# Ascon PRF, MAC, and Short-Input MAC

Christoph Dobraunig[3,4], Maria Eichlseder[1], Florian Mendel[2], and Martin Schläffer[2]

[1]Graz University of Technology, Austria
[2]Infineon Technologies AG, Germany
[3]Intel Labs, USA
[4]Lamarr Security Research, Austria

https://ascon.iaik.tugraz.at

The cipher suite Ascon v1.2 already provides authenticated encryption schemes, hash, and extendable output functions. Furthermore, the underlying permutation is also used in two instances of Isap v2.0, an authenticated encryption scheme designed to provide enhanced robustness against side-channel and fault attacks. In this paper, we enrich the functionality one can get out of Ascon's permutation by providing efficient Pseudorandom Functions (PRFs), a Message Authentication Code (MAC) and a fast short-input PRF for messages up to 128 bits.

**Keywords:** Pseudorandom function · MAC · Ascon.

## 1 Introduction

The Ascon family of authenticated encryption schemes [DEMS14] was first published in the beginning of 2014 as a submission to the CAESAR Competition [Cae14]. After 5 years of public scrutiny, the authenticated encryption schemes Ascon-128 and Ascon-128a (v1.2) [DEMS16] were recommended as the first choice for lightweight applications in the final portfolio of CAESAR for resource-constrained environments. Furthermore, the cipher suite Ascon v1.2 [DEMS21a; DEMS21b] containing Ascon-128 and Ascon-128a, as well as the hash function Ascon-Hash and extendable output function Ascon-Xof, are finalists in the NIST lightweight cryptography (LWC) standardization process [Nat18]. Ascon's permutation also serves as a basis for two instances of Isap v2.0 [DEM+20; DEM+21], an authenticated encryption scheme designed to provide enhanced robustness against side-channel and fault attacks on algorithmic level. Isap v2.0 is also a NIST LWC finalist.

In this paper, we define two lightweight and efficient pseudorandom function (PRF) families, Ascon-Prf and Ascon-PrfShort. Ascon-Prf processes inputs of arbitrary length and produces outputs of length up to $2^{32}$ bits. In contrast, Ascon-PrfShort operates only on short inputs $\leq 128$ bits producing outputs of short length $\leq 128$ bits. Ascon-Prf and Ascon-PrfShort are adaptions of the full-keyed sponge (FKS) mode [BDPV07; BDPV12; MRV15; DMV17]. Ascon-Prf uses a rate of 256 bits during absorption and a rate of 128 bits in the squeezing phase. Overall, Ascon-Prf is an efficient choice for general-purpose lightweight message authentication, so we define the corresponding message authentication code (MAC) Ascon-Mac based on Ascon-Prf. Ascon-PrfShort excels whenever short data needs to be authenticated, e.g., the authentication of pointers, in challenge-response protocols, or protocols that derive symmetric keys of entities from a master key.

# 2 Specification

In this section, we introduce the state layout and notation for our functions and specify the modes of operation for the PRFs and the MAC.

## 2.1 State and Notation

Our algorithms operate on a 320-bit state $S$ which is updated using the $a$-round permutation $p^a$. The state $S$ is divided into an outer part of $r$ bits and an inner part of $c$ bits, where the rate $r$ and capacity $c = 320 - r$ depend on the variant.

For the description and application of the round transformations (Section 3), the 320-bit state $S$ is split into five 64-bit words $x_i$, as illustrated in Figure 3a:

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

Whenever $S$ needs to be interpreted as a byte-array (or bitstring) used in the sponge interface, the array starts with the most significant byte (or bit) of $x_0$ as byte 0 and ends with the least significant byte (or bit) of $x_4$ as byte 39. Table 1 lists the notation.

