Submission to NIST

Ascon

v1.2

Christoph Dobraunig Maria Eichlseder
Florian Mendel Martin Schläffer

March 29, 2019

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

Contents

1 Introduction

In this document, 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, the hash function Ascon-Hash, the eXtendable output function Ascon-Xof, and a new variant Ascon-80pq with increased resistance against quantum key-search. The recommendation for NIST includes Ascon-Hash combined with Ascon-128 or Ascon-128a. 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 and hashing.

The Ascon suite and especially the underlying 320-bit permutation have been designed for high security and robustness in practice with a very low area footprint in hardware while providing good performance in software and hardware implementations. Ascon's 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 allows for fast bitsliced implementations on 32-, 16-, and 8-bit platforms. Ascon's low-degree S-box allows masked implementations with a small overhead in hardware and software. 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.

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, 6 were selected for this portfolio in 3 use-cases. During this competition, Ascon and its permutation have undergone a thorough public evaluation. So far, this has resulted in more than 15 publications giving insight in the security of Ascon, and in a total of over 80 publications that discuss Ascon more generally. All existing analysis shows a comfortable security margin. There are no indications of any weakness regarding Ascon-128 and Ascon-128a. The best attacks on round-reduced versions target the initialization reduced to 7 out of 12 rounds, and even those attacks are far from being a practical threat.

Ciphers are not used in an ideal world. Therefore, Ascon's authenticated encryption has been designed to provide robustness against certain implementation mistakes and attacks: 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 trivial forgeries.

2 Specification

This chapter provides a complete and self-contained specification of the Ascon cipher suite, starting with an overview of the algorithms in Section 2.1, the individual recommended parameter sets in Section 2.2, and the notation in Section 2.3. Afterwards, the authenticated encryption modes are specified in Section 2.4, the hashing mode in Section 2.5, and the underlying permutation in Section 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 parametrized 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 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 parametrized 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 $h = 256$, Ascon-Xof with $h = 0$ for unlimited output.

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 [DEMS16] selected as primary choices for lightweight use-cases in the final CAESAR portfolio [Cae14].

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

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

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

Recommended Pairing for Authenticated Encryption and Hashing. If it is desirable to pair authenticated encryption with hashing, it is recommended to pair the primary proposals, Ascon-128 and Ascon-Hash, since both have the same rate.

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.

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.

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

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

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

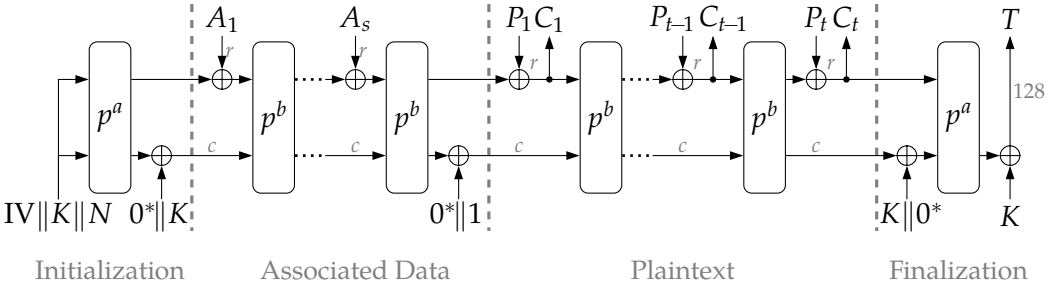
Table 3 lists the notation and symbols used in this document.

Table 3: Notation used for Ascon's interface, mode, and permutation

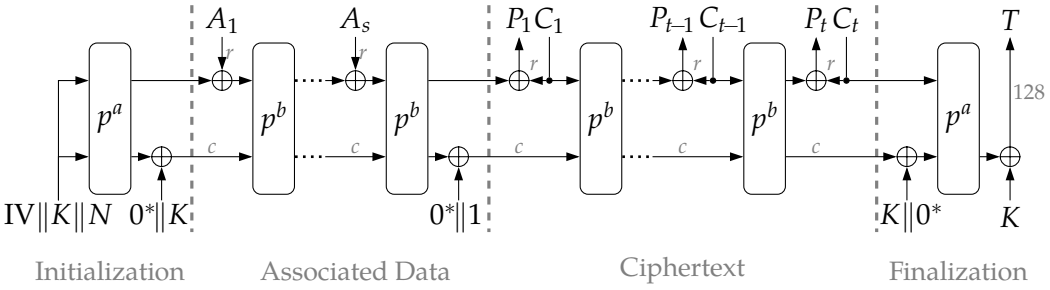
K	Secret key K of $k \leq 160$ bits
N, T	Nonce N, tag T, all of 128 bits
P, C, A	Plaintext P, ciphertext C, associated data A (in r-bit blocks P_i, C_i, A_i)
M, H	Message M, hash value H (in r-bit blocks M_i, H_i)
$\perp$	Error, verification of authenticated ciphertext failed
S	The 320-bit state S of the sponge construction
S_r, S_c	The r-bit rate and c-bit capacity 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
$\lvert x \rvert$	Length of the bitstring x in bits
$\lfloor x \rfloor_k$	Bitstring x truncated to the first (most significant) k bits
$\lceil x \rceil^k$	Bitstring x truncated to the last (least significant) 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

2.4 Authenticated Encryption

The mode of operation of Ascon for authenticated encryption is based on duplex modes like MonkeyDuplex [Dae12], but uses a stronger keyed initialization and keyed finalization function. The encryption and decryption operations are illustrated in Figure 1a and Figure 1b and specified in Algorithm 1.



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



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

Figure 1: Ascon's mode of operation.

