Cryptanalysis of ESSENCE

María Naya-Plasencia[1,*], Andrea Röck[2], Jean-Philippe Aumasson[3,†], Yann Laigle-Chapuy[1], Gaëtan Leurent[4], Willi Meier[3,‡], and Thomas Peyrin[5]

[1] INRIA project-team SECRET, France
[2] Helsinki University of Technology, Finland
[3] FHNW, Windisch, Switzerland
[4] École Normale Supérieure, Paris, France
[5] Ingenico, France

Abstract. ESSENCE is a hash function submitted to the NIST Hash Competition that stands out as a hardware-friendly and highly parallelizable design, and that has thus far remained unbroken. Preliminary analysis in its documentation argues that it resists standard differential cryptanalysis. This paper disproves this claim, showing that advanced techniques can be used to significantly reduce the cost of such attacks: using a manually found differential path and a nontrivial search algorithm, we obtain shortcut collision attacks on the full ESSENCE-256 and ESSENCE-512, with respective complexities $2^{67.4}$ and $2^{134.7}$. As an aside, we show how to use these attacks for forging valid message/MAC pairs for HMAC-ESSENCE-256 and HMAC-ESSENCE-512, essentially at the same cost as a collision.

1 Introduction

Recent years have seen a surge of research on cryptographic hashing, since devastating attacks [12, 11, 2, 10] on the two most deployed hash functions, MD5 and SHA-1. The consequent lack of confidence in the current NIST standard SHA-2 [7], stemming from its similarity with those algorithms, motivated NIST to launch the *NIST Hash Competition*, a public competition to develop a new hash standard, which will be called SHA-3 and announced by 2012. NIST received 64 submissions, and accepted 51 as first round candidates. As of June 2009, more than 20 of those were shown to have significant weaknesses[6]. That competition catches the attention not only from many academics, but also from industry—with candidates from IBM, Hitachi, Intel, Sony—and from governmental organizations.

ESSENCE [3, 4] is a first round candidate in the NIST Hash Competition that like many others has two main instances, operating on 32- and 64-bit words, respectively: ESSENCE-256 and ESSENCE-512. These functions process messages

[*] Supported in part by the French Agence Nationale de la Recherche under contract ANR-06-SETI-013-RAPIDE.

[†] Supported by the Swiss National Science Foundation under project no. 113329.

[‡] Supported by GEBERT RÜF STIFTUNG, project no. GRS-069/07.

[6] See ECRYPT's SHA-3 Zoo: `http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo`.

using a binary tree structure, and use a simple compression algorithm based on two non-linear feedback shift registers (NFSR's). ESSENCE shares similarities with MD6 [9], for example the tree-hashing and the hardware-friendly compression function.

This paper presents shortcut collision attacks on ESSENCE-256 and ESSENCE-512. At the heart of our attacks is a single differential path, and our main technical achievement is a nontrivial method for searching inputs conforming to this path at a reduced cost. As an aside, we describe how to use these attacks for forging valid message/MAC pairs for HMAC-ESSENCE-256 and HMAC-ESSENCE-512 in far fewer than $2^{n/2}$ trials. These findings show that ESSENCE does not satisfy the security requirements set by NIST for the future SHA-3.

In a parallel work, Mouha et al. [6] present results on reduced versions of ESSENCE, including a pseudo-collision attack on ESSENCE-512 reduced to 31 steps. They exploit a differential path of a different type than ours, and also use different techniques to search for conforming inputs.

The rest of the paper is organized as follows: §2 briefly introduces ESSENCE; §3 describes our method for searching collisions and its complexity analysis; §4 shows how to attack the HMAC construction when instantiated with ESSENCE, and finally §5 concludes.

2 Brief description of ESSENCE

We give a brief description of the ESSENCE hash functions, which should be sufficient to understand our attacks. A complete specification can be found in [3, 4]. Henceforth statements of (non-) linearity are relative to the field $\mathrm{GF}(2) = \{0, 1\}$ or to an extension thereof.

2.1 Structure

ESSENCE processes a message by constructing a balanced binary tree of bounded depth whose leaves correspond to calls to a compression function with message chunks as input. More precisely, each leaf corresponds to a series of three sequential compressions, with a unique initial value for each leaf that depends on several parameters of the hash function. Likewise, nodes correspond to series of three compressions, with children chaining values as input.

After creation of all message blocks, one appends a *final block* to the data to be hashed. This block contains parameters of the function as well as message-dependent information, and it potentially assists prevent near-collision attacks.

2.2 Compression Function

The compression function of ESSENCE takes as input an eight-word chaining value and an eight-word message block. Words are 32-bit for ESSENCE-256 and 64-bit for ESSENCE-512, so those values are respectively 256- and 512-bit.

Versions of ESSENCE with 224- and 384-bit digests are derived from the main instances by tweaking parameters and truncation of the final digest.

The compression function uses two NFSR's, each operating on a register of eight words:

- $r = (r0, \ldots, r7)$ is initialized with the chaining value, and
- $k = (k0, \ldots, k7)$ is initialized with the message block.

At each step of the compression algorithm, the mechanism in Fig. 1 is clocked using a non-linear function F (see Fig. 2), and a linear function L that provides diffusion across word slices. This mechanism defines a permutation and the compression function returns as new chaining value the XOR of the r register with its initial value, as in the Davies-Meyer scheme.

The documentation of ESSENCE recommends at least 24 steps, and set 32 steps in the actual submission for extra precaution [4, §4].

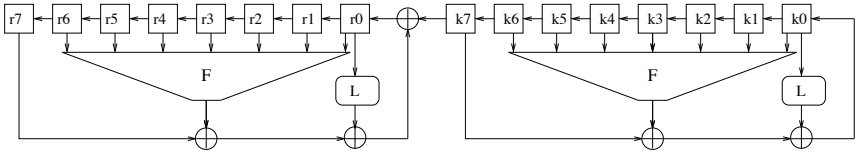

Fig. 1. Overview of the compression function of ESSENCE.

$$
\begin{aligned}
F(a,b,c,d,e,f,g) = \; & abcdefg + abcdef + abcefg + acdefg + abceg + \\
& abdef + abdeg + abefg + acdef + acdfg + acefg + \\
& adefg + bcdfg + bdefg + cdefg + abcf + abcg + \\
& abdg + acdf + adef + adeg + adfg + bcde + \\
& bceg + bdeg + cdef + abc + abe + abf + abg + \\
& acg + adf + adg + aef + aeg + bcf + bcg + bde + \\
& bdf + beg + bfg + cde + cdf + def + deg + dfg + \\
& ad + ae + bc + bd + cd + ce + df + dg + ef + fg + \\
& a + b + c + f + 1
\end{aligned}
$$

Fig. 2. The F function of ESSENCE, which takes seven words as input and operates in a bitsliced way (that is, the i-th bit of the output word only depends on the i-th bits of the input words).

