A Closer Look at the S-box: Deeper Analysis of Round-Reduced ASCON-HASH

Xiaorui Yu[1], Fukang Liu[2],Gaoli Wang[1], Siwei Sun[3], Willi Meier[4]

[1]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China
[2]Tokyo Institute of Technology, Tokyo, Japan
[3]School of Cryptology, University of Chinese Academy of Sciences, Beijing, China
[4]FHNW, Windisch, Switzerland

2023.8.16

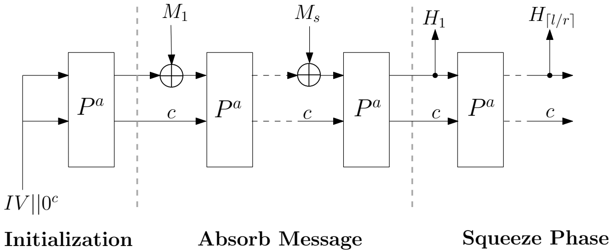
Overview

Lightweight Cryptography Standard

- In 2013, NIST started the lightweight cryptography project.
- In 2016, NIST provided an overview of the project and decided to seek for some new algorithms as a lightweight cryptography standard.
- In 2019, NIST received 57 submissions and 56 of them became the first round candidates after the initial review.
- On February 7, 2023, NIST announced the selection of the ASCON family for the lightweight cryptography standardization.

ASCON-HASH

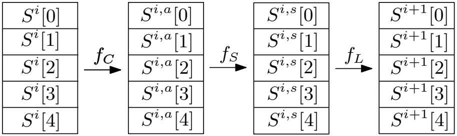
- ASCON-HASH is one of the hash functions provided by ASCON.
- Sponge-based construction
- 320-bit state (r=64,c=256)
- 256-bit hash value



Round Function of ASCON-HASH

- Round function

$$S^i \xrightarrow{f_C} S^{i,a} \xrightarrow{f_S} S^{i,s} \xrightarrow{f_L} S^{i+1}$$



- $S^{i,a} = S^i[0]||S^i[1]||S^i[2] \oplus C_i||S^i[3]||S^i[4]$
- $S^{i,s} =$S-box$(S^{i,a})$
- $S^{i+1} = \Sigma_0(S^{i,s}[0])||\Sigma_1(S^{i,s}[1])||\Sigma_2(S^{i,s}[2])||\Sigma_3(S^{i,s}[3])||\Sigma_4(S^{i,s}[4])$

S-box and Linear Diffusion of ASCON-HASH

- 5-bit S-box for each 5-bit column.

$$\begin{cases} y_0 = x_4x_1 \oplus x_3 \oplus x_2x_1 \oplus x_2 \oplus x_1x_0 \oplus x_1 \oplus x_0, \\ y_1 = x_4 \oplus x_3x_2 \oplus x_3x_1 \oplus x_3 \oplus x_2x_1 \oplus x_2 \oplus x_1 \oplus x_0, \\ y_2 = x_4x_3 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1, \\ y_3 = x_4x_0 \oplus x_4 \oplus x_3x_0 \oplus x_3 \oplus x_2 \oplus x_1 \oplus x_0, \\ y_4 = x_4x_1 \oplus x_4 \oplus x_3 \oplus x_1x_0 \oplus x_1. \end{cases}$$

- 5 independent linear functions for each line (64-bit word).

$$\begin{cases} X_0 \leftarrow \Sigma_0(X_0) = X_0 \oplus (X_0 \ggg 19) \oplus (X_0 \ggg 28), \\ X_1 \leftarrow \Sigma_1(X_1) = X_1 \oplus (X_1 \ggg 61) \oplus (X_1 \ggg 39), \\ X_2 \leftarrow \Sigma_2(X_2) = X_2 \oplus (X_2 \ggg 1) \oplus (X_2 \ggg 6), \\ X_3 \leftarrow \Sigma_3(X_3) = X_3 \oplus (X_3 \ggg 10) \oplus (X_3 \ggg 17), \\ X_4 \leftarrow \Sigma_4(X_4) = X_4 \oplus (X_4 \ggg 7) \oplus (X_4 \ggg 41). \end{cases}$$