## 2.2 Algorithms

**Pseudorandom functions.** Ascon-Prf is parameterized by the key length of $k$ bits, output rate of $r$ bits, internal round number $a$, and maximum output length of $0 < t < 2^{32}$ bits (or $t = 0$ for unlimited output). The algorithm $\mathcal{G}_{k,r,a,t}$ takes as its input a secret key $K$ with $k$ bits, input data $M$ of arbitrary length, and a requested output length $\ell \leq t$. It returns an output $T$ of size $\ell$ bits:

$$\mathcal{G}_{k,r,a,t}(K, M, \ell) = T \,.$$

Ascon-PrfShort is parameterized by the key length $k$ bits, input length $m \leq 128$ bits, internal round number $a$, and output size $t \leq 128$ bits. The algorithm $\mathcal{F}_{k,m,a,t}$ takes as its input a secret key $K$ with $k$ bits and some input data $M$ of $m$ bits. It produces an output $T$ of size $t$ bits:

$$\mathcal{F}_{k,m,a,t}(K, M) = T \,.$$

Table 1: Notation used for ASCON's interface, mode, and permutation

| | |
|---|---|
| $K$ | Secret key $K$ of $k \leq 128$ bits |
| $M, D, T$ | Message $M$, data $D$, output/tag $T$ (in $r$-bit blocks $M_i, D_i, T_i$) |
| $S$ | The 320-bit state $S$ of the sponge construction |
| $p, p^a$ | Permutation $p^a$consisting of $a$ update rounds $p$ |
| $x \in \{0,1\}^k$ | Bitstring $x$ of length $k$ (variable if $k = *$) |
| $0^k$ | Bitstring of $k$ bits (variable length if $k = *$), all 0 |
| $\lvert x \rvert$ | Length of the bitstring $x$ in bits |
| $\lfloor x \rfloor_k$ | Bitstring $x$ truncated to the first (most significant) $k$ bits |
| $\lceil x \rceil^k$ | Bitstring $x$ truncated to the last (least significant) $k$ bits |
| $x \parallel y$ | Concatenation of bitstrings $x$ and $y$ |
| $x \oplus y$ | XOR of bitstrings $x$ and $y$ |
| $x \bmod y$ | Remainder in integer division of $x$ by $y$ |
| $\lceil x \rceil$ | Ceiling function, smallest integer larger than $x$ |
| $p_C, p_S, p_L$ | constant-addition, substitution and linear layer of $p = p_L \circ p_S \circ p_C$ |
| $x_0, \ldots, x_4$ | The five 64-bit words of the state $S$ |
| $x_{0,i}, \ldots, x_{4,i}$ | Bit $i$, $0 \leq i < 64$, of words $x_0, \ldots, x_4$, with $x_{\cdot,0}$ the rightmost bit (LSB) |
| $x \oplus y$ | Bitwise XOR of 64-bit words or bits $x$ and $y$ |
| $x \odot y$ | Bitwise AND of 64-bit words or bits $x$ and $y$ (denoted $x\,y$ in the ANF) |
| $x \ggg i$ | Right-rotation (circular shift) by $i$ bits of 64-bit word $x$ |

**Message authentication.** ASCON-MAC is parameterized by the key length $k$ bits, output rate $r$, internal round number $a$, and tag length $t$. It specifies an authentication algorithm $\mathcal{A}_{k,r,a,t}$ and a verification algorithm $\mathcal{V}_{k,r,a,t}$, both calling $\mathcal{G}_{k,r,a,t}$. The authentication algorithm $\mathcal{A}_{k,r,a,t}$ takes as its input a secret key $K$ with $k$ bits and a message $M$ of arbitrary length. It produces a tag $T$ of length $t$ as its output:

$$\mathcal{A}_{k,r,a,t}(K, M) = T\,.$$

The verification procedure $\mathcal{V}_{k,r,a,t}$ takes as input the key $K$, message $M$ and tag $T$, and outputs either pass if the verification of the tag is correct or fail if it fails:

$$\mathcal{V}_{k,r,a,t}(K, M, T) \in \{\mathsf{pass}, \mathsf{fail}\}\,.$$

## 2.3 Recommended Parameter Sets

Table 2 lists our recommended instances for PRFs. Table 2 shows our recommended instance for the MAC and specifies its parameters, including the key size $k$, the rate $r$, the maximum tag length $t$, and the number of rounds $a$ for the permutation $p^a$.

