the last 128 bits of the key K:

$$\begin{aligned} S &\leftarrow p^a(S \oplus (0^r \| K \| 0^{c-k})) \\ T &\leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128} \end{aligned}$$

The encryption algorithm returns the tag T together with the ciphertext $C_1 \| \dots \| \tilde{C}_t$. The decryption algorithm returns the plaintext $P_1 \| \dots \| \tilde{P}_t$ only if the calculated tag value matches the received tag value.

2.5. *Hashing*

The mode of operation for hashing is the sponge construction [14]. Both the hash function ASCON-HASH with fixed output size and the extendable output function ASCON-XOF with

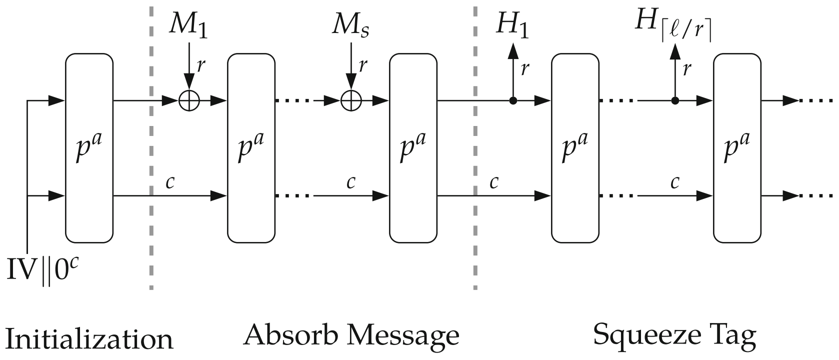


Fig. 2. Hashing mode $\mathcal{X}_{h,r,a}$ in Ascon-Hash, Ascon-Xof .

variable output size internally use the same hashing algorithm $\mathcal{X}_{h,r,a}$ (see Table 2), which is illustrated in Fig. 2 and specified in Algorithm 2.

Initialization The 320-bit initial state of Ascon-Xof and Ascon-Hash is defined by a constant IV that specifies the algorithm parameters in a similar format as for Ascon (including $k = 0$, the rate r, and round numbers a and $b = 0$, each written as an 8-bit integer), followed by the maximal output length of h bits as a 32-bit integer (with $h = \ell = 256$ for Ascon-Hash and $h = 0$ for unlimited output in Ascon-Xof) and a 256-bit zero value. The a-round permutation p^a is applied to initialize the state S:

$$\begin{aligned} \mathrm{IV}_{h,r,a} &\leftarrow 0^8 \| r \| a \| 0^8 \| h = \begin{cases} \texttt{00400c0000000000} & \text{for Ascon-Xof} \\ \texttt{00400c0000000100} & \text{for Ascon-Hash} \end{cases} \\ S &\leftarrow p^a(\mathrm{IV}_{h,r,a} \| 0^{256}) \end{aligned}$$

The initial 320-bit state S can be precomputed for each instance and we get for Ascon-Hash (left) and Ascon-Xof (right):

$$S \leftarrow \begin{matrix} \texttt{ee9398aadb67f03d} \; \| \\ \texttt{8bb21831c60f1002} \; \| \\ \texttt{b48a92db98d5da62} \; \| \\ \texttt{43189921b8f8e3e8} \; \| \\ \texttt{348fa5c9d525e140} \end{matrix} \qquad S \leftarrow \begin{matrix} \texttt{b57e273b814cd416} \; \| \\ \texttt{2b51042562ae2420} \; \| \\ \texttt{66a3a7768ddf2218} \; \| \\ \texttt{5aad0a7a8153650c} \; \| \\ \texttt{4f3e0e32539493b6} \end{matrix}$$

Absorbing Message Ascon-Xof and Ascon-Hash process the message M in blocks of r bits. The padding process is the same as for the plaintext of Ascon: it appends a single 1 and the smallest number of 0s to M such that the length of the padded message is a multiple of r bits. The resulting padded message is split into s blocks of r bits, $M_1 \| \ldots \| M_s$:

$$M_1, \ldots, M_s \leftarrow r\text{-bit blocks of } M \| 1 \| 0^{r-1-(|M| \bmod r)}$$

The message blocks M_i with $i = 1, \dots, s$ are processed as follows. Each block M_i is xored to the first r bits S_r of the state S, followed by an application of the a-round permutation p^a to S:

$$S \leftarrow p^a((S_r \oplus M_i) \| S_c), \qquad 1 \le i \le s$$

Squeezing The hash output is extracted from the state in r-bit blocks H_i until the requested output length $\ell \le h$ is completed after $t = \lceil \ell/r \rceil$ blocks. After each extraction, the internal state S is transformed by the a-round permutation p^a:

$$\begin{aligned} H_i &\leftarrow S_r \\ S &\leftarrow p^a(S), \qquad 1 \le i \le t = \lceil \ell/r \rceil \end{aligned}$$

The last output block H_t is truncated to $\ell \bmod r$ bits (unless r divides ℓ) and $H = H_1 \| \dots \| \tilde{H}_t$ returned:

$$\tilde{H}_t \leftarrow \lfloor H_t \rfloor_{\ell \bmod r}$$

Algorithm 2: Hashing

```
Extendable output function $\mathcal{X}_{h,r,a}(M, \ell) = H$
Input: message $M \in \{0,1\}^*$, output bitsize $\ell \le h$ or $\ell$ arbitrary if $h = 0$
Output: hash $H \in \{0,1\}^\ell$
Initialization
  $S \leftarrow p^a(\mathrm{IV}_{h,r,a} \| 0^c)$
Absorbing
  $M_1 \dots M_s \leftarrow M \| 1 \| 0^*$
  for $i = 1, \dots, s$ do
    $S \leftarrow p^a((S_r \oplus M_i) \| S_c)$
Squeezing
  for $i = 1, \dots, t = \lceil \ell/r \rceil$ do
    $H_i \leftarrow S_r$
    $S \leftarrow p^a(S)$
  return $\lfloor H_1 \| \dots \| H_t \rfloor_\ell$
```

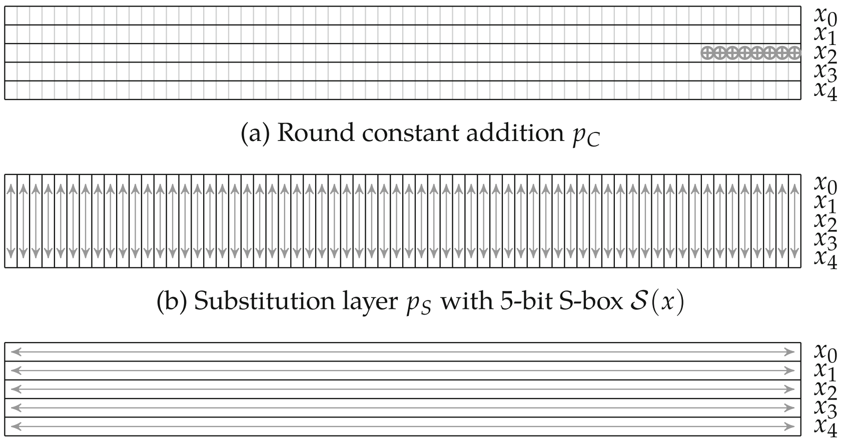
2.6. *Permutation*

The main components of the schemes Ascon, Ascon-Xof, and Ascon-Hash are the two 320-bit permutations p^a and p^b. The permutations iteratively apply an SPN-based round transformation p that in turn consists of three steps p_C, p_S, p_L:

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

p^a and p^b differ only in the number of rounds. The number of rounds a and the number of rounds b are tunable security parameters.

For the description and application of the round transformations, the 320-bit state S is split into five 64-bit registers words x_i, $S = x_0 \| x_1 \| x_2 \| x_3 \| x_4$ (see Fig. 3).



(a) Round constant addition p_C

(b) Substitution layer p_S with 5-bit S-box $\mathcal{S}(x)$

(c) Linear layer with 64-bit diffusion functions $\Sigma_i(x_i)$

Fig. 3. Register words of the 320-bit state S and operations $p_L \circ p_S \circ p_C$.

Table 6. 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`

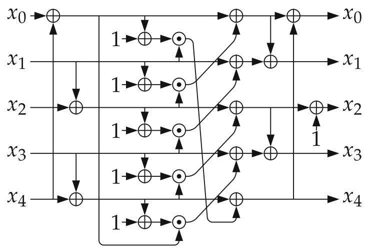
Table 7. Ascon's 5-bit S-box $\mathcal{S}$ as a lookup table.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$\mathcal{S}(x)$	4	b	1f	14	1a	15	9	2	1b	5	8	12	1d	3	6	1c	1e	13	7	e	0	d	11	18	10	c	1	19	16	a	f	17

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 (see Fig. 3a). 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 6):

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

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 Fig. 4a to each bit-slice of the five registers $x_0 \ldots x_4$ (Fig. 3b). It is typically implemented in this bitsliced form with operations performed on the entire 64-bit words, as in the example code in Fig. 5. The lookup table of $\mathcal{S}$ is given in Table 7, where x_0 is the MSB and x_4 the LSB.



(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 $\Sigma_i(x_i)$

Fig. 4. Ascon's substitution layer and linear diffusion layer.

Linear Diffusion Layer The linear diffusion layer p_L provides diffusion within each 64-bit register word x_i (Fig. 3c). It applies a linear function $\Sigma_i(x_i)$ defined in Fig. 4b to each word x_i:

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

3. Security Claims

3.1. *Authenticated Encryption*

All Ascon family members provide 128-bit security in the notion of nonce-based authenticated encryption with associated data (AEAD), that is, they protect the confidentiality of the plaintext (except its length) and the integrity of ciphertext including the associated data (under adaptive forgery attempts). The number of processed plaintext and associated data blocks protected by the encryption algorithm is limited to a total of 2^{64} blocks per key, which corresponds to 2^{67} bytes (for Ascon-128, Ascon-80pq) or 2^{68} bytes (for Ascon-128a). We consider this as more than sufficient for lightweight applications in practice. In order to fulfill the security claims stated in Table 8, implementations must take care that the nonce (public message number) is never repeated for two encryptions under the same key, and that decrypted plaintexts are only released after successful verification of the final tag. The difference between the family members is in their robustness against other adversaries beyond the classical security claim and is discussed in the following. In particular, Ascon-128a offers a higher throughput at the cost of reduced robustness.

Ascon has been designed for robust security in case of certain implementation errors that violate these requirements, such as repeated nonces. For instance, the security claims of Table 8 can even be fulfilled if nonces are reused a few times by accident as long as the combination of nonce and associated data stays unique. Furthermore, even a full recovery of a single secret state during the processing of the associated data, plaintext, or ciphertext (e.g., with implementation attacks) does not imply practical global attacks such as key recovery or forgeries without significant additional computations. In this case, forgeries can be obtained with complexity $2^{c/2}$ by constructing collisions on the

Table 8. Security claims for authenticated encryption parameter sets.

Requirement	Security in bits		
	Ascon-128	Ascon-128a	Ascon-80pq
Confidentiality of plaintext	128	128	128
Integrity of plaintext	128	128	128
Integrity of associated data	128	128	128
Integrity of public message number	128	128	128

inner part, so the robustness of Ascon-128a ($c = 192$) is lower than that of Ascon-128 ($c = 256$). The same holds for key recovery attacks. We do not expect that key recovery attacks for Ascon-128a and Ascon-128 can be found with complexity significantly below $\min(2^k, 2^{c/2})$ (2^{96} and 2^{128}, respectively) even if a few internal states can be recovered.

Except for the single-use requirement, there are no constraints on the choice of the nonce (public message number); in particular, 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 some classes of implementation attacks.

As for most encryption algorithms, the ciphertext length leaks the plaintext length since the two lengths are equal (excluding the tag length). If the plaintext length is confidential, users must compensate this by padding their plaintexts.

We emphasize that we do not require ideal properties for the permutations p^a, p^b. Non-random properties of the permutations p^a, p^b are known and do not afflict the claimed security properties of the entire encryption algorithm. For a detailed security analysis of Ascon, we refer to Sect. 6.

3.2. *Hashing*

Both Ascon-Hash and Ascon-Xof provide 128-bit security against collision attacks and (second) pre-image attacks, as stated in Table 9. Note that the security of Ascon-Xof is reduced if the output size ℓ is less than 256 bits. Like other sponge-based hash functions, both Ascon-Hash and Ascon-Xof also resist length extension attacks and second-preimage attacks for long messages.

Table 9. Security claims for recommended parameter configurations of Ascon-Hash with 256-bit hash output and Ascon-Xof with an output size of ℓ bits.

