Addendum to the CAESAR Hardware API v1.0

1. Minor change to Section 1.9 Supported maximum size of AD/plaintext/ciphertext

We replace the current size limits:

single-pass authenticated ciphers:	2^{32} bytes
two-pass authenticated ciphers:	2^{11} bytes

with

single-pass authenticated ciphers:	2^{32}-1 bytes
two-pass authenticated ciphers:	2^{11}-1 bytes

Justification:
The current sizes unnecessarily increase the sizes of internal counters/registers by one bit, while supporting just one additional (not very likely) size of AD/plaintext/ciphertext.

2. Clarification regarding the Length segment, described in the last but one paragraph of Section 3 Communication Protocol

We restrict the use of the Length Segment to "offline" algorithms, such as AES-CCM, understood as algorithms that require the availability of the lengths of the AD and plaintext (ciphertext) in advance, before the authenticated encryption (decryption) starts.

The Length segment must not be used in the implementations of "online" algorithms, such as AES-GCM, in which all lengths can be calculated as the AD/plaintext/ciphertext arrives and is processed.

For the "offline" authenticated ciphers, which are permitted to use the Length Segment, we make its format common for all algorithms, and define it as follows:

High-speed implementations:

For $32 \leq w < 64$
    Header (w bits)
    AD length (w bits)
    Data length (w bits)

For $64 \leq w \leq 256$
    Header (w bits)
    AD length (32 bits) || Data length (32 bits) || $0^{(w-64)}$

Lightweight implementations:

For w=8, 16, 32

> Header (32 bits) || AD length (32 bits) || Data length (32 bits)
> All divided into w-bit words, provided at the PDI bus, starting from the leftmost word.

Notation: Data length means Plaintext length for encryption and Ciphertext length for decryption.

Justification:
The CAESAR HW API Specification v.1.0 is not explicitly encouraging the use of the Length segment for "online" algorithms. Vice versa, it introduces this concept only in the context of "offline" algorithms, such as AES-CCM, for which the entire lengths of associated data and plaintext/ciphertext have to be known before the encryption/decryption starts [NIST Special Publication 800-38D].

Letting the designers to choose
1) whether to use the Length segment or not
2) the position of the Length segment among other segments
3) the exact length of the length segment
4) the exact length of its individual fields (with multiple valid choices)

clearly leads to the potential incompatibility, and noticeable performance/resource utilization differences, among the implementations of the same algorithm by different groups. On top of that there are also clear implications in terms of the relative security and the division of tasks between AEAD and the preceding software/hardware units.

The authors have been doing their best to avoid these kinds of incompatibilities by the precise definition of all other segments, all minimum compliance criteria, and even timing waveforms. This addendum eliminates the remaining major source of incompatibility, and thus it makes the CAESAR Hardware API specification more precise, and less prone to different interpretations.

3. Recommended interface of two-pass algorithms

The recommended interface of two-pass algorithms is shown in Fig. A1.

Compared to the interface of single-pass algorithms, shown in Fig. 1, additional ports used for communication with the external Two-Pass FIFO have been added. The width of the data buses of these ports is defined by an additional constant (determined by the corresponding generic of AEAD_TP.vhd), denoted in Fig. A1 as fw. The value of fw is typically the same as the block size of the implemented authenticated cipher, and therefore we leave it unconstrained.