Linear function and S-box

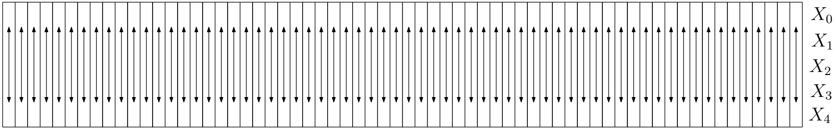


Figure: S-box

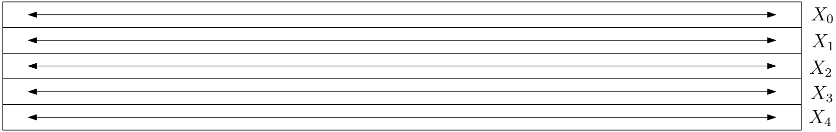


Figure: Linear Function

Notations

Table: Notations

r	the length of the rate part for ASCON-HASH, $r = 64$
c	the length of the capacity part for ASCON-HASH, $c = 256$
S_j^i	the input state of round i when absorbing the message block M_j
$S^i[j]$	the j-th word (64-bit) of S_i
$S^i[j][k]$	the k-th bit of $S^i[j]$, $k = 0$ means the least significant bit and k is within modulo 64
x_i	the i-th bit of a 5-bit value x, x_0 represents the most significant bit
M	message
M_i	the i-th block of the padded message
$\ggg$	right rotation (circular right shift)
$a\%b$	$a \mod b$
0^n	a string of n zeroes

Collision Attacks on ASCON-HASH

Table: Summary of collision attacks on ASCON-HASH

Attack Type	Rounds	Time complexity	Memory Complexity	Reference
collision attack	2	2^{125*}	negligible	[1]
	2	2^{103}	negligible	[2]
	2	$2^{62.6}$	negligible	This paper.
	3	$2^{121.85}$	2^{121}	[3]
	4	$2^{126.77}$	2^{126}	[3]

* The characteristic used is invalid.

[1] Rui Zong, Xiaoyang Dong, and Xiaoyun Wang. *Collision Attacks on Round-Reduced GIMLI-HASH/ASCON-XOF/ASCON-HASH*. Cryptology ePrint Archive, Paper 2019/1115. https://eprint.iacr.org/2019/1115. 2019. URL: https://eprint.iacr.org/2019/1115.

[2] David Gérault, Thomas Peyrin, and Quan Quan Tan. "Exploring Differential-Based Distinguishers and Forgeries for ASCON". In: *IACR Trans. Symmetric Cryptol.* 2021.3 (2021), pp. 102–136. DOI: 10.46586/tosc.v2021.i3.102-136. URL: https://doi.org/10.46586/tosc.v2021.i3.102-136.

[3] Lingyue Qin et al. *Weak-Diffusion Structure: Meet-in-the-Middle Attacks on Sponge-based Hashing Revisited*. Cryptology ePrint Archive, Paper 2023/518. https://eprint.iacr.org/2023/518. 2023. URL: https://eprint.iacr.org/2023/518.

Basic Attack Strategy for Sponge-based Hash Functions

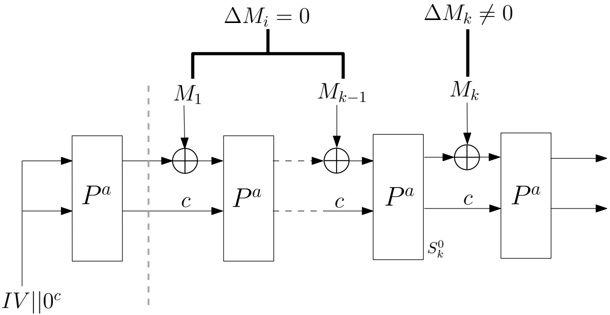
■Requirements for differential characteristic:

- For input difference, only non-zero difference in rate part.
- For output difference, the same as above.
- Active S-boxes should be as few as possible in the whole characteristic.

General 2-step attack framework.

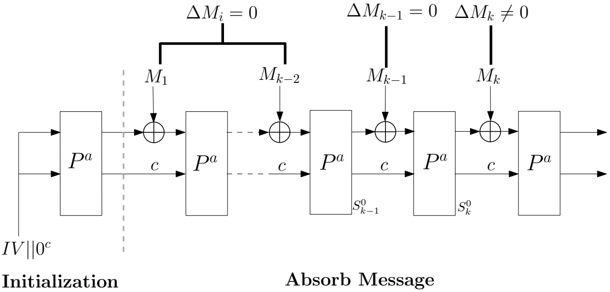
■Suppose that there are n_c bit conditions on the capacity part of S_k^0 and the remaining conditions hold with probability 2^{-n_k}.

