Combined Masking and Shuffling for Side-Channel Secure Ascon on RISC-V

Linus Mainka, Kostas Papagiannopoulos

03-04-2025

Universiteit van Amsterdam

Setting

Ascon

- Selected by the U.S. NIST as the standard for lightweight cryptography (LWC)
- Extensively evaluated and found to be mathematically secure
- However: Side-Channel attacks are possible [MBA$^+$23, WP23]
  - Full key recovery through Correlation Power Analysis (CPA) with 8,000 traces
  - CPA using deep learning techniques: 1,000 traces
  - Partial key recovery from first-order masked implementation

Setting

The Goal

How can we create a software implementation of Ascon on a 32-bit architecture that should be side-channel secure by a comfortable margin?

How can we create a software implementation of Ascon on a 32-bit architecture that should be side-channel secure by a comfortable margin?

- Still retain security even if masking order is halved [BGG+14]
- Do not rely on a single countermeasure only
- Investigate multiple approaches
- Analyse security benefit through the Mutual Information (MI) framework [SMY06] using shortcut formulas [ABG+22]

Setting

Previous Work

Bitslice Masking and Improved Shuffling:
How and When to Mix Them in Software? by Azouaoui et al.:

- Mask first, then shuffle
- Three theoretical approaches for combining masking and shuffling
  - Shuffle Tuples
  - Shuffle Shares
  - Shuffle Everything

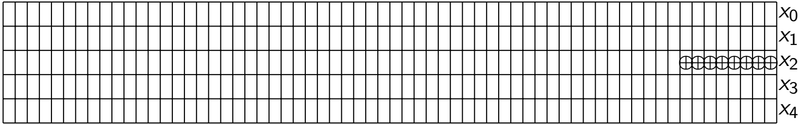
- Non-linear (AND) operations: ISW
- Provide Mutual Information (MI) shortcut formulas

Our contribution:

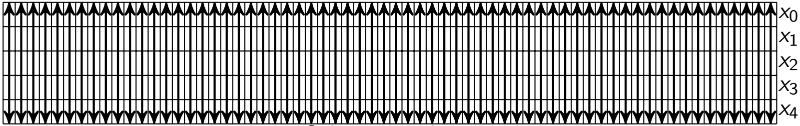
- Mask first, then shuffle
- Five Ascon implementations on 32-bit RISC-V for combining masking and shuffling
  - Shuffle Tuples
  - Shuffle Shares
  - Shuffle Everything "Light"
  - Unshuffled, but masked implementation
  - Levelled implementation
- Non-linear (AND) operations: PINI [CS20]
- Use Mutual Information (MI) shortcut formulas
- Third-order masking

Background

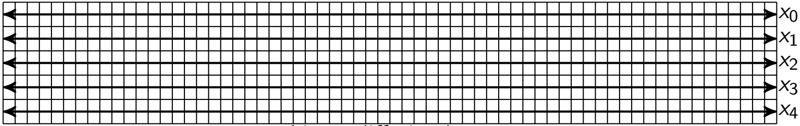
Ascon



Round constant addition



Substitution layer



Linear diffusion layer

- Split a value x into $d+1$ shares $x_0, \ldots x_d \rightarrow$ "d-th order masking"
  - $x_0, \ldots, x_{d-1}$ are random values $r_0, \ldots, r_{d-1}$
  - $x_d = x \oplus r_0 \oplus \cdots \oplus x_{d-1}$
- Perform each operation on all shares of x
- To obtain the original value, we can recombine all shares

Background

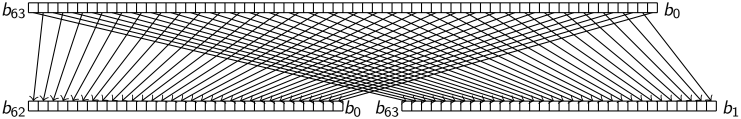
Shuffling

- Take a sequence of independent operations $[x_0 \circ y_0, \ldots, x_n \circ y_n]$
- Randomise the order in which they are executed according to a permutation θ
- At step i, perform the operation $x_{\theta_i} \circ y_{\theta_i}$

Background

Bit Interleaving

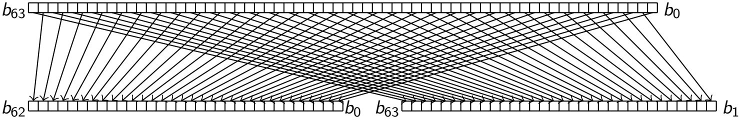
Splitting a 64-bit register into interleaved 32-bit registers:



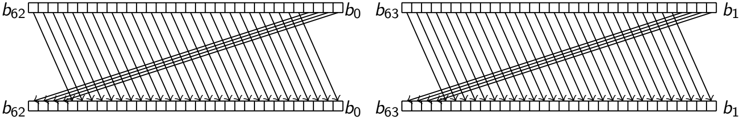
Background

Interleaving

Splitting a 64-bit register into interleaved 32-bit registers:

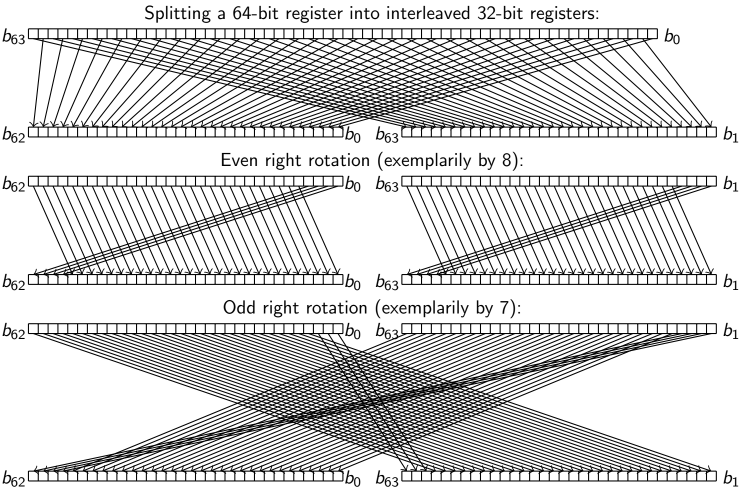


Even right rotation (exemplarily by 8):



