Efficient Implementation of Permutation-Based Hash Functions for the RISC-V Architecture

Issam Jomaa, Hao Cheng, Johann Großschädl, and Peter Y. A. Ryan

DCS and SnT, University of Luxembourg,
6, Avenue de la Fonte, L–4364 Esch-sur-Alzette, Luxembourg
issam.jomaa.001@student.uni.lu
{hao.cheng,johann.groszschaedl,peter.ryan}@uni.lu

Abstract. The National Institute of Standards and Technology has recently finished an evaluation of algorithms for authenticated encryption and hashing aimed at resource-restricted devices. Benchmarking results published during this evaluation show that the three permutation-based AEAD algorithms Ascon, Schwaemm, and Xoodyak perform very well in software, especially on 32-bit microcontroller architectures like ARM and RISC-V. While there exist numerous benchmarks for authenticated encryption algorithms, relatively little is known about the performance of permutation-based hash functions on RISC-V. We fill this gap in the present paper by describing optimized RISC-V implementations of the three hash functions Ascon-Hash, Esch256, and Xoodyak, which we benchmarked on a Nuclei Systems RV-STAR development board. The underlying permutations are written in RISC-V Assembly, whereby we developed implementations based on the core RV32I instruction set as well as variants using the RV32B BitManip extension. Our experiments show that, although the three permutations are based on different design principles, they achieve roughly the same execution time (around 1500 clock cycles) on our RV-STAR prototyping board. In addition, all three permutations can be significantly accelerated by the dedicated rotation instructions of the BitManip extension. However, when it comes to the execution time of the full hash, Ascon-Hash and Esch256 introduce higher overhead than Xoodyak, mainly due to auxiliary operations like the conversion to (resp. from) bit-interleaved representation in the case of Ascon-Hash or the "indirect" injection of data blocks into the state of Esch256. We describe how the performance penalty caused by these auxiliary operations can be minimized on RISC-V microcontrollers.