Efficient Second-Order Masked Software Implementations of Ascon in Theory and Practice

Barbara Gigerl, Florian Mendel, **Robert Primas**, Martin Schläffer

NIST LWC Workshop 2023

Outline

Ascon Implementation Overview

Existing 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: 2-3 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), 6502
- Framework to generate C/ASM/masked implementations

Existing 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: 2-3 shares, leveled ⇐ Goal: 1st-order security

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), 6502
- Framework to generate C/ASM/masked implementations

Our Presented Implementations:

- Goal: Efficient 2nd-order security using 3 shares on ARM
  - Features additional SCA hardening
  - Withstand bivariate t-tests
- No online randomness
  - Approx. same performance with/without hardware RNG
- Formally verified masking on RISC-V IBEX core
- Code will be available at: https://github.com/ascon/ascon-c
- Paper preprint: https://ascon.iaik.tugraz.at/files/ascon-masked-implementation.pdf

Masked Designs of Ascon

Ascon Designed with SCA in Mind

- Keyed initialization/finalization limit 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)
- Algebraic degree 2 of S-box
- Masking using Toffoli gate [DDE+20]

Masking using Toffoli Gate: $c \leftarrow c \oplus \overline{a}b$

- 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
- Other benefits of invertible shared function:
  - SIFA: Reduced attack surface if used with redundancy [DDE+20]

1^{st}-order Masked Toffoli Gate: $c \leftarrow c \oplus \overline{a}b$

Name: $p_{\chi 2S}$ [DDE+20]

In-/Output: $\{c_0, c_1, a_0, a_1, b_0, b_1\}$

$c_0 \leftarrow c_0 \oplus \overline{a_0} b_1$

$c_0 \leftarrow c_0 \oplus \overline{a_0} b_0$

$c_1 \leftarrow c_1 \oplus a_1 b_1$

$c_1 \leftarrow c_1 \oplus a_1 b_0$

1st-order Masked Keccak S-box

Name: χ_{2S} [DDE+20]

In-/Output: $\{a_0, a_1, b_0, b_1, c_0, c_1, d_0, d_1, e_0, e_1, r_0, r_1\}$

$p_{\chi 2S}(r_0, r_1, e_0, e_1, a_0, a_1)$

$p_{\chi 2S}(a_0, a_1, b_0, b_1, c_0, c_1)$

$p_{\chi 2S}(c_0, c_1, d_0, d_1, e_0, e_1)$

$p_{\chi 2S}(e_0, e_1, a_0, a_1, b_0, b_1)$

$p_{\chi 2S}(b_0, b_1, c_0, c_1, d_0, d_1)$

$d_0 \leftarrow d_0 \oplus r_0$

$d_1 \leftarrow d_1 \oplus r_1$

2^{nd}-order Masked Toffoli Gate: $c \leftarrow c \oplus \overline{a}b$

Name: $p_{\chi 3S}$

In-/Output: $\{c_0, c_1, c_2, a_0, a_1, a_2, b_0, b_1, b_2, R_0, R_1, R_2\}$

$c_0 \leftarrow c_0 \oplus a_0 b_2$

$c_0 \leftarrow c_0 \oplus a_0 b_1 \oplus R_0$

$c_0 \leftarrow c_0 \oplus \overline{a_0} b_0$

$c_1 \leftarrow c_1 \oplus a_1 b_2$

$c_1 \leftarrow c_1 \oplus \overline{a_1} b_1 \oplus R_1$

$c_1 \leftarrow c_1 \oplus a_1 b_0$

$c_2 \leftarrow c_2 \oplus \overline{b_0} a_2$

$c_2 \leftarrow c_2 \oplus a_2 b_1 \oplus R_2$

$c_2 \leftarrow c_2 \oplus a_2 \barwedge b_2$

2^{nd}-order Masked Toffoli Gate: $c \leftarrow c \oplus \overline{a}b$

Name: $p_{\chi 3S}$

In-/Output: $\{c_0, c_1, c_2, a_0, a_1, a_2, b_0, b_1, b_2, R_0, R_1, R_2\}$

$c_0 \leftarrow c_0 \oplus a_0 b_2$

$c_0 \leftarrow c_0 \oplus a_0 b_1 \oplus R_0$

$c_0 \leftarrow c_0 \oplus \overline{a_0} b_0$

$c_1 \leftarrow c_1 \oplus a_1 b_2$

$c_1 \leftarrow c_1 \oplus \overline{a_1} b_1 \oplus R_1$

$c_1 \leftarrow c_1 \oplus a_1 b_0$

$c_2 \leftarrow c_2 \oplus \overline{b_0} a_2$

$c_2 \leftarrow c_2 \oplus a_2 b_1 \oplus R_2$

$c_2 \leftarrow c_2 \oplus a_2 \barwedge b_2$

2^{nd}-order Masked Keccak S-box

Name: χ_{3S}

In-/Output: $\{a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2, d_0, d_1, d_2, e_0, e_1, e_2, r_0, r_1, r_2, R_0, R_1, R_2\}$

