Ascon v1.2

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, 8010 Graz, Austria

Infineon Technologies Austria AG
Babenbergerstraße 10, 8020 Graz, Austria

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

September 15, 2016

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

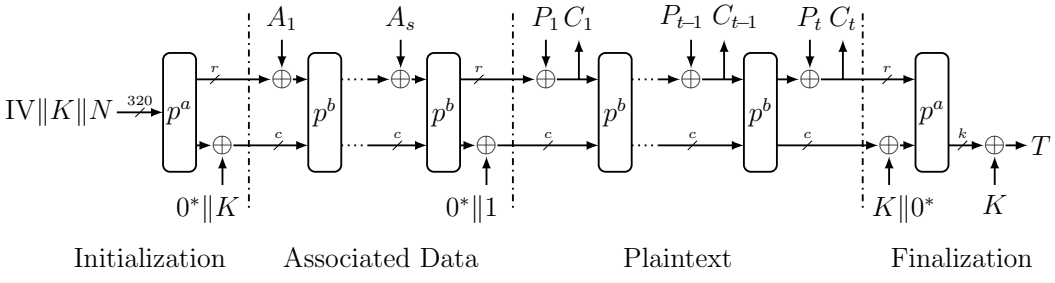
1.3 Notation

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

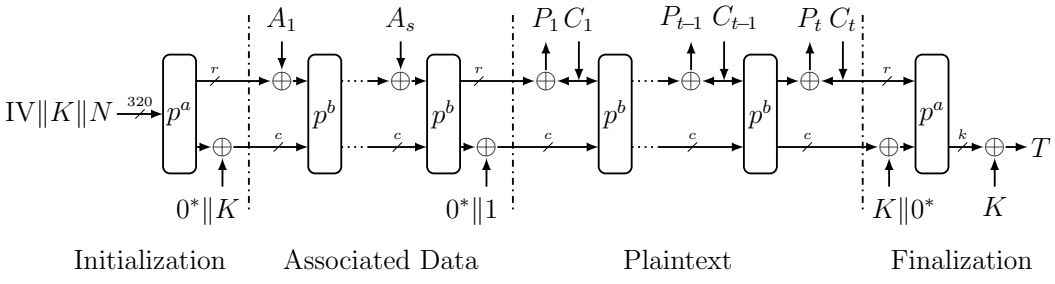
$x \in \{0,1\}^k$	Bitstring x of length k (variable if $k = *$)
$0^k, 0^*$	Bitstring of k bits or variable length, all 0
$\lvert x \rvert$	Length of the bitstring x in bits
$\lfloor x \rfloor_k$	Bitstring x truncated to the first (most significant) k bits
$\lceil x \rceil^k$	Bitstring x truncated to the last (least significant) k bits
$x \oplus y$	Xor of bitstrings x and y
$x \,\|\, y$	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 [13], 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 1a and Figure 1b and specified in Algorithm 1.



(a) Encryption



(b) Decryption

Figure 1: Ascon's mode of operation.