Table 1. Differential path for finding collisions on (both versions of) ESSENCE; α and β are differences such that $\beta = L(\alpha)$ and $\alpha \vee \beta \vee L(\beta) = \alpha \vee \beta$. A "$\cdot$" denotes an absence of difference. Values in the column "Pr" are *heuristic approximations* of the probability to reach the *next* difference (exact probabilities significantly differ, and can be estimated empirically, cf. §§3.3).

Pr	Chaining value part									Message part								Pr
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	0	α	β	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\beta\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	1	β	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	2	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	3	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	4	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	5	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	6	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	7	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\rvert}$
1	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	8	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	1
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	9	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$2^{-\lvert\alpha\rvert}$
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	10	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	β	$2^{-\lvert\alpha\vee\beta\rvert}$
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	11	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	β	$\cdot$	$2^{-\lvert\alpha\vee\beta\rvert}$
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	12	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	β	$\cdot$	$\cdot$	$2^{-\lvert\alpha\vee\beta\rvert}$
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	13	$\cdot$	$\cdot$	$\cdot$	α	β	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\vee\beta\rvert}$
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	14	$\cdot$	$\cdot$	α	β	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\vee\beta\rvert}$
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	15	$\cdot$	α	β	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\vee\beta\rvert}$
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	16	α	β	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\beta\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	17	β	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	18	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	19	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	20	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	21	$\cdot$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	22	$\cdot$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	23	$\cdot$	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$2^{-\lvert\alpha\rvert}$
1	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	24	α	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	1
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	25	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	1
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	26	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	?	1
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	27	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	?	?	1
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	28	$\cdot$	$\cdot$	$\cdot$	$\cdot$	α	?	?	?	1
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	29	$\cdot$	$\cdot$	$\cdot$	α	?	?	?	?	1
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	30	$\cdot$	$\cdot$	α	?	?	?	?	?	1
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	31	$\cdot$	α	?	?	?	?	?	?	1
1	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	32	α	?	?	?	?	?	?	?	1

3 Collision Attacks on ESSENCE

Table 1 presents a differential path for finding collisions on the compression function of ESSENCE. It is used for both ESSENCE-256 and ESSENCE-512. We found this path manually, i.e., without the assistance of any automated search. Because it has no input difference in the chaining value, it can directly be used for searching colliding message blocks with respect to a same chaining value. The collision attack will then consist in

1. Finding one message block that fulfills the path on the right part.
2. Trying chaining values until one conforms to the path on the left part.

For the second phase of the attack, distinct pseudorandom chaining values are obtained by picking a first pseudorandom (sequence of) message block(s), and then checking differences after the insertion of the next message block.

The subsequent sections work out the details of the attack as follows:

- §§3.1 explains how the path works.
- §§3.2 presents an efficient method for finding a message block that conforms to the path.
- §§3.3 discusses calculation of the complexity; contrary to many similar differential attacks, an approximation solely based on Hamming weight is insufficient to obtain accurate probability estimates. Actually such heuristics *underestimate* the actual complexity of the attack, as we will see later.

Thereafter we use the following notations: $\vee$ for logical OR between two bits (or two words); $\wedge$ for logical AND; $\neg$ for bitwise negation; $|w|$ for the Hamming weight of word w; w_i for the i-th bit of word w, $0 \leq i < 32$ for ESSENCE-256, and $0 \leq i < 64$ for ESSENCE-512.

3.1 The Differential Path

The differential path on Table 1 starts with a difference in the message block, and no difference in the chaining value. To follow the path, the only assumption that we will make is that the function F will "absorb" certain differences (actually most of them) and "preserve" some others (at step 11). Therefore, the probability that a randomly chosen input conforms to the differential path will essentially depend on the Hamming weight of the wordwise differences α and $\beta = L(\alpha)$. Critical steps are listed below:

- **Step 0**: α is fed back to the rightmost cell of the right register via an XOR, and it does not enter F, unlike β. To ensure that no difference will appear in the output of F, we need all the $|\beta|$ bit differences be absorbed, which is expected to occur with probability $2^{-|\beta|}$ (such heuristic estimates should not be used systematically, as discussed later).

- **Step 1**: the relation $\beta = L(\alpha)$ makes differences introduced in the new rightmost cell vanish. This always works, but we also need that α adds no difference, that is, F needs to absorb $|\alpha|$ bit differences, thus the probability $2^{-|\alpha|}$ on both parts.
- **Steps 2 to 7**: we assume again that the $|\alpha|$ differences introduced in F are absorbed.
- **Step 8**: the two α differences cancel out in the middle of the mechanism, but α is also fed back to the rightmost cell of the message register.
- **Step 9**: unlike as in step 1, α will introduce a difference $L(\alpha) = \beta$, which will propagate during steps 11 to 17.
- **Step 10**: to avoid the introduction of new differences, we need the output of F to have differences $L(\beta)$, in order for the differences to vanish in the feedback operation. This will only be possible if $\alpha \vee \beta \vee L(\beta) = \alpha \vee \beta$. As we will see later, to avoid impossibilities of differential paths, we also have to add the condition $L(\beta) \wedge \alpha \wedge \neg\beta = 0$.
- **Steps 16 to 24**: the path is the same as in steps 0 to 8.
- **Steps 25 to 32**: note that differences in the right side after 32 steps do not affect the value returned by the compression function. We thus put no condition on those particular differences.

After finding this generic path, it remains to search for an α that minimizes the cost of the attack. But before that, we will present a generic method for finding a message block conforming to the right part of the path.

3.2 Efficient Search for a Conforming Block

Once we have found low-weight α and $\beta = L(\alpha)$ such that

$$\alpha \vee \beta \vee L(\beta) = \alpha \vee \beta \text{ and}$$
$$L(\beta) \wedge \alpha \wedge \neg\beta = 0 \ ,$$

the complexity of finding a conforming block by repeated trials is heuristically

$$2^{15|\alpha|+2|\beta|+6|\alpha\vee\beta|} \ .$$

This complexity is well above the birthday bound $2^{n/2}$ for all differences we found, let alone the fact that it underestimates the real complexity. For example, for the difference that we use to attack ESSENCE-256, the above expression yields a complexity 2^{210}, whereas a birthday attack needs only 2^{128} trials.