Requirement	Security in bits	
	Ascon-Hash	Ascon-Xof
Collision resistance	128	$\min(128, \ell/2)$
(Second) Pre-image resistance	128	$\min(128, \ell)$

Like for authenticated encryption, we emphasize that we do not require ideal properties for the permutations. Non-random properties of the permutations are known and do not afflict the claimed security properties of Ascon-Hash and Ascon-Xof.

3.3. *A Note on Post-Quantum Security*

Since Ascon is targeting lightweight applications, we do not claim resistance against all possible quantum attacks. However, Ascon provides enough robustness and agility to provide basic resistance against certain classes of attacks once the availability of appropriate quantum computer resources become evident.

Therefore, the Ascon suite provides an additional AEAD scheme Ascon-80pq. The only difference between Ascon-80pq and Ascon-128 is the increased key length of 160 bits. This increased key length provides additional protection against Grover's algorithm for exhaustive key search [57]. The resulting security against Grover's key search is about 80 bits. Since all other tunable security parameters (the number of rounds of the permutations) are not changed, the security claim for Ascon-80pq against classical attacks stays the same as for Ascon-128, which is 128 bits (see Table 8).

Furthermore, Ascon-Hash and Ascon-Xof use the sponge construction with a capacity of $c = 256$ bits and an output size of ℓ bits (with $\ell = 256$ for Ascon-Hash). The post-quantum security of the sponge construction has been analyzed in [24]. The resulting estimated bound is $\min(c/3,\ \ell/3)$. However, note that their proof applies to sponge functions with non-invertible functions instead of a permutation as in Ascon. In [22], the complexity of finding collisions using a quantum computer is assumed to be $2^{\ell/3}$. Although the complexity of the best quantum attack on the sponge construction using a random permutation is unknown, no quantum attack with a complexity below $\min(c/3,\ \ell/3)$ is known yet.

4. Features

The Ascon suite supports authenticated encryption and hashing with the same lightweight permutation. Ascon-128 and Ascon-128a have been selected as the "primary choice" for lightweight authenticated encryption in the final portfolio of the CAESAR competition. Ascon achieves high security and robustness in practice with a very low area footprint in hardware while providing good performance in both software and hardware implementations, particularly for short messages. We believe that ciphers which operate efficiently and securely on very resource-constrained devices, on modern high-end systems, and also in the area between these two extremes will be of rising importance in the future. A typical example for such dual environments is the Internet of Things (IoT), where a large number of very constrained devices need to communicate efficiently with high-performance back-end servers. In the following, we summarize the most important properties of Ascon and justify that the cipher suite is a perfect fit for such applications.

4.1. *Properties of* Ascon

- *Authenticated Encryption and Hashing* Ascon offers authenticated encryption and hashing with the same underlying permutation. Sharing a single primitive for all schemes not only reduces the area requirements for hardware implementations that want to provide both, but also allows to restrict the code base that has to be maintained. This reduces the workload necessary for efficient and secure implementations.

- *High Cryptanalytic Security* Ascon-128 and Ascon-128a have been selected as the "primary choice" for lightweight authenticated encryption in the final portfolio of the CAESAR competition after five years of evaluation. During this competition, Ascon and its permutation have undergone a thorough public analysis. So far, this has resulted in more than 15 publications giving insight in the security of Ascon(see Sect. 6.4). All existing analysis shows a comfortable security margin, with no indication of weaknesses regarding Ascon-128 and Ascon-128a. A detailed discussion about the security of Ascon can be found in Sect. 6.
- *Simplicity* Ascon is natively defined on 64-bit words using only the bitwise Boolean functions AND, XOR, NOT, and ROT (bitwise rotation). This significantly reduces the effort of implementing the algorithm on new target platforms.
- *Lightweight and Flexible in Hardware* Current implementation results show that Ascon provides excellent implementation characteristics in terms of size and speed. Balanced round-based CAESAR API implementations of Ascon-128 and Ascon-128a achieve a throughput of 4.9–7.3 Gbps using less than 10 kGE. Due to the small state size and the elegant structure of Ascon's round function, it is additionally possible to provide hardware implementations that are optimized toward either a smaller area or higher speed. More details about hardware implementations are provided in Sect. 7.
- *Efficient in Software* Ascon is designed to facilitate bitsliced software implementations. Its internal 64-bit operations are also well-suited for processors with smaller word sizes, and can take advantage of pipelining and parallelization features of high-end processors. In particular, the substitution and linear layers have been specifically designed to support high instruction parallelism. In addition, the small state of Ascon allows to hold the whole state within the CPU's registers for a wide range of platforms, thus reducing reloads from the cache to a minimum. Further discussions about the performance in software can be found in Sect. 7.
- *Balanced Cross-Platform Design* Ascon follows a balanced design approach, instead of optimizing for only one particular platform or use-case at the cost of efficiency on other platforms. In particular, Ascon has been designed to provide lightweight implementation characteristics in both hardware and software while still offering competitive performance on both. Hence, Ascon is highly suited for scenarios where many lightweight devices communicate with a back-end server, a typical use-case in the Internet of Things (IoT).
- *Easy Integration of Side-Channel Countermeasures* Ascon can be implemented efficiently for platforms and applications where side-channel resistance is important. The very efficient bitsliced implementation of the S-boxes prevents cache-timing attacks, since no lookup tables are required. Furthermore, the low algebraic degree of the S-box facilitates both first- and higher-order protection using masking or sharing-based side-channel countermeasures. More information about the integration of countermeasures against implementation attacks can be found in Sect. 7.

- *Robust Security in Practice* Ascon's sponge-based mode of operation for nonce-based authenticated encryption features a strengthened keyed initialization and finalization. This improves the cipher's robustness in case of misuse attacks, for example against a nonce-reuse attacker. A potential recovery of the secret state during data processing due to misuse attacks thus does not directly lead to a key-recovery or universal forgery.
- *Online and Single-Pass* All Ascon algorithms are online and can process the data blocks before the complete input or its length are known. For both Ascon encryption and decryption, just one pass over the data is required.
- *Inverse-Free* Ascon does not need to implement any inverse operations since the permutations p^a and p^b are only evaluated in one direction for both encryption and decryption, which significantly reduces the area overhead.
- *High Key Agility* Ascon does not use a key schedule or expand the key by any other means, so there are no hidden setup costs when the key is changed.

4.2. *Features for Lightweight Applications*

- *Small hardware area* Ascon's small state and simple round function are well-suited for small implementations, without compromising on the full security of 128 bits. Existing lightweight implementations of Ascon's authenticated encryption functionality are as small as 2.6 kGE [59]. The round-based implementations are smaller than 10 kGE and still offer a throughput of 4.9–7.3 Gbps, which is already sufficient to encrypt a Gigabit Ethernet connection. More details about (protected) hardware implementations are provided in Sect. 7.
- *Reuse of core component* Implementing the Ascon permutation once is enough to get authenticated encryption as well as decryption with a very small overhead, since decryption does not require the inverse of the permutation (that is, Ascon is inverse-free). Together with Ascon-Hash and Ascon-Xof, it also provides hashing functionality using the same permutation.
- *Efficiency in hardware* Ascon is not only small and fast, but can also be efficiently implemented on a wide variety of platforms [49]. It allows many trade-offs between throughput, latency, gate count, power consumption, etc. [59]. Comparison of implementation results in [49] show that throughput per area of both Ascon variants is very good compared to many other CAESAR candidates. Further discussion about the performance in hardware can be found in Sect. 7.
- *Bit-interleaved implementations* Ascon's permutation is naturally defined on 64-bit words, with rotation operations performed on them and hence, lends itself to natural bitsliced implementations on 64-bit processors. However, on architectures with a smaller word-size, it is possible to implement Ascon using bit interleaving as introduced for Keccak [13]. In short, bit interleaving involves sorting the single bits in n registers of $64/n$ bits (with $n = 2, 4, 8$), such that a single rotation on one 64-bit word can be implemented using n rotations on each of the n $(64/n)$-bit words. Hence, when neglecting the effort to interleave the bits, the number of operations per round on smaller architectures only increases roughly to $n \cdot \ell$, where ℓ are the number of operations needed with 64-bit registers. Thus, Ascon allows not only for fast bitsliced implementations on 64-bit processors, but also allows

for fast bitsliced implementations on smaller architectures that do not require any data-dependent lookup tables. Further insights about the performance in software on various platforms are given in Sect. 7.

- *Natural side-channel protection* This is one of the primary design goals of Ascon. For protection against side-channel attacks, it is important that the S-box is easy to protect. Ascon's S-box has a low algebraic degree of 2 and a low number of Boolean multiplications, which is well-suited for threshold implementations [75] and similar protection approaches. More information about the integration of countermeasures against implementation attacks can be found in Sect. 7.4.
- *Limited damage in misuse settings* Ascon uses nonce-based authenticated encryption. As with any nonce-based authenticated encryption scheme, repeating nonces is a misuse setting, and implies a loss of semantic security. But compared to other sponge-based constructions, Ascon provides better robustness in case of a potential state recovery, since both initialization and finalization are keyed additionally. A recovery of the secret state during data processing does not directly lead to a key-recovery or universal forgery. Furthermore, Ascon's mode is compatible with alternative decryption interfaces for secure implementations in memory-constrained settings [2].
- *Low overhead for short messages* Ascon is among the fastest CAESAR candidates for short messages according to current software benchmarking results[1,45], since its initialization and finalization overhead is much smaller compared to most blockcipher-based constructions, stream ciphers, or large-state sponge functions. For instance, if the associated data is empty, no additional permutation calls are necessary. Ascon's small rate of 8 or 16 bytes is ideally suited for short messages that are typical for lightweight applications.

4.3. *Features for High-Performance Applications*

- *Efficiency on modern CPUs* The bitsliced design of Ascon using simple instructions makes it easy to implement efficiently on a wide range of platforms. The native word-size of Ascon is 64 bits, which makes it especially efficient on high-end CPUs. Up to five instructions can be carried out in parallel in nearly every step of the permutation which makes Ascon fast in software on 64-bit as well as 32-bit CPUs. Further insights about the performance in software on various platforms are given in Sect. 7.
- *Efficiency on dedicated hardware* The linear and nonlinear layer in Ascon are designed to use a small number of simple bitwise Boolean functions. Hence, it is easy to build dedicated hardware or reuse SIMD instructions for Ascon.
- *Natural side-channel protection* Ascon is a bitsliced design with a small state size, which means that straightforward software implementations require no data-dependent table lookups or other cache accesses. On many platforms, all data can be kept in registers during computations. This is for instance important in cloud applications to prevent cross-VM attacks and other cache-based attacks.

5. Design Rationale

The main goal of the Ascon suite is a very low memory footprint in hardware and software, while still being fast, robust, and secure with a well-analyzed and generous security margin. The design rationale behind Ascon is to provide the best trade-off between security, size and speed in both software and hardware, with a focus on size.

The Ascon suite is based on the sponge design methodology [14]. The permutation of Ascon uses an iterated substitution-permutation-network (SPN), which provides good cryptographic properties and fast diffusion at a low cost. To provide these properties, the main components of Ascon are inspired from standardized and well-analyzed primitives. The substitution layer uses an affine equivalent of the S-box used in the χ mapping of Keccak [19,27] designed to add diffusion. The permutation layer uses linear functions similar to the Σ functions used in SHA-2. The resulting design has itself been thoroughly analyzed during the CAESAR competition, and the published results show a comfortable security margin. Details on the design principles for each component are given in the following sections.

5.1. *Design of the Modes*

Choice of the Mode for Authenticated Encryption The design principles of Ascon's authenticated encryption mode follow the sponge methodology [14]. More precisely, they are similar to SpongeWrap [17] and MonkeyDuplex [20]. The sponge-based design has several advantages compared to other available construction methods like some blockcipher- or hash-based modes and other dedicated designs:

- The sponge construction is well-studied and has been analyzed and proven secure for different applications in a large number of publications. Moreover, the sponge construction is used in the SHA-3 winner Keccak.
- Flexible to adapt for other functionality (hash, MAC, cipher).
- Elegant and simple design, clear state size, no key schedule.
- Plaintext and ciphertext blocks can both be computed online, without waiting for the complete message or even the message length.
- Little implementation overhead for decryption, which uses the same permutations as encryption.
- Weak round transformations can be used to process additional plaintext blocks, improving the performance for long messages.

Compared to other sponge-based authenticated encryption designs, Ascon uses a stronger keyed initialization and keyed finalization phase. As a result, even in case an attacker somehow manages to recover the internal state during data processing (e.g., due to side-channel attacks), this does not directly lead to the recovery of the secret key or forgeries without significant additional computations. To allow this additional robustness, Ascon has to set the possibility of full state absorption aside. However, we value robustness for lightweight use-cases more than a potential increase in performance.

The addition of $0^{319}\|1$ after the last processed associated data block and the first plaintext block acts as a domain separation to prevent attacks that change the role of plaintext and associated data blocks.

