IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2016, No. 1, pp. 1–12. DOI:10.13154/tosc.v2016.i1.1-12

Cryptanalysis of Haraka

Jérémy Jean

Agence nationale de la sécurité des systèmes d'information (ANSSI) Crypto Lab,
51, boulevard de La Tour-Maubourg 75700 Paris 07 SP, France
Jeremy.Jean@ssi.gouv.fr

Abstract. In this paper, we describe attacks on the recently proposed Haraka hash functions. First, for the two hash functions Haraka-256/256 and Haraka-512/256 in the family, we show how two colliding messages can be constructed in about 2^{16} function evaluations. Second, we invalidate the preimage security claim for Haraka-512/256 with an attack finding one preimage in about 2^{192} function evaluations. These attacks are possible thanks to symmetries in the internal state that are preserved over several rounds.

Keywords: Hash Function · Preimage Attack · Collision Attack · Haraka

1 Introduction

We analyze in this paper the recent Haraka hash function proposed in [Ste16] and presented at the FSE 2016 rump session. It has been designed by Kölbl, Lauridsen, Mendel and Rechberger, and relies on an AES-based permutation π in Davies-Meyer mode. The main security goal is to provide (second) preimage resistance, while at the same time only considering very short inputs. The rationale of the proposals consists of cryptographic applications where the collision resistance is not required. However, the designers nevertheless claim that 128-bit security for collision resistance can be achieved with a slightly stronger inner permutation π.

In the specifications of the design, the designers put a lot of effort to analyze the capabilities of an attacker applying the rebound attack [MRST09]. This technique has been used extensively during the SHA-3 competition to analyze the security of hash functions, e.g. on the finalists Grøstl [JNP12], Keccak [DGPW12], JH [NTV11] and Skein [KNR10]. However, other vectors of attacks are not discussed.

Our Contributions. In the remaining of the document, we present two different attacks on the Haraka hash functions that break the security claims of the proposals for preimage and collision resistances. The main observation for both attacks uses symmetric properties of the keyless AES round function that are not prevented in Haraka due to highly structured round constants. This kind of structural weakness has already been used in the past against some primitives, for instance the submission PAES [JNSW14] to the CAESAR competition [Ber] in [JNSW16], or the lightweight block cipher Midori [BBI+15] in [GJN+15].

We first present in Section 4 collision attacks for the two hash functions Haraka-256/256 and Haraka-512/256. The pair of colliding messages is constructed in about 2^{16} operations. We additionally give concrete examples of 5-collisions produced for the public reference implementations; that is, sets $\{m_1, \dots, m_5\}$ of inputs such that $h(m_i) = h(m_j)$ for all (i, j), h being one of the Haraka hash functions (Appendix A).

Then in Section 5, we show how Haraka-512/256 can be inverted in 2^{192} computations, saving a factor 2^{64} over exhaustive search. The strategy also uses a 3-round distinguishing

Received: 2016-06-01, Accepted: 2016-08-01, Published: 2016-12-01

property of the underlying permutation, which can be extended to a preimage attack by considering the partial diffusion of 1.5 round of AES.

2 Specifications of Haraka

We recall here the specifications of the two hash functions of the Haraka family; namely: Haraka-256/256 and Haraka-512/256. For more details, we refer to the submission document [Ste16].

Unlike general cryptographic hash functions, Haraka only handles short messages, so that the input space is finite. Namely, the signatures of the two functions are given by:

$$\texttt{Haraka-256/256} : GF(2^{256}) \longrightarrow GF(2^{256}),$$
$$\texttt{Haraka-512/256} : GF(2^{512}) \longrightarrow GF(2^{256}).$$

The rationale behind this choice pertains to most applications that only process short inputs and do not require collision resistance.

Both functions uses an inner permutation in a Davies-Meyer mode (see Figure 1): π_{256} for Haraka-256/256 acting on 256-bit values, and π_{512} for Haraka-512/256 acting on 512-bit values.

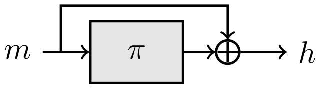

Figure 1: Haraka uses the Davies-Meyer construction with a permutation π.

The internal states are comprised of b AES states, with $b = 2$ for Haraka-256/256 and $b = 4$ for Haraka-512/256. Both functions output a 256-bit hash value, where Haraka-512/256 truncates the final internal state by keeping only eight columns at predetermined positions (see Figure 2).

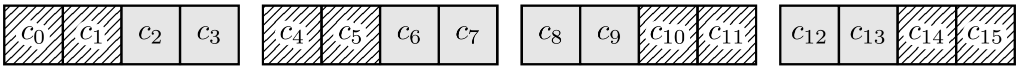

Figure 2: Truncation in Haraka-512/256: the dashed columns are truncated away. Each column c_i contains four bytes.

One step of each construction applies two AES rounds and is depicted in Figure 3 and Figure 4. The function A on the figures applies one round of keyless AES. The round constants RC_{2i} and RC_{2i+1} are injected during Round i, and are detailed in the subsequent Section 3. The linear layers ending each step function that permute the AES column is called $\mathbf{mix}_{256}$ in π_{2656} and $\mathbf{mix}_{512}$ in π_{512}. For completeness, we give:

$$\begin{aligned}\mathbf{mix}_{256}(c_0, \ldots, c_7) &= (c_0, c_4, c_1, c_5, c_2, c_6, c_3, c_7)\\ \mathbf{mix}_{512}(c_0, \ldots, c_{15}) &= (c_3, c_{11}, c_7, c_{15}, c_8, c_0, c_{12}, c_4,\\ &\qquad c_9, c_1, c_{13}, c_5, c_2, c_{10}, c_6, c_{14}).\end{aligned}$$

Security Claims. In the submission document [Ste16], the designers claim that step functions with *five* steps allows to achieve (second) preimage resistance for both functions. Additionally, they claim that using a stronger permutation using *six* steps instead of five allows to achieve 128-bit security against collision attacks for both functions.