Algorithm 1: Authenticated encryption and decryption procedures

Authenticated Encryption $\mathcal{E}_{a,b,k,r}(K, N, A, P)$	Verified Decryption $\mathcal{D}_{a,b,k,r}(K, N, A, C, T)$
Input: key $K \in \{0,1\}^k$, $k \leq 128$, nonce $N \in \{0,1\}^k$, plaintext $P \in \{0,1\}^*$, associated data $A \in \{0,1\}^*$	**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: ciphertext $C \in \{0,1\}^*$, tag $T \in \{0,1\}^k$	**Output:** plaintext $P \in \{0,1\}^*$ or $\perp$
Initialization $c \leftarrow 320 - r$ $P_1 \dots P_t \leftarrow \text{pad}_r(P)$ $\ell = \lvert P \rvert \mod r$ $A_1 \dots A_s \leftarrow \text{pad}_r^*(A)$ $S \leftarrow \text{IV} \,\|\, K \,\|\, N$ $S \leftarrow p^a(S) \oplus (0^{320-k} \,\|\, K)$	**Initialization** $c \leftarrow 320 - r$ $\ell = \lvert C \rvert \mod r$ $A_1 \dots A_s \leftarrow \text{pad}_r^*(A)$ $S \leftarrow \text{IV} \,\|\, K \,\|\, N$ $S \leftarrow p^a(S) \oplus (0^{320-k} \,\|\, K)$
Processing Associated Data **for** $i = 1, \dots, s$ **do** $S \leftarrow p^b((S_r \oplus A_i) \,\|\, S_c)$ $S \leftarrow S \oplus (0^{319} \,\|\, 1)$	**Processing Associated Data** **for** $i = 1, \dots, s$ **do** $S \leftarrow p^b((S_r \oplus A_i) \,\|\, S_c)$ $S \leftarrow S \oplus (0^{319} \,\|\, 1)$
Processing Plaintext **for** $i = 1, \dots, t-1$ **do** $S_r \leftarrow S_r \oplus P_i$ $C_i \leftarrow S_r$ $S \leftarrow p^b(S)$ $S_r \leftarrow S_r \oplus P_t$ $C_t \leftarrow \lfloor S_r \rfloor_\ell$	**Processing Ciphertext** **for** $i = 1, \dots, t-1$ **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 $S \leftarrow p^a(S \oplus (0^r \,\|\, K \,\|\, 0^{c-k}))$ $T \leftarrow \lceil S \rceil^k \oplus K$ **return** $C_1 \,\|\, \dots \,\|\, C_t, T$	**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** $P_1 \,\|\, \dots \,\|\, P_t$ **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:

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

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

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:

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

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:

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

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:

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

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

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

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:

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

The encryption algorithm returns the tag T together with the ciphertext $C_1, \ldots, C_t$. The decryption algorithm returns the plaintext $P_1, \ldots, P_t$ 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, as illustrated in Figure 2:

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

1.5.1 Addition of Constants

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

$$x_2 \leftarrow x_2 \oplus c_r$$

Ascon uses the round constants c_r for p^a and c_{a-b+r} for p^b. The values for the first round constants as required for the recommended number of rounds are given in Table 2.

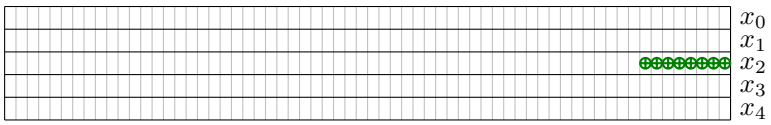


Figure 2: The register words of the 320-bit state S, and position of the constant addition.

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

p^{12}	p^8	p^6	Constant	p^{12}	p^8	p^6	Constant
0			00000000000000000f0	6	2	0	0000000000000000096
1			00000000000000000e1	7	3	1	0000000000000000087
2			00000000000000000d2	8	4	2	0000000000000000078
3			00000000000000000c3	9	5	3	0000000000000000069
4	0		00000000000000000b4	10	6	4	000000000000000005a
5	1		00000000000000000a5	11	7	5	000000000000000004b

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 3, 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.

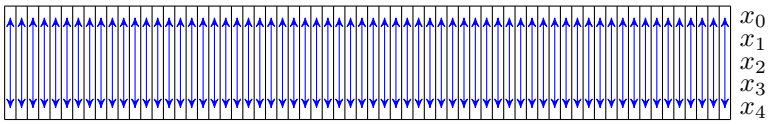


Figure 3: The substitution layer of Ascon applies a 5-bit S-box $\mathcal{S}(x)$ to the state.

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 4 illustrates a bitsliced computation of the S-box values. This sequence of bitsliced instructions is well-suited for pipelining, as the implementation with five temporary registers $t_0, \ldots, t_4$ in Figure 5 shows.