If no associated data and only an incomplete plaintext block is present, the two initialization and finalization calls to p^a are sufficient and no additional intermediate round transformation p^b is needed. To prevent that key additions between the two applications of p^a cancel each other out in this case, they are added to different parts of the inner part S_c of the state.

Choice of the Mode for Hashing and Extendable Output Function As Ascon-128 and Ascon-128a are already well-established as the primary recommendations for lightweight use-cases in the final portfolio of the CAESAR competition, we extend the functionality that can be provided by using the same well-analyzed permutation. It is a natural decision to also base the hashing and extendable output functionality on sponge functions [14]. For hashing, sponge functions provide similar benefits as for authenticated encryption:

– The sponge construction is well-studied and has been analyzed and proven secure for different applications in a large number of publications. Moreover, the sponge construction is used in the SHA-3 winner Keccak.
– The core component (permutation) can be reused if Ascon for authenticated encryption is already implemented, reducing the implementation overhead.
– The elegant and simple design has an obvious state size.
– The construction is flexible to adapt for other functionalities (authenticated encryption, MAC, cipher).
– Message blocks can be processed online, without waiting for the complete message or even the message length initially be present.

Choice of the Family Members The Ascon suite is built around the well-analyzed authenticated encryption schemes Ascon-128 and Ascon-128a. The newly added schemes Ascon-80pq, Ascon-Hash, and Ascon-Xof are designed to provide the same security level as Ascon-128 and Ascon-128a, which is 128 bits of security against attacks in the classical setting (e.g., no quantum computers are available), as detailed in Sect. 3.

The rationale behind this is that 128 bits of security against classical attacks is generally considered to provide enough security for lightweight applications for the next decades. Furthermore, choosing and providing instances that give more security against classical attacks would require more resources without providing any benefit in the foreseeable future, which contradicts the use of lightweight ciphers in the first place. In the following, we still justify our decision in providing three different instances Ascon-128, Ascon-128a, and Ascon-80pq for authenticated encryption with the same security level.

Ascon-128 and Ascon-128a provide the same level of security in a black-box scenario if the nonce is used correctly, but there is a trade-off regarding performance and robustness. Ascon-128a doubles the rate compared to Ascon-128, at the cost of slightly more rounds in p^b. This decreases the capacity, which also reduces the robustness of the scheme. For example, if an attacker manages to recover a single internal state during the processing of associated data or plaintext/ciphertext, the decreased capacity of Ascon-128a leads to a slight benefit in finding collisions to compute forgeries (complexity 2^{96} vs. 2^{128}). Note that a single state recovery still does not directly lead to an efficient key recovery attack on the scheme.

Ascon-Hash and Ascon-Xof reuse the 12-round variant of the Ascon permutation from the initialization and finalisation of Ascon-128 and Ascon-128a. Pairing this well-scrutinized building block with the extensively analyzed and researched sponge construction [14,15] provides a well-secured and efficient hash function. We have also defined an extendable output function Ascon-Xof in addition to the fixed-size hash function Ascon-Hash, since the sponge construction naturally allows this functionality and it may be more useful in practice.

A message authentication code (MAC) can be constructed from Ascon-Hash in a straightforward way using the KMAC construction [73]. However, using the sponge construction, a full state absorption is also possible [43], which can improve the efficiency of an Ascon-based MAC significantly.

Choice of the Initial Values The main purpose of the initial values is to provide a separation of the different instances. For all schemes of the Ascon suite, the IV is added to the first word and encodes parameters of the scheme such as the key length, rate, number of rounds, or hash output length. The IV provides a separation between the different primitives. In the case of Ascon-Hash and Ascon-Xof, the first call on the permutation including the IV is done without any data and hence, an equivalent initial state can be precomputed, stored and used instead.

5.2. *Design of the Permutation*

Ascon's permutation consists of three layers: the round constant addition p_C, the substitution layer p_S, and the linear layer p_L. The substitution layer provides nonlinearity and additionally doubles as a diffusion layer along the vertical axis, between words. The linear layer then provides diffusion along the horizontal axis, within words. In the following, we detail the rationale for these individual layers.

Choice of the Round Constants The round constants have been chosen large enough to avoid slide, rotational, self-similarity or similar attacks. Their values were chosen in a simple, obvious way (increasing and decreasing counter for the two halves of the affected byte), which makes them easy to compute using a simple counter and inversion operation. Their low entropy is an indicator that the constants are not used to implement backdoors. The pattern can also easily be extended for up to 16 rounds if a higher security margin is desired.

The position for inserting the round constants (in word x_2) was chosen so as to allow pipelining with the next or previous few operations (message injection in the first round or the following instructions of the bit-sliced S-box implementation).

Choice of the Substitution and Vertical Diffusion Layer The substitution layer contains the only nonlinear part of the round transformation. It mixes a sequence of 5 bits, each at the same bit position of the five state words, using 5-bit S-boxes. The S-box was designed according to the following criteria:

- Invertible and no fix-points,
- Efficient bit-sliced implementation with few, well pipelinable instructions,
- Each output bit depends on at least 4 input bits,
- Algebraic degree 2 to facilitate threshold implementations and masking,
- Maximum differential probability and linear bias 1/4,
- Differential and linear branch number 3,
- Avoid trivially iterable differential properties in the data injection positions.

The χ mapping of KECCAK fulfills several of these properties and is already well-analyzed. In addition, the χ mapping is highly parallelizable and has a compact description with relatively few instructions. This makes χ fast in both software and hardware. The drawback of χ in this context are its differential and linear branch numbers (both 2), a fix-point at value zero, and that each output bit only depends on 3 input bits (only two of them nonlinearly).

For a better interaction with the linear layer of ASCON and a better trade-off between performance and security, we require a branch number of 3. This and the other additional requirements can be achieved without destroying other properties by adding lightweight affine transformations to the input and output of χ. Adding this affine transformation to the substitution layer (mixes columns) instead of the linear layer (mixes rows) results in a simpler design which also facilitates the analysis of ASCON.

The costs of these affine transformations are quickly amortized since a branch number of 3 (together with an according linear layer) essentially doubles the number of active S-boxes from round to round (in sparse trails). There are only a handful of options for a lightweight transformation (few XOR operations) that achieve both required branch numbers. We experimentally selected the candidate that provided the best diffusion in combination with the selected linear layer.

The bit-sliced design of the S-box has several benefits: it is highly efficient to implement parallel invocations on 64-bit processors (and other architectures), and no lookup tables are necessary. This effectively precludes typical cache-timing attacks for software implementations.

The algebraic degree of 2 theoretically makes the S-box more prone to analysis with algebraic attacks. However, we did not find any practical attacks. We consider it more important to allow efficient implementation of side-channel countermeasures, such as threshold implementation [75] and masking [25,54], which are facilitated by the low degree.

The differential and linear probabilities of the S-box are not ideal, but using one of the available 5-bit AB/APN functions like in Fides [8] was not an option due to their much more costly bit-sliced implementation. Considering the relatively lightweight linear layer, repeating more rounds of the cheaper, reasonably good S-box is more effective than fewer rounds of a perfect, but very expensive S-box.

Choice of the Horizontal Linear Diffusion Layer The linear diffusion layer mixes the bits within each 64-bit state word. For resistance against linear and differential cryptanalysis, we required a branch number of at least 3. Additionally, the interaction between the linear layers for separate words should provide very good diffusion, so different linear functions are necessary for the 5 different words. These requirements should be achieved at a minimal cost. Although simple rotations are almost for free in hardware and relatively cheap in software, the slow diffusion requires a very large number of rounds. Moreover, the best performance can be achieved by balancing the costs of the substitution and linear layer.

On the other hand, mixing layers as used in AES-based designs provide a high branch number, but are too expensive to provide an acceptable speed at a small size. The mixing layer of KECCAK is best used with a large number of large words. Other possible candidates are the linear layers of Luffa [44], Hamsi [65], or other SPN-based designs. However, these candidates were either too slow or provide a less optimal diffusion.

The linear diffusion layer and rotation values in Ascon have been chosen similar to the Σ functions in SHA-2 [71,72]. These functions offer a branch number of 4. Additionally, if we choose one rotation constant of each Σ function to be zero, the performance can be improved while the branch number stays the same. On the other hand, the cryptographic strength can be improved by using different rotation constants for each 64-bit word without sacrifice on the performance. In this case, the branch number of the substitution and linear layer amplify each other which increases the minimum number of active S-boxes. We have chosen the rotation constants to achieve a good diffusion after 3 rounds of Ascon.

6. Security Analysis

The Ascon authenticated cipher with its permutations p^a, p^b was first published as a submission to the CAESAR competition in 2014. Since then, the cryptographic research community has published numerous analyses of Ascon's design. The results have confirmed Ascon's security with a generous security margin. We provide a summary of the best results in Sect. 6.1. For a list of publications and comments, we refer to Sect. 6.4. In Sect. 6.2, we discuss the security of the modes of operation for authenticated encryption and hashing. In Sect. 6.3, we provide details on the cryptanalytic properties of the Ascon permutation and their relevance for attacks.

6.1. *Overview of Best Known Attacks*

Table 10 summarizes the best published attacks on the Ascon permutation as well as on the Ascon authenticated encryption, Ascon-Hash, and Ascon-Xof. As stated in the original design document, Ascon's permutations are not considered to be ideal 320-bit permutations. However, when used in the recommended modes of operation, Ascon retains a generous security margin. The currently best cryptanalytic attacks on the Ascon authenticated encryption (excluding misuse scenarios) can recover the secret key with a time complexity of about 2^{104} only if the initialization is reduced to 7 of 12 rounds, which corresponds to a security margin of 5 rounds or 42%.

6.2. *Analysis of the Modes*

Hashing and Extendable Output Function The mode of operation in Ascon-Hash and Ascon-XofUses the sponge construction proposed by Bertoni et al. [14] and profits from the extensive literature on sponge functions, particularly the results on its indifferentiability up to about $2^{c/2}$ calls to the permutation or its inverse, where c is the capacity in bits [15].

Authenticated Encryption The mode of operation in Ascon is based on the duplex construction [17], or more specifically, a variant of the AEAD mode MonkeyDuplex with its reduced number of rounds in the data processing phases [20]. In contrast to MonkeyDuplex, however, Ascon's mode uses a double-keyed initialization and double-keyed finalization in order to improve the robustness of the scheme.

Table 10. Best known analysis of the Ascon modes and permutation.

Type	Target	Rounds	Time	Method	Reference	
(a) Best known analysis of the Ascon *permutation*						
Distinguisher	Permutation	12/12	2^{130}	Zero-sum	[34]	
	Permutation	11/12	2^{315}	Integral	[91]	
	Permutation	5/12	2^{193}	Differential	Section 6.3	
	Permutation	5/12	2^{189}	Linear	[31]	
Distinguisher	Permutation	11/12	2^{85}	Zero-sum	Section 6.3	
	Permutation	7/12	2^{65}	Integral	[91]	
	Permutation	5/12	2^{109}	Truncated Differential	[89]	
	Permutation	4/12	2^{107}	Differential	Section 6.3	
	Permutation	4/12	2^{101}	Linear	[31]	
Distinguisher	Permutation	5/12	–	Zero-Correlation	Section 6.3	
	Permutation	5/12	–	Impossible Differential	Section 6.3	
	Permutation	3/12	–	Subspace Trails	[69]	
(b) Best known analysis of Ascon *authenticated encryption modes (*⊘ = misuse*)*						
Key recovery	Ascon initialization	7/12	2^{104}	Cube-like	[67]	
	Ascon initialization	5/12	2^{36}	Diff.-linear	[34]	
	Ascon initialization	7/12	2^{97} ⊘	Cube-like	[70]	
Forgery	Ascon finalization	4/12	2^{101}	Differential	[34]	
	Ascon finalization	6/12	2^{33} ⊘	Cube tester	[70]	
State recovery	Ascon-128a iteration	2/8	—	Sat-Solver	[42]	
	Ascon-128 iteration	5/6	2^{66} ⊘	Cube-like	[70]	
(c) Best known analysis of Ascon-Hash *and* Ascon-Xof *(*⊘ = chosen IV*).*						
Preimage	Ascon-Xof	64	2/12	2^{39}	Cube-like	[37]
	Ascon-Xof	64	6/12	$2^{63.3}$	Algebraic	[37]
Collision	Ascon-Hash	256	2/12	2^{125}	Differential	[95]
	Ascon-Xof	64	2/12	–	Differential	[95]
	Ascon-Xof	all	4/12	– ⊘	Differential	[37]

AEAD modes using the duplex construction have also enjoyed considerable attention from the research community, and several security proofs with different bounds have been provided. The first proofs indicate that the duplex modes can provide security beyond the birthday bound on the capacity c, as long as the online data complexity remains well-below this birthday bound $2^{c/2}$ [18,62]. Andreeva et al. [3] show that the time complexity is at least $\min\{2^k, 2^c/\mu\}$, where μ is the multiplicity [16], which is small for nonce-based schemes.