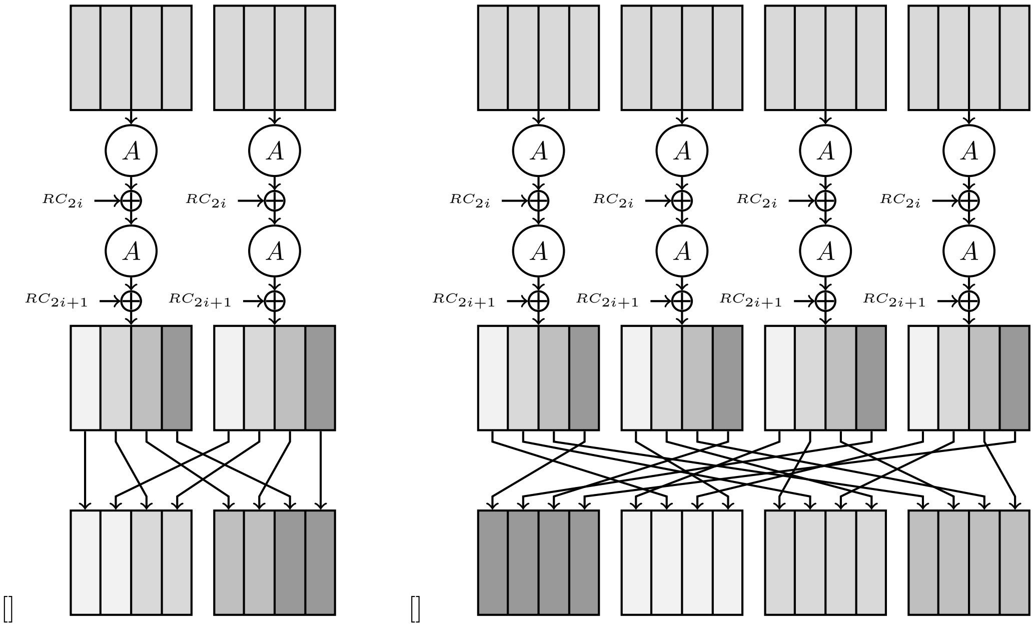

Figure 4: `Haraka-512/256` step function.

3 Symmetries in `Haraka`

Before describing the actual attacks, we briefly recall the symmetries of the keyless `AES` round function and describe how it generalizes to the `Haraka` step function.

3.1 Preliminaries: the Case of `AES`

The observation that we recall here has first been established in [LSWD04] and considers the *keyless* `AES` round function. It can be stated as follows: given a state X such that its left and right halves are equal, then the keyless `AES` round function maintains this property. We omit the proof which simply tracks the behavior of each transformation; namely, SubBytes, ShiftRows and MixColumns. We define in particular an even more retricted case, when all four columns of a state are equal (see Figure 5). We call such a state *strongly symmetric.*

$$\begin{array}{|c|c|c|c|}\hline a&a&a&a\\\hline b&b&b&b\\\hline c&c&c&c\\\hline d&d&d&d\\\hline\end{array} \xrightarrow{\mathsf{SB}} \begin{array}{|c|c|c|c|}\hline a'&a'&a'&a'\\\hline b'&b'&b'&b'\\\hline c'&c'&c'&c'\\\hline d'&d'&d'&d'\\\hline\end{array} \xrightarrow{\mathsf{SR}} \begin{array}{|c|c|c|c|}\hline a'&a'&a'&a'\\\hline b'&b'&b'&b'\\\hline c'&c'&c'&c'\\\hline d'&d'&d'&d'\\\hline\end{array} \xrightarrow{\mathsf{MC}} \begin{array}{|c|c|c|c|}\hline a''&a''&a''&a''\\\hline b''&b''&b''&b''\\\hline c''&c''&c''&c''\\\hline d''&d''&d''&d''\\\hline\end{array}$$

Figure 5: An `AES` state with four equal columns is called strongly symmetric, and stays strongly symmetric after the application of (any number of) the keyless `AES` round function.

3.2 Symmetry for π_{256} and π_{512}

We observe that in the two permutations π_{256} and π_{512}, the round constant RC_i injected in place of the `AES` subkey at Round i is highly structured. Namely, the round constants

RC_i consist of strongly symmetric states as previously defined. For instance, the two first subkeys equal:

$$RC_0 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad RC_1 = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Subsequent constants RC_j are deduced from RC_{j-1} by applying the same function to each column of the state, hence preserving the strong symmetry.

Since in the `Haraka` permutation π_{256} (resp. π_{512}), the columns of state are reordered by the $\mathbf{mix}_{256}$ (resp. $\mathbf{mix}_{512}$) linear layer, the strong symmetric property is also maintained by the `Haraka` step functions.

Consequently, starting with a strongly symmetric state yields a strongly symmetric state after any number of steps of the two permutations π_{256} and π_{512}. We note that there are a total of 2^{32} different strongly symmetric states.

3.3 Symmetry for π_{512}

We now describe another property for the large permutation π_{512}. Again, it relies on internal symmetries but only covers three steps. Our main goal is to increase the number of states that verify some symmetric property over three steps of π_{512}. Both classes presented has been found by hand by careful analysis of the step functions.

