New Ascon Implementations

Christoph Dobraunig, Maria Eichlseder, Florian Mendel, Robert Primas, Martin Schläffer

NIST LWC Workshop 2022
Outline

Ascon Overview

Performance and Code Size

Implementation Techniques

Side-channel Protection

Evaluation and Verification
Ascon Overview
Ascon Mode for Authenticated Encryption

Initialization

Associated Data

Plaintext

Finalization

- Designed in 2014 [DEMS16], Journal of Cryptology in 2021 [DEMS21c]
- First choice for lightweight AEAD in CAESAR portfolio
- Extensive published cryptanalysis confirming its security margin
- Additional modes for Hash, XOF, MAC, PRF [DEMS21a; DEMS21b]
Ascon Permutation with \{6, 8, 12\} Rounds

S-box layer

Linear layer

\[ x_0 := x_0 \oplus (x_0 \gg 19) \oplus (x_0 \gg 28) \]
\[ x_1 := x_1 \oplus (x_1 \gg 61) \oplus (x_1 \gg 39) \]
\[ x_2 := x_2 \oplus (x_2 \gg 1) \oplus (x_2 \gg 6) \]
\[ x_3 := x_3 \oplus (x_3 \gg 10) \oplus (x_3 \gg 17) \]
\[ x_4 := x_4 \oplus (x_4 \gg 7) \oplus (x_4 \gg 41) \]
ASCON-128 vs ASCON-128a

- ASCON-128a: 33% more performance, more rounds, larger rate
- Same security, different trade-off (rate vs. number of rounds)
- Both scrutinized for 8 years in cryptographic competitions
- Most security analysis can be applied to both algorithms
- Similar security margin, no clear preference
Ascon Implementations

https://github.com/ascon/ascon-c (Ascon team)
- AEAD, Hash, XOF, MAC, PRF
- C: ref, speed/area optimized, combined
- ASM: esp32, armv6, armv6m, armv7m, rv32
- Masked C+ASM: 2-4 shares, leveled

https://github.com/rweather/ascon-suite (Rhys Weatherley)
- AEAD, Hash, HKDF, ISAP, KMAC, PBKDF2, PRNG, SIV, XOF
- 8/32/64-bit C, AVR, ARM, RISC-V, m68k, Xtensa (ESP32)
- Framework to generate C/ASM/masked implementations
Performance and Code Size
New Ascon Implementations
(Improvements in the Final Round)

- Fewer instructions for S-box [CJL+20]: -10%
- Improved 8-bit AVR [ascon-suite] (time/size): -11%/-44%
- Combined Ascon AEAD+Hash [ascon-c] (size): -17%
- Improved low-size [ascon-c] (size 128/128a/Hash): -7%/-30%/-20%
- Bit-interleaved interface [ascon-c] (time 128/128a/Hash): -17%/-23%/-5%
- ESP32 implementations [ascon-c][Bac22] (time/size): -66%/-64%
- RV32 implementations [ascon-c][Bac22] (RV32,RV32I,RV32B): New
- Masked ARMv6/RV32 [ascon-c] (leveled, 2-4 shares): New
- Ascon-Hasha, Ascon-Xofa [DEMS21b] (time): -33%
- Ascon-MAC, Ascon-PRF compared to Ascon-KMAC [DEMS21a] (time): -66%
Microcontroller Benchmarking

\[\text{ascon—nocrypt \over \text{best—nocrypt}}\] for primary submission @las3

<table>
<thead>
<tr>
<th>Performance (time)</th>
<th>Code size (ROM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uno: 1.34x</td>
<td>Uno: 3.22x</td>
</tr>
<tr>
<td>F1: 1.06x</td>
<td>F1: 1.62x</td>
</tr>
<tr>
<td>ESP: 1.92x</td>
<td>ESP: 1.31x</td>
</tr>
<tr>
<td>F7: 1.02x</td>
<td>F7: 1.10x</td>
</tr>
<tr>
<td>R5: 0.61x</td>
<td>R5: 1.07x</td>
</tr>
</tbody>
</table>

https://lwc.las3.de/ [2020/10/14]
Microcontroller Benchmarking

Ascon-128: best primary finalist in most categories

<table>
<thead>
<tr>
<th>Performance (time)</th>
<th>Code size (ROM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uno: 1.24x</td>
<td>Uno: 1.59x</td>
</tr>
<tr>
<td>F1: 1.28x</td>
<td>F1: 0.80x</td>
</tr>
<tr>
<td>ESP: 0.58x</td>
<td>ESP: 0.55x</td>
</tr>
<tr>
<td>F7: 0.89x</td>
<td>F7: 0.87x</td>
</tr>
<tr>
<td>R5: 0.55x</td>
<td>R5: 0.90x</td>
</tr>
</tbody>
</table>