Daemen et al. [43] provide stronger bounds based on distinguishing Ascon's mode from an Ideal Extendable Input Function (IXIF) and consider a multi-user setting as well as both respecting and misuse adversaries. Their results show that (without considering robustness and the specifics of the permutation) the data limit or key size could be further increased.

The main difference from other duplex-based modes of operation is the double-keyed initialization and finalization. As a result, even if an attacker manages to recover the internal state in some way (e.g., with implementation attacks such as side-channels or with misuse attacks such as massive nonce reuse or release of unverified plaintext), this attack cannot easily be extended to key recovery or forgeries without significant additional computations. Possible attacks after such a state recovery include forgeries by finding internal collisions ($2^{c/2}$ time).

6.3. *Analysis of the Permutation*

Differential and Linear Properties Ascon's permutation design is based on two lightweight operations with non-ideal individual differential and linear properties, but with good combined properties. The best known characteristics with probability $> 2^{-128}$ cover 4 rounds of the permutation.

DDT and LAT Table 11a lists the differential distribution table (DDT) of the Ascon S-box. The maximum differential probability of the S-box is 2^{-2} and its differential branch number is 3. Table 11b lists the linear approximation table (LAT). The maximum linear bias of the S-box is 2^{-2} and its linear branch number is 3.

The differential and linear branch number of the linear Σ_i functions is 4.

Characteristics and Active S-Boxes The minimum number of active S-boxes of 3 rounds is 15 (for differential characteristics) and 13 (for linear characteristics).

For results on more than 3 rounds, we used heuristic search tools to find good differential and linear characteristics for more rounds to get close to the real bound. The results are listed in Table 12. The best differential and linear characteristics for 4 rounds are given in Table 13a, b, respectively. We could not find any differential and linear characteristics for 5 or more rounds with less than 64 active S-boxes. The best differential and linear characteristics, we could find for 5 rounds already have 78 and 67 active S-boxes, respectively. However, note that due to the larger search space for 5 rounds, we restricted our search to differential and linear trails that are sparse in the middle (rounds 1–3).

Characteristics with Constraints Besides the differential propagation in Ascon, an attacker is in particular interested in collision-producing differentials, i.e., differentials with only differences in the outer part S_r of the state at the input and output of p^b, since such differentials might be used for a forgery attack on the authenticated encryption scheme. However, considering the good differential properties of p^b and the results of the previous sections, it is very unlikely that such differentials with a good probability exist. The best collision-producing differential trails we could find for p^b in Ascon-128 (Table 14a) and Ascon-128a (Table 14b) using a heuristic search algorithm have 117 and 192 active S-boxes, respectively.

For forgery attacks on Ascon's finalization, the input difference must be in the outer part, but there are no restrictions on the output difference. A corresponding characteristic for 4 out of 12 rounds is provided in Table 14c [34].

Impossible Differentials and Zero-Correlation Approximations Using an automated search tool, we were able to find impossible differentials [64] for up to 5 rounds (Table 15a) and zero-correlation linear hulls [23] for up to 5 rounds (Table 15b) of the permutation. It is possible that impossible differentials for more rounds exist. However, we have not found any practical attacks on Ascon using this property of the permutation.

Table 11. Differential and linear profile of the Ascon S-box.

(a) Differential distribution table: $\mathrm{DDT}[\alpha, \beta] = |\{x : \mathcal{S}(x \oplus \alpha) \oplus \mathcal{S}(x) = \beta\}|$

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
0	32	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
1	·	·	·	·	·	·	·	·	·	4	·	4	·	4	·	4	·	·	·	·	·	·	·	·	4	·	4	·	4	·	4	·
2	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	4	·	4	·	4	·	4	·	4	·	4	·	4	·	4
3	·	4	·	·	·	4	·	·	·	4	·	·	·	4	·	·	4	·	·	·	4	·	·	·	4	·	·	·	4	·	·	·
4	·	·	·	·	·	·	8	·	·	·	·	·	·	·	8	·	·	·	·	·	·	·	8	·	·	·	·	·	·	·	8	·
5	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	4	·	4	4	·	4	·	4	·	4	·	·	4	·	4
6	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2
7	·	·	4	4	·	·	4	4	·	·	4	4	·	·	4	4	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	4	4	·	·	·	·	·	·	4	4	·	·	·	·	·	·	4	4	·	·	·	·	·	·	4	4
9	·	2	·	2	2	·	2	·	2	·	2	·	·	2	·	2	2	·	2	·	·	2	·	2	·	2	·	2	2	·	2	·
a	·	2	2	·	2	·	·	2	·	2	2	·	2	·	·	2	·	2	2	·	2	·	·	2	·	2	2	·	2	·	·	2
b	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2
c	·	8	·	·	·	·	·	·	8	·	·	·	·	·	·	·	8	·	·	·	·	·	·	·	·	8	·	·	·	·	·	·
d	·	2	·	2	·	2	·	2	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	·	2	·	2	·	2	·	2
e	·	4	4	·	4	·	·	4	·	·	·	·	·	·	·	·	·	4	4	·	4	·	·	4	·	·	·	·	·	·	·	·
f	·	·	·	·	·	·	·	·	4	4	·	·	4	4	·	·	·	·	·	·	·	·	·	·	4	4	·	·	4	4	·	·
10	·	·	·	·	·	·	·	·	·	8	·	8	·	·	·	·	·	·	·	·	·	·	·	·	8	·	8	·	·	·	·	·
11	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	8	·	8	·	8	·	8	·	·	·	·	·	·	·	·
12	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·
13	·	·	8	·	8	·	·	·	·	·	8	·	8	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
14	·	·	·	·	4	4	4	4	·	·	·	·	4	4	4	4	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
15	·	·	·	·	·	4	·	4	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·	·	·	·	·	·	·	4	·	4
16	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
17	·	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·
18	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2
19	·	·	·	4	·	·	4	·	4	·	·	·	·	4	·	·	4	·	·	·	·	4	·	·	·	·	·	4	·	·	4	·
1a	·	2	2	·	·	2	2	·	2	·	·	2	2	·	·	2	·	2	2	·	·	2	2	·	2	·	·	2	2	·	·	2
1b	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·
1c	·	4	·	4	·	·	·	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·	·	·	·	4	·	4	·	·	·	·
1d	·	·	·	4	·	4	·	·	4	·	·	·	·	·	4	·	4	·	·	·	·	·	4	·	·	·	·	4	·	4	·	·
1e	·	·	·	·	·	·	·	·	2	2	2	2	2	2	2	2	·	·	·	·	·	·	·	·	2	2	2	2	2	2	2	2
1f	·	·	4	4	4	4	·	·	·	·	·	·	·	·	·	·	·	·	4	4	4	4	·	·	·	·	·	·	·	·	·	·

Other published properties include a differential-linear attack on up to 5 rounds of Ascon's initialization with practical complexity [11,34] and truncated differential distinguishers based on undisturbed bits for up to 5 rounds with 2^{109} data [89].

Algebraic Properties Ascon's algebraic degree of 2 for each round is useful for efficient secure implementations, but requires a sufficient number of rounds to prevent algebraic attacks. The best known algebraic attacks cover 7 out of 12 rounds of Ascon's initialization.

Algebraic Normal Form (ANF) Let $x_{0,i}, \ldots, x_{4,i}$ denote the bits in column i, $0 \leq i < 64$, where $x_{0,0}$ is the least significant (rightmost) bit of the first register word (outer part) of the state. Let $y_{0,i}, \ldots, y_{4,i}$ denote the same bit position after application of either the

Table 11. continued.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
(b) Linear approximation table: $\mathrm{LAT}[\alpha, \beta] = \lvert\{x : \alpha^\top \cdot x = \beta^\top \cdot \mathcal{S}(x)\}\rvert - 16$																																
0	16	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
1	·	·	·	·	·	·	8	·	·	4	4	·	·	-4	4	·	·	·	4	4	·	·	4	-4	4	·	-4	·	-4	·	-4	·
2	·	·	·	·	·	·	-8	8	·	·	4	4	·	·	4	4	·	·	4	4	·	·	-4	-4	·	·	·	·	·	·	·	·
3	·	8	·	·	·	·	·	·	·	4	·	4	·	4	·	-4	-8	·	·	·	·	·	·	·	4	·	4	·	4	·	-4	·
4	·	·	·	4	·	-4	·	·	·	·	4	·	·	4	-4	-4	·	·	4	·	-4	·	·	·	·	-8	·	-4	-4	·	4	-4
5	·	·	·	4	·	4	·	·	·	-4	·	·	·	·	·	-4	·	·	·	-4	4	·	-4	-4	4	·	-4	4	·	-8	·	-4
6	·	·	·	4	·	-4	·	·	·	·	·	-4	·	4	·	·	·	·	·	-4	-4	·	-4	-4	·	8	·	-4	-4	·	-4	4
7	·	·	·	-4	·	-4	·	·	·	4	4	4	·	·	-4	·	·	·	-4	·	-4	·	·	·	-4	·	-4	4	·	-8	·	4
8	·	·	·	·	·	·	·	·	·	·	4	4	·	·	-4	-4	·	·	·	·	·	·	·	·	·	8	-4	4	·	8	4	-4
9	·	·	·	·	·	·	·	-8	·	-4	·	4	·	4	·	4	·	·	4	4	·	·	-4	4	4	·	·	4	-4	·	·	4
a	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	4	4	·	·	4	4	·	8	4	-4	·	-8	4	-4
b	·	8	·	·	·	·	·	·	·	-4	4	·	·	-4	-4	·	8	·	·	·	·	·	·	·	4	·	·	-4	4	·	·	4
c	·	·	-8	4	-8	-4	·	·	·	·	·	4	·	-4	·	·	·	·	-4	·	4	·	·	·	·	·	4	·	-4	·	·	·
d	·	·	·	-4	-8	4	·	·	·	4	-4	-4	·	·	-4	·	·	·	·	4	-4	·	-4	-4	4	·	·	·	·	·	4	·
e	·	·	·	-4	8	-4	·	·	·	·	-4	·	·	-4	-4	-4	·	·	·	4	4	·	-4	-4	·	·	4	·	-4	·	·	·
f	·	·	8	-4	-8	-4	·	·	·	-4	·	·	·	·	·	-4	·	·	4	·	4	·	·	·	-4	·	·	·	·	·	-4	·
10	·	·	·	·	·	·	-8	·	·	4	·	-4	-4	·	-4	·	·	·	·	·	4	-4	4	4	4	·	-4	·	-4	·	-4	·
11	·	·	·	·	·	·	·	·	-8	·	-4	4	-4	-4	·	·	·	8	4	-4	-4	-4	·	·	·	·	·	·	·	·	·	·
12	·	-8	·	·	·	·	·	·	·	-4	4	·	-4	·	·	-4	·	·	-4	4	-4	-4	·	·	4	·	4	·	4	·	-4	·
13	·	·	·	·	·	·	-8	-8	·	·	·	·	4	-4	4	-4	·	·	·	·	-4	4	4	-4	·	·	·	·	·	·	·	·
14	·	·	·	4	·	4	·	·	·	4	4	-4	-4	-4	·	-4	·	·	4	·	·	4	-4	4	-4	·	4	4	·	·	·	4
15	·	·	·	4	·	-4	·	·	·	·	·	-4	4	·	-4	4	·	8	·	4	·	4	·	·	·	·	·	4	4	·	-4	-4
16	·	·	·	-4	·	-4	·	·	·	4	·	·	-4	4	4	·	8	·	·	-4	·	4	·	·	4	·	4	4	·	·	·	-4
17	·	·	·	4	·	-4	·	·	8	·	-4	·	-4	·	·	·	·	·	4	·	·	-4	4	-4	·	·	·	4	4	·	4	4
18	·	·	·	·	·	·	·	-8	·	4	4	·	-4	·	·	4	·	·	·	·	4	-4	-4	-4	-4	·	·	-4	4	·	·	-4
19	·	·	·	·	·	·	·	·	·	·	·	·	4	-4	-4	4	·	-8	4	-4	-4	-4	·	·	·	·	4	4	·	·	-4	-4
1a	·	8	·	·	·	·	·	·	·	-4	·	-4	-4	·	4	·	·	·	-4	4	-4	-4	·	·	-4	·	·	4	-4	·	·	-4
1b	·	·	·	·	·	·	·	·	8	·	-4	4	-4	-4	·	·	·	·	·	·	-4	4	-4	4	·	·	-4	-4	·	·	-4	-4
1c	·	·	8	4	·	-4	·	·	·	4	·	·	4	-4	4	·	·	·	-4	·	·	-4	-4	4	4	·	·	·	·	·	4	·
1d	·	·	·	-4	·	4	·	·	8	·	4	·	4	·	·	·	·	8	·	-4	·	-4	·	·	·	·	4	·	-4	·	·	·
1e	·	·	·	4	·	4	·	·	·	4	-4	4	4	4	·	-4	8	·	·	4	·	-4	·	·	-4	·	·	·	·	·	-4	·
1f	·	·	8	4	·	4	·	·	·	·	·	4	-4	·	-4	4	·	·	-4	·	·	4	4	-4	·	·	4	·	-4	·	·	·

Table 12. Minimum number of active S-boxes in R-round differential and linear characteristics for p^R.