To find a conforming block at a reduced cost, we use a strategy similar in spirit to that of the rebound attack [5], namely, we start by finding conforming values for the low-probability path in the middle, then we check that they follow the simpler paths in both directions. What we call the *middle section* will correspond to *steps 8 to 17*, inclusive. More precisely, we will

1. Find many values that conform to the middle section (i.e., steps 8 to 17);

2. Search, among those values, one that conforms to the differential path in steps 0 to 8, and 17 to 24 (any such value will then follow the path up to step 32).

Note that we need to find approximately $2^{14|\alpha|+|\beta|}$ messages in the first phase, in order to have a conforming one with high probability in the second phase. Below we expose our strategy for efficiently finding many values conforming to the path between steps 8 and 17.

First we introduce some notations, to describe the state during the middle section: in Table 2 each x_j corresponds to a 32 or 64-bit word, depending on the version used. We write $\mathcal{S}$ the set of all indices where $\alpha \vee \beta$ is nonzero, that is,

$$\mathcal{S} = \{i, 0 \leq i < 32, \alpha_i \vee \beta_i = 1\} \quad \text{for ESSENCE-256,}$$
$$\mathcal{S} = \{i, 0 \leq i < 64, \alpha_i \vee \beta_i = 1\} \quad \text{for ESSENCE-512.}$$

We write $s = |\alpha \vee \beta| = |\mathcal{S}|$ the cardinality of $\mathcal{S}$. For example, if $\alpha =$ `80000000` and $\beta =$ `00000004`, then $\alpha_{31} = \beta_2 = 1$, and so $\mathcal{S} = \{2, 31\}$ and $s = 2$. We also write ℓ for the word bitlength (32 or 64, depending on the version of ESSENCE).

Table 2. Message part in steps 8-17

8	$x_0 \oplus \alpha$	x_1	x_2	x_3	x_4	x_5	x_6	x_7
9	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$x_8 \oplus \alpha$
10	x_2	x_3	x_4	x_5	x_6	x_7	$x_8 \oplus \alpha$	$x_9 \oplus \beta$
11	x_3	x_4	x_5	x_6	x_7	$x_8 \oplus \alpha$	$x_9 \oplus \beta$	x_{10}
12	x_4	x_5	x_6	x_7	$x_8 \oplus \alpha$	$x_9 \oplus \beta$	x_{10}	x_{11}
13	x_5	x_6	x_7	$x_8 \oplus \alpha$	$x_9 \oplus \beta$	x_{10}	x_{11}	x_{12}
14	x_6	x_7	$x_8 \oplus \alpha$	$x_9 \oplus \beta$	x_{10}	x_{11}	x_{12}	x_{13}
15	x_7	$x_8 \oplus \alpha$	$x_9 \oplus \beta$	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
16	$x_8 \oplus \alpha$	$x_9 \oplus \beta$	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
17	$x_9 \oplus \beta$	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	$x_{16} \oplus \alpha$

To search for values conforming to the middle section, we first look at an arbitrary slice i, and we count the number of possible tuples $(x_1, \ldots, x_{15})_i$ that fulfill the path between steps 8 and 17. Writing $\gamma = L(\beta)$, this corresponds to all tuples that satisfy the subsequent equations:

$$\begin{aligned}
F(x_1, x_2, x_3, x_4, x_5, x_6, x_7)_i &= F(x_1, x_2, x_3, x_4, x_5, x_6, x_7)_i \\
F(x_2, x_3, x_4, x_5, x_6, x_7, x_8)_i &= F(x_2, x_3, x_4, x_5, x_6, x_7, x_8 \oplus \alpha)_i \\
F(x_3, x_4, x_5, x_6, x_7, x_8, x_9)_i &= F(x_3, x_4, x_5, x_6, x_7, x_8 \oplus \alpha, x_9 \oplus \beta)_i \oplus \gamma_i \\
F(x_4, x_5, x_6, x_7, x_8, x_9, x_{10})_i &= F(x_4, x_5, x_6, x_7, x_8 \oplus \alpha, x_9 \oplus \beta, x_{10})_i \\
F(x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11})_i &= F(x_5, x_6, x_7, x_8 \oplus \alpha, x_9 \oplus \beta, x_{10}, x_{11})_i \\
F(x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12})_i &= F(x_6, x_7, x_8 \oplus \alpha, x_9 \oplus \beta, x_{10}, x_{11}, x_{12})_i \\
F(x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13})_i &= F(x_7, x_8 \oplus \alpha, x_9 \oplus \beta, x_{10}, x_{11}, x_{12}, x_{13})_i \\
F(x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14})_i &= F(x_8 \oplus \alpha, x_9 \oplus \beta, x_{10}, x_{11}, x_{12}, x_{13}, x_{14})_i \\
F(x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15})_i &= F(x_9 \oplus \beta, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15})_i
\end{aligned}$$

This property is only interesting for $i \in \mathcal{S}$, since for $i \notin \mathcal{S}$ there are no differences.

For slices such that $\gamma_i = 1$, we have to produce a difference in F in order to erase γ_i. Table 3 reports the number of solutions for the x_i's depending on $(\alpha_i, \beta_i, \gamma_i)$. The case $(1, 0, 1)$ is never going to be used, because it leads to an impossibility of following the differential path.

Table 3. Number of solutions for the $(x_1, \ldots, x_{15})$ depending on the input differences.