$p_{\chi 3S}(r_0, r_1, r_2, e_0, e_1, e_2, a_0, a_1, a_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(c_0, c_1, c_2, d_0, d_1, d_2, e_0, e_1, e_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(e_0, e_1, e_2, a_0, a_1, a_2, b_0, b_1, b_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(b_0, b_1, b_2, c_0, c_1, c_2, d_0, d_1, d_2, R_0, R_1, R_2)$

$d_0 \leftarrow d_0 \oplus r_0$

$d_1 \leftarrow d_1 \oplus r_1$

$d_2 \leftarrow d_2 \oplus r_2$

2^{nd}-order Masked KECCAK S-box

Name: χ_{3S}

In-/Output: $\{a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2, d_0, d_1, d_2, e_0, e_1, e_2, r_0, r_1, r_2, R_0, R_1, R_2\}$

$p_{\chi 3S}(r_0, r_1, r_2, e_0, e_1, e_2, a_0, a_1, a_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(c_0, c_1, c_2, d_0, d_1, d_2, e_0, e_1, e_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(e_0, e_1, e_2, a_0, a_1, a_2, b_0, b_1, b_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(b_0, b_1, b_2, c_0, c_1, c_2, d_0, d_1, d_2, R_0, R_1, R_2)$

$d_0 \leftarrow d_0 \oplus r_0$

$d_1 \leftarrow d_1 \oplus r_1$

$d_2 \leftarrow d_2 \oplus r_2$

2^{nd}-order Masked KECCAK S-box

In-/Output: $\{a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2, d_0, d_1, d_2, e_0, e_1, e_2, r_0, r_1, r_2\}$

$(R_0, R_1, R_2) \leftarrow (r_0, r_1, r_2) \ggg 1$

$p_{\chi 3S}(r_0, r_1, r_2, e_0, e_1, e_2, a_0, a_1, a_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(c_0, c_1, c_2, d_0, d_1, d_2, e_0, e_1, e_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(e_0, e_1, e_2, a_0, a_1, a_2, b_0, b_1, b_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$p_{\chi 3S}(b_0, b_1, b_2, c_0, c_1, c_2, d_0, d_1, d_2, R_0, R_1, R_2)$

$(R_0, R_1, R_2) \leftarrow (R_0, R_1, R_2) \ggg 1$

$r_0 \leftarrow r_0 \oplus R_0$

$r_1 \leftarrow r_1 \oplus R_1$

$r_2 \leftarrow r_2 \oplus R_2$

$d_0 \leftarrow d_0 \oplus r_0$

$d_1 \leftarrow d_1 \oplus r_1$

$d_2 \leftarrow d_2 \oplus r_2$

2nd-order Masked Ascon-p

- Ascon S-box is affine equivalent to Keccak S-box
  - Extension to Ascon-p is simple using shared linear/affine operations
- Additional SCA hardening through rotation offsets between shares
  - Ascon-p is rotation invariant
  - $x_0 = x_0$
    $x_1 = x_1 \ggg 5$
    $x_2 = x_2 \ggg 10$
  - Offsets need to be reversed during non-linear operations
  - Helps avoid transitions/glitches on stack or memory buses

Performance Evaluation

Processing one plaintext block in cycles/byte (X+0 encrypt long)

Implementation	ARM `STM32F303`	RISC-V Ibex
Plain	59	-
Leveled	89	-
2-shares	318	260*
3-shares	542	500*

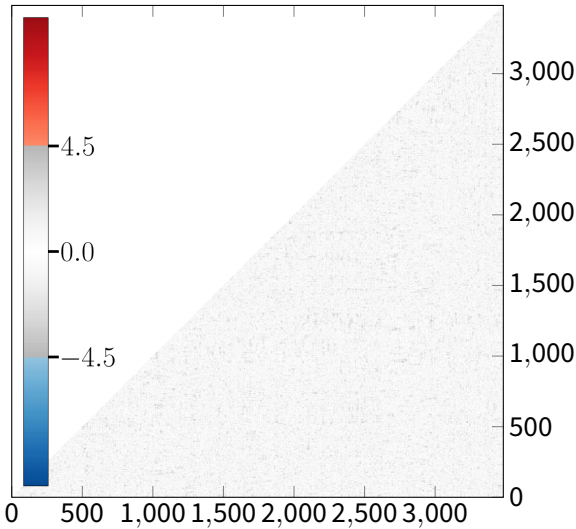
*Estimated based on cycle counts of linear and non-linear layer.