- Step1: Find a solution of $(M_1, \ldots, M_{k-1})$ such that the n_c bit conditions on the capacity part of S_k^0 can hold.
- Step2: Exhaust M_k and check whether remaining n_k bit conditions can hold. If there is a solution, a collision is found. Otherwise, return to Step 1.



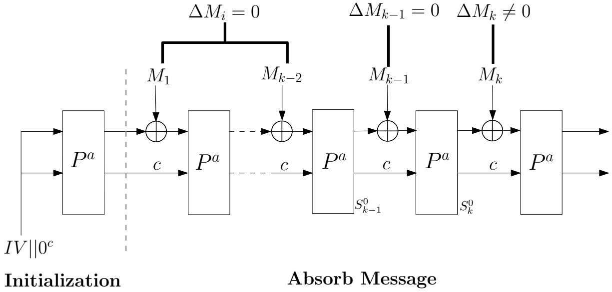
General 3-step attack strategy

- Main idea: Further convert the n_c conditions on the capacity part of S_k^0 into some n_c^1 conditions on the capacity part of S_{k-1}^0.



General 3-step attack strategy

- Step 1: Find a solution of $(M_1, \ldots, M_{k-2})$ such that the n_c^1 bit conditions on the capacity part of S_{k-1}^0 can hold.
- Step 2: Enumerate all the solutions of M_{k-1} such that the conditions on the capacity part of S_k^0 can hold.
- Step 3: Exhaust M_k and check whether remaining n_k bit conditions can hold. If there is a solution, a collision is found. Otherwise, return to Step 1.



Time complexity estimation

- The time complexity of Step 1, 2 and 3 is denoted as $T_{\texttt{pre1}}$, $T_{\texttt{k-1}}$ and $T_{\texttt{k}}$.
  - The general complexity estimation:

    $$T_{\texttt{total}} = (k-2) \cdot 2^{n_k+n_c-2r} \cdot T_{\texttt{pre1}} + 2^{n_k+n_c-2r} \cdot T_{\texttt{k-1}} + 2^{n_k-r} \cdot T_{\texttt{k}}.$$

  - To optimize $T_{\texttt{pre1}}$ as $T_{\texttt{pre1}} = 2^{n'_c}$, we can improve this complexity as below, where n'_c refers to the number of the conditions on S^0_{k-1}, converted from those n^1_c conditions on S^0_k.

    $$T_{\texttt{total}} = (k-2) \cdot 2^{n_k+n_c+n'_c-2r} + 2^{n_k+n_c-2r} \cdot T_{\texttt{k-1}} + 2^{n_k-r} \cdot T_{\texttt{k}}.$$

Algebraic properties of the S-box

■ With special input and output differences, we can get some linear conditions from the ANF of the S-box.

$$\begin{cases} y_0 = x_4x_1 \oplus x_3 \oplus x_2x_1 \oplus x_2 \oplus x_1x_0 \oplus x_1 \oplus x_0, \\ y_1 = x_4 \oplus x_3x_2 \oplus x_3x_1 \oplus x_3 \oplus x_2x_1 \oplus x_2 \oplus x_1 \oplus x_0, \\ y_2 = x_4x_3 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1, \\ y_3 = x_4x_0 \oplus x_4 \oplus x_3x_0 \oplus x_3 \oplus x_2 \oplus x_1 \oplus x_0, \\ y_4 = x_4x_1 \oplus x_4 \oplus x_3 \oplus x_1x_0 \oplus x_1. \end{cases}$$

Algebraic properties of the S-box

Property 1 For an input difference $(\Delta_0, \ldots, \Delta_4)$ satisfying $\Delta x_1 = \Delta x_2 = \Delta x_3 = \Delta x_4 = 0$ and $\Delta x_0 = 1$, the following constraints hold:

- For the output difference:

$$\begin{cases} \Delta y_0 \oplus \Delta y_4 = 1, \\ \Delta y_1 = \Delta x_0, \\ \Delta y_2 = 0. \end{cases} \tag{1}$$

- For the input value:

$$\begin{cases} x_1 = \Delta y_0 \oplus 1, \\ x_3 \oplus x_4 = \Delta y_3 \oplus 1. \end{cases} \tag{2}$$

Bit Conditions from Difference

Table: The 2-round differential characteristic.