γ_i	(α_i, β_i)		
	$(0,1)$	$(1,0)$	$(1,1)$
0	96	96	96
1	128	120	176

Then, for each slice $i \in \mathcal{S}$ we fix one of these tuples and try to compute the missing bits. The number of possibilities to choose the tuples for $i \in \mathcal{S}$ is

$$N_\alpha = 96^{|\alpha \wedge \neg\beta \wedge \neg\gamma|} \times 96^{|\alpha \wedge \beta \wedge \neg\gamma|} \times 96^{|\neg\alpha \wedge \beta \wedge \neg\gamma|} \times 176^{|\alpha \wedge \beta \wedge \gamma|} \times 128^{|\neg\alpha \wedge \beta \wedge \gamma|} .$$

Note that to follow the path, the equations below (directly derived from the ESSENCE mechanism) must hold:

$$L(\overbrace{x_7}^{s \text{ bits fixed}}) = x_0 \oplus \overbrace{x_8 \oplus F(x_1, x_2, x_3, x_4, x_5, x_6, x_7)}^{s \text{ bits fixed}} \quad (1)$$

$$L(\overbrace{x_8}^{s \text{ bits fixed}}) = \overbrace{x_1 \oplus x_9 \oplus F(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \oplus \alpha)}^{s \text{ bits fixed}} \quad (2)$$

$$L(\overbrace{x_9}^{s \text{ bits fixed}}) = \overbrace{x_2 \oplus x_{10} \oplus \gamma \oplus F(x_3, x_4, x_5, x_6, x_7, x_8 \oplus \alpha, x_9 \oplus \beta)}^{s \text{ bits fixed}} \quad (3)$$

$$L(\overbrace{x_{10}}^{s \text{ bits fixed}}) = \overbrace{x_3 \oplus x_{11} \oplus F(x_4, x_5, x_6, x_7, x_8 \oplus \alpha, x_9 \oplus \beta, x_{10})}^{s \text{ bits fixed}} \quad (4)$$

$$L(\overbrace{x_{11}}^{s \text{ bits fixed}}) = \overbrace{x_4 \oplus x_{12} \oplus F(x_5, x_6, x_7, x_8 \oplus \alpha, x_9 \oplus \beta, x_{10}, x_{11})}^{s \text{ bits fixed}} \quad (5)$$

$$L(\overbrace{x_{12}}^{s \text{ bits fixed}}) = \overbrace{x_5 \oplus x_{13} \oplus F(x_6, x_7, x_8 \oplus \alpha, x_9 \oplus \beta, x_{10}, x_{11}, x_{12})}^{s \text{ bits fixed}} \quad (6)$$

$$L(\overbrace{x_{13}}^{s \text{ bits fixed}}) = \overbrace{x_6 \oplus x_{14} \oplus F(x_7, x_8 \oplus \alpha, x_9 \oplus \beta, x_{10}, x_{11}, x_{12}, x_{13})}^{s \text{ bits fixed}} \quad (7)$$

$$L(\overbrace{x_{14}}^{s \text{ bits fixed}}) = \overbrace{x_7 \oplus x_{15} \oplus F(x_8 \oplus \alpha, x_9 \oplus \beta, x_{10}, x_{11}, x_{12}, x_{13}, x_{14})}^{s \text{ bits fixed}} \quad (8)$$

$$L(\overbrace{x_{15}}^{s \text{ bits fixed}}) = x_{16} \oplus \overbrace{x_8 \oplus F(x_9 \oplus \beta, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15})}^{s \text{ bits fixed}} \quad (9)$$

The bits fixed in $x_1, \ldots, x_{15}$ are those in slice $i \in S$. Consider new intermediate variables R_8,R_9,...,R_{14} corresponding to the value of the right hand sides of Eq. (2)-(8). Each of these equations corresponds to a *linear* system

$$L(x_j) = R_j \ ,$$

for j in $\{8, \ldots, 14\}$. These are systems of ℓ equations between bits, wherein $2s$ variables are fixed and $2(\ell - s)$ variables are free. Due to the linearity, we can rewrite them as

$$L(x_{j,\bar{\mathcal{S}}}) + R_{j,\bar{\mathcal{S}}} = L(x_{j,\mathcal{S}}) + R_{j,\mathcal{S}} \ , \tag{10}$$

where $\bar{\mathcal{S}}$ is the complement of the set $\mathcal{S}$ and $x_{j,E}$ is the vector $(x_{j,i})$ with values 0 for i not in E. The position of the free variables depends only on $\mathcal{S}$. We can therefore perform a Gaussian elimination once for all on the left hand side of Equation (10).

We have more equations than free variables, so if the system is of maximal rank, we obtain $2s - \ell$ equations which must be satisfied by the fixed variables in order that their exists solutions. For our seven linear systems we have in total $7(2s - \ell)$ equations. Thus for any choice of $(x_1, \ldots, x_{15})$ fixed at $i \in \mathcal{S}$ we have a probability $2^{-7(2s-\ell)}$ of finding a valid solution for all the 7 systems.

Once we know that our choice correspond to a solution, we can compute efficiently the remaining bits of $x_j, R_j j \in \{8, \ldots, 14\}$ by the other $7(2\ell - 2s)$ equations of the Gaussian elimination.

To solve the systems of all seven equations, one thus proceeds as follows:

1. Fix the s bits in $x_1, \ldots, x_{15}$ to one of the N_α admissible values;
2. Try to solve the linear systems

   $$L(x_j) = R_j \ ,$$

   for j in $\{8, \ldots, 14\}$. If there is no solution, go back to step 1. Once we have a solution verifying the seven systems, we have all bits of x_j, R_j fixed for j in $\{8, \ldots, 14\}$. In $x_1, \ldots, x_7, x_{15}$ we only have the s bit fixed from the previous step, and in x_0, x_{16} no bits at all are fixed.
3. Freely choose the value of the $(\ell - s)$ remaining bits in x_7 since modifying those bits will not affect the previous steps.
4. We have now to consider the system

$$\begin{aligned}
R_8 &= x_1 \oplus x_9 \oplus F(x_2, x_3, x_4, x_5, x_6, x_7, x_8) && (11)\\
R_9 &= x_2 \oplus x_{10} \oplus F(x_3, x_4, x_5, x_6, x_7, x_8, x_9) && (12)\\
R_{10} &= x_3 \oplus x_{11} \oplus F(x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) && (13)\\
R_{11} &= x_4 \oplus x_{12} \oplus F(x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}) && (14)\\
R_{12} &= x_5 \oplus x_{13} \oplus F(x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}) && (15)\\
R_{13} &= x_6 \oplus x_{14} \oplus F(x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}) && (16)\\
R_{14} &= x_7 \oplus x_{15} \oplus F(x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}) && (17)
\end{aligned}$$

where R_j are entirely fixed. In the previous equations, we can skip α, β and γ since we chose admissible values for $(x_1, \ldots, x_{15})$. This system is almost in triangular form : Eq. (17) fixes x_{15}; Eq. (16) fixes x_6; Eq. (15) fixes x_5; Eq. (14) fixes x_4; Eq. (13) fixes x_3; Eq. (12) fixes x_2; Eq. (11) fixes x_1. Finally, Eq. (1) fixes x_0 and Eq. (9) fixes x_{16}.

