Ascon v1.1

Submission to the CAESAR Competition

Christoph Dobraunig, Maria Eichlseder,
Florian Mendel, Martin Schläffer

Institute for Applied Information Processing and Communications
Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria

Infineon Technologies Austria AG
Babenbergerstraße 10, A-8020 Graz, Austria

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

August 29, 2015

Chapter 1

Specification

1.1 Parameters

Ascon is a family of authenticated encryption designs $\text{Ascon}_{a,b}$-k-r. The family members are parametrized by the key length $k \leq 128$ bits, the rate r and internal round numbers a and b. Each design specifies an authenticated encryption algorithm $\mathcal{E}_{a,b,k,r}$ and a decryption algorithm $\mathcal{D}_{a,b,k,r}$.

The inputs for the authenticated encryption procedure $\mathcal{E}_{a,b,k,r}$ are the plaintext P, associated data A, a secret key K with k bits and a public message number (nonce) N with k bits. No secret message number is used, i.e., its length is 0 bits. The output of the authenticated encryption procedure is an authenticated ciphertext C of exactly the same length as the plaintext P, and an authentication tag T of size k bits, which authenticates both A and P:

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

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

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

1.2 Recommended parameter sets

Tunable parameters include the key size k, the rate r, as well as the number of rounds a for the initialization and finalization permutation p^a, and the number of rounds b for the intermediate permutation p^b processing the associated data and plaintext. Table 1 contains our recommended parameter configurations. The list is sorted by priority, i.e., the primary recommendation is Ascon-128 and the secondary recommendation is Ascon-128a.

Table 1: Recommended parameter configurations for Ascon.

name	algorithm	bit size of				rounds	
		key	nonce	tag	data block	p^a	p^b
Ascon-128	$\text{Ascon}_{12,6}$-128-64	128	128	128	64	12	6
Ascon-128a	$\text{Ascon}_{12,8}$-128-128	128	128	128	128	12	8

1.3 Notation

The following table specifies the notation and symbols used in this document.

$x \in \{0,1\}^k$	Bitstring x of length k (variable if $k = *$)
$0^k, 0^*$	Bitstring of k bits or variable length, 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
$x \oplus y$	Xor of bitstrings x and y
$x \,\Vert\, y$	Concatenation of bitstrings x and y
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
$x_0, \ldots, x_4$	The five 64-bit words of the state S
K, N, T	Secret key K, nonce N, tag T, all of $k \leq 128$ bits
P, C, A	Plaintext P, ciphertext C, associated data A (in blocks P_i, C_i, A_i)
$\perp$	Error, verification of authenticated ciphertext failed
p, p^a, p^b	Permutations p^a, p^b consisting of a, b update rounds p, respectively

1.4 Mode of operation

The mode of operation of Ascon is based on duplex sponge modes like MonkeyDuplex [8], but uses a stronger keyed initialization and keyed finalization function. The core permutations p^a and p^b operate on a sponge state S of size 320 bits, with a rate of r bits and a capacity of $c = 320 - r$ bits. For a more convenient notation, the rate and capacity parts of the state S are denoted by S_r and S_c, respectively. The encryption and decryption operations are illustrated in Figure 1 and Figure 2 and specified in Algorithm 1.

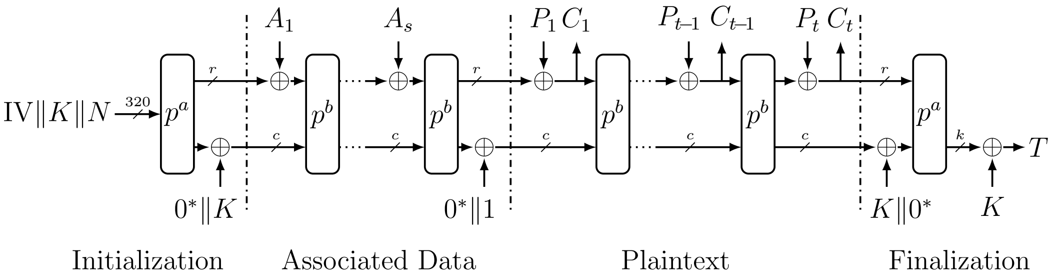

Figure 1: The encryption of Ascon.