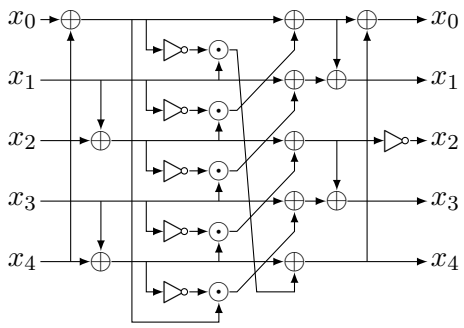


Figure 4: Bitsliced implementation of the 5-bit S-box $\mathcal{S}(x)$.

```
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 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 6. We apply a linear function $\Sigma_0(x_0), \ldots, \Sigma_4(x_4)$ to each word x_i separately,

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

where the functions Σ_i are defined as follows:

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

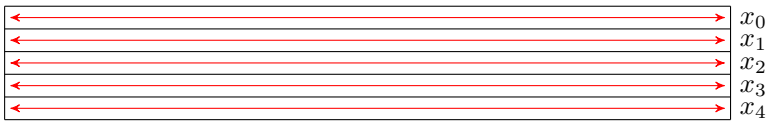


Figure 6: The linear diffusion layer of Ascon mixes bits within words using $\Sigma_i(x_i)$.

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

Features

Several features of the Ascon design help to allow its lightweight implementation characteristics in both hardware and software while still offering good performance on both platforms. 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 good performance is also required.

3.1 Properties of Ascon

- **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 trimmed towards providing either a smaller area (e.g., 2.5 kGE [17]) or higher speed (e.g., 13.2 Gbps [17]).

- **Bitsliced in Software.** Ascon is designed to facilitate bitsliced software implementations. The internal permutation is based on very simple operations that are intuitively defined in terms of simple word-wise (64-bit) standard operations. These 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. Up to 5 instructions can be carried out in parallel in nearly every phase of the permutation. In addition, the rather small state of Ascon allows to hold the whole state within the CPU's registers for a wide range of platforms reducing reloads from the cache to a minimum.

- **Easy Integration of Side-Channel Countermeasures.** 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. Furthermore, the low algebraic degree of the S-box facilitates first- *and* higher-order protection using masking or sharing-based side-channel countermeasures such as threshold implementations [22] and consolidated masking schemes [12, 23]. These countermeasures have been applied to the S-box of Keccak in [5, 10, 20] and similar results can be expected for Ascon.

- **Balanced Design.** While there is arguably a need for ciphers designed for corner cases, such as extremely lightweight designs with very low area footprints, or ciphers designed to provide very high speed on specific software platforms, we follow a more balanced design approach. In particular, Ascon has been designed to provide a lightweight implementation characteristics in both hardware and software while still offering good 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).
- **Online.** The Ascon cipher is online and can encrypt plaintext blocks before subsequent plaintexts or the plaintext length are known. The same holds for decryption, which decrypts ciphertext blocks online in the order they were computed during encryption.
- **Single-Pass.** For both encryption and decryption, just one pass over the data is required.
- **Inverse-Free.** Ascon does not need to implement any inverse operations. In other words, the permutations p^a and p^b are only evaluated in one direction for both encryption and decryption, which significantly reduces the area overhead for implementations.
- **High Key Agility.** Ascon neither needs a key schedule, nor expands the key by any other means. Therefore, there are no hidden setup costs when the key is changed.
- **Simplicity.** Ascon is intuitively defined on 64-bit words using only the common bitwise Boolean functions AND, OR, XOR, NOT, and ROT (bitwise rotation). This significantly reduces the effort of implementing the algorithm on new target platforms.
- **Robustness.** Ascon is a nonce-based scheme. As with any authenticated encryption scheme, repeating nonces is a misuse setting, and implies a loss of semantic security. Since Ascon is based on the MonkeyDuplex construction [13], it inherits many of its properties in misuse settings. For example, repeating the nonce once for two different messages reveals the difference between the first differing plaintext blocks. If nonces are reused excessively or many plaintexts for unverified tags are released, structural attacks may lead to recovery of the internal state. However, 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.

3.2 CAESAR Use Cases

At the end of Round 2, the committee identified three use cases for authenticated encryption schemes [24]: (1) lightweight applications, (2) high-performance applications, and (3) defense in depth in misuse scenarios. Below, we restate the relevant profile for these use cases, and discuss which use cases Ascon is most suitable for.

Ascon follows a balanced design approach, instead of optimizing for only one particular platform or use case. For this reason, Ascon satisfies the requirements for both use cases 1 and 2, and provides a number of additional features that support efficient and secure implementations in a variety of applications. For instance, Ascon is designed for small implementations in hardware that can be easily protected against side-channel attacks, which is essential for use case 1. At the same time, Ascon can be implemented efficiently in software, especially on 64-bit platforms, providing constant-time bitsliced implementations as required by use case 2. We believe that ciphers which operate efficiently and securely in