2.4.1 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):

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

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

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:

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

Algorithm 1: Authenticated encryption and decryption procedures

Authenticated Encryption $\mathcal{E}_{k,r,a,b}(K, N, A, P)$	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 $A \in \{0,1\}^*$, plaintext $P \in \{0,1\}^*$	**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: ciphertext $C \in \{0,1\}^{\lvert P\rvert}$, tag $T \in \{0,1\}^{128}$	**Output:** plaintext $P \in \{0,1\}^{\lvert C\rvert}$ or $\perp$
Initialization $S \leftarrow \mathrm{IV}_{k,r,a,b} \,\|\, K \,\|\, N$ $S \leftarrow p^a(S) \oplus (0^{320-k} \,\|\, K)$	**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 $\lvert A\rvert > 0$ **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 Associated Data
	if $\lvert A\rvert > 0$ **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** $S_r \leftarrow S_r \oplus P_i$ $C_i \leftarrow S_r$ $S \leftarrow p^b(S)$ $S_r \leftarrow S_r \oplus P_t$ $\tilde{C}_t \leftarrow \lfloor S_r \rfloor_{\lvert P\rvert \bmod r}$	**Processing Ciphertext** $C_1 \ldots C_{t-1}\tilde{C}_t \leftarrow r$-bit blocks of C, $0 \leq \lvert\tilde{C}_t\rvert < r$ **for** $i = 1, \ldots, t-1$ **do** $P_i \leftarrow S_r \oplus C_i$ $S \leftarrow C_i \,\|\, S_c$ $S \leftarrow p^b(S)$ $\tilde{P}_t \leftarrow \lfloor S_r \rfloor_{\lvert\tilde{C}_t\rvert} \oplus \tilde{C}_t$ $S_r \leftarrow S_r \oplus (\tilde{P}_t \,\|\, 1 \,\|\, 0^*)$
Finalization $S \leftarrow p^a(S \oplus (0^r \,\|\, K \,\|\, 0^{320-r-k}))$ $T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$ **return** $C_1 \,\|\, \ldots \,\|\, C_{t-1} \,\|\, \tilde{C}_t, T$	**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** $\perp$

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

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

Each block A_i with $i = 1, \ldots, s$ 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:

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

2.4.3 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, $P_1 \| \ldots \| P_t$:

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

Encryption. In each iteration, one padded plaintext block P_i with $i = 1, \ldots, t$ 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:

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

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:

$$P_i \leftarrow S_r \oplus C_i$$
$$S \leftarrow p^b(C_i \parallel S_c), \qquad 1 \le i < t$$

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

$$\tilde{P}_t \leftarrow \lfloor S_r \rfloor_\ell \oplus \tilde{C}_t$$
$$S \leftarrow (S_r \oplus (\tilde{P}_t \parallel 1 \parallel 0^{r-1-\ell})) \parallel S_c$$

2.4.4 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 (least significant) 128 bits of the state xored with the last 128 bits of the key K:

$$S \leftarrow p^a(S \oplus (0^r \parallel K \parallel 0^{c-k}))$$
$$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$$

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

2.5 Hashing

The mode of operation for hashing is based on sponges [BDPV07]. Both the hash function Ascon-Hash with fixed output size and the eXtendable output function Ascon-Xof with variable output size internally use the same hashing algorithm $\mathcal{X}_{h,r,a}$ (see Table 2), which is illustrated in Figure 2 and specified in Algorithm 2.

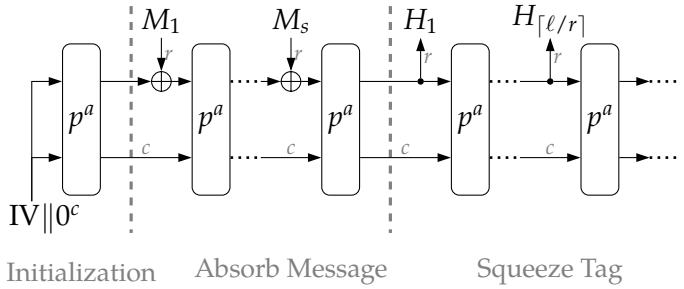


Figure 2: Hashing mode $\mathcal{X}_{h,r,a}$ in Ascon-Hash, Ascon-Xof

2.5.1 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:

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

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

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

Algorithm 2: Hashing

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

Input: message $M \in \{0,1\}^*$, output bitsize $\ell \leq h$ or ℓ arbitrary if $h = 0$
Output: hash $H \in \{0,1\}^\ell$

Initialization
  $S \leftarrow p^a(\mathrm{IV}_{h,r,a} \,\|\, 0^c)$
Absorbing
  $M_1 \ldots M_s \leftarrow M \,\|\, 1 \,\|\, 0^*$
  for $i = 1, \ldots, s$ **do**
    $S \leftarrow p^a((S_r \oplus M_i) \,\|\, S_c)$
Squeezing
  for $i = 1, \ldots, t = \lceil \ell / r \rceil$ **do**
    $H_i \leftarrow S_r$
    $S \leftarrow p^a(S)$
  return $\lfloor H_1 \,\|\, \ldots \,\|\, H_t \rfloor_\ell$

2.5.2 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, $M_1 \| ... \| M_s$:

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

The message blocks M_i with $i = 1, \ldots, s$ 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:

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

2.5.3 Squeezing

The hash output is extracted from the state in r-bit blocks H_i until the requested output length $\ell \leq h$ 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:

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

The last output block H_t is truncated to $\ell \bmod r$ bits and $H = H_1 \,\|\, \ldots \,\|\, \tilde{H}_t$ returned:

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

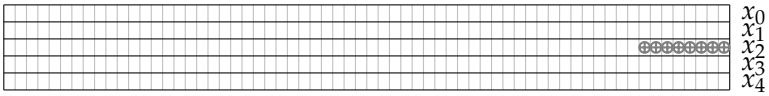
2.6 Permutation

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:

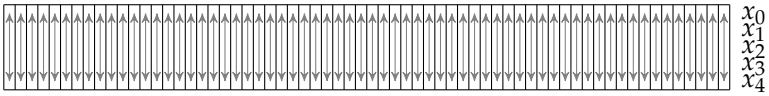
$$p = p_L \circ p_S \circ p_C \,.$$

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.

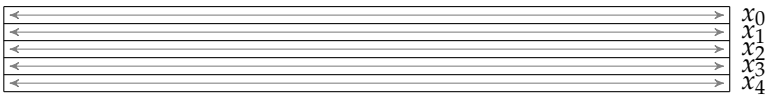
For the description and application of the round transformations, the 320-bit state S is split into five 64-bit registers words x_i, $S = x_0 \,\|\, x_1 \,\|\, x_2 \,\|\, x_3 \,\|\, x_4$ (see Figure 3).



(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 $\Sigma_i(x_i)$

Figure 3: The register words of the 320-bit state S and operations $p_L \circ p_S \circ p_C$.