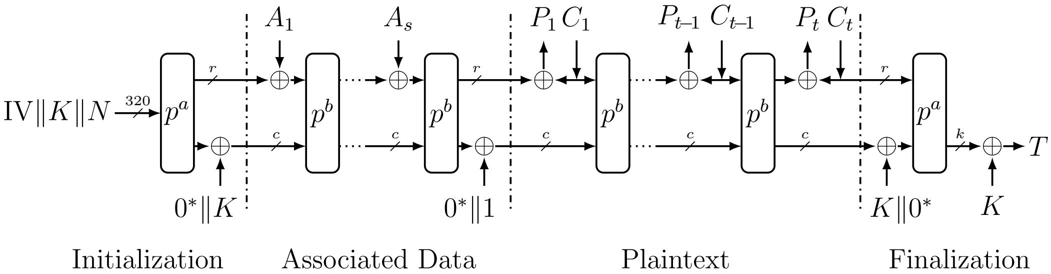

Figure 2: The decryption of Ascon.

Algorithm 1: Authenticated encryption and decryption procedures

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

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

Initialization
$c \leftarrow 320 - r$
$P_1 \ldots P_t \leftarrow \text{pad}_r(P)$
$\ell = |P| \mod r$
$A_1 \ldots A_s \leftarrow \text{pad}_r^*(A)$
$S \leftarrow \text{IV} \| K \| N$
$S \leftarrow p^a(S) \oplus (0^{320-k} \| K)$
Processing Associated Data
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
for $i = 1, \ldots, t-1$ **do**
$S_r \leftarrow S_r \oplus P_i$
$C_i \leftarrow S_r$
$S \leftarrow p^b(S)$
$S_r \leftarrow S_r \oplus P_t$
$C_t \leftarrow \lfloor S_r \rfloor_\ell$
Finalization
$S \leftarrow p^a(S \oplus (0^r \| K \| 0^{c-k}))$
$T \leftarrow \lceil S \rceil^k \oplus K$
return $C_1 \| \ldots \| C_t, T$

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

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

Initialization
$c \leftarrow 320 - r$
$\ell = |C| \mod r$
$A_1 \ldots A_s \leftarrow \text{pad}_r^*(A)$
$S \leftarrow \text{IV} \| K \| N$
$S \leftarrow p^a(S) \oplus (0^{320-k} \| K)$
Processing Associated Data
for $i = 1, \ldots, s$ **do**
$S \leftarrow p^b((S_r \oplus A_i) \| S_c)$
$S \leftarrow S \oplus (0^{319} \| 1)$
Processing Ciphertext
for $i = 1, \ldots, t-1$ **do**
$P_i \leftarrow S_r \oplus C_i$
$S \leftarrow C_i \| S_c$
$S \leftarrow p^b(S)$
$P_t \leftarrow \lfloor S_r \rfloor_\ell \oplus C_t$
$S_r \leftarrow C_t \| (\lceil S_r \rceil^{r-\ell} \oplus (1 \| 0^{r-1-\ell}))$
Finalization
$S \leftarrow p^a(S \oplus (0^r \| K \| 0^{c-k}))$
$T^* \leftarrow \lceil S \rceil^k \oplus K$
if $T = T^*$ **return** $P_1 \| \ldots \| P_t$
else return $\perp$

1.4.1 Padding

Ascon has a message block size 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: $P_1\|...\|P_t$. The same padding process is applied to split the associated data A into s blocks of r bits: $A_1\|...\|A_s$, except if the length of the associated data A is zero. In this case, no padding is applied and no associated data is processed:

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

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

1.4.2 Initialization

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

$$\mathrm{IV} = k \,\|\, r \,\|\, a \,\|\, b \,\|\, 0^{288-2k} = \begin{cases} \texttt{80400c0600000000} & \text{for Ascon-128} \\ \texttt{80800c0800000000} & \text{for Ascon-128a} \end{cases}$$

$$S = \mathrm{IV} \,\|\, K \,\|\, N$$

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:

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

1.4.3 Processing Associated Data

Each (padded) associated data block A_i with $i = 1, \ldots, s$ is processed as follows. The block A_i is xored to the first r bits S_r of the internal state S. Then, the whole state S is transformed by the permutation p^b using b rounds:

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

After the last associated data block has been processed (also if $A = \varnothing$), a single-bit domain separation constant is xored to the internal state S:

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

1.4.4 Processing Plaintext/Ciphertext

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

$$C_i \leftarrow S_r \oplus P_i$$

$$S \leftarrow \begin{cases} p^b(C_i \,\|\, S_c) & \text{if } 1 \leq i < t, \\ C_i \,\|\, S_c & \text{if } 1 \leq i = t. \end{cases}$$

The last ciphertext block is truncated to the unpadded length of the last plaintext block-fragment, $\ell = |P| \mod r$:

$$C_t \leftarrow \lfloor C_t \rfloor_\ell \,.$$

Thus, the length of the last ciphertext block C_t 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.

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 b rounds of the permutation p^b:

$$P_i \leftarrow S_r \oplus C_i$$

$$S \leftarrow p^b(C_i \,\|\, S_c), \qquad 1 \leq i < t$$

For the last, truncated ciphertext block with $0 \leq \ell < r$ bits, the procedure differs slightly:

$$P_t \leftarrow \lfloor S_r \rfloor_\ell \oplus C_t$$

$$S \leftarrow C_t \,\|(\lceil S_r \rceil^{r-\ell} \oplus (1 \,\|\, 0^{r-1-\ell})) \,\|\, S_c$$

The plaintext is returned only if the tag T has been successfully verified in the finalization.

1.4.5 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 k bits of the state xored with the key K:

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

The encryption algorithm returns the tag T together with the ciphertext $C_1, \ldots, C_t$. The decryption algorithm returns the ciphertext $P_1, \ldots, P_t$ only if the calculated tag value matches the received tag value.

1.5 The Permutations

The main components of Ascon are two 320-bit permutations p^a (used in the initialization and finalization) and p^b (used during data processing). The permutations iteratively apply an SPN-based round transformation p that in turn consists of three subtransformations p_C, p_S and 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 for initialization and finalization, and the number of rounds b for intermediate rounds 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,

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

as illustrated in Figure 3.

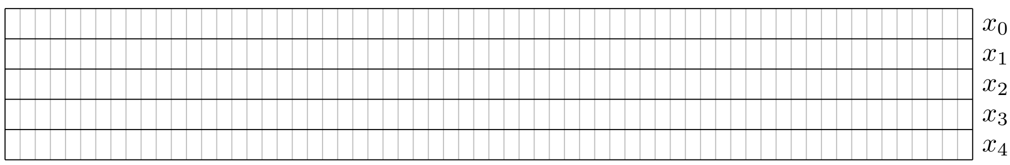

Figure 3: The register word representation of the 320-bit state S.

1.5.1 Addition of Constants

Each round p starts with the constant-addition operation p_C which adds a round constant c_r to the register word x_2 of the state S:

$$x_2 \leftarrow x_2 \oplus c_r$$

The round constant is different for each round; the values for the first round constants as required for the recommended number of rounds are given in Table 2.

