Documentation of Protected Ascon Implementations

1 Protection Method

The provided protected implementations of Ascon-128 feature Domain-oriented masking (DOM) [4] with protection orders 1 and 2. The concrete implementations are inspired by the description of masked Ascon implementation in Section 6.3 (page 72) of [3].

All implementations require 2 cycles to compute one permutation round, the masking schemes hence adds one additional cycle of latency per round. Absorption and squeezing happens concurrently for all shares and is hence as fast as in case of unprotected implementations. The throughput of masked implementations is hence increased to 1.75 cycles/byte when using a 32-bit interface (per share). In comparison, a corresponding unprotected implementations achieve a throughput of 1 cycle/byte.

The implementations are not optimized for low randomness requirements and hence require the expected 320 bits (960 bits) of fresh randomness every other cycle when computing DOM-AND gates on the 320-bit Ascon state in case of 1st (2nd) order implementations. We want to point out that techniques such as changing of the guards [1] could be used to significantly reduce the amount of required fresh randomness.

The tag comparison during decryption is currently simply implemented in an unmasked fashion.

In the following we point out the differences between the provided implementations in more detail:

v1 1st-order DOM.
Incomplete register layer (updated on falling clk) after first affine layer to avoid glitchy-dependend inputs of indep-DOM-AND gates in the subsequent keccak sbox layer.
Incomplete register layer (updated on rising clk) after DOM-compute step to avoid glitch-related issues.

v2 1st-order DOM.
Incomplete register layer (updated on falling clk) after first affine layer to avoid glitchy-dependend inputs of indep-DOM-AND gates in the subsequent keccak sbox layer.
Complete register layer (updated on rising clk) after DOM-compute step to avoid glitch-related issues and potentially allow an overall higher maximum clock frequency.

v3 Same as v1 except for the 2nd-order protection level.

v4 Same as v2 except for the 2nd-order protection level.

2 Preliminary Security Evaluation

We have successfully formally verified the correctness of our masked implementations of Ascon-p in the glitch-extended probing model and for the respective protection order using the tool Coco [2].

References

[1] J. Daemen. "Changing of the Guards: A Simple and Efficient Method for Achieving Uniformity in Threshold Sharing". In: *CHES*. Vol. 10529. Lecture Notes in Computer Science. Springer, 2017, pp. 137–153.

[2] B. Gigerl, V. Hadzic, R. Primas, S. Mangard, and R. Bloem. "Coco: Co-Design and Co-Verification of Masked Software Implementations on CPUs". In: *USENIX Security Symposium*. USENIX Association, 2021, pp. 1469–1468.

[3] H. Groß. "Domain-Oriented Masking-Generically Masked Hardware Implementations". In: *PhD Thesis*. 2018. URL: https://diglib.tugraz.at/download.php?id=5c80ea0c43a56&location=browse.

[4] H. Groß, S. Mangard, and T. Korak. "Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order". In: *TIS@CCS*. ACM, 2016, p. 3.