2.6.1 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 Figure 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 4):

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

Table 4: The round constants c_r used in each round i of p^a and p^b.

p^{12}	p^8	p^6	Constant c_r	p^{12}	p^8	p^6	Constant c_r
0			0000000000000000f0	6	2	0	0000000000000000096
1			0000000000000000e1	7	3	1	0000000000000000087
2			0000000000000000d2	8	4	2	0000000000000000078
3			0000000000000000c3	9	5	3	0000000000000000069
4	0		0000000000000000b4	10	6	4	000000000000000005a
5	1		0000000000000000a5	11	7	5	000000000000000004b

2.6.2 Substitution Layer

The substitution layer p_S updates the state S with 64 parallel applications of the 5-bit S-box $\mathcal{S}(x)$ defined in Figure 4a to each bit-slice of the five registers $x_0 \dots x_4$ (Figure 3b). It is typically implemented in this bitsliced form with operations performed on the entire 64-bit words, as in the example code in Figure 5 (page 40). The lookup table of $\mathcal{S}$ is given in Table 5, where x_0 is the MSB and x_4 the LSB.

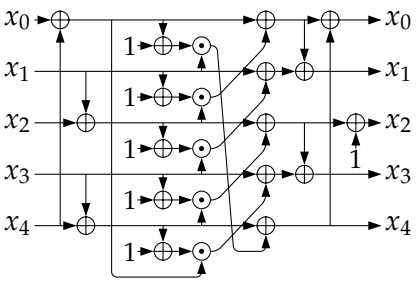
Table 5: Ascon's 5-bit S-box $\mathcal{S}$ as a lookup table.

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

2.6.3 Linear Diffusion Layer

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

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



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

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

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

Figure 4: Ascon's substitution layer and linear diffusion layer.

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 6, implementations must ensure 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.

Table 6: Security claims for recommended parameter configurations of Ascon.

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

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 6 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 trivial forgeries. In this case, forgeries can be obtained with complexity $2^{c/2}$, 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 2^{96} and 2^{128} even

if a few internal states can be recovered. In fact, it is easy to see that the product of data and time complexity for a key-recovery attack remains above 2^{128}.

Ascon-80pq has an increased key-size to provide more resistance against a quantum adversary using Grover's algorithm for key search. Since Ascon-128 and Ascon-80pq share the same building blocks and same parameters except the size of the key, we claim the same security for Ascon-80pq against classical attacks as for Ascon-128.

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.

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 Chapter 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 7. 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 other attacks, including length extension attacks and second-preimage attacks for long messages.

Table 7: 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.

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, and in a total of over 80 publications that discuss Ascon more generally. All existing analysis shows a comfortable security margin. There are no indications of any weakness regarding Ascon-128 and Ascon-128a. The best attacks on round-reduced versions target the initialization reduced to 7 out of 12 rounds.
- **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 towards either a smaller area or higher speed. More details about hardware implementations are provided in Chapter 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 Chapter 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 Section 7.4.
- **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.

4.1.1 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 [GWDE15]. 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 Chapter 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 [GA16]. It allows many trade-offs between throughput, latency, gate count, power consumption, etc. [GWDE15]. Comparison of implementation results in [GA16] 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 Chapter 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 [BDP+12]. 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 modern 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 Chapter 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 and similar protection approaches. More information about the integration of countermeasures against implementation attacks can be found in Section 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 [ACS15].
- **Low overhead for short messages.** Ascon is among the fastest CAESAR candidates for short messages according to current software benchmarking results[AA16; BL], since its initialization and finalization overhead is much smaller compared to most blockcipher-based constructions, stream ciphers, or large-state sponges. 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.

4.1.2 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 make it especially efficient on high-end CPUs. Up to 5 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 Chapter 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 [BDPV07]. 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 [BDPV11c] designed to improve the 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

5.1.1 Choice of the Mode for Authenticated Encryption

The design principles of Ascon's authenticated encryption mode follow the sponge methodology [BDPV07]. More precisely, they are similar to SpongeWrap [BDPV11a] and MonkeyDuplex [Dae12]. 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 amount 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 trivial forgeries. 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 capacity part S_c of the state.

5.1.2 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 sponges [BDPV07]. For hashing, sponges provide similar benefits as for authenticated encryption:

- Sponges are well-studied and have been analyzed and proven secure for different applications in a large amount 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).
- Plaintext and ciphertext blocks can both be computed online, without waiting for the complete message or even the message length initially be present.

5.1.3 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 Chapter 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 the scheme is not used correctly, an attacker is more likely to recover the internal state during the data processing. This still does not directly lead to an efficient key recovery attack on the scheme.

The only difference between Ascon-80pq and Ascon-128 is the increased length of the key. This increased key length provides additional protection against exhaustive key search in the case the availability of quantum computers becomes evident. Since the other tunable security parameters (the number of rounds of the permutations) have not been increased, the security claim for Ascon-80pq against classical attacks stays the same as for Ascon-128.

Ascon-Hash and Ascon-Xof reuse the 12-round variant of the Ascon permutation that 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 [BDPV07; BDPV08] provides a secure 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.

5.1.4 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

5.2.1 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 shows that the constants are not used to implement any backdoors.

The pattern can also easily be extended for up to 16 rounds if a very high security margin is desired. Adding more than 16 rounds is not expected to further improve the security margin.

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).

5.2.2 Choice of the Substitution Layer

The substitution layer is the only non-linear part of the round transformation. It mixes 5 bits, each at the same bit position in one of the 5 state words. 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 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 non-linearly.

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. 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 [NRS11] and masking [CJRR99; GP99], 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 [BBK+13] 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.

5.2.3 Choice of the 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 [DSW09], Hamsi [Küç09], 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 [Nat08]. 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 Section 6.1. For a full list of publications and comments, we refer to Section 6.4. In Section 6.2, we discuss the security of the modes of operation for authenticated encryption and hashing. In Section 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 8 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 42 %.

Table 8: Best known analysis of the Ascon permutation.

Type	Target	Rounds	Time	Method	Reference
Distinguisher	Permutation	12 / 12	2^{130}	Zero-sum	[DEMS15]
	Permutation	11 / 12	2^{315}	Integral	[Tod15]
	Permutation	5 / 12	2^{193}	Differential	Section 6.3
	Permutation	5 / 12	2^{189}	Linear	[DEM15]
Distinguisher	Permutation	11 / 12	2^{85}	Zero-sum	Section 6.3
	Permutation	7 / 12	2^{65}	Integral	[Tod15]
	Permutation	5 / 12	2^{109}	Truncated Differential	[Tez16]
	Permutation	4 / 12	2^{107}	Differential	Section 6.3
	Permutation	4 / 12	2^{101}	Linear	[DEM15]
Distinguisher	Permutation	5 / 12	–	Zero-Correlation	Section 6.3
	Permutation	5 / 12	–	Impossible Differential	Section 6.3
	Permutation	3 / 12	–	Subspace Trails	[LTW18]

Table 9: Best known analysis of Ascon (⊘ = misuse).