Consider the state described by eight independent columns $c_0, \ldots, c_7$ represented in the first line of Figure 6. This figure also shows how this state is transformed after three steps of π_{512}. We note that there are 2^{256} internal states of this structure.

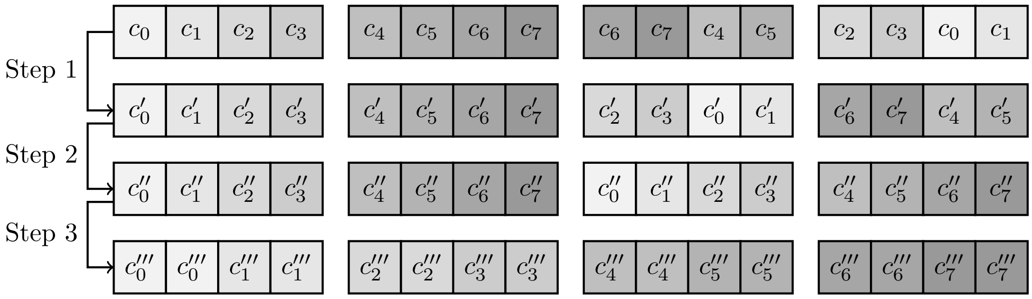
Step 1
Step 2
Step 3
c_0 c_1 c_2 c_3 | c_4 c_5 c_6 c_7 | c_6 c_7 c_4 c_5 | c_2 c_3 c_0 c_1
c'_0 c'_1 c'_2 c'_3 | c'_4 c'_5 c'_6 c'_7 | c'_2 c'_3 c'_0 c'_1 | c'_6 c'_7 c'_4 c'_5
c''_0 c''_1 c''_2 c''_3 | c''_4 c''_5 c''_6 c''_7 | c''_0 c''_1 c''_2 c''_3 | c''_4 c''_5 c''_6 c''_7
c'''_0 c'''_0 c'''_1 c'''_1 | c'''_2 c'''_2 c'''_3 c'''_3 | c'''_4 c'''_4 c'''_5 c'''_5 | c'''_6 c'''_6 c'''_7 c'''_7

Figure 6: In `Haraka-512/256`, an internal state verifying the column-wise symmetry of the first line maintains some symmetry after three steps of the permutation π_{512}.

The first two steps essentially boil down to the following property: two states S and S' that have equal halves but swapped have the same structure after one round of keyless `AES`. More precisely, let

$$S_1 = \begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{12} & b_{15} \end{bmatrix} \quad \text{and} \quad S_2 = \begin{bmatrix} b_8 & b_{12} & b_0 & b_4 \\ b_9 & b_{13} & b_1 & b_5 \\ b_{10} & b_{14} & b_2 & b_6 \\ b_{12} & b_{15} & b_3 & b_7 \end{bmatrix},$$

then the states after the application of $f = \mathsf{ShiftRows} \circ \mathsf{SubBytes}$ can be written as:

$$f(S_1) = \begin{bmatrix} b'_0 & b'_4 & b'_8 & b'_{12} \\ b'_5 & b'_9 & b'_{13} & b'_1 \\ b'_{10} & b'_{14} & b'_2 & b'_6 \\ b'_{15} & b'_3 & b'_7 & b'_{12} \end{bmatrix} \quad \text{and} \quad f(S_2) = \begin{bmatrix} b'_8 & b'_{12} & b'_0 & b'_4 \\ b'_{13} & b'_1 & b'_5 & b'_9 \\ b'_2 & b'_6 & b'_{10} & b'_{14} \\ b'_7 & b'_{12} & b'_{15} & b'_3 \end{bmatrix},$$

where the two halves are still equal, which is maintained by the subsequent MixColumns ($b'_i = \mathsf{S}(b_i)$ for all i, with S the AES Sbox). The round constant addition does not change this, and the same procedure is repeated a second time during this step. The last $\mathbf{mix}_{512}$ transformation finally reorders the columns and creates two new pairs of states with swapped halves.

After three steps (see Figure 6), the last the $\mathbf{mix}_{512}$ stops the process, and the internal state is comprised of adjacent pairs of equal columns (last line of Figure 6).

4 Collision Attack on Haraka

Distinguisher. In the previous section, we have seen that the image of a strongly symmetric state by π_{256} and π_{512} is also strongly symmetric. The simple observation that the feed-forward XOR of Davies-Meyer maintains this property allows to trivially distinguish Haraka-256/256.

For Haraka-512/256, we additionally note that the final truncation is performed column-wise, so that the observation also provides an efficient distinguisher for Haraka-512/256.

Collisions. Based on the previous distinguisher, we therefore expect to collide on the 256-bit hash values after trying approximately 2^{16} strongly symmetric messages drawn from the reduced 32-bit input space. We give such values in Table 1 below, produced from the public C implementation of Haraka [Ste].[1]

The input space being small, we also provide examples of 5-collisions generated by exhausting over the 2^{32} input space of strongly symmetric messages. We recall the result from [STKT06] which states that a t-way collision on a random n-bit map is expected after $(t!)^{1/t} \cdot 2^{n(t-1)/t}$ function evaluations. There are a total of 31 different 5-collisions that we list in Appendix A. There is no 6-collision or more.