Each valid solution in step 2 gives us $2^{\ell-s}$ results, by exploiting the extra degrees of freedom in step 3. We obtain in total about $N_\alpha \cdot 2^{7(\ell-2s)} \cdot 2^{\ell-s} \cdot 2^{-1}$ possible pairs that satisfy the path from step 8 to 17. The factor 2^{-1} comes from the fact that we counted each possible pair twice.

We can improve this general method in two ways.

First, we can do better than trying $2^{7(2s-\ell)}$ tuples to find a solution in step 2. This is based on a Gaussian elimination on the $2s-l$ equations allowing us to explore the set of all candidate tuples by a depth-first search procedure. For the sake of simplicity, we will explain the details later on the example of ESSENCE-256 in §§3.4. Without this method we would need $2^{7(2s-\ell)-s}$ trials to find one solution, which would increases the complexity a lot.

Secondly, we can improve the choice of $(x_1, \ldots, x_{15})_i, i \in \mathcal{S}$. This time we only consider a tuple $(x_1, \ldots, x_{15})_i$ admissible if it can be extended to a whole path from 0 to 24. This reduces the values in Table 3 to the following ones.

Table 4. Number of solutions for the $(x_1, \ldots, x_{15})$ depending on the input differences.

γ_i	(α_i, β_i)		
	$(0,1)$	$(1,0)$	$(1,1)$
0	96	2	4
1	128	0	2

Then, for each slice $i \in \mathcal{S}$ we fix one of these tuples and try to compute the missing bits. The number of possibilities to choose the tuples for $i \in \mathcal{S}$ is

$$\tilde{N}_\alpha = 2^{|\alpha \wedge \neg\beta \wedge \neg\gamma|} \times 4^{|\alpha \wedge \beta \wedge \neg\gamma|} \times 96^{|\neg\alpha \wedge \beta \wedge \neg\gamma|} \times 2^{|\alpha \wedge \beta \wedge \gamma|} \times 128^{|\neg\alpha \wedge \beta \wedge \gamma|} \ .$$

This method increases the probability of passing the rest of the path as we will see in §§3.3.

The choice of rounds $8 - 17$ for the middle part was done to get the lowest possible value for $\tilde{N}_\alpha$, see Appendix A.

The subsequent sections discuss the complexity of performing the search of the rest of the path, and give concrete complexity estimates for each instance of ESSENCE.

3.3 Finding Accurate Probabilities

Relying only on the Hamming weight to approximate the probability of the differential path gives unacceptably inaccurate approximations. Indeed, for a

given word slice, probabilities to be absorbed at each step are not independent, and neglecting this leads to estimates far from actual values. For example, a single bit difference is absorbed during seven steps with probability $2^{-8.41}$, which is significantly lower than the heuristic estimate 2^{-7}. However, for the paths considered, the dependency between word slices seems negligible. We thus give complexities with respect to empirical estimates, computed independently for each word slice. That is, we compute the probability of the differential path as 32 (or 64) independent differential paths, i.e., one for each slice. We could then estimate the real probability of our path for any given difference α. We found that having $\alpha_i = 1$, $\beta_i = 0$ and $L(\beta)_i = 1$ leads to an impossibility (the differential will never be satisfied for that α). This is why we need the condition

$$L(\beta) \wedge \alpha \wedge \neg\beta = 0 \ .$$

When considering the middle section, we also computed the real probability of verifying the sliced path once this part of the path is satisfied. The complexities given in the next section were computed with respect to those empirical estimates, not with the heuristic values based only on the Hamming weight.

Reusing the notation α, β, γ from §§3.2, we give below the probabilities for *a given slice* i to follow the complete path on 32 steps (the impossible cases—of probability zero—are not included), depending on $(\alpha_i, \beta_i, \gamma_i) \in \{0,1\}^3$:

- $(0,0,0) : 1$
- $(0,1,0) : 2^{-9.47}$
- $(0,1,1) : 2^{-9.05}$
- $(1,0,0) : 2^{-24.42}$
- $(1,1,0) : 2^{-23}$
- $(1,1,1) : 2^{-26}$

The probability that a random input follows the path is then the product of those probabilities, with each raised to a power that equals the number of slices corresponding to this case. For the α's used in our attacks and the exact values of the probabilities, we obtain probabilities $2^{-240.6}$ and $2^{-478.9}$, respectively for ESSENCE-256 and ESSENCE-512.

Taking into account our basic technique in §§3.2 for solving the middle at a reduced cost, we obtain the probabilities

- $(0,0,0) : 1$
- $(0,1,0) : 2^{-1.05}$
- $(0,1,1) : 2^{-1.05}$
- $(1,0,0) : 2^{-16}$
- $(1,1,0) : 2^{-14.59}$
- $(1,1,1) : 2^{-18.46}$

If we consider only those tuples $(x_1, \ldots, x_{15})$ where there is at least one possibility of verifying the whole path we get the following values:

- $(0,0,0) : 1$

- $(0,1,0) : 2^{-1.05}$
- $(0,1,1) : 2^{-1.05}$
- $(1,0,0) : 2^{-10.42}$
- $(1,1,0) : 2^{-10}$
- $(1,1,1) : 2^{-12}$

Given those numbers, we find that the probability that a value conforming to the middle section follows the rest of the path is $2^{-87.07}$ for ESSENCE-256 and $2^{-158.64}$ for ESSENCE-512 with the basic method and respectively $2^{-62.11}$ and $2^{-116.09}$ for the improved one.

There are at least two ways to compute the total number of message pairs that is going to satisfy the whole path. As we will get nearly the same result with both of them, we can verify that our estimations are correct. First, some additional notations are required: we let $\rho_0, \ldots, \rho_{\ell-1}$ denote the probabilities for each slice in $0, \ldots, \ell-1$ of conforming to the differential, i.e., each ρ_i will lie in $\{1, 2^{-9.47}, 2^{-9.05}, 2^{-24.42}, 2^{-23}, 2^{-26}\}$; and we let $\tau_0, \ldots, \tau_{\ell-1}$ be the conditional probabilities for each slice to follow the differential path, assuming that the middle section is satisfied. Now, the two equivalent ways to express the number of conforming messages are:

1. The probability of the whole path is $\prod_{i=0}^{\ell-1} \rho_i$, hence the number of pairs of conforming messages is
$$2^{8\ell} \cdot \prod_{i=0}^{\ell-1} \rho_i \ ,$$
where 8ℓ is the digest bit length.
2. The probability of the path once the middle section is satisfied is $\prod_{i=0}^{\ell-1} \tau_i$; calling N the number of pairs conforming to the middle section, the number of conforming message pairs is then
$$N \cdot \prod_{i=0}^{\ell-1} \tau_i \ .$$

In both cases we find each possible message pair twice. We verified that these two ways of computing the total number yield similar values (up to rounding approximations), which gives evidence of the accuracy of our estimates.