Rounds R	1	2	3	4	5
Minimum # of differentially active S-boxes	1	4	15	≤ 44	≤ 78
Minimum # of linearly active S-boxes	1	4	13	≤ 43	≤ 67

Results for $R \geq 4$ are from heuristic search

S-box layer p_S or the linear layer p_L. The ANF of the S-box layer p_S is given by:

$$\begin{aligned}
y_{0,i} &= x_{4,i}\, x_{1,i} \oplus x_{3,i} \oplus x_{2,i}\, x_{1,i} \oplus x_{2,i} \oplus x_{1,i}\, x_{0,i} \oplus x_{1,i} \oplus x_{0,i}, \\
y_{1,i} &= x_{4,i} \oplus x_{3,i}\, x_{2,i} \oplus x_{3,i}\, x_{1,i} \oplus x_{3,i} \oplus x_{2,i}\, x_{1,i} \oplus x_{2,i} \oplus x_{1,i} \oplus x_{0,i}, \\
y_{2,i} &= x_{4,i}\, x_{3,i} \oplus x_{4,i} \oplus x_{2,i} \oplus x_{1,i} \oplus 1, \\
y_{3,i} &= x_{4,i}\, x_{0,i} \oplus x_{4,i} \oplus x_{3,i}\, x_{0,i} \oplus x_{3,i} \oplus x_{2,i} \oplus x_{1,i} \oplus x_{0,i}, \\
y_{4,i} &= x_{4,i}\, x_{1,i} \oplus x_{4,i} \oplus x_{3,i} \oplus x_{1,i}\, x_{0,i} \oplus x_{1,i}.
\end{aligned} \tag{1}$$

Table 13. Best known differential and linear characteristics for 4 and 5 rounds of p, given in truncated notation with the pattern of active S-boxes $\mathcal{S}$ in each round r and the corresponding probability or bias [31,34].

r	Active S-boxes	$\#\mathcal{S}$	$\log_2(p)$	$\log_2(\varepsilon)$
(a) Differential 4-round characteristic				
0	`441326c0236cca84`	23	– 47	
1	`8040000000040000`	3	– 12	
2	`0000100004040000`	3	– 6	
3	`c0489800262500a0`	15	– 42	
$\sum$		44	– 107	
(b) Linear 4-round characteristic				
0	`8181224200028685`	15		– 19
1	`0100004000000001`	3		– 4
2	`0000000010080001`	3		– 7
3	`04314f4725f80001`	22		– 23
$\sum$		43		– 50
(c) Differential 5-round characteristic				
0	`c01d986058edb14f`	29	– 58	
1	`0040800000000001`	3	– 12	
2	`0000100040000001`	3	– 9	
3	`022030304201080d`	13	– 30	
4	`732533f46a0d0a2d`	30	– 84	
$\sum$		78	– 193	
(d) Linear 5-round characteristic				
0	`8181224200028685`	15		– 19
1	`0100004000000001`	3		– 4
2	`0000000010080001`	3		– 7
3	`04314f4725f80001`	22		– 43
4	`04364206f5a80802`	24		– 25
$\sum$		67		– 94

The ANF of the linear layer p_L is as follows, with index computations mod 64:

$$\begin{aligned} y_{0,i} &= x_{0,i} \oplus x_{0,i+19} \oplus x_{0,i+28} \\ y_{1,i} &= x_{1,i} \oplus x_{1,i+61} \oplus x_{1,i+39} \\ y_{2,i} &= x_{2,i} \oplus x_{2,i+1} \oplus x_{2,i+6} \\ y_{3,i} &= x_{3,i} \oplus x_{3,i+10} \oplus x_{3,i+17} \\ y_{4,i} &= x_{4,i} \oplus x_{4,i+7} \oplus x_{4,i+41}. \end{aligned} \tag{2}$$

Algebraic Degree The algebraic degree of the round function p is 2, so the degree after R rounds is upper-bounded by 2^R. A tighter bound based on the general bounds by Canteaut [9, Theorem 1 with $\ell = 192$ for both $\mathcal{S}$ and $\mathcal{S}^{-1}$] and Boura et al. [10, Theorem 2 with $\gamma = 3$ for both $\mathcal{S}$ and $\mathcal{S}^{-1}$] is listed in Table 16.
Diffusion Table 17 provides an overview of the diffusion properties of up to 3 rounds of Ascon's permutation. After 3 rounds, almost all input bits appear in the ANF of each output bit (Table 17a). Finally, we list the maximum monomial degree for each input bit $x_{w,0}$ in the ANF after 1 round (Table 17c) and after 2 rounds (Table 17d).

Table 14. Differential characteristics for forgeries in ASCON .

r	Active S-boxes	$\# \mathcal{S}$
(a) Collision-producing differential for ASCON-128*'s 6-round state update*		
0	`8000000000000000`	1
1	`8100000001400004`	5
2	`9902a00003c64086`	17
3	`fcf7eee14feefdf7`	48
4	`dba6fe7b4fef8cef`	45
5	`0000400000000000`	1
$\sum$		117
(b) Collision-producing differential for ASCON-128a*'s 8-round state update*		
0	`8000000000000000`	1
1	`c200000000000000`	3
2	`e238e10000000000`	11
3	`73b7fbf67f6f19f0`	44
4	`bb4ffe8fd5dddf7f`	48
5	`fffffdffffffffff`	63
6	`2d0486c240902436`	20
7	`2080000000000000`	2
$\sum$		192
(c) Truncated differential for 4/12 rounds of ASCON*'s finalization,* $p = 2^{-101}$ [34]		
0	`8000000000000000`	1
1	`8000100801000004`	5
2	`d302904803844086`	18
3	`fbbff36d73e4f045`	41

Table 15. Impossible differential and zero-correlation linear hull for 5 rounds of p, shown using bitwise input and output differences and masks.

(a) Impossible differential (5 rounds)	Input difference		Output difference
x_0:	`0000000000000000`		`0100000000100002`
x_1:	`0000000000000000`		`0000000000000000`
x_2:	`0000000000000000`	$\rightarrow$	`0000000000000000`
x_3:	`0000000000000000`		`0000000000000000`
x_4:	`8000000000000000`		`0000000000000000`

(b) Zero-correlation linear hull (5 rounds)	Input mask		Output mask
	`8000000000000000`		`0000000000000000`
	`0000000000000000`		`0000000000000000`
	`0000000000000000`	$\rightarrow$	`fe08629e8e4b766a`
	`0000000000000000`		`0000000000000000`
	`0000000000000000`		`0000000000000000`

Table 16. Upper bound for the algebraic degree after R rounds [9, Theorem 1], [10, Theorem 2] (not including effects of initial structures).

Rounds R	1	2	3	4	5	6	7	8	9	10	11	12
$\deg(p^R) \leq$	2	4	8	16	32	64	128	256	298	312	317	319
$\deg(p^{-R}) \leq$	3	9	27	81	209	283	307	314	318	319		

Table 17. Diffusion statistics of the Ascon permutation after round r.

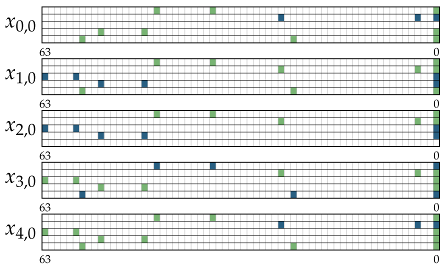
(a) # Variables in ANF of $x_{w,i}$

r	0		1		2	
	p_S	p_L	p_S	p_L	p_S	p_L
$x_{0,i}$	5	15	51	125	219	313
$x_{1,i}$	5	15	51	115	219	308
$x_{2,i}$	4	12	41	107	218	316
$x_{3,i}$	5	15	51	130	219	305
$x_{4,i}$	4	12	43	107	193	306

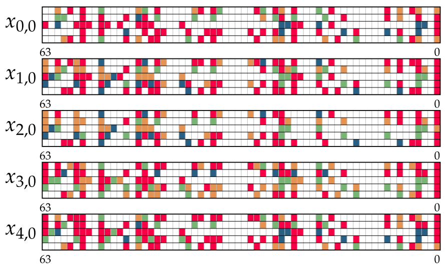
(b) # Monomials in ANF of $x_{w,i}$

0		1		2
p_S	p_L	p_S	p_L	p_S
7	21	746–898	2459–2614	7504022–8329829
7–8	20–24	670–763	1999–2181	5833573–6407756
4–5	12–13	315–327	931–967	3244871–3575653
7–8	21–22	600–666	1851–1964	7594245–8300027
5–5	15–15	585–693	1890–2045	5957375–6660105

(c) Diffusion of input bit $x_{w,0}$ after 1 round (🟦 deg 1, 🟩 deg 2)



(d) Diffusion of input bit $x_{w,0}$ after 2 rounds (🟦 deg 1, 🟩 deg 2, 🟧 deg 3, 🟥 deg 4)



Linearization and Initial Structures Distinguishers based on the degree can be combined with different initial structures that linearize the first few rounds in order to create a vector space or linear intermediate ANF with respect to the selected input variables. Besides generic structures (e.g., 0, 1, or 5 cube variables at each S-box input), several structures using the specific properties of Ascon's S-box have been proposed [34, 67]. For example, input bits $x_{2,i}$ are multiplied with neither $x_{0,i}$ nor $x_{4,i}$ in the first round.

By imposing bit conditions on certain input bits (corresponding to the key in Ascon), it is possible to find sufficiently large cubes such that no cube variables multiply after 1 round and one specific cube variable does not multiply with any others after 2 rounds [67]. An alternative approach suggested by Li et al. [67] does allow quadratic monomials after 1 round, but ensures that they are not multiplied with any other monomials after 2 rounds.

Zero-Sum and Cube Attacks The low degree of the S-box permits inside-out zero-sum distinguishers on the permutations p^a and p^b, so they can not be considered as perfect

Table 18. Division property results.

(a) S-box property [58]

k	0	1	2	3	4	5
D_k^5	0	1	1	2	2	5

(b) Integral distinguishers for the permutation [91]

Rounds R	5	6	7	8	9	10	11
$\log_2$(data)	18	35	65	130	258	300	315

random permutations. The full 12-round permutation can be distinguished with 2^{130} data [34] (4 inverse rounds, free middle round, and 7 forward rounds, see Table 16), or 11 rounds with complexity 2^{85} below the security bound ($4 + 1 + 6$ rounds, with the data complexity a multiple of the S-box size 5 for the free inner round). However, we are not aware of attacks able to exploit these properties for attacks on the authenticated cipher or hash function.

For key-recovery attacks in a nonce-respecting setting, cube variables can be positioned in the nonce and cube-like attacks can exploit that the cube sum after the round-reduced initialization only depends on selected key bits [34]. Using conditional initial structures of dimension 65 that ensure degree 2 after 2 rounds and thus degree at most 64 after 7 rounds, Li et al. [67] propose conditional cube attacks on 7 of 12 rounds of Ascon's initialization.

In a similar spirit to initial structures, it is also possible to linearize a few rounds of Ascon's permutation in order to find preimages for heavily round-reduced versions of Ascon-Xof as shown in [37]. Apart from that, an upper bound on the degree of the round-reduced Ascon permutation can be used to marginally speed-up a brute-force search for preimages as suggested by Bernstein [21]. For instance, it is possible to find a preimage for a version of Ascon-Xof where the number of rounds is reduced to 6 out of 12 and the output is truncated to 64 bits with a complexity of $2^{63.3}$ [37].

Other Properties Besides linear, differential, and algebraic properties, other relevant cryptographic properties exist. We discuss some of them next.

Integral Distinguishers and Division Property Based on the division property, Todo [91] proposes integral distinguishers for the Ascon permutation, where up to 7 rounds can be evaluated using less than 2^{128} data (Table 18b).

Göloğlu et al. [58] list the division properties of Ascon's S-box $\mathcal{S}$ and conclude that these values are optimal with respect to the degree (Table 18a).