Table 1: Examples of colliding messages for Haraka. The messages m and their outputs $h(m)$ consist of n equal blocks of 32-bit, with $n = 8$ for $h =$ Haraka-256/256 and $n = 16$ for $h =$ Haraka-512/256.

m	$h(m)$
000325eb...000325eb	9ca2e0a5...9ca2e0a5
00016734...00016734	9ca2e0a5...9ca2e0a5
0b1bdced...0b1bdced	04db0fe0...04db0fe0
0a0ef844...0a0ef844	04db0fe0...04db0fe0
0964701e...0964701e	04db0fe0...04db0fe0
0387c94b...0387c94b	04db0fe0...04db0fe0
016549d0...016549d0	04db0fe0...04db0fe0

5 Preimage Attack on Haraka-512/256

We continue the analysis of Haraka by presenting a preimage attack on Haraka-512/256. The main idea also uses symmetries, but with the larger equivalence class presented in Section 3.3. Indeed, in the case of collisions, we want the equivalence class to be as small as possible to collide as quickly as possible in the reduced output space.

[1] We emphasize that we used a permutation implementing six steps.

In the preimage attack presented in the sequel, we use an equivalence class of size 256 bits, and rely on the truncation to invert any 256-bit value. This explains why the same strategy cannot be directly applied to `Haraka-256/256`.

Let h be the 256-bit challenge value for the preimage attack. To reduce the search space, we rely on the symmetry class presented before in Section 3.3 and assume that a symmetric state is reached at the output of the `Haraka` permutation. We already note that this happens with probability 2^{-256}, but since the truncation removes exactly 256 bits from the internal state to produce h, we expect that one of the 2^{256} preimages of h should be symmetric.

We denote by $x \in GF(2^{512})$ the preimage of h. Assume that x is defined by the following `AES` states

c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}

and is transformed to the symmetric state

c'_0	c'_0	c'_1	c'_1	c'_2	c'_2	c'_3	c'_3	c'_4	c'_4	c'_5	c'_5	c'_6	c'_6	c'_7	c'_7

after the π_{512} permutation. We observe that the final feed-forward truncating 8 of the 16 columns produces the 256-bit values composed of the following columns:

$c'_1 \oplus c_2$	$c'_1 \oplus c_3$	$c'_3 \oplus c_6$	$c'_3 \oplus c_7$	$c'_4 \oplus c_8$	$c'_4 \oplus c_9$	$c'_6 \oplus c_{12}$	$c'_6 \oplus c_{13}$

Consequently, the XORs of the consecutive pairs of columns only depends on the input state:

$$\Delta_0 \overset{\text{def}}{=} c_2 \oplus c_3, \qquad \Delta_1 \overset{\text{def}}{=} c_6 \oplus c_7,$$
$$\Delta_2 \overset{\text{def}}{=} c_8 \oplus c_9, \qquad \Delta_3 \overset{\text{def}}{=} c_{12} \oplus c_{13}.$$

Therefore, from the value h, one can deduce a 128-bit constraint that the 512-bit message has to verify to reach a symmetric state at the output of π_{512}. More specifically, each of the four `AES` states defining the input message only has 96 bits of entropy.

In terms of freedom degrees, there are 2^{256} possible symmetric states at the output of the permutation, and $2^{256-128} = 2^{128}$ that verifies the restrictions from the Δ_i on the message input. Among these 2^{128} states, one expects only one to verify the 128-bit unchecked values (c'_1, c'_3, c'_4, c'_6) from h. Indeed, 128 bits are already considered through the differences Δ_i.

Preimage Algorithm. If there exists an algorithm enumerating the 2^{128} symmetric states in less than 2^{256} operations, then it directly yields a preimage attack. In the following, we show how to do this in 2^{192} operations.

Since we assume to end the π_{512} permutation with a symmetric state, we know the inner structure of the state three steps before due to the observation made in Section 3.3. Consequently, we only have to analyze two steps of `Haraka`.

To do this, we need to go down one level by inspecting the `AES` states after each `AES` round functions. Let us denote the four AES states at the output of the second step by

$$\begin{bmatrix}\alpha_0 & \alpha_4 & \alpha_8 & \alpha_{12}\\ \alpha_1 & \alpha_5 & \alpha_9 & \alpha_{13}\\ \alpha_2 & \alpha_6 & \alpha_{10} & \alpha_{14}\\ \alpha_3 & \alpha_7 & \alpha_{12} & \alpha_{15}\end{bmatrix} \begin{bmatrix}\alpha_{16} & \alpha_{20} & \alpha_{24} & \alpha_{28}\\ \alpha_{17} & \alpha_{21} & \alpha_{25} & \alpha_{29}\\ \alpha_{18} & \alpha_{22} & \alpha_{26} & \alpha_{30}\\ \alpha_{19} & \alpha_{23} & \alpha_{27} & \alpha_{31}\end{bmatrix} \begin{bmatrix}\alpha_{24} & \alpha_{28} & \alpha_{16} & \alpha_{20}\\ \alpha_{25} & \alpha_{29} & \alpha_{17} & \alpha_{21}\\ \alpha_{26} & \alpha_{30} & \alpha_{18} & \alpha_{22}\\ \alpha_{27} & \alpha_{31} & \alpha_{19} & \alpha_{23}\end{bmatrix} \begin{bmatrix}\alpha_8 & \alpha_{12} & \alpha_0 & \alpha_4\\ \alpha_9 & \alpha_{13} & \alpha_1 & \alpha_5\\ \alpha_{10} & \alpha_{14} & \alpha_2 & \alpha_6\\ \alpha_{12} & \alpha_{15} & \alpha_3 & \alpha_7\end{bmatrix}.$$