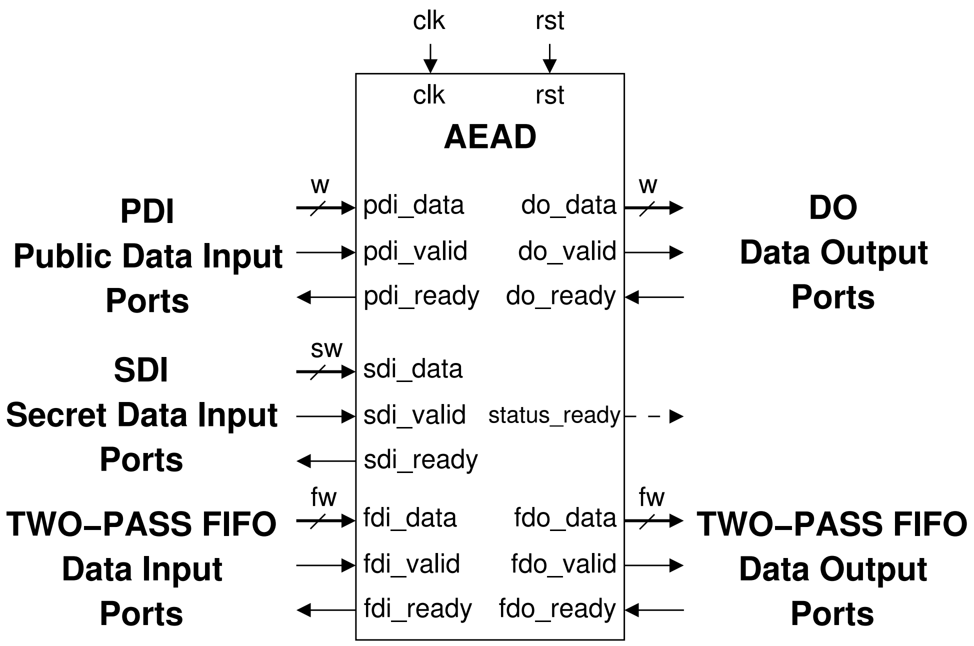

Fig. A1: AEAD Interface for Two-Pass Algorithms

Justification:

The amount of resources required for the Two-Pass FIFO can be easily estimated using the size of FIFO and the value of fw. Thus, there is no reason to include this FIFO in every benchmarking run for AEAD.vhd. The inclusion of memory inside of AEAD would makes it more difficult to gauge the resource utilization required by the primary cipher functionality.

4. Recommended support for two maximum lengths of AD/plaintext/ciphertext in case of single-pass algorithms

We recommend that all single-pass algorithms are implemented in such a way that

1. The Verilog/VHDL code supports at least two maximum lengths of AD/plaintext/ciphertext:
    a. Maximum length for single-pass algorithms: 2^{32}-1
    b. Maximum length for two-pass algorithms: 2^{11}-1
2. The choice between these two lengths is possible at the time of synthesis, by changing the value of a single generic or constant, named G_MAX_LEN, with at least two supported values, representing $\log_2$(maximum length + 1):
    a. SINGLE_PASS_MAX=32 (representing 2^{32}-1)
    b. TWO_PASS_MAX=11 (representing 2^{11}-1)
3. The design for the case of a smaller maximum size, 2^{11}-1, should be optimized in terms of resource utilization and maximum clock frequency (e.g., by choosing smaller sizes of counters and registers used to calculate and store the respective lengths).

Justification:
In order to fairly compare the implementations of single-pass and two-pass algorithms, the compared implementations should support the same maximum lengths of AD, plaintext, and ciphertext. Since the implementations of two-pass algorithms in modern FPGAs cannot typically reach the maximum length of 2^{32}-1 bytes (without using off-chip memory), it makes sense to perform the comparison for the maximum length supported by both types of algorithms, namely 2^{11}-1.

The implementations of single-pass algorithms can take advantage of this smaller limit to reduce the resource utilization and/or minimum clock period. Although, typically this gain is relatively small, we would like to give the designs teams an opportunity to clearly demonstrate its magnitude in order to avoid any unjustified claims about "comparing apples with oranges".

It should be stressed that even if both implementations support the same maximum length of 2^{11}-1 bytes, the implementations of two-pass algorithms will require, on top of the logic resources (such as Slices, LUTs, ALMs, ALUTs, etc.) used to implement a typical single-pass AEAD unit, also a Two-Pass FIFO, with the total capacity of 2^{11} bytes = 2 kbytes = 16 kbits.

In modern FPGAs, this Two-Pass FIFO will be implemented using block memories (such as BRAMs of Xilinx FPGAs and embedded memory blocks of Altera FPGAs). A FIFO with the capacity of 2^{11} bytes can be built using a negligible percentage of the total capacity of on-chip block memories. Thus, the two-pass algorithms are not in any significant way disadvantaged compared to single-pass algorithms.

In ASICs, the Two-Pass FIFO consumes the same resources (transistors, silicon area) as the AEAD core. As a result, a potential overhead caused by an external FIFO can be much more significant, and should be clearly determined during benchmarking, *independently* of benchmarking the AEAD core itself. The combined results should be then taken into consideration during the comparison with results for single-pass algorithms.

The comparison for the maximum length of AD/plaintext/ciphertext equal to 2^{11}-1 = 2047 bytes is additionally justified by the fact that the threshold of 2047 bytes is greater than the Maximum Transmission Unit (MTU) for most Ethernet networks (1500 bytes), on which the size of packets in popular secure networking protocols, such as IPSec, is based. These secure networking protocols are then the primary targets for high-speed hardware implementations of authenticated encryption. Authenticated encryption without intermediate tags (which is a focus of hardware benchmarking using current CAESAR API) is in general not a very good match for applications requiring protection of large volumes of data (especially data at rest), due to large latency required to access decrypted data.