## 2.4 Arbitrary-Length Pseudorandom Functions

The mode of operation of ASCON-PRF is based on full-state keyed sponge modes [BDPV07] such as the DonkeySponge [BDPV12] mode. The PRF is illustrated in Figure 1 and specified in Algorithm 1.

Table 2: Parameters for recommended **Pseudorandom Functions (PRF)** and **Message Authentication Codes (MAC)**. Unlimited input/output lengths ('unlim.') are implicitly limited by the security claim to $\leq 2^{72}$ bits.

| Name | Algorithms | Bit size of | | | | | Rounds |
|---|---|---|---|---|---|---|---|
| | | key | data | (block) | output | (block) | $p^a$ |
| | | $k$ | $m$ | | $t$ | $r$ | $a$ |
| Ascon-Mac | $\mathcal{A}, \mathcal{V}_{128,128,12,128}$ | 128 | unlim. | 256 | 128 | 128 | 12 |
| Ascon-Prf | $\mathcal{G}_{128,128,12,0}$ | 128 | unlim. | 256 | unlim. | 128 | 12 |
| Ascon-PrfShort | $\mathcal{F}_{128,*,12,*}$ | 128 | $\leq 128$ | 128 | $\leq 128$ | 128 | 12 |

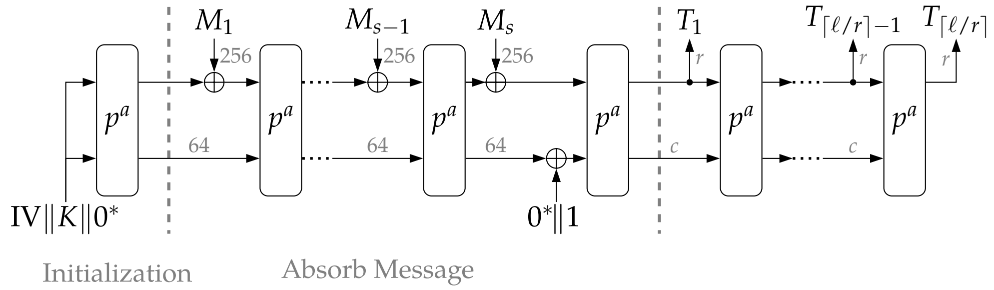



Figure 1: Pseudorandom Function $\mathcal{G}_{k,r,a,t}$ with output length $\ell$ ($\ell \leq t$ or $t = 0$).

### 2.4.1 Initialization

The 320-bit initial state of Ascon is formed by the secret key $K$ of $k$ bits and an IV specifying the algorithm. The 64-bit IV of Ascon-Prf specifies the algorithm parameters in a similar format as for Ascon, including $k$ and the rate $r$ each written as an 8-bit integer and round number $a$ encoded as an 8-bit integer as $2^7 + a = \texttt{80} \oplus a$, followed by the maximum output length of $t$ bits as a 32-bit integer, or $t = 0$ for arbitrarily long output:

$$\mathrm{IV}_{k,r,a,t} \leftarrow k \,\|\, r \,\|\, (1\|0^7) \oplus a \,\|\, 0^8 \,\|\, t$$
$$S \leftarrow \mathrm{IV}_{k,r,a,t} \,\|\, K \,\|\, 0^{256-k}$$

In the initialization, the $a$-round permutation $p^a$ is applied to the initial state:

$$S \leftarrow p^a(S)$$

### 2.4.2 Absorb Message

The PRF processes the padded message $M$, in blocks of 256 bits. The padding process appends a single 1 and the smallest number of 0s to $M$ such that the length of the padded message is a multiple of 256 bits. The resulting padded message is split into $s$ blocks of 256 bits, $M_1 \,\|\, \ldots \,\|\, M_s$:

$$M_1, \ldots, M_s \leftarrow 256\text{-bit blocks of } M \,\|\, 1 \,\|\, 0^{255-(|M| \bmod 256)}$$