Type	Target	Rounds	Time	Method	Reference
Key recovery	Ascon initialization	7 / 12	2^{104}	Cube-like	[LDW17]
	Ascon initialization	5 / 12	2^{36}	Diff.-linear	[DEMS15]
	Ascon initialization	7 / 12	2^{97} ⊘	Cube-like	[LZWW17]
Forgery	Ascon finalization	4 / 12	2^{101}	Differential	[DEMS15]
	Ascon finalization	6 / 12	2^{33} ⊘	Cube tester	[LZWW17]
State recovery	Ascon-128a iteration	2 / 8	–	Sat-Solver	[DKM+17]
	Ascon-128 iteration	5 / 6	2^{66} ⊘	Cube-like	[LZWW17]

Table 10: Best known analysis of Ascon-Hash and Ascon-Xof (⊘ = chosen IV).

Type	Target	Size	Rounds	Time	Method	Reference
Preimage	Ascon-Xof	64	2 / 12	2^{39}	Cube-like	[DEMS19]
	Ascon-Xof	64	6 / 12	$2^{63.3}$	Algebraic	[DEMS19]
Collision	Ascon-Xof	all	4 / 12	– ⊘	Differential	[DEMS19]

6.2 Analysis of the Modes

6.2.1 Hashing and EXtendable Output Function

The mode of operation in Ascon-Hash and Ascon-Xof is closely based on the sponge methodology proposed by Bertoni et al. [BDPV07] and profits from the extensive literature on sponges, 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 [BDPV08].

6.2.2 Authenticated Encryption

The mode of operation in Ascon is based on the duplex construction [BDPV11a], or more specifically, a variant of the AEAD mode MonkeyDuplex with its reduced number of rounds in the data processing phases [Dae12]. 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.

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 $2^{c/2}$ [BDPV11b; JLM14]. Andreeva et al. [ADMV15] show that the time complexity is at least $\min\{2^k, 2^c/\mu\}$, where μ is the multiplicity [BDPV10], which is small for nonce-based schemes.

Daemen et al. [DMV17] provide stronger bounds based on distinguishing Ascon 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 trivial forgeries. Possible attacks after such a state recovery include forgeries by finding internal collisions ($2^{c/2}$ time).

6.3 Analysis of the Permutation

6.3.1 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 and Table 13b, respectively. We could not find any differential and linear characteristics for more than 4 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).

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	·	·	·	·	·	·	·	·	·	·

(b) Linear approximation table: $\mathrm{LAT}[\alpha,\beta] = |\{x : \alpha^\top \cdot x = \beta^\top \cdot \mathcal{S}(x)\}| - 16$

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

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

Table 13: The 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 [DEMS15; DEM15].

(a) Differential 4-round characteristic

r	Active S-boxes	$\#\mathcal{S}$	$\log_2(p)$
0	441326c0236cca84	23	-47
1	8040000000040000	3	-12
2	0000100004040000	3	-6
3	c0489800262500a0	15	-42
$\sum$		44	-107

(b) Linear 4-round characteristic

r	Active S-boxes	$\#\mathcal{S}$	$\log_2(\varepsilon)$
0	8181224200028685	15	-19
1	0100004000000001	3	-4
2	0000000010080001	3	-7
3	04314f4725f80001	22	-23
$\sum$		43	-50

(c) Differential 5-round characteristic

r	Active S-boxes	$\#\mathcal{S}$	$\log_2(p)$
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

r	Active S-boxes	$\#\mathcal{S}$	$\log_2(\varepsilon)$
0	8181224200028685	15	-19
1	0100004000000001	3	-4
2	0000000010080001	3	-7
3	04314f4725f80001	22	-43
4	04364206f5a80802	24	-25
$\sum$		67	-94

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 rate 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 chapters, 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 rate, but there are no restrictions on the output difference. A corresponding characteristic for 4 out of 12 rounds is provided in Table 14c [DEMS15].

Table 14: Differential characteristics for forgeries in Ascon.

(a) Collision-producing differential for Ascon-128's 6-round state update

r	Active S-boxes	$\#\mathcal{S}$
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

r	Active S-boxes	$\#\mathcal{S}$
0	8000000000000000	1
1	c200000000000000	3
2	e238e10000000000	11
3	73b7fbf67f6f19f0	44
4	bb4ffe8fd5dddf7f	48
5	ffffdfffffffffff	63
6	2d0486c240902436	20
7	2080000000000000	2
$\sum$		192

(c) Truncated differential for 4 / 12 rounds of Ascon's finalization, $p = 2^{-101}$ [DEMS15]

r	Active S-boxes	$\#\mathcal{S}$
0	8000000000000000	1
1	8000100801000004	5
2	d302904803844086	18
3	fbbff36d73e4f045	41

Impossible Differentials and Zero-Correlation Approximations

Using an automated search tool, we were able to find impossible differentials for up to 5 rounds (Table 15a) and zero-correlation linear hulls 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 attack on Ascon using this property of the permutation.

Table 15: Impossible differential and zero-correlation linear hull for 5 rounds of p.

(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

Other published properties include a differential-linear attack on up to 5 rounds of Ascon's initialization with practical complexity [DEMS15; BDKW19] and truncated differential distinguishers based on undisturbed bits for up to 5 rounds with 2^{109} data [Tez16].

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

$$\begin{aligned}
y_{0,i} &= x_{4,i}\,x_{1,i} \oplus x_{3,i} \oplus x_{2,i}\,x_{1,i} \oplus x_{2,i} \oplus x_{1,i}\,x_{0,i} \oplus x_{1,i} \oplus x_{0,i},\\
y_{1,i} &= x_{4,i} \oplus x_{3,i}\,x_{2,i} \oplus x_{3,i}\,x_{1,i} \oplus x_{3,i} \oplus x_{2,i}\,x_{1,i} \oplus x_{2,i} \oplus x_{1,i} \oplus x_{0,i},\\
y_{2,i} &= x_{4,i}\,x_{3,i} \oplus x_{4,i} \oplus x_{2,i} \oplus x_{1,i} \oplus 1,\\
y_{3,i} &= x_{4,i}\,x_{0,i} \oplus x_{4,i} \oplus x_{3,i}\,x_{0,i} \oplus x_{3,i} \oplus x_{2,i} \oplus x_{1,i} \oplus x_{0,i},\\
y_{4,i} &= x_{4,i}\,x_{1,i} \oplus x_{4,i} \oplus x_{3,i} \oplus x_{1,i}\,x_{0,i} \oplus x_{1,i}\,.
\end{aligned} \tag{6.1}$$