Next, we trace the evolution of the columns through $\mathbf{mix}_{512}^{-1}$:

$$\begin{bmatrix}\alpha_{20} & \alpha_{28} & \alpha_8 & \alpha_0\\ \alpha_{21} & \alpha_{29} & \alpha_9 & \alpha_1\\ \alpha_{22} & \alpha_{30} & \alpha_{10} & \alpha_2\\ \alpha_{23} & \alpha_{31} & \alpha_{12} & \alpha_3\end{bmatrix} \begin{bmatrix}\alpha_{28} & \alpha_{20} & \alpha_0 & \alpha_8\\ \alpha_{29} & \alpha_{21} & \alpha_1 & \alpha_9\\ \alpha_{30} & \alpha_{22} & \alpha_2 & \alpha_{10}\\ \alpha_{31} & \alpha_{23} & \alpha_3 & \alpha_{12}\end{bmatrix} \begin{bmatrix}\alpha_{16} & \alpha_{24} & \alpha_{12} & \alpha_4\\ \alpha_{17} & \alpha_{25} & \alpha_{13} & \alpha_5\\ \alpha_{18} & \alpha_{26} & \alpha_{14} & \alpha_6\\ \alpha_{19} & \alpha_{27} & \alpha_{15} & \alpha_7\end{bmatrix} \begin{bmatrix}\alpha_{24} & \alpha_{16} & \alpha_4 & \alpha_{12}\\ \alpha_{25} & \alpha_{17} & \alpha_5 & \alpha_{13}\\ \alpha_{26} & \alpha_{18} & \alpha_6 & \alpha_{14}\\ \alpha_{27} & \alpha_{19} & \alpha_7 & \alpha_{15}\end{bmatrix}.$$

The columns only being reordered, the symmetry is maintained, and the bytes are simply reordered by the subsequent $\mathsf{ShiftRows}^{-1}$. Consequently, at the output of the first layer of AES round functions of the second step, we have the following shape of states:

$$\begin{bmatrix}\beta_{20} & \beta_{28} & \beta_8 & \beta_0\\ \beta_1 & \beta_{21} & \beta_{29} & \beta_9\\ \beta_{10} & \beta_2 & \beta_{22} & \beta_{30}\\ \beta_{31} & \beta_{12} & \beta_3 & \beta_{23}\end{bmatrix} \begin{bmatrix}\beta_{28} & \beta_{20} & \beta_0 & \beta_8\\ \beta_9 & \beta_{29} & \beta_{21} & \beta_1\\ \beta_2 & \beta_{10} & \beta_{30} & \beta_{22}\\ \beta_{23} & \beta_3 & \beta_{12} & \beta_{31}\end{bmatrix} \begin{bmatrix}\beta_{16} & \beta_{24} & \beta_{12} & \beta_4\\ \beta_5 & \beta_{17} & \beta_{25} & \beta_{13}\\ \beta_{14} & \beta_6 & \beta_{18} & \beta_{26}\\ \beta_{27} & \beta_{15} & \beta_7 & \beta_{19}\end{bmatrix} \begin{bmatrix}\beta_{24} & \beta_{16} & \beta_4 & \beta_{12}\\ \beta_{13} & \beta_{25} & \beta_{17} & \beta_5\\ \beta_6 & \beta_{14} & \beta_{26} & \beta_{18}\\ \beta_{19} & \beta_7 & \beta_{15} & \beta_{27}\end{bmatrix}.$$

Observe that the 512-bit state still maintains some symmetry and can be described with the 32 bytes $\beta_0, \dots, \beta_{31}$. The subsequent layer of AES round functions destroys the symmetry since the columns can no longer be paired.

We now consider the three layers of AES round functions that link the message input to this last state. We introduce the following notations (see Figure 7): for $i = 0$, the internal state comprised of the four AES state $(X_i, X'_i, X''_i, X'''_i)$ represents the input to π_{512}. For $i = 1$, this tuple represents the input to the subsequent AES round, for $i = 2$ the input to the next $\mathbf{mix}_{512}$ layer, for $i = 3$ its output, and for $i = 4$ the output of the following AES round.

The algorithm starts by enumerating (X_4, X'_4) in 2^{128} operations. For each (X_4, X'_4), we compute (X_3, X'_3) and then deduce the first and last columns of X_2, X'_2, X''_2 and X'''_2 and therefore two diagonals (8 bytes) of each X_1, X'_1, X''_1 and X'''_1.

Then, we observe that only 2^{32} states X_0 can match the partial information deduced in X_1. Indeed, there are only 2^{96} inputs with the correct Δ_0 difference between the two last columns, and the additional known bytes after the MixColumns operation introduce a 64-bit constraint. To generate all the matching states X_0, we first enumerate all 2^{16} values for (y_8, y_9) (see Figure 8), then we can deduce x_8, x_{13}, x_2 and x_7 in X_0. Consequently, we linearly deduce x_{12} and x_9 from Δ_0. Then, by also enumerating all 2^{16} values of (y_4, y_{13}), we deduce all remaining values, in particular the full state X_0.

By repeating similar algorithms for all four branches, we therefore construct four lists L, L', L'' and L''' that each contains 2^{32} values of X_0, X'_0, X''_0 and X'''_0 respectively.

Next, we enumerate all 2^{64} pairs of states $(X_0, X'_0) \in L \times L'$, and deduce all intermediate values in the permutation. In particular, we get partial values of both X''_4 and X'''_4. We store the partial (X''_4, X'''_4) in a table T. We do the same for all 2^{64} pairs of states $(X''_0, X'''_0) \in L'' \times L'''$ deduce the other half (X''_4, X'''_4) and look for collisions in T. Since the two states are defined with only 128 bits due to the symmetry, we expect one solution (X''_4, X'''_4) on average that verifies the symmetry.

