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 $\pi$ 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 $\pi$.

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 $\{m_1, \ldots, m_5\}$ 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
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:

\[
\begin{align*}
\text{Haraka-256/256} & : GF(2^{256}) \rightarrow GF(2^{256}), \\
\text{Haraka-512/256} & : GF(2^{512}) \rightarrow GF(2^{256}).
\end{align*}
\]

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

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

\[ m \xrightarrow{\pi} h \]

**Figure 1:** Haraka uses the Davies-Meyer construction with a permutation \(\pi\).

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).

\[
\begin{array}{ccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
\end{array}
\]

**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 \(\text{mix}_{256}\) in \(\pi_{256}\) and \(\text{mix}_{512}\) in \(\pi_{512}\). For completeness, we give:

\[
\begin{align*}
\text{mix}_{256}(c_0, \ldots, c_7) &= (c_0, c_4, c_1, c_5, c_2, c_6, c_3, c_7) \\
\text{mix}_{512}(c_0, \ldots, c_{15}) &= (c_3, c_{11}, c_7, c_{15}, c_8, c_9, c_{12}, c_4, c_9, c_1, c_{13}, c_5, c_2, c_{10}, c_6, c_{14}).
\end{align*}
\]

**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.
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 restricted case, when all four columns of a state are equal (see Figure 5). We call such a state strongly symmetric.

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 $\pi_{256}$ and $\pi_{512}$

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

Figure 4: Haraka-512/256 step function.
RC₀ consist of strongly symmetric states as previously defined. For instance, the two first subkeys equal:

\[
RC₀ = \begin{pmatrix}
1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}, \quad RC₁ = \begin{pmatrix}
2 & 2 & 2 & 2 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}.
\]

Subsequent constants RCₖ are deduced from RCₖ₋₁ by applying the same function to each column of the state, hence preserving the strong symmetry.

Since in the Haraka permutation π₂₅₆ (resp. π₅₁₂), the columns of state are reordered by the mix₂₅₆ (resp. mix₅₁₂) 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 π₂₅₆ and π₅₁₂. We note that there are a total of 2^32 different strongly symmetric states.

### 3.3 Symmetry for π₅₁₂

We now describe another property for the large permutation π₅₁₂. 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 π₅₁₂. Both classes presented has been found by hand by careful analysis of the step functions.

Consider the state described by eight independent columns c₀,...,c₇ represented in the first line of Figure 6. This figure also shows how this state is transformed after three steps of π₅₁₂. We note that there are 2^256 internal states of this structure.

![Figure 6](image-url) 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 π₅₁₂.

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

\[
S₁ = \begin{bmatrix}
b₀ & b₄ & b₈ & b₁₂ \\
b₁ & b₅ & b₉ & b₁₃ \\
b₂ & b₆ & b₁₀ & b₁₄ \\
b₃ & b₇ & b₁₂ & b₁₅
\end{bmatrix} \quad \text{and} \quad S₂ = \begin{bmatrix}
b₈ & b₁₂ & b₀ & b₄ \\
b₉ & b₁₃ & b₁ & b₅ \\
b₁₀ & b₁₄ & b₂ & b₆ \\
b₁₁ & b₁₅ & b₃ & b₇
\end{bmatrix},
\]

then the states after the application of \(f = \text{ShiftRows} \circ \text{SubBytes}\) can be written as:

\[
f(S₁) = \begin{bmatrix}
b₀' & b₄' & b₈' & b₁₂' \\
b₁' & b₅' & b₉' & b₁₃' \\
b₂' & b₆' & b₁₀' & b₁₄' \\
b₃' & b₇' & b₁₂' & b₁₅'
\end{bmatrix} \quad \text{and} \quad f(S₂) = \begin{bmatrix}
b₈' & b₁₂' & b₀' & b₄' \\
b₉' & b₁₃' & b₁' & b₅' \\
b₁₀' & b₁₄' & b₂' & b₆' \\
b₁₁' & b₁₅' & b₃' & b₇'
\end{bmatrix}.
\]
where the two halves are still equal, which is maintained by the subsequent MixColumns \((b'_i = S(b_i))\) 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 \(\text{mix}_{512}\) transformation finally reorders the columns and creates two new pairs of states with swapped halves.

After three steps (see Figure 6), the last the \(\text{mix}_{512}\) 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 \(\pi_{256}\) and \(\pi_{512}\) 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].