Practical Leakage Evaluation

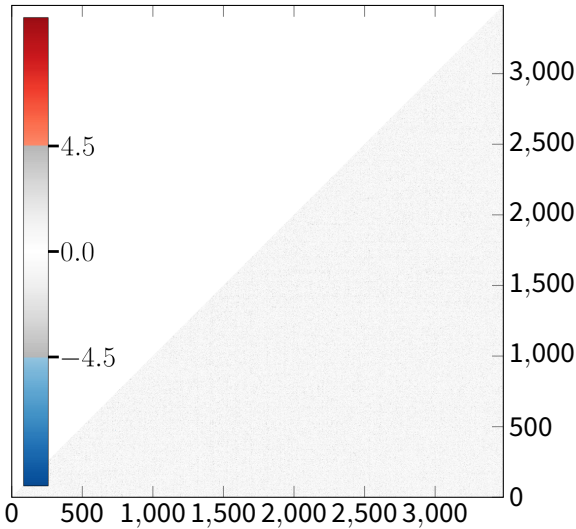
Testvector Leakage Assessment

- Goal: Practical 2st-order protection with 3 shares
- Evaluation setup:
  - ChipWhisperer-Lite
  - UFO Board
  - `STM32F303`
- Bivariate t-test scenarios:
  - 3 shares, 1 rounds, share-rotations, 4 samples/clk
  - 3 shares, 4 rounds, share-rotations, 1 samples/clk
  - 3 shares, 1 rounds, no share-rotations, 4 samples/clk
  - 2 shares, 1 rounds, share-rotations, 4 samples/clk

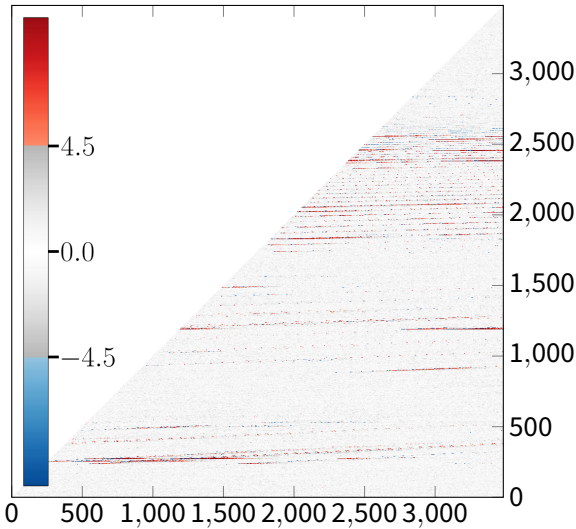
3 shares, 1 rounds, share-rotations, 4 samples/clk



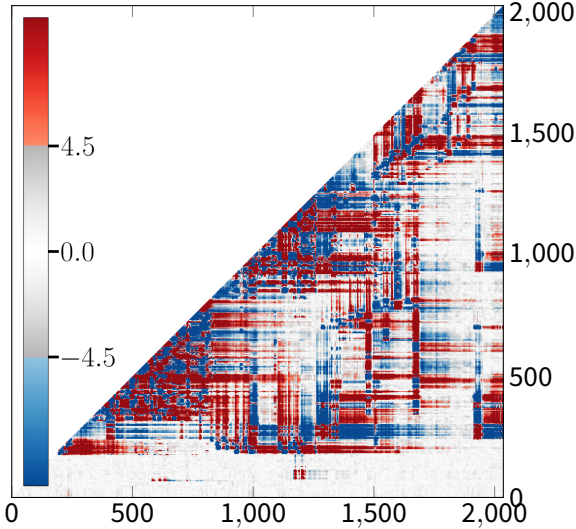
3 shares, 4 rounds, share-rotations, 1 samples/clk



3 shares, 1 rounds, no share-rotations, 4 samples/clk



2 shares, 1 rounds, share-rotations, 4 samples/clk



Formal Masking Verification

Formal Masking Verification

- Formal verification of masking in SW/HW using Coco [GHP+21]
- 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 Results

Implementation	Input Labels	Order	Stable		Transient	
			Result	Time	Result	Time
2-share Ascon-*p* round	$5 \times 64 \times 2$ bits	1	✓	3m	✓	5h 20m
3-share Ascon S-box	$5 \times 32 \times 3$ bits	2	✓	26m	✓	1h 17m

* Verification runtimes stem from single-threaded executions on an Intel Core i7 notebook processor with 16GB of RAM.

Questions

Bibliography I

[BBC+20] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. **Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography - A Practical Guide Through the Leakage-Resistance Jungle**. Advances in Cryptology - CRYPTO 2020. Vol. 12170. Lecture Notes in Computer Science. Springer, 2020, pp. 369–400. DOI: 10.1007/978-3-030-56784-2_13. URL: https://doi.org/10.1007/978-3-030-56784-2%5C_13.

[DDE+20] Joan Daemen, Christoph Dobraunig, Maria Eichlseder, Hannes Groß, Florian Mendel, and Robert Primas. **Protecting against Statistical Ineffective Fault Attacks**. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020.3 (2020), pp. 508–543.

[GHP+21] Barbara Gigerl, Vedad Hadzic, Robert Primas, Stefan Mangard, and Roderick Bloem. **Coco: Co-Design and Co-Verification of Masked Software Implementations on CPUs**. USENIX Security Symposium. USENIX Association, 2021, pp. 1469–1468.