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

$$\begin{aligned}
y_{0,i} &= x_{0,i} \oplus x_{0,i+19} \oplus x_{0,i+28}\\
y_{1,i} &= x_{1,i} \oplus x_{1,i+61} \oplus x_{1,i+39}\\
y_{2,i} &= x_{2,i} \oplus x_{2,i+1} \oplus x_{2,i+6}\\
y_{3,i} &= x_{3,i} \oplus x_{3,i+10} \oplus x_{3,i+17}\\
y_{4,i} &= x_{4,i} \oplus x_{4,i+7} \oplus x_{4,i+41}\,.
\end{aligned} \tag{6.2}$$

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 Boura and Canteaut [BC10, Theorem 1 with $\ell = 192$ for both $\mathcal{S}$ and $\mathcal{S}^{-1}$] and Boura et al. [BCD11, Theorem 2 with $\gamma = 3$ for both $\mathcal{S}$ and $\mathcal{S}^{-1}$] is listed in Table 16.

Table 16: Upper bound for the algebraic degree after R rounds [BC10, Theorem 1], [BCD11, 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		

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

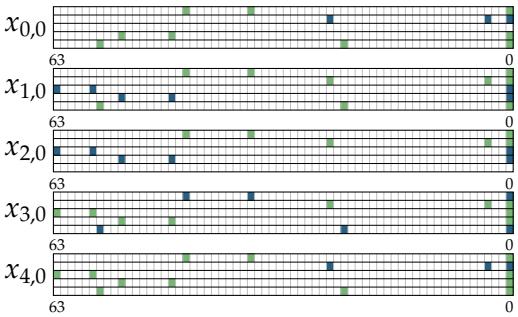
(a) # Variables in ANF of $x_{w,i}$

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 [DEMS15; LDW17]. 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 [LDW17]. An alternative approach suggested by Li et al. [LDW17] 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 random permutations. The full 12-round permutation can be distinguished with 2^{130} data [DEMS15] (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 depend on selected key bits [DEMS15]. 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. [LDW17] 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 [DEMS19]. 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 [Ber10]. 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 $2^{63.3}$ [DEMS19].

6.3.3 Other Properties

Integral Distinguishers and Division Property

Based on the division property, Todo [Tod15] 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. [GRW16] 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).

Table 18: Division property results

(a) S-box property [GRW16]

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

(b) Integral distinguishers for the permutation [Tod15]

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

Subspace Trails

Leander et al. [LTW18] 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. [DKM+17] 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 [Cae14], 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.

Closer analysis of Ascon's differential-linear properties:

- Achiya Bar-On, Orr Dunkelman, Nathan Keller, and Ariel Weizman. "DLCT: A New Tool for Differential-Linear Cryptanalysis". In: EUROCRYPT 2019. LNCS. Springer, 2019. IACR: 2019/256.

Subspace trails for a small number of rounds for Ascon's permutation:

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

Evaluation of the security of Ascon in misuse settings:

- Serge Vaudenay and Damian Vizár. "Can Caesar Beat Galois? – Robustness of CAESAR Candidates Against Nonce Reusing and High Data Complexity Attacks". In: ACNS 2018. Vol. 10892. LNCS. Springer, 2018, pp. 476–494. DOI: 10.1007/978-3-319-93387-0_25. IACR: 2017/1147.

Cube-like key-recovery attack on 7 (out of 12) rounds of the initialization of Ascon in $2^{103.9}$ time:

Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. “Conditional Cube Attack on Round-Reduced ASCON”. In: IACR Transactions on Symmetric Cryptology 2017.1 (2017), pp. 175–202. DOI: 10.13154/tosc.v2017.i1.175-202. IACR: 2017/160. URL: https://github.com/lizhengcn/Ascon_test.

Cube-like attacks in a nonce-misuse setting on round-reduced Ascon:

Yanbin Li, Guoyan Zhang, Wei Wang, and Meiqin Wang. “Cryptanalysis of round-reduced ASCON”. In: SCIENCE CHINA Information Sciences 60.3 (2017), p. 38102. DOI: 10.1007/s11432-016-0283-3.

SAT-based state recovery on 2 (out of 8) rounds of the data processing of Ascon-128a:

Ashutosh Dhar Dwivedi, Miloš Klouček, Pawel Morawiecki, Ivica Nikolič, Josef Pieprzyk, and Sebastian Wójtowicz. “SAT-based Cryptanalysis of Authenticated Ciphers from the CAESAR Competition”. In: SECRYPT ICETE 2017. SciTePress, 2017, pp. 237–246. DOI: 10.5220/0006387302370246. IACR: 2016/1053.

Evaluating the properties of Ascon’s authenticated encyption mode regarding reforgeability:

Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. “Reforgeability of Authenticated Encryption Schemes”. In: ACISP 2017. Vol. 10343. LNCS. Springer, 2017, pp. 19–37. DOI: 10.1007/978-3-319-59870-3_2. IACR: 2017/332.

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:

Cihangir Tezcan. “Truncated, Impossible, and Improbable Differential Analysis of Ascon”. In: ICISSP 2016. SciTePress, 2016, pp. 325–332. DOI: 10.5220/0005689903250332. IACR: 2016/490.

Security of Ascon’s S-box with respect to the division property:

Faruk Göloğlu, Vincent Rijmen, and Qingju Wang. “On the division property of S-boxes”. 2016. IACR: 2016/188.

Several linear characteristic for Ascon’s permutation:

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. “Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates”. In: ASIACRYPT 2015. Vol. 9453. LNCS. Springer, 2015, pp. 490–509. DOI: 10.1007/978-3-662-48800-3_20. IACR: 2015/1200.

Ascon’s authenticated encryption mode supports secure implementations on limited-memory devices:

Megha Agrawal, Donghoon Chang, and Somitra Sanadhya. “sp-AELM: Sponge Based Authenticated Encryption Scheme for Memory Constrained Devices”. In: ACISP 2015. Vol. 9144. LNCS. Springer, 2015, pp. 451–468. DOI: 10.1007/978-3-319-19962-7_26.

Evaluation of Ascon’s permutation using the division property:

Yosuke Todo. “Structural Evaluation by Generalized Integral Property”. In: EUROCRYPT 2015. Vol. 9056. LNCS. Springer, 2015, pp. 287–314. DOI: 10.1007/978-3-662-46800-5_12. IACR: 2015/090.