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 \(\left(\frac{t!}{t}\right) \cdot 2^{n(t-1)/t}\) 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 \(h = \text{Haraka-256/256}\) and \(n = 16\) for \(h = \text{Haraka-512/256}\).

<table>
<thead>
<tr>
<th>(m)</th>
<th>(h(m))</th>
</tr>
</thead>
<tbody>
<tr>
<td>000325eb...000325eb</td>
<td>9ca2e0a5...9ca2e0a5</td>
</tr>
<tr>
<td>00016734...00016734</td>
<td>9ca2e0a5...9ca2e0a5</td>
</tr>
<tr>
<td>0b1bdced...0b1bdced</td>
<td>04db0fe0...04db0fe0</td>
</tr>
<tr>
<td>0a0ef844...0a0ef844</td>
<td>04db0fe0...04db0fe0</td>
</tr>
<tr>
<td>0964701e...0964701e</td>
<td>04db0fe0...04db0fe0</td>
</tr>
<tr>
<td>0387c94b...0387c94b</td>
<td>04db0fe0...04db0fe0</td>
</tr>
<tr>
<td>016549d0...016549d0</td>
<td>04db0fe0...04db0fe0</td>
</tr>
</tbody>
</table>

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

\[
\begin{array}{ccccccccccccc}
c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15}
\end{array}
\]

and is transformed to the symmetric state

\[
\begin{array}{ccccccccccccc}
c'_0 & c'_1 & c'_2 & c'_3 & c'_4 & c'_5 & c'_6 & c'_7 & c'_8 & c'_9 & c'_{10} & c'_{11} & c'_{12} & c'_{13} & c'_{14} & c'_{15}
\end{array}
\]

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

\[
\begin{array}{cccccccc}
c'_1 \oplus c_2 & c'_1 \oplus c_3 & c'_1 \oplus c_6 & c'_3 \oplus c_7 & c'_4 \oplus c_8 & c'_4 \oplus c_9 & c'_6 \oplus c_{12} & c'_6 \oplus c_{13}
\end{array}
\]

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

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

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 $\pi_{512}$. 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 $\Delta_i$ 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 $\Delta_i$.

**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 $\pi_{512}$ 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 \(\text{mix}^{-1}_{512} \). The columns only being reordered, the symmetry is maintained, and the bytes are simply reordered by the subsequent \(\text{ShiftRows}^{-1} \). Consequently, at the output of the first layer of AES round functions of the second step, we have the following shape of states:
\[
\begin{bmatrix}
\beta_{20} & \beta_{28} & \beta_8 & \beta_0 \\
\beta_1 & \beta_{21} & \beta_{29} & \beta_9 \\
\beta_{10} & \beta_{22} & \beta_{30} & \beta_{23} \\
\beta_{31} & \beta_{12} & \beta_3 & \beta_{23}
\end{bmatrix}
\begin{bmatrix}
\beta_{18} & \beta_{20} & \beta_0 & \beta_8 \\
\beta_9 & \beta_{29} & \beta_{21} & \beta_1 \\
\beta_{2} & \beta_{10} & \beta_{30} & \beta_{22} \\
\beta_{33} & \beta_{3} & \beta_{12} & \beta_{31}
\end{bmatrix}
\begin{bmatrix}
\beta_{16} & \beta_{24} & \beta_{12} & \beta_4 \\
\beta_5 & \beta_{17} & \beta_{25} & \beta_{13} \\
\beta_{14} & \beta_{16} & \beta_{26} & \beta_{18} \\
\beta_{27} & \beta_{15} & \beta_{19} & \beta_{27}
\end{bmatrix}
\begin{bmatrix}
\beta_{24} & \beta_{16} & \beta_4 & \beta_{12} \\
\beta_{25} & \beta_{17} & \beta_{13} & \beta_{25} \\
\beta_{26} & \beta_{28} & \beta_6 & \beta_{26} \\
\beta_{19} & \beta_{17} & \beta_{15} & \beta_{27}
\end{bmatrix}.
\]

Observe that the 512-bit state still maintains some symmetry and can be described with the 32 bytes \(\beta_0, \ldots, \beta_{31} \). 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 \((X_1, X_2, X_3, X_4)\) represents the input to \(\pi_{512} \). For \(i = 1 \), this tuple represents the input to the subsequent AES round, for \(i = 2 \) the input to the next \(\text{mix}_{512} \) layer, for \(i = 3 \) its output, and for \(i = 4 \) the output of the following AES round.