Table 2: The round constants used in each round of p^a and p^b.

round	constant	round	constant
0	`0x000000000000000000f0`	6	`0x00000000000000000096`
1	`0x000000000000000000e1`	7	`0x00000000000000000087`
2	`0x000000000000000000d2`	8	`0x00000000000000000078`
3	`0x000000000000000000c3`	9	`0x00000000000000000069`
4	`0x000000000000000000b4`	10	`0x0000000000000000005a`
5	`0x000000000000000000a5`	11	`0x0000000000000000004b`

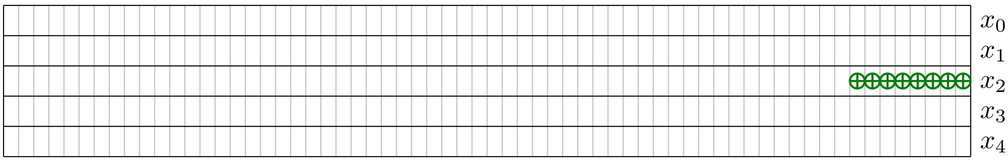

Figure 4: The constants are added to word x_2 of the state.

1.5.2 Substitution Layer

In the substitution layer p_S, 64 parallel applications of the 5-bit S-box $\mathcal{S}(x)$ defined in Table 3 are performed on the 320-bit state. As illustrated in Figure 5, the S-box is applied to each bit-slice of the five registers $x_0, ..., x_4$, where x_0 acts as the MSB and x_4 as the LSB of the S-box.

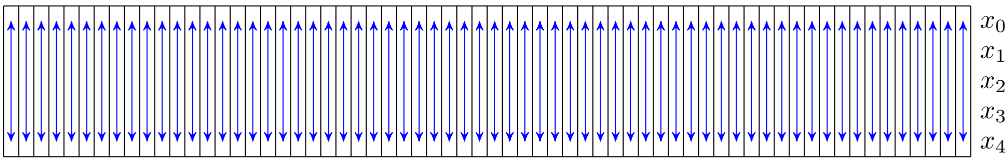

Figure 5: The substitution layer of Ascon applies a 5-bit S-box $\mathcal{S}(x)$ to the state.

Table 3: The 5-bit S-box $\mathcal{S}(x)$ of Ascon.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathcal{S}(x)$	4	11	31	20	26	21	9	2	27	5	8	18	29	3	6	28
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\mathcal{S}(x)$	30	19	7	14	0	13	17	24	16	12	1	25	22	10	15	23

The S-box will typically be implemented in its bitsliced form, with operations performed on the entire 64-bit words. Figure 6 illustrates a bitsliced computation of the S-box values.

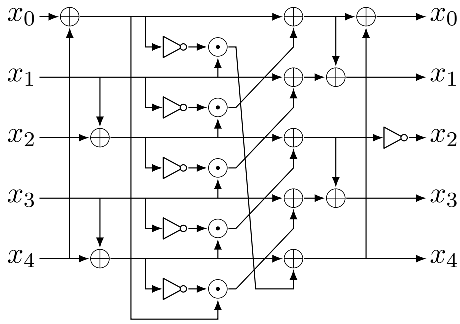

Figure 6: Bitsliced implementation of the 5-bit S-box $\mathcal{S}(x)$.