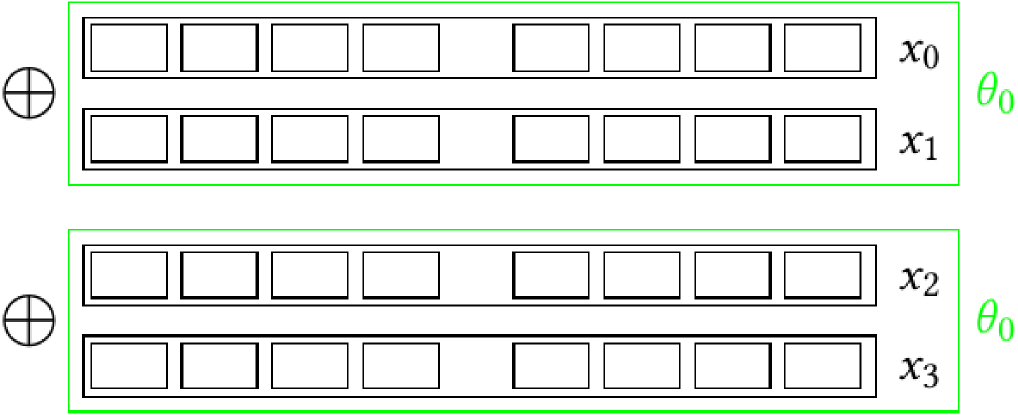
Background

Interleaving



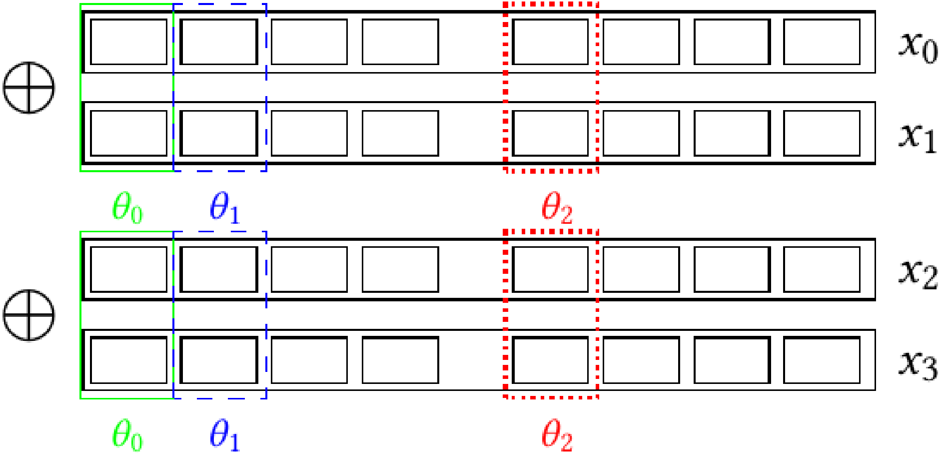
Shuffle Tuples

- Ignore Masking when shuffling
- Still shuffle entire operations



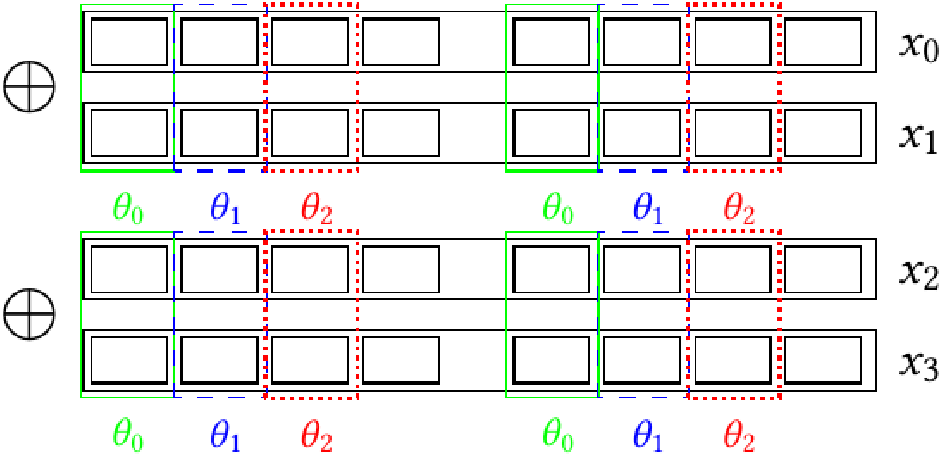
Shuffle Shares

- Instead of shuffling across operations, we shuffle across shares
- We do not shuffle across shares of the same value
- We shuffle across shares with the same index of different values



Shuffle Everything "Light"

- Adaptation of previous scheme
- Utilising the structure of bit interleaving



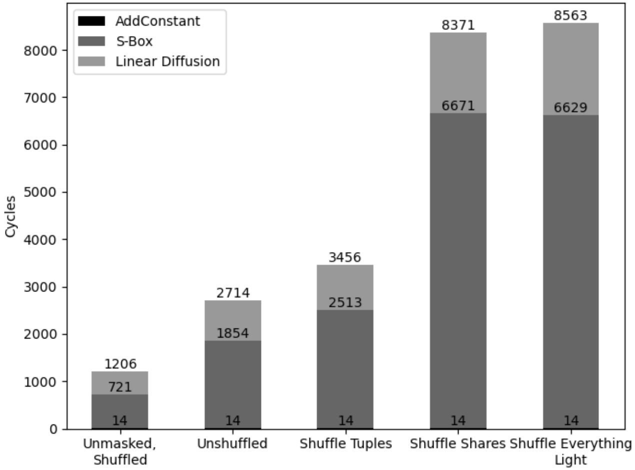
PINI AND

Algorithm 1 PINI AND gadget with linear memory requirements

Inputs: $a = [a_0, \ldots, a_d]$, $b = [b_0, \ldots, b_d]$
for $i = 0$ to d **do**
  $c_i \leftarrow a_i b_i$
end for
for $i = 0$ to d **do**
  for $j = i + 1$ to d **do**
    $r_{ij} \xleftarrow{\$} \mathbb{F}_{2^{32}}$; $r_{ji} \leftarrow r_{ij}$
    $z_{ij} = (a_i + 1) \cdot r_{ij} + a_i \cdot (b_j + r_{ij})$
    $z_{ji} = (a_j + 1) \cdot r_{ji} + a_j \cdot (b_i + r_{ji})$
    $c_i \leftarrow c_i + z_{ij}$
    $c_j \leftarrow c_j + z_{ji}$
  end for
end for
Outputs: $c = [c_0, \ldots, c_d]$ so that $c = a \wedge b$

Results

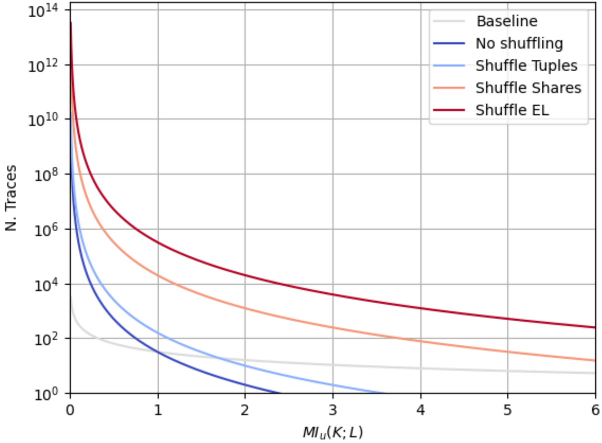
Performance



$d = 3$ for all masked schemes

Results

Mutual Information



$d = 3$ for all masked schemes

Results

Cycles vs. MI

	Unshuffled	Shuffle Tuples	Shuffle Shares	Shuffle EL
$d = 3$	0.0752	0.1125	0.376	0.7521