ΔS^0 (2^{-54})	ΔS^1 (2^{-102})	ΔS^2
0xbb450325d90b1581	0x2201080000011080	0xbaf571d85e1153d7
0x0	0x2adf0c201225338a	0x0
0x0	0x0	0x0
0x0	0x0000000100408000	0x0
0x0	0x2adf0c211265b38a	0x0

- Note:
  - Totally 4 message blocks will be used.
  - Totally 54 bit conditions on S^0.
  - 27 on $S^0[1]$ and 27 on $S^0[3] \oplus S^0[4]$.

Bit conditions on S^1

We further study the 28 active S-boxes in the second round. We observe that from ΔS^1 to $\Delta S^{1,s}$, there are only 3 different possible difference transitions $(\Delta x_0, \ldots, \Delta x_4) \rightarrow (\Delta y_0, \ldots, \Delta y_4)$ through the S-box, as shown below:

$$
\begin{aligned}
(1,1,0,0,1) &\rightarrow (1,0,0,0,0), \\
(0,0,0,1,1) &\rightarrow (1,0,0,0,0), \\
(0,1,0,0,1) &\rightarrow (1,0,0,0,0).
\end{aligned}
$$

Bit Conditions from Difference

Table: The 2-round differential characteristic.

ΔS^0 (2^{-54})	ΔS^1 (2^{-102})	ΔS^2
0xbb450325d90b1581	0x2201080000011080	0xbaf571d85e1153d7
0x0	0x2adf0c201225338a	0x0
0x0	0x0	0x0
0x0	0x0000000100408000	0x0
0x0	0x2adf0c211265b38a	0x0

- Note:
  - Totally 102 bit conditions on S^1.
  - 21 on $S^1[2]$.

Algebraic properties of the S-box

■ Carefully, after the capacity part of S_3^0 is fixed, $S^1[2]$ is independent to $S^0[0]$ since

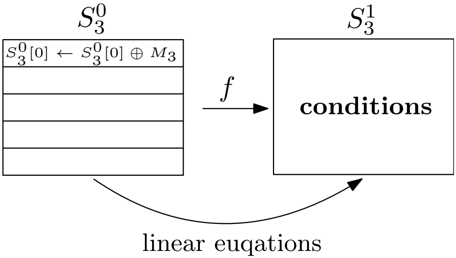
$$y_2 = x_4 x_3 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1.$$

.

- After calculation, there are 21 such conditions on $S^1[2]$.
- So apart from the 54 linear conditions on the capacity part of S^0, it needs to add 21 nonlinear conditions on it.
- As a result, the linear conditions on S^1 reduced to 81.

Optimize Ehausting M_3

Now we don't need to exhaust message pairs (M_3, M_3'). With 81 linear conditions, we can establish 81 linear equations for M_3.



Property 2

For $(y_0, \ldots, y_4) = \texttt{SB}(x_0, \ldots, x_4)$, if $x_3 \oplus x_4 = 1$, y_3 will be independent of x_0.

Proof.

We can rewrite y_3 as follows:

$$y_3 = (x_4 \oplus x_3 \oplus 1)x_0 \oplus (x_4 \oplus x_3 \oplus x_2 \oplus x_1).$$

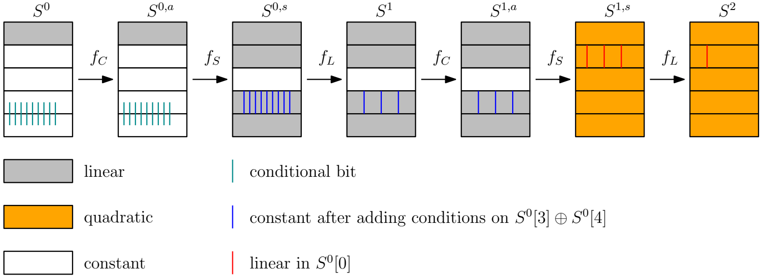
Hence, if $x_3 \oplus x_4 = 1$, y_3 is independent of x_0. $\square$

Property 3

Let

$$(S^1[0], \ldots, S^1[4]) = f(S^0[0], \ldots, S^0[4]),$$
$$(S^2[0], \ldots, S^2[4]) = f(S^1[0], \ldots, S^1[4]),$$

where $(S^0[1], S^0[2], S^0[3], S^0[4])$ are constants and $S^0[0]$ is the only variable. Then, it is always possible to make u bits of $S^2[1]$ linear in $S^0[0]$ by adding at most $9u$ bit conditions on $S^0[3] \oplus S^0[4]$.