This sequence of bitsliced instructions is well-suited for pipelining, as the following implementation with five temporary registers $t_0, \dots, t_4$ shows:

```
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 7: Pipelinable instructions for the 5-bit S-box $\mathcal{S}(x)$.

1.5.3 Linear Diffusion Layer

The linear diffusion layer p_L of Ascon is used to provide diffusion within each of the five 64-bit register words x_i of the 320-bit state S, as illustrated in Figure 8. We apply a linear function $\Sigma_0(x_0), \dots, \Sigma_4(x_4)$ to each word x_i separately,

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

where the functions Σ_i are defined as follows:

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

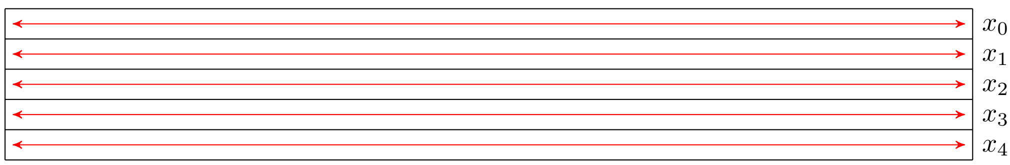

Figure 8: The linear diffusion layer of Ascon mixes bits within words using $\Sigma_i(x_i)$.

Chapter 2

Security Claims

Table 4: Security claims for recommended parameter configurations of Ascon.

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

There is no secret message number. The public message number is a nonce, i.e., the security claims are void if two plaintexts are encrypted under the same key and the same public message number. In particular, reusing the nonce for two messages allows to detect plaintexts with common prefixes and to deduce the xor difference of the first block pair that differs between the two messages. Except for the single-use requirement, there are no constraints on the choice of message numbers.

The decryption algorithm may only release the decrypted plaintext after verification of the final tag. Similar to GCM, a system or protocol implementing the algorithm should monitor and, if necessary, limit 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.

The number of processed plaintext and associated data blocks protected by the encryption algorithm is limited to 2^{64} blocks per key. This requirement also imposes a message length limit of 2^{64} blocks, which corresponds to 2^{67} (Ascon-128) or 2^{68} (Ascon-128a) bytes (for plaintext and associated data).

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 automatically afflict the claimed security properties of the entire encryption algorithm.

Chapter 3

Security Analysis

3.1 Basic Properties

In this section, we give some known properties of the S-box used in Ascon. Table 9 in Appendix A shows the differential probabilities corresponding to input and output differences. As can be seen in the table, the maximum differential probability of the S-box is 2^{-2} and its differential branch number is 3. Table 10 shows the biases of the linear approximation defined by corresponding input and output masks. The maximum linear probability of the S-box is 2^{-2} and its linear branch number is 3.

Let x_0, x_1, x_2, x_3, x_4 and y_0, y_1, y_2, y_3, y_4 be the 5-bit input and output of the S-box, where x_0 refers to the most significant bit or the first register word of the S-box. Then the algebraic normal form (ANF) of the S-box is given by:

$$\begin{aligned}
y_0 &= x_4x_1 + x_3 + x_2x_1 + x_2 + x_1x_0 + x_1 + x_0, \\
y_1 &= x_4 + x_3x_2 + x_3x_1 + x_3 + x_2x_1 + x_2 + x_1 + x_0, \\
y_2 &= x_4x_3 + x_4 + x_2 + x_1 + 1, \\
y_3 &= x_4x_0 + x_4 + x_3x_0 + x_3 + x_2 + x_1 + x_0, \\
y_4 &= x_4x_1 + x_4 + x_3 + x_1x_0 + x_1.
\end{aligned}$$

Note that the number of monomials which appear in the polynomial representation is smaller than that of a randomly generated S-box and the algebraic degree is 2. Though one might claim that this S-box is weak in terms of algebraic attacks, we have not found any practical attack on Ascon using these properties.

However, it should be remarked that the low algebraic degree of the S-box and the small number of rounds of p^a and p^b results in rather efficient zero-sum distinguishers [9] for the two permutations. Hence, the two permutations cannot be considered as perfect random permutations.

3.2 Differential and Linear Propagation

In this section, we will discuss the security of Ascon against differential and linear cryptanalysis. It is easy to see that the branch number of Σ_i is only 4 and that this alone might not be enough to get good bounds against differential and linear attacks in Ascon. However, in combination with the S-box, which has branch number 3, and the fact that different rotation values are used in all the Σ_i, the number of active S-boxes is increased significantly. We have confirmed that the minimum number of active S-boxes of 3 rounds is at least 15 and 13 for any differential and linear trail.

For results on more than 3 rounds, we used a heuristic search tool to find good differential and linear trails for more rounds to get close to the real bound. The results are listed in