https://lwc.las3.de/ [2022/05/05]
# Microcontroller Benchmarking

Ascon-128: best primary finalist in most categories

<table>
<thead>
<tr>
<th>Performance (time)</th>
<th>Code size (ROM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uno: 1.24x</td>
<td>Uno: 1.59x</td>
</tr>
<tr>
<td>F1: 1.28x</td>
<td>F1: 0.80x</td>
</tr>
<tr>
<td>ESP: 0.58x</td>
<td>ESP: 0.55x</td>
</tr>
<tr>
<td>F7: 0.89x</td>
<td>F7: 0.87x</td>
</tr>
<tr>
<td>R5: 0.55x</td>
<td>R5: 0.90x</td>
</tr>
</tbody>
</table>

[https://lwc.las3.de/](https://lwc.las3.de/) [2022/05/05]
High-end Benchmarking

(Imagine Ascon hardware instructions)

<table>
<thead>
<tr>
<th></th>
<th>AMD Ryzen 9:</th>
<th>ARM Cortex-A72:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascon-128a</td>
<td>5.1 c/b</td>
<td>Ascon-128a</td>
</tr>
<tr>
<td>Ascon-128</td>
<td>7.8 c/b</td>
<td>Ascon-128</td>
</tr>
<tr>
<td>Ascon-HashA</td>
<td>10.6 c/b*</td>
<td>Ascon-HashA</td>
</tr>
<tr>
<td>Ascon-Hash</td>
<td>15.9 c/b</td>
<td>Ascon-Hash</td>
</tr>
</tbody>
</table>

https://bench.cr.yp.to/ [2022/05/03]

* estimated, not yet benchmarked
Implementation Techniques
Flexibility of Ascon Components

- Parallelism: S-box and linear layer support up to 5 ALUs
- Small state: 10 32-bit registers, 2 temporary, 1 for loop
- S-box: new description with fewer instructions
- Linear: 64-bit rotate or bit interleaving or funnel shift
- Modes: combine absorb, squeeze, insert (xor, read, write)
- Rate: loop for combined implementations (rate 64, 128)
- Short messages: only init and final needed
Ascon Hardware Extensions

- Fast, lightweight Ascon round instruction for 32-bit ARM/RV32 [SP20]
  - RI5CY Ascon-\(p\) with 4.7kGE: speedup factor 50x
  - Reuse 10 registers of CPU register file
- ARM Custom Datapath Extension, RISC-V Bitmanip Extension, ...
  - 32-bit funnel shift instructions (RV32B: \texttt{FSRI}, ESP32: \texttt{SRC})
  - 32-bit interleaving instructions (RV32B: \texttt{ZIP/UNZIP}, ARM CDE: \texttt{CX3})
  - Fused AND/XOR, BIC/XOR instructions (ARM A64: \texttt{BCAX}, ARM CDE: \texttt{CX3A})
  - SHA-2 like Sigma instructions (ARM CDE: \texttt{CX3DA})
Bit-interleaved Interface
(ascon128bi32, ascon128abi32, asconhashbi32, asconhashabi32)

- Convention: data is stored/transmitted in bit interleaved format
- Communication parties need to agree, similar to endianess
- Improved performance on 32-bit ARM platforms:
  - ASCON-128/ASCON-128a: -17%/-23%
- Also demonstrates improvement of ASCON with
  - Bit-interleaving instructions (obvious)
  - Funnel shift instructions (same effect!)
Side-channel Protection
Designed with SCA in Mind

- Algebraic degree 2 of S-box
- Limited damage if state is recovered
- Leveled implementations [BBC+20]
  - Higher protection order for Init/Final (key)
  - Lower protection order for AD/PT/CT processing (data)
- Masking using Toffoli gate [DDE+20]
Masking using Toffoli Gate

- More efficient than masked AND gate
  - Fewer instructions, registers, randomness
- No fresh randomness needed during round computation
  - Randomness is not lost (invertible shared Toffoli gate)
  - Randomness of previous round can be reused
- Benefits of invertible shared function:
  - Uniform by design
  - SIFA: Reduced attack surface if used with redundancy [DDE+20]
**1st-order Masked Keccak S-box**