All in all, after trying 2^{128} values for (X_4, X'_4), we get a symmetric pair for (X''_4, X'''_4) in about 2^{64} computations. This algorithm therefore generates 2^{128} candidates for the output of the permutation, where one is expected to verify the remaining 128-bit constraint on the target value h.

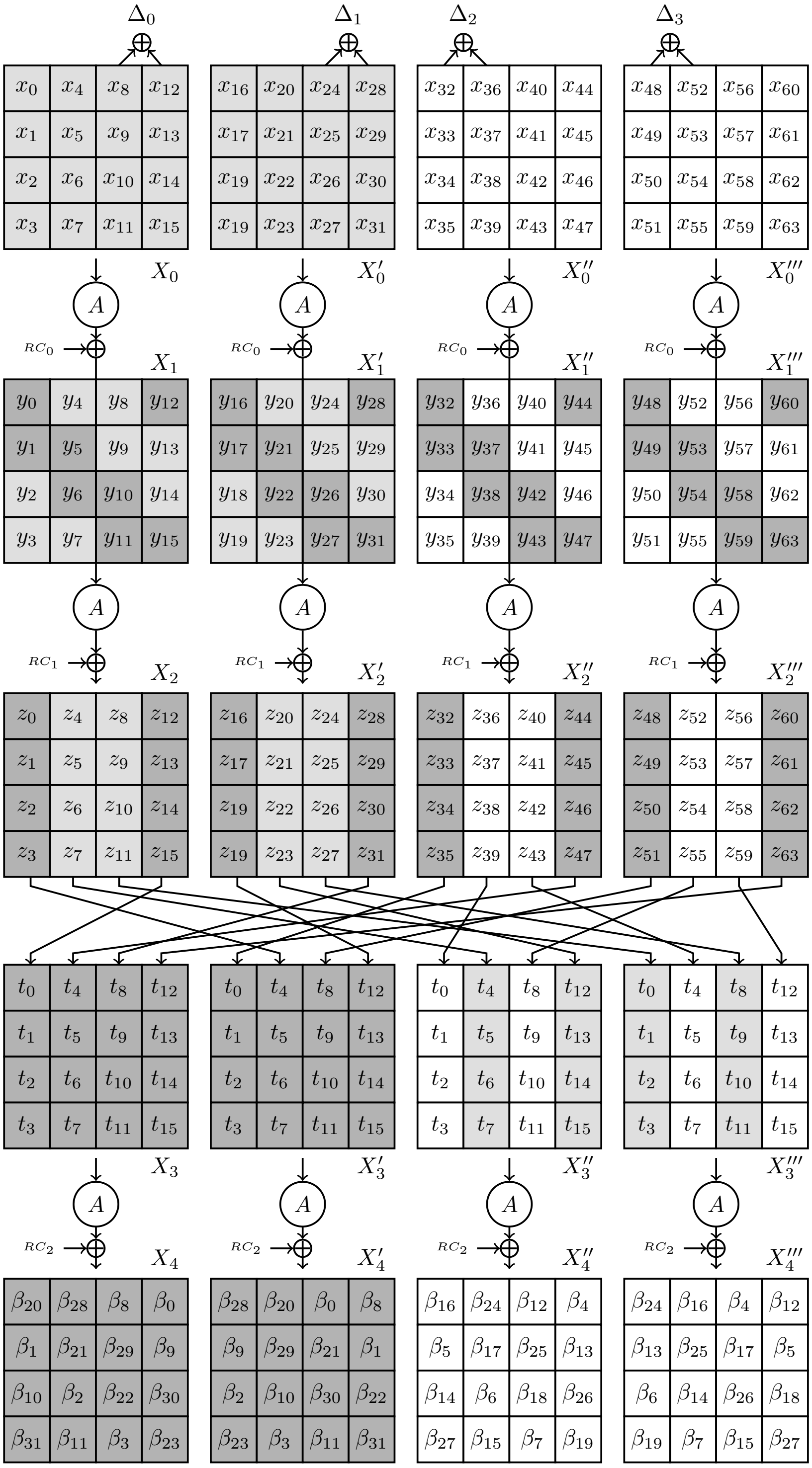
Δ_0
Δ_1
Δ_2
Δ_3
x_0 x_4 x_8 x_{12}
x_1 x_5 x_9 x_{13}
x_2 x_6 x_{10} x_{14}
x_3 x_7 x_{11} x_{15}
x_{16} x_{20} x_{24} x_{28}
x_{17} x_{21} x_{25} x_{29}
x_{19} x_{22} x_{26} x_{30}
x_{19} x_{23} x_{27} x_{31}
x_{32} x_{36} x_{40} x_{44}
x_{33} x_{37} x_{41} x_{45}
x_{34} x_{38} x_{42} x_{46}
x_{35} x_{39} x_{43} x_{47}
x_{48} x_{52} x_{56} x_{60}
x_{49} x_{53} x_{57} x_{61}
x_{50} x_{54} x_{58} x_{62}
x_{51} x_{55} x_{59} x_{63}
X_0
X_0'
X_0''
X_0'''
A
RC_0
X_1
X_1'
X_1''
X_1'''
y_0 y_4 y_8 y_{12}
y_1 y_5 y_9 y_{13}
y_2 y_6 y_{10} y_{14}
y_3 y_7 y_{11} y_{15}
y_{16} y_{20} y_{24} y_{28}
y_{17} y_{21} y_{25} y_{29}
y_{18} y_{22} y_{26} y_{30}
y_{19} y_{23} y_{27} y_{31}
y_{32} y_{36} y_{40} y_{44}
y_{33} y_{37} y_{41} y_{45}
y_{34} y_{38} y_{42} y_{46}
y_{35} y_{39} y_{43} y_{47}
y_{48} y_{52} y_{56} y_{60}
y_{49} y_{53} y_{57} y_{61}
y_{50} y_{54} y_{58} y_{62}
y_{51} y_{55} y_{59} y_{63}
A
RC_1
X_2
X_2'
X_2''
X_2'''
z_0 z_4 z_8 z_{12}
z_1 z_5 z_9 z_{13}
z_2 z_6 z_{10} z_{14}
z_3 z_7 z_{11} z_{15}
z_{16} z_{20} z_{24} z_{28}
z_{17} z_{21} z_{25} z_{29}
z_{19} z_{22} z_{26} z_{30}
z_{19} z_{23} z_{27} z_{31}
z_{32} z_{36} z_{40} z_{44}
z_{33} z_{37} z_{41} z_{45}
z_{34} z_{38} z_{42} z_{46}
z_{35} z_{39} z_{43} z_{47}
z_{48} z_{52} z_{56} z_{60}
z_{49} z_{53} z_{57} z_{61}
z_{50} z_{54} z_{58} z_{62}
z_{51} z_{55} z_{59} z_{63}
t_0 t_4 t_8 t_{12}
t_1 t_5 t_9 t_{13}
t_2 t_6 t_{10} t_{14}
t_3 t_7 t_{11} t_{15}
X_3
X_3'
X_3''
X_3'''
A
RC_2
X_4
X_4'
X_4''
X_4'''
β_{20} β_{28} β_8 β_0
β_1 β_{21} β_{29} β_9
β_{10} β_2 β_{22} β_{30}
β_{31} β_{11} β_3 β_{23}
β_{28} β_{20} β_0 β_8
β_9 β_{29} β_{21} β_1
β_2 β_{10} β_{30} β_{22}
β_{23} β_3 β_{11} β_{31}
β_{16} β_{24} β_{12} β_4
β_5 β_{17} β_{25} β_{13}
β_{14} β_6 β_{18} β_{26}
β_{27} β_{15} β_7 β_{19}
β_{24} β_{16} β_4 β_{12}
β_{13} β_{25} β_{17} β_5
β_6 β_{14} β_{26} β_{18}
β_{19} β_7 β_{15} β_{27}