Table 5. The best truncated differential and linear trails for 4 rounds is given in Table 6a and Table 6b, respectively. We want to note that we could not find any differential and linear trails for more than 4 rounds with less than 64 active S-boxes.

Table 5: Number of active S-boxes for up to 4 rounds of p (* from heuristic search).

rounds	# active S-boxes	
	differential	linear
1	1	1
2	4	4
3	15	13
4	*44	*43

Table 6: The best known trails for 4 rounds of p (in truncated notation).

(a) Differential 4-round trail

Round	Truncated trail	# active S-boxes
0	`b008db32a11104c9`	23
1	`0000010000201000`	3
2	`0001010000000004`	3
3	`880909022a100226`	15
total		44

(b) Linear 4-round trail

Round	Truncated trail	# active S-boxes
0	`0014342c0c091210`	15
1	`0000000808000200`	3
2	`8040000800000000`	3
3	`2fc00008218a7a39`	22
total		43

3.3 Collision-Producing Differential

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 truncated collision-producing differential trails we could find for p^b in Ascon-128 and Ascon-128a using a heuristic search algorithm have 117 and 192 active S-boxes, respectively. The truncated differential trails are given in Tables 7a and 7b.

Table 7: Collision-producing differential trails for Ascon (in truncated notation).

(a) 6-round trail for Ascon-128

Round	Truncated trail	# active S-boxes
0	8000000000000000	1
1	8100000001400004	5
2	9902a00003c64086	17
3	fcf7eee14feefdf7	48
4	dba6fe7b4fef8cef	45
5	0000400000000000	1
total		117

(b) 8-round trail for Ascon-128a

Round	Truncated trail	# active S-boxes
0	8000000000000000	1
1	c200000000000000	3
2	e238e10000000000	11
3	73b7fbf67f6f19f0	44
4	bb4ffe8fd5dddf7f	48
5	fffffdffffffffff	63
6	2d0486c240902436	20
7	2080000000000000	2
total		192

3.4 Impossible Differentials

In this section, we will discuss the application of impossible differential cryptanalysis to Ascon. Using an automated search tool, we were able to find impossible differentials for up to 5 rounds of the permutation and it is likely that impossible differentials for more rounds exist. However, we have not found any practical attack on Ascon using this property of the permutation. An impossible differential for 5 rounds of the permutation is given in Table 8.

Table 8: Impossible differential for Ascon, covering 5 rounds of p.

	input differential		output differential after 5 rounds
x_0	0000000000000000		0000000000100000
x_1	0000000000000000		0000000000000000
x_2	0000000000000000	$\rightarrow$	0000000000000000
x_3	0000000000000000		0000000000000000
x_4	8000000000000000		0000000000000000

Chapter 4

Features

The main feature of Ascon is its lightweight implementation characteristics in both hardware and software while still being reasonably fast. In particular, Ascon was designed to allow efficient implementation of side-channel resistance features. Ascon is not intended to compete with very fast parallel authenticated encryption schemes on unconstrained devices. However, Ascon has been designed to use a minimum number of instructions while still maximizing the parallelism of these instructions. Therefore, Ascon is best used where size and implementation security matters but reasonable performance is also required.

The Ascon cipher is online and can encrypt plaintext blocks before subsequent plaintexts or the plaintext length are known. The same holds for the decryption, which decrypts the ciphertext blocks online in the order they were computed during encryption. However, during decryption, the plaintexts must not be released until the tag has been verified. The cipher does not need to implement any inverse operations and decryption is equally fast as encryption.

Since Ascon uses many well-studied components such as the sponge construction and an SPN-based permutation, it is easy to analyze. Furthermore, it provides strong security arguments and bounds for the linear and differential probability to exclude certain classes of attacks.

Additionally, Ascon can be implemented efficiently on platforms and applications where side-channel resistance is important. The very efficient bitsliced implementation of the S-boxes prevents cache-timing attacks, since no look-up tables are required. The low algebraic degree of the S-box facilitates first-order masking or sharing-based side-channel countermeasures such as threshold implementations [13], which have previously been applied to the S-box of Keccak [2].

The internal permutation is based on very simple operations that are easy and efficient to implement both in hardware and in software, in particular on processors using the modern standard word size of 64 bits. All required steps are intuitively defined in terms of simple word-wise (64-bit) standard operations, which significantly reduces the effort of implementing the algorithms on new target platforms. The 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 bitsliced implementations.