both of these use cases 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. We justify the suitability of Ascon for these use cases in detail below.

In summary, both Ascon-128 and Ascon-128a are recommended for use cases 1 and 2, in this order. Since the performance constraints of use case 1 are often more critical than those of use case 2, this is the primary focus of our design. Ascon is not intended as a misuse-resistant design for use case 3, but its novel mode of operation nevertheless offers some robustness in misuse settings.

Table 5: Recommended use cases for Ascon.

Algorithm	Security level	Use cases
Ascon-128	128-bit	1 and 2
Ascon-128a	128-bit	1 and 2

Use Case 1: Lightweight Applications

This is the primary recommended use case of Ascon. The following properties qualify Ascon 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 are as small as 2.6 kGE [17]. The round-based implementations of both Ascon variants 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.

- **Efficiency in hardware.** Ascon is not only small and fast, but can also be efficiently implemented on a wide variety of platforms [16]. It allows many trade-offs between throughput, latency, gate count, power consumption, etc. [17]. Comparison of implementation results in [16] show that throughput per area of both Ascon variants is very good compared to many other CAESAR candidates.

- **Natural side-channel protection.** This is one of the primary design goals of Ascon. For protected hardware implementations, 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. Also, since Ascon uses Keccak's S-box core, it benefits from existing efficient protected implementations [5]. Several threshold implementations of Ascon have already been proposed, ranging from small (less than 8 kGE [17]) to high-speed (9 Gbps [17]) use cases.

- **Limited damage in misuse settings.** Ascon is a nonce-based scheme. As with any 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 [1].

- **Low overhead for short messages.** Ascon is among the fastest CAESAR candidates for short messages according to current software benchmarking results [3, 4], since its initialization and finalization overhead is much smaller compared to most block cipher 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 these applications.

Use Case 2: High-Performance Applications

This is the secondary recommended use case of Ascon.

- **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.
- **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 look-ups 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.

Use Case 3: Defense in Depth

Although this is not a recommended use case for Ascon, it nevertheless offers some robustness in misuse settings as discussed in Section 3.1.

While Ascon should never be used without fresh nonces, and unverified plaintexts should never be released, we have added countermeasures to limit the damage in such scenarios. Compared to other sponge-based designs, Ascon uses a keyed initialization and a keyed finalization. Hence, a potential recovery of the internal state during data processing triggered by a nonce misuse or release of unverified plaintext neither leads to a recovery of the secret key, nor allows universal forgery. However, in practical settings, if the nonce is only reused very few times and keys are exchanged after a few authentication failures, we expect state recoveries to be infeasible.

3.3 Comparison with AES-GCM

Compared to AES-GCM, the advantages of Ascon are its small state size of 320 bits, its low area in hardware implementations, and significantly less overhead to provide side-channel resistant implementations. In general, Ascon is significantly easier to implement securely 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 4

Security Analysis

4.1 Basic Properties

In this section, we give some known properties of the S-box used in Ascon. Table 10 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 11 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 [15] for the two permutations. Hence, the two permutations cannot be considered as perfect random permutations.

4.2 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 [14] to find good differential and linear trails for more rounds to get close to the real bound. The results are

listed in Table 6. The best truncated differential and linear trails for 4 rounds is given in Table 7a and Table 7b, 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 6: 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 7: 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

4.3 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 8a and 8b.

Table 8: 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

4.4 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 9.