Algorithm 1: PRF

PRF $\mathcal{G}_{k,r,a,t}(K, M, \ell) = T$

**Input:** key $K \in \{0,1\}^k$, input $M \in \{0,1\}^*$, output bitsize $\ell \leq t$ or $\ell$ arbitrary if $t = 0$
**Output:** output $T \in \{0,1\}^\ell$

**Initialization**
  $S \leftarrow p^a(\mathrm{IV}_{k,r,a,t} \,\|\, K \,\|\, 0^{256-k})$
**Absorbing**
  $M_1 \dots M_s \leftarrow$ 256-bit blocks of $M \| 1 \| 0^*$
  **for** $i = 1, \dots, s-1$ **do**
    $S \leftarrow p^a(S \oplus (M_i \,\|\, 0^{64}))$
  $S \leftarrow p^a(S \oplus (M_s \,\|\, 0^{63} \,\|\, 1))$
**Squeezing**
  $u = \lceil \ell / r \rceil$
  **for** $i = 1, \dots, u-1$ **do**
    $T_i \leftarrow \lfloor S \rfloor_r$
    $S \leftarrow p^a(S)$
  $T_u \leftarrow \lfloor S \rfloor_r$
  **return** $\lfloor T_1 \,\|\, \dots \,\|\, T_u \rfloor_\ell$

The message blocks $M_i$ with $i = 1, \dots, s-1$ are processed as follows. Each block $M_i$ is xored to the state $S$, followed by an application of the $a$-round permutation $p^a$ to $S$. For the last message block $M_s$ a single 1 is xored to the state in addition to the message block:

$$S \leftarrow \begin{cases} p^a(S \oplus (M_i \,\|\, 0^{64})) & \text{if } 1 \leq i \leq s-1 \\ p^a(S \oplus (M_i \,\|\, 0^{63} \,\|\, 1)) & \text{if } i = s \end{cases}$$

### 2.4.3 Squeeze Tag

Then the output is extracted from the state in $r$-bit blocks $T_i$ until the requested output length $\ell \leq t$ (or $\ell$ arbitrary if $t = 0$) is completed after $u = \lceil \ell / r \rceil$ blocks. After each extraction (except the last one), the internal state $S$ is transformed by the $a$-round permutation $p^a$:

$$\begin{aligned} T_i &\leftarrow \lfloor S \rfloor_r \\ S &\leftarrow p^a(S), \qquad 1 \leq i \leq u = \lceil t / r \rceil \end{aligned}$$

The last output block $T_u$ is truncated to $\ell$ mod $r$ bits and $\lfloor T_1 \,\|\, \dots \,\|\, T_u \rfloor_\ell$ is returned.

## 2.5 Message authentication

The message authentication code Ascon-Mac mainly relays inputs to the underlying PRF algorithm $\mathcal{G}_{k,r,a,t}$ and, for verification, checks if the transmitted tag $T$ matches the computed tag $T^*$. The MAC is specified in Algorithm 2.

Algorithm 2: Authentication and verification procedures

**Authentication**

$\mathcal{A}_{k,r,a,t}(K, M)$

**Input:** key $K \in \{0,1\}^k$, $k \leq 128$,
message $M \in \{0,1\}^*$

**Output:** tag $T \in \{0,1\}^t$

$T \leftarrow \mathcal{G}_{k,r,a,t}(K, M, t)$
**return** $T$

**Verification**

$\mathcal{V}_{k,r,a,t}(K, M, T)$

**Input:** key $K \in \{0,1\}^k$, $k \leq 128$,
message $M \in \{0,1\}^*$,
tag $T \in \{0,1\}^t$

**Output:** pass or fail

$T^* \leftarrow \mathcal{G}_{k,r,a,t}(K, M, t)$
**if** $T = T^*$ **return** pass
**else return** fail

### 2.6 Short-Input Pseudorandom Functions

The mode of operation of Ascon-PrfShort is essentially the initialization of Ascon-128 with a different initial value, and the nonce replaced by a single message block $M$ of length $m \leq 128$ bits. The resulting PRF Ascon-PrfShort is illustrated in Figure 2 and specified in Algorithm 3.