Subspace Trails Leander et al. [69] analyze the existence of subspace trails. For Ascon's permutation, they show that the longest subspace trails using 1-linear structures cover 3 rounds (dimension 298) or 1 inverse round (dimension 125).

SAT Solvers Dwivedi et al. [42] use SAT solvers for a state recovery attack on 2 (out of 8) rounds of the data processing phase of Ascon-128a.

6.4. *List of Published Analysis*

As the primary recommendation for lightweight authenticated encryption in the final portfolio of the CAESAR competition [87], Ascon has received a lot of attention and several results regarding its security have been published. All the results published so far support the security of Ascon and its underlying permutation. In other words, no security vulnerabilities have been shown so far and the best attacks target the initialization of Ascon reduced to 7 (out of 12) rounds, concluding that Ascon has a security margin of 5 rounds (42% of the 12 rounds).

The following list contains both results evaluating the permutation and evaluation of the security of Ascon's authenticated encryption, either using variants of Ascon's permutation, or idealized versions of it.

- Detailed analysis of Ascon's differential-linear properties [11].
- Improved 4-round differential-linear analysis and subspace trails [90].
- Integral distinguishers for the round-reduced inverse Ascon [94].
- Subspace trails for a small number of rounds for Ascon's permutation [69].
- Evaluation of the security of Ascon in misuse settings [92].
- Cube-like key-recovery attack on 7 (out of 12) rounds of the initialization of Ascon in $2^{103.9}$ time [67].
- Cube-like attacks in a nonce-misuse setting on round-reduced Ascon [70].
- SAT-based state recovery on 2 (out of 8) rounds of the data processing of Ascon-128a [42].
- Evaluating the properties of Ascon's authenticated encyption mode regarding re-forgeability [48].
- Truncated, impossible, and improbable differential distinguishers for 4 and 5 rounds of Ascon's permutation. Differential distinguishers based on undisturbed bits for to 5 rounds reduced variants of Ascon with 2^{109} data [89].
- Security of Ascon's S-box with respect to the division property [58].
- Several linear characteristic for Ascon's permutation [31].
- Ascon's authenticated encryption mode supports secure implementations on limited-memory devices [2].
- Evaluation of Ascon's permutation using the division property [91].
- Suggestions to absorb authenticated data more efficiently [86].
- Evaluation of the resistance of Ascon's permutation against algebraic, differential, linear, and differential-linear attacks. Cube-like and differential-linear key recovery attacks on round-reduced variants of Ascon. Differential-based forgery attacks on round-reduced Ascon [34].
- Security proof for Ascon's authenticated encryption mode even for higher rates [62].
- Security analysis and bounds for the full-state keyed duplex with application to Ascon-128 and Ascon-128a [43].
- Discussing security and leakage-resilience of Ascon's mode of operation [55].

7. Implementation

Since Ascon is based on the sponge and duplex constructions, it just relies on the evaluation of cryptographic permutations in forward direction to allow hashing and authenticated encryption. In particular, there is no need to implement the inverse of the

Table 19. Ascon-128 and Ascon-128a software performance in cycles per byte.

Message length	1	8	16	32	64	1536	Long
(a) Ascon-128							
AMD Ryzen 7 1700[a]					14.5	8.8	8.6
Intel Xeon E5-2609 v4[a]					17.3	10.8	10.5
Cortex-A53 (ARMv8)[a]					18.3	11.3	11.0
Intel Core i5-6300U	367	58	35	23	17.6	11.9	11.4
Intel Core i5-4200U	521	81	49	32	23.9	16.2	15.8
Cortex-A15 (ARMv7)[a]					69.8	36.2	34.6
Cortex-A7 (NEON)	2705	250	150	99	73.2	48.8	47.9
Cortex-A7 (ARMv7)	1871	292	175	115	86.6	58.3	57.2
ARM1176JZF-S (ARMv6)	2189	340	202	133	97.9	64.4	65.3
(b) Ascon-128a							
AMD Ryzen 7 1700[a]					12.0	6.0	5.7
Intel Xeon E5-2609 v4[a]					14.1	7.3	6.9
Cortex-A57 (ARMv8)[a]					15.1	7.6	7.3
Intel Core i5-6300U	365	47	31	19	13.5	8.0	7.8
Intel Core i5-4200U	519	67	44	27	18.8	11.0	10.6
Cortex-A15 (ARMv7)[a]					60.3	25.3	23.8
Cortex-A7 (NEON)	2805	274	133	83	57.6	33.5	32.6
Cortex-A7 (ARMv7)	1911	255	161	102	71.3	42.3	41.2
ARM1176JZF-S (ARMv6)	2267	303	191	120	84.4	50.0	50.2

Message length is length of encrypted plaintext with empty associated data
[a]Results taken from eBACS [45]

permutation, or other often used components in authenticated encryption schemes like a key schedule, masks, Galois field multiplications, etc. This together with the small state size of 320 bits minimizes the code size and register pressure in software and the area requirements in hardware. Still, the state size of 320 bits is large enough to provide both hashing and authenticated encryption with 128 bits of security.

Software Implementations A preliminary overview of the software performance of Ascon-128 and Ascon-128a is given in Table 19a, b. Detailed software performance results and comparisons for a large number of platforms are given in eBACS, the ECRYPT Benchmarking of Cryptographic Systems [45]. Additional software benchmarking results for micro controllers (including size and runtime) of lightweight authenticated encryption schemes submitted to NIST are maintained at [76]. The software performance of Ascon-Hash and Ascon-Xof is largely the same as for Ascon-128 with doubled cycles per byte.

Hardware Implementations Detailed hardware performance results and comparisons for a large number of implementations are given in the Athena project [5]. A preliminary overview of the hardware performance of Ascon-128 and Ascon-128a for different use cases is given in Table 20a, b. Note that the CAESAR API implies a certain overhead, in particular for lightweight designs like Ascon. However, this cost can be significantly reduced by using a dedicated lightweight interface as shown in [59].

Table 20. Hardware reference implementations of Ascon-128/ Ascon-128a using the CAESAR Hardware API [56,61]. Excluding area for Pre-Processor (869/1491 GE), Post-Processor (1032/1344 GE), HDR Buffer (836 GE).

Design	Area [kGE]	Throughput [Mbps]
(a) Ascon-128		
1 round	9.42	4888
2 rounds	12.99	8482
3 rounds	16.59	10343
6 rounds	27.28	12261
(b) Ascon-128a		
1 round	9.68	7326
2 rounds	13.25	11743
4 rounds	20.38	16675

7.1. *Efficiency for Short Messages*

The simplicity of the design and the small state play also a crucial role in the efficiency of Ascon's authenticated encryption for small messages. For instance, if no associated data is present, Ascon-128 can encrypt plaintexts strictly smaller than 8 bytes and Ascon-128a can encrypt plaintexts strictly smaller than 16 bytes with just two calls to the permutation p^a. Preliminary software performance results for several short messages and platforms are also shown in Table 19a, b.

Ankele and Ankele [1] give a detailed performance overview of the second round CAESAR candidates for short messages. In many scenarious (e.g., SSH with 5 bytes of associated data and 1 byte of plaintext), Ascon-128a is able to perform very well, even when compared to AES-based designs which use native AES instructions on Intel Skylake processors [1, Figure 6].

7.2. *Flexibility of the Permutation*

The permutation of Ascon is naturally defined on 64-bit words using only bitwise Boolean functions (and, not, xor) and rotations within these 64-bit words. As a consequence, Ascon does not require any data-dependent table lookups. Hence, it lends itself to bitsliced implementations in software as well as simple and clean hardware implementations.

Instruction Parallelism The S-box and the linear layer provide some flexibility regarding the number of instructions that can be carried out in parallel and additional temporary registers that are needed to store intermediate computations in software implementations. A bitsliced implementation of the S-box that focuses on instruction parallelism is shown in Fig. 5. Considering that the linear layer is defined separately on each of the 5 64-bit words, up to 5 instructions can be carried out in parallel in nearly every phase of the permutation. This implementation aspect of Ascon allows for short critical paths in hardware and makes use of the out-of-order execution capabilities of high-end processors.

```
x0 ^= x4;    x4 ^= x3;    x2 ^= x1;
t0  = x0;    t1  = x1;    t2  = x2;    t3  = x3;    t4  = x4;
t0 = ~t0;    t1 = ~t1;    t2 = ~t2;    t3 = ~t3;    t4 = ~t4;
t0 &= x1;    t1 &= x2;    t2 &= x3;    t3 &= x4;    t4 &= x0;
x0 ^= t1;    x1 ^= t2;    x2 ^= t3;    x3 ^= t4;    x4 ^= t0;
x1 ^= x0;    x0 ^= x4;    x3 ^= x2;    x2 = ~x2;
```

Fig. 5. Pipelinable instructions for bitsliced implementation of 5-bit S-box $\mathcal{S}(x)$.

```
x0 ^= x4;             x4 ^= x3;             x2 ^= x1;
t0  = (~x4) & x0;
t1  = (~x1) & x2;     x0 ^= t1;
t1  = (~x3) & x4;     x2 ^= t1;
t1  = (~x0) & x1;     x4 ^= t1;
t1  = (~x2) & x3;     x1 ^= t1;
x3 ^= t0;
x1 ^= x0;             x3 ^= x2;             x0 ^= x4;    x2 = ~x2;
```

Fig. 6. Reducing register pressure for bitsliced implementation of the 5-bit S-box $\mathcal{S}(x)$.

Bit Interleaving However, Ascon can also be implemented on systems that do not have a natural 64-bit datapath, like 8-, 16-, and 32-bit processors. This can be done by employing a technique called bit interleaving as described in the Keccak implementation overview [13]. By using this technique, the single bits of one of Ascon's 64-bit words are stored interleaved in two 32-bit, four 16-bit, or eight 8-bit registers. This technique allows to translate rotations within the 64-bit words to rotations (and re-labeling) of the smaller registers. Since the other operations of Ascon are bitwise Boolean functions, they are unaffected by the changed representations.

That bit interleaving is a very viable strategy can be seen in the work of Bangma [6], where the performance of Ascon-128 is compared with implementations of the CAESAR finalists ACORN, AEGIS-128L, Deoxys-II-128, and MORUS-1280-128 on an ARM Cortex-A8. Here, Ascon-128 is the fastest cipher for short plaintexts of 64 bytes [6, Table 5.1].

Reducing Register Pressure To reduce register pressure on resource constrained devices, the S-box of Ascon can also be implemented using just two temporary registers as shown in Fig. 6. This low register implementation was inspired by [26,41] and later extended to protect the masked Ascon S-box against SIFA in [30]. In particular, masked implementations benefit from this low register usage since the number of registers increase linearly with the masking order. Additionally, the resulting implementation of the S-box is uniform (a requirement for threshold implementations [75]) even without the need for additional online random input data [30].

7.3. *Further Reading on Efficiency*

Benchmarking Efforts Several teams are working on benchmarking efforts where updates are expected throughout the NIST lightweight cryptography project. Similarly, the

CAESAR competition has inspired several such efforts. Benchmarking initiatives with online resources include the following:

- Athena: Hardware evaluation and CAESAR HW API [5,61,88,93] https://cryptography.gmu.edu/athena/index.php?id=LWC
- eBACS: ECRYPT Benchmarking of Cryptographic Systems [45] http://bench.cr.yp.to
- LaS^3: NIST LWC software performance benchmarks on microcontrollers [81] https://lwc.las3.de
- Rhys Weatherly: Software benchmarking for 32-bit embedded architectures https://rweather.github.io/lightweight-crypto/index.html
- Ralph&Robin Ankele: Software benchmarking of 2nd round CAESAR candidates [1] https://github.com/TheBananaMan/caesar_benchmarks_secondround
- BRUTUS: Testing CAESAR authenticated encryption candidates for weaknesses [82] https://github.com/mjosaarinen/brutus

Hardware implementations are reported by [40,46,47,63,80,84,88,93].
Software implementations are reported by [6,38,39,66,85].

7.4. *Implementation Security and Robustness*

Ascon's permutation uses S-boxes of degree 2 and thus lends itself to efficient countermeasures against side-channel attacks by masking with a low overhead. The low-register implementation of the Ascon S-box given in Fig. 6 can be extended to any masking order as shown in [30]. Using this approach, only two additional temporary registers are needed for each share and no additional randomness is needed to get a uniform implementation of the Ascon S-box. As a result, masked software implementations of Ascon result in a performance penalty with low overhead.

Additionally, leveled implementations of Ascon are possible to improve the robustness and/or security of the design [4,7]. Examples are side-channel protected implementations with different strength of the key XORs, p^a and/or p^b. Also, the plaintext leakages can be reduced by limiting the number of decryption failures.