The ciphertext size for Ascon in bits is exactly the same as for the (unpadded) plaintext size, thus allowing the encryption of short messages with very little transmission overhead. On the other hand, like many sponge constructions, such as the MonkeyDuplex construction, Ascon uses only a relatively weak intermediate permutation for each additional plaintext block, which is beneficial for the performance for long multi-block plaintexts.

The default recommended version of Ascon-128 uses a key, nonce and tag size of 128 bits and a rate of 64 bits. It is designed to provide more than adequate security and reasonable performance characteristics for a variety of applications. For increased performance, Ascon-128a can be used, which allows to process blocks of twice the size with only a slightly higher number of rounds in the intermediate permutations.

Compared to AES-GCM, the advantages of Ascon are its relatively small state size of 320 bits, its low area in hardware and less overhead to provide side-channel resistant implementations. In general, Ascon is significantly easier to implement from scratch than AES-GCM in both hardware and software. The disadvantages of Ascon compared to AES-GCM are that Ascon is not parallelizable (on a message block level) and, since it is a dedicated design, cannot profit from existing high-performance implementations of AES such as Intel's AES-NI instruction set.

Chapter 5

Design Rationale

The main goal of Ascon is a very low memory footprint in hardware and software, while still being fast and providing a simple analysis and good bounds for the security. 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.

Ascon is based on the sponge design methodology [3]. 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 improved version of the S-box used in the χ mapping of Keccak [5]. The permutation layer uses linear functions similar to the Σ functions used in SHA-2. Details on the design principles for each component are given in the following sections.

5.1 Choice of the Mode

The design principles of Ascon follow the sponge construction [3], to be more precise, they are very similar to SpongeWrap [4] and MonkeyDuplex [8]. The sponge-based design has several advantages compared to other available construction methods like some block cipher- or hash function-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) or to designs that are nonce-reuse resistant and secure under release-of-unverified-plaintext.
- Elegant and simple design, obvious 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 round permutation as encryption.
- Weak round transformations can be used to process additional plaintext blocks, improving the performance for long messages.

Compared to other sponge-based designs, Ascon uses a stronger keyed initialization and keyed finalization phase. The result is that even an entire state recovery is not sufficient to recover the secret key or to allow universal forgery.

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 are present, there is no additional intermediate round transformation p^b, only the initialization and finalization calls p^a. To prevent that key additions between the two applications of p^a cancel each other out, they are added to different parts of the capacity part S_c of the state.

5.2 Choice of the Round Constants

The round constants have been chosen large enough to avoid slide, rotational, self-similarity or other 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. In addition, 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).

Similar to the round constants, the initialization vector is forced to be asymmetric in each word by including the parameters k, r, a, b in fixed positions and fixed 0 bits in others. This inclusion of the parameters, in particular b and r, also serves to distinguish the different members of the Ascon family.

5.3 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 and linear probability 1/4,
- Differential and linear branch number 3,
- Avoid trivially iterable differential properties in the message injection positions.

The χ mapping of Keccak fulfills several of the aforementioned 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 look-up 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 [13] and masking, which is 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 [6] 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.

5.4 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 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 [7], Hamsi [11], other SPN-based designs. However, these candidates were either too slow or provide a less optimal diffusion.

The rotation values of the linear diffusion layer in Ascon are chosen similar to those of Σ in SHA-2 [12]. 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 of 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.