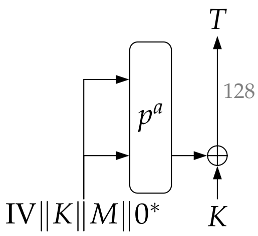



Figure 2: PRF $\mathcal{F}_{k,m,a,t}$ for short inputs

Algorithm 3: Short-input PRF. In an implementation, $m$ and $t$ can be inputs (instead of parameters).

PRF $\mathcal{F}_{k,m,a,t}(K, M) = T$

**Input:** key $K \in \{0,1\}^k$, $k \leq 128$,
input $M \in \{0,1\}^m$, $m \leq 128$
**Output:** output $T \in \{0,1\}^t$

$S \leftarrow p^a(\mathrm{IV}_{k,m,a,t} \,\|\, K \,\|\, M \,\|\, 0^{256-k-m})$
$T \leftarrow \lceil S \rceil^t \oplus \lceil K \rceil^t$
**return** $T$

As shown in Algorithm 3, the 320-bit input to $p^a$ is formed by an IV specifying the algorithm, the secret key $K$ of $k$ bits, and the message $M$ of $m$ bits. The 64-bit IV of $\mathcal{F}_{k,m,a,t}$ includes the key length $k$, the size of the input block $m$, and the size of the output block $t$, each written as an 8-bit integer, and the round number $a$ encoded as an 8-bit integer as $2^6 + a = \texttt{40} \oplus a$:

$$\mathrm{IV}_{k,m,a,t} \leftarrow k \,\|\, m \,\|\, (0\|1\|0^6) \oplus a \,\|\, t \,\|\, 0^{32}$$

This IV is concatenated with the secret key $K$ and the message $M$. Note that no padding is applied to $M$ and hence, $\mathcal{F}_{k,m,a,t}$ only accepts $M$ matching the length $m$. The permutation $p^a$ is applied to this state:

$$S \leftarrow p^a(\mathrm{IV}_{k,m,a,t} \,\|\, K \,\|\, M \,\|\, 0^{256-k-m})$$

The last $t$ bits of the state are then extracted as the tag $T$. The additional xor with the key $K$ is performed similarly to the authenticated encryption schemes:

$$T \leftarrow \lceil S \rceil^t \oplus \lceil K \rceil^t$$

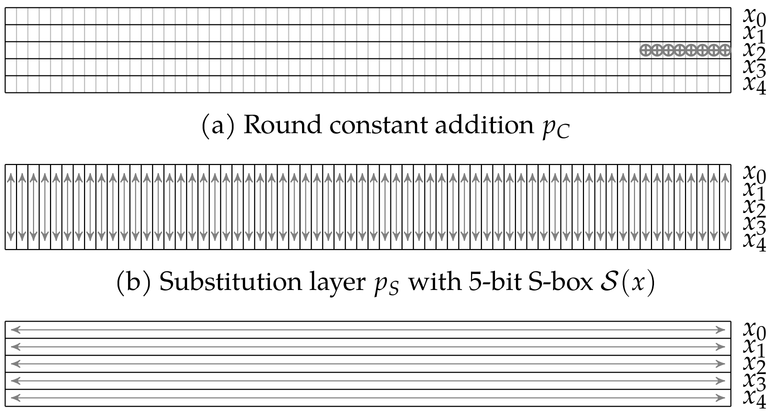
## 3 Permutation

The main component of the PRFs is the 320-bit permutation $p^a$. The permutation iteratively applies an SPN-based round transformation $p$ that in turn consists of three steps; $p_C$, $p_S$, $p_L$:

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

The number of rounds $a$ is a tunable security parameter.

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

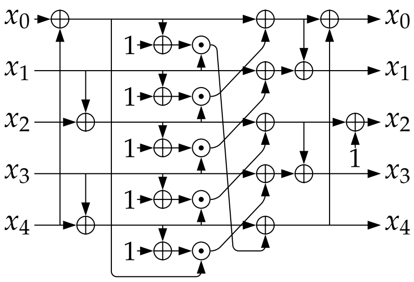



(a) Round constant addition $p_C$

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

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

Figure 3: The five 64-bit words of the 320-bit state $S$ and operations $p_L \circ p_S \circ p_C$.