Many protected hardware implementations of Ascon have been published already. Gross et al. [59] provide threshold implementations of Ascon-128 as small as 7.97 kGE. Additionally, many other state-of-the-art masking approaches have been applied on Ascon, like UMA [51] and DOM [53], even for high protection order (see Table 21). Links to various implementations of Ascon, including DOM and UMA implementations, can be found on the Ascon website[1].

Next, we give a list of papers that either evaluate the side-channel and fault resistance of Ascon or elaborate protection mechanisms against side-channel and fault attacks [4,7,12,28–30,50,51,55,60,68,77–79,83].

[1] https://ascon.iaik.tugraz.at/.

Table 21. DOM implementations for various protection orders [51,52].

Protection Order	Pipelined		Parallel	
	[kGE]	[Mbps]	[kGE]	[Mbps]
1	10.86	108	28.89	2246
2	16.19	108	53.00	1896
3	21.59	110	81.21	1903
4	27.13	71	118.27	1786
5	32.76	95	161.87	1868
…				
13	81.20	70	726.00	1833
14	87.75	71	828.19	1439
15	94.24	50	926.34	1480

8. Conclusion

Ascon-128 and Ascon-128a have been shown to be robust authenticated encryption schemes with a comfortable security margin that can be implemented efficiently on a wide range of platforms including hardware and software. In addition, their design allows for the addition of countermesures against implementation attacks, most notably side-channel attacks, at a rather low overhead.

Based on the well-analyzed permutation underlying Ascon-128 and Ascon-128a, we have specified Ascon-Hash and Ascon-Xof to enhance the functionality of the Ascon cipher suite. Thus, implementing Ascon's permutation allows to realize authenticated encryption and hashing without the need of additional cryptographic primitives. The whole package has been submitted to the NIST lightweight cryptography standardization process.

Acknowledgements

The authors would like to thank all researchers contributing to the design, analysis and implementation of Ascon. In particular, we want to thank Hannes Gross for all his support and various implementations of Ascon. Part of this work has been supported by the Austrian Science Fund (FWF): P26494-N15 and J 4277-N38, by the European Union's Horizon 2020 research and innovation programme (H2020 ICT 644052: HECTOR) and the European Research Council (ERC 681402: SOPHIA), and by the Austrian Government (FFG/SFG COMET 836628: SeCoS and FIT-IT 835919: SePAG).

Funding Open access funding provided by Graz University of Technology.

References

[1] R. Ankele, R. Ankele, Software benchmarking of the 2nd round CAESAR candidates (IACR, 2016). https://ia.cr/2016/740

[2] M. Agrawal, D. Chang, S. Sanadhya, sp-AELM: sponge based authenticated encryption scheme for memory constrained devices, in *ACISP 2015*. LNCS, vol. 9144 (Springer, 2015), pp. 451–468. https://doi.org/10.1007/978-3-319-19962-7-26

[3] E. Andreeva, J. Daemen, B. Mennink, G. Van Assche, Security of keyed sponge constructions using a modular proof approach, in *FSE 2015*. LNCS, vol. 9054 (Springer, 2015), pp. 364–384. https://doi.org/10.1007/978-3-662-48116-5-18

[4] A. Adomnicai, J.J.A. Fournier, L. Masson, Masking the lightweight authenticated ciphers ACORN and ascon in software (IACR, 2018). https://ia.cr/2018/708

[5] Athena project, Automated tool for hardware evaluation—CAESAR hardware API (2016). https://cryptography.gmu.edu/athena/index.php?id=CAESAR_source_codes

[6] N. Bangma, Ascon: an attempt in NEON on the Cortex-A8. Bachelor's Thesis (2018). https://www.cs.ru.nl/bachelorscripties/2018/Noel_Bangma___4433939___Ascon_An_attempt_in_NEON_on_the_Cortex-A8.pdf

[7] D. Bellizia, O. Bronchain, G. Cassiers, V. Grosso, C. Guo, C. Momin, O. Pereira, T. Peters, F.-X. Standaert, Mode-level vs. implementation-level physical security in symmetric cryptography—a practical guide through the leakage-resistance jungle, in *CRYPTO 2020*. LNCS, vol. 12170 (Springer, 2020), pp. 369–400. https://doi.org/10.1007/978-3-030-56784-2-13

[8] B. Bilgin, A. Bogdanov, M. Knezevic, F. Mendel, Q. Wang, Fides: lightweight authenticated cipher with side-channel resistance for constrained hardware, in *CHES 2013*. LNCS, vol. 8086 (Springer, 2013), pp. 142–158. https://doi.org/10.1007/978-3-642-40349-1-9. IACR https://ia.cr/2015/424

[9] C. Boura, A. Canteaut, A zero-sum property for the Keccak-f permutation with 18 Rounds, in *ISIT 2010* (IEEE, 2010), pp. 2488–2492. https://doi.org/10.1109/ISIT.2010.5513442

[10] C. Boura, A. Canteaut, C. De Cannière, Higher-order differential properties of Keccak and Luffa, in *FSE 2011*. LNCS, vol. 6733 (Springer, 2011), pp. 252–269. https://doi.org/10.1007/978-3-642-13858-4-15. IACR https://ia.cr/2010/589

[11] A. Bar-On, O. Dunkelman, N. Keller, A. Weizman, DLCT: a new tool for differential-linear cryptanalysis, in *EUROCRYPT 2019*. LNCS, vol. 11476 (Springer, 2019), pp. 313–342. https://doi.org/10.1007/978-3-030-17653-2-11. IACR https://ia.cr/2019/256

[12] S. Belaïd, P.-É. Dagand, D. Mercadier, M. Rivain, R. Wintersdorff, Tornado: automatic generation of probing-secure masked bitsliced implementations, in *EUROCRYPT 2020*. LNCS, vol. 12107 (Springer, 2020), pp. 311–341. https://doi.org/10.1007/978-3-030-45727-3-11

[13] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, R. Van Keer, Keccak implementation overview version 3.2 (2012). https://keccak.team

[14] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, Sponge functions, in *Ecrypt Hash Workshop 2007* (2007). https://keccak.team/files/SpongeFunctions.pdf

[15] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, On the Indifferentiability of the Sponge Construction, in *EUROCRYPT 2008*. LNCS, vol. 4965 (Springer, 2008), pp. 181–197. https://doi.org/10.1007/978-3-540-78967-3-11. https://keccak.team/files/SpongeIndifferentiability.pdf

[16] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, Sponge-based pseudo-random number generators, in *CHES 2010*. LNCS, vol. 6225 (Springer, 2010), pp. 33–47. https://doi.org/10.1007/978-3-642-15031-9-3. https://keccak.team/files/SpongePRNG.pdf

[17] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, Duplexing the sponge: single-pass authenticated encryption and other applications, in *SAC 2011*. LNCS, vol. 7118 (Springer, 2011), pp. 320–337. https://doi.org/10.1007/978-3-642-28496-0-19. IACR https://ia.cr/2011/499

[18] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, On the security of the keyed sponge construction, in *SKEW 2011* (2011). https://keccak.team/files/SpongeKeyed.pdf

[19] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, The Keccak reference. Submission to NIST (Round 3) (2011). https://keccak.team

[20] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, Permutation-based encryption, authentication and authenticated encryption, in *DIAC 2012*, July 2012 (2012)

[21] D. J. Bernstein, Second preimages for 6 (7 (8??)) rounds of Keccak? Posted on the NIST mailing list (2010). https://ehash.iaik.tugraz.at/uploads/6/65/NIST-mailing-list_Bernstein-Daemen.txt

[22] G. Brassard, P. Høyer, A. Tapp, Quantum cryptanalysis of hash and claw-free functions, in *LATIN '98*. LNCS, vol. 1380 (Springer, 1998), pp. 163–169. https://doi.org/10.1007/BFb0054319

[23] A. Bogdanov, V. Rijmen, Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Des. Codes Cryptogr.* **70**(3), 369–383 (2014). https://doi.org/10.1007/s10623-012-9697-z

[24] J. Czajkowski, L.G. Bruinderink, A. Hülsing, C. Schaffner, D. Unruh, Post-quantum security of the sponge construction, in *PQCrypto 2018*. LNCS, vol. 10786 (Springer, 2018), pp. 185–204. https://doi.org/10.1007/978-3-319-79063-3-9

[25] S. Chari, C. S. Jutla, J. R. Rao, P. Rohatgi, Towards sound approaches to counteract power-analysis attacks, in *CRYPTO '99*. LNCS, vol. 1666 (Springer, 1999), pp. 398–412. https://doi.org/10.1007/3-540-48405-1

[26] J. Daemen, Personal communication about implementing the 3-bit c layer. Dec. 2018 (2018)

[27] J. Daemen, Cipher and hash function design. Strategies based on linear and differential cryptanalysis. PhD thesis (Katholieke Universiteit Leuven, 1995). https://www.esat.kuleuven.be/cosic/publications/thesis-6.pdf

[28] L. De Meyer, Looking at the NIST lightweight candidates from a masking point-of-view. IACR Cryptology ePrint Archive, Report 2020/699 (2020). IACR https://ia.cr/2020/699

[29] W. Diehl, A. Abdulgadir, F. Farahmand, J.-P. Kaps, K. Gaj, Comparison of cost of protection against differential power analysis of selected authenticated ciphers, in *HOST 2018* (IEEE Computer Society, 2018), pp. 147–152. https://doi.org/10.1109/HST.2018.8383904

[30] J. Daemen, C. Dobraunig, M. Eichlseder, H. Gross, F. Mendel, R. Primas, Protecting against Statistical Ineffective Fault Attacks". In: IACR Transactions on Cryptographic Hardware and Embedded Systems 2020.3 (2020), pp. 508–543. https://doi.org/10.13154/tches.v2020.i3.508-543. IACR https://ia.cr/2019/536

[31] C. Dobraunig, M. Eichlseder, F. Mendel, Heuristic tool for linear cryptanalysis with applications to CAESAR candidates, in *ASIACRYPT 2015*. LNCS, vol. 9453 (Springer, 2015), pp. 490–509. https://doi.org/10.1007/978-3-662-48800-3-20. IACR https://ia.cr/2015/1200

[32] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer, Ascon v1. Submission to the CAESAR competition (2014). https://ascon.iaik.tugraz.at

[33] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer, Ascon v1.1. Submission to Round 2 of the CAESAR competition (2015). https://ascon.iaik.tugraz.at

[34] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer, Cryptanalysis of Ascon, in *CT-RSA 2015*. LNCS, vol. 9048 (Springer, 2015), pp. 371–387. https://doi.org/10.1007/978-3-319-16715-2-20. IACR https://ia.cr/2015/030

[35] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer, Ascon v1.2. Submission to Round 3 of the CAESAR competition (2016). https://ascon.iaik.tugraz.at

[36] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer, Ascon v1.2. Submission to NIST (2019). https://ascon.iaik.tugraz.at

[37] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer, Preliminary analysis of Ascon-Xof and Ascon-Hash. Technical Report (2019). https://ascon.iaik.tugraz.at

[38] L.C. dos Santos, Software implementation of authenticated encryptionalgorithms on ARM processors. Master's Thesis (2018). http://repositorio.unicamp.br/jspui/bitstream/REPOSIP/332624/1/Santos_LuanCardosoDos_M.pdf

[39] L.C. dos Santos, J. Großschädl, A. Biryukov, FELICSAEAD: benchmarking of lightweight authenticated encryption algorithms, in *CARDIS 2019*. LNCS, vol. 11833 (Springer, 2019), pp. 216–233. https://doi.org/10.1007/978-3-030-42068-0-13

[40] W. Diehl, K. Gaj, RTL implementations and FPGA benchmarking of selected CAESAR Round Two authenticated ciphers. *Microprocess. Microsyst.* **52**, 202–218 (2017). https://doi.org/10.1016/j.micpro.2017.06.003

[41] J. Daemen, S. Hoffert, G. Van Assche, R. Van Keer, The design of Xoodoo and Xoofff. *IACR Trans. Symmetric Cryptol.* **2018**(4), 1–38 (2018). https://doi.org/10.13154/tosc.v2018.i4. IACR https://ia.cr/2018/767

[42] A.D. Dwivedi, M. Kloucek, P. Morawiecki, I. Nikolic, J. Pieprzyk, S. Wójtowicz, SAT-based cryptanalysis of authenticated ciphers from the CAESAR competition, in *SECRYPT ICETE 2017* (SciTePress, 2017), pp. 237–246. https://doi.org/10.5220/0006387302370246. IACR https://ia.cr/2016/1053

[43] J. Daemen, B. Mennink, G. Van Assche, Full-state keyed duplex with built-in multi-user support. In: ASIACRYPT 2017. LNCS, vol. 10625 (Springer, 2017), pp. 606–637. https://doi.org/10.1007/978-3-319-70697-9-21. IACR https://ia.cr/2017/498