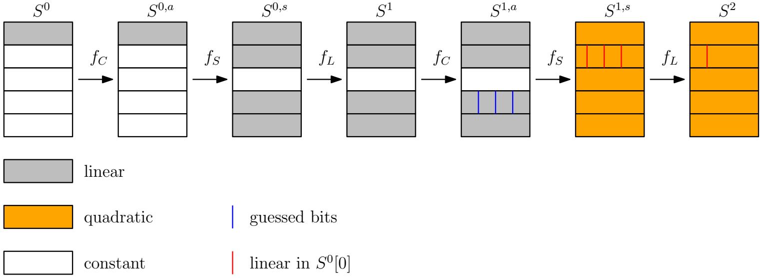


Property 4

Let

$$(S^1[0], \ldots, S^1[4]) = f(S^0[0], \ldots, S^0[4]),$$
$$(S^2[0], \ldots, S^2[4]) = f(S^1[0], \ldots, S^1[4]),$$

where $(S^0[1], S^0[2], S^0[3], S^0[4])$ are constants and $S^0[0]$ is the only variable. Then, it is always possible to make u bits of $S^2[1]$ linear in $S^0[0]$ by guessing $3u$ linear equations in $S^0[0]$.



The Framework of Improving the Attack

- Assume that the capacity part of S_2^0 is known.
  1. Add $9u_1$ conditions on the capacity part of $S_2^0 \Longrightarrow u_1$ bits of $S_3^0[1]$ can be linear in M_2.
  2. Guess $3u_2$ linear equations in $M_2 \Longrightarrow u_2$ bits of $S_3^0[1]$ can be linear in M_2.
  3. Set up $u_1 + 4u_2$ linear equations in 64 variables to satisfy $u_1 + u_2$ out of the original 27 bit conditions.
  4. Apply Gaussian elimination on these $u_1 + 4u_2$ linear equations and obtain

$$u_3 = 64 - u_1 - 4u_2$$

     free variables.

Improve Exhausting M_2

1. Guess $3u_2 = 42$ bits of M_2 and construct $4u_2 + u_1$ linear equations.
2. Apply the Gaussian elimination to the system and obtain $u_3 = 64 - u_1 - 4u_2$ free variables.
3. Construct $54 - u_1 - u_2$ quadratic equations in these u_3 variables and solve the equations.
4. Check whether the remaining 21 quadratic conditions on the capacity part of S_3^0 can hold for each obtained solution.

The Optimal Guessing Strategy

- Assume that one round of the ASCON permutation takes about $15 \times 64 \approx 2^{10}$ bit operations
- The optimal choice of (u_1, u_2, u_3) is as follows:

$$u_1 = 3, \quad u_2 = 13 \quad u_3 = 9.$$

- The total time complexity can be estimated as

$$T_{\texttt{total}} = 2^{28} \times 2^{27} + 2^{28} \times 2^{56.6-11} + 2^{17} \times 2^{19-11} \approx 2^{73.6}$$

calls to the 2-round ASCON permutation.

Further Improving.

- The core problem is to make

$$(S_2^1[3][i], S_2^1[3][i+61], S_2^1[3][i+39])$$

constant by either guessing their values or adding bit conditions on $S_2^0[3] \oplus S_2^0[4]$.

So for the same conditional bit, we can use a hybrid guessing strategy.

Further Improving

- Add u_4 conditions on $S_2^0[3] \oplus S_2^0[4]$ and guess u_5 bits of $S_2^1[3]$.
- Set up u_6 linear equations for u_6 conditional bits of $S_2^2[1]$.
- We have in total $u_5 + u_6$ linear equations.
- After the Gaussian elimination, we can set up $54 - u_6$ quadratic equations in $u_7 = 64 - u_5 - u_6$ free variables.

Result: We propose to choose

$$u_4 = 31, \quad u_5 = 28, \quad u_6 = 27$$

The new total time complexity is

$$T_{\texttt{total}} = 2^{28} \times 2^{31} + 2^{28} \times 2^{28} \times (2^{17.6} + 2^{15.3}) \times 2^{-11} + 2^{17} \times 2^{19-11} \approx 2^{62.6}$$

hash function calls.

Conclusion and Future work

- The attack complexity is reduced from 2^{103} to $2^{62.6}$ hash function calls.
- The complexity of the attack is greatly related to the differential characteristic.
- Finding the better characteristic and make the time complexity more practical will be token as our future work.
- Studying more underlying properties of the round functions.