Table 9: 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 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 [6]. 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 [8]. 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 [6], to be more precise, they are very similar to SpongeWrap [7] and MonkeyDuplex [13]. 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 [22] 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 [9] 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 [11], Hamsi [19], 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 [21]. 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] Megha Agrawal, Donghoon Chang, and Somitra Sanadhya. sp-AELM: Sponge based authenticated encryption scheme for memory constrained devices. In Ernest Foo and Douglas Stebila, editors, *Information Security and Privacy – ACISP 2015*, volume 9144 of *LNCS*, pages 451–468. Springer, 2015.

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

[3] Ralph Ankele and Robin Ankele. Software benchmarking of the 2nd round CAESAR candidates. Cryptology ePrint Archive, Report 2016/740, 2016.

[4] Daniel J. Bernstein and Tanja Lange (editors). eBACS: ECRYPT Benchmarking of Cryptographic Systems. https://bench.cr.yp.to. (accessed 15 September 2016).

[5] Guido Bertoni, Joan Daemen, Nicolas Debande, Thanh-Ha Le, Michaël Peeters, and Gilles Van Assche. Power analysis of hardware implementations protected with secret sharing. In *IEEE/ACM International Symposium on Microarchitecture – MICRO 2012*, pages 9–16. IEEE Computer Society, 2012.

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

[7] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography – SAC 2011*, volume 7118 of *LNCS*, pages 320–337. Springer, 2011.

[8] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Keccak specifications. Submission to NIST (Round 3), 2011. http://keccak.noekeon.org.

[9] 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, *Cryptographic Hardware and Embedded Systems – CHES 2013*, volume 8086 of *LNCS*, pages 142–158. Springer, 2013.

[10] Begül Bilgin, Joan Daemen, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Gilles Van Assche. Efficient and first-order DPA resistant implementations of keccak. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications – CARDIS 2013*, volume 8419 of *LNCS*, pages 187–199. Springer, 2013.

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

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

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

[14] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Heuristic tool for linear cryptanalysis with applications to CAESAR candidates. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 490–509. Springer, 2015.

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

[16] Kris Gaj and ATHENa Team. ATHENa: Automated Tool for Hardware Evaluation, 2016. https://cryptography.gmu.edu/athena/.

[17] Hannes Groß, Erich Wenger, Christoph Dobraunig, and Christoph Ehrenhofer. Suit up! - Made-to-measure hardware implementations of ASCON. In *Digital System Design – DSD 2015*, pages 645–652. IEEE Computer Society, 2015.

[18] 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, *Advances in Cryptology – ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 85–104. Springer, 2014.

[19] Özgül Küçük. The Hash Function Hamsi. Submission to NIST (Round 2), 2009. http://homes.esat.kuleuven.be/~okucuk/hamsi/.

[20] Michael Muehlberghuber, Thomas Korak, Philipp Dunst, and Michael Hutter. Towards evaluating DPA countermeasures for keccak K1012ECCAK on a real ASIC. In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design – COSADE 2015*, volume 9064 of *LNCS*, pages 222–236. Springer, 2015.

[21] 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. http://csrc.nist.gov/publications/fips/fips180-3/fips-180-3_final.pdf.

[22] 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.

[23] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, volume 9215 of *LNCS*, pages 764–783. Springer, 2015.

[24] The CAESAR Committee. CAESAR Use Cases. Cryptographic competitions mailing list, 2016. https://groups.google.com/forum/#!topic/crypto-competitions/DLv193SPSDc.

Appendix A

S-box distribution tables

A.1 Differential distribution table

Table 10: 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 11: 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 (Round 1) to v1.1 (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. [18] and at FSE 2015 by Andreeva et al. [2]. 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 [15] to Section 4.
- Figures 1a, 1b and 4 updated for clarity wrt. Ascon-128a.
- Typos and minor inconsistencies corrected.

B.2 Changes from v1.1 (Round 2) to v1.2 (Round 3)

Functional changes (tweak)

- Modification of the round constant schedule:

  Change: p^b now uses round constants $c_{a-b}, \ldots, c_{a-1}$ instead of $c_0, \ldots, c_{b-1}$ of p^a (Table 2).

  Justification: To increase compatibility of the permutations with other sponge modes, such as FIPS 202. This change is not expected to have any effects on the security analysis or performance of Ascon.

Document updates

- Rewrote Features section to reflect the CAESAR use cases, and moved this section before Security Analysis.
- Added references to recent analysis and implementations.
- Minor editorial updates.