Suggestions to absorb authenticated data more efficiently:

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

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:

Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. “Cryptanalysis of Ascon”. In: CT-RSA 2015. Vol. 9048. LNCS. Springer, 2015, pp. 371–387. DOI: 10.1007/978-3-319-16715-2_20. IACR: 2015/030.

Security proof for Ascon ’s authenticated encryption mode even for higher rates:

Philipp Jovanovic, Atul Luykx, and Bart Mennink. “Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes”. In: ASIACRYPT 2014. Vol. 8873. LNCS. Springer, 2014, pp. 85–104. DOI: 10.1007/978-3-662-45611-8_5. IACR: 2014/373.

Security analysis and bounds for the full-state keyed duplex with application to Ascon-128 and Ascon-128a:

Joan Daemen, Bart Mennink, and Gilles Van Assche. “Full-State Keyed Duplex with Built-In Multi-user Support”. In: ASIACRYPT 2017. Vol. 10625. LNCS. Springer, 2017, pp. 606–637. DOI: 10.1007/978-3-319-70697-9_21. IACR: 2017/498.

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 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.

Detailed software performance results and comparisons for a large number of platforms are (and will be) given in eBACS, the ECRYPT Benchmarking of Cryptographic Systems [BL]. A preliminary overview of the software performance of Ascon-128 and Ascon-128a is given in Table 19a and Table 19b. The software performance of Ascon-Hash and Ascon-Xof is largely the same as for Ascon-128 with doubled cycles per byte.

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 and Table 19b.

Ankele and Ankele [AA16] 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 [AA16, 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,

Table 19: Ascon-128 and Ascon-128a software performance in cycles per byte. Message length is length of encrypted plaintext with empty associated data.

(a) Ascon-128

Message Length	1	8	16	32	64	1536	long
AMD Ryzen 7 1700*					14.5	8.8	8.6
Intel Xeon E5-2609 v4*					17.3	10.8	10.5
Cortex-A53 (ARMv8)*					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)*					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

*Results taken from eBACS [BL].

(b) Ascon-128a

Message Length	1	8	16	32	64	1536	long
AMD Ryzen 7 1700*					12.0	6.0	5.7
Intel Xeon E5-2609 v4*					14.1	7.3	6.9
Cortex-A57 (ARMv8)*					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)*					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

*Results taken from eBACS [BL].