[44] C. De Cannière, H. Sato, D. Watanabe, Hash function luffa: specification. Submission to NIST (Round 2) (2009). https://www.hitachi.com/rd/yrl/crypto/luffa/

[45] eBACS: ECRYPT benchmarking of cryptographic systems. https://bench.cr.yp.to (visited on 02/14/2019)

[46] F. Farahmand, W. Diehl, A. Abdulgadir, J.-P. Kaps, K. Gaj, Improved lightweight implementations of CAESAR authenticated ciphers, in *FCCM 2018* (IEEE Computer Society, 2018), pp. 29–36. https://doi.org/10.1109/FCCM.2018.00014

[47] M. Fivez, Energy Efficient Hardware Implementations of CAESAR Submissions. Master's Thesis (2016). https://www.esat.kuleuven.be/cosic/publications/thesis-279.pdf

[48] C. Forler, E. List, S. Lucks, J. Wenzel, Reforgeability of authenticated encryption schemes, in *ACISP 2017*. LNCS, vol. 10343 (Springer, 2017), pp. 19–37. https://doi.org/10.1007/978-3-319-59870-3-2. IACR https://ia.cr/2017/332

[49] K. Gaj, ATHENa Team, ATHENa: automated tool for hardware evaluation (2016). https://cryptography.gmu.edu/athena/

[50] H. Gross, R. Iusupov, R. Bloem, Generic low-latency masking in hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**(2), 1–21 (2018). https://doi.org/10.13154/tches.v2018.i2.1-21. IACR https://ia.cr/2017/1223

[51] H. Gross, S. Mangard, Reconciling d+1 masking in hardware and software, in *CHES 2017*. LNCS, vol. 10529 (Springer, 2017), pp. 115–136. https://doi.org/10.1007/978-3-319-66787-4-6. IACR https://ia.cr/2017/103

[52] H. Gross, S. Mangard, A unified masking approach. *J. Cryptogr. Eng.* **8**(2), 109–124 (2018). https://doi.org/10.1007/s13389-018-0184-y

[53] H. Groß, S. Mangard, T. Korak, Domain-oriented masking: compact masked hardware implementations with arbitrary protection order, in *TIS@CCS 2016* (ACM, 2016), p. 3. https://doi.org/10.1145/2996366.2996426. IACR https://ia.cr/2016/486

[54] L. Goubin, J. Patarin, DES and differential power analysis (the "duplication" method), in *CHES '99*. LNCS, vol. 1717 (Springer, 1999), pp. 158–172. https://doi.org/10.1007/3-540-48059-5

[55] C. Guo, O. Pereira, T. Peters, F.-X. Standaert, Towards low-energy leakage-resistant authenticated encryption from the duplex sponge construction. *IACR Trans. Symmetric Cryptol.* **2020**(1), 6–42 (2020). https://doi.org/10.13154/tosc.v2020.i1. IACR https://ia.cr/2019/193

[56] H. Gross, Reference hardware implementations of Ascon-128 and Ascon-128a using the CAESAR Hardware API v1.0. (2016). https://github.com/IAIK/ascon_hardware/tree/master/caesar_hardware_api_v_1_0_3/ASCON_ASCON

[57] L. K. Grover, A fast quantum mechanical algorithm for database search, in *STOC '96* (ACM, 1996), pp. 212–219. https://doi.org/10.1145/237814.237866

[58] F. Göloglu, V. Rijmen, Q. Wang, On the division property of S-boxes. IACR Cryptology ePrint Archive, Report 2016/188 (2016). IACR https://ia.cr/2016/188

[59] H. Gross, E. Wenger, C. Dobraunig, C. Ehrenhöfer, Suit up!—made-to-measure hardware implementations of Ascon, in *DSD 2015* (IEEE Computer Society, 2015), pp. 645–652. https://doi.org/10.1109/DSD.2015.14. IACR https://ia.cr/2015/034

[60] H. Gross, E. Wenger, C. Dobraunig, C. Ehrenhöfer, Ascon hardware implementations and side-channel evaluation. *Microprocess. Microsyst.* **52**, 470–479 (2017). https://doi.org/10.1016/j.micpro.2016.10.006

[61] E. Homsirikamol, W. Diehl, A. Ferozpuri, F. Farahmand, P. Yalla, J.-P. Kaps, K. Gaj, CAESAR hardware API. IACR Cryptology ePrint Archive, Report 2016/626 (2016). IACR https://ia.cr/2016/626

[62] P. Jovanovic, A. Luykx, B. Mennink, Beyond $2^c/2$ security in sponge-based authenticated encryption modes, in *ASIACRYPT 2014*. LNCS, vol. 8873 (Springer, 2014), pp. 85–104. https://doi.org/10.1007/978-3-662-45611-8-5. IACR https://ia.cr/2014/373

[63] J.-P. Kaps, W. Diehl, M. Tempelmeier, F. Farahmand, E. Homsirikamol, K. Gaj, A comprehensive framework for fair and efficient benchmarking of hardware implementations of lightweight cryptography. IACR Cryptology ePrint Archive, Report 2019/1273. 2019. IACR https://ia.cr/2019/1273

[64] L. Knudsen, DEAL—a 128-bit block cipher (1998)

[65] Ö. Küçük, The hash function Hamsi. Submission to NIST (Round 2) (2009). https://www.esat.kuleuven.be/cosic/publications/article-1203.pdf

[66] R. Kumar Pal, Implementation and evaluation of authenticated encryption algorithms on Java Card Platform (master's thesis). Master's Thesis (2017). https://is.muni.cz/th/448415/fi_m/MSThesis_IS.pdf

[67] Z. Li, X. Dong, X. Wang, Conditional cube attack on round-reduced ASCON. *IACR Trans. Symmetric Cryptol.* **2017**(1), 175–202 (2017). https://doi.org/10.13154/tosc.v2017.i1.175-202. IACR: https://ia.cr/2017/160, https://github.com/lizhengcn/Ascon_test

[68] L. Lerman, O. Markowitch, N. Veshchikov, Comparing Sboxes of ciphers from the perspective of side-channel attacks, in *Asian-HOST 2016* (IEEE Computer Society, 2016), pp. 1–6. https://doi.org/10.1109/AsianHOST.2016.7835556. IACR https://ia.cr/2016/993

[69] G. Leander, C. Tezcan, F. Wiemer, Searching for subspace trails and truncated differentials. *IACR Trans. Symmetric Cryptol.* **2018**(1), 74–100 (2018). https://doi.org/10.13154/tosc.v2018.i1.74-100

[70] Y. Li, G. Zhang, W. Wang, M. Wang, Cryptanalysis of round-reduced ASCON. *Sci. China Inf. Sci.* **60**(3), 38102 (2017). https://doi.org/10.1007/s11432-016-0283-3

[71] National Institute of Standards and Technology, FIPS PUB 180-2: secure hash standard. Federal Information Processing Standards Publication 180-2, U.S. Department of Commerce (2002)

[72] National Institute of Standards and Technology, FIPS PUB 180-4: Secure hash standard. Federal Information Processing Standards Publication 180-4, U.S. Department of Commerce. 2015. https://doi.org/10.6028/NIST.FIPS.180-4

[73] National Institute of Standards and Technology, NIST SP 800-185: SHA-3 derived functions: cSHAKE, KMAC, TupleHash, ParallelHash. NIST Special Publication 800-185, U.S. Department of Commerce (2016). https://doi.org/10.6028/NIST.SP.800-185

[74] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the lightweight cryptography standardization process (2018). https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf

[75] S. Nikova, V. Rijmen, M. Schläffer, Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptol.* **24**(2), 292–321 (2011). https://doi.org/10.1007/s00145-010-9085-7

[76] NIST LWC Software Performance Benchmarks on Microcontrollers. https://lwc.las3.de/ (visited on 06/24/2020)

[77] K. Ramezanpour, P. Ampadu, W. Diehl, A statistical fault analysis methodology for the Ascon authenticated cipher, in *HOST 2019* (IEEE, 2019), pp. 41–50. https://doi.org/10.1109/HST.2019.8741029

[78] K. Ramezanpour, P. Ampadu, W. Diehl, FIMA: fault intensity map analysis, in *COSADE 2019*. LNCS, vol. 11421 (Springer, 2019), pp. 63–79. https://doi.org/10.1007/978-3-030-16350-1-5

[79] K. Ramezanpour, P. Ampadu, W. Diehl, SCARL: side-channel analysis with reinforcement learning on the Ascon authenticated cipher. CoRR. 2006.03995

[80] B. Rezvani, W. Diehl, Hardware implementations of NIST lightweight cryptographic candidates: a first look. IACR Cryptology ePrint Archive, Report 2019/824 (2019). IACR https://ia.cr/2019/824

[81] S. Renner, E. Pozzobon, J. Mottok, Benchmarking software implementations of 1st round candidates of the NIST LWC project on MCUs, in *NIST Lightweight Cryptography Workshop* (2019). https://csrc.nist.gov/Presentations/2019/benchmarking-software-implementations-of-1st-round

[82] M.-J.O. Saarinen, The BRUTUS automatic cryptanalytic framework—Testing CAESAR authenticated encryption candidates forweaknesses. *J. Cryptogr. Eng.* **6**(1), 75–82 (2016). https://doi.org/10.1007/s13389-015-0114-1

[83] N. Samwel, J. Daemen, DPAon hardware implementations of Ascon and Keyak, in *CF'17* (ACM, 2017), pp. 415–424. https://doi.org/10.1145/3075564.3079067

[84] N. Samir, A.S. Hussein, M. Khaled, A.N. El-Zeiny, M. Osama, H. Yassin, A. Abdelbaky, O. Mahmoud, A. Shawky, H. Mostafa, ASIC and FPGA comparative study for IoT lightweight hardware security algorithms. *J. Circuits Syst. Comput.* **28**(12), 1930009:1–1930009:25 (2019). https://doi.org/10.1142/S0218126619300095

[85] K. Stoffelen, Optimizing s-box implementations for several criteria using SAT solvers, in *FSE 2016*. LNCS, vol. 9783 (Springer, 2016), pp. 140–160. https://doi.org/10.1007/978-3-662-52993-5-8. IACR https://ia.cr/2016/198

[86] Y. Sasaki, K. Yasuda, How to incorporate associated data in sponge-based authenticated encryption, in *CT-RSA 2015*. LNCS, vol. 9048 (Springer, 2015), pp. 353–370. https://doi.org/10.1007/978-3-319-16715-2-19

[87] The CAESAR committee, CAESAR: competition for authenticated encryption: security, applicability, and robustness (2014). https://competitions.cr.yp.to/caesar-submissions.html

[88] M. Tempelmeier, F. De Santis, G. Sigl, J.-P. Kaps, The CAESAR-API in the real world—towards a fair evaluation of hardware CAESAR candidates, in *HOST 2018* (IEEE Computer Society, 2018), pp. 73–80. https://doi.org/10.1109/HST.2018.8383893

[89] C. Tezcan, Truncated, impossible, and improbable differential analysis of Ascon, in *ICISSP 2016* (SciTePress, 2016), pp. 325–332. https://doi.org/10.5220/0005689903250332. IACR https://ia.cr/2016/490

[90] C. Tezcan, Distinguishers for reduced round Ascon, DryGASCON, and Shamash permutations, in *NIST Lightweight Cryptography Workshop* (2019). https://csrc.nist.gov/Presentations/2019/distinguishers-forreduced-round-ascon-drygascon-a

[91] Y. Todo, Structural evaluation by generalized integral property, in *EUROCRYPT 2015*. LNCS, vol. 9056 (Springer, 2015), pp. 287–314. https://doi.org/10.1007/978-3-662-46800-5-12. IACR https://ia.cr/2015/090

[92] S. Vaudenay, D. Vizár, Can Caesar beat Galois?—Robustness of CAESAR candidates against nonce reusing and high data complexity attacks, in *ACNS 2018*. LNCS, vol. 10892 (Springer, 2018), pp. 476–494. https://doi.org/10.1007/978-3-319-93387-0-25. IACR https://ia.cr/2017/1147

[93] P. Yalla, J.-P. Kaps, Evaluation of the CAESAR hardware API for lightweight implementations, in *ReConFig 2017* (IEEE, 2017), pp. 1–6. https://doi.org/10.1109/RECONFIG.2017.8279790 (p. 33)

[94] H. Yan, X. Lai, L. Wang, Y. Yu, Y. 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. https://doi.org/10.1049/iet-ifs.2018.5263

[95] R. Zong, X. Dong, X. Wang, Collision attacks on round-reduced Gimli-Hash/Ascon-Xof/Ascon-Hash. IACR Cryptology ePrint Archive, Report 2019/1115 (IACR, 2019), p. 21. https://ia.cr/2019/1115