State: \([a_0, a_1, b_0, b_1, c_0, c_1, d_0, d_1, e_0, e_1, r_0]\)

\((r_1, r_0) \leftarrow \text{clone}(r_0)\)
\(\text{toffoli}_\text{shared}(r_0, r_1, e_0, e_1, a_0, a_1)\)
\(\text{toffoli}_\text{shared}(a_0, a_1, b_0, b_1, c_0, c_1)\)
\(\text{toffoli}_\text{shared}(c_0, c_1, d_0, d_1, e_0, e_1)\)
\(\text{toffoli}_\text{shared}(e_0, e_1, a_0, a_1, b_0, b_1)\)
\(\text{toffoli}_\text{shared}(b_0, b_1, c_0, c_1, d_0, d_1)\)
\(d_0 \leftarrow \text{xor}(d_0, r_0)\)
\(d_1 \leftarrow \text{xor}(d_1, r_1)\)

- Similar constructions for higher degree S-boxes may be less efficient [DDE+20]
Further SCA Optimizations

- Preliminary Goal: Achieve 1st-order protection with 2/3 shares in C\(^1\)
  - Rotation offset between shares
  - Minimum number of ASM instructions (Toffoli gate)
  - Some register clears/NOPS needed
  - Extension to 3-shares with trick from [SM21]

- Performance in cycles/byte (green: evaluated)

<table>
<thead>
<tr>
<th>impl/shares</th>
<th>armv6 flags</th>
<th>C -O2</th>
<th>C -Os</th>
<th>2-1-2 -O2</th>
<th>2-1-2 -Os</th>
<th>2 -O2</th>
<th>2 -Os</th>
<th>3 -O2</th>
<th>3 -Os</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM1176JZF</td>
<td></td>
<td>58</td>
<td>70</td>
<td>85</td>
<td>88</td>
<td>100</td>
<td>260</td>
<td>343</td>
<td>524</td>
</tr>
<tr>
<td>STM32F415</td>
<td></td>
<td>59</td>
<td>84</td>
<td>90</td>
<td>90</td>
<td>98</td>
<td>320</td>
<td>378</td>
<td>650</td>
</tr>
</tbody>
</table>

\(^1\)Our implementations should be considered as a starting point to generate device specific C/ASM implementations
Evaluation and Verification
Testvector Leakage Assessment

- Goal: $1^{\text{st}}$-order protection with 2/3 shares
- Evaluation setup:
  - ChipWhisperer-Lite
  - UFO Board
  - STM32F303, STM32F415
  - We set $p^a, p^b = 2$ due to limited sample buffer
- We present decryption results of $\text{protected\_bi32\_armv6}$
- More implementations/results available at: https://github.com/ascon/simpleserial-ascon
TVLA Results

- STM32F303
- 3 (rotated) shares
- No device-specific fixes
- 8m traces
TVLA Results

- STM32F415
- 2 (rotated) shares
- Device-specific fixes
- 4m traces
TVLA Results

- STM32F415
- 2 (rotated) shares
- Device-specific fixes
- 5m traces
Formal Masking Verification

- Formal verification of masking in SW/HW using Coco [GHP+21]
  - Based on ideas of REBECCA [BGI+18]
- Verifies masked software in “hardware probing model” on CPU netlists
  - Considers stable signals, transitions, glitches
  - RISC-V IBEX core (comparable to ARM Cortex-M0)
- Also suitable for masked hardware circuits with/without state machines
Coco Verification Flow

1. Masked Cipher
   - Create Testbench
   - Verilator Testbench
   - Parse
   - Secured IBEX
   - Circuit Graph

2. .sv
   - Parse
   - Yosys

3. .py
   - Create
   - Verilator
   - Trace
   - Execution Trace
   - Verification Configuration

4. .v
   - .vcd
   - .py

Verification Result:
- Yes, secure.
- No, not secure. Leak in cycle ... at gate ...
Coco Verification Results

- Hardened RISC-V IBEx core from [GHP+21] as reference
- We mapped one round of 2-share Ascon-$p$ round from to RISC-V ASM
- We verified 1\textsuperscript{st}-order probing security (incl. transitions/glitches)
  - No online randomness
  - Performance of 260 c/b
  - Multi-round correctness due to uniformity of masking
Questions
Bibliography I


Bibliography IV