it lends itself to bitsliced implementations in software as well as simple and clean hardware implementations. 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 Figure 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;
```

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

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 [BDP+12]. 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. For such implementations on resource constrained devices, it is beneficial that the S-box of Ascon can also be implemented using just two temporary registers as shown in Figure 6.

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

Figure 6: Reducing register pressure for bitsliced implementation of the 5-bit S-box $\mathcal{S}(x)$ (inspired by [Dae18; DHVV18]).

That bit interleaving is a very viable strategy can be seen in the work of Noël Bangma [Ban18], 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 plaintext messages of 64 bytes [Ban18, Table 5.1].

As shown in Table 20, the flexibility of Ascon also translates to hardware, where a round-based implementation of Ascon needs 7.08 kGE, but also very small implementations requiring just 2.57 kGE are possible [GWDE15].

Table 20: Ascon-128 hardware implementations taken from [GWDE15]

Design	Chip Area w/o interface	Chip Area w/ interface	Throughput	Power at 1 MHz	Energy
	[kGE]	[kGE]	[Mbps]	[µW]	[µJ/byte]
Ascon-fast					
1 round	7.08	7.95	5,524	43	33
2 rounds	10.61	11.48	8,425	72	27
3 rounds	14.26	15.13	10,407	102	25
6 rounds	24.93	25.80	**13,218**	184	**23**
Ascon64-bit	4.99	5.86	72	32	1,397
Ascon-x-low-area	**2.57**	**3.75**	14	**15**	5,706

7.3 Further Reading on Efficiency

CAESAR API

- Michael Tempelmeier, Fabrizio De Santis, Georg Sigl, and Jens-Peter 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. DOI: 10.1109/HST.2018.8383893.

- William Diehl and Kris Gaj. "RTL implementations and FPGA benchmarking of selected CAESAR Round Two authenticated ciphers". In: Microprocessors and Microsystems 52 (2017), pp. 202–218. DOI: 10.1016/j.micpro.2017.06.003.

- Ralph Ankele and Robin Ankele. "Software Benchmarking of the 2nd round CAESAR Candidates". 2016. IACR: 2016/740.

Other implementations

- Ko Stoffelen. "Optimizing S-Box Implementations for Several Criteria Using SAT Solvers". In: FSE 2016. Vol. 9783. LNCS. Springer, 2016, pp. 140–160. DOI: 10.1007/978-3-662-52993-5_8. IACR: 2016/198.

- Noël Bangma. "Ascon: An attempt in NEON on the Cortex-A8". Bachelor's Thesis. 2018. URL: https://www.cs.ru.nl/bachelorscripties/2018/Noel_Bangma___4433939___Ascon_An_attempt_in_NEON_on_the_Cortex-A8.pdf.

- Rajesh Kumar Pal. "Implementation and Evaluation of Authenticated Encryption Algorithms on Java Card Platform (master's thesis)". Master's Thesis. 2017. URL: https://is.muni.cz/th/448415/fi_m/MSThesis_IS.pdf.

Michael Fivez. “Energy Efficient Hardware Implementations of CAESAR Submissions”. Master’s Thesis. 2016. URL: http://securewww.esat.kuleuven.be/cosic/publications/thesis-279.pdf.

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. Gross et al. [GWDE15] provide threshold implementations of Ascon-128 as small as 7.97 kGE. Besides this, many other state-of-the-art masking approaches have been applied on Ascon, like UMA [GM17] and DOM [GMK16], even for high protection order (see Table 21). Links to various implementations of Ascon, including DOM and UMA implementations, can be found at https://ascon.iaik.tugraz.at/.

Table 21: DOM implementations for various protection orders [GM17; GM18].

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

Next, we give a list of papers that either evaluate the side-channel resistance of Ascon or elaborate protection mechanisms against side-channel attacks:

Niels Samwel and Joan Daemen. “DPA on hardware implementations of Ascon and Keyak”. In: Computing Frontiers – CF’17. ACM, 2017, pp. 415–424. DOI: 10.1145/3075564.3079067.

Niels Samwel. “Side-Channel Analysis of Keccak and Ascon”. Master’s Thesis. 2016. URL: http://www.ru.nl/publish/pages/769526/niels_samwel.pdf.

Alexandre Adomnicai, Jacques J. A. Fournier, and Laurent Masson. “Masking the Lightweight Authenticated Ciphers ACORN and Ascon in Software”. 2018. IACR: 2018/708.

Hannes Gross, Rinat Iusupov, and Roderick Bloem. “Generic Low-Latency Masking in Hardware”. In: IACR Transactions on Cryptographic Hardware and Embedded Systems 2018.2 (2018), pp. 1–21. DOI: 10.13154/tches.v2018.i2.1-21. IACR: 2017/1223.

Hannes Gross, Erich Wenger, Christoph Dobraunig, and Christoph Ehrenhöfer. "Ascon hardware implementations and side-channel evaluation". In: Microprocessors and Microsystems 52 (2017), pp. 470–479. DOI: 10.1016/j.micpro.2016.10.006.

Hannes Gross and Stefan Mangard. "Reconciling d+1 Masking in Hardware and Software". In: CHES 2017. Vol. 10529. LNCS. Springer, 2017, pp. 115–136. IACR: 2017/103.

Liran Lerman, Olivier Markowitch, and Nikita Veshchikov. "Comparing Sboxes of Ciphers from the Perspective of Side-Channel Attacks". In: AsianHOST 2016. IEEE Computer Society, 2016, pp. 1–6. DOI: 10.1109/AsianHOST.2016.7835556. IACR: 2016/993.

Acknowledgments

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.

This work was initiated while all authors were working at Graz University of Technology. 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 by the Austrian Government (FFG/SFG COMET 836628: SeCoS and FIT-IT 835919: SePAG).

Bibliography

[AA16] Ralph Ankele and Robin Ankele. "Software Benchmarking of the 2nd round CAESAR Candidates". 2016. IACR: 2016/740 (pp. 20, 38).

[ACS15] Megha Agrawal, Donghoon Chang, and Somitra Sanadhya. "sp-AELM: Sponge Based Authenticated Encryption Scheme for Memory Constrained Devices". In: ACISP 2015. Vol. 9144. LNCS. Springer, 2015, pp. 451–468. DOI: 10.1007/978-3-319-19962-7_26 (p. 20).

[ADMV15] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. "Security of Keyed Sponge Constructions Using a Modular Proof Approach". In: FSE 2015. Vol. 9054. LNCS. Springer, 2015, pp. 364–384. DOI: 10.1007/978-3-662-48116-5_18 (p. 27).

[Ban18] Noël Bangma. "Ascon: An attempt in NEON on the Cortex-A8". Bachelor's Thesis. 2018. URL: https://www.cs.ru.nl/bachelorscripties/2018/Noel_Bangma___4433939___Ascon_An_attempt_in_NEON_on_the_Cortex-A8.pdf (p. 40).

[BBK+13] Begül Bilgin, Andrey Bogdanov, Miroslav Knezevic, Florian Mendel, and Qingju Wang. "Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware". In: CHES 2013. Vol. 8086. LNCS. Springer, 2013, pp. 142–158. DOI: 10.1007/978-3-642-40349-1_9. IACR: 2015/424 (p. 25).

[BC10] Christina Boura and Anne Canteaut. "A zero-sum property for the Keccak-f permutation with 18 Rounds". In: ISIT 2010. IEEE, 2010, pp. 2488–2492. DOI: 10.1109/ISIT.2010.5513442 (p. 32).

[BCD11] Christina Boura, Anne Canteaut, and Christophe De Cannière. "Higher-Order Differential Properties of Keccak and Luffa". In: FSE 2011. Vol. 6733. LNCS. Springer, 2011, pp. 252–269. DOI: 10.1007/978-3-642-13858-4_15. IACR: 2010/589 (p. 32).

[BDKW19] Achiya Bar-On, Orr Dunkelman, Nathan Keller, and Ariel Weizman. "DLCT: A New Tool for Differential-Linear Cryptanalysis". In: EUROCRYPT 2019. LNCS. Springer, 2019. IACR: 2019/256 (p. 31).

[BDP+12] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. "Keccak implementation overview version 3.2". 2012. URL: https://keccak.team (pp. 19, 40).