For example for our α: for ESSENCE-256 we find $2^{256} \cdot 2^{240.6} = 2^{15.4}$ message pairs considering the whole path . With our improved version we have $\tilde{N}_\alpha = 2^{106.51}$; a probability of 2^{-42} of solving the seven linear systems; 2^{13} times more solutions for free and thus $N = 2^{77.51}$. Using the probability $2^{-62.11}$ of passing the rest of the path we again get $2^{15.4}$ message pairs.

3.4 Collisions for ESSENCE-256

For ESSENCE-256, we could perform an exhaustive search over all differences and found as optimal value $\alpha =$ `80102040`, for which $|\alpha| = 4$, $|\beta| = 18$, and

$|\alpha \vee \beta| = s = 19$. Heuristic estimates suggest that we need about $2^{14\times4+18} = 2^{74}$ messages that conform to the middle section to find at least one conforming to the differential path on the right side. However, the real complexity is (cf. §§3.3) approximately $2^{87.07}$ for the basic method and $2^{62.11}$ for the improved one.

In the following, we will focus on the improved version.

Solving the right side. For that α, we have in total

$$96^6 \times 2^1 \times 128^9 \times 2^3 \approx 2^{106.51}$$

possibilities to set the bits in $\mathcal{S}$. We have a probability $2^{7(32-2\times19)} = 2^{-42}$ of finding a solution to the seven systems defined by Eq. (3) to (9). Following our assumption in §§3.2, we get about $2^{64.51}$ solutions. For each solution, we obtain 2^{13} additional solutions by varying bits in slices i not in $\mathcal{S}$, yielding in total up to $2^{77.51}$ solutions.

For each message pair found, we must check that it satisfies the rest of the path. As found in §§3.3, we need about $2^{62.11}$ values conforming to the middle section to find one value following the rest of the path. Below we detail the cost of finding those messages.

We look for solutions of systems (2) to (8). The linear systems $L(x_j) = R_j$ consist each of 32 equations and 26 free variables. We have thus 6 linear equations which the fixed bits must fulfill to guaranty that there exists a solution of the linear system.

We can choose those equations such that:

- choosing the values of the bit slices 0,1,2,3,5,6,9,10,12,16,18 fixes the parity of the first equation. This will not change with the choice of the remaining slices;
- if we chose in addition the values of the bit slices 7,11,13, we fix the second equation;
- if we chose in addition the values of the bit slices 4,17, we fix the third equation;
- if we chose in addition the values of the bit slices 14,15, we fix the fourth equation;
- finally, choosing the value for the last slice, the 8th one, fixes the parity of the remaining two equations.

This allows us to explore the set of candidate tuples efficiently by a depth-first search. Moreover, we can precompute the parity corresponding to the last 3 equations for any 3-tuple of choices for slices 8,14 and 15. That way, we do not even need to test the different tuples, but only to enumerate the ones giving us a valid solution. The cost to find a solution is therefore very low.

Using the degree of liberty coming from the step 3 of the solving procedure, our implementation is able to generate solutions for the middle part systems and test the rest of the path at a rate of approximately 650 cycles per candidate on an Intel Core2 processor, on comparison to about 1600 for hashing 256 bits[7].

[7] see eBASH: ECRYPT Benchmarking of All Submitted Hashes for SHA-3 `http://bench.cr.yp.to/ebash.html`

Solving the left side. Once a conforming pair of message blocks is found, we just need to try approximately $2^{67.4}$ distinct random chaining values to find a collision (for comparison, the heuristic estimate is $2^{14\times 4} = 2^{56}$). This value limits our attack. Since there is no α with a hamming weight of 3, which verifies the path on the right side, we cannot improve this value. Note that our attack can be carried out with negligible memory (the $2^{62.11}$ messages that satisfy the middle section don't have to be stored: we test repeatedly each candidate message, and discard it if it does not conform to the full path).

3.5 Collisions for ESSENCE-512

For ESSENCE-512, the best difference is $\alpha = \texttt{8408400000480082}$, giving $|\alpha| = 8$, $|\beta| = 35$, and $|\alpha \vee \beta| = s = 39$. As discussed in §§3.3, with the basic method we need about $2^{158.64}$ solutions of the middle section to find one solution for the right side of the path (against 2^{147} with heuristic estimates). With the improved method we only need $2^{116.09}$ solutions. In the following, we consider only the improved method.

Solving the right side. For our α we have

$$96^{14} \times 2^4 \times 4^3 \times 128^{17} \times 2^1 \times \approx 2^{222.19}$$

possibilities for the tuples at the indices $i \in \mathcal{S}$ and a probability of about 2^{-98} to find a solution for all the systems of Eq. (2)-(8). Thus, we expect about $2^{124.19}$ solutions. Using the free bits, we get for each solution $2^{64-39} = 2^{25}$ additional solutions. In total, there are thus about $2^{149.19}$ solutions, which will be good enough for finding one conforming to the full path (trying $2^{116.09}$ is sufficient).

Solving the left side. Now, we have a pair of messages that verify the differential path. The probability for a random chaining value of verifying the differential path is approximately $2^{-134.7}$. Once again, this value is the limiting part of our attack.

4 Attacking HMAC-ESSENCE

HMAC [1] is a widely used construction for building message authentication codes out of hash functions. Proposed in 1996 by Bellare, Canetti, and Krawczyk, HMAC has been standardized by NIST in 2002 [8] and requirements for SHA-3 include compatibility with HMAC.

The results in §3 can directly be turned into a distinguisher for ESSENCE-256 and ESSENCE-512 when used in keyed mode, be it with an unknown prefix message, or within HMAC. We just make the standard assumption that we can query an oracle (non-adaptively) with messages, and that this returns the digests produced by the keyed ESSENCE with this message as input, for a randomly preselected key.

A distinguisher then works as follows:

1. Find a pair of blocks (x, y) that conforms to the message part differential.
2. Repeat until a collision is found:
3. Pick a unique prefix m.
4. Query for oracle with $m\|x$ and $m\|y$.