Figure 7: General overview of the final phase of the preimage attack. The darkgray bytes are guessed and the lightgray bytes are deduced.

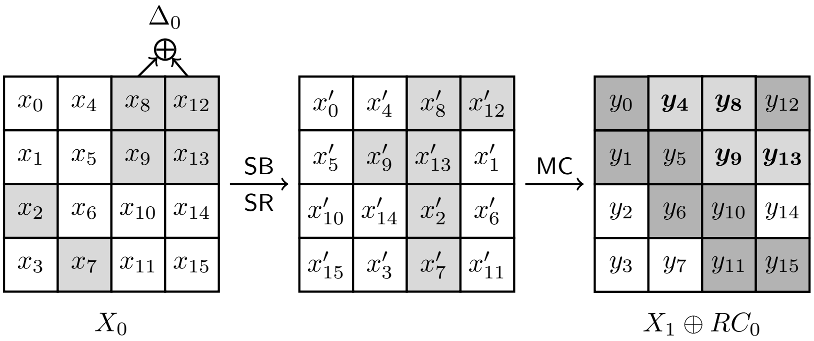
Δ_0
x_0 x_4 x_8 x_{12}
x_1 x_5 x_9 x_{13}
x_2 x_6 x_{10} x_{14}
x_3 x_7 x_{11} x_{15}
X_0
SB
SR
x'_0 x'_4 x'_8 x'_{12}
x'_5 x'_9 x'_{13} x'_1
x'_{10} x'_{14} x'_2 x'_6
x'_{15} x'_3 x'_7 x'_{11}
MC
y_0 $\boldsymbol{y_4}$ $\boldsymbol{y_8}$ y_{12}
y_1 y_5 $\boldsymbol{y_9}$ $\boldsymbol{y_{13}}$
y_2 y_6 y_{10} y_{14}
y_3 y_7 y_{11} y_{15}
$X_1 \oplus RC_0$

Figure 8: First `AES` round in the preimage attack: the darkgray bytes are known and we guess (y_4, y_8, y_9, y_{13}) to enumerate all the 2^{32} states X_0 that match all the known bytes. The lightgray bytes are deduced, and allow to deduce the values in all remaining bytes.

Complexity. The total time complexity of this algorithm can be expressed as follows:

$$2^{128} \cdot \left(4 \cdot 2^{32} + 2 \cdot 2^{64}\right) \approx 2^{192}$$

and the memory complexity essentially stores the final meet-in-the-middle table T of 2^{64} values.

6 Conclusion

In this paper, we have provided attacks against `Haraka` hash functions, which breaks the preimage resistance of the large function `Haraka-512/256`, and the collision resistance of both `Haraka-256/256` and `Haraka-512/256`. Our preimage attack finds a valid input in about 2^{192} computations, while our collision attack finds colliding messages from a small subspace in about 2^{16} function evaluations. The attacks rely on symmetric states that are preserved for several steps of the inner permutation which are not destroyed due to highly structured round constants. We note that these attacks can easily be thwarted by using different round constants without affecting the software efficiency of the designs.