[BDPV07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Sponge functions". Ecrypt Hash Workshop 2007. 2007. URL: http://sponge.noekeon.org/SpongeFunctions.pdf (pp. 11, 21–23, 27).

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "On the Indifferentiability of the Sponge Construction". In: EUROCRYPT 2008. Vol. 4965. LNCS. Springer, 2008, pp. 181–197. DOI: 10.1007/978-3-540-78967-3_11. URL: http://keccak.team/files/SpongeIndifferentiability.pdf (pp. 23, 27).

[BDPV10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Sponge-Based Pseudo-Random Number Generators". In: CHES 2010. Vol. 6225. LNCS. Springer, 2010, pp. 33–47. DOI: 10.1007/978-3-642-15031-9_3. URL: http://sponge.noekeon.org/SpongePRNG.pdf (p. 27).

[BDPV11a] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications". In: SAC 2011. Vol. 7118. LNCS. Springer, 2011, pp. 320–337. DOI: 10.1007/978-3-642-28496-0_19. IACR: 2011/499 (pp. 21, 27).

[BDPV11b] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "On the security of the keyed sponge construction". In: SKEW 2011. 2011. URL: http://sponge.noekeon.org/SpongeKeyed.pdf (p. 27).

[BDPV11c] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "The Keccak Reference". Submission to NIST (Round 3). 2011. URL: https://keccak.team (p. 21).

[Ber10] Daniel J. Bernstein. "Second preimages for 6 (7 (8??)) rounds of Keccak?" Posted on the NIST mailing list. 2010. URL: http://ehash.iaik.tugraz.at/uploads/6/65/NIST-mailing-list_Bernstein-Daemen.txt (p. 34).

[BL] "eBACS: ECRYPT Benchmarking of Cryptographic Systems". URL: https://bench.cr.yp.to (visited on 02/14/2019) (pp. 20, 38, 39).

[Cae14] The CAESAR committee. "CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness". 2014. URL: https://competitions.cr.yp.to/caesar-submissions.html (pp. 6, 35).

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. "Towards Sound Approaches to Counteract Power-Analysis Attacks". In: CRYPTO '99. Vol. 1666. LNCS. Springer, 1999, pp. 398–412. DOI: 10.1007/3-540-48405-1 (p. 25).

[Dae12] Joan Daemen. "Permutation-based Encryption, Authentication and Authenticated Encryption". DIAC 2012. July 2012 (pp. 8, 21, 27).

[Dae18] Joan Daemen. "Personal communication about implementing the 3-bit χ layer". Dec. 2018 (p. 40).

[DEM15] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates". In: ASIACRYPT 2015. Vol. 9453. LNCS. Springer, 2015, pp. 490–509. DOI: 10.1007/978-3-662-48800-3_20. IACR: 2015/1200 (pp. 26, 30).

[DEMS15] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Cryptanalysis of Ascon". In: CT-RSA 2015. Vol. 9048. LNCS. Springer, 2015, pp. 371–387. DOI: 10.1007/978-3-319-16715-2_20. IACR: 2015/030 (pp. 26, 27, 30, 31, 33, 34).

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon v1.2". Submission to Round 3 of the CAESAR competition. 2016. URL: https://ascon.iaik.tugraz.at (p. 6).

[DEMS19] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Preliminary Analysis of Ascon-Xof and Ascon-Hash". Technical Report. 2019. URL: https://ascon.iaik.tugraz.at (pp. 27, 34).

[DHVV18] Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. "The design of Xoodoo and Xoofff". In: IACR Transactions on Symmetric Cryptology 2018.4 (2018), pp. 1–38. DOI: 10.13154/tosc.v2018.i4.1-38. IACR: 2018/767 (p. 40).

[DKM+17] Ashutosh Dhar Dwivedi, Miloš Klouček, Pawel Morawiecki, Ivica Nikolič, Josef Pieprzyk, and Sebastian Wójtowicz. "SAT-based Cryptanalysis of Authenticated Ciphers from the CAESAR Competition". In: SECRYPT ICETE 2017. SciTePress, 2017, pp. 237–246. DOI: 10.5220/0006387302370246. IACR: 2016/1053 (pp. 27, 35).

[DMV17] Joan Daemen, Bart Mennink, and Gilles Van Assche. "Full-State Keyed Duplex with Built-In Multi-user Support". In: ASIACRYPT 2017. Vol. 10625. LNCS. Springer, 2017, pp. 606–637. DOI: 10.1007/978-3-319-70697-9_21. IACR: 2017/498 (p. 28).

[DSW09] Christophe De Cannière, Hisayoshi Sato, and Dai Watanabe. "Hash Function Luffa: Specification". Submission to NIST (Round 2). 2009. URL: http://www.hitachi.com/rd/yrl/crypto/luffa/ (p. 25).

[GA16] Kris Gaj and ATHENa Team. "ATHENa: Automated Tool for Hardware Evaluation". 2016. URL: https://cryptography.gmu.edu/athena/ (p. 19).

[GM17] Hannes Gross and Stefan Mangard. "Reconciling d+1 Masking in Hardware and Software". In: CHES 2017. Vol. 10529. LNCS. Springer, 2017, pp. 115–136. IACR: 2017/103 (p. 42).

[GM18] Hannes Gross and Stefan Mangard. "A unified masking approach". In: Journal of Cryptographic Engineering 8.2 (2018), pp. 109–124. DOI: 10.1007/s13389-018-0184-y (p. 42).

[GMK16] Hannes Gross, Stefan Mangard, and Thomas Korak. "Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order". Cryptology ePrint Archive, Report 2016/486. 2016. URL: https://eprint.iacr.org/2016/486 (p. 42).

[GP99] Louis Goubin and Jacques Patarin. "DES and Differential Power Analysis (The "Duplication" Method)". In: CHES'99. Vol. 1717. LNCS. Springer, 1999, pp. 158–172. DOI: 10.1007/3-540-48059-5 (p. 25).

[GRW16] Faruk Göloğlu, Vincent Rijmen, and Qingju Wang. "On the division property of S-boxes". 2016. IACR: 2016/188 (p. 34).

[GWDE15] Hannes Gross, Erich Wenger, Christoph Dobraunig, and Christoph Ehrenhöfer. "Suit up! – Made-to-Measure Hardware Implementations of Ascon". In: DSD 2015. IEEE Computer Society, 2015, pp. 645–652. DOI: 10.1109/DSD.2015.14. IACR: 2015/034 (pp. 19, 40–42).

[JLM14] Philipp Jovanovic, Atul Luykx, and Bart Mennink. "Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes". In: ASIACRYPT 2014. Vol. 8873. LNCS. Springer, 2014, pp. 85–104. DOI: 10.1007/978-3-662-45611-8_5. IACR: 2014/373 (p. 27).

[Küç09] Özgül Küçük. "The Hash Function Hamsi". Submission to NIST (Round 2). 2009. URL: http://homes.esat.kuleuven.be/~okucuk/hamsi/ (p. 25).

[LDW17] Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. "Conditional Cube Attack on Round-Reduced ASCON". In: IACR Transactions on Symmetric Cryptology 2017.1 (2017), pp. 175–202. DOI: 10.13154/tosc.v2017.i1.175-202. IACR: 2017/160. URL: https://github.com/lizhengcn/Ascon_test (pp. 27, 33, 34).

[LTW18] Gregor Leander, Cihangir Tezcan, and Friedrich Wiemer. "Searching for Subspace Trails and Truncated Differentials". In: IACR Transactions on Symmetric Cryptology 2018.1 (2018), pp. 74–100. DOI: 10.13154/tosc.v2018.i1.74-100 (pp. 26, 35).

[LZWW17] Yanbin Li, Guoyan Zhang, Wei Wang, and Meiqin Wang. "Cryptanalysis of round-reduced ASCON". In: SCIENCE CHINA Information Sciences 60.3 (2017), p. 38102. DOI: 10.1007/s11432-016-0283-3 (p. 27).

[Nat08] National Institute of Standards and Technology. "FIPS PUB 180-3: Secure Hash Standard". Federal Information Processing Standards Publication 180-3, U.S. Department of Commerce. 2008. URL: http://csrc.nist.gov/publications/fips/fips180-3/fips-180-3_final.pdf (p. 25).

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. "Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches". In: Journal of Cryptology 24.2 (2011), pp. 292–321. DOI: 10.1007/s00145-010-9085-7 (p. 25).

[Tez16] Cihangir Tezcan. "Truncated, Impossible, and Improbable Differential Analysis of Ascon". In: ICISSP 2016. SciTePress, 2016, pp. 325–332. DOI: 10.5220/0005689903250332. IACR: 2016/490 (pp. 26, 31).

[Tod15] Yosuke Todo. "Structural Evaluation by Generalized Integral Property". In: EUROCRYPT 2015. Vol. 9056. LNCS. Springer, 2015, pp. 287–314. DOI: 10.1007/978-3-662-46800-5_12. IACR: 2015/090 (pp. 26, 34).