Ideally 2^{256} trials are expected before a collision for ESSENCE-256, but here we'll make only $2^{67.4}$ trials in average, after a precomputation of complexity $2^{62.1}$. For ESSENCE-512, we have a complexity $2^{134.7}$ instead of 2^{512} ideally.

We can also mount an *existential forgery* attack by making one additional adaptive query:

1. Run the distinguisher above to obtain blocks m, x, y such that $m\|x$ and $m\|y$ collide by HMAC-ESSENCE.
2. Pick an arbitrary block m'.
3. Query the oracle for the MAC of $m\|x\|m'$, obtain a value z.
4. Return z as forgery of $m\|y\|m'$.

The complexity of this attack is essentially the same as that of the simple distinguisher.

5 Conclusion

We presented collision attacks on ESSENCE-256 and ESSENCE-512 of respective complexities $2^{67.4}$ and $2^{134.7}$. More precisely, these values are upper bounds on the cost of running our attacks, in terms of compression-equivalent units. Implementations of our attacks can use negligible memory, and in particular avoid expensive memory accesses. These attacks also apply to the versions of ESSENCE with 224- and 384-bit digests.

We showed a direct application of those collision attacks to the HMAC construction instantiated with ESSENCE, giving a distinguisher and an existential forgery attack with same complexity as the collision attacks.

Although far from practical, our attacks reveal significant weaknesses in the version of ESSENCE submitted to NIST. However, they do not highlight any flaw inherent to the ESSENCE design, but rather exploit the low number of rounds.

Acknowledgments

We would like to thank for their help: Anne Canteaut, Stéphane Jacob, Nicky Mouha, Gautham Sekar, and Fabien Viger.

A Choice of the Position of the Middle Part

We can see in Table 5 that the choice of rounds $8 - 17$ minimizes the number of solutions for the middle part $(x_1, \ldots, x_{15})$.

Table 5. Number of solutions for the $(x_1, \ldots, x_{15})$ depending on the input differences and on the rounds

rounds	$(\alpha_i, \beta_i, \gamma_i)$				
	$(1,0,0)$	$(0,1,0)$	$(1,1,0)$	$(0,1,1)$	$(1,1,1)$
0-9	12	3968	8	4960	4
1-10	12	1984	8	2480	4
2-11	8	3072	8	3840	4
3-12	8	2160	8	2640	4
4-13	4	1152	4	1408	4
5-14	4	576	4	704	4
6-15	4	288	8	352	4
7-16	4	192	8	224	4
8-17	2	96	4	128	2
9-18	4	96	8	128	4
10-19	4	96	12	128	4
11-20	4	176	12	208	4
12-21	4	352	12	384	4
13-22	4	512	12	640	4
14-23	4	1024	16	1280	4

B Empirical Results of the Right Side

C Practical Collision on a Reduced Version of ESSENCE-215

We can apply our method to find easily a collision on 23 out of 32 steps of ESSENCE-256 by using the path in Tab. 7. For example, the initial values in Tab. 8 leads to a collision for the α used in §3.4.

References

1. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *LNCS*, pages 1–15. Springer, 1996.
2. Christophe De Cannière and Christian Rechberger. Finding SHA-1 characteristics: General results and applications. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.
3. Jason Worth Martin. ESSENCE: A candidate hashing algorithm for the NIST competition. Submission to NIST, 2008.
4. Jason Worth Martin. ESSENCE: A family of cryptographic hashing algorithms. Submission to NIST, 2008.
5. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In Orr Dunkelman, editor, *FSE*, LNCS. Springer, 2009. to appear.

Table 6. Comparison between the *heuristic approximations* of the probability to reach the *next* difference and the *empirical values*.

	Message part								heuristic approximation	empirical value
0	α	β	·	·	·	·	·	·	2^{-18}	$2^{-19.34}$
1	β	·	·	·	·	·	·	α	2^{-4}	2^{-4}
2	·	·	·	·	·	·	α	·	2^{-4}	$2^{-1.66}$
3	·	·	·	·	·	α	·	·	2^{-4}	2^{-4}
4	·	·	·	·	α	·	·	·	2^{-4}	2^{-4}
5	·	·	·	α	·	·	·	·	2^{-4}	1
6	·	·	α	·	·	·	·	·	2^{-4}	2^{-4}
7	·	α	·	·	·	·	·	·	2^{-4}	1
8	α	·	·	·	·	·	·	·	—	—
9	·	·	·	·	·	·	·	α	—	—
10	·	·	·	·	·	·	α	β	—	—
11	·	·	·	·	·	α	β	·	—	—
12	·	·	·	·	α	β	·	·	—	—
13	·	·	·	α	β	·	·	·	—	—
14	·	·	α	β	·	·	·	·	—	—
15	·	α	β	·	·	·	·	·	—	—
16	α	β	·	·	·	·	·	·	—	—
17	β	·	·	·	·	·	·	α	2^{-4}	1
18	·	·	·	·	·	·	α	·	2^{-4}	2^{-3}
19	·	·	·	·	·	α	·	·	2^{-4}	2^{-1}
20	·	·	·	·	α	·	·	·	2^{-4}	2^{-4}
21	·	·	·	α	·	·	·	·	2^{-4}	2^{-3}
22	·	·	α	·	·	·	·	·	2^{-4}	2^{-8}
23	·	α	·	·	·	·	·	·	2^{-4}	2^{-4}
24	α	·	·	·	·	·	·	·	1	1
25	·	·	·	·	·	·	·	α	1	1
26	·	·	·	·	·	·	α	?	1	1
27	·	·	·	·	·	α	?	?	1	1
28	·	·	·	·	α	?	?	?	1	1
29	·	·	·	α	?	?	?	?	1	1
30	·	·	α	?	?	?	?	?	1	1
31	·	α	?	?	?	?	?	?	1	1
32	α	?	?	?	?	?	?	?	1	1
								total	2^{-74}	2^{-60}

Table 7. Differential path for finding collisions on 23 out of 32 steps of ESSENCE-256.