5.5 Statement

The designers have not hidden any weaknesses in this cipher.

Chapter 6

Intellectual Property

The submitters are not aware of any patent involved in Ascon, and it will not be patented. If any of this information changes, the submitters will promptly (and within at most one month) announce these changes on the crypto-competitions mailing list.

Chapter 7

Consent

The submitters hereby consent to all decisions of the CAESAR selection committee regarding the selection or non-selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitters understand that the committee will not comment on the algorithms, except that for each selected algorithm the committee will simply cite the previously published analyses that led to the selection of the algorithm. The submitters understand that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitters acknowledge that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitters understand that if they disagree with published analyses then they are expected to promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

Acknowledgments

The authors would like to thank all researchers contributing to the design, analysis and implementation of ASCON. The work has been supported in part by the Austrian Science Fund (project P26494-N15), in part by the Austrian Government through the research program COMET (Project SeCoS, Project Number 836628) and through the research program FIT-IT Trust in IT Systems (Project SePAG, Project Number 835919).

Bibliography

[1] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of keyed sponge constructions using a modular proof approach. In Gregor Leander, editor, *FSE*, volume 9054 of *LNCS*, pages 364–384. Springer, 2015.

[2] Guido Bertoni, Joan Daemen, Nicolas Debande, Thanh-Ha Le, Michael Peeters, and Gilles Van Assche. Power analysis of hardware implementations protected with secret sharing. In *MICRO Workshops*, pages 9–16. IEEE Computer Society, 2012.

[3] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Sponge Functions. ECRYPT Hash Workshop 2007, May 2007.

[4] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In Ali Miri and Serge Vaudenay, editors, *SAC*, volume 7118 of *LNCS*, pages 320–337. Springer, 2011.

[5] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Keccak specifications. Submission to NIST (Round 3), 2011.

[6] Begül Bilgin, Andrey Bogdanov, Miroslav Knezevic, Florian Mendel, and Qingju Wang. Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES 2013*, volume 8086 of *LNCS*, pages 142–158. Springer, 2013.

[7] Christophe De Cannière, Hisayoshi Sato, and Dai Watanabe. Hash Function Luffa: Specification. Submission to NIST (Round 2), 2009.

[8] Joan Daemen. Permutation-based Encryption, Authentication and Authenticated Encryption. DIAC – Directions in Authenticated Ciphers, July 2012.

[9] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Cryptanalysis of ascon. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 371–387. Springer, 2015.

[10] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT*, volume 8873 of *LNCS*, pages 85–104. Springer, 2014.

[11] Özgül Küçük. The Hash Function Hamsi. Submission to NIST (Round 2), 2009.

[12] 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.

[13] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011.

Appendix A

S-box distribution tables

A.1 Differential distribution table

Table 9: The differential profile of the Ascon S-box.

	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	·	·	·	·	·	·	·	·	·	·

A.2 Linear distribution table

Table 10: The linear profile of the Ascon S-box.

	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	·	·	·

Appendix B

Changelog

B.1 Changes from v1 to v1.1

We detail here the differences made between v1 of this document (as submitted to CAESAR round 1) and the current v1.1 (as submitted to CAESAR round 2).

Functional changes (tweak)

- Modification of secondary recommendation Ascon-96:

  Change: The key size and security claim for Ascon-96 was increased from 96 bits to 128 bits, and Ascon-96 consequently renamed to Ascon-128a.

  Justification: With this change, we take advantage of recent results on beyond-$c/2$ security of sponge modes, in particular of the proofs presented at ASIACRYPT 2014 by Jovanovic et al. [10] and at FSE 2015 by Andreeva et al. [1]. These results allow to benefit from the doubled rate of Ascon-96 (128 bits, with 8-round permutation) compared to Ascon-128 (64 bits, with 6-round permutation), without having to decrease the security level for the smaller capacity.

Document updates

- Added cryptanalysis results published at CT-RSA 2015 [9] to Section 3.
- Figures 1, 2 and 6 updated for clarity wrt. Ascon-128a.
- Typos and minor inconsistencies corrected.