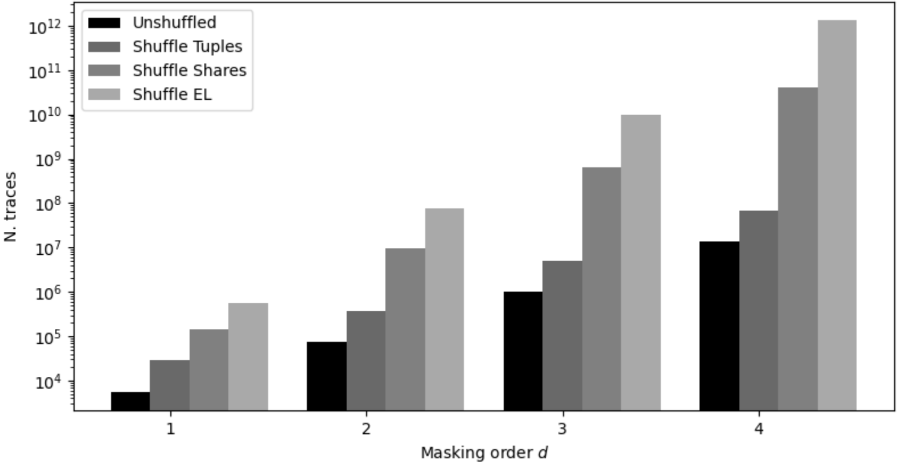
Table: The MI values per scheme so that an adversary needs 10^6 traces.

	Unshuffled	Shuffle Tuples	Shuffle Shares	Shuffle EL
	2,714	3,456	8,371	8,563

Table: The number of cycles needed to compute one round of the permutation.

Results

Traces vs. Masking Order



Conclusion

Takeaways

- Shuffle Shares and Shuffle EL are better than just increasing d
  (Assuming no shuffling permutation leakage)
- Benefit of Shuffle EL increases as register size goes down
- Implementation is Ascon-specific, the schemes are not

Conclusion

Takeaways

- Shuffle Shares and Shuffle EL are better than just increasing d
  (Assuming no shuffling permutation leakage)
- Benefit of Shuffle EL increases as register size goes down
- Implementation is Ascon-specific, the schemes are not

Caveats

- (Micro-)architectural effects will likely reduce the practical security
- Requires significant randomness
- No physical evaluation

Code:
`https://uva-hva.gitlab.host/l.mainka/side-channel-secure-ascon`

Conclusion

Takeaways

- Shuffle Shares and Shuffle EL are better than just increasing d *(Assuming no shuffling permutation leakage)*
- Benefit of Shuffle EL increases as register size goes down
- Implementation is Ascon-specific, the schemes are not

~~Caveats~~ Future Work

- (Micro-)architectural effects will likely reduce the practical security
- Requires significant randomness
- No physical evaluation

Code:
`https://uva-hva.gitlab.host/l.mainka/side-channel-secure-ascon`

References I

Melissa Azouaoui, Olivier Bronchain, Vincent Grosso, Kostas Papagiannopoulos, and François-Xavier Standaert.
Bitslice masking and improved shuffling: How and when to mix them in software?
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022(2):140–165, Feb. 2022.

Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert.
On the cost of lazy engineering for masked software implementations.
Cryptology ePrint Archive, Paper 2014/413, 2014.
https://eprint.iacr.org/2014/413.

Gaëtan Cassiers and François-Xavier Standaert.
Trivially and efficiently composing masked gadgets with probe isolating non-interference.
IEEE Transactions on Information Forensics and Security, 15:2542–2555, 2020.

Kamyar Mohajerani, Luke Beckwith, Abubakr Abdulgadir, Eduardo Ferrufino, Jens-Peter Kaps, and Kris Gaj.
Sca evaluation and benchmarking of finalists in the nist lightweight cryptography standardization process.
Cryptology ePrint Archive, Paper 2023/484, 2023.

Francois-Xavier Standaert, Tal G. Malkin, and Moti Yung.
A unified framework for the analysis of side-channel key recovery attacks (extended version).
Cryptology ePrint Archive, Paper 2006/139, 2006.
https://eprint.iacr.org/2006/139.

Léo Weissbart and Stjepan Picek.
Lightweight but not easy: Side-channel analysis of the ascon authenticated cipher on a 32-bit microcontroller.
Cryptology ePrint Archive, Paper 2023/1598, 2023.