A 5-Collisions in Haraka

Table 2: Examples of 5-collisions for Haraka. The five strongly symmetric messages M_i are constructed by the concatenation m_i^n of n equal 32-bit blocks m_i, and they all produce the same strongly symmetric value $h(M_i)$ described in the last column. We have $n = 8$ for $h =$ Haraka-256/256 and $n = 16$ for $h =$ Haraka-512/256.

m_1	m_2	m_3	m_4	m_5	$h(m_i^n)$
0b1bdced	0a0ef844	0964701e	0387c94b	016549d0	04db0fe0
08b9de4f	06f38a45	052f752a	04c3e89d	026232c4	139bf2ce
0eeed09b	0c086017	0779efa9	04b5ddef	02922ab3	1652a79c
0da38dc7	0b9204e6	0b6fd895	065ba81d	02b95866	1b6fdcb2
0bf82512	0956b85e	02f62f91	00bb53d3	0062841c	24569de0
0eea626b	0bc42c33	0697f466	054846e5	04568d19	2667df18
0bdd2fdc	0a8738e0	053d935f	03f60e70	01ab7946	27dbb0b7
08a74a16	03e50376	029dc641	01410df7	003a9ff6	3a2d117d
0e8e613b	0d1ffc8d	0bbee25a	09e2d933	0204e5a5	4124e4fa
0ada449b	0a22394c	09c30d48	064d2b2d	0515afc9	41a94ca8
0d3d43b7	0a9b9633	093c2e0d	07a00cc3	05389fc4	478ba06e
0ff0ed1e	0f9d64c6	0e5c1620	092f106a	06565ea7	50d7dec3
0c52acee	0c20e041	0a89e1f8	07278c95	00dfea66	518e9672
0eaf780d	0a5268e0	02e59fb4	01d2c9d0	01806b17	59b65309
0ef0fa28	0cf30920	0b9f5486	055dae46	03bb1cf8	72afced6
0d015142	0c95c29e	09a827fd	05962873	03dece6a	739e2600
0f32a906	0d5ed761	0bae83c2	0aef5f7f	032e9da3	7d9a3a76
0cd26dcd	06e53d53	0596b253	0516918e	00d8d609	833d2572
0fb50b4b	0dd1849d	0a015abb	07769307	020b1e46	8bf474de
0e479f28	0bd4cf82	067c41b5	010d5eee	00d20d14	8f85488f
0faa8dfe	0cba2273	08059f27	07897bf5	064adf44	947c915d
0e58e47d	0bf25d39	09213d54	05ee5bf0	028d5d64	96ae6849
0f5579e2	0f4459b9	0f1e4a6c	0d11d8ac	04aec011	9979b3b0
0e465eb8	0844dcfd	07b59704	03f9ce67	027ea25e	ad2448ab
0bb8afd5	0ab837f3	05a9c72e	054721b3	041233a6	d306dc1a
0ea32558	0b0a3a0f	0ab943c2	05077deb	00772f11	ebec4664
0af64a8e	092df02b	0706d013	061cd4d3	04190c3b	efa32c4f
0524b410	05219e31	048509f3	025ff2e1	008d4a14	f0971607
0f91659a	0b098ab1	05b13941	04060205	034aaecc	f3bc0541
0efcb18e	0bf347de	041fd72d	03e9787b	037682d8	f8fea34e
0f93bbda	0db0455e	0a6a0dec	09a8d1da	090467a1	f9c8fbd7

References

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. LNCS, pages 411–436. Springer, December 2015.

[Ber] Daniel Bernstein. CAESAR Competition. http://competitions.cr.yp.to/caesar.html.

[DGPW12] Alexandre Duc, Jian Guo, Thomas Peyrin, and Lei Wei. Unaligned rebound attack: Application to Keccak. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 402–421. Springer, March 2012.

[GJN+15] Jian Guo, Jérémy Jean, Ivica Nikolić, Kexin Qiao, Yu Sasaki, and Siang Meng Sim. Invariant Subspace Attack Against Full Midori64. Cryptology ePrint Archive, Report 2015/1189, 2015.

[JNP12] Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Improved rebound attack on the finalist Grøstl. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 110–126. Springer, March 2012.

[JNSW14] Jérémy Jean, Ivica Nikolic, Yu Sasaki, and Lei Wang. Practical Cryptanalysis of PAES. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 2014.

[JNSW16] Jérémy Jean, Ivica Nikolic, Yu Sasaki, and Lei Wang. Practical Forgeries and Distinguishers against PAES. *IEICE Transactions*, 99-A(1):39–48, 2016.

[KNR10] Dmitry Khovratovich, Ivica Nikolic, and Christian Rechberger. Rotational rebound attacks on reduced Skein. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 1–19. Springer, December 2010.

[LSWD04] Tri Van Le, Rüdiger Sparr, Ralph Wernsdorf, and Yvo Desmedt. Complementation-Like and Cyclic Properties of AES Round Functions. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *AES Conference*, volume 3373 of *Lecture Notes in Computer Science*, pages 128–141. Springer, 2004.

[MRST09] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In Orr Dunkelman, editor, *FSE 2009*, volume 5665 of *LNCS*, pages 260–276. Springer, February 2009.

[NTV11] María Naya-Plasencia, Deniz Toz, and Kerem Varici. Rebound attack on JH42. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 252–269. Springer, December 2011.

[Ste] Stefan Kölbl and Martin M. Lauridsen and Florian Mendel and Christian Rechberger. Haraka - Public Implementations. https://github.com/kste/haraka.

[Ste16] Stefan Kölbl and Martin M. Lauridsen and Florian Mendel and Christian Rechberger. Haraka - Efficient Short-Input Hashing for Post-Quantum Applications. Cryptology ePrint Archive, Report 2016/098, 2016.

[STKT06] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. *Birthday Paradox for Multi-collisions*, pages 29–40. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.