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

(a) Ascon's 5-bit S-box $\mathcal{S}(x)$ (b) Ascon's linear layer with 64-bit functions $\Sigma_i(x_i)$

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

### 3.1 Addition of Constants

The constant addition step $p_C$ adds a round constant $c_r$ to word $x_2$ of the state $S$ in round $i$ (see Figure 3a). Both indices $r$ and $i$ start from zero and we use $r = i$ for $p^a$ (see Table 3):

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

Table 3: The round constants $c_r$ used in each round $i$ of $p^a$.

| $p^{12}$ | Constant $c_r$ | $p^{12}$ | Constant $c_r$ |
|---|---|---|---|
| 0 | `00000000000000f0` | 6 | `0000000000000096` |
| 1 | `00000000000000e1` | 7 | `0000000000000087` |
| 2 | `00000000000000d2` | 8 | `0000000000000078` |
| 3 | `00000000000000c3` | 9 | `0000000000000069` |
| 4 | `00000000000000b4` | 10 | `000000000000005a` |
| 5 | `00000000000000a5` | 11 | `000000000000004b` |

### 3.2 Substitution Layer

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

Table 4: Ascon's 5-bit S-box $\mathcal{S}$ as a lookup table.

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathcal{S}(x)$ | 4 | b | 1f | 14 | 1a | 15 | 9 | 2 | 1b | 5 | 8 | 12 | 1d | 3 | 6 | 1c |

| $x$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1a | 1b | 1c | 1d | 1e | 1f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathcal{S}(x)$ | 1e | 13 | 7 | e | 0 | d | 11 | 18 | 10 | c | 1 | 19 | 16 | a | f | 17 |

### 3.3 Linear Diffusion Layer

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

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

## 4 Security Claim

Ascon-Prf, Ascon-PrfShort, and Ascon-Mac are designed to help provide 128-bit security against key recovery and $\min(128, t)$ security against guessing $T$ for a random key and newly queried $M$. The number of message blocks processed by the algorithms is limited to a total of $2^{64}$ blocks per key. We consider this as more than sufficient for lightweight applications in practice.

It is beneficial that a system or protocol implementing the MAC algorithms monitors and, if necessary, limits the number of tag verification failures per key. After reaching this limit, the verification rejects all tags.

We emphasize that we do not require ideal properties for the permutation $p^a$. Non-random properties of the permutation $p^a$ are known and are unlikely to affect the claimed security properties of the algorithm.

## 5 Software Performance

A preliminary overview of the software performance of Ascon-Mac, Ascon-Prf and Ascon-PrfShort is given in Table 5a and Table 5b in comparison with Table 5c and Table 5d.

**Acknowledgments.** The authors would like to thank all researchers contributing to the design, analysis, and implementation of Ascon. In particular, we want to thank Hannes Gross and Robert Primas for all their support and various implementations of Ascon. Furthermore, we want to thank Bart Mennink for giving feedback on this document.

Part of this work has been supported by the Austrian Science Fund (FWF): P26494-N15 and J 4277-N38, by the European Union's Horizon 2020 research and innovation programme (H2020 ICT 644052: HECTOR), and by the Austrian Government (FFG/SFG COMET 836628: SeCoS and FIT-IT 835919: SePAG).

Table 5: Software performance in cycles per byte of Ascon-Mac, Ascon-Prf and Ascon-PrfShort compared to Ascon and Ascon-128a.