Pr	Chaining value part									Message part								Pr
1	·	·	·	·	·	·	·	·	0	·	·	·	·	·	·	·	α	$2^{-\lvert\alpha\rvert}$
1	·	·	·	·	·	·	·	·	1	·	·	·	·	·	·	α	β	$2^{-\lvert\alpha\vee\beta\rvert}$
1	·	·	·	·	·	·	·	·	2	·	·	·	·	·	α	β	·	$2^{-\lvert\alpha\vee\beta\rvert}$
1	·	·	·	·	·	·	·	·	3	·	·	·	·	α	β	·	·	$2^{-\lvert\alpha\vee\beta\rvert}$
1	·	·	·	·	·	·	·	·	4	·	·	·	α	β	·	·	·	$2^{-\lvert\alpha\vee\beta\rvert}$
1	·	·	·	·	·	·	·	·	5	·	·	α	β	·	·	·	·	$2^{-\lvert\alpha\vee\beta\rvert}$
1	·	·	·	·	·	·	·	·	6	·	α	β	·	·	·	·	·	$2^{-\lvert\alpha\vee\beta\rvert}$
1	·	·	·	·	·	·	·	·	7	α	β	·	·	·	·	·	·	$2^{-\lvert\beta\rvert}$
$2^{-\lvert\alpha\rvert}$	·	·	·	·	·	·	·	α	8	β	·	·	·	·	·	·	α	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	·	·	·	·	·	·	α	·	9	·	·	·	·	·	·	α	·	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	·	·	·	·	·	α	·	·	10	·	·	·	·	·	α	·	·	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	·	·	·	·	α	·	·	·	11	·	·	·	·	α	·	·	·	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	·	·	·	α	·	·	·	·	12	·	·	·	α	·	·	·	·	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	·	·	α	·	·	·	·	·	13	·	·	α	·	·	·	·	·	$2^{-\lvert\alpha\rvert}$
$2^{-\lvert\alpha\rvert}$	·	α	·	·	·	·	·	·	14	·	α	·	·	·	·	·	·	$2^{-\lvert\alpha\rvert}$
1	α	·	·	·	·	·	·	·	15	α	·	·	·	·	·	·	·	1
1	·	·	·	·	·	·	·	·	16	·	·	·	·	·	·	·	α	1
1	·	·	·	·	·	·	·	·	17	·	·	·	·	·	·	α	?	1
1	·	·	·	·	·	·	·	·	18	·	·	·	·	·	α	?	?	1
1	·	·	·	·	·	·	·	·	19	·	·	·	·	α	?	?	?	1
1	·	·	·	·	·	·	·	·	20	·	·	·	α	?	?	?	?	1
1	·	·	·	·	·	·	·	·	21	·	·	α	?	?	?	?	?	1
1	·	·	·	·	·	·	·	·	22	·	α	?	?	?	?	?	?	1
1	·	·	·	·	·	·	·	·	23	α	?	?	?	?	?	?	?	1

Table 8. Collision on 23 out of 32 steps of ESSENCE-256.

r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0
9483a50c	041e0ffb	662bea7a	28f65824	29e793c1	dab147cd	e55bece1	d3b17a2d
k_7	k_6	k_5	k_4	k_3	k_2	k_1	k_0
e3d6a0fc	20090ca8	5f83d19d	0c45d5a1	76c541bd	ce91a250	5feb86af	e2e811b1

Table 9. Exemple of a message passing 28 out of the 32 rounds of the differential path, for $\alpha = 80102040$ and $\beta = 537874eb$.

	initial values								
	15e59bfe	fe4a42de	509d2ffe	27ed0f9d	cdb864ed	02e2160c	b4b03a5d	014f1e87	
round	differences								round
0	80102040	537874eb	0	0	0	0	0	0	0
1	537874eb	0	0	0	0	0	0	80102040	1
2	0	0	0	0	0	0	80102040	0	2
3	0	0	0	0	0	80102040	0	0	3
4	0	0	0	0	80102040	0	0	0	4
5	0	0	0	80102040	0	0	0	0	5
6	0	0	80102040	0	0	0	0	0	6
7	0	80102040	0	0	0	0	0	0	7
8	80102040	0	0	0	0	0	0	0	8
9	0	0	0	0	0	0	0	80102040	9
10	0	0	0	0	0	0	80102040	537874eb	10
11	0	0	0	0	0	80102040	537874eb	0	11
12	0	0	0	0	80102040	537874eb	0	0	12
13	0	0	0	80102040	537874eb	0	0	0	13
14	0	0	80102040	537874eb	0	0	0	0	14
15	0	80102040	537874eb	0	0	0	0	0	15
16	80102040	537874eb	0	0	0	0	0	0	16
17	537874eb	0	0	0	0	0	0	80102040	17
18	0	0	0	0	0	0	80102040	0	18
19	0	0	0	0	0	80102040	0	0	19
20	0	0	0	0	80102040	0	0	0	20
21	0	0	0	80102040	0	0	0	80102040	21
22	0	0	80102040	0	0	0	80102040	d36874eb	22
23	0	80102040	0	0	0	80102040	d36874eb	cebd3034	23
24	80102040	0	0	0	80102040	d36874eb	cebd3034	1f79a8e7	24
25	0	0	0	80102040	d36874eb	cebd3034	1f79a8e7	9634a6b1	25
26	0	0	80102040	d36874eb	cebd3034	1f79a8e7	9634a6b1	cadf9d04	26
27	0	80102040	d36874eb	cebd3034	1f79a8e7	9634a6b1	cadf9d04	3ceaab4a	27
28	80102040	d36874eb	cebd3034	1f79a8e7	9634a6b1	cadf9d04	3ceaab4a	f14fca92	28
29	d36874eb	cebd3034	1f79a8e7	9634a6b1	cadf9d04	3ceaab4a	f14fca92	174117fe	29
30	cebd3034	1f79a8e7	9634a6b1	cadf9d04	3ceaab4a	f14fca92	174117fe	1cca654a	30
31	1f79a8e7	9634a6b1	cadf9d04	3ceaab4a	f14fca92	174117fe	1cca654a	2ead1557	31

6. Nicky Mouha, Gautham Sekar, Jean-Philippe Aumasson, Thomas Peyrin, Søren S. Thomsen, Meltem Sönmez Turan, and Bart Preneel. Cryptanalysis of the ESSENCE compression function. Available online at `http://www.nickymouha.be/papers/Essence-MouhaSekar.pdf`, 2009.
7. NIST. FIPS 180-2 – secure hash standard, 2002.
8. NIST. FIPS 198 – the keyed-hash message authentication code (HMAC), 2002.
9. Ronald L. Rivest. The MD6 hash function – a proposal to NIST for SHA-3. Submission to NIST, 2008.
10. Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *LNCS*, pages 1–22. Springer, 2007.
11. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
12. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.