The algorithm starts by enumerating \((X_4, X_3)\) in \(2^{128} \) operations. For each \((X_4, X_3)\), we compute \((X_3, X_2)\) and then deduce the first and last columns of \(X_2, X_1, X_0^r\) and \(X_0^l\) and therefore two diagonals (8 bytes) of each \(X_1, X_0^r, X_0^l\) and \(X_0^r\).

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 \(\Delta_0 \) difference between the two last columns, and the additional known bytes after the \(\text{MixColumns} \) 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 \(x_8, x_{13}, x_2, x_7 \) in \(X_0 \). Consequently, we linearly deduce \(x_{12} \) and \(x_9 \) from \(\Delta_0 \). 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 \(X_0, X_0^r, X_0^l \) and \(X_0'' \) respectively.

Next, we enumerate all \(2^{64} \) pairs of states \((X_0, X_0') \) and deduce all intermediate values in the permutation. In particular, we get partial values of both \(X_0'' \) and \(X_0''' \). We store the partial \((X_4'', X_3'')\) in a table \(T \). We do the same for all \(2^{64} \) pairs of states \((X_0'', X_0''') \) and look for collisions in \(T \). Since the two states are defined with only 128 bits due to the symmetry, we expect one solution \((X_4'', X_3'') \) 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 \(h \).
**Figure 7**: General overview of the final phase of the preimage attack. The dark gray bytes are guessed and the light gray bytes are deduced.
Complexity. The total time complexity of this algorithm can be expressed as follows:

\[ 2^{128} \cdot (4 \cdot 2^{32} + 2 \cdot 2^{64}) \approx 2^{192} \]

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.
## 5-Collisions in Haraka

Table 2: Examples of 5-collisions for *Haraka*. The five strongly symmetric messages $M_i$ are constructed by the concatenation $m^n_i$ 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 $h = \text{Haraka-256/256}$ and $n = 16$ for $h = \text{Haraka-512/256}$.

<table>
<thead>
<tr>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
<th>$m_4$</th>
<th>$m_5$</th>
<th>$h(m^n_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b1bdced</td>
<td>0a0ef844</td>
<td>0964701e</td>
<td>0387c94b</td>
<td>016594d0</td>
<td>04dbf0e0</td>
</tr>
<tr>
<td>08b9de4f</td>
<td>06f38a45</td>
<td>052f752a</td>
<td>04c3e89d</td>
<td>026232c4</td>
<td>139bf2ce</td>
</tr>
<tr>
<td>0eed09b</td>
<td>0c086017</td>
<td>0779efe9</td>
<td>04b5ddf</td>
<td>02922ab3</td>
<td>1652a79c</td>
</tr>
<tr>
<td>0da38dc7</td>
<td>0b9204e6</td>
<td>0b6f9895</td>
<td>065ba81d</td>
<td>02b95866</td>
<td>1b6fdec2</td>
</tr>
<tr>
<td>0bf82512</td>
<td>095685e</td>
<td>02f2f2f9</td>
<td>00b53d3</td>
<td>062841c</td>
<td>2456de0</td>
</tr>
<tr>
<td>0eea626b</td>
<td>0bc42c33</td>
<td>0697f466</td>
<td>05486e5</td>
<td>0456d19</td>
<td>2667f18</td>
</tr>
<tr>
<td>0bdf2fcd</td>
<td>08738e0</td>
<td>053935f</td>
<td>03f6e70</td>
<td>01ab7946</td>
<td>27dbbb7</td>
</tr>
<tr>
<td>08a74a16</td>
<td>03e50376</td>
<td>029dc641</td>
<td>01410df7</td>
<td>0039ffe6</td>
<td>3a2d117d</td>
</tr>
<tr>
<td>0ee8613b</td>
<td>0d1f8c8d</td>
<td>0bcee25a</td>
<td>09ed933</td>
<td>0204e5a5</td>
<td>4124efa</td>
</tr>
<tr>
<td>0ada449b</td>
<td>0a22394c</td>
<td>09c30d48</td>
<td>0642b2d</td>
<td>0515acf9</td>
<td>41a9ca8</td>
</tr>
<tr>
<td>0dd43b7</td>
<td>0a9b9633</td>
<td>093c2e0d</td>
<td>07a00cc</td>
<td>05389fc4</td>
<td>478a06e</td>
</tr>
<tr>
<td>0ff0ed1e</td>
<td>0f9d64c6</td>
<td>0e5c1620</td>
<td>092f106a</td>
<td>0656ea7</td>
<td>507dec3</td>
</tr>
<tr>
<td>0c52acee</td>
<td>0c20e041</td>
<td>0a89e1f8</td>
<td>07278c9</td>
<td>00defa6</td>
<td>518e9672</td>
</tr>
<tr>
<td>0eaf780d</td>
<td>0a52b9e0</td>
<td>02a59fb4</td>
<td>012d90</td>
<td>01806b17</td>
<td>59b65309</td>
</tr>
<tr>
<td>0ef0fa28</td>
<td>0cf30920</td>
<td>0b9f5486</td>
<td>055dae6</td>
<td>03b1c8</td>
<td>72afced6</td>
</tr>
<tr>
<td>0d015142</td>
<td>0c95c29e</td>
<td>09a82f7d</td>
<td>05968273</td>
<td>03dece6a</td>
<td>739e2600</td>
</tr>
<tr>
<td>0f32a906</td>
<td>0d567671</td>
<td>0babe3c2</td>
<td>0aef5f7f</td>
<td>0329eda3</td>
<td>7d9a376</td>
</tr>
<tr>
<td>0cd25cd</td>
<td>0653d53</td>
<td>0596b253</td>
<td>0516918e</td>
<td>0048d609</td>
<td>83342572</td>
</tr>
<tr>
<td>0fb5b6bb</td>
<td>0d1849d</td>
<td>0a015aab</td>
<td>07769307</td>
<td>00be146</td>
<td>8bfe47de</td>
</tr>
<tr>
<td>0e479f28</td>
<td>0bd4cf82</td>
<td>067c41b5</td>
<td>010d5ee</td>
<td>0020d14</td>
<td>8f858488</td>
</tr>
<tr>
<td>0faa8dfe</td>
<td>0c0a2273</td>
<td>08059f27</td>
<td>07879bf5</td>
<td>064af6</td>
<td>9479c15</td>
</tr>
<tr>
<td>0e5847d</td>
<td>0b2f3d9</td>
<td>09213d54</td>
<td>05e5b0f</td>
<td>028d6d4</td>
<td>96ae6849</td>
</tr>
<tr>
<td>0f5579e2</td>
<td>0f4459b9</td>
<td>01e4a6c</td>
<td>0d11d8ac</td>
<td>04a0c11</td>
<td>9979b3b0</td>
</tr>
<tr>
<td>0e465eb8</td>
<td>0844cfd</td>
<td>07b59704</td>
<td>03f9ce67</td>
<td>027ea25e</td>
<td>ad244ab</td>
</tr>
<tr>
<td>0bb8afdf</td>
<td>0a83f73</td>
<td>05a9c72e</td>
<td>054721b3</td>
<td>041233e</td>
<td>d306d1ca</td>
</tr>
<tr>
<td>0ea32558</td>
<td>0b0a3a0f</td>
<td>0ab943c2</td>
<td>05077dee</td>
<td>0077f11</td>
<td>ebec4664</td>
</tr>
<tr>
<td>0af64a8e</td>
<td>09df02b</td>
<td>0706d013</td>
<td>061cd4d4</td>
<td>0419c3b</td>
<td>efa324cf</td>
</tr>
<tr>
<td>0524b410</td>
<td>05219e31</td>
<td>048509f3</td>
<td>025ff2e1</td>
<td>08da14</td>
<td>f971607</td>
</tr>
<tr>
<td>09f1659a</td>
<td>0b09ab1</td>
<td>05b13941</td>
<td>04060205</td>
<td>034aaec</td>
<td>f3b0541</td>
</tr>
<tr>
<td>0efb18e</td>
<td>0bf347de</td>
<td>041f72d</td>
<td>03e978b7</td>
<td>0376b2d8</td>
<td>f8f3ae4e</td>
</tr>
<tr>
<td>0f93bbda</td>
<td>0db455e</td>
<td>0a6a0dec</td>
<td>09a81da</td>
<td>090467a1</td>
<td>f9c8fbd7</td>
</tr>
</tbody>
</table>
References