(a) Ascon-Mac and Ascon-Prf

| Message Length | 1 | 8 | 16 | 32 | 64 | 1536 | long |
|---|---|---|---|---|---|---|---|
| Intel® Core™ i5-6300U | 427 | 48 | 24 | 21 | 13.3 | 6.4 | 6.2 |
| Intel® Core™ i5-4200U | 517 | 64 | 33 | 25 | 16.3 | 8.7 | 8.4 |
| ARM1176JZF-S (ARMv6) | 1803 | 237 | 123 | 89 | 60.1 | 33.1 | 33.4 |

(b) Ascon-PrfShort

| Message Length | 1 | 8 | 16 |
|---|---|---|---|
| Intel® Core™ i5-6300U | 188 | 23 | 12 |
| Intel® Core™ i5-4200U | 257 | 33 | 17 |
| ARM1176JZF-S (ARMv6) | 1098 | 136 | 72 |

(c) Ascon-128

| Message Length | 1 | 8 | 16 | 32 | 64 | 1536 | long |
|---|---|---|---|---|---|---|---|
| Intel® Core™ i5-6300U | 367 | 58 | 35 | 23 | 17.6 | 11.9 | 11.4 |
| Intel® Core™ i5-4200U | 521 | 81 | 49 | 32 | 23.9 | 16.2 | 15.8 |
| ARM1176JZF-S (ARMv6) | 2136 | 312 | 186 | 123 | 91.6 | 61.8 | 62.2 |

(d) Ascon-128a

| Message Length | 1 | 8 | 16 | 32 | 64 | 1536 | long |
|---|---|---|---|---|---|---|---|
| Intel® Core™ i5-6300U | 365 | 47 | 31 | 19 | 13.5 | 8.0 | 7.8 |
| Intel® Core™ i5-4200U | 519 | 67 | 44 | 27 | 18.8 | 11.0 | 10.6 |
| ARM1176JZF-S (ARMv6) | 2118 | 261 | 170 | 107 | 75.6 | 46.0 | 46.6 |

# Bibliography

[BDPV07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Sponge functions". Ecrypt Hash Workshop 2007. 2007. URL: http://sponge.noekeon.org/SpongeFunctions.pdf (pp. 2, 3).

[BDPV12] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Permutation-based Encryption, Authentication and Authenticated Encryption". DIAC 2012. July 2012 (pp. 2, 3).

[Cae14] The CAESAR committee. "CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness". 2014. URL: https://competitions.cr.yp.to/caesar-submissions.html (p. 1).

[DEM+20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. "Isap v2.0". In: IACR Transactions on Symmetric Cryptology 2020.S1 (2020), pp. 390–416. DOI: 10.13154/tosc.v2020.iS1.390-416 (p. 1).

[DEM+21] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. "Isap v2.0 (Submission to NIST)". Finalist of NIST lightweight cryptography standardization process, https://csrc.nist.gov/Projects/Lightweight-Cryptography/. 2021 (p. 1).

[DEMS14] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon v1". Submission to the CAESAR competition. 2014. URL: https://ascon.iaik.tugraz.at (p. 1).

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon v1.2". Submission to Round 3 of the CAESAR competition. 2016. URL: https://ascon.iaik.tugraz.at (p. 1).

[DEMS21a] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon v1.2 (Submission to NIST)". Finalist of NIST lightweight cryptography standardization process, https://csrc.nist.gov/Projects/Lightweight-Cryptography/. 2021 (p. 1).

[DEMS21b] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon v1.2: Lightweight Authenticated Encryption and Hashing". In: Journal of Cryptology 34.3 (2021), p. 33. DOI: 10.1007/s00145-021-09398-9. URL: https://doi.org/10.1007/s00145-021-09398-9 (p. 1).

[DMV17] Joan Daemen, Bart Mennink, and Gilles Van Assche. "Full-State Keyed Duplex with Built-In Multi-user Support". In: ASIACRYPT 2017. Vol. 10625. LNCS. Springer, 2017, pp. 606–637. DOI: 10.1007/978-3-319-70697-9_21. IACR: 2017/498 (p. 2).

[MRV15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. "Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption". In: ASIACRYPT 2015. Vol. 9453. LNCS. Springer, 2015, pp. 465–489. DOI: 10.1007/978-3-662-48800-3_19 (p. 2).

[Nat18] National Institute of Standards and Technology. "Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process". https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